
Isbar JavaTM

Iskaa u Baro Luuqadda Kumbuyuutarada ee Java

Hussein ISMAIL

December 19, 2015

Copyright © 2013 by Hussein Ismail
All rights reserved.

No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without the prior written permission of the writer.

ISBN: 978-952-93-2164-3
Web: www.isbarjava.com
Cover: Ras Kontti

Waxan u hibaynayaa xaaskayga oo ii dulqaadatay intaan la rafanaayey buuggan iyo dhammaan
walidkayga

Tusmada Buugga

Liiska Sawirrada	v
IsbarJava oo bilaash laga dhigay	ix
Mahad-naq	xi
Qoraaga buugga	xiii
Hordhac	xv

I TAARIIKHDA IYO DHISMAHA LUUQADDA JAVA 1

1 Dulmar Luuqadda Java	5
1.1 Waa maxay Java	5
1.1.1 Noocyada Java	6
1.2 Waa maxay Java Virtual Machine (JVM)	8
1.3 Waa maxay OOP	8
1.4 Waa maxay Objects	8
1.5 Waa maxay Classes	9
1.6 Waa maxay UML	11
1.7 Waa maxay Inheritance	11
1.8 Waa maxay TDD	12
1.9 Waa maxay JUnit	13
2 Diyaargarow	15
2.1 Qorista borogaraamyada Java	15
2.2 Borogaraamyada aad u baahantahay	16
2.2.1 Integrated Development Environment (IDE)	16
2.2.2 Text Editor, SDK iyo Apache Ant	21
2.3 Borogaraamka - Hello World!	24

II BARASHADA LUUQADDA JAVA 31

3 Qofka ama Shakhsiga	37
3.1 Macluumaadka Shakhsiga	37
3.2 Gunaanad	46
4 Ardayda	49
4.1 Gunaanad	55
5 Naxwaha Luuqadda Java	57
5.1 Ereyada u gaarka ah Java	57
5.2 Noocyada	57
5.3 Sifooyinka	58
5.4 Ficilada	60
5.4.1 Constructor	61
5.5 This	63
5.6 Sharaxista	64
5.7 Doorsoomeyaasha	65

5.7.1	Local variables	65
5.7.2	Parameters	66
5.8	Qiimaha null	69
5.9	Operators	69
5.9.1	Assignment operator	69
5.9.2	Arithmetic operators	70
5.9.3	Unary Operators	71
5.9.4	Equality and Relational Operators	72
5.9.5	Conditional Operators	73
5.10	Expressions, Statements iyo Blocks	74
5.11	Control Flow Statements	75
5.11.1	If statement	75
5.11.2	If-else statement	75
5.11.3	If-else if-else statement	75
5.11.4	Switch statement	76
5.11.5	While statement	78
5.11.6	For statement	79
5.11.7	Foreach statement	80
5.11.8	Break iyo Continue statements	80
5.12	Strings	80
5.13	Packages	82
5.14	Casting iyo Instanceof	83
6	Macalimiinta	85
6.1	Exceptions	88
6.1.1	Waa maxay exceptions	88
6.1.2	Waa maxay call stack	88
6.1.3	Tuurista walxaha exception-ka	89
6.1.4	Noocyada walxaha exception-ka	90
6.1.5	Qabashada walxaha exception-ka	92
6.1.6	Tijaabinta exceptions-ka	94
6.2	Gunaanad	98
7	Koorsooyinka	99
7.1	ToString	104
7.2	Object	105
7.3	Equals	106
7.4	Gunaanad	109
8	Fadhiyada Koorsooyinka	111
8.1	Java SE API Documentation	119
8.2	Interfaces	120
8.3	Collections	122
8.4	Modifier-ka final	123
8.5	Gunaanad	124
9	Model-ka Borogaraamka	125
9.1	Gunaanad	131
10	Java IO	133
10.1	Write to file - Kuqorista faylka	134
10.2	Read a file - Akhrista fayl	136
10.3	User Input- Waydiinta macluumaad isticmaalaha	138
III	DHISIDDA WAJIGA AMA ARAGGA (GUI)	141
11	Java Swing	145
11.1	Hello Swing	145
12	Qorshaha	147

13 Daaqadda Borogaraamka	149
13.1 Method overloading - rarista ficilka	152
13.2 Method overriding - burinta ficilka	152
13.3 Icon-ka daaqadda	152
13.4 Menu-da daaqadda	153
13.5 Ka bax ama Exit	155
14 Unugyada Swing	157
15 Dhagaysteyaasha dhacdooyinka	161
15.1 Nooc-hoosaadka iyo nooc-hoosaad aan magaca lahayn	163
16 Caloosha Daaqadda Borogaraamka	167
16.1 Ku xidhka Model-ka	167
16.2 Model-View-Controller (MVC)	168
16.3 Khafiifinta xidhiidhka noocyada	170
16.4 Adeegga u dhaxeeya Controller-ka iyo View-ga	171
16.4.1 Adeegga ay Controller-ku uga baahantahay View-ga	171
16.4.2 Adeegga uu View-gu uga baahanyahay Controller-ka	172
16.5 Bilawga Controller-ka	173
16.6 Bilawga View-ga iyo ku xidhka qaybaha kale	175
16.7 Aabaha caleemaha	177
16.8 JTable iyo isticmaalkiisa	178
16.9 Miisaska caleemaha	180
16.10Caleenta - Fadhiyada	181
16.11Caleenta - Koorsooyinka	193
16.12Caleenta - Ardayda	197
16.12.1 Arday cusub	206
16.12.2 Tuurista ardayga	208
16.12.3 Ardayda fadhiga	208
16.13Menu - Gal	210
16.13.1 Kaydi	210
16.13.2 U Kaydi...	214
16.13.3 Fur...	214
16.14Menu - Warbixin	215
16.14.1 Warbixin Dhan...	215
17 Sheeko gaaban: JavaFX	221
17.1 Waa maxay JavaFX	221
17.2 Dhisidda borogaraamyada JavaFX	222
17.2.1 Lasoo dagista NetBeans	222
Jahaynta mustaqbalka	229
Liiska Ereyada	231
Tixraac	239
Tilmaame	240

Liiska Sawirrada

1.1	Soosaarrada Java ee kala duwan iyo siday u kala duwanyihiin	7
1.2	Dirista fariinta (Qaabkani waa sida inta badan loo sawiro walxaha. Xariijinta magaca hoosta laga mariyey iyo labada dhibcod.)	9
1.3	Nooca Door (Door Class. Noocyada inta badan habkan ayaa loo sawiraa). Nooc kastaaba wuxu leeyahay magac, sifooyin iyo habab ama ficilo.	10
1.4	Takhasus (Inheritance. Habkani waa sida takhasuska inta badan loo sawiro). Falaadhu waxay ku tuseysaa in AlarmDoor laga soo dhiraandhiriyeey Door, sidaas awgeed uu yahay Door takhasusay.	11
1.5	Talaabooyinka Test-Driven Development (TDD)	12
1.6	Xidhiidhka ka dhexeeya JUnit, Java, Test Code iyo Code. Code waa qoraalka borogaraamka aad qorayso ee la intifaacsanaayo, Test Code waa qoraalka borogaraamka ti-jaabiyaha ah.	14
1.7	Nooca Door iyo nooca tijaabinaaya ee DoorTest	14
2.1	Talaabooyinka ay ka koobantahay qorista borogaraamyada luuqadda Java.	15
2.2	Eclipse SDK (Windows)	17
2.3	Lasoo dagista borogaraamka Eclipse talaabada 1-aad	18
2.4	Lasoo dagista borogaraamka Eclipse talaabada 2-aad	18
2.5	Shidista ama daarista borogaraamka Eclipse	18
2.6	Lasoo dagista borogaraamka Eclipse talaabada 4-aad	19
2.7	Lasoo dagista borogaraamka Eclipse talaabada 5-aad	19
2.8	Lasoo dagista borogaraamka Eclipse talaabada 6-aad	19
2.9	Lasoo dagista borogaraamka Eclipse talaabada 7-aad	20
2.10	Qaybaha kala duwan ee borogaraamka Eclipse ka koobanyahay. Halka ugu sareysa waa Menu. Waxad ka helayso talaabooyin badan sida File->New->Java Project, File->New->Class, iwm. Waxa kale oo aad Menu-ga ka helayso Run->Run As->1 Java Program, Run->Run As->2 JUnit Test, iwm. Menu af-soomaali Dr. Barwaaqo wuxu ku macneeeyey xashin [1], laakiin anigu waxan adeegsanayaa ereygaas af-ingiriisiga ah. Perspective-ku waa inuu kuu taagnaado Java inta badan. Toolbar-ka waxad ka helayso waxyaalo badan oo aad ka helayso Menu-ga. Qaybta bidixda ee Project Explorer waxad ka helayso faylashaada (galalka). Qaybta dhexe waa meeshii wax lagu qoraayey.	20
2.11	Borogaraamka Eclipse marka lagu isticmaalaayo Linux	21
2.12	Ku kaydinta borogaraamka JRE kumbuyuutarkaaga	22
2.13	Windows Notepad	22
2.14	Borogaraamka Notepad++	23
2.15	Java Development Kit (JDK). Sawirka waxad ka arkeysaa waxay ku kala duwanyihiin JRE iyo JDK. Arinta ugu muhiimsan ee aad sawirka ka arkayso waa in JDK ay ka koobantahay JRE iyo agab kale.	24
2.16	Command line-ka Windows	25
2.17	Environment variables-ka Windows iyo sida wax looga badali karo.	26
2.18	Samaynta Project cusub oo Java ah. Meesha Project name waa qasab inaad buuxiyo, meelaha kale loomabaahna inaad badasho.	27
2.19	Project cusub oo Java ah samaynta kadib.	28
2.20	Samaynta nooc cusub oo Java ah. Inta badan meelaha aad buuxinayso waa Name iyo Package. Meelaha kale sida Source folder ama meelaha kale waxa loo baahanyahay inaad badasho marmar iyo dhif.	29
2.21	Natiijada borogaraamka Hello World marka la kexeeyo, waxad ka arkaysaa daaqadda Console ee borogaraamka Eclipse. Daaqaddaas waxad ka arkaysaa inta badan wixii fariimo ah ee borogaraamkaagu tusaayo isticmaalaha.	30

2.22	Samaynta Project-ka nimi kadib. Haddii aanad tuurin Project-kii hore ee helloworld , Eclipse markaad furto waa inuu u ekaado sidan. Halkan laga bilaabo, waxaynu isticmaalaynaa project-ka nimi, markaa waa inaad isaga doorato.	34
2.23	Folder-ka src waxa lagu kaydinayaa qoraalka borogaraamka (source code), folder-ka test waxa lagu kaydinayaa qoraalka borogaraamka tijaabiyaha ah (test code) sidaad arki doonto	35
3.1	Nooca Person iyo tijaabiyihiisa PersonTest	38
3.2	Bilaabista nooc cusub oo tijaabiye ah	39
3.3	JUnit ayaad u baahantahay si aad u qorto qoraalka tijaabiyaha ah. Sidaad sawirka ka arkayso JUnit 4 ayaynu adeegsanaynaa. JUnit 4 wuxu la socdaa Eclipse , sidaas awgeed looma baahna inaad gooni ulasoo dagto.	40
3.4	Ficilka assertEquals wuxu isu eegayaa qiimaha bidixda ee Expected (qiimaha la sugaayey) iyo kan midigta ee Actual (qiimaha soo baxay). Halkan waxay isu eegaysaa qiimaha "Jaamac Cilmi" iyo wixii uu ficilka getName() ee walaxda person1 soo celiyo.	40
3.5	Tijaabadii ugu horeysay oo fail-garowday ama guuldaraysatay. Sawirka xaga sare, meelaha aan calaamadiyey waxad ka arkaysaa tirakoob, inta tijaabo ee la kaxeeyey, inta cilad daraadeed u guuldaraysatay iyo inta guuldaraysatay (sababtoo ah borogaraamka ayaan u shaqayn sidii laga sugaayey)	41
3.6	Tijaabadii ugu horeysa oo pass-garowday	42
3.7	Saddexda tijaabo ee testCreatePerson, testPersonTelephone iyo testChangePersonInformation ee nooca PersonTest oo guulaysanaaya ama ansaxaaya (passing)	47
4.1	Nooca Student iyo Tijaabihiiisa StudentTest. Student waa qof, sidaas awgeed wuxu isticmaalayaa takhasuska oo waxa laga soo dhiraandhiriyeey nooca Person.	49
4.2	Natijada kuusoo baxaysa markaad kaxayso nooca AllTests.	54
5.1	Tusaalaha 5.7 oo sawir ahaan loo muujiyey iyo ereyga this.	63
5.2	Kaladuwanaanshaha doorsoomeyaasha PDT-ga ah iyo kuwa kale	67
5.3	Kaladuwanaanshaha doorsoomeyaasha PDT-ga ah iyo kuwa kale markay timaado ficilada	68
5.4	Borogaraamka 5.23 oo la kaxeeyey	78
5.5	Qoraalka xaraf waliba wuxu leeyahay boos ama position. Ficilada indexOf iyo lastIndexOf ayaa ku siinaaya booska uu xarafkaasi ku jiro. Haddii aad garanayso booskii oo aad raadinayso xarafka booskaas ku jira, waxad isticmaalaysaa ficilka charAt. Tusaale ahaan qoraalka sare meesha ugu dambaysa ee uu xarafka 'e' kaga jiro waa booska 6-aad, sida uu inoo sheegay borogaraamka 5.31. Xafarka ugu horeeya booskiisu waa eber.	82
6.1	Nooca Teacher iyo tijaabihiiisa TeacherTest. Teacher waa qof, sidaas awgeed wuxu isticmaalayaa takhasuska Person.	86
6.2	Haddii borogaraamka midigtu maraayo ficilka c() ee nooca Test, call stack-kiisu wuxu u eekaanayaa sawirka bidixda. Falaadhaha hoos u jeeda waxay sheegayaan ficilba ficilka uu sii wacay. Falaadhaha kor u jeeda ee googo'an waxay sheegayaan raadinta exception handler-ka, markay ciladi timaado. Raadintu waxay ka bilaabmaysaa ficilka la maraayo laftiisa, kadibna waxa dib loo raacayaa call stack-ka. Haddii ficilka main laga waayo exception handler ku haboon exception object-ka lasoo tuuray, borogaraamku wuu istaagayaa ciladdaas awgeed, maadaama ay tahay meeshii ugu dambaysay ee laga raadinaayey exception handler-ka.	89
6.3	Noocyada exception-ka iyo qaabka ay isugu xidhanyihiin. Nooca Error waxa loogu tala-galay JVM-ka oo looma baahna inaynu borogaraamyadeena ku isticmaalo, waxaanu ka turjumayaa ciladaha aan laga soo kaban karin. Noocyadaada waxad kasoo dhiraandhirinaysaa nooca Exception, sida nooca QiimeQaladAhException oo kale. Nooca RuntimeException ee laga soo dhiraandhiriyeey nooca Exception, isna wuxu leeyahay macno gaar ah, laakiin isna looma baahna inaynu buuggan kaga hadalo, waayo waa laga maarmi karaa, kaliya waxa kugu filan inaad ogaato inuu jiro.	91
6.4	Call Stack	94
6.5	Ficilka testSalary ee nooca TeacherTest siduu u soconaayo marka la kaxeeyo sida caadiga ah	95
6.6	Ficilada testSalary iyo setSalary iyo saameynta exceptions-ku ku leeyihiin kaxayntooda. Waxad arkaysaa in labada ficilba wax iska badalayaan.	96
7.1	Nooca Course iyo Tijaabihiiisa CourseTest.	100
7.2	Nooca TestCourse oo ka kooban laba tijaabo testCreateCourse iyo testCourseNumber oo la kaxeeyey	103

7.3	Noocyada Java kulligood (kuwaad adigu qorto iyo kuwa diyaarka ahba) waxa laga soo dhiiraandhiyey nooca Object, kaas oo ay ka dhaxlayaan ficillo dhawr ah oo uu ka midyahay ficilka toString	106
7.4	Nooca AllTests oo lagu kordhiyey nooca CourseTest. Dhammaan tijaabooyinka aynu illaa hadda haysano waa 11 sida ka muuqata sawirkan.	108
8.1	Nooca CourseSession iyo noocyada kale ee uu xidhiidhka la leeyahay	112
8.2	Dhammaan tijaabooyinka aynu illaa hadda haysano oo dhammaantood cagaar ah	120
8.3	Java™ Platform, Standard Edition 6 API Specification, oo aad ka helayso sharaxaad ku saabsan dhammaan wixii noocyo ah ee ku jira	121
8.4	Nooca Person oo implement-garaynaaya interface Named	122
8.5	Interface-yada collections-ka ee Java. Waxad sawirkan ka arkaysaa in noocyada ArrayList iyo LinkedList, labadooduba yihiin List, maadaama ay implement-garaynayaan interface-kaas.	123
9.1	Model-ka borogaraamka nimi	126
10.1	Stream-ku waa sida tuubo oo kale, tuubadaas oo isku xidhaysa borogaraamkaaga iyo qalabka kala duwan sida: Internet-ka, shaashadda, keyboard-ka, jiirka, iwm.	134
10.2	Borogaraamka 10.1 natiijadiisa. Si aad u aragto faylka uu borogaraamku sameeyey, dooro marka koobaad dooro project-ka kadibna refresh-garee (badhanka f5 guji), kadib waa inaad aragto faylka cusub. Intaas kadib laba jeer guji faylkaas cusub si aad u furto.	135
10.3	Borogaraamka 10.3 natiijadiisa. Wixii faylka ku qornaa waxad ka arkaysaa caleenta console-ka	137
10.4	Borogaraamka sare oo laga isticmaalay command line-ka. Marka hore waa inaad tagto folder-ka <i>bin</i> oo ku hoos jirta meesha uu project-ka nimi kumbuyuutarkaaga kaga kayd-sanyahay.	139
11.1	Borogaraamka 11.1 caynka uu eegyahay marka lagu isticmaalo OS-yo kala duwan	146
12.1	Qaabka aynu isleenahay u eekaysiia wajiga borogaraamka	147
13.1	Daaqadda ama Frame-ka sare ee nimi	150
13.2	Menu-da borogaraamka iyo sharaxaad ku saabsan qaybaha kala duwan ee ay ka koobantahay. Waxan isticmaalidoonaa magacyada menubar (JMenuBar), menu (JMenu) iyo menuitem (JMenuItem). Waxa aan sawirka lagu calaamadin JSeparator oo ah xariijinta tusaale ahaan u dhaxaysa "Fur..." iyo "Kaydi".	154
15.1	Dhagaysteyaasha dhacdooyinka. Event source waa ishii dhacdada, event listener waa dhagaystihii, event object waa dhacdadii ama fariintii.	161
15.2	Mar kasta oo aad gujiso badhanka waxa kordhaaya tirada jeer ee la gujiyey (Sawirkan waa Windows 8)	162
15.3	Labada borogaraam ee 15.1 iyo 15.2 waxay ku kala duwanyiin marka la eego dhanka walxaha. Falaadha goo'go'an waxay ku tusaysaa in walaxda IsbarJava caloosha ku haysato labada walxood ee JLabal iyo JButton. Xaaladda koobaad walaxda IsbarJava kama qayb qaadanayso dhagaysiga dhacdooyinka JButton, laakiin xaaladda labaad waa iyada walaxda dhageysanaysa badhanka, marka la gujiyo kadibna badalaysa qoraalka JLabel. Walax kasta nooceeda waan ku hoos qoray, laakiin xaaladda koobaad walaxda dhagay-senaysa badhanka, marka la gujiyo kadibna badalaysa qoraalka nooceeda waxan saaray su'aal, maadaama aan la garanaynin magaca nooceeda. Waxan ku sharaxayaa arintan hoos.	163
16.1	Sawir ku tusaaya qaybaha ay ka koobantahay MVC iyo sida ay isula xidhiidhayaan. Dhaqdhaqaaqa isticmaalaha sida gujinta badhanada waxa marka hore qaabilaaya View-ga. View-gu wuxu u gudbinayaa Controller-ka, kadibna Controller-ku waxay amraysaa oo badalaysaa Model-ka. View-gu wuxu iska diiwaangelinayaa Model-ka. Model-ku markasta oo wax iska badalaan waxay wargalinaysaa dhammaan wixii View iska diiwaangelisay. Model-ka waxa iska diiwaangelinkara dhagaysteyaal kale oo aan ahayn View.	169
16.2	Naqshadan oo aynu buuggan ku adeegsandoono kaliya waxa la masaxay xidhiidhkii u dhaxeeyey View-ga iyo Model-ka. Model-ku imika marka ay is badasho waxay wargalinaysaa tusaale ahaan Controller-ka, haddii ay iska diiwaangeliso. Laakiin wargalinta buuggan kusoo qaadanmayno. Naqshadan waxa loo yaqaan <i>Passive View</i>	170
16.3	Isticmaalka Interfaces-ka si loogu khafiifiyo xidhiidhka ka dhexeeya laba nooc	171

16.4	Dhammaan caleemaha waxa laga soo dhiraandhiriyeey nooca NimiTab	177
16.5	Swing JTable	179
16.6	Tusaale Swing JTable	180
16.7	Nooca AbstractTableModel wuxu ka midyahay Java, waxaanu implement-garaynayaa interface-ka TableModel. Laakiin maadaama uu yahay abstract, lama isticmaali karo, waxa loo baahantahay in laga soo dhiraandhiriyo nooc aad leedahay.	181
16.8	Walaxda JTable, Walaxda model-ka u ah iyo data-da xidhiidhka ka dhexeeya	182
16.9	Muuqaalka borogaraamka oo caleenta fadhiyada oo lagu kordhiyey	183
16.10	Caleenta fadhiyada oo lagu kordhiyey badhanada kantaroolka ee hoose, kuwaas oo aan wali waxba qabanaynin marka la gujiyo	185
16.11	Caleenta fadhiyada oo lagu kordhiyey miiska sare	189
16.12	Caleenta koorsooyinka	197
16.13	Caleenta ardayda	203
16.14	Kudar arday fadhiga	209
16.15	Daaqadda soo boodaysa markaad gujiso Kaydi	214
16.16	Library-ga Freemarker ku soo dar borogaraamka, sidaad u isticmaasho	216
16.17	Sawir ka turjumaaya qaabka ay Freemarker u shaqaynayso	220
16.18	Warbixin dhamaystiran oo borogaraamka nimi soo tufay marka la gujiyo Warbixin->Warbixin Dhan...	220
17.1	Qaabka ay u dhisantahay iyo waxyaalaha ay ka koobantahay ama isticmaalayso JavaFX	222
17.2	Lasoo dagista borogaraamka Netbeans	223
17.3	Borogaraamka Netbeans oo lagu isticmaalaayo Windows 8, gaar ahaan markii ugu ho-raysay ee la shiday	223
17.4	Daaqadda samaynta project-ka cusub ee NetBeans	224
17.5	Macluumaadka project-ka cusub	224
17.6	Qaababka kala duwan ee aad isticmaali karto borogaraamka JavaFX	226
17.7	Borogaraamka marka loo isticmaalo qaabka Desktop Application	226
17.8	Borogaraamka JavaFX marka lagu kaxeeyo Browser-ka caloosheeda	227

IsbarJava oo bilaash laga dhigay

Sanadka 2015 dhammaadkiisa (December 2015) waxan go'aansaday inaan buuggan bilaash ka dhigo oo Internet-ka ku faafiyo, si ciddii rabta uu u gaadho meel kasta oo ay aduunka kaga nooshahay. Go'aankan waxa igu kalifay laba arimood oo kala ah sidan:

1. In cilmigan teknooloojiyaddu yahay mid isbadal badan oo mar walba waxyaalo cusub kusoo biirayaan, qaar horena meesha ka baxayaan. Taas macnaheedu ma aha in waxyaalaha buuggan ku qorani ay gaabooben. Waxan qabaa in waxyaalaha buuggan ku qorani jiridoonaan inta luuqada kumbuyuutarka sida Java ay jiraan. Kaliya waa xaqiiqada guud ee ka jirta aduunka teknooloojiyadda.
2. In aanan dadka buuggan u baahan dhamaantood ku gaarsiin karin qaabka buug la daabacay. Taasna arinta ugu wayn ee iga hortaagan waa dhaqaale.

Waxan buuggan qoristiisa bilaabay 29-kii Abriil sanadkii 2010, waxaanan dhameeyey qoristiisa oo bilaabay daabacaaddii 15-kii Diseembar 2013. Halkaas waxa kaagasoo baxaysa in aan waday qoristiisa qiyaastii 3 sanadood iyo badh. Halkaas waxad ka arkaysaa inaan wakhti badan siiyay oo aan u huray inaan ku guulaysto dhammaystirkiisa. Sidaas awgeed haddii buuggani ku anfacu oo aad jeclaysato inaad waxyar sida \$2, \$5, \$10 ama wax kale aad ku caawiso qoraaga waxad iigusoo diri kartaa siyaabo kala duwan. Haddii aad joogto Somalia, Puntland ama Somaliland sida ugu fudud waa inaad ku tuurto telefoonka +252 63 4767170 (XUSEEN MAXAMED), haddii aad qurbaha joogtana waxad isticmaali kartaa account-ka: IBAN: FI9557316520077852, BIC: OKOYFIHH ama waxad adeegsan kartaa xawaaladaha. Intaas kadib waxan rajaynayaa in buuggani idin anfacu. Sidaas iyo akhris wanaagsan. Fadlan gaarsii ciddii aad is leedahay way u baahantahay buuggan.

Mahad-naq

Waxan u mahadcelinayaa dhammaan dadkii gacan igu siisay inuu buuggani daabacmo oo uu soo baxo. Waxan si gaar ah ugu mahad naqayaa Abdulrashid Areh (Fanax), Jamaal Ali iyo Nasir Jamac, oo dhammaantood iga caawiyey dhanka sixitaanka buugga. Waxan si gaar ah ugu mahad naqayaa Nasir Jamac oo runtii wakhti badan ii huray si uu buuggani usoo baxo.

Qoraaga buugga

Magacayga waxa la yidhaahdaa Hussein Ismail (Xuseen Ismaaciil). Waxan ku dhashay wadanka Soomaaliya dhammaadkii sanadkii 1980-kii. Waxan imi wadanka Finland sanadkii 1992-kii, markaas oo qiyaastii da'daydu ahayd 11-jir.

Waxan waxbarashadayda ku qaatay illaa jaamacadda laga gaadho wadanka Finland, marka laga reebo koorsooyin gaaban oo aan ku qaatay wadanka Ingiriiska. Waxan sanadkii 2007-dii ka qalinjabiyey jaamacad ku taalla magaalada Tampere¹ ee koonfurta wadanka Finland, oo magaceeda la yidhaahdo Tampere University of Technology (TUT)², halkaas oo aan ka qaatay shahaadada loo yaqaano Master of Science³ ama Master oo kaliya.

Waxan shaqo ka bilaabay jaamacadda kadib shirkadda telefoonada moobilada sameysa ee Nokia Corporation⁴, taas oo aan wali u shaqeeyo xilligan qorista buugga. Waxaan ku shaqeeyaa cinwaanka *Senior Software Designer*, oo ah magaca loogu yeedho dadka sameeya borogaraamyada kumbuyuutarada mid yar iyo mid waynba, gaar ahaan dadka muddo shaqadaas ka shaqaynaayey. Mobilada qudhoodu waa nooc ka mid ah kumbuyuutarada, maadaama ay leeyihiin sifooyinkii kumbuyuutarada badankood: sida xasuusta dhexe (*main memory*), fuliyaha (*processor*), shaashadda (*screen*), iwm.

¹Bogga Internet-ka ee Tampere: <http://www.tampere.fi/>

²Bogga Internet-ka ee Jaamacadda: <http://www.tut.fi>

³Waxa inta badan loo soo gaabiyaa M.Sc. ama M.S.

⁴Bogga hore ee Internet-ka ee Nokia: <http://www.nokia.com/>

Hordhac

Buuggani wuxu ku saabsanyahay ama ka hadlayaa luuqadda kumbuyuutarka (ama kumbuyuutarada) ee Java. Luuqadda **Java** iyo luuqado kale oo dhawr ah waxay aasaas u yihiin shayga loo yaqaanno kumbuyuutarka, oo roog gundhig u ah hormarka ay gaadheen wadamada loo yaqaano *Developed Countries*. Hormarkaas oo inta badan lagu qiyaaso dhanka dhaqaalaha, iyada oo la adeegsanaayo cabirka *Gross Domestic Product (GDP)* calaa qof, oo ka dhigan isugeynta qiimaha alaabta iyo adeegyada uu wadankaasi soo saaro sanadkiiba oo loo qaybiyey tirada dadka ku nool wadankaas, si loo helo qiimaha qofkiiba uu helaayo. Marka la eego liiska ay soo saartay qaramada midoobay (QM) noofembar 4, 2010, shanta wadan ee aduunka ugu sareeya waxa weeye sidan: Norway, Australia, New Zealand, United States iyo Ireland.

Waxa lagama maarmaan ah in la barto lana adeegsado kumbuyuutarka, haddii bulshadu (ama shakhsigu) rabto iney hormar gaadho sida wadamaas aan soo sheegay oo kale. Waa inay bulshadu barato sida loo sameeyo borogaraamyada kumbuyuutarka, borogaraamyadaas oo danaheeda gaarka ah u fulinaaya. Haddii kale waxay noqonaysaa inay bulshadaasi dibedda ka keensato dad u sameeya borogaraamyada, taas oo runtii lacag badan ku kici karta. Arintani waa mid laga sugaayo wadamada soo koraaya, gaar ahaan wadamada Afrika.

Waxan soo qaadanayaa tusaalooyin kooban, oo ka mid ah waxyaalaha lagama maarmaanka ah, ee inta badan loo adeegsado kumbuyuutarada baryahan dambe. Waxan ku bilaabayaa **World Wide Web (WWW)** oo inta badan dadku ku qaldaan Internet-ka. WWW waa system daadsan oo ku shaqeeya Internetka dushiisa, kana mid ah adeegyada Internet-ka [2]. Halkaas waxad ka arkeysaa inaanu adeega WWW jiri karin Internet iyo kumbuyuutar la'aantiis. WWW wuxu ku caanbaxay bogagga Internet-ka, sida <http://www.google.com/>, <http://www.yahoo.com/>, <http://www.hotmail.com/> iyo qaar kale oo badan. Markaad adaraysyada bogagga Internet-ka u fiirsato, waxa dhammaantood ku jira WWW iyo **Hypertext Transfer Protocol (HTTP)** ama **HTTPS** oo uu WWW ku caanbaxay. Waxyaalaha kale ee kumbuyuutarka aad loogu adeegsado waxa ka mid ah *Email* (tusaale ahaan Hotmail, Gmail, Yahoo Mail, Outlook, iwm), **Word Processing** (Word, Excel, PowerPoint, Access, Text Editors, iwm), *Games* oo ah ciyaaraha lagu ciyaaro kumbuyuutarka, dhagaysiga muusigga, daawashada fiidyawiyada, samaynta borogaraamyada, xidhiidh dadka iyo waxyaale kale oo badan oo aan halkan lagusoo koobi karin. Waxa jira borogaraamyo badan oo shirkaduhu gaar u leeyihiin oo iyana kumbuyuutar la'aantiis aan shaqaynaynin.

Buuggan waxa loogu talagalay qof kasta oo raba inuu ogaado meesha iyo habka ay ku timid luuqadda kumbuyuutarka ee Java. Waliba gaar ahaan qofka raba inuu bilaabo inuu barto luuqadda Java, kuna barto luuqaddiisa hooyo. Waxaynu odhan karnaa dadka ugu muhiimsan ee buuggan loogu talagalay waa ardayda iskuulada iyo dadka shakhsi ahaan raba iney bartaan luuqaddan. Dadka aan is leeyahay buuggan way ka faa'iidayaan gaar ahaan, waa kuwa bilaw ku ah qorista borogaraamyada kumbuyuutarka. Waxan aaminsanay iney Java tahay luuqad fiican oo lagu bilaabi karo barashada dhisidda borogaraamyada kumbuyuutarka. Buuggani wuxu ka mid noqonayaa kuwii ugu horeeyey, haddii aanu ahayn kii ugu horeeyey oo mawduucan laga qoro, kaas oo ah af-soomaali ah. Waxa muhiim ah inuu akhristuhu ogaado inaan mawduucan af-soomaali lagaga dhargikarin, laakiin wuxu noqonayaa buuggani fure lagu bilaabo mawduucan.

Dadka dhisa ama qora borogaraamyada kumbuyuutarka waxa loo yaqaan magacyo kala duwan, sida: *developer*, *programmer*, *coder*, *software engineer*, *software designer* iyo *software analyst*. Dhisidda ama qorista borogaraamyada kumbuyuutarku waa qayb muhiim ah oo ka mid ah xirfadaha dadkaas.

Waxan ogahay oo kollay dad badani ila qabaan in waxbarashadu u badantahay iskaa u baro. Shaqada macalinku waa inuu jidka ku tuso, laakiin waa hawl ardayga u taal inuu jidkaas ka gun gaadho. Sidaas daraadeed waxan odhan karaa buuggani wuxu ku tusayaa jidka oo kaliya. Waxan ku dadaalayaa qore ahaan inaan jidkaas akhristaha si fiican u tuso, oo tilmaan fiican siiyo.

Haddii aan waxyar ka sheego dhismaha buuggan. Buuggani wuxu ka koobanyahay saddex qaybood. Qaybta hore waxad ku barandoontaa taariikhda iyo hab dhismeedka luuqadda Java. Waxaanay ku siindoontaa sawir guud oo ku saabsan luuqadda Java. Waxay ku ficantahay akhristaha raba inuu sawir guud ka helo luuqadda Java, si uu tusaale ahaan go'aan uga gaadho inuu sii barto iyo in kale. Haddii uu akhristuhu hubo inuu rabo inuu barto luuqadda Java, sawir guudna ka haysto wuu ka boodi karaa qaybtaas. Qaybta labaad waxad ku barandoontaa aasaaska luuqadda Java adiga oo guriga hoosta ama

aasaaska kasoo dhisaaya. Kadib markaad guriga udubyo iyo dhigo adag u sameyso, waxad u gudbi doontaa qaybta saddexaad oo aad guriga waji ama sumcad lagu garto u samayndoonto. Guriga waxan ula jeedaa borogaraamka aynu dhisaynno. Safar wanaagsan!

Afeef

Waxan afeef (justification) ka dhiganayaa afar arimood oo ay ka mid yihiin sababta buuggan qoristiisa igu kalliftay:

1. Arinta ugu muhiimsan waxa weeyaan inaan dadka soomaalida ah ee mawduucan xiisaysa uga shee-keeyo shakhsi ahaan wixii aan ka bartay ama kala kulmay mudaddii aan ku dhexjiray. Kumbuyu-utarada waxan ku dhexjiray muddo qiyaastii 12 sanadood ah. Waxan buuggan kusoo bandhigayaa wixii aan luuqadda Java ka bartay mudadaas. Si qofka af-soomaaliga ku hadla ee raba inuu mawdu-ucan barto uu u helo meel uu ka bilaabo. Nasiibwanaag maadaama uu yahay buuggii igu horeeyay, wuxu ku qoranyahay af-soomaali fudud
2. Arinta labaad ee igu kalliftay qorista buuggan, waxay tahay in maanta dadka soomaalida ah ee wax qoraa aad u yaryihiin. Anigu markii aan isu fiirsaday waxan isu arkay inaan ka mid ahay dadka maanta laga sugaayo iney wax qoraan, si dad ahaan iyo luuqadba aynu usii jirno mustaqbalka. Waxan aaminsanahay inaaney shuruud ahayn inaad qoraalka ku fiicnaato si aad buugaag u qorto. Waxaan shakhsi ahaan rumaysaney iney muhiimaddu tahay inaad waxuun qorto, kadibna gadaal laga saxo wixii qalad aan ulakac ahayn ee kaa dhaca
3. Arinta saddexaad waxa weeye in aan anigu shakhsi ahaan luuqaddayda hooyo barto, aniga oo adeegsanaaya kuna qoraaya buuggan. Arintani waxay igu kaliffeysaa inaan fikiro oo aan raadiyo ereyo badan oo af-soomali ah oo aan adeegsado
4. Arinta afraad waa habka qoraalka af-soomaaliga oo siduu sheegay Boobe Yuusuf Ducaale buug-giisa Timacadde [4]: “Inkastoo af-soomaaligu qornaa tan iyo 1972:kii, haddana xaqiiqada jirtaa waxay tahay in aanu ilaa maanta jirin qaab laga simanyahay (standard) oo ay isku wada raacsan yihiin dadka af-soomaaliga ku hadlaa. Inteenaa soomaali u dhalatay, af-soomaaliga dugsiyo uma aynu gelin ee qof waliba halkuu ku dhashay ama ku barbaaray ayuu kaga hadaaq-bartay.” Sida qoraaguba sheegay waxan aniguna buuggan u adeegsaday af-soomaligii aan noloshayda inteedii badnayd ku hadlayay.

Abwaan Maxamed Xaashi Dhamac oo dadku u yaqaano Gaariye ayaa isaga oo waqooyiga Amer-ica jooga sanadkii 2006 wuxu yidhi “af-ku waa qalab lagu fikiro, oo waxad ku fikiri kartaa kaliya inta vocabulari-gaagu leegyahay”, taas oo abwaanku ula jeeday (waa sidaan anigu u fahmay) in qofku mar-walba ku fikiraayo luuqaddiisa iyo mugga ereyadiisa. Sababtoo ah ereyada uu qofku yaqaano waxay u badanyihiin inta badan luuqaddiisa hooyo. Taasi waxay ku tusaysaa in qofku adeegsado afkiisa hooyo markuu fikiraayo ama doonaayo inuu arin fahmo. Taasi waxay inna tusaysaa oo kale muhiimadda uu leeyahay buug ama buugaag af-soomaali lagu qoro.

Hammiga buuggani waa inuu mawduucan adag u sharaxo si uu fahmikaro, qofkasta oo aan mawdu-ucan aqoon hore u lahayn.

Maxaa loogu baahanyahay borogaraamyada kumbuyuutarka

Si aynu dhadhanka akhristaha u kicino aynu waxyar ka nidhaahno sababta loogu baahanyahay borog-araamyada kumbuyuutarka. Sideedaba ereyga af-ingiriisiga ee *programming* waa erey runtii macno balaadhan leh. Af-soomaaliga waxaynu ku macnayn karnaa borogaraamgarayn ama borogaraamayn. Ereygaas waxa guud ahaan loola jeedaa in nidaamka, kumbuyuutarka, qofka, xayawaanka, iwm. lagaga keeno inay qabtaan ama fuliyaan arimaha lagu amray. Taasi waxay ku tusaysaa in waxyaalo badan oo kala duwan la borogaraamgaraynkaro. Buuggani wuxu ka hadlayaa sida kumbuyuutarka loogu amrikaro inuu qabto waxaad ka rabto ama ku amartay. Sideedaba kumbuyuutarku ma aha shay fikira ama kalidii go’aan qaata, kaliya wuxu fuliyaa amarka la siiyo. Sidaas awgeed amarku waa inuu noqdo mid cad oo aan inyarna shaki ku jirin ama si kale loo turjumankarin. Amarkaasi waa inuu ka koobnaado talaa-booyin cadcad oo isku xiga, oo *luuqad uu kumbuyuutarku fahmaayo* ku qoran. Intaas kadib shaqada kumbuyuutarku waa inuu kaliya fuliyo amarkaas.

Sidaad kollay maqashayba kumbuyuutarada waxa loo kala qaadaa inta badan laba **Hardware** iyo **Software**. Ta hore waa birta iyo electronic-ka aan isbadalaynin, ta dambena waa waxa gudaha ka hagaaya. Birtu waa bir guud, taas oo fulinaysa amarka ka imanaaya ta dambe. Birta si fudud looma badali karo inta badan, laakiin software-ka si aad u fudud ayaa loo badali karaa. Birta waxa caado u noqotay inay

marba marka ka dambaysa sii yaraato. Haddii aanay jirin kala qaybintani, waxa iman lahayd in bir gaar ah loo sameeyo baahikasta oo timaada, taasina wax macquul ah ma aha.

Kumbuyuutaradii maanta meel walba way galeen. Waliba waxa intaas dheer way isku xidhanyihiin. Maanta walxaha iyo kumbuyuutarada isticmaalaaya Internet-ka ayaa ka badan dadka isticmaalaaya. To-baneeyadii sano ee ugu dambaysay waxa soo baxay waxa lagu magacaabay *The Internet of Things*. Waxa bilaabantay in tusaale ahaan qalabka guryaha; kumbuyuutarada, mishiinada sida ka weelka, dharka, talaajadda, TV-ga, electronic-ga kale, iwm. ay wada hadlaan oo macluumaad isu gudbiyaan iyo in Internet-ka laga hagikaro. Waxyaalaha sidoo kale caadiga noqday waxa kamid ah tusaale ahaan kam-eradaha lagu ilaaliyo meelaha kala duwan sida hudheelada ama dukaanada. Xitaa qufuladii albaabada waxay noqdeen qaar kumbuyuutar uu hagaayo oo tusaale ahaan qof meel kale jooga uu kaa furikaro. Lamasoo koobikaro meelaha uu galay kumbuyuutarku. Waxyaalaha aan tusaale ahaan anigu sida joog-tada ah u adeegsado waxa kamid ah inaan toos moobilkayga ugala socdo meesha baskaygii maraayo. Dhammaan waxyaalahaas waxa hagaaya borogaraamyo kumbuyuutar oo ay dad qoreen.

Luuqaddee fiican inaan ku bilaabo

Maanta waxa jira luuqado kumbuyuutar oo badan oo ay luuqadda Java ee buuggani ka hadlaayo ka mid tahay. Waxay dad badani iswaydiinayaan luuqaddii ay baran lahaayeen. Waxan isku dayayaa inaan fikirkayga ku saabsan arintan ka dhiibto intaynaan hore u socon. Arinta ugu muhiimsan ee inta badan luuqadda loo doorto waa inaad mustaqbalka shaqo ku hesho. Sidaas awgeed waxa suurtoagal ah inay dadka qofba luuqad kuu tilmaamo oo kugula taliyo. Waa muhiim inaad sawir guud arintan ka haysato.

Anigu waxan qabaa shakhsi ahaan inay fiicantahay inaanad ku koobnaan hal luuqad oo kaliya ee aad barato laba ama saddex. Taasi waxay kuu saamaxaysaa inaad kolba tii ku haboon hawsha aad qabanayso aad adeegsato. Waxyaalaha inta badan laysku raacay waxa ka mid in borogaraamyada la rabo inay dheereeyaan lagu qoro luuqadaha C ama C++. Sidaas awgeed way fiicantahay inaad wax ka taqaano, laakiin qasab ma aha inaad ku bilawdo, waayo waxa laysku raacay inay tusaale ahaan Java ka fududahay xaga barashada. Waxa kale oo la ogaaday inay programmers-ka fiican u fududahay inay bartaan luuqad cusub, waayo luuqaduhu waxyaalo badan ayay iskaga dhawyihiin.

Habka buuggu u qoranyahay

Buuggu wuxu ku qoranyahay *font* (farta qoraalka) ama font-yo kala duwan. Tusaalooyinku waxay ku qoranyihiin font ka duwan font-ka qoraalka caadiga ah. Sidoo kale kalmadaha cusub ee ingiriisiga ah waxay marka ugu horaysa ku qoranyihiin font *italic* ah ama janjeedhan/qaloocda. Haddii ereygu mar labaad soo laabto wuxu la mid noqonayaa qoraalka caadiga ah.

Waxa jira ereyo badan oo af-ingiriisi ah oo aan af-soomaali turjumaad loo haynin wali ama anigu aanan aqoon u lahayn. Sidaas awgeed buuggani wuxu ku qoranyahay iskujir af-soomaali iyo ereyo badan oo af-ingiriisi ah. Waxan odhankaraa waa badhax af-soomaali iyo af-ingiriisi. Ereyada qalaad waxan u adeegsanayaa siyaasaddan:

1. Inaan ku dadaalo inaan ereyga saxda ah ee af-soomaaliga ah u raadiyo. Sidaas awgeed waxan aad u adeegsanayaa qaamuuska english-somali ee [8]
2. Inaan marka ugu horeysa ee ereygaasi yimaaddo si fiican u macneeyo. Kadibna ereygu ka mid noqdo qoraalka caadiga ah
3. Ereyga qalaad waxan u qorayaa sidiisa, si ardaygu meel kale uga raadsado macnihiisa. Laakiin waxa marmarka qaar ereyga xaga dambe ka raaceysa -ka ama -ta, taas oo af-soomaaligu keenaayo (tusaale: programmer-ka, source code-ka, ...)
4. Qoraalka borogaraamka inta badan waxan u adeegsanayaa ereyo af-ingiriisi ah, sababtoo ah isaga ayaa kaga haboon af-soomaaliga, maadaama aad af-ingiriisiga ka helayso ereyo gaagaaban oo mac-nahooduna cad yahay. Sideedaba inta badan qoraalka borogaraamka waxa lagu qoraa af-ingiriisi, si ay dadka kale ee akhrista ama aad wada shaqaynaysaan ee aan af-soomaaliga aqooni u fahmaan
5. Ereyga af-ingiriisiga ah ee *program*, waxan u adeegsanaaya ereyga borogaraam, halka dadka qaar u adeegsadeen ereyada qorme ama barnaamij, si aanu inooga gadmanin.

Dhammaadka buuggan waxad ka helayaa liiska ereyada muhiimka ah, oo markaad u baahato aad toos uga raadinkarto ereyga aad u baahantahay.

Dardaraan

Shay kasta oo cusub barashadiisu marka hore way adagtahay, sidaas daraadeed waxa loo baahanyahay dulqaad iyo adkaysi. Gaar ahaan barashada luuqadaha kumbuyuutarku waxa la odhan karaa waa waxyaalaha ugu adag ee la barto, laakiin marka wakhti la siiyo oo muddo lagu dhexjiro waa layska fahmayaa. Aniga qudhayda ayey qorista buuggani ii tahay tijaabo, loo baahanyahay inaan u dhabaradeygo, maadaama ay jiraan jirina doonaan dhibaatooyin kala duwan oo iga hor imandoona.

QAYBTA I

TAARIIKHDA IYO DHISMAHA
LUUQADDA JAVA

Qaybtani waa qaybtii ugu horeysay buuggan. Waa qayb gogoldhig iyo dulmar isugu jirta. Waxa ugu muhiimsan oo loogu talagalay iney akhristaha sawir guud ka siiso luuqadda kumbuyuutarka ee Java, waxaanay u kala baxaysaa laba qayb-hoosaad ama cutub.

Cutubka hore wuxu kuu dulmarayaa luuqadda Java iyo marxaladaha kala duwan ee ay soo martay, waxaanu guud ahaan ka hadlayaa taariikhda luuqadda Java. Waxa kale oo uu sharaxaad kooban ka bixinayaa arimaha ugu muhiimsan ee loo baahanyahay inuu akhristuhu ogaado/barto.

Cutubka dambe wuxu ku saabsanyahay qalabka loo baahanyahay si loo barto luuqadda Java, loogana shaqeeyo casharada iyo tusaalooyinka buuggan ku qoran.

Buugga qaybtiisan koobaad waxaynu kaga imaneynaa kor xujada inna hortaagan oo ah barashada luuqadda Java, si akhristuhu uga helo sawir sare luuqadda Java. Korkaas aynu kaga imaneyno waa sida ugu sahlan ee aynu ku heli karno sawir guud oo ku saabsan dhulka aynu ku dagayno oo ah luuqadda Java iyo wixii ku hareeraysan. Markaynu sawir guud ka helo Java, waxaynu hoos ugu daadagaynaa qaybaha ama xubnaha muhiimka ah ee luuqadda Java ka koobantahay. Waxan aaminsaney iney kaartadani tahay ta ugu sahlan ee aynu ku fahmi karno luuqadda Java.

Cutub 1

Dulmar Luuqadda Java

1.1 Waa maxay Java

Shirkadda weyn ee la yidhaahdo Sun Microsystems¹ ayaa sanadkii 1995 soo deysay luuqadda kumbuyuutarka ee Java. Ugu horeyntii waxay ugu talagashay in loo isticmaalo dhinaca Web-ka ama bogagga Internet-ka, oo lagu sameeyo borogaraamyada yaryar oo borogaraamyada *Browser*-ka la dhexgaliyo, kuwaas oo loo yaqaano *Java Applets*. Browser waxa loo yaqaan borogaraamyada lagu akhristo bogagga Internet-ka, waxaana ka mid ah Internet Explorer (oo soomaalidu inta badan u taqaan e-da), Firefox, Safari, Chrome, ... iyo qaar kale. Si kastaba wixii loogu talagaley iyo wixii laga sugaayeyba way dhaaftay luuqadda Java.

Wakhtigan aan buuggan qoraayo Java waxay maraysaa soosaarkeedii ama *Release*-keedii todobaad. Waxyaalihii laga sugaayeyna badankooda waa laga helay, waxaaney maraysaa heer ay qiyaastii 10 miljan oo dad ahi bartaan kuna shaqaystaan². Waxaana laga isticmaalaa shirkadaha aduunka ugu waaweyn sida NASA³, IBM, Eclipse Project, Nokia Corporation, Apache Project iyo qaar kale. Jaamacadaha aduunkana badankooda waa lagu bartaa. Sidaas daraadeed buuggani wuxu fursad u yahay ardayda soomaaliyeed. Fursad ay ku bilaabaan barashada luuqadda Java.

Meelaha loo isticmaalo ama loo adeegsado Java waxa ka mid ah:

1. *Web Servers* - Mishiinada kalawada ama -haga aduunka Internet-ka. Waxaana adeegsada inta badan shirkadaha iyo dawladaha ama guud ahaan meelaha looga baahanyahay in macluumaad badan la kantaroolo ama maamulo
2. *Relational databases* - Mishiinada lagu kaydiyo macluumaadka
3. *Cellular phones* - Telefoonada gacanta
4. *Orbiting telescopes* - Dayaxgacmeedada samada loo diro

Maanta luuqadda Java waxay maraysaa heer meelo badan oo kala duwan loo isticmaalo oo waxay noqotay *general-purpose programming language* ama luuqad kumbuyuutar oo guud (calashaan meelo badan loo adeegsankaro). Taariikhda luuqadda si fiican ayaa loo hayaa, maadaama ay gaabantahay. Ninka lagu magacaabo James Gosling oo u shaqayn jirey shirkadda Sun Microsystems ayaa ah aabaha Java. James wuxu ka shaqeynaayey mashruuc ama project TV-ga loogu talagalay oo *interactive* ah, waxaanu quus ka taagnaa luuqaddii ay isticmaaleysay shirkaddu wakhtigaas oo ahayd C++ oo aan ku habooneyn shaqadan.

Si hadaba uu u gaadho hammigiisa wuxu bilaabay inuu hindiso luuqad cusub. Inkastoo uu ku guuldareystay mashruucii uu gacanta ku hayey, haddana shaqadii uu qabtay waxay faa'iido u noqotay dhanka Web-ka iyo shirkadda Sun. Sidaas awgeed Sun waxay soo saartey daabacaaddii ugu horeysay ee luuqadda Java sanadkii 1995.

Wakhtigaas waxa Java loo isticmaali jirey in lagu sameeyo borogaraamyada yaryar ee Applets-ka, sidaan hore u soo sheegayba. Kuwaas oo la dhexgalinaayo bogagga Internet-ka ee borogaraamka Browser-ka ah ee *Netscape Navigator*⁴.

Inkastoo ay wax fudud ahaayeen, haddana muddo gaaban ayay Java kusoo jiidatay dad aad u tiro badan. Sababtoo ah waxay markii ugu horeysay saamaxdey ama suurtoogelisey in bogaggii Internetka

¹Waxa iibsatey shirkadda Oracle sanadkii 2009

²<http://www.langpop.com/>

³Waa shirkadda samaysa dajaxgacmeedyada ee duulimaadyada samada

⁴Netscape waxa aad loo isticmaali jirey 1990-2000. Waxaaney dadku uga guureen Microsoft Internet Explorer

laga dhigo interactive. Xitaa markii uu dhamaadey wakhtigii Applets-ku, muu damin dabkii luuqadda Java. Sababtoo ah dadka sameeya borogaraamyada kumbuyuutarada waxay arkeen faa'iidooyinkii Java.

Luuqadda Java marka loo eego luuqadaha kale sida C ama C++ way ka fududahay barashadeedu, waaney ka nabadgelyo fiicantahay, waaney adagtahay sida qalad loogu adeegsadaa.

Waxyaalaha ay Java ku caanbaxdey waxa kamid:

- *Object-Oriented Programming (OOP)* oo ah hab borogaramyada loo dhiso oo soo baxay tobaneeyo sano ka hor, markii ay adkaatey samayntii ama dhisitaankii borogaramyada kumbuyuutarada ee waaweyn. Waliba markii habkii hore ee borogaramyada loo samayn jiray oo ku caanbaxay *Spagetti Code* uu noqdey gaabis. OOP waxay ku caanbaxdey waxa loo yaqaano *Object*, oo af-soomaali lagu macneyn karo Walxo (jamac). Faahfaahin dheeraad ah way soo socotaa
- *Platform neutral* oo ka dhigan inaad borogaramkaaga Java adiga oon waxba ka badalin ku kaxaynkarto (*run*) mishiin ka duwan kii aad ku sameysay. Tusaale ahaan haddii aad borogaramkaaga ku sameyso mishiin uu ku jiro *Windows*, waxad ku kaxaynkartaa mishiin kale oo *Linux* ama *OS X* ku jiro. Qodobkani waa arinta ugu muhiimsan ee aynu ku dooranayno Java.

Waxyaalaha ay Java kaga duwantahay C++ waxa ka mid ah:

1. Java hab automatic ah ayay u ilaalisaa xasuusta kumbuyuutarka. C++ waa inaad adigu ilaalisoo xasuusta
2. Java ma laha waxa loo yaqaano *Pointers*. Pointers waa arimaha ugu adag ee hortaagan ardayga raba inuu barto C++
3. Java waxay leedahay hal *inheritance* oo kaliya, halka C++ ka leeyahay *multiple inheritance*. Inheritance dib ayaynu kaga hadlaynaa
4. Java waa *interpreted* oo macnaheedu tahay in markasta oo la rabo in la kaxeeyo borogaraamka Java, loo baahanyahay borogaraam kale oo loo yaqaano *Java Virtual Machine (JVM)* kaas oo loo adeegsado kaliya kaxaynta. C++ uma baahna borogaraam noocan oo kale ah.

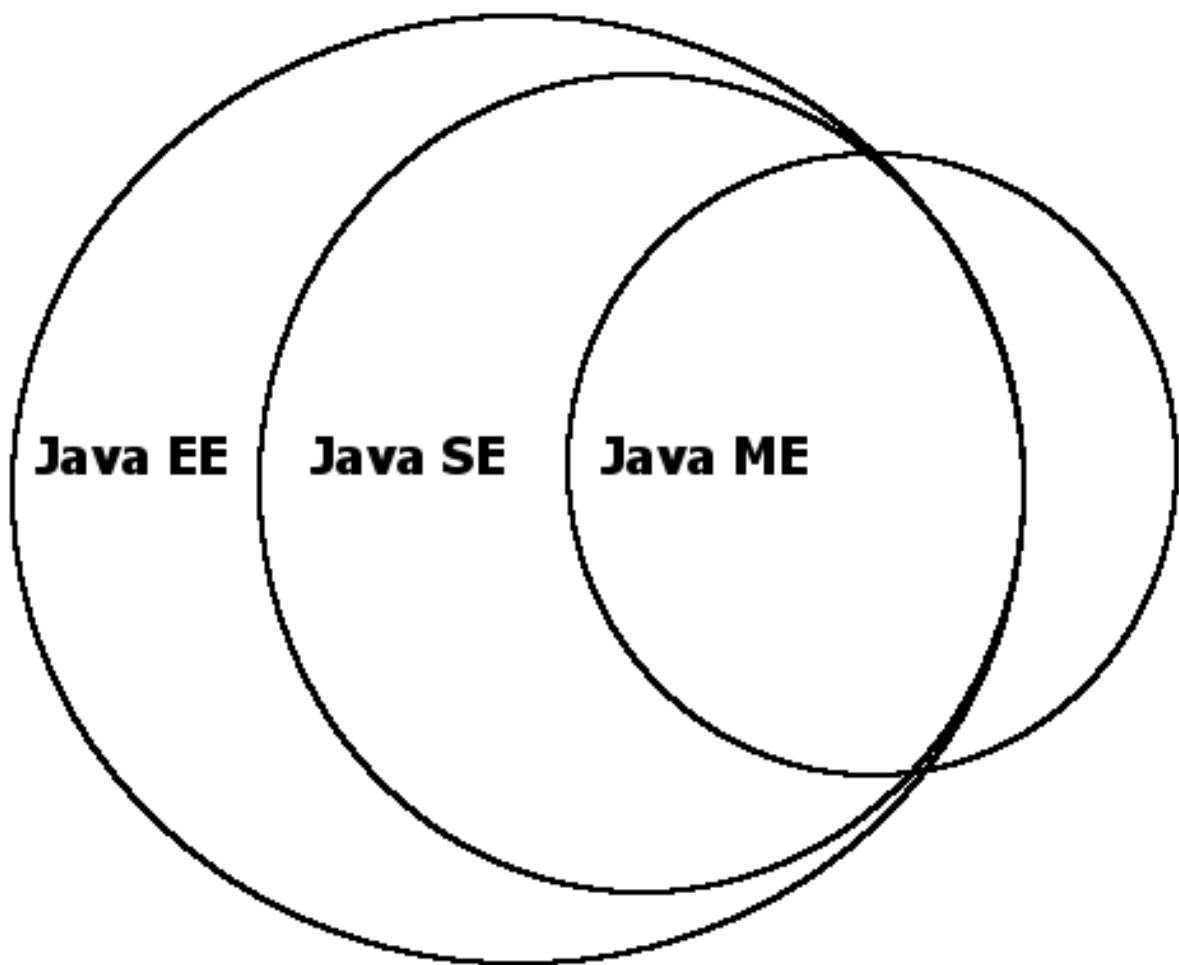
Arinta ugu muhiimsan ee luuqadaha C iyo C++ kaga fiicanyihiin luuqadda Java waa xawaaraha ama *performance*-ka borogaraamka. Haddii aan si kale u dhigo borogaraamyada lagu qoray luuqadda C iyo C++ way ka dheereeyaan kuwa lagu qoray luuqadda Java. Taasna waxa keenaaya borogaraamka dheeraadka ah ee loo baahanyahay si loo kaxeeyo borogaraamyada luuqadda Java. Laakiin inta badan xawaaraha borogaraamyada Java wuu ku filanyahay dadka, sidaas awgeed looma baahna inaad ka warwarto arintaas.

1.1.1 Noocyada Java

Java waxay kusoo baxdaa ugu yaraan afar nooc oo kala duwan oo loogu talagalay marxaladaha kala duwan ee kumbuyuutarada iyo baahida. Tusaale ahaan moobilada iyo kumbuyuutarada inkastoo ay ka simanyihiin waxyaalo badan, haddana waxay ku kala duwanyihiin meelo badan tusaale ahaan mugga xasuusta iyo quwadda fulinta. Sidaas awgeed Java waxay leedahay soosaar loogu talagalay duruufaha telefoonada moobilada iyo kumbuyuutarada gacanta lagu qaato.

Sidoo kale kumbuyuutarada dadka caadiga ay isticmaalaan iyo kuwa ay shirkadaha waaweyn adeegsadaan waxa jirta waxyaalo ay ku kala duwanyihiin oo dhanka baahida la xidhiidha. Shirkadaha waaweyn inta badan waxay adeegsadaan nidaam daadsan oo ka kooban kumbuyuutarro farabadan, kuwaas oo qaarkood ay yihiin nooca *server*-ka. Sidaas awgeed waxa jirta daabacaad Java oo loogu talagalay shirkadaha gaar ahaan. Laakiin ma jirto wax diidaaya in dadka caadiga ah ay isticmaalaan daabacaadda shirkadaha. Dhammaan daabacaadahaas waxay ka simanyihiin waxyaalo badan sida ka muuqata sawirka 1.1.

Sawirka 1.1 waxad ka arkaysaa soosaarada ama daabacaadaha Java ee kala duwan iyo siday u kala duwanyihiin. Sidaad sawirka ka arkayso *Java EE* iyo *Java SE* waa isku mid kaliya waxay ku kala duwanyihiin in Java EE laga helaayo waxyaalo aan ka mid ahayn Java SE. Waxyaalaha ugu muhiimsan ee ku jira Java EE ee aan ku jirin Java SE waa waxyaalaha looga baahanyahay nidaamyada daadsan iyo dhanka Web-ka gaar ahaan samaynta Web Server-rada. Java EE waxad ka helayso qaybiihii luuqadda Java ka midka ahaa ee aad u adeegsan lahayd si aad kumbuyuutarkaaga uga dhigto Web Server laga soo galikaro kumbuyuutarada kale. Sidaan soo sheegayba Java EE waxa loogu talagalay shirkadaha si ay ugu dhisaan nidaamyadooda qaybta dhanka *server*-ka. Java SE waxa loogu talagalay kumbuyuutarrada caadiga ah ee dadku isticmaalaan. Buuggan waxaynu ku adeegsanaynaa Java SE. *Java ME* waa qayb kamid ah Java SE, laakiin waxa jira waxyaalo iyada u gaar ah oo aan kamid ahayn Java SE, sidaad sawirka ka arkayso. Java ME waxa loogu talagalay kumbuyuutarrada gacanta iyo moobilada, oo inta badan awooddoodu iyo xasuustoodu yartahay marka loo eego kumbuyuutarada caadiga ah. Waxyaalaha



Sawirka 1.1: Soosaarrada Java ee kala duwan iyo siday u kala duwanyihiin

kale ee inta badan u gaarka ah kumbuyuuttarada fudud iyo kuwa gacanta waa ku xidhnaantooda Internet-ka oo aan inta badan joogto ahayn. Java ME waxa loo isticmaali karaa meel kasta oo kumbuyuutar khafiif ah looga baahanyahay. Tusaale ahaan waaggii aan jaamacadda ka shaqaynaayey ee aan diyaarinaayey buuggii dhammaadka ama Thesis-kii, waxaanu Java Me u adeegsanaynay hagista mishiinada warshadaha.

Ugu dambayntii Java SE waxa loo yaqaaney J2SE, Java EE J2EE, Java ME:n J2ME. Waxay shirkadda Sun badashay magacyadii sanadkii 2006, oo ay nambarka 2 ka tuurtay magacyadii.

1.2 Waa maxay Java Virtual Machine (JVM)

Sidaan hore u soo sheegayba **JVM** (jey-fii-am) waa borogaraam lagama maarmaan ah si loo kexeeyo ama loo isticmaalo borogaraamyada lagu qoray luuqadda Java. Ujeedada ama hadafka JVM waxa weeye iney kala dhexgasho borogaraamka lagu qoray Java iyo *Operating System (OS)*-ka. Waxa la rabaa in aan laysku xidhiin borogaraamka iyo OS-ka, sababtoo ah OS-ku wuu is badali karaa oo waxa laga yaabaa in loo baahdo in borogaraamka lagu isticmaalo OS ka duwan kii hore ee lagu dhisay. JVM waxay saamaxaysaa in borogaraamkii lagu qoray luuqadda Java lagu isticmaalo OS-yo kala duwan. Sidaan hore u soo sheegayba arintani waxay ka midtahay waxyaalaha ay Java kaga duwantahay C iyo C++.

Qodobkani waa waxyaalaha ugu muhiimsan ee lagu kala sooco luuqadaha kumbuyuutarka. Luuqadaha sida Java oo kale u baahan borogaraam dheeraad ah si loo kexeeyo waxa loo yaqaan *interpreted*. Halka kuwa kale sida C iyo C++ loo yaqaano *compiled language*.

1.3 Waa maxay OOP

Haddii aad dooneyso inaad barato luuqadda Java waa lagama maarmaan inaad isla markaa fahamto hab dhismeedka *OOP* (ow-ow-be). Waad ku qasbantahay, sababtoo ah Java waxay ku saleysantahay OOP. OOP waa hab borogaraamyada kumbuyuuttarada loo dhiso iyada oo lagu dayanaayo habka waxyaalaha kale ee *physical*-ka ah ama muuqaalka leh sida guryaha, baabuurta, iwm loo dhiso. Muhiimaduna waa in borogaraamyada kumbuyuutarka dhismahooda la fududeeyo, maadaama ay adayg iyo weynaansho isku darsadeen. OOP waa hab lagu wajahaajo waxa lagu magacaabo *Software Crisis* oo ah khalkhal ku yimid dhisidda borogaraamyada kumbuyuuttarada. Dhibaatanada ama khalkhalkan waxa keenay horumarka dhakhsaha badan ee sanad walba ku yimaada qalabka *hardware* kumbuyuuttarada. Sida xasuusta, iyo quwadda fulinta (*processing power*).

OOP wuxuu dadka u sahlayaa in ay dhisaan borogaraamyada dhismahoodu habeysanyahay, kuwaas oo ay fududahay⁵ in wax laga badalo ama la fahmo dhismahooda. Borogaramyadii hore (ka hor OOP iyo luuqadaha casriga ah sida Java) waxay ahaayeen qaar dhismahoodu isdhex daadsanyahay oo aan laheyn wax *structure* ah. Waxey aad ugu dhawaayeen hardware-ka sida processor-ka kumbuyuutarka. Waxa toos loo qori jirey nambarada ay processor-ku fahmeyso. Waxaaney ku caanbaxeen magaca spaghetti code, sababtoo ah macluumaadka (*data*) iyo koodhku (*code*) way isdhex daadsanaayeen. Kadib waxa yimid *data structures*, oo waxa la kala qaadey data-dii iyo code-kii xukumaayey. Data-da isla xidhiidha inta laysu uruuriyo ayaa magac loo bixinayaa, waxaana loo yaqaan data structure. Waxa ka dambeeyey hidisiddii *module*-ka, oo saamaxdey in code-kana la qayb-qaybiyo oo wixii isla xidhiidha hal module lagu uruuriyo. Waxaana horumarkan ugu dambeeyey Objects-ka oo ah casrigan imika socda. Waxaynu toos ula jaanqaadeynaa casrigan imika socda.

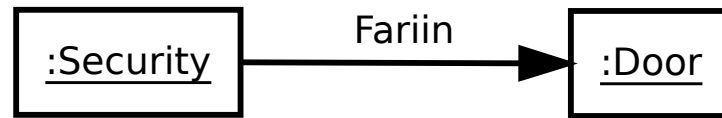
1.4 Waa maxay Objects

Object af-soomaaliga waxa lagu macneyn karaa walax. Markaas *objects* waxay noqoneysaa walxo oo wadar ah. Walxuhu waxay fure u yihiin luuqadda Java. Haddii aad hareerahaaga eegto, waxad arki kartaa tusaalooyin badan oo ah walxaha aduunka (real-world objects), sida: gurigaaga, gaadhiigaaga, baaskeelkaaga, bisaddaada, tv-gaaga iyo qaar kale oo badan. Waxlaha aduunku waxay badankood leeyihiin laba arimood: Kulligood waxay leeyihiin xaalad (state) iyo hab ama ficil (behavior, method). Tusaale ahaan bisadda xaaladaheeda waxa ka mid ah magaceeda (waa bisadda la dhaqdo), kalarkeeda, jinsigeeda (race) iyo iney gaajaysantahay. Ficiilkeedana waxa ka mid ah ci'yitaanka (barking), ugaadhsiga iyo raaxaysiga. Sidoo kale baaskeelku wuxuu leeyahay xaalad (kalarka, geerka ku jira, xawaaraha, iwm) iyo hab (balad geerka, wareejinta cagaha, qabasho bireegga ama fariinka).

Walxaha aduunka inta badan waxa laga codsankaraa ficil, tusaale ahaan bisadda iney socoto, baaskeelka in geerku is badalo, gaadhigu inuu istaadhmo, iwm. Waxa kale oo la badali karaa xaladooda,

⁵Ugu yaraan ka fudud habkii hore

tusaale ahaan gaadhiga oo istaagga laga dhaqaajiyo, bisadda oo cunto la siiyo, iwm. Waxaynu arki-doona in sidoo kale walxaha borogaraamka laga codsankaro ficil ama la badali karo xaaladooda. Haddaba, waa muhiim inaad fahamto ama garwaaqsato xaaladaha iyo hababka walxaha aduunka. Taasi waxay kaa caawineysaa fikirka iyo dhisidda borogaraamyadaada kumbuyuutarka. Waxa jiri kara walxo badan oo aan aduunka ka jirin, laakiin borogaraamka kumbuyuutarka ku jira oo ka muuqda, marka la eego qoraalka borogaraamka oo loo yaqaano *source code*.



Sawirka 1.2: Dirista fariinta (Qaabkani waa sida inta badan loo sawiro walxaha. Xariijinta magaca hoosta laga mariyey iyo labada dhibcodd.)

Walxuhu way wada xidhiidhi karaan sida ka muuqata sawirka 1.2 oo waxay isu diri karaan fariimo (message) kala duwan (walxaha aduunka xidhiidhkooda haddii aynu eegno, waa sida laba qof oo wada-hadlaaya ama qof gaadhi kicinaaya ama laba bisadood oo islaynaysa). Sawirka waxad ka arkeysaa walax qaabilsan ammaanka (Security) oo fariin u direysa walax qaabilsan albaabka (Door). Fariintaasi waxay tusaale ahaan noqon kartaa "xidh albaabka". U dirista fariinta iyo codsashada ficilku waa isku arin. Habkuu albaabka u xidhaayo isagey u taallaa.

Labada walxood ee sawir 1.2 midba shaqo gooni ah oo uu isagu yaqaano habka loo fulinaayo ayuu qabanayaa. Mid wuxu qaabilsanyahay amaanka midna albaabka. Waxa loo yaqaan arintaas *modularity*. Tusaale ahaan kasoo qaad in guri weyn oo albaabo badan leh nidaamkan lagu xidhay ama lagu rakibey. Albaab walba waa in lagu xidho hal walax oo nooceedu yahay Door. Taas oo qaabilsan albaabkaas xidhitaankiisa iyo furitaankiisa. Dirista fariintu waxay ka dhigantahay codsi ficil ama hab oo laga codsanaayo walaxda loo dirayo.

Haddii wakhti dambe la rabo in albaabka si kale loo xidho, waxa la badelayaa walaxda qaabilsan albaabka oo ah Door. Badalka, walaxda shaqada qabaneysa wakhtigii la rabo la badali karo waa arin aad muhiim u ah oo kamid ah sifooyinka fikirka ku saleysan walxaha (OOP), waxaana loo yaqaan *polymorphism*. Waxase shardi ah in aaney walaxda Security fahmin badalka walaxda albaabka. Taasi waxay ku imanaysaa haddii, aaney waxba iska badalin habkii ay walaxda Security u isticmaali jirtay walaxda Door.

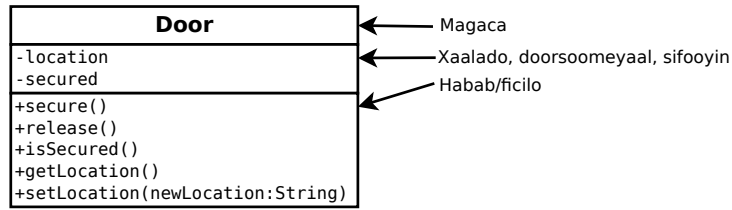
Sidoo kale habka albaabka loo xidhaayo waxa taqaanna walaxda Door oo kaliya, walxaha kale (sida Security) ma yaqaanaan (waayo iyagaa xidhi lahaa, haddii ay yaqaanaan). Marka la rabo in albaabka la xidho waa in laga codsado walaxda Door. Sifadan waxa loo yaqaan *encapsulation*, waxaaney ka mid tahay sifooyinka ugu muhiimsan ee fikirkan.

Waxa kale oo muhiim ah inaad fahamto in qoraalka walxahan ay dad kala duwani qori karaan, oo isu diri karaan. Taasi waxay saamaxaysaa in dad kala duwan oo aan tusaale ahaan isku meel joogin ay borogaraamka wada dhisaan. Waxase muhiim ah iney ku heshiiyaan protocol-ka labada walxood ku wada hadlayaan oo loo yaqaano inta badan *interface*. Sidoo kale haddii mar hore la qoray walaxda, dib ayaa loo isticmaali karaa mar dambe haddii loo baahdo, waxaana loo yaqaan *code re-use* arintaas.

Waxaynu halkan ku dulmarey ugu yaraan shan arimood oo muhiim ah oo kala ah: modularity, polymorphism, encapsulation, interface iyo code re-use. Kalmadahani way soo laaban doonaan inta buuggani socdo. Waxa kale oo aynu cadeynay ineynu ereyga walax u isticmaali doono ereyga ingiriisiga ah ee object.

1.5 Waa maxay Classes

Haddii aad hareerahaaga eegto waxad arkeysaa walxo aduun oo badan oo dhammaantood isku nooc ah. Tusaale ahaan, waxad arki kartaa baaskeelo badan oo isku nooc ama moodhel ah, kuwaas oo ka sameysan qaybo isku mid ah. Marka la eego fikirka ku saleysan walxaha, waxa la odhanayaa baaskeelku waa mid ka mid ah walxo isku mid ah oo loo yaqaano baaskeelo. Haddaba, *Class* waa waxa qeexaaya qaabka ay baaskeeladu u dhisanyihiin, lagana sameynkaro baaskeelo cusub. Haddii aan luuqadda ingiriisiga ku sheego: A class is a template, or blueprint, for constructing new objects.



Sawirka 1.3: Nooca Door (Door Class. Noocyada inta badan habkan ayaa loo sawiraa). Nooc kastaaba wuxu leeyahay magac, sifooyin iyo habab ama ficilo.

Haddii aan ku laabto sawirka 1.2, tusaale ahaan, waxa laga yaabaa in dhismaha ay ku yaalaan albaabo dhawr ah, kuwaas oo midwalba uu gooni u ilaalinaayo walax Door ah. Walxahan dhammaantood waa isku nooc, oo noocoodu waa Door sida ka muuqata sawirka 1.3. Dhammaantood waxay leeyihiin laba sifo (attribute, field) Dhammaantood waxay leeyihiin habab ama ficilo (methods, behaviors), sida xidh (secure), fur (release), sheeg inaad xidhanyahay iyo in kale (isSecured), sheeg meesha aad ku taalo (getLocation) iyo badal meesha aad ku taalo (setLocation). Sidoo kale waxay dhammaantood leeyihiin laba sifo oo kala ah meesha uu albaabku ku yaalo (location) iyo inuu xidhyahay iyo in kale (secured).

Haddii la eego source code-ka nooca sawirka 1.3 ee Door wuxu u eekaan karaa sida tusaalaha 1.1. Wuu yara dheeryahay qoraalku, laakiin waxa jira qaybo badan oo aan ka mid ahayn borogaraamka la intifaacsanaayo, laakiin loogu talagalay in fahamka qoraalka borogaraamka lagu kordhiyo.

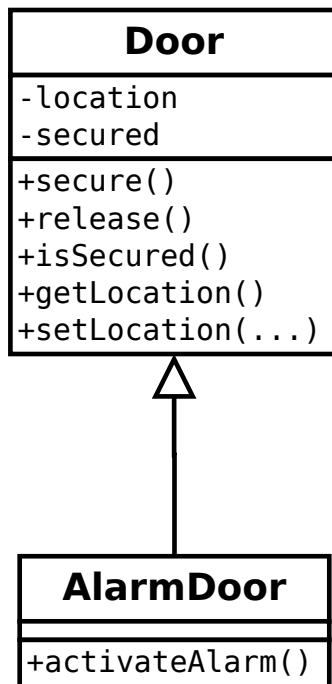
```

1 package door;
2
3 /**
4  * Nooca door
5  */
6 public class Door {
7
8     private String location; // Meesha albaabku ku yaallo
9     private boolean secured;
10
11     /**
12      * Xidh albaabka
13      */
14     public void secure() {
15         // halkan ku qor qaabka xidhista albaabka
16         this.secured = true;
17     }
18
19     /**
20      * Fur albaabka
21      */
22     public void release() {
23         // halkan ku qor qaabka furista albaabka
24         this.secured = false;
25     }
26
27     public String getLocation() {
28         return location;
29     }
30
31     public void setLocation(String newLocation) {
32         this.location = newLocation;
33     }
34
35     public boolean isSecured() {
36         return this.secured;
37     }
38 }

```

Tusaale 1.1: Nooca Door

Luuqadda Java haddii aynu u soo laabano, noocyadu ama class-yadu waa waxa laga sameynaayo walxaha ama objects-ka. Arintan waxa loo yaqaan *instantiation*. Markaad eegto qoraalka borogaraamka waxad ku arkeysaa inta badan qoraal u eeg tusaalaha 1.2, kaas oo ku tusaaya sameynta walax cusub. Waxaynu ku barandoona buuggan qaybtiisa labaad si fiican oo qoto dheer arimahan.



Sawirka 1.4: Takhasus (Inheritance. Habkani waa sida takhasuska inta badan loo sawiro). Falaadhu waxay ku tuseysaa in AlarmDoor laga soo dhiraandhiriyeey Door, sidaas awgeed uu yahay Door takhasusay.

```

...
2 // samee walax cusub oo Door ah, una bixi d1
Door d1 = new Door();
4 /* ka codso d1 inuu is xidho */
d1.secure();
6 ...

```

Tusaale 1.2: Instantiation ama sameynta walax cusub

1.6 Waa maxay UML

Haddii la rabo in guri la dhiso inta badan waa in design-kiisa la dajiyo oo warqad lagu sawiro, si injineerada ama dadka fahmaaya loo tuso oo ay aragtidooda ka dhiibtaan isuguna fahmaan. Dhisidda borogaraamyada kumbuyuutarku waa la mid. Inta badan waxa lasoo diyaariyaa sawirada kor ka muuqda oo kale, kuwaas oo ay ka muuqdaan noocyada ku jira borogaraamka. Laakiin way ka weynyihiin tusaalooyinkan yaryar ee aynu soo qaadnay, maadaama system-ka la dhisayaa ka weynyahay tusaalooyinkan. Sawirada noocan oo kale ah waxa loo yaqaan *UML diagrams* ama *Class diagrams*. Waxad ka heli kartaa Internet-ka macluumaad badan oo laga qoray *Unified Modeling Language (UML)*. Borogaraamka aan u adeegsaday sawirista *UML*-ka waxa la yidhaahdaa DIA⁶.

1.7 Waa maxay Inheritance

Waxa jirta arin kale oo aad ugu badan fikirka walxaha, taas oo loo yaqaano *inheritance* ama *generalization*. Af-soomaali waxan ku macneeyey takhasus, sababtoo ah waxay la xidhiidhaa takhasuska caadiga ah ee dadku wadaayaqano. Takhasuska waxan ku sharaxayaa sawirka 1.4. Kasoo qaad in dhismaha aad u sameyneyso system-kan ay ku yaallaan albaabo ka yara duwan kuwa kale, kuwaas oo ay saacad ama qayliso ku xidhantahay (Alarm). Albaabadan sida kuwa kale waa la xidhikaraa (secure) waana la furikaraa (release), laakiin waxay dheeryihiin kuwa kale ficilka daarista qaylisada (activateAlarm). Albaabada noocan oo kale ah waa in lagu xidho walxo noocoodu yahay AlarmDoor, si markii la doono fariinta (activateAlarm) loogu diro, marka la rabo in qaylisada la daaro. Halkan waxa kugu filan inaad aragti guud ka hesho takhasuska, waayo waxad isaga oo la isticmaalaayo ku arki doontaa casharada dambe. Nooca AlarmDoor wuxu gali karaa meel kasta oo nooca Door uu galikaro. Noocyada sida noocan AlarmDoor oo kale ah ee laga soo dhiraandhiriyeey nooc kale, waxa loo yaqaan *Subclass*, halka nooca laga soo dhiraandhiriyeey sida Door loo yaqaano *Superclass*. Qasab ma aha in takhasusku laba nooc oo kaliya ku eekaado, ee tirada noocyada ama dhererka takhasusku wuu dheeraan karaa oo wuxu ka koobnaaan karaa noocyo badan. Tusaale ahaan waxaynu samayn karnaa nooc saddexaad oo laga

⁶<http://projects.gnome.org/dia/>

soo dhiraandhiriyeey nooca AlarmDoor, laakiin halkii uu kaliya qaylisada ka daari lahaa fariin u diraaya maamulaha guriga tusaale ahaan fariin telefoonkiisa moobilka ah. Noocaas waxaynu tusaale ahaan u bixin karnaa SmsAlarmDoor.

Fikirka takhasusku wuu fududyahay, laakiin aad ayuu u quwad weynyahay. Haddii uu jiro nooc leh ficillo aad u baahantahay, waxad noocaaga kasoo dhiraandhirin kartaa noocaas. Sidaas ayaad ku dhaxlaysaa wixii ficillo iyo sifooyin uu lahaa noocaasi.

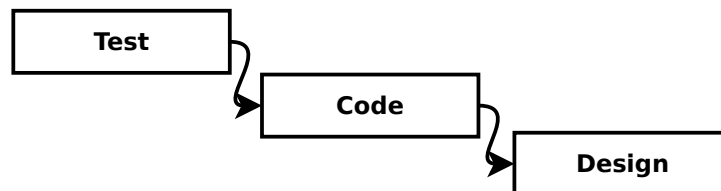
1.8 Waa maxay TDD

Dhisidda borogaraamyada kumbuyuutarka ee waaweyn ma aha shaqo fudud. Waxay ka bilaabmaysaa qeexista borogaraamka loo baahanyahay, waxaanay soconaysaa inta borogaraamkaas la adeegsanaayo. Dadka ugu muhiimsan ee arintan ka qaybqaata waa macmiilka iyo shaqaalaha shirkadda borogaraamka dhisaysa. Waa mashruuc u baahan maamul fiican. Qorista qoraalka borogaraamku (source code) waa qaybta ugu muhiimsan ee mashuuc. Si arintaas loo fududeeyo waxa la hindisay nidaamyo badan oo la raaco, kuwaas oo lagu magacaabay *software development methodologies*. Nidaamyadaas waxa kamid ah *Waterfall*, *Prototyping*, *Spiral*, *Scrum* iyo *XP*. Buuggani kama hadlaayo nidaamyadaas iyo waxay ku kala duwanyihiin. Nidaamyadii ugu dambeeyey inta badan waxa loo yaqaan *Agile software development*, waxaana kamid ah *Scrum* iyo *XP*, kuwaas oo aan anigu aqoon u leeyahay. Nidaamka **Extreme Programming (XP)** waxa qayb muhiim ah ka ah TDD, taas oo la adeegsado marka la qoraayo source code-ka borogaraamka.

Haddii aan waxyar ka sheego **Test-Driven Development (TDD)**. Haddii aan ku bilaabo ereyga ugu horeeya oo ah *Test*, af-soomaalina ka dhigan tijaabin. Tijaabintu waxay waligeed ahayd qayb muhiim ah oo ka mid ah sameynta ama dhisidda borogaraamyada kumbuyuutarka. Waaggii hore iyo runtii wali illaa hadda socota, waxa borogaraamka marka la dhameeyo loo dhiibijirey qof (ama dad) loo yaqaanno *tester* ama tijaabiye, kaas oo shaqadiisu tahay inuu borogaraamka ka raadiyo cilado ama qaladaad, kadibna u gudbiyo qoraaga borogaraamka.

Halkii awal hore laga isticmaali jirey kaliya qof (ama dad) tijaabiye ah, ayaa waayahan dambe waxa la bilaabey in la isticmaalo tijaabiyihii iyo borogaraam tijaabiye ah. Borogaraamkaas tijaabiyaha ah qoristiisa waxa iska leh shakhsiga qoraaya borogaraamka. Muhiimadda borogaraamka tijaabiyaha ah ma aha inuu badalo qofka tijaabiyaha ah ee waa inuu kalkaaliyo. Borogaraamka tijaabiyuhu wuxu yeeshaa heerar kala duwan. Heerka ugu hooseeya oo loo yaqaano *unit testing* waxa qorra isla qofka qoraaya borogaraamka la intifaacsanaayo oo ah developer-ka ama programmer-ka sidaan hore u soo sheegay. Borogaraamka tijaabiyaha ah shaqadiisu waa inuu soo qabto qaladaadka ku jira borogaraamka ugu dambayntii la intifaacsanaayo wakhti hore ama wakhtiga la sameynaayo. Heerka ugu sareeya oo loo yaqaano *Acceptance Testing* waxa suurto gal ah inay dad kale qoraan.

TDD waxay ku leedahay ku horee qorista borogaraamka tijaabiyaha, oo mar kasta oo aad wax ku darayso borogaraamka waa inaad horta qorto borogaraamkii tijaabin lahaa waxaas. Buuggana waxaynu ku isticmaaleynaa **TDD**, markaynu qorayno borogaraamyadeena tusaalaha ah. Waxa muhiim ah inuu akhristuhu ogaado inaanu buuggani ka sheekaynaynin **TDD** si qoro dheer, ee kaliya uu adeegsanaayo qaybyar oo ka mida.



Sawirka 1.5: Talaabooyinka Test-Driven Development (TDD)

TDD waxay ka koobantahay saddex talaabo oo lagu celcelinaayo, kuwaas oo aad ka arkayso sawirka 1.5. Talaabada ugu horaysa ee *Test* ayaad ka bilaabaysaa, halkaas oo aad qorayso borogaraam yar oo tijaabiye ah. Waxad ku xigsiinaysaa talaabada labaad ee *Code*, oo aad qorayso borogaraamkii la intifaacsan lahaa ee tijaabadu tijaabinaysay. Talaabada ugu dambeysa ee *Design* waxay leedahay magac kale oo ah *Refactor*. Ujeedada Design ama Refactor waa in borogaraamka dhismihiisa la fiicnaysiiyo iyada oo aan habka uu u shaqaynaayo waxba iska badalin, oo aan la jabinaynin tijaabooyinka. Saddexdan talaabo ee **TDD** waxa loo kala sameeyay kalarro. Test kalarkeedu waa casaan (Red). Code kalarkeedu

waa doog ama cagaar (Green). Design kalarkeedu waa cagaar iyana (Green). Isticmaalka **TDD** waxad is fiican ugu barandoontaa marka aynu bilaabno ineynu dhisno borogaraamka tusaalaha ah.

TDD ma aha hab Java oo kaliya ku eeg ee waa nidaam ama waddo guud oo la raaci karo iyada oo xitaa la adeegsanaayo luuqad kumbuyuutar oo ka duwan Java. Waxa muhiim ah inuu akhristuhu ogaado inaanay qasab ahayn inaad adeegsado **TDD**, oo aad ka boodikarto dhammaan wixii la xidhiidha. Laakiin waan kugula talinayaa inaad adeegsato intaad karikarto.

1.9 Waa maxay JUnit

JUnit⁷ sida uu u dhigay ninkii qoray Kent Beck "JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.". J-du waxay ka imanaysaa Java, laakiin xUnit waa qaab-dhismeed guud. JUnit waxa loogu talagalay in lagu qoro borogaraamka kale (*Test code*), kaas oo shaqadiisu tahay inuu tijaabiyo borogaraamkaaga la intifaacsanaayo (*Code*). Oo sidaan hore u soo sheegnay, shaqadiisu tahay inuu buuxiyo qofka tijaabiyaha ah. Borogaraamkan kale shaqadiisu waa inuu ciladaha (*Bug* ama *Bugs*) ku jira borogaraamka aad qoreyso soo qabto isla marka aad qoreyso. JUnit waxaynu u adeegsaneynaa talaabada koobaad ee nidaamka **TDD** sidaan kusoo marnay qaybta 1.8.

Sida sawirka 1.6 ka muuqata, markaad qoreyso borogaraamka tijaabiyaha ah, waxad adeegsanaysaa JUnit. Markaad qoreyso borogaraamka la intifaacsanaayo waxad adeegsanaysaa Java. JUnit iyo Java waa qaab-dhismeed guud (frameworks), waxaana lagu dhisi karaa borogaraamyada kumbuyuutarka.

Sidee ayaan JUnit u adeegsanayaa? Markaad rabto inaad nooc ama class cusub qorto waxad ku bilaabaysaa inaad marka hore qorto nooc ama class kale oo tijaabiyaana, kaas oo inta badan magaciisu ku dhamaado *Test. Tusaale ahaan haddii aan rabo inaan qoro nooc cusub oo ah Door sida ka muuqata sawirka 1.7, waxan ku bilaabayaa inaan qoro noocii tijaabinaayey oo aan u bixin karo DoorTest (waa qaabka aynu u magacaabeyno buuggan dhammaan noocyada tijaabiyaha ah, laakiin haddii aad rabto waxad ula bixi kartaa magac ka duwan).

Haddii nooca Door uu u eegyahay sida ka muuqata tusaalaha 1.1 ee aynu soo dhaafnay, sidee ayuu u eegyahay nooca DoorTest? Nooca DoorTest wuxu u eekaan karaa tusaalaha 1.3. Nooca DoorTest shaqadiisu waxa weeye inuu tijaabiyo nooca Door oo hubiyo inuu sidii la rabay u shaqaynaayo. Waxyaalaha ay labadan nooc ku kala duwanyihiin haddii aad eegto dhanka qoraalka, waxad arkaysaa in nooca DoorTest ficiladiisa ay ka sarayso @Test, taas oo ah calaamad lagu garanaayo ficilada tijaabiyaha ah. Waxa kale oo aad qoraalka DoorTest iyo nooc kasta oo tijaabiye ah ku arki doontaa `assertNotNull(...)`, `assertTrue(...)`, `assertFalse(...)`, `assertEquals(...)`, iwm, kuwaas oo aynu u soo laaban doono.

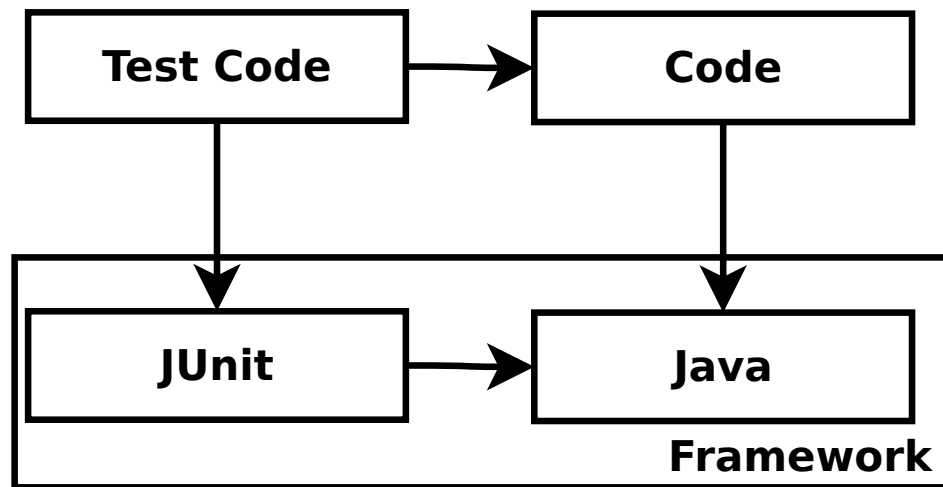
```

1 package door;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class DoorTest {
8
9     @Test
10    public void testCreate() {
11        Door albaabkaHore = new Door();
12        assertNotNull(albaabkaHore);
13    }
14
15    @Test
16    public void testChangeLocation() {
17        Door albaab = new Door();
18        albaab.setLocation("Xagga dambe");
19        assertEquals("Xagga dambe", albaab.getLocation());
20    }
21
22    @Test
23    public void testXidhFur() {
24        Door albaab = new Door();
25        albaab.secure();
26        assertTrue(albaab.isSecured());
27        albaab.release();
28        assertFalse(albaab.isSecured());
29    }
30 }

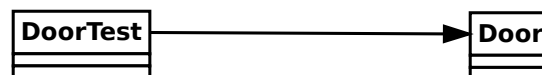
```

Tusaale 1.3: Nooca DoorTest

⁷<http://www.junit.org/>



Sawirka 1.6: Xidhiidhka ka dhexeeya JUnit, Java, Test Code iyo Code. Code waa qoraalka borogaraamka aad qorayso ee la intifaacsanaayo, Test Code waa qoraalka borogaraamka tijaabiyaha ah.



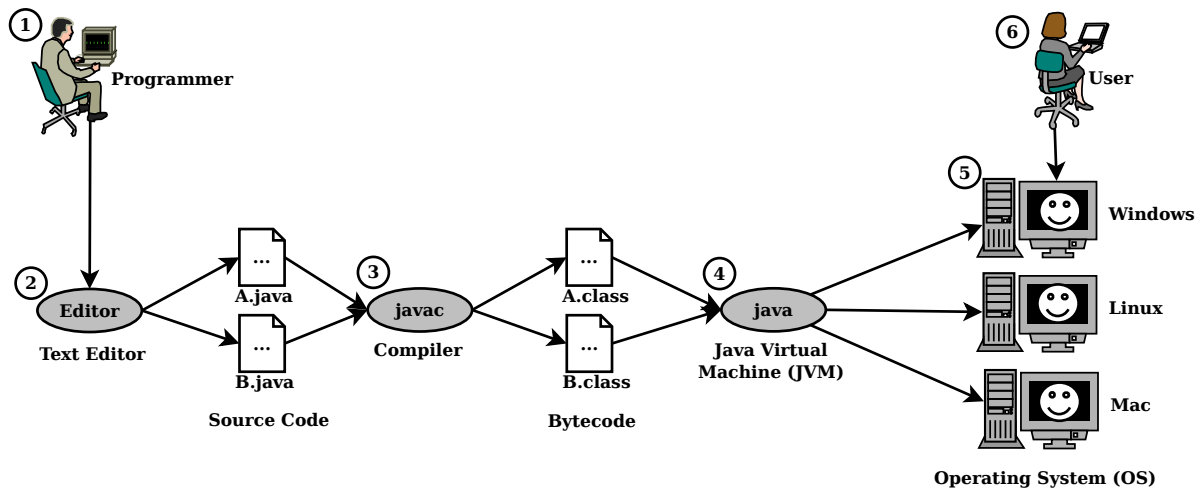
Sawirka 1.7: Nooca Door iyo nooca tijaabinaaya ee DoorTest

Cutub 2

Diyaargarow

2.1 Qorista borogaraamyada Java

Waxan sawir aad u sareeya ka bixinayaa qorista borogaraamyada Java talaabooyinka kala duwan ee ay ka koobantahay.



Sawirka 2.1: Talaabooyinka ay ka koobantahay qorista borogaraamyada luuqadda Java.

Sawirka 2.1 wuxu ku tusayaa talaabooyinka kala duwan ee ay ka koobantahay qorista ama sameynta borogaraamyada luuqadda Java lagu qoray ama lagu dhisay. Waxan rabaa inaan sharaxaad ka bixiyo talaabooyinkaas.

1. Marka ugu horeysa waxa loo baahanyahay qof ama shakhsi, kaas oo aqoon u leh luuqadda Java ama ku jira barashadeeda. Shakhsigaas waxa loo yaqaan **programmer** ama **developer**, sidaan hore u soo sheegay. Shakhsigaasi wuxu noqonkaraa wiil ama gabadh. Dad badan ayaa aaminsan in shaqada qorista borogaraamyada kumbuyuutarku tahay shaqo rag, laakiin waxa jira dumar badan oo ku shaqaysta xirfaddan. Haddaba shakhsigaasi si uu u qoro borogaraamyada Java, waxa lagama maarmaan ah inuu haysto kumbuyuutar uu hawshaas u adeegsado, sababtoo ah kumbuyuutar la'aantiis suurtoagal ma aha in la qoro borogaraamyo kumbuyuutar, waxay la mid tahay adiga oo yidhaahda waxan baranayaa darawalnimo bilaa gaadhi. Kumbuyuutarka waa in laga helo dhammaan borogaraamyadii kale ee lagama maarmaanka ahaa si hawshaas loo fuliyo
2. Borogaraamyada kumbuyuutarku sidoodaba, marka la qoraayo, waxay ka koobanyihiin faylal (**files**), kuwaas oo noqon kara noocyo kala duwan, sida qoraal, sawir, cod, video, iwm. Faylashaas borogaraamku ka koobanyahay waxa loo yaqaan **source code**. Marka laga hadlaayo Java faylashaas badankooda magacoodu wuxu ku dhamaadaa **.java** (dhibic java), sida ka muuqata sawirka 2.1. Waxa loo baahanyahay borogaraam lagu qoro faylashan, kaas oo loo yaqaan **Editor** (waa faylasha qoraalka ah, kuwa kale waxa loo isticmaalayaa borogaraamyo kale). Source code-ka borogaraamka sawirka 2.1 wuxu ka koobanyahay laba fayl oo kaliya, oo kala ah **A.java** iyo **B.java**
3. Qorista kadib, waxa loo baahanyahay borogaraam kale oo source code-kii lagu badalo, kaas oo loo yaqaano **compiler**. Compiler-ka Java magaciisa waxa la yidhaahdaa **javac** (jafak ama java-

see). Compiler-ka Java waxay soo tufaysaa waxa loo yaqaano *Bytecode* (bait'koodh) oo ah faylal kale oo magacoodu ku dhamaado `.class` (dhibic class). Inta badan fayl kasta oo source code-ka ku jira oo magaciisu ku dhamaado `.java` compiler-ku waxay soo tufaysaa fayl la magac ah, laakiin magaciisu ku dhamaado `.class`. Bytecode-ka waxa loogu talagalay kumbuyuutarku inuu fahmo, sidaas awgeed dadku waxba kama fahmayaan haddii aad tusaale ahaan furto. Sawirka 2.1 compiler-ku waxay soo tuftay laba fayl oo kala ah `A.class` iyo `B.class`, oo u kala dhigma labada fayl ee uu source code-ku ka koobanyahay

4. Kumbuyuutarka ugu dambeyn lagu isticmaalayo borogaraamka waa in laga helo *JVM*, sidaan kusoo sheegnay qaybta 1.2. Magaca borogaraamka JVM waxa la yidhaahdaa `java`. *JVM* waxay fahmaysaa byte code-ka
5. Borogaraamka waxa lagu isticmaalayaa kumbuyuutar uu ku jiro *OS*. OS-ku wuxu noqon karaa Windows, Linux ama nooc kale. Borogaraamkaaga markaad kaxeeyso, waa inuu shaashadda soo fuulo oo uu isticmaaluhu arko. Haddii aad tahay dhisaha borogaraamka waa inaad u diyaargarawdo in borogaraamkaaga lagu isticmaalo OS ka duwan kii aad ku dhistay, sida haddii aad Windows ku dhistay, waa in lagu isticmaali karo mishiin uu ku jiro Linux
6. Ugu dambeyntii waxa loo baahanyahay shakhsi isticmaala ama adeegsada borogaraamka. Shakhsi-gaasi wuxu noqon karaa adiga oo ah dhisihii ama qof kale oo isticmaale *User* ah. Isticmaaluhu borogaraamka wuxuu ku fushanayaa hawshiisa.

2.2 Borogaraamyada aad u baahantahay

Si aad u qorto borogaraamyadaada Java waxad u baahantahay qalab ama borogaraamyo aad adeegsato. Borogaraamyadaas waxa loo yaqaan *Development Tools*, waxaana lagama maarmaan ah inaynu halkan kuusoo qaadano qaar ka mid ah, maadaama qaybtani tahay diyaargarawgii. Halkan waxaan kusoo qaadaneynaa qaar ka mid ah borogaraamyadaas aynu adeegsankarno markaynu qorayno borogaraamyada Java. Sida ka muuqata sawirka 2.1, borogaraamyada aad u baahantahay waxa ka mid Editor, compiler iyo JVM. Waxaad kala dooran kartaa laba doorasho midkood oo aad midkood isticmaali karto:

1. *Integrated Development Environment (IDE)*
2. Text Editor, *SDK* iyo *Command line*

Waxan sharaxaad yar ka bixinayaa labadan doorasho midkoodba gooni iyo waxay ku kala fiicanyihiin siday u kala horeeyaan.

2.2.1 Integrated Development Environment (IDE)

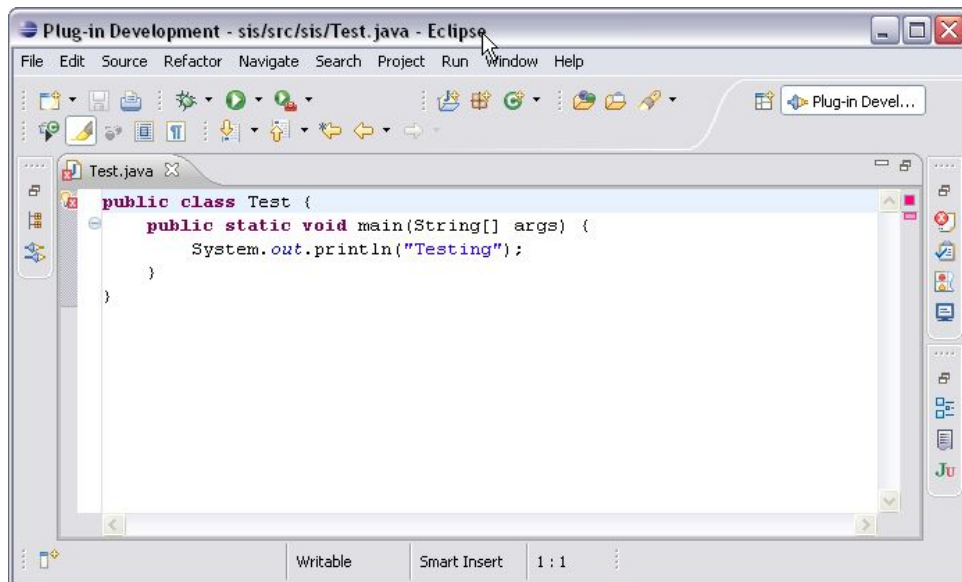
IDE waxa loo yaqaan borogaraam dhamaystiran oo la adeegsankaro marka la qoraayo borogaraamyada kumbuyuutarka. Waxa jira borogaraamyo badan oo noocan oo kale ah. Qaar bilaash ah iyo qaar lacag ah labaduba way jiraan. Marka la samaynaayo ama la qoraayo borogaraamyada luuqadda Java lagu qoray inta badan waxa la adeegsadaa saddex IDE oo kala ah: *IntelliJ IDEA*, *NetBeans* iyo *Eclipse*. Saddexdooda waxa ugu fiican *IntelliJ IDEA*, laakiin waa lacag. Labada kale waa bilaash oo waxad ka helayso internetka. Waxaynu halkan kusoo qaadaneynaa *Eclipse* oo aad ka arkeyso sawirka 2.2. *Eclipse* waxad ka helayso dhammaan waxyaalaha aad u baahantahay markaad dhiseyso borogaraamyada Java, waana bilaash. Waxad ka helayso borogaraamyada aad u baahantahay ee Text Editor iyo compiler-ka Java ee `javac`.

Borogaraamka *Eclipse* waxad ka helayso adarayska:

<http://www.eclipse.org/downloads/>. Waxan go'aansaday inaynu buuggan ku adeegsano *Eclipse*, maadaama uu isagu ugu fududahay. Laakiin waxa muhiim ah in aad ogaato in *Eclipse* laftiisa laga maarmi karo, waa haddii aad qaadato doorasha labaad.

Ku kaydso kumbuyuutarkaaga Eclipse

Halkan waxan ku sharaxayaa sidaad ugu kaydskarto kumbuyuutarkaaga borogaraamka *Eclipse*. Halkan waxaynu ku adeegsanaynaa *Eclipse* soosaarkiisa 3.5 oo loo yaqaano *Galileo* ama 3.6 oo loo yaqaano *Helios* ama 3.7 oo loo yaqaano *Indigo*, laakiin kuwa ka dambeeyey ee ka cusub waad isticmaali kartaa, haddii aanay wax wayni iska badalin. Waxan sharaxayaa sidaad *Eclipse* ugu kaydinkarto (install) Windows iyo Linux gaar ahaan Ubuntu.

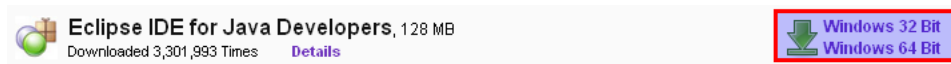


Sawirka 2.2: Eclipse SDK (Windows)

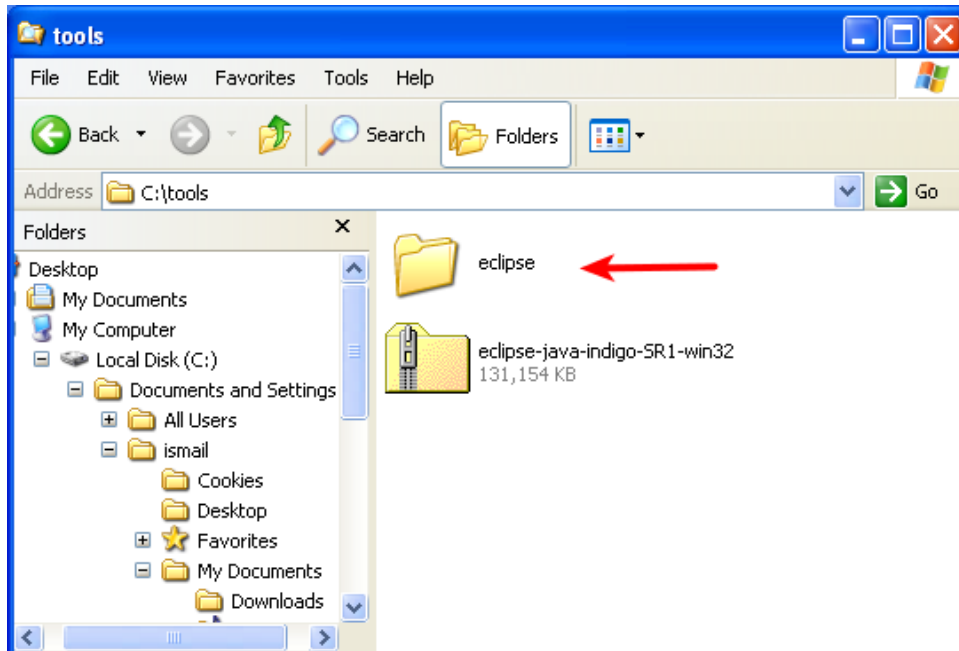
Windows

Waxaynu halkan **Eclipse** ku kaydinaaynaa kumbuyuutar uu ku jiro Windows XP. Innaga oo raacayna talaabooyinkan hoose:

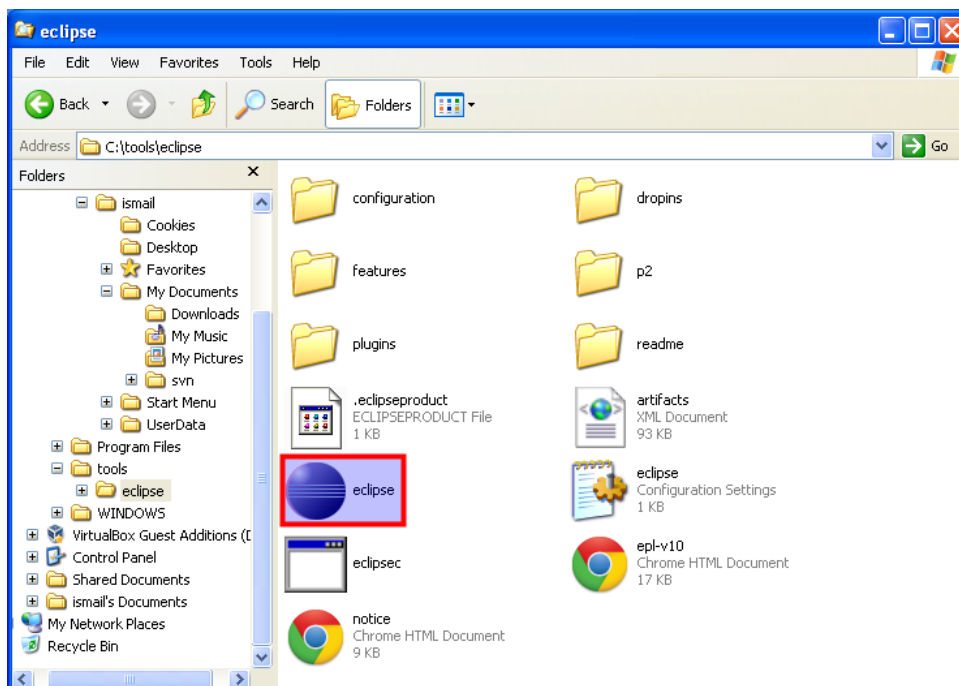
1. Waxad gashaa bogga Internet-ka ee aad **Eclipse** kala soo dagi karto:
<http://www.eclipse.org/downloads/>
2. Sida ka muuqata sawirka 2.3, waxad raadisaa *Eclipse IDE for Java Developers*, kadibna dhanka midigta ka dooro *Windows 32 Bit* ama *Windows 64 Bit*. 32-ka dooro maadaama uu isagu mar walba kumbuyuutarkaaga ku shaqaynaayo
3. Bogga labaad waxad gujisaa falaadha cagaarka ah ee hoos u jeedda
4. Kadib faylkaas magaciisu ku dhamaado **.zip**, waxad ku kaydisaa meel kumbuyuutarkaaga ka mida. Anigu inta badan waxan ku kaydiyaa borogaraamyada noocan oo kale ah **c:\tools**
5. Waxad ku furfurtaa faylkaas isla meesha aad ku kaydisay. Waxad adeegsan kartaa borogaraamyada faylasha noocodu yahay **.zip**-ka lagu akhriyo sida **7zip**, **WinZip**, iwm. Windows-ka laftiisu wuu furfuri karaa faylasha noocan oo kale ah. Waa inaad furfurka kadib hesho gol (*folder*) cusub oo magaceedu yahay **eclipse**, sida ka muuqata sawirka 2.4
6. Laba jeer guji golka **eclipse**. Kadibna, laba jeer guji faylka **eclipse** sida ka muuqata sawirka 2.5. Markasta oo aad rabto inaad **Eclipse** shiddo faylkan laba jeer guji
7. Maadaama **Eclipse** laftiisa lagu qoray Java, wuxu u baahanyahay JVM si loo kexeeyo, markaas waxa laga yaabaa inaad hesho daaqadda sawirka 2.6, oo uu **Eclipse** leeyahay waxan u baahanahay JRE ama JDK, taas oo kumbuyuutarkaaga ka maqan. Ku kaydi kumbuyuutarkaaga JRE, si aad **Eclipse** u kaxayso, halkanna ugasii socoto intaad ku booddo qaybta hoose ee *Ku kaydso kumbuyuutarkaaga JRE*. Waxay ku kala duwan yihiin JRE, JVM iyo JDK waxad ka helaya qaybta soo socota
8. **Eclipse** marka ugu horaysa ee la shido wuxu ku waydiinayaa meesha uu ku kaydindoona faylashaada (projects), sida ka muuqata sawirka 2.7. Calaamadi meesha yar ee hoose kadibna OK guji. Wixii aad borogaraam qorto wuxu ku kaydinaayaa meeshaas
9. **Eclipse** waa diyaar markaad hesho sawirka 2.8. Guji meesha la calaamadiyey ee falaadha midigta, si aad ugu gudubto sawirka 2.9. **Eclipse** waa diyaar. Waxad sawirka 2.10 ka arkaysaa qaybaha kala duwan ee borogaraamka **Eclipse** ka koobanyahay. Tusaale ahaan markaan ku leeyahay **File->New->Class**, waa inaad kor Menu-da ka doorato **File**, kadibna daaqadda soo baxda ka doorato **New**, ugu dambeyntiina daaqadda soo baxda ka doorato **Class** si aad u hesho sawirka 2.20.



Sawirka 2.3: Lasoo dagista borogaraamka Eclipse talaabada 1-aad



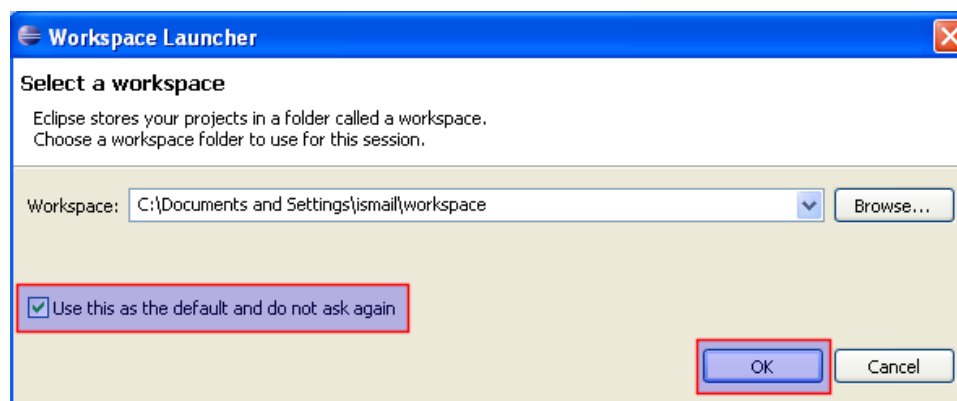
Sawirka 2.4: Lasoo dagista borogaraamka Eclipse talaabada 2-aad



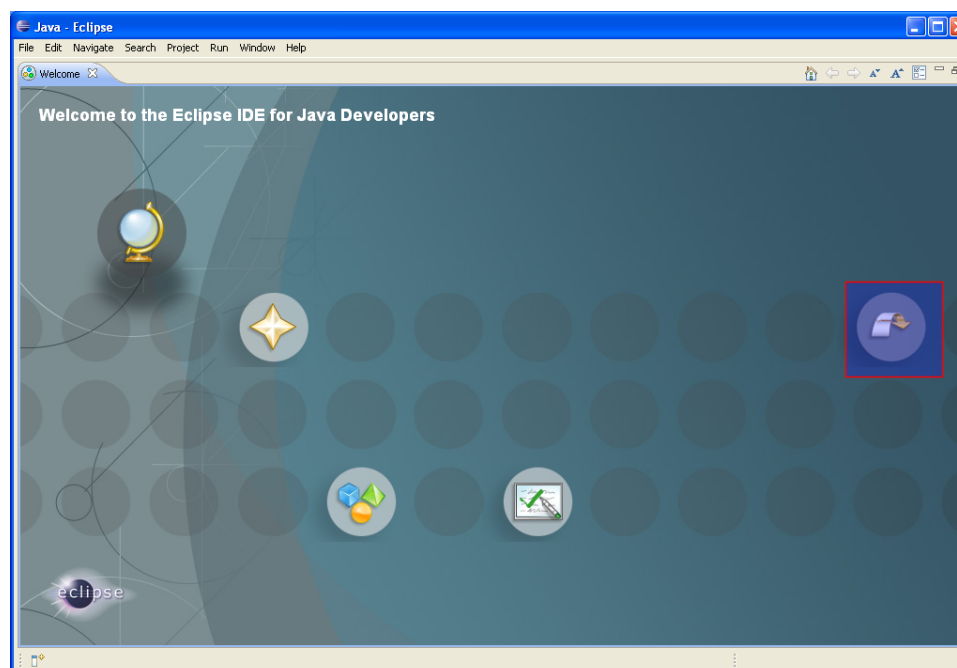
Sawirka 2.5: Shidista ama daarista borogaraamka Eclipse



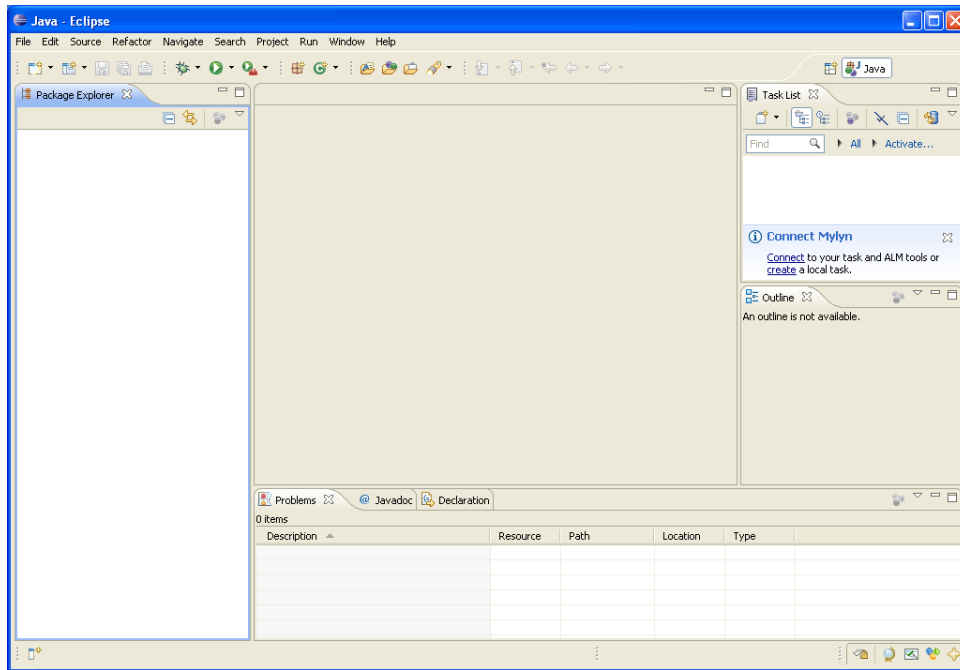
Sawirka 2.6: Lasoo dagista borogaraamka Eclipse talaabada 4-aad



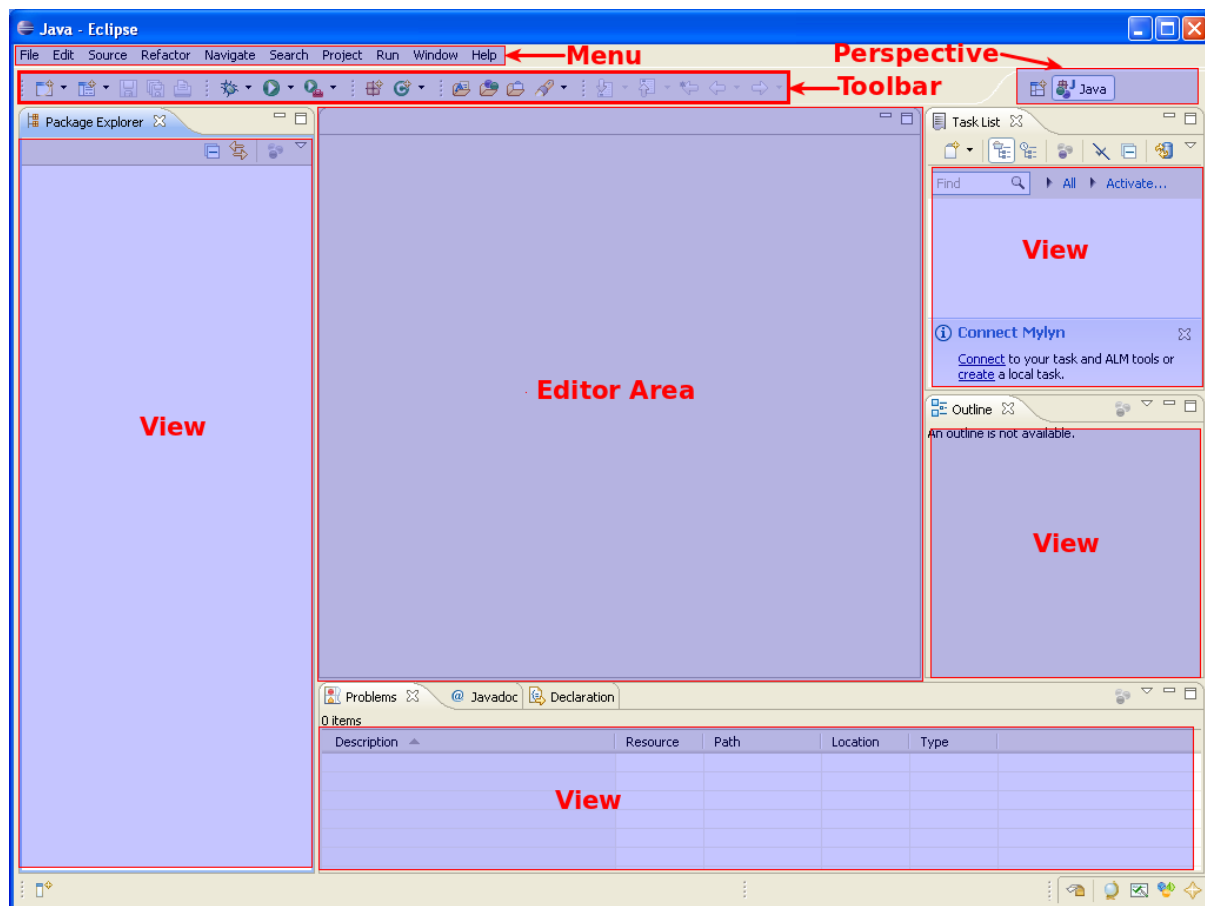
Sawirka 2.7: Lasoo dagista borogaraamka Eclipse talaabada 5-aad



Sawirka 2.8: Lasoo dagista borogaraamka Eclipse talaabada 6-aad



Sawirka 2.9: Lasoo dagista borogaraamka Eclipse talaabada 7-aad



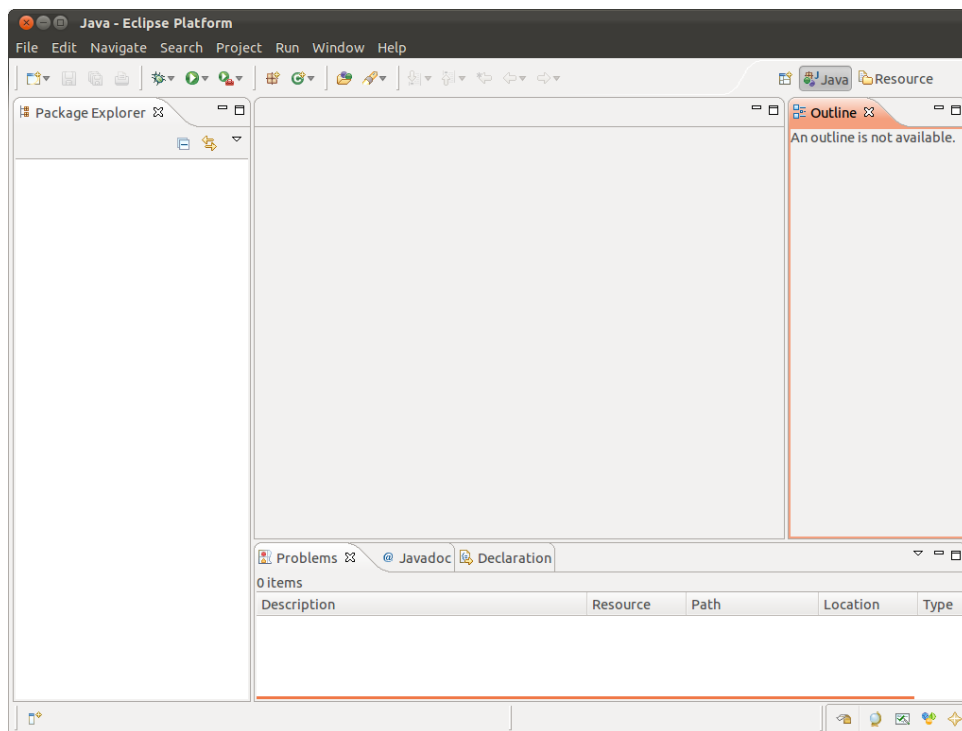
Sawirka 2.10: Qaybaha kala duwan ee borogaraamka Eclipse ka koobanyahay. Halka ugu sareysa waa *Menu*. Waxad ka helayaas talaabooyin badan sida File->New->Java Project, File->New->Class, iwm. Waxa kale oo aad Menu-ga ka helayaas Run->Run As->1 Java Program, Run->Run As->2 JUnit Test, iwm. Menu af-soomaali Dr. Barwaaqo wuxu ku macneeyey *xashin* [1], laakiin anigu waxan adeegsanayaa ereygaas af-ingiriisiga ah. *Perspective*-ku waa inuu kuu taagnaado Java inta badan. *Toolbar*-ka waxad ka helayaas waxyaalo badan oo aad ka helayso Menu-ga. Qaybta bidixda ee *Project Explorer* waxad ka helayaas faylashaada (galalka). Qaybta dhexe waa meeshii wax lagu qoraayey.

Ubuntu

Haddii aad isticmaalayso kumbuyuutar uu ku jiro OS-ka Linux, gaar ahaan **Ubuntu**. Waxad adeegsataa talaabooyinkan hoose:

1. Fur borogaraamka Terminal-ka
2. Waxad ku qortaa `sudo apt-get install eclipse` kadibna **Enter** guji. Sug intuu dhamaynaayo.
3. Intaas kadib waa inaad kumbuyuutarkaaga ha hesho **Eclipse**, isaga oo u eeg sawirka **2.11**.

Markaad isu eegto sawirka **2.9** iyo **2.11**, waxad arkaysaa in waxyar xaga muuqaalka borogaraamku uu is badalay, taasna waxa keentay kala duwanaanshana labada OS. Laakiin borogaraamka **Eclipse** isku si ayuu uga shaqaynayaa labada dhinacba.



Sawirka 2.11: Borogaraamka Eclipse marka lagu isticmaalaayo Linux

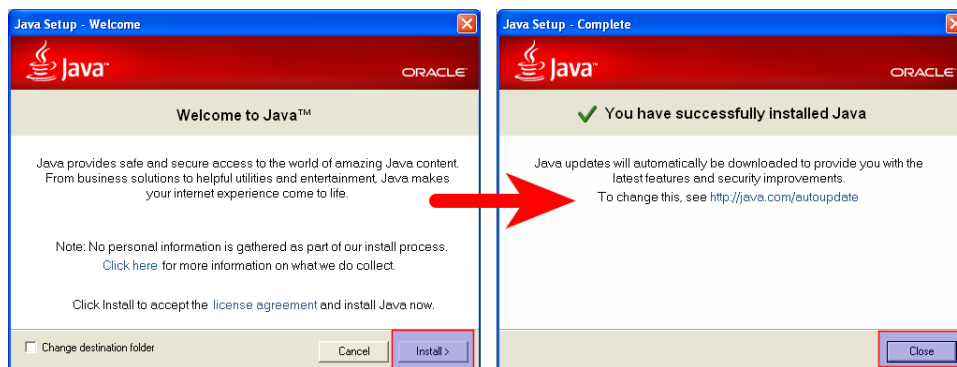
Ku kaydso kumbuyuutarkaaga JRE (Windows)

Borogaraamka **Java Runtime Environment (JRE)** waa lagama maarmaan si aad u kaxayso tusaale ahaan borogaraamka **Eclipse** iyo kuwa kale ee Java lagu qoray. Haddii borogaraamka **Eclipse** markiiba kuu istaadhmo, wuu ku jiraa kumbuyuutarkaaga borogaraamka **JRE**, sidaas awgeed waad ka boodi kartaa talaabooyinkan. Talaabooyinkan raac si aad kumbuyuutarkaaga ugu kaydiso borogaraamka **JRE**.

1. Booqo adarayska <http://java.com/en/download/>
2. Guji badhanka *Free Java Download*. Mar labaad guji *Agree and Start Free Download*
3. Markuu faylku soo dago, labo jeer guji, kadibna dhamaystir installation-ka sida ka muuqata sawirka **2.12**. Intaas kadib waxad diyaar u tahay inaad kaxayso borogaraamka **Eclipse**.

2.2.2 Text Editor, SDK iyo Apache Ant

Haddii aanad adeegsan **IDE**, sida **Eclipse** ama mid u dhiganta sababo jira awgeed waxad adeegsan kartaa borogaraamyadan soo socda oo wada socda.

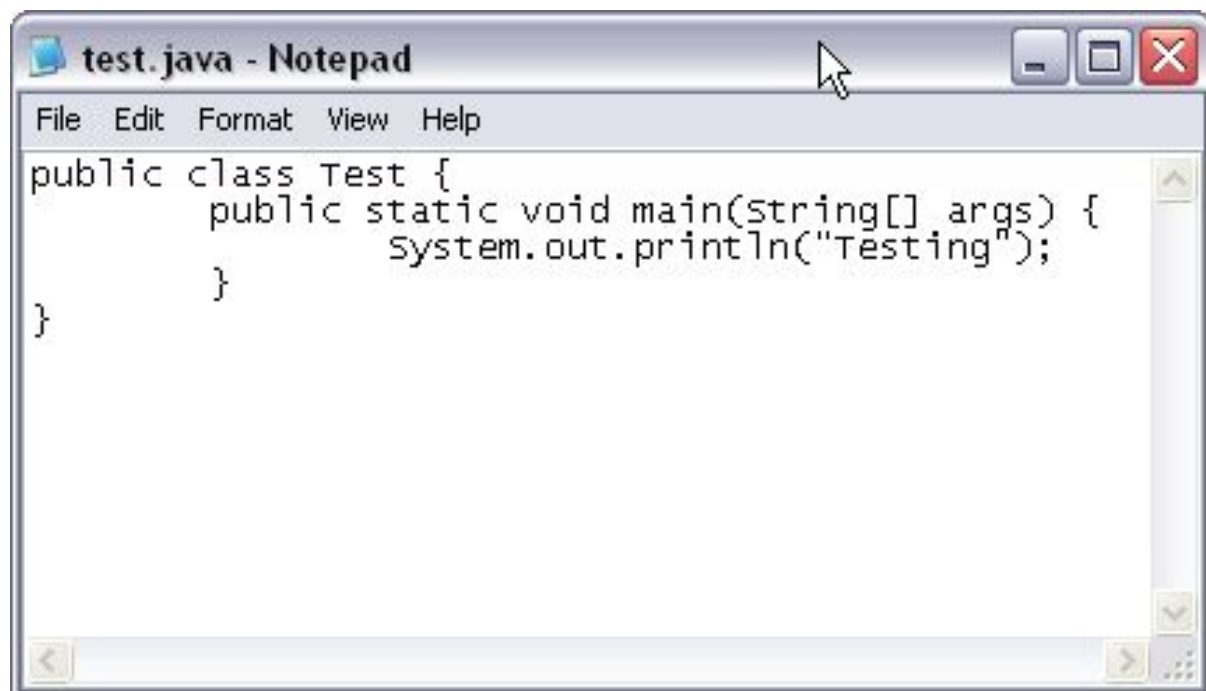


Sawirka 2.12: Ku kaydinta borogaraamka JRE kumbuyuutarkaaga

Text Editor

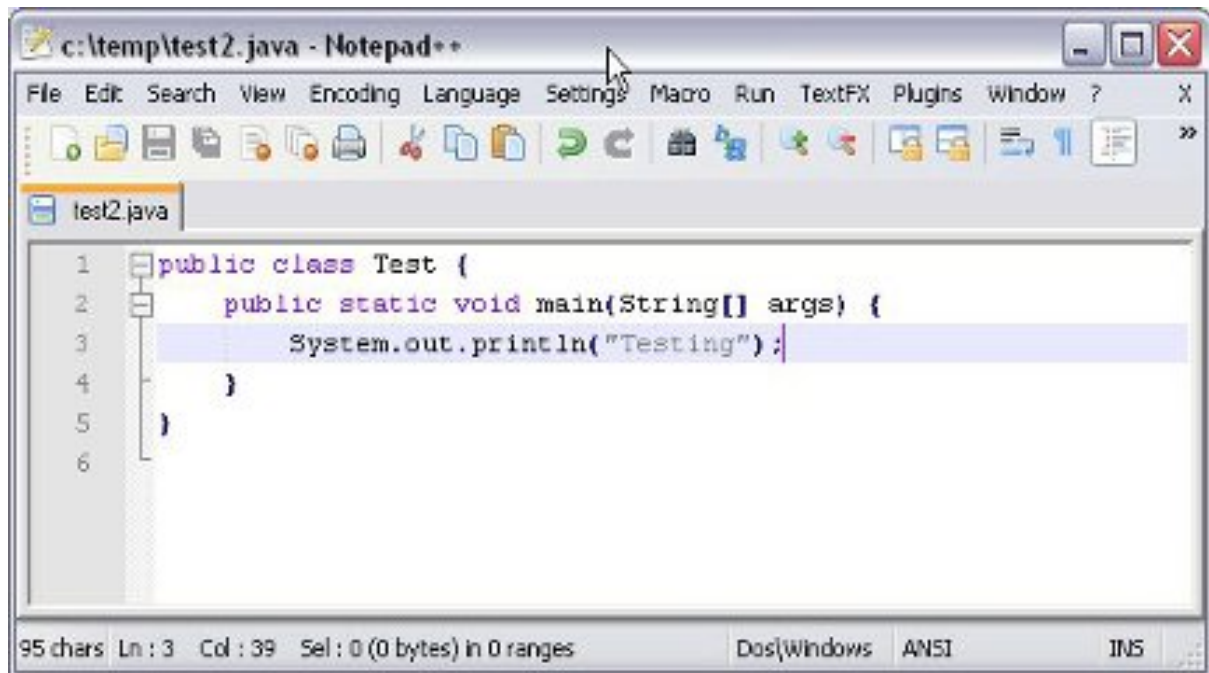
Text Editor waxad u baahanaysaa markaad qorayso borogaraamka. Waxaad adeegsan kartaa borogaraamyadan midkood:

1. Borogaraamka hore waa **Notepad** waxaanad ka arkeysaa sawirka 2.13. Borogaraamkani wuxu la socdaa Windows-ka, markaas looma baahna inaad meel kale ka raadiso. Haddii aad haysato Operating System ka duwan Windows, waxad hubaal ka helayaas system-kaaga borogaraam u dhigma
2. Borogaraamka labaad ee aynu halkan kusoo qaadaneyno waxa magaciisa la yidhaahdaa *Notepad++*, waxaanad ka arkeysaa sawirka 2.14. Borogaraamkani xaga quwadda iyo waxyaalaha laga helaayaba wuu ka sareeyaa borogaraamka hore ee Notepad. Borogaraamkan waxad ka helayaas adarayska: <http://notepad-plus-plus.org/download>. Waxa jira borogaraamyo kale oo noocan oo kale ah sida Textpad, Ultraedit, Slicedit iyo qaar kale. Dhammaantood waxad ka helayaas net-ka
3. Haddii aad adeegsanayso Ubuntu, waxad isticmaali kartaa borogaraamka **GEdit** ee la socda Ubuntu.



Sawirka 2.13: Windows Notepad

Haddii aanad adeegsan IDE oo aad dooratay talaabada labaad waxad u baahantahay oo kale JDK, Apache Ant iyo Command Line. Waxan sharaxaad yar ka bixinayaa borogaraamyadan iyana midkiiba.



Sawirka 2.14: Borogaraamka Notepad++

JDK

Java Development Kit (JDK) waa qalab ka kooban dhammaan waxyaalihii uu programmer-ku u baahnaa, si uu u qoro borogaraamyada Java. Waxa ku jira borogaraamyo muhiim ah oo aad u baahaneyso markaad dhiseyso borogaraamyada Java. Kaliya waxa ka maqan Editor-kii. Tusaale ahaan borogaraamka Eclipse ee aynu kor kaga hadalay wuxu adeegsanayaa ama isticmaalayaa borogaraamyo ka mid ah JDK-da. Waxa ka mid ah compiler-ka Java ee *javac*. Waxa kale oo ka mid ah *JRE* oo ah borogaraamyadii loo baahnaa si loo kaxeeyo borogaraamyada Java. JRE-da borogaraamka ugu muhiimsan waa JVM oo aynu kaga hadalay qaybta 1.2.

Borogaraamka Eclipse laftiisa waxa lagu sameeyay luuqadda Java, sidaas awgeed si loo kaxeeyo waxa loo baahanayaa JRE, sidaan soo sheegayba. JDK iyo JRE waxad ka helaya isku meel. JDK iyo JRE xidhiidhka ka dhexeeya waxad ka arkeysaa sawirka 2.15. Borogaraamka Eclipse ee aynu adeegsaneyno wuxu wataa qalabka JDK, markaas looma baahna ineynu gooni ulasoo dagno. Haddiise aynu adeegsaneyno borogaraam kale sida Notepad++ ee sawirka 2.14 waa lagama maarmaan inaynu lasoo dagno JDK.

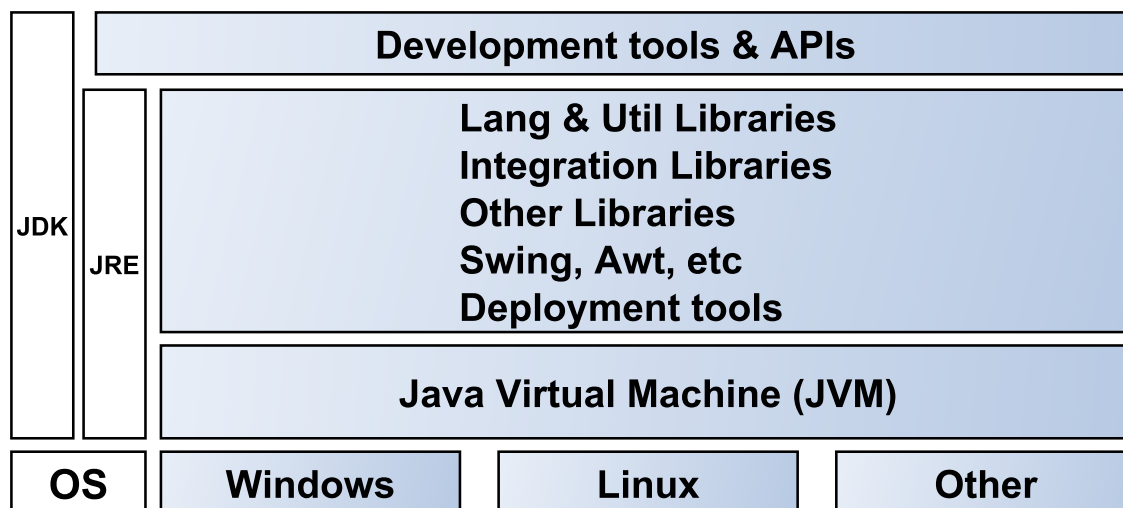
Apache Ant

Sida ka muuqata sawirka 2.1, qorista borogaraamyada luuqadda Java waxay ka koobantahay ugu yaraan 6 talaabo. Waxa loo baahanyahay borogaraam lagu fududeeyo talaabooyinkaas. Programmer-ka waxa looga baahanyahay kaliya inuu qoro source code-ka borogaraamkiisa. *Apache Ant* ayey dadka qora borogaraamyada Java hawshan u adeegsadaan. Haddii aynu eegno luuqadda ingiriisiga, borogaraamkan waxa loo yaqaan *software build tool*. Ant waxad ka helaya adareyska <http://ant.apache.org/bindownload.cgi>. Gaar ahaan wakhtigan aan buugga qoraayo wuxu marayaa soosaarkiisii 1.8.2. Borogaraamkan waxa lagu qaban karaa hawlo badan oo kala duwan, oo kuma koobna kaliya borogaraamyada luuqadda Java. Haddii aanu kumbuyuutarkaaga ku jirin borogaraam lagu furo faylasha noocoodu yahay ZIP, waxad lasoo dagi kartaa 7zip, oo aad ka helayso <http://www.7-zip.org/>.

Apache Ant waxa laga maamulaa inta badan fayl magaciisa la yidhaahdo *build.xml*, kaas oo inta badan la socda source code-ka borogaraamka.

Command line

Command line ama *command prompt* (oo loo yaqaano *shell* ama *terminal* marka laga hadlaayo aduunka Linux) waa borogaraam la socda OS-ka, gaar ahaan Windows-ka. Waa borogaraamka laga hago ama amarka laga siiyo OS-ka. Gaar ahaan waxa aad u adeegsada dadka sameeya borogaraamyada kumbuyuutarka. Sawirka 2.16 waxad ka arkeysaa *command line*-ka Windows oo laga helaayo waddada Start->All Programs->Accessories->Command Prompt. Si haddaba aad JDK iyo Apache Ant uga



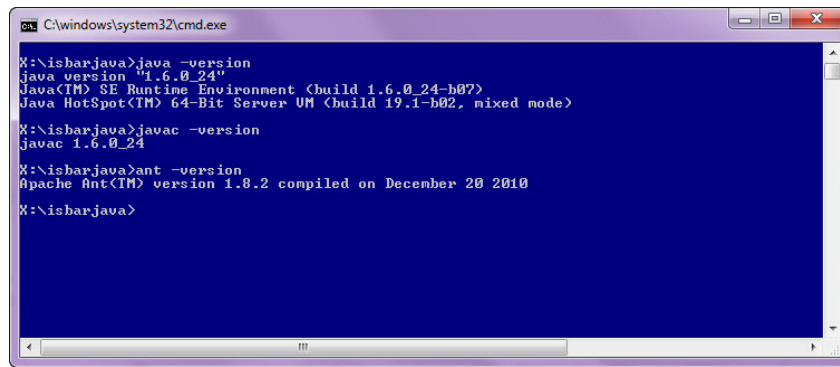
Sawirka 2.15: Java Development Kit (JDK). Sawirka waxad ka arkeysaa waxay ku kala duwanyihiin JRE iyo JDK. Arinta ugu muhiimsan ee aad sawirka ka arkayso waa in JDK ay ka koobantahay JRE iyo agab kale.

isticmaasho command line-ka sida ka muuqata sawirka 2.16, waa inaad *environment variable*-ka system-kaaga ee loo yaqaano `PATH` ku darto meesha ay system-kaaga kaga jiraan borogaraamyadaasi. Sidoo kale waa inaad sameyso *environment variable* cusub oo magaceeda la yidhaahdo `JAVA_HOME`, taas oo sheegaysa meesha ay JDK-da Java kaga kaydsantahay kumbuyuutarkaaga, waxaana adeegsanaaya gaar ahaan borogaraamka Apache Ant. *Environment Variables*-ka Windows waxad ka arkeysaa sawirka 2.17 (waa XP). Waxa kale oo aad sawirkaas ka arkeysaa sida loogu darikaro *variable* cusub iyo sida loo badali karo qiimaha kuwa jira, sida `PATH` oo kale. Talaabooyinka aad qaadeyso waa sidan:

1. OS-ka Windows waxad gashaa **Control Panel**, Kadib raadi **System**, kadibna laba jeer guji
2. Haddii uu kumbuyuutarkaaga ku jiro Windows 7, waxad dhinaca bidixda ka doorataa **Advanced System Settings**, haddii kale toos daaqadda soo boodda uga raadi caleenta **Advanced**. Intaas kadib waa iney kuusoo baxdo daaqadda sawirka 2.17
3. Intaas kadib, sida sawirka ka muuqata waxad gujisaa badhanka **Environment Variables** ee hoos ku yaalla. Kadib guji nambarka saddex (New...) sida ka muuqata sawirka, waxaanad ku qortaa daaqadda soo boodda `JAVA_HOME` (variable name) iyo meesha uu kumbuyuutarkaaga JDK kaga kaydsantahay (variable value). Kumbuyuutarkayga JDK waxay kaga kaydsantahay meesha sawirka ka muuqata
4. Talaabada afraad waa inaad liiska ka raadiso `PATH`, kadibna xagga dambe kaga darto `JAVA_HOME\bin` iyo meesha uu ku kaydsanyahay Apache Ant. Kumbuyuutarkayga waxan ku kaydiyey `C:\tools\apache-ant-1.8.2\bin` (waxa halkan looga baahanyahay folder-ka bin kaliya). Apache Ant waa fayl noociisu yahay `ZIP`, markaas meeshii kumbuyuutarkaaga aad jeceshahay ku furfur, sidaan hore ugasoo hadlay. Laakiin markaad qiimaha `PATH` badaleyso, waa inaad xasuusato inaad u dhaxeysiiso calaamadda ; (oo loo yaqaano *path separator*) qiimeyaasha kala duwan
5. Intaas kadib, waxad command line-ka ku qortaa `java -version`, `javac -version` iyo `ant -version` sida ka muuqata sawirka 2.16. Waa inuu kuusoo baxo qoraalka hoose (kaliya waxa isbadali kara nambarka oo loo yaqaano *software version*).

2.3 Borogaraamka - Hello World!

Salaan Aduunka (Hello World)! ama Waan ku salaamey aduunka! waa borogaraamka ugu fudud ee Java ee inta badan loo soo qaato tusaalaha ugu horeeya ee borogaraamyada lagu qoray Java. Maadaama ay caado noqotay ka leexanmayno wadadaas innaguna. Waxaynu halkan kusoo qaadaneyna borogaraamkaas yar innaga oo adeegsanayna borogaraamka **Eclipse**. Waxad raacdaa talaabooyinkan:



Sawirka 2.16: Command line-ka Windows

1. Shid borogaraamka Eclipse
2. Guji File->New->Java Project si aad u samayso project cusub sida ka muuqata sawirka 2.18. Waxad u bixisaa helloworld, kadibna guji Finish. Project-ka shaqadiisu waa inuu isu uruuriyo faylashaada uu ka koobanyahay borogaraamka aad samaynayso (NewProject)
3. Waa inaad hesho project cusub, sida ka muuqata sawirka 2.19
4. Samee nooc cusub adiga oo raacaya File->New->Class, waa inay kuusoo baxdo daaqadda sawirka 2.20. Waxad u bixisaa HelloWorld, waxaanad calaamadisaa meesha ay ku qorantahay **public static void** main(String[] args) sida ka muuqata isla sawirka. Kadib guji Finish. U eekaysii borogaraamka tusaalaha 2.1 (NewClass)
5. Intaas kadib, kaxee (run) borogaraamka, adiga oo dooranaaya Run->Run As->1 Java Application (RunJava). Waa inaad fariintii borogaraamka ka aragto qaybta Console-ka, sida ka muuqata sawirka 2.21.

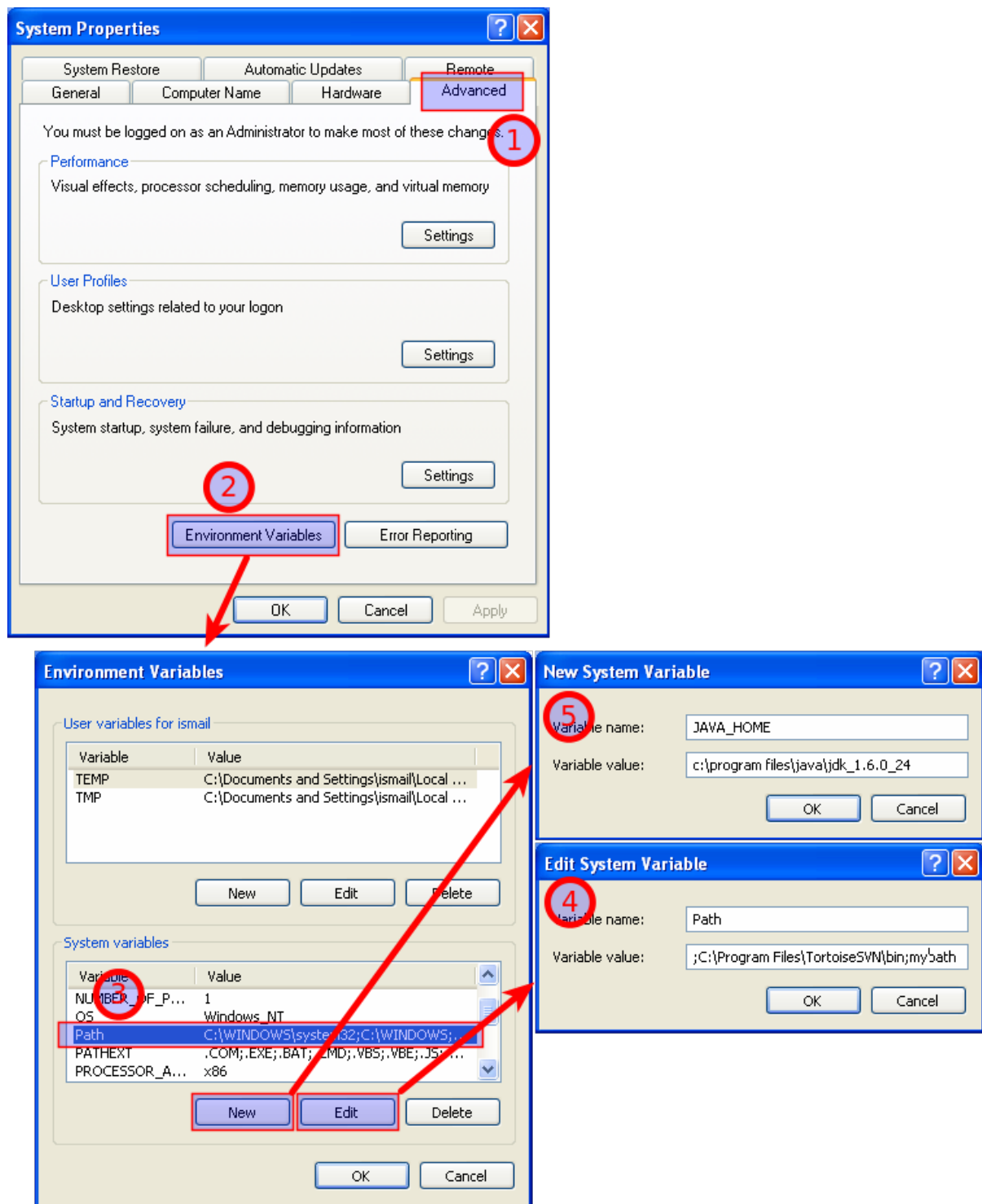
```

1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Waan ku salaamay aduunka!");
4     }
5 }

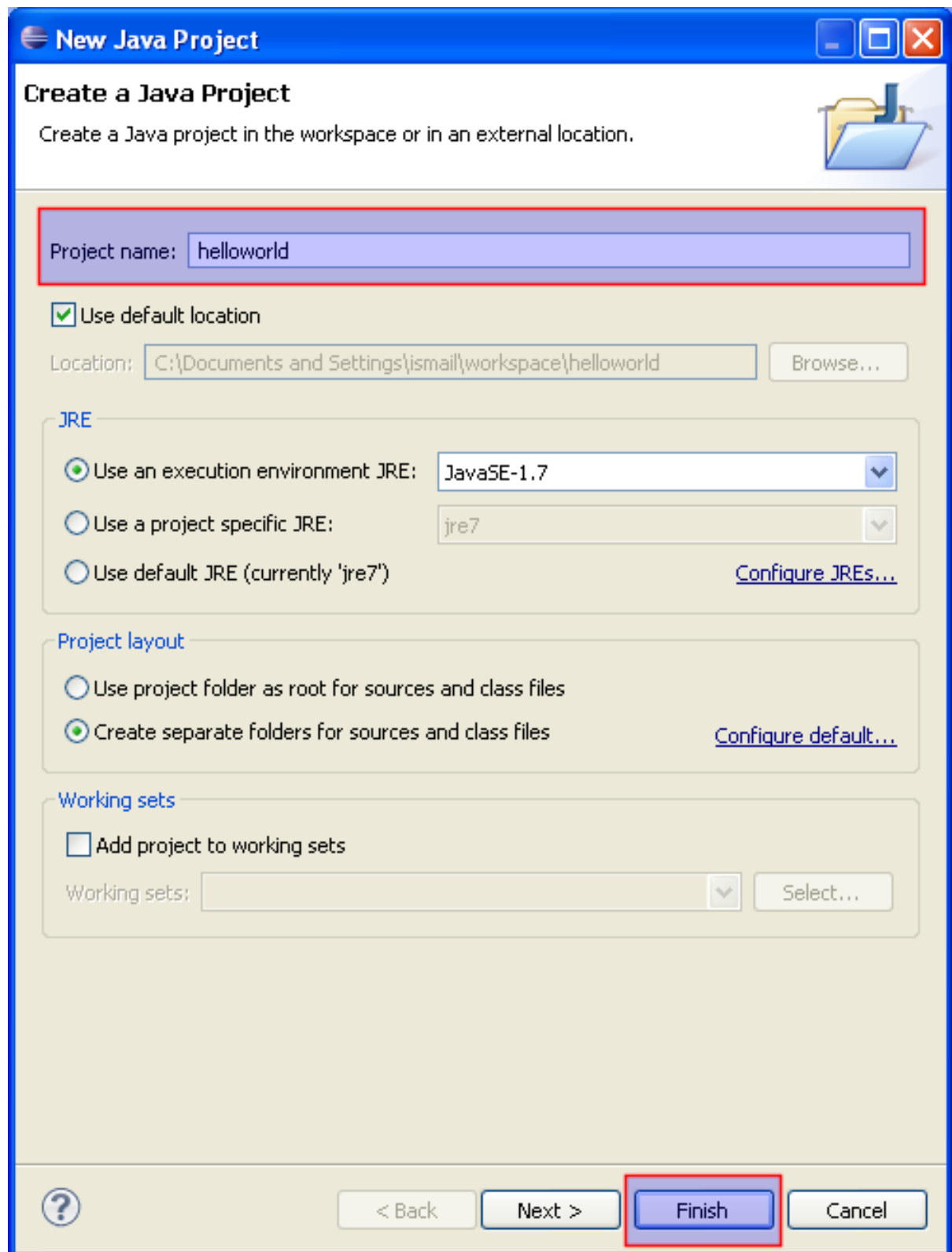
```

Tusaale 2.1: Borogaraamka Salaan Aduunka (Hello World)

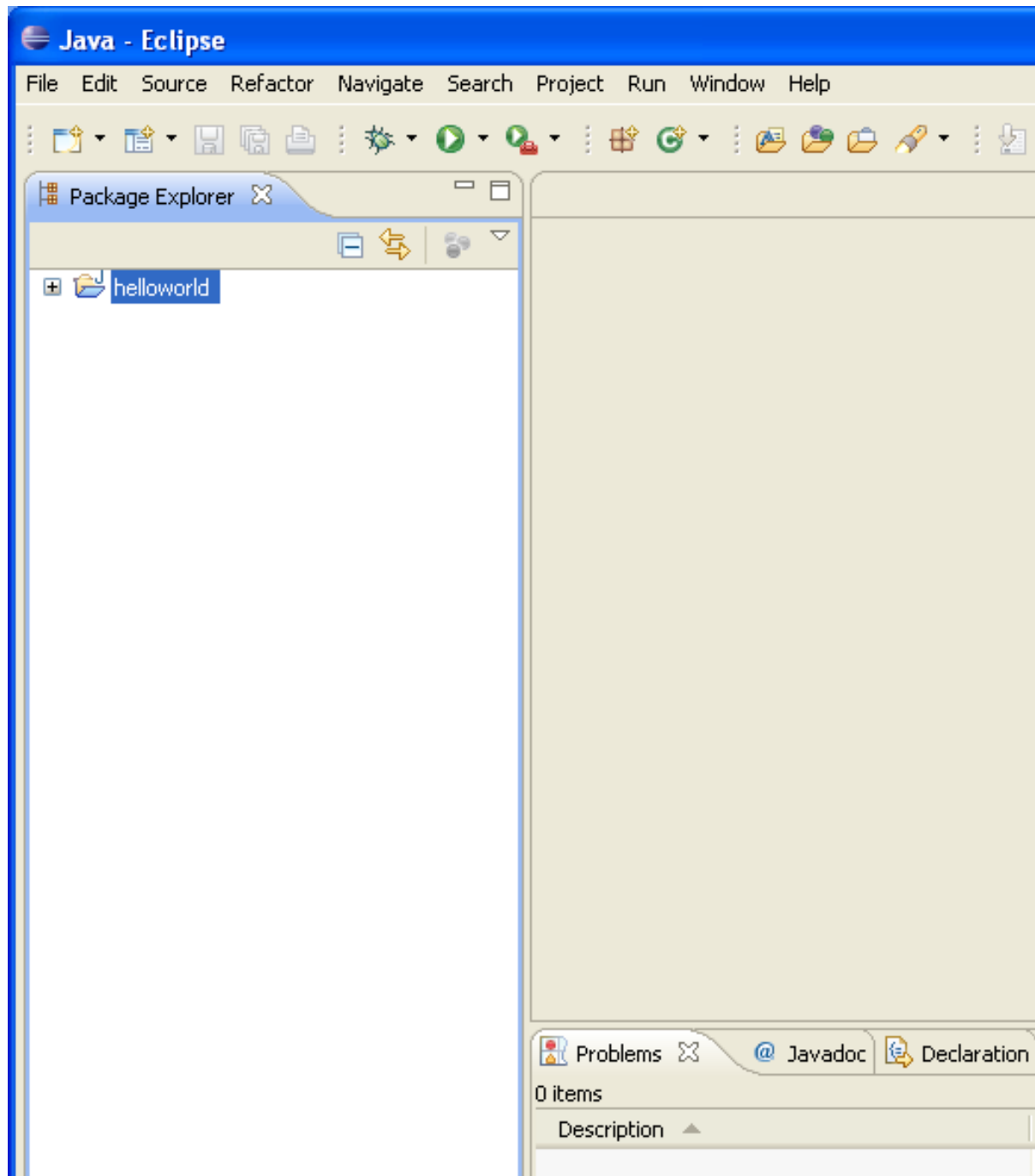
Talaabooyinka NewProject, NewClass iyo RunJava waa talaabooyin guud oo aad u baahanayso mar kasta oo aad rabto inaad bilowdo project cusub, nooc cusub ama aad rabto inaad kaxayso borogaraamka Java. Sidaas daraadeed waxa fiican inaad xasuusato, sababtoo ah waxad ku arkidoontaa meelo badan oo soo socda buuggan. Intaanad ka gudbin qaybtan badal fariinta borogaraamka 2.1 oo ka dhig waxaad rabto tusaale ahaan “Java waan jeclahay inaan barto”, kadibna kaxee oo hubi inay kuusoo baxdo fariinta cusubi. Halkaas waxa ku dhamaatay qaybtii ugu horaysay ee buuggan. Waxaynu u gudbaynaa qaybtii labaad oo aad hoos ugu daadagi doonto luuqadda Java.



Sawirka 2.17: Environment variables-ka Windows iyo sida wax looga badali karo.



Sawirka 2.18: Samaynta Project cusub oo Java ah. Meesha Project name waa qasab inaad buuxiyo, meelaha kale loomabaahna inaad badasho.



Sawirka 2.19: Project cusub oo Java ah samaynta kadib.

New Java Class

Java Class

The use of the default package is discouraged.

Source folder:

Package: (default)

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ public static void main(String[] args)

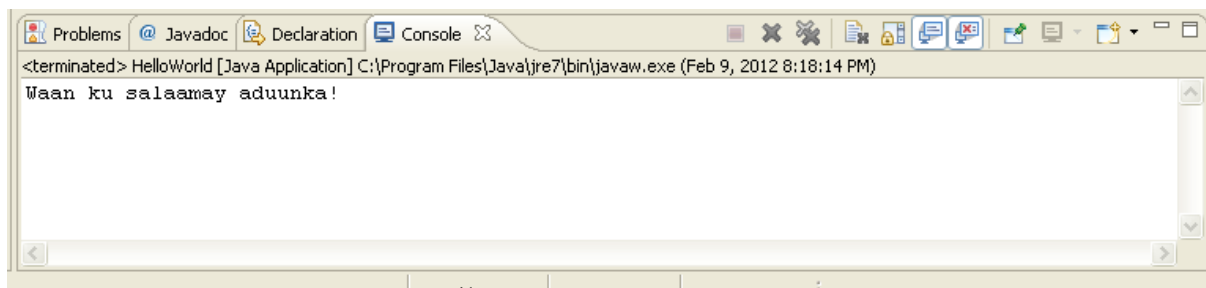
☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Sawirka 2.20: Samaynta nooc cusub oo Java ah. Inta badan meelaha aad buuxinayso waa **Name** iyo **Package**. Meelaha kale sida **Source folder** ama meelaha kale waxa loo baahanyahay inaad badasho marmar iyo dhif.



Sawirka 2.21: Natijada borogaraamka Hello World marka la kexeeyo, waxad ka arkaysaa daaqadda **Console** ee borogaraamka **Eclipse**. Daaqaddaas waxad ka arkaysaa inta badan wixii fariimo ah ee borogaraamkaagu tusaayo isticmaalaha.

QAYBTA II

BARASHADA LUUQADDA JAVA

Qaybtani waa qaybtii labaad ee buuggan. Waxa fiican inteynaan gudagalin barashada luuqadda Java, ineynu waxyar ka sheekayno borogaraamka aynu rabno in aynu dhisno, wuxu yahay iyo caynka uu u eegyahay. Borogaraamka aynu dhisayno waxaynu aasaas uga dhigaynaa sheeko curis ah, laakiin suurto gal noqon karta. Borogaraamkan aynu dhiseyno waxa innaga soo codsadey iskuul ku yaalla magaalo ka mid magaalooyinka geeska Africa. Gaar ahaan maamulaha iskuulkaas oo khaatiyaan ka taagan buugga iyo qalinka uu maalin walba is dabasido. Maamulahaas innaga soo codsaday borogaraamka kumbuyuutarka wuxu noqonayaa *customer*-keenii ama rukunkeenii. Dhibaatooyinka uu buuggu leeyahay way badan yihiin. Marka hore way adagtahay sida *copy* looga sameeyo. Copy-gaas oo la galinkaro meel xafid ah, oo loo baahandoono haddii tusaale ahaan uu buuggu gubto, la xado, xumaado, ama ay ku dhacdo dhibaato kale. Marka xigta way adagtahay sida dhakhso looga raadiyo macluumaadka loo baahdo. Marka saddexaad waa in far fiican lagu qoro si loo fahmo. Intaas iyo dhibaatooyin kale oo badan ayuu buuggu leeyahay. Maamulaha iskuulku wuxu u baahanyahay borogaraam kumbuyuutar oo hawshan u fududeeya, kaas oo uu kula soconkaro ardayda iskuulka, macalimiinta iyo jedwelka koorsooyinka. Arimaha uu maamuluhu ka rabo nidaamkan cusub waxa kamid ah (ugu yaraan):

1. Diiwaangelinta
2. Darajooyinka/buundooyinka
3. Jedwelka koorsooyinka
4. Biilasha ama lacagaha laga qaado ardayda
5. Repoortiyo ama warbixin kala duwan (liiska ardayda, liiska koorsooyinka, iwm)

Haddii tusaale ahaan aynu eegno waxyaala diiwaangelintu ka koobantahay:

1. Diiwaangelinta ardayda cusub
2. Nidaamka ka tuur arday hore
3. Badel macluumaadka arday nidaamka ku kaydsan (tusaale ahaan telefoonkiisa)
4. Koorso cusub diiwaangeli
5. Koorso hore ka tuur nidaamka
6. Iyo waxyaalaha kale ee la xiriira diiwaangelinta

Sida halkan ka muuqata iskuulkan (dugsigan) waxa aasaas u ah koorsooyin. Koorsooyinkaas oo noqon kara tusaale ahaan: xisaab, juquraafi, af-ingiriisi, af-soomaali, iwm. Koorsooyinkaas waxay yeelan kartaa dhawr fadhi sanadkiiba. Fadhiyadu waxay dhici karaan isku mar haddii tusaale ahaan ardaydu badantahay. Fadhi kastaaba wuxu yeelahay taariikh uu bilaabmo, koorso uu la xiriiro, macalin dhiga iyo dhibco laga helaayo marka ardaygu dhameeyo koorsooyinka. Waxaynu si fiican uga hadlaynaa marka aynu gaadho cutubyada soo socda.

Ardaygu markuu dhameeyo tirada koorsooyinka laga rabo, wuxu helayaa shahaado ama diploma, taas oo cadeyneysa inuu kasoo qalinjebiyey iskuulkaas.

Borogaraamkan ama nidaamkan waxaynu u sameyn karnaa wajiyo kala duwan. Sameynta wajiga ama muuqaalka borogaraamka waxaynu kaga hadlaynaa qaybta ugu dambeysa ee buuggan.

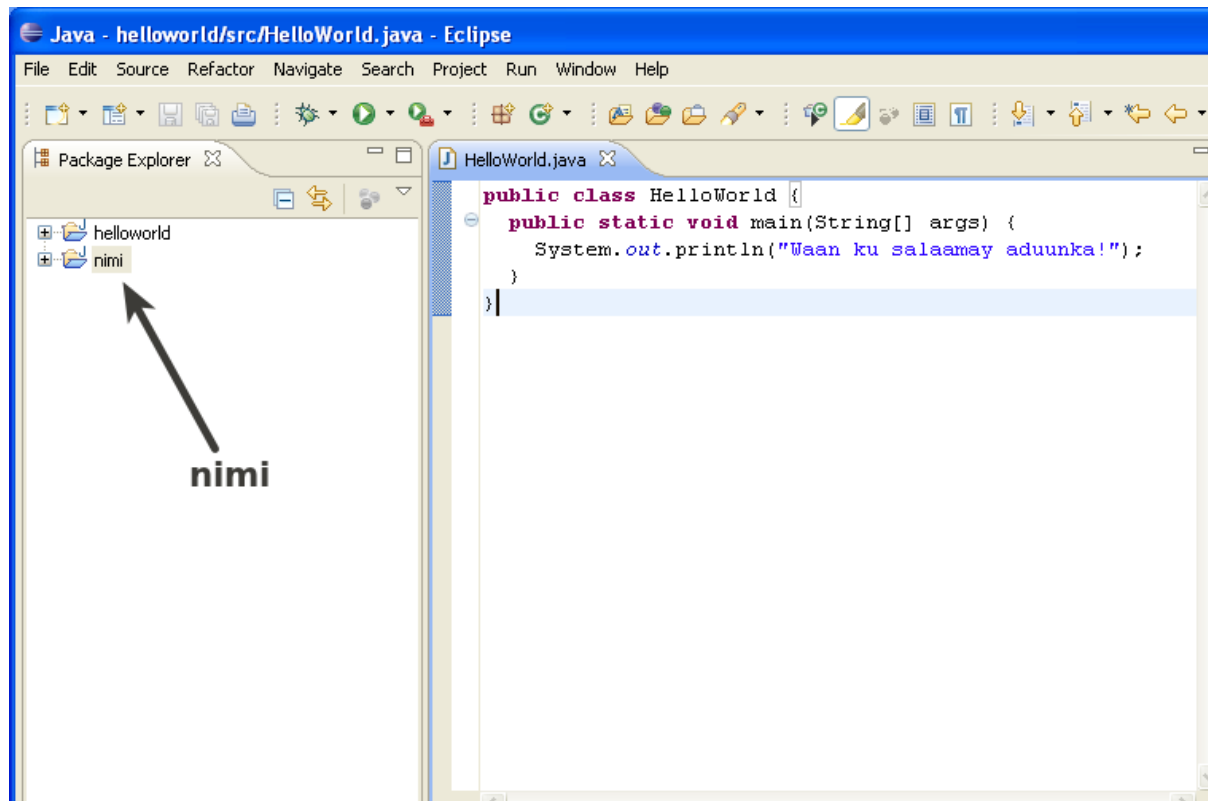
Borogaraamka aynu halkan ku dhisayno ee aynu maamulaha ugu talagalay, waxaan u bixinay **NIMI**, waxaana laga soo gaabiyey ereyada **N**idaamka **M**acluumaadka **I**skuulka. Oo u dhiganta luuqadda af-ingiriisiga **sis** oo laga soo gaabiyey **S**tudent **I**nformation **S**ystem. Waxan fikirka guud kasoo xigtay buugga Jeff Langr ee [7], laakiin aragti, fikir iyo isbadal badan oo aan anigu iska leeyahay ayaan buuggan kusoo bandhigay. Sidaas awgeed haddii ay akhristaha u suurto gasho waxa fiican inuu buuggaas akhristo kan ka dib. Borogaraamkaasi wuxu fulindoonaa baahiyaha maamulaha intooda badan.

Waxaynu barandoonaan sida loo adeegsado talaabooyinka **TDD** ee aynu kaga hadalay qaybta 1.8.

Intaynaan u gudbin cutubyada hoose, waxaynu samaynaynaa talaabo yar oo diyaargarow ah. Waxad Eclipse ku samaysaa Project cusub adiga oo raacaya talaabada **NewProject** ee qaybtii hore. Waxad u bixisaa **nimi**. Intaas kadib, waa inaad hesho xaalad u dhiganta sawirka 2.22.

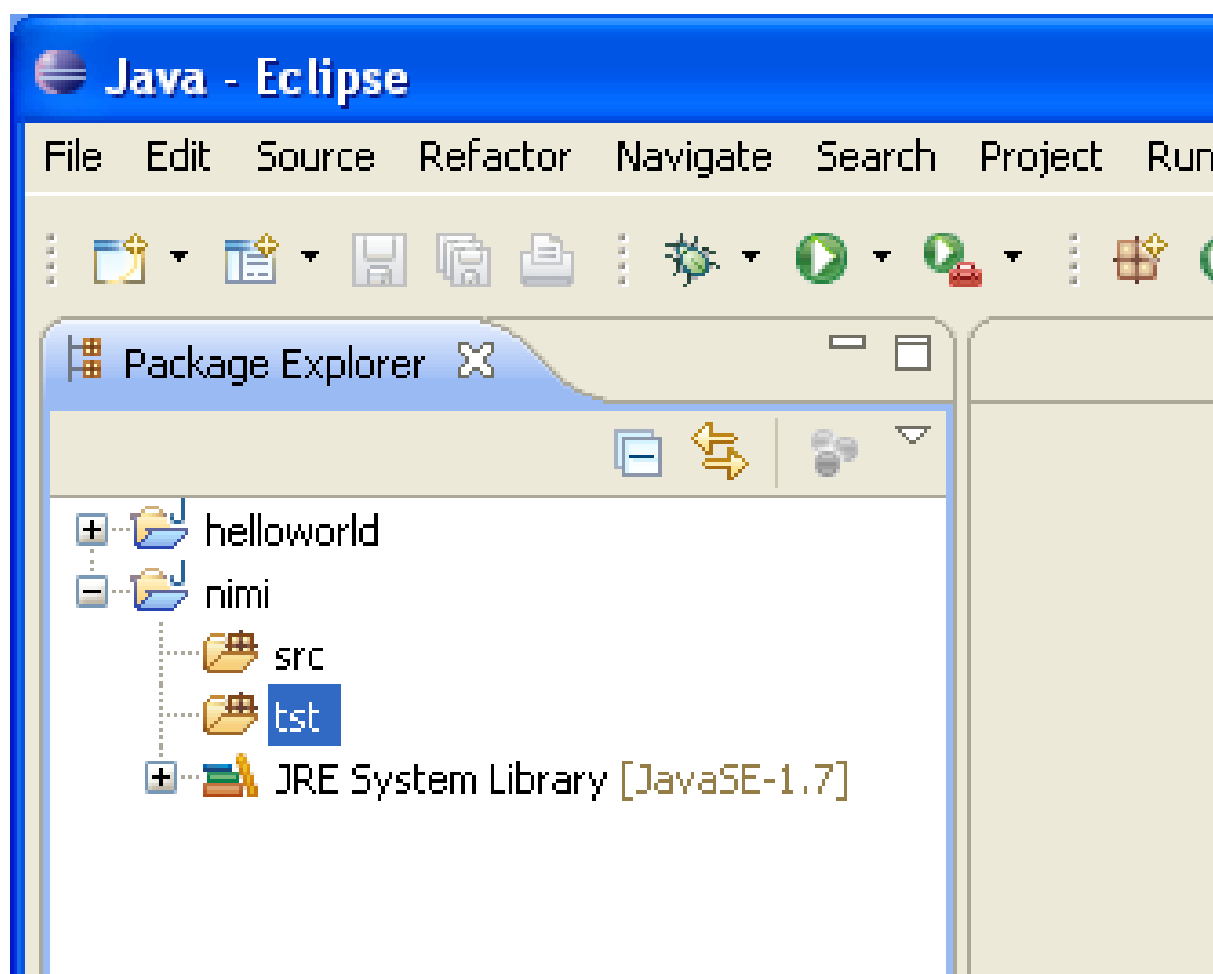
Intaynaan hore u socon waxaynu samaynaynaa hal arin oo yar oo kale. Waxad samaysaa gol cusub oo aad u bixinayso **tst** oo laga soo gaabiyey **test**, adiga oo raacaya talaabooyinka:

1. Dooro Project-ka **nimi**
2. Waxad gujisaa **File->New->Source Folder**



Sawirka 2.22: Samaynta Project-ka **nimi** kadib. Haddii aanad tuurin Project-kii hore ee **helloworld**, Eclipse markaad furto waa inuu u ekaado sidan. Halkan laga bilaabo, waxaynu isticmaaleynaa project-ka **nimi**, markaa waa inaad isaga doorato.

3. Halkay ku qorantahay Folder name, waxad ku qortaa **tst**
4. Kadibna waxad gujisaa **Finish**. Waa in Project-ka **nimi** u ekaado sawirka **2.23**, marka aad kala bixiso (expand). Intaas kadib waxad diyaar u tahay inaad u gudubto cutubka soo socda.



Sawirka 2.23: Folder-ka `src` waxa lagu kaydinayaa qoraalka borogaraamka (source code), folder-ka `tst` waxa lagu kaydinayaa qoraalka borogaraamka tijaabiyaha ah (test code) sidaad arki doonto

Cutub 3

Qofka ama Shakhsiga

Arimaha ugu muhiimsan ee laga rabo nidaamkan waxa ka mid ah in dadka iskuulka lagu maamulo, oo lagula socdo marxaladaha kala duwan ee qof kastaaba maraayo. Dadka waxan halkan ula jeedaa macalimiinta, ardayda iyo shaqaalaha kale ee iskuulka. Marka tirada dadku badato, waxa adkaaneysa sidii buugga iyo qalinka loogu maamuli lahaa. Arintaasi waxay kamid tahay dhibaatooyinka maamulaha iskuulka haysata, ee keentay inuu borogaraam kumbuyuutar xal ka raadiyo.

3.1 Macluumaadka Shakhsiga

Macluumaadka ama sifooyinka qof kasta looga baahanyahay in laga kaydiyo waxa ka mid ah:

1. ID-da qofka (tusaale ahaan: 55, 89, 144, ...)
2. Magaca qofka (sida: Nuur Jaamac, Maxamud Cabdi, ...)
3. Telefoonka qofka (sida: 123456, 456789, ...)

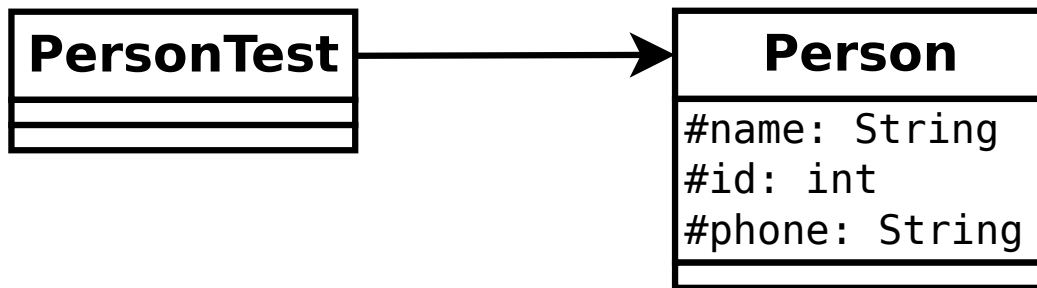
Saddexdaas arimood waa inta aynu marka hore ku bilaabayno. Haddhow waxa soo kordhi kara macluumaad dheeraad ah oo loo baahankaro in nidaamka lagu daro. Tusaale ahaan naanaysta qofka, meesha uu daganyahay qofku, da'da qofka, iyo waxyaale kale oo badan. Si aynu u fududayno tusaalooyinka waxaynu halkan kuusoo qaadaneysnaa kaliya saddexdaas arimood.

Marka qofka laga tuuro ama laga masaxo nimaadka, waa in la tuuro dhammaan macluumaadka qofka laga hayo. Sidaasna lagu ilaawo qofkaas. Haddii iskuulku rabo in aan macluumaadka qofka laga tuurin nidaamka, laakiin kaliya la calaamadiyo in qofkaasi ka baxay iskuulka waa in lagu daro sifooyinka qofka. Taasi waxay saamaxaysaa in wakhti dambe la ogaado qofkii iskuulka dhiganjirtay ama ka shaqaynjiray iyo wakhtigii uu ka baxay. Wakhtiga (uu qofku iskuulka soo galay ama ka baxay) laftiisu wuxu u baahanyahay sifo cusub.

Mar kale, si aynu u fududayno borogaraamka aynu dhisayno waxaynu macluumaadka qofka ku koobaynaa kaliya saddexdaas qodob ee aynu kor ku xusnay. Marka qofka laga tuuro nidaamka, dhammaan wixii macluumaad laga kaydiyey qofkaas waynu tuuraynaa.

Waxaynu ku bilaabaynaa nooc cusub oo matalaaya qofka ama shakhsiga (person), kaas oo aad ka arkayso sawirka [3.1](#). Laakiin intaynaan noocaas qoristiisa bilaabin, waa ineynu qorno noocii tijaabin lahaa, maadaama aynu adeegsanayno **TDD**. **TDD** waxa lagusoo koobay hal layn “Only ever write code to fix a failing test” [6]. Taas oo ka dhigan in aad marka hore qorto tijaabada, kadibna qoraalka borogaraamka. Tijaabadu marka hore way guuldaraysanaysaa oo waxay keenaysaa casaan/guduud marka la kaxeeyo, sababtoo ah borogaraamkii ay tijaabinaysay wali ma jiro. Kadib markaad borogaraamka qorto waa inay tijaabadu cagaar noqoto, taas oo ku tusaysa in borogaraamku sidii aad ka rabtay u shaqaynaayo. Aan ka socdo. Magaca noocan tijaabiyaha ah waxaynu ka dhigaynaa `PersonTest` sida ka muuqata isla sawirka [3.1](#). Bilaabista nooc cusub oo tijaabiye ah waxyar ayey ka duwantahay nooca caadiga ah. Waxad raacdaa talaabooyinkan si aad u bilawdo qorista nooc cusub oo tijaabiye ah. Waxaynu talaabooyinkan u bixinaynaa `NewTest`, si aynu marka dambe ee aynu u baahanno u tixraacno:

1. Waxad doorataa `File->New->JUnit Test Case`. Buuxbuuxi meelaha `Source folder`, `Package` iyo `Name` sida ka muuqata sawirka [3.2](#)
2. Kadibna guji `Finish`. Maadaama ay tahay markii ugu horeysa ee aad noocan oo kale samayso, waxa soo boodaya daaqadda [3.3](#) oo kuleh waxad u baahantahay JUnit. Guji `OK`. Intaas kadib qoraalka noocan tijaabiyaha ah u ekaysii sida borogaraamka [3.1](#).



Sawirka 3.1: Nooca Person iyo tijaabiyihiisa PersonTest

```

1 package nimi.person;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class PersonTest {
8     @Test
9     public void testCreatePerson() throws Exception {
10         Person person1 = new Person(1, "Jaamac Cilmi");
11         assertEquals("Jaamac Cilmi", person1.getName());
12         assertEquals(1, person1.getId());
13
14         Person person2 = new Person(2, "Sahra Ahmed");
15         assertEquals("Sahra Ahmed", person2.getName());
16         assertEquals(2, person2.getId());
17     }
18 }
  
```

Tusaale 3.1: Nooca PersonTest siduu ku bilaabmay

Intaynaan hore u socon aynu sharaxaad kooban ka bixino borogaraamka 3.1 ee nooca PersonTest. Faahfaahin dheeraad ah waxaynu ku baranaynaa cutubyada soo socda. Haddii aan soo koobo, borogaraamkaasi wuxu samaynayaa laba walxood oo Person ah, kuwaas oo uu midkoodba siinaayo magac iyo ID, kadibna wuxu hubinayaa magacooda iyo ID-dooda marka la waydiiyo inuu yahay qiimihii laga sugaayey. Aan ku xigsiiyo sharaxaad layn layn ah:

1. Laymanka 9-17 waa ficil magaciisa la yidhaahdo testCreatePerson, kaas oo loogu talagalay in lagu tijaabiyo samaynta walax (ama walxo) cusub oo Person noocoodu yahay
2. Laynka 10 waxaynu ku samaynaynaa walax cusub oo Person ah, kaas oo ID-diisu tahay 1, magaciisuna yahay “Jaamac Cilmi”. Walaxdaa waxaynu u bixinaynaa person1
3. Laynka 11 waxaynu hubinaynaa in marka walaxda person1 la waydiiyo magaceeda inay soo celi-nayso “Jaamac Cilmi” oo ah magacii aynu siinay. Waydiinta magaceeda waxaynu u adeegsanayaa ficilkeeda getName. Hubinta waxaynu u adeegsanay ficilka assertEquals ee JUnit
4. Laynka 12 waxaynu hubinaynaa in marka walaxda person1 la waydiiyo ID-deeda iney soo celinayso 1 oo ah ID-dii aynu siinay. Waydiinta ID-deeda waxaynu u adeegsanaynaa ficilkeeda getId
5. Laymanka 14-16 waxay la midyihiin laymanka 10-12, waxay samaynayaan walax kale oo la yidhaahdo person2, taas oo magaca “Sahra Ahmed” iyo ID-da 2 la siiyey
6. Calaamadda @Test ee aad ku arkayso laynka 8-aad, waa calaamad lagu calaamadinaayo ficil kasta oo tijaabiye ah. Si aad calaamadaas u adeegsato waa inaad qorto laynka 5-aad. Sidoo kale si aad

New JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder: Browse...

Package: Browse...

Name:

Superclass: Browse...

Which method stubs would you like to create?

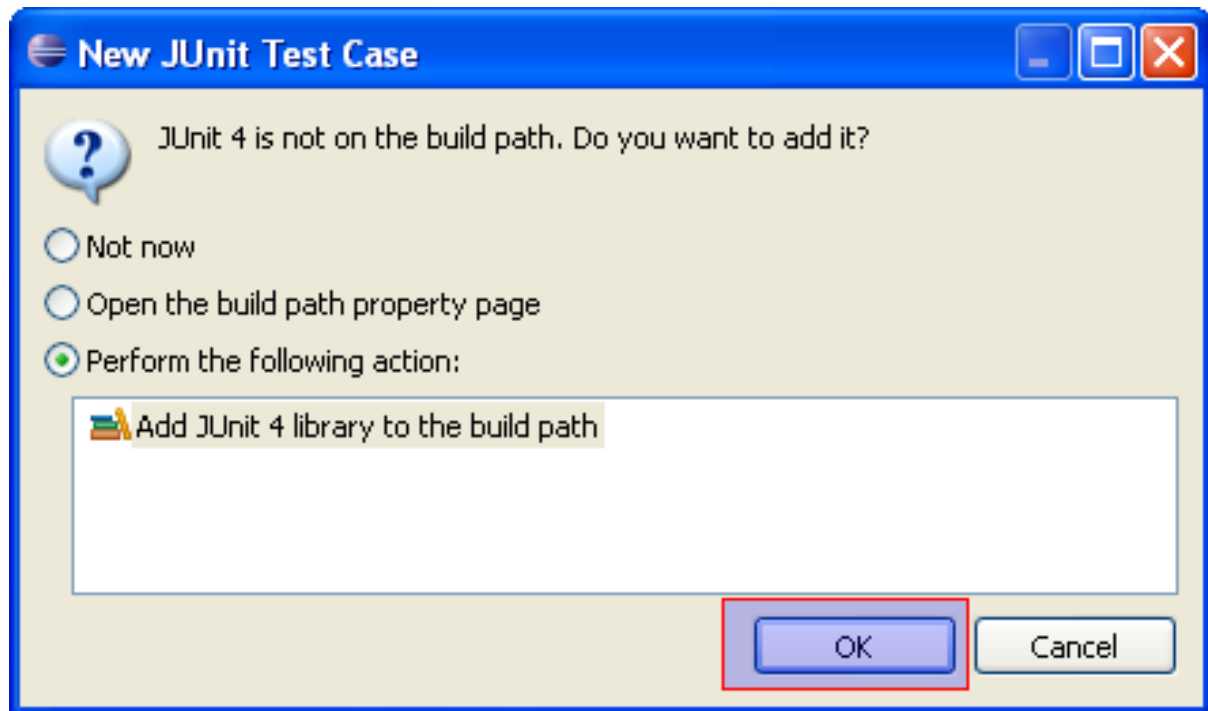
☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test: Browse...

Sawirka 3.2: Bilaabista nooc cusub oo tijaabiye ah



Sawirka 3.3: JUnit ayaad u baahantahay si aad u qorto qoraalka tijaabiyaha ah. Sidaad sawirka ka arkayso JUnit 4 ayaynu adeegsanaynaa. JUnit 4 wuxu la socdaa Eclipse, sidaas awgeed looma baahna inaad gooni ulasoo dagto.

u adeegsato ficilka `assertEquals` ee aad meelo badan oo caloosha ficilka `testCreatePerson` ku arkayso, waa inaad qorto laynka 3-aad

7. Laynka 1-aad waxa loo yaqaan *package declaration*. Waa qasab isna in aad qorto. Qodobadani waxay ahaayeen dulmar kooban, waxaynu arimahan si fiican ugaga hadlaynaa cutubyada soo socda ee haka walwalin imika, kaliya sidaas u qor.

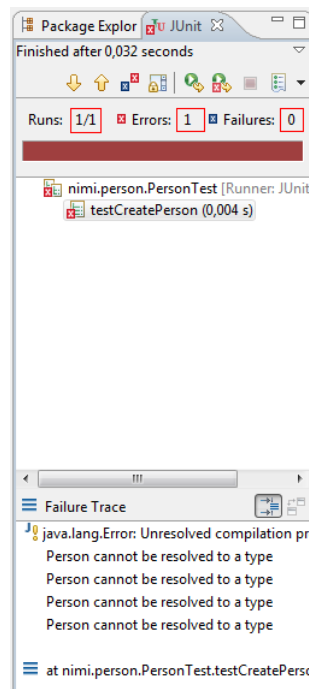
Ereyga `assertEquals` ee lagu isticmaalay laymanka 11,12,15 iyo 16 ayaa u baahan in waxyar laga taabto intaynaan hore u socon. Sawirka 3.4 ayaad ka arkaysaa qaabka uu u yaallo.

Expected	Comma	Actual
<code>Jaamac Cilmi</code>	↓	<code>person1.getName()</code>

Sawirka 3.4: Ficilka `assertEquals` wuxu isu eegayaa qiimaha bidixda ee Expected (qiimaha la sugaayey) iyo kan midigta ee Actual (qiimaha soo baxay). Halkan waxay isu eegaysaa qiimaha "Jaamac Cilmi" iyo wixii uu ficilka `getName()` ee walaxda `person1` soo celiyo.

Sideedaba `assertEquals` (iyo dhammaan ficilada ka bilaabma `assert*`) waa ficil ka imanaaya JUnit oo uu keenaayo laynka 3-aad. Waxay ka midyihiin nooca `org.junit.Assert`. Luuqadda Java waxay kuu saamaxaysaa inaad sifooyinka iyo ficilada `static`-ka ah aad oo `import`-garayso adiga oo adeegsanaaya ereyga `import static`. Sidaas awgeed lamanka 3 iyo 5 tijaabo kasta oo aynu qorno waad ku arkaysaa. Boroga-raamka 3.1 waxad ku arkaysaa ereyo badan oo luuqadda Java u gaar, oo macno gooniya leh. Cutubyada soo socda ayaynu sharaxaad kaga bixinaynaa ereyadaas. Aan ka socdo.

Ficilka `assertEquals` waxaynu u isticmaaleynaa inaynu laba qiime isugu eegno. Labadaas qiime oo la rabo iney isku mid noqdaan. Haddii taasi dhiciwaydo oo ay kala duwanaadaan, waxa imanaysa in tijaabadii guuldaraysato oo casaan noqoto marka la kaxeeyo.



Sawirka 3.5: Tijaabadii ugu horeysay oo fail-garowday ama guuldaraysatay. Sawirka xaga sare, meelaha aan calaamadiyey waxad ka arkaysaa tirakoob, inta tijaabo ee la kaxeeyey, inta cilad daraadeed u guuldaraysatay iyo inta guuldaraysatay (sababtoo ah borogaraamka ayaan u shaqayn sidii laga sugaayey)

Intaas kadib, kaxee (run) borogaraamka 3.1 adiga oo gujinaaya **Run->Run As->JUnit Test** (talaabadan magaceedu waa **RunTest**) ama isku qabo badhanada (**Shift+Alt+X T**). Eclipse wuxu ku tusayaa daaqad yar oo uu leeyahay cilado ayaa borogaraamka ku jira, haka walwalin ee guji **Proceed**. Intaas kadib, waa iney kuusoo baxdo daaqadda sawirka 3.5, kaas oo aad ka arkeyso in hal tijaabo la kaxeeyey **testCreatePerson**, halkiina cilad ku timid awgeed ay guuldaraysatay. Laynka cas wuxu ku tusayaa in tijaabadaasi guuldaraysatay. Isla sawirkaas hoos meesha ay ku qorantahay “**Failure Trace**” waxad ka arkaysaa sharaxaad ku saabsan sababta ay tijaabadu u guuldaraysatay.

Waxaynu qornay tijaabadii ugu horaysay ee guuldaraysata oo casaan keenta marka la kexeeyo. Waxaynu buuxinnay talaabadii ugu horaysay ee **TDD**, laakiin waxa maqan borogaraamkii ay tijaabineysay ee la intifaacsan lahaa. Waxaynu qoraynaa borogaraamkii ugu horeeyey ee aan tijaabiyaha ahayn, taas oo ah talaabadii labaad ee **TDD** (sidaad ka arkayso sawirka 1.5). Borogaraamka Eclipse wuxu hoosta xariijin cas ka marinayaa magaca **Person** ee laymanka 10 iyo 14 ku jira ee tusaalaha 3.1, isaga oo ku tusaaya qoraalkan, markaad jiirka dulgayso magaca **Person** (waxa kale oo aad ka arkaysaa sawirka 3.5 meesha hoose):

Person cannot be resolved to a type

Taas oo macnaheedu tahay inaanu Eclipse garanaynin nooca la yidhaahdo **Person**. Waxaynu rabnaa ineynu ciladdaas saxno, si layn cagaar (doog) ah inoogusoo baxo, markaynu tijaabada kaxayno. Laakiin, intaynaan bilaabin arintaas aan hal arin carabka ku dhufto. Markii aynu qoraynay borogaraamka 3.1, waxaynu ku fikirnay inuu jiro nooc layidhaahdo **Person**, kaas oo magac iyo ID la siinkaro, marka loo baahdana la waydiinkaro. Arintaasi waxay innagu kalifaysaa inaynu ka fikirno noocyada aynu u baahanahay caynka loo adeegsanaayo ama ay u shaqaynayaan, iyaga oo aan wali *jirin*. Warintan [5] wuxu ku magacaabay *programming by intention*. Waxad sameysaa nooc cusub adiga oo raacaya talaabada **NewClass**, adiga oo u buuxuuxinaaya sidan: Source folder=**nimi/src**, Package=**nimi.person**, Name=**Person**. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo **Person**, laakiin madhan.

Haddii aad ku laabato nooca **PersonTest**, Eclipse wuxu hoosta ka xariiqayaa **new Person(1, "Jaamac Cilmi")** iyo **new Person(2, "Sahra Ahmed")** ee laymanka 10 iyo 14, isaga oo leh markaad jiirka dulgayso:

The constructor Person(int, String) is undefined

Constructor waxay tahay dib ayaynu ka sharaxaynaa, laakiin imika si aad u saxdo ciladdan u eekaysii nooca **Person** sida tusaalaha 3.2.

```
package nimi.person;
```

```

1 public class Person {
4     private int id;
6     private String name;

8     public Person(int id, String name) {
        this.id = id;
10        this.name = name;
12    }
}

```

Tusaale 3.2: Nooca Person bilawgiisa

Wali borogaraamka Eclipse wuxu ka cabanayaa ficilada getName iyo getId ee lagu isticmaalay nooca PersonTest ee tusaalaha 3.1 ku qoran. Isaga oo leh:

The method getName() is undefined for the type Person
 The method getId() is undefined for the type Person

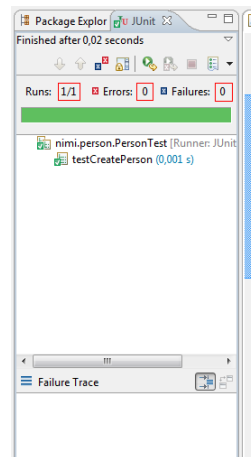
Taas oo uu ula jeedo inaanu nooca Person, lahayn labadaas ficil. Si aynu labadaas ciladood u saxno nooca Person waxad ku dartaa laymanka 7 illaa 13 ee tusaalaha 3.3 (waxad kaga dartaa laymanka 11 iyo 12 dhexdooda). Saddexda dhibcodd (...) ee laynka 5-aad waxa loo yaqaan *ellipsis* luuqadda af-ingiriisiga. Waxan u adeegsanayaa in qoraalka aan cusbayn aan ku qariyo si aanu meesha iiga buuxin. Waxad xasuusanaysaa in dhibcaasi qarinyaan qoraalkii hore ee aan cusbayn ee aynu hore u soo qornay.

```

1 package nimi.person;
2
3
4 public class Person {
5     ...
6
7     public String getName() {
8         return name;
9     }
10
11     public int getId() {
12         return id;
13     }
14 }

```

Tusaale 3.3: Labada ficil ee getName iyo getId ee nooca Person



Sawirka 3.6: Tijaabadii ugu horeysa oo pass-garowday

Intaas kadib mar kale kaxee nooca PersonTest adiga oo raacaya talaabada RunTest. Waa inaad hesho cagaar sida sawirka 3.6. Maadaama aynu cagaar helay, waxaynu ka gudubnay talaabadii labaad ee nidaamka TDD, oo ah Code. Talaabada saddexaad ee Design ama Refactor, waxay sheegaysaa inaad qoraalka nadiifiso oo design-kiisa fiicnaysiiso, adiga oo aan marnaba jabinaynin tijaabooyinka (waa inaanay casaan noqon marna). Maadaama borogaraamka aynu qornay yaryahay uma baahnin imika talaabadaas oo waxa loo baahanayaa marka qoraalka borogaraamku dheeraado. Halkaas waxaynu ku dhamaynay hal wareeg oo nidaamka TDD ah. Markasta oo aynu borogaraamka wax ku kordhinayno waa inaynu saddexdan talaabo marno (ta ugu dambaysa waa laga boodi karaa haddii aanay jirin wax la nadiifiyo). Aan ka socdo.

Intaynaan hore u socon aynu sharaxaad yar ka bixino meesha aynu marayno illaa hadda. Nooca Person isaga oo dhamaystiran waa inuu u eekaado tusaalaha 3.4 xaaladda uu maraayo illaa hadda.

```

1 package nimi.person;
2
3 public class Person {
4     private int id;
5     private String name;
6
7     public Person(int id, String name) {
8         this.id = id;
9         this.name = name;
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public int getId() {
17        return id;
18    }
19 }
20

```

Tusaale 3.4: Nooca Person qoraalkiisa illaa hadda

1. Waxaynu haysanaa laba nooc (class) oo kala ah PersonTest iyo Person. PersonTest waxa loogu tala-galay inuu tijaabiyo nooca Person isticmaalkiisa. Nooca PersonTest qoraalkiisa hadda ka hor ayaynu sharaxaad ka bixinay, laakiin qoraalka nooca Person, wali waxba kamaynaan sheegin
2. Haddii aynu ka bilawno qoraalka 3.4, oo ah xaaladda uu maraayo Person illaa hadda. Laymanka 5 iyo 6 waa sifooyinkii aynu u baahnayn qaarkood ID-dii iyo magacii. Waxad arkaysaa in telefoonkii halkaas ka maqanyahay wali
3. Laymanka 8 illaa 11 waa ficil magaciisu la mid yahay ka nooca Person. Ficil kasta oo la magac ah nooca waxa loo yaqaan *Constructor*. Ujeedada constructor waa iney samayso walax cusub oo noocaas ah. Way soo socotaa sharaxaad ku saabsan
4. Laymanka 13 illaa 15 waa ficilka getName oo loogutalogaley inuu ku siiyo magaca walaxda Person marka laga codsado
5. Laymanka 17 illaa 19 waa ficilka getId oo loogu talagalay in ID-da qofka lagu ogaado. Qofkasta waxa mataleysa hal walax oo nidaamka ku jirta, sidaas awgeed walax kasta waxa la waydiin karaa magaceeda iyo ID-deeda. Labadaas ficil caynka loo adeegsanaayo waxad ka arkaysaa nooca PersonTest.

Waxaynu rabnaa intaynaan hore u socon inaynu nooca Person ku darno telefoonkii qofka sidaynu isku ogeynba, maadaama uu wali ka maqanyahay. Waxaynu ku bilaabaynaa tijaabo cusub oo aynu ku tijaabinayno telefoonka qofka iyo isticmaalkiisa. Waxaynu mar kale ka bilaabaynaa talaabada ugu horaysa ee TDD. Waxad nooca PersonTest dhammaadka kaga dartaa laymanka 9 illaa 15 ee tusaalaha 3.5.

```

1 package nimi.person;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class PersonTest {
8     ...
9     @Test
10    public void testPersonTelephone() throws Exception {
11        Person cigaal = new Person(123, "Cigaal Shiidaad");
12        String phoneNumber = "123098";
13        cigaal.setPhone(phoneNumber);
14        assertEquals(phoneNumber, cigaal.getPhone());
15    }
16 }

```

Tusaale 3.5: Tijaabinta telefoonka qofka

Intaas kadib kaxee nooca `PersonTest` adiga oo raacaya talaabada `RunTest`. Waxad haysataa laba tijaabo oo midkood guulaysanayso. Borogaraamka `Eclipse` wuxu ka cabanayaa laynka 13 iyo 14 (haa tirada laynku way ka duwantahay markuu faylku dhammaystiranyahay) oo markaad jiirka dulgayso wuxu leeyahay:

The method `setPhone(String)` is undefined for the type `Person`

The method `getPhone()` is undefined for the type `Person`

Taas oo macnaheedu tahay in nooca `Person` ay ka maqanyihiin labada ficil ee kala ah `setPhone` iyo `getPhone`. Ta hore ee `setPhone` waxa loogu talagalay in telefoonka qofka lagu badalo, ficilka kale ee `getPhone` waxa loogu talagalay in qofka telefoonkiisa lagu waydiiyo. Si aad labadaas ciladood u saxdo, nooca `Person` kudar laymanka 7 iyo 11-17 ee tusaalaha 3.6.

```

1 package nimi.person;
2
3
4 public class Person {
5
6     private int id;
7     private String name;
8     private String phone;
9
10    ...
11
12    public void setPhone(String phone) {
13        this.phone = phone;
14    }
15
16    public String getPhone() {
17        return phone;
18    }
19 }

```

Tusaale 3.6: Ficilada telefoonka ee nooca `Person`

Intaas kadib haddii aad kaxayso tijaabada `PersonTest`, waa inaad hesho cagaar. Waa war fiican. Waxad haysataa laba tijaabo oo cagaar ama doog ah.

Waxa innaga dhiman in magaca iyo ID-ga qofka la badali karo marka loo baahdo. Tusaale ahaan haddii uu qaldamo magaca ama ID-ga qofku (markaas looma baahna in walax cusub la sameeyo, ee tii jirtay ayaa macluumaadkeeda ama xaaladeeda la badalayaa). Sidoo kale telefoonka waa in la badali karo, marka loo baahdo. Waxaynu u baahannahay tijaabo arintaas ku saabsan. Waxad nooca `PersonTest` ku dartaa laymanka 9-21 ee 3.7.

```

1 package nimi.person;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class PersonTest {
8     ...
9     @Test
10    public void testChangePersonInformation() throws Exception {
11        Person person1 = new Person(0, "");
12        assertEquals(0, person1.getId());
13        assertEquals("", person1.getName());
14        assertEquals(null, person1.getPhone());
15        person1.setId(100);
16        assertEquals(100, person1.getId());
17        person1.setName("Cigal Shiidaad");
18        assertEquals("Cigal Shiidaad", person1.getName());
19        person1.setPhone("911911");
20        assertEquals("911911", person1.getPhone());
21    }
22 }

```

Tusaale 3.7: Tijaabinta badalista macluumaadka qofka

Aynu sharaxaad yar ka bixinno borogaraamka 3.7 intaynaan hore u socon:

1. Laynka 11 waxaynu samaynaynaa qof cusub (`person1`) kaas oo ID-diisu tahay 0, magaciisuna madhanyahay ""
2. Labada layn ee ka dambeeya waxaynu hubinaynaa in ID-da qofkaas (`person1`) tahay 0 iyo magaciisuna madhanyahay, taas oo ah sidii aynu ka filanaynay

3. Laynka 14 waxaynu hubinaynaa inaanu qofkaasi lahayn telefoon ama telefoonkiisu aanu jirin oo yahay qiimaha **null** (**null** waa qiime gaar ah oo aynu dib kaga hadlidoono)
4. Laynka 15 waxaynu badalaynaa ID-da qofkaas, oo aynu ka dhigaynaa 100. Laynka ka dambeeya waxaynu hubinaynaa in qofkaas marka la waydiiyo ID-diisu inuu soo celiyo 100
5. Laynka 17 waxaynu badalaynaa magaca qofkaas (person1), kadib waxaynu hubinaynaa in magacii is badaley
6. Laynka 19 waxaynu badalaynaa telefoonka qofkaas (person1), kadib waxaynu hubinaynaa in telefoonkii is badaley.

Kaxee si aad u aragto in laba tijaabo guulaysanayaan, laakiin ta ugu dambeysa guulgarraaysano oo casaan keenayso. Eclipse wuxu ka cabanayaa laymanka 15 iyo 17 oo uu leeyahay:

The method setId(int) is undefined for the type Person
The method setName(String) is undefined for the type Person

Intaas kadib, nooca Person kudar labada ficil ee setId iyo setName ee tusaalaha 3.8.

```

1 package nimi.person;
2
3 public class Person {
4     ...
5
6     public void setId(int id) {
7         this.id = id;
8     }
9
10    public void setName(String name) {
11        this.name = name;
12    }
13 }
14

```

Tusaale 3.8: Ficilada setId() iyo setName() ee nooca Person

Labadan ficil ee setId iyo setName ee nooca Person, shaqadoodu waa inay qiimaha la siiyey ka dhigaan ID-da ama magaca qofka. Intaynaan hore u socon aynu soo bandhigno qoraalka PersonTest iyo Person oo dhamaystiran.

```

1 package nimi.person;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class PersonTest {
8     @Test
9     public void testCreatePerson() throws Exception {
10         Person person1 = new Person(1, "Jaamac Cilmi");
11         assertEquals("Jaamac Cilmi", person1.getName());
12         assertEquals(1, person1.getId());
13
14         Person person2 = new Person(2, "Sahra Ahmed");
15         assertEquals("Sahra Ahmed", person2.getName());
16         assertEquals(2, person2.getId());
17     }
18
19     @Test
20     public void testPersonTelephone() throws Exception {
21         Person cigaal = new Person(123, "Cigaal Shiidaad");
22         String phoneNumber = "123098";
23         cigaal.setPhone(phoneNumber);
24         assertEquals(phoneNumber, cigaal.getPhone());
25     }
26
27     @Test
28     public void testChangePersonInformation() throws Exception {
29         Person person1 = new Person(0, "");
30         assertEquals(0, person1.getId());
31         assertEquals("", person1.getName());
32         assertEquals(null, person1.getPhone());
33         person1.setId(100);
34     }
35 }

```

```

34 assertEquals(100, person1.getId());
    person1.setName("Cigal Shiidaad");
36 assertEquals("Cigal Shiidaad", person1.getName());
    person1.setPhone("911911");
38 assertEquals("911911", person1.getPhone());
    }
40 }

```

Tusaale 3.9: Qoraalka PersonTest oo dhamaystiran

```

package nimi.person;
2
public class Person {
4
    private int id;
    private String name;
    private String phone;
8
    public Person(int id, String name) {
10        this.id = id;
        this.name = name;
12    }

    public String getName() {
14        return name;
16    }

    public int getId() {
18        return id;
20    }

    public void setPhone(String phone) {
22        this.phone = phone;
24    }

    public String getPhone() {
26        return phone;
28    }

    public void setId(int id) {
30        this.id = id;
32    }

    public void setName(String name) {
34        this.name = name;
36    }
}

```

Tusaale 3.10: Qoraalka Person oo dhamaystiran

Mar kale kaxee PersonTest si aad u aragto in saddexdiisa tijaaboba cagaar yihiin sida ka muuqata sawirka 3.7.

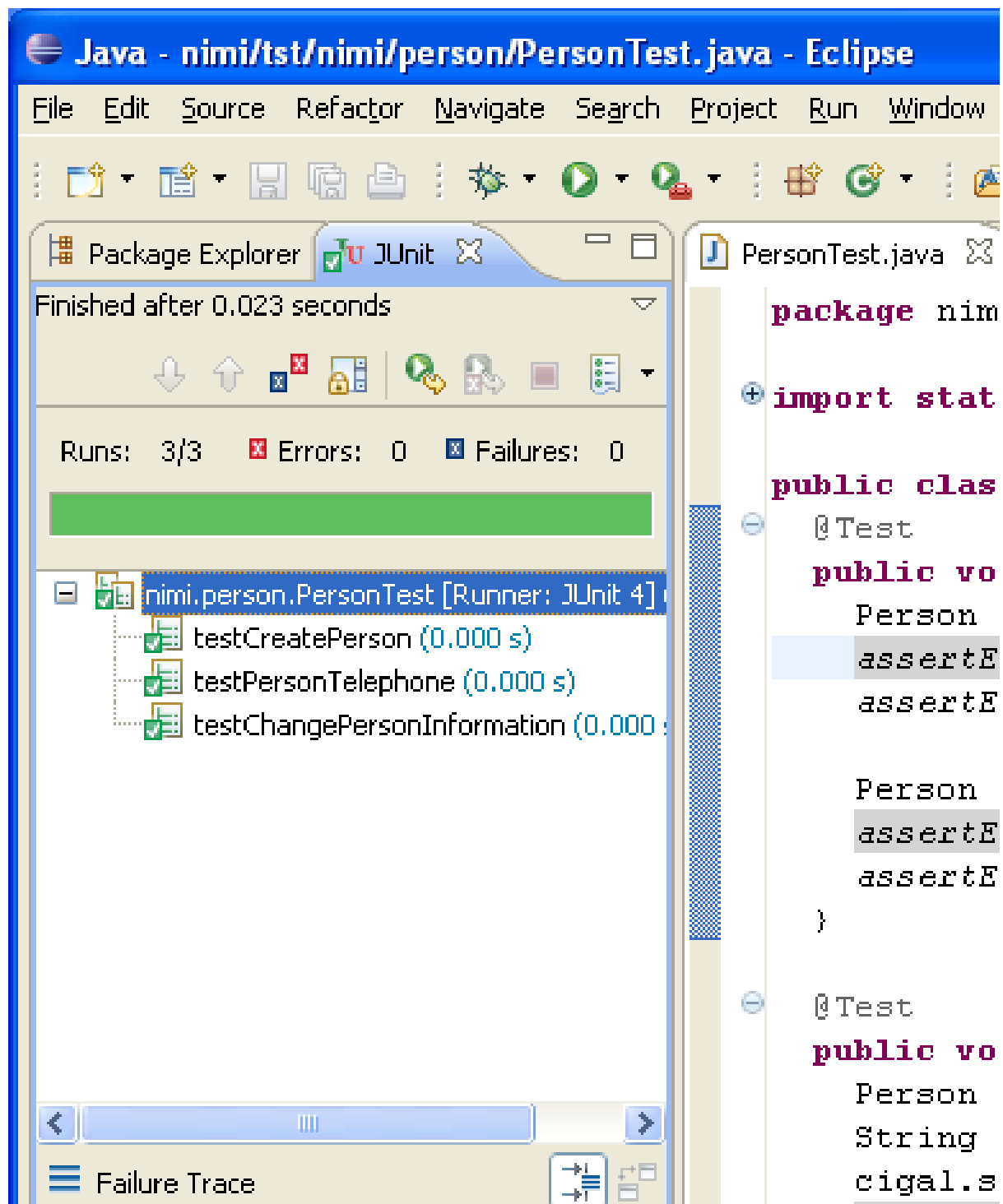
3.2 Gunaanad

Waxaynu qaybtan ku qornay noocii ugu horeeyay ee Person oo matalaaya qofka iyo tijaabiyihiisii oo magaciisa la yidhaahdo PersonTest. Intaynaan bilaabin qorista nooca Person, waxaynu bilawnay qorista nooca PersonTest. Sababta aynu sidaas u yeelay waa raacista nidaamka TDD ee sawirka 1.5. TDD sideedaba waxay keenaysaa hal xujo oo u baahan inaad garwaaqsato. Waxaynu bilawnay inaynu tijaabo u qorno nooc aan jirin. Sidee ayaad u tijaabin kartaa borogaraam ama nooc aan jirin? Waa inaad *ku fikirto inuu jiro* oo waliba sidii aad rabtay uu u yaallo. Sidaas awgeed markaynu bilawnay qorista nooca PersonTest waxaynu ku fikirnay inuu jiro nooc la yidhaahdo Person. Arintaasi waxay kugu kalifaysaa siduu sheegay [6] inaad ka fikirto borogaraamka aad qoridoonto caynka uu u eekaanaayo iyo sida loo isticmaalaayo.

Waa yahay. Arin kasta oo aynu uga baahanno nooca Person, waxaynu ku bilaabaynay inaynu ku qorno fiicil tijaabiya arintaas nooca PersonTest. Arinta kale ee u baahan in aad ogaato waa markasta oo aynu kaxaynno nooca PersonTest oo cagaar inoosoo baxo, waxaynu hubaal ognahay in nooca Person uu u shaqaynaayo sidii aynu rabnay.

Sidaynu soo sheegnayba talaabada ugu dambaysa ee TDD waad ka boodi kartaa haddii aaney jirin qoraal aad nadiifiso ama design aad wax ka badasho.

Waxa kale oo aynu cutubkan ku baranay sida guud ahaan borogaraamka Eclipse loo isticmaalo.



Sawirka 3.7: Saddexda tijaabo ee testCreatePerson, testPersonTelephone iyo testChangePersonInformation ee nooca PersonTest oo guulaysanaaya ama ansaxaaya (passing)

Waxa kale oo aad aragtay in xaga qoraalka nooca PersonTest uu leegyahay ama ka waynyahay nooca Person. Taasi waxay ku tusaysaa in nidaamka **TDD** uu keenaayo qoraal badan oo tijaabiye ah. Intaynaan ka gudbin qaybtan waxan rabaa inaan halkan ka sheego arin runtii aad muhiim u ah oo u baahan inaad xasuusato. Inta borogaraamku socdo waxa wax qabanaaya ee shaqaynaaya ee hawlaha kala wada waa walxaha. Noocyada way yartahay inta loo baahanaayo. Inta badan waxa loo baahanayaa kaliya marka walax cusub la samaynaayo.

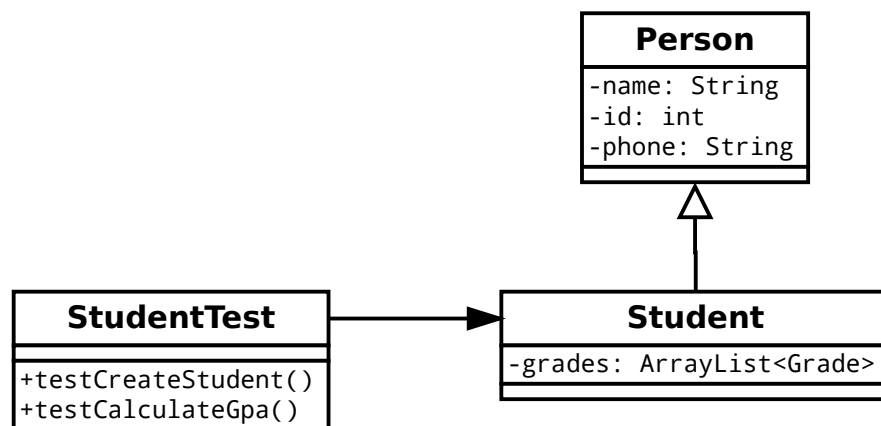
Cutub 4

Ardayda

Ardaygu waa qof, laakiin waxa jira waxyaale ardayga u gaar ah. Dhibcaha uu ka helo koorsooyinka ayaa ka mid ah. Mar kasta oo uu koorso fadhigeed dhameeyo ardaygu wuxu helayaa dhibcihii loogu talagalay in laga helo koorsoadaas, waxaana lagu darayaa dhibcaha uu haysto. Waxa la xisaabin karaa **Grade Point Average (GPA)**-ka ardayga, oo ah heerka celceliska darajada ama dhibcaha waxbarashada uu ka helay ardaygu. Sidaas daraadeed ardayga sifooyinkiisu waa kuwa qofka iyo kuwan oo ardayga u gaar ah:

1. Dhibcaha ardayga

Way jiraan sifooyin kale oo ardayga u gaar ah, laakiin halkan waxaynu ku soo qaadanay kaliya intan. Waxaynu ku bilaabaynaa nooc cusub oo tijaabiye ah. Maxaynu u bixinaynaa? Waxaynu u bixinaynaa `StudentTest`, kaas oo loogu talagalay inuu tijaabiyo nooca `Student` oo matalidoona ardayga. Qaabka ay saddexdaas nooc isula xidhiidhaan waxad ka arkaysaa sawirka 4.1. Sida sawirka ka muuqata waxaynu markii ugu horaysay isticmaaleynaa takhasuska (inheritance), maadaama ardaygu yahay qof. Takhasuska waxaynu kasoo hadalay hore.



Sawirka 4.1: Nooca `Student` iyo Tijaabihiisa `StudentTest`. `Student` waa qof, sidaas awgeed wuxu isticmaalayaa takhasuska oo waxa laga soo dhiraandhiriyeey nooca `Person`.

Waxad samaysaa nooc cusub oo tijaabiye ah adiga oo raacaya talaabada `NewTest`, waxad u buuxbuxisaa sidan: `Source folder=nimi/tst`, `Package=nimi.student`,

Name=StudentTest. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo StudentTest, laakiin madhan. StudentTest waxad ku bilawdaa sida tusaalaha 4.1.

```

1 package nimi.student;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class StudentTest {
8
9     @Test
10    public void testCreateStudent() {
11        String studentName = "Anisa Maxamed";
12        int studentId = 123;
13        String studentPhone = "102030";
14        Student student = new Student(studentId, studentName);
15        assertEquals(studentName, student.getName());
16        assertEquals(studentId, student.getId());
17        student.setPhone(studentPhone);
18        assertEquals(studentPhone, student.getPhone());
19    }
20 }

```

Tusaale 4.1: Qoraalka StudentTest bilawgiisa

Aynu sharaxaad yar ka bixiyo borogaraamka tusaalaha 4.1 intaynaan hore u socon:

1. Inta kale waxaynu soo aragnay hore, kaliya waxa is badalay magaca nooca oo imika ah StudentTest, halka kii hore ka ahaa PersonTest
2. Laynka 1-aad waa package declaration, laakiin imika waa nimi.student, halka nooca PersonTest uu ka ahaa nimi.person. Waxad arkaysaa iney faylasha PersonTest.java iyo StudentTest.java ku kaydsanyihiin kala meel, taasna waxa keenaysa kala duwanaansha package-kooda. Package-ka dib ayaynu kaga hadlidoonaa
3. Laymanka 9-19 waa ficil magaciisa la yidhaahdo testCreateStudent, kaas oo loogu talagalay in lagu tijaabiyo samaynta walxo cusub oo Student ah
4. Laynka 11 waxaynu samaynaynaa waxa loo yaqaan *local variable*, taas oo magaceeda layidhaahdo studentName, nooceedu yahay String qiimaheeduna yahay "Anisa Maxamed". Local variables-ka waxaynu kaga hadlaynaa cutubka 5.3. Sidoo kale labada layn ee ku xiga waxaynu samaynaynaa local variables studentId iyo studentPhone
5. Laynka 14 waxaynu samaynaynaa arday cusub, kaas oo aynu siinayno ID-ga iyo magaca ku kala kaydsan studentId iyo studentName
6. Laynka 15 waxaynu hubinaynaa magaca ardaygaas aynu samaynay marka la waydiiyo inuu qiimaha uu soo celiyo la midyahay ka ku kaydsan local variable-ka studentName. Laynka ka hoose waa la mid, laakiin wuxu eegayaa ID-ga
7. Laynka 17 waxaynu badalaynaa telefoonka ardayga oo aynu qiimaha ku kaydsan local variable-ka studentPhone ka dhigayno
8. Ugu dambayntii laynka 18 waxaynu hubinaynaa telefoonka ardayga.

Kaxee nooca StudentTest, si aad u aragto inuu guuldarraysanaayo. Eclipse wuxu ka cabanayaa laynka 14, oo uu leeyahay:

Student cannot be resolved to a type

Taas oo macnaheedu tahay inaanu garanaynin nooca Student (wali ma jiro noocaasi, kaliya waxaynu ku fikirnay inuu jiro). Waxad samaysaa nooc cusub adiga oo raacaya talaabada NewClass. Waxad u buuxbuuxisaa sidan: Source folder=nimi/src, Package=nimi.student, Name=Student. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo Student, laakiin madhan. Student wuxu ku bilaabmayaa sida tusaalaha 4.2.

```

1 package nimi.student;
2
3 import nimi.person.Person;
4

```

```

6 public class Student extends Person {
8     public Student(int id, String name) {
9         super(id, name);
10    }

```

Tusaale 4.2: Qoraalka Student bilawgiisa

Intaas kadib kaxee nooca StudentTest. Waxad helayaas markiiba cagaar. Sababtoo ah tijaabadan waxaynu ku xaqiijinay in nooca Student uu u shaqaynaayo sida nooca Person. Taasi waxay ku tusaysaa in noocu uu nooca laga soo dhiraandhiriyeey ka dhaxlaayo dhammaan wixii ficillo iyo sifooyin uu noocaasi lahaa. Qodobkani waa astaan aad u muhiim ah oo uu leeyahay takhasusku ama Inheritance-ku.

Laakiin wali waxa maqan wixii nooca Student u gaarka ahaa, oo ah dhibcaha ardayga. Dhibcaha aynu halkan ku isticmaalayno waa sidan: A=4, B=3, C=2, D=1 iyo F=0. Taas oo macnaheedu tahay in haddii ardaygu tusaale ahaan koorsada ka helo A ay u dhiganto 4 dhibcood marka la xisaabinaayo GPA-ga. Sidoo kale B, wuxu ardaygu ka helayaa saddex dhibcood. Ardaygu haddii uu koorsadaas ka helo F, waxay ka dhigantahay inuu ardaygaasi ku dhacay koorsadaas, oo wax dhibco ah ka helimaayo marka la xisaabinaayo GPA-ga ardaygaas. Dhibcahani waa tiro taxan, oo qiime walba loo bixiyey magac. Luuqadda Java tirooyinka noocan oo kale ah waxa loo adeegsadaa nooca enum. Waxad samaysaa enum cusub adiga oo adeegsanaaya talaabooyinkan:

1. Menu-ga Eclipse ka dooro File->New->Enum
2. Daaqadda soo boodda u buuxbuuxi sidan: Source Folder=nimi/src, Package=nimi.student, Name=Grade. Intaas kadib guji **Finish**
3. Waa inaad hesho enum cusub oo madhan oo tusaalaha 4.3 u eeg. enum iyo **class** aad ayey isugu dhawyihiin. Labadoodu waxay yeelan karaan ficiyo iyo sifooyin, laakiin xasuus enum waxa loo adeegsadaa kaliya tirooyinka taxanaha ah ee tiro walba magaca leedahay. enum looma adeegsan karo takhasuska (inheritance), taasna wuu kaga duwanyahay nooca **class**
4. Badal qoraalka nooca Grade oo u eekaysii sida tusaalaha 4.4.

```

2 package nimi.student;
4 public enum Grade {

```

Tusaale 4.3: Nooc madhan oo enum ah

```

1 package nimi.student;
3 public enum Grade {
4     A(4), B(3), C(2), D(1), F(0);
5     private int points;
7     Grade(int points) {
8         this.points = points;
9     }
11    public int getPoints() {
12        return this.points;
13    }

```

Tusaale 4.4: Qoraalka Grade oo dhamaystiran

Sharaxaad yar aynu ka bixino tusaalaha 4.4, intaynaan hore u socon:

1. Nooca enum waxa luuqadda Java lagusoo daray soosaarkeedii 5-aad (sidaas awgeed buuggani wuxu ku saabsanyahay soosaarkii 5-aad iyo wixii ka dambeeyey)
2. Laynka 4-aad ayaa ugu muhiimsan. Waxaynu samaynaynaa shanta darajo oo kala ah A, B, C, D, F iyo qiimeyaashooda (ama dhibcohoodii) oo kala 4,3,2,1,0 siday isugu xigaan
3. Laynka 5-aad waa sifada kaydinaaya qiimaha darajada tusaale ahaan darajada A wuxu kaydinaaya qiima 4
4. Ficilka getPoints waxa loogu talagalay in darajo kasta lagu waydiiyo qiimaheeda ama dhibcaheeda

5. Looma baahna inaynu tijaabino noocan enum-ka ah, sababtoo ah kaliya wuxu matalayaa tiro (ama magacyo) taxane ah oo shaqo kale ma qabanaayo.

Intaas kadib waxaynu nooca ardayga ee Student ku daraynaa dhibcihiis ama darajooyinkii iyo xisaabinta GPA. Marwalba waxaynu ku horaynaynaa tijaabaada waayo waxaynu raacaynaa nidaamka **TDD**. Waxad ku dartaa nooca StudentTest laymanka 14-28 ee tusaalaha 4.5.

```

1 package nimi.student;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class StudentTest {
8
9     @Test
10    public void testCreateStudent() {
11        ...
12    }
13
14    @Test
15    public void testCalculateGpa() throws Exception {
16        Student student = new Student(123, "Anisa Maxamed");
17        assertEquals(0.0, student.getGpa(), 0.05);
18        student.addGrade(Grade.A);
19        assertEquals(4.0, student.getGpa(), 0.05);
20        student.addGrade(Grade.B);
21        assertEquals(3.5, student.getGpa(), 0.05);
22        student.addGrade(Grade.C);
23        assertEquals(3.0, student.getGpa(), 0.05);
24        student.addGrade(Grade.D);
25        assertEquals(2.5, student.getGpa(), 0.05);
26        student.addGrade(Grade.F);
27        assertEquals(2.0, student.getGpa(), 0.05);
28    }
29 }

```

Tusaale 4.5: Tijaabinta xisaabinta GPA-da

Aynu sharaxaad yar ka bixino tusaalaha 4.5, intaynaan hore u socon:

1. Ficilka testCreateStudent hore ayaynu ugasoo hadalay, sidaas awgeed caloosha waan qariyey. Ficilka testCalculateGpa wuxu tijaabinayaa xisaabinta GPA-da ardayga. Laynka 16 waxaynu samaynaynaa arday cusub, innaga oo siinayna ID iyo magac
2. Laynka xiga ee 17 waa assertEquals oo aynu hore u soo marnay. Waxaynu waydiinaynaa ardayga GPA:diisa, kadibna waxaynu hubinaynaa inay la midtahay 0.0, maadaama aanu ardaygu illaa hadda wax dhibco ah helin. Qiimaha saddexaad ee 0.05 waa qiimaha ugu wayn ee ay ku kala duwanaan karaan labada qiime, maadaama ay jajab yihiin. Sideedaba isu eegista laba qiime oo jajab ah way adagtahay, taas ayaa keenta inaynu siino qiimaha wayn ee ay ku kala duwanaan karaan. Haddii kala duwanaanshahoodu ka hooseeyo qiimahaas, labada tiro ee jajabka ah waxa laga soo qaadayaa inay isku mid yihiin)
3. Intaas kadib laynka 18 waxaynu ardayga dhibcihiisa ku daraynaa darajada A, oo laga helaayo nooca Grade (Grade.A), innaga oo adeegsanayna ficilka addGrade (mar kale ficilkaasi ma jiro, waxaynu kasoo qaadaynaa inuu jiro)
4. Intaas kadib mar kale ayaynu ardayga waydiinaynaa GPA:diisa, kadibna waxaynu ka sugaynaa inuu inoosoo celiyo qiimaha 4.0, maadaama aanu illaa hadda dhibco kale haysan A mooyee
5. Laynka 20 waxaynu ardayga siinaynaa darajo ama dhibic ah B (Grade.B), kadibna waxaynu ka sugaynaa in GPA:diisu noqoto $3.5 (= \frac{4.0+3.0}{2})$
6. Sidaas ayaynu ku wadaynaa illaa uu ardaygu helaayo darajada ugu hoosaysa ee F, kadibna iskucel-celiskiisuna noqonaayo $2.0 (= \frac{4.0+3.0+2.0+1.0+0.0}{5})$.

Intaas kadib kaxee nooca StudentTest. Waa iney tijaabada hore ee testCreateStudent guulaysato, laakiin ta dambe ee testCalculatePga guuldarraysato, maadaama aynaan nooca Student ku darin ficilka getPga oo xisaabinaaya GPA:da ardayga iyo ficilka addGrade oo darajo cusub ardayga darajooyinkiisa ku darayaa. Eclipse wuxu ka cabanayaa getPga iyo addGrade isaga oo markaad midkiiba jiirka dulgayso odhanaaya:

The method `getGpa()` is undefined for the type `Student`

...

The method `addGrade(Grade)` is undefined for the type `Student`

Taas oo ka dhigan in labadaas ficil ka maqanyihiin nooca `Student`. Waxaynu bilaabaynaa inaynu nooca `Student` ku darno labadaas ficil, waxaanad ka arkaysaa tusaalaha 4.6.

```

1 package nimi.student;
2
3 import java.util.ArrayList;
4
5 import nimi.person.Person;
6
7 public class Student extends Person {
8     private ArrayList<Grade> grades;
9
10    public Student(int id, String name) {
11        super(id, name);
12        this.grades = new ArrayList<Grade>();
13    }
14
15    /**
16     * Xisaabi qiimaha iskucelceliska ee dhibcaha ardayga
17     * @return iskucelceliska ardayga
18     */
19    public double getGpa() {
20        if (grades.isEmpty()) {
21            return 0.0;
22        }
23        double total = 0;
24        for (Grade grade : grades) {
25            total += grade.getPoints();
26        }
27        return total / grades.size();
28    }
29
30    public void addGrade(Grade grade) {
31        this.grades.add(grade);
32    }
33 }

```

Tusaale 4.6: `Student` oo lagu daray ficilada `getGpa` iyo `addGrade`

Waxyaalo cusub ayaa soo kordhay oo u baahan in la falanqeeyo.

1. Marka hore laymanka 15-18 waxa loo yaqaan *comment* oo kama qaybqaadanayaan borogaraamka. Cutubka soo socda ayaynu kaga hadlidoonaa comments-ka
2. Laynka 8-aad waa sifadii kaydinaysay dhibcaha ardayga. Wax kasta oo kayd u baahan waa in sifo loo sameeyo. Nooca `ArrayList` oo kamid ah luuqadda Java ayaa inta badan loo adeegsadaa in lagu kaydiyo qiimeyaasha taxanaha ah, tusaale ahaan guruub arday ah. Noocan laftiisa cutubyada soo socda ayaynu kaga hadlaynaa, laakiin imika sidaas u qor
3. Laymanka 10-13 waa ficilkii constructor-ka ahaa. Laynka 11-aad waxaynu Id-da iyo magaca u gudbinaynaa nooca sare ee `Person` innaga oo adeegsanayna ereyga **super**. Laynka 12-aad waxaynu samaynaynaa walax cusub oo noocedu yahay `ArrayList` taas oo kaydinaysa dhibcaha ardayga
4. Laymanka 19-28 waa ficilka `getGpa` oo xisaabinaaya GPA-ga ardayga. Laymanka 20-22 haddii ardaygu aanu haysan wax dhibco ah wuxu ficilku soo celinayaa qiimaha 0.0, haddii kale wuxu xisaabinayaa GPA-da ardayga, taas oo ah marka hore isugee dhammaan dhibcaha ardayga, kadibna u qaybi inta ay yihiin ama tiradooda
5. Laymanka 30-32 waa ficilka `addGrade` kaas oo dhibic ama darajo cusub ku darayaa dhibcaha ardayga
6. Meelo badan oo u baahan sharaxaad faahfaahsan ayaa jira ee haka warwarin imika dib ayaynu kaga hadlaynaa.

Intaas kadib mar kale kaxee tijaabada `StudentTest` si aad u hesho cagaar. Illaa hadda waxaynu haysanaa labo nooc oo tijaabiye ah, oo kala ah `PersonTest` iyo `StudentTest`. Midkoodba gooni ayaynu u kaxaynkarnaa, laakiin inta badan waxaynu rabnaa ineynu helno meel mar kaliya laga wada kaxaynkaro wixii nooc ee tijaabiye ah. Si aynu arintaas u fulino waxad samaysaa nooc cusub oo tijaabiye ah adiga oo raacaya talaabada **NewTest**, adiga oo u buuxuuxinaaya sidan: Source folder=`nimi/tst`, Package=`nimi`, Name=`AllTests`. Intaas ka dib u eekaysii qoraalkiisa tusaalaha 4.7.

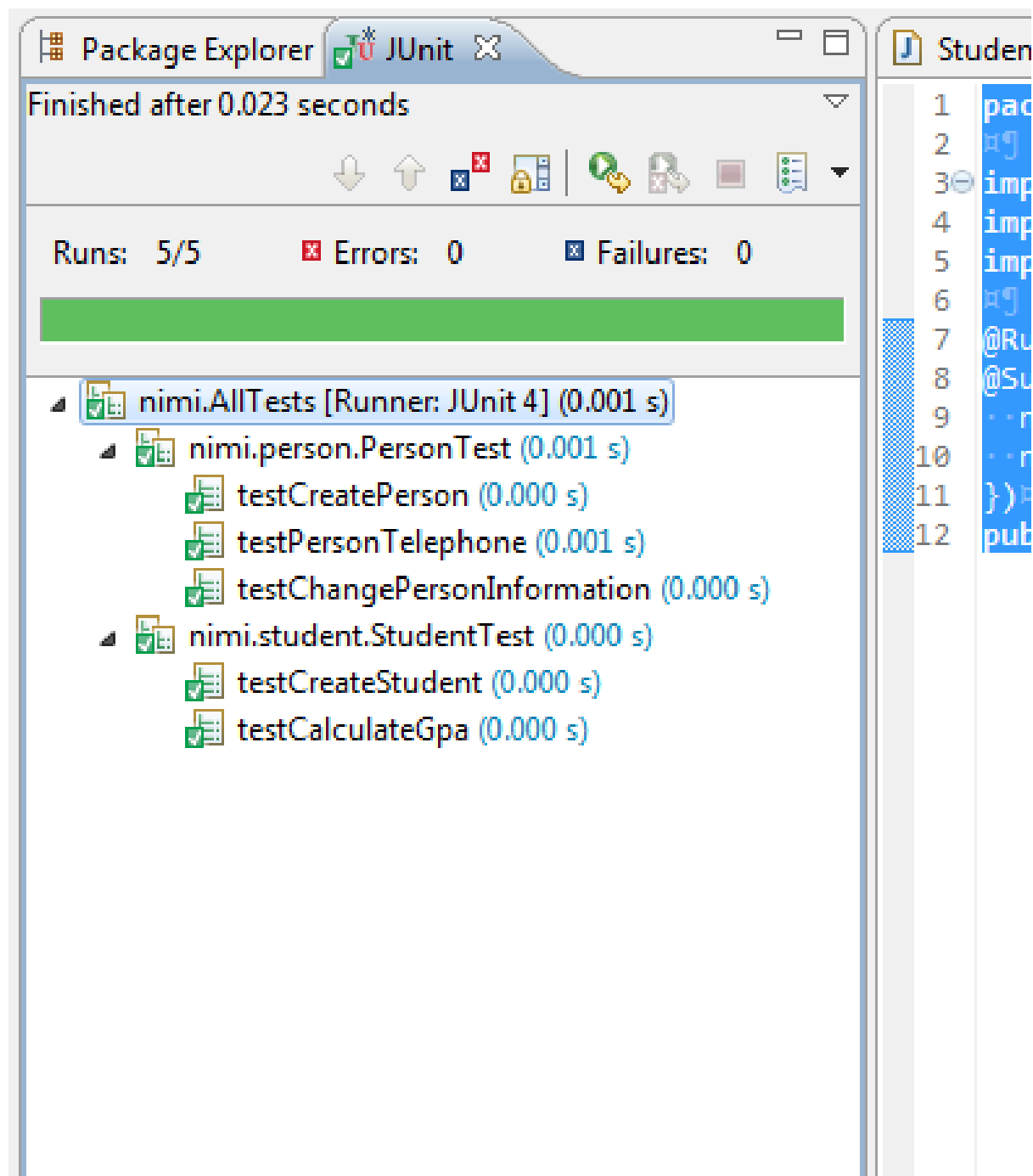
```

package nimi;
2
import org.junit.runner.RunWith;
4 import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;
6
@RunWith(Suite.class)
8 @SuiteClasses({
    nimi.person.PersonTest.class, // PersonTest
10    nimi.student.StudentTest.class // StudentTest
    })
12 public class AllTests {}

```

Tusaale 4.7: Nooca AllTests wuxu uruurinayaa noocyada kale ee tijaabiyeesha ah. Waxaanu noqonayaa hal meel oo mar kaliya laga wada kaxaynkaro.

Intaas kadib kaxee nooca AllTests, adiga oo raacaa talaabada RunTest. Waa inaad hesho sawir 4.2.



Sawirka 4.2: Natijada kuusoo baxaysa markaad kaxayso nooca AllTests.

Intaynaan ka gudbin halkan waxaynu samaynaynaa erey-bixin yar. Nooca AllTests waxa loo yaqaan *Test Suite*. Nooca PersonTest (ama StudentTest) waxa loo yaqaan *Test Case*. Ficilada Testcase-ka tusaale ahaan testCreatePerson, waxa loo yaqaan kaliya *Test* ama *Single Test*.

4.1 Gunaanad

Cutubkan waxaynu kaga hadalay ardayda. Ardaygu waa shakhsi ama qof oo wuxu leeyahay dhammaan sifooyinka shakhsiga, sidaas awgeed waxaynu adeegsanay takhasuska. Waxa soo kordhay waxyaalo badan oo cusub, sidaas awgeed waa lagama maarmaan inaynu hoos u daadagno oo faahfaahin dheeraad ka bixino.

Cutub 5

Naxwaha Luuqadda Java

Cutubkani wuxu ka hadlayaa naxwaha luuqadda Java (Java Language Syntax). Naxwaha luuqadda af-ingiriisiga waxa lagu yidhaahdaa *grammar*, marka laga hadlaayo luuqadaha dadku ku hadlaan. Laakiin marka laga hadlaayo luuqadaha kumbuyuutarka waxa inta badan la isticmaalaa ereyga *Syntax*. Naxwuhu wuxu qeexayaa shuruudaha la raacaayo marka la qoraayo qoraalada luuqadaha dadka sida af-soomaaliga, af-ingiriisiga, af-carabiga, iwm. Naxwe waxan cutubkan ugu bixiyey maadaama aan rabo inaan kaga hadlo shuruudaha la raacaayo marka la qoraayo qoraalada borogaraamyada Java. Markaas haddii qoraalka (source code-ka) borogaraamka ku jirto qalad ama qaladaad xagga naxwaha ah waxa la yidhaahdaa waa *syntax error*.

5.1 Ereyada u gaarka ah Java

Ereyada u gaarka ah luuqadda Java *Java Keywords* waxad ka arkaysaa miiska 5.1. Ereyadan uma isticmaalikartid magacyada noocyada, magacyada sifooyinka, magacyada ficilada iyo magacyada door-soomeyaasha, kuwaas oo guud ahaan loo yaqaano *identifiers*.

Miiska 5.1: Ereyada u gaarka ah luuqadda Java (Java keywords)

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Qaar badan oo ka mid ereyada miiska 5.1 waad soo aragtay oo tusaalooyinkii aynu soo qaadnay ayaad kusoo aragtay, laakiin sharaxaad badan kamaynaan bixin illaa hadda. Ereyada aad soo aragtay waxa ka mid ah tusaale ahaan **package**, **import**, **public**, **class**, **enum**, **int**, **return**, **this**, **new**, **throws**, **extends** iyo **super**. Sharaxaadooda badankood waxad ku baran qaybaha soo socda ee buuggan. Qaarkoodna waxad baran mustaqbalka markaad dhexgasho luuqadda Java. Qaarkoodna waxa laga yaabaa inaad waligaa u baahan. Qoraalka borogaraamka haddii aad eegto, waxad arkaysaa in ereyada Java u gaarka ah kalarkoodu ka duwanyahay qoraalka kale.

5.2 Noocyada

Noocyadu (classes) waa aasaaska luuqadda Java, maadaama Java tahay **OOP** language. **OOP** waxaynu ugasoo hadalay hore. Waxaynu soo aragnay dhawr nooc oo ay kamid yihiin tusaale ahaan **PersonTest** iyo **Person**, markaas waxaynu ka haysanaa sawir guud habka ay u eegyihiin noocyadu. Imika waxaynu sharaxaad ka bixinaynaa qaabka ama shuruudda ay u yaallaan noocyada Java.

```
2 AccessModifier [abstract|final] class ClassName [extends  
SuperClassName implements InterfaceName] {  
... // caloosha nooca oo ka kooban
```

```

4 ... // sifooyin, ficillo iyo noocy kale
  }

```

Tusaale 5.1: Qaabka noocyada Java u qoranyihiin

Tusaalaha 5.1 wuxu ku tusayaa qaabka uu u dhisanyahay qoraalka noocyada Java. Waa shuruudda aad raacayso markaad qorayso nooc cusub. Aan waxyar ka taabto qaybaha kala duwan ee uu ka koobanyahay qoraalka noocu aniga oo tixraacaaya tusaalaha 5.1:

1. *AccessModifier* (**Access Modifier**) waxay noqon kartaa **public** (oo aynu illaa hadda isticmaalaynay) ama way madhnaan kartaa. Innagu waxaynu buuggan ku isticmaaleynaa marwalba **public**, taas oo ka dhigan in noocan borogaraamka meel kasta laga isticmaali karo. *AccessModifier*-ka kadib waxa imankara ereyada **abstract** ama **final** midkood, laakiin qasab ma aha inaad midkood isticmaasho, sidaas awgeed innagu labadoodaba waynu ka tagaynaa. Waxyar ayey labadaas erey ka badalayaan qaabka nooca loo isticmaalaayo, sidaas awgeed looma baahna inaynu halkan kaga hadalo. **abstract** waxaynu ku arkidoonaa qaybta ugu dambaysa ee buuggan. Halkaas waxad ka arkaysaa in nooc walba qoristiisa lagu bilaabaayo **public** oo uu ku xigo ereyga Java u gaarka ah ee **class**, kaas oo sheegaaya in nooc cusub bilaabmaayo
2. Bilawga **public class** ka dib waxa imanaysa *ClassName* oo ah magaca nooca. Waa inuu xaraf weyn ku bilaabmo magacu. Haddii magacu ka koobanyahay dhawr qaybood, waa in qayb waliba xaraf weyn ku bilaabanto. Waxaynu soo aragnay tusaale ahaan magacyada *Person*, *PersonTest*, *Student*, *StudentTest*. Waxa shuruud ah inaanu magaca noocu noqon erey kamid ah ereyada u gaarka ah luuqadda Java. Buuggan inta badan waxaynu ku isticmaalaynaa magacyo af-ingiriisi ah, maadaama ay ka gaaban yihiin af-soomaaliga inta badan. Laakiin adigu haddii aad rabto noocyadaada waxad ula bixi kartaa magacyo af-soomaali ah sida *Qof*, *QofTijaabiye*, *Arday* iyo *ArdayTijaabiye*
3. Qaybta xerada ku jirta ee [**extends** *SuperClassName* **implements** *InterfaceName*] qasab ma aha, inkasoo aynu **extends** kusoo aragnay markaynu ka hadleynay ardayda, waxaana la isticmaalaa markaad nooca kasoo dhiraandhirinayso nooc kale sida *Student* **extends** *Person*. Sidaas awgeed waxay la xidhiidhaan takhasuska
4. Intaas kadib waxa imanaysa calooshii nooca oo ku jirta calaamadda { ee laynka 2-aad iyo calaamadda } ee laynka 5-aad dhexdooda. Waa qasab labadaas calaamadood, sababtoo ah waxay qeexayaan bilawga iyo dhamaadka nooca
5. Ugu dambayntii haddii aanu noocu buuxin shuruudahaas aynu kor kusoo xusnay, noocaas sax looma qorin, waxaanu keenayaa waxa loo yaqaan *syntax error*, oo ah in qoraalka noocu qaldanyahay borogaraamka *Eclipse*:na wuxu ku calaamadinayaa casaan, isaga oo kuu sheegaaya meesha ciladdu ka jirto.

Noocu wuxu caloosha ku haysan karaa sifooyin, ficillo iyo noocy kale ama wuu madhnaan karaa. Waxaynu sharaxaad ka bixinaynaa sifooyinka iyo ficilada.

5.3 Sifooyinka

Waxaynu soo marnay sifooyinka (**Attributes**, **Fields**) qofka oo ah ID (id), magac (name) iyo telefoon (phone). Waxa kale oo aynu sheegnay in mustaqbalka loo baahankaro sifooyin kale sida: da'da, emailka, adarayska, iwm. Sifooyinka waxa lagu qorayaa nooca calooshiisa, kadibna walax kasta oo noocaas laga sameeyo waxay yeelanaysaa sifooyinkaas, laakiin waxay walxahaasi ku kala duwanaan karaan qiimaha sifooyinkaas. Sifooyinka noocu sidoodaba waxay u eegyihiin tusaalaha 5.2, gaar ahaan laynka 3-aad, marka la qoraayo.

Tusaale ahaan saddexda sifo ee nooca *Person* waxad ka arkaysaa laymanka 5 illaa 7 ee tusaalaha 3.10.

```

2 public class ClassName {
  AccessModifier [final | Static] TypeName FieldName [= InitialValue];
  ...
4 }

```

Tusaale 5.2: Sifooyinka noocu qaabkay u yaalaan

Sifooyinka inta badan waxa lagu qoraa bilawga nooca. Aan waxyar ka taabto qaabka ay u qoranyihiin sifooyinka noocu aniga oo tixraacaaya tusaalaha 5.2:

Miiska 5.2: Primitive Data Types-ka Java

Magaca	Sharaxaad
byte	Waa nambar 8-bit ah. Qiimaha ay qaadi karto waa inta u dhaxaysa -128 illaa 127
short	Waa nambar 16-bit ah. Waa labo jibaar qaadka byte-ka. Waxay qaadi kartaa inta u dhaxaysa -32768 illaa 32767
int	Waa nambar 32-bit ah. Qaadkeedu waa inta u dhaxaysa -2147483648 illaa 2147483647.
long	Waa nambar 64-bit ah. Waa labo jibaarka int-ka.
float	Waa 32-bit oo floating point ah. Waxa loogu talagalay tirada jajabka ah sida 3.14
double	Waa laba jibaarka float-ka oo ah 64-bit nambar jajab ah
boolean	Waa true ama false
char	Waa xaraf 16-bit ah oo Unicode ah. Waxa loogu talagalay xarfaha qoraalka.

1. *AccessModifier* waxay noqon kartaa **private**, **public**, **protected** ama way madhnaan kartaa. Innagu waxaynu inta badan adeegsanaynaa **private** iyo **protected**. Haddii sifadu ay noqoto **public**, waxay ka dhigantahay in nooca dibeddiisa laga isticmaali karo sifadaas oo noocyada kale ay toos u arkayaan, badalikaaraana qiimaha sifadaas. Haddii sifadu tahay **protected**, waxay u muuqanaysaa wixii nooc ee noocan laga soo dhiraandhiriyo. Sidaas awgeed tusaale ahaan sifooyinka nooca Person waxaynu odhankarnaa waxa ku haboonayd **protected**, si ay ugu muuqdaan nooca Student
2. *FieldName* waa magaca sifada, oo waxay noqon kartaa magacaad rabto. Waa inaad ku bilawdo xaraf yar. Haddii magacu laba ama wax ka badan oo qaybood ka koobanyahay waa inaad qaybaha dambe xaraf weyn ku bilawdo. Tusaale ahaan name, firstName, gameOver, modelNumber, iwm.
3. *TypeName* waa nooca sifada, waxaanay noqon kartaa nooc (class) sida String, Object, Door, Person, Student, iwm, mid ka mid ah *Primitive Data Types (PDT)*-ka Java oo aad ka arkayso miiska 5.2 ama nooc enum ah. Sadexdaa midkood waa inay noqoto
4. Ugu dambayntii qaybta xerada ku jirta ee [= InitialValue] oo aan ahayn qasab, waxad u isticmaalaysaa haddii aad rabto inaad sifadaas hore ka siiso qiime tusaale ahaan **private int** age = 20. Laakiin waxay qasab noqonaysaa haddii sifadaasi tahay **final** oo ka horayn karta TypeName.

Maadaama uu kumbuyuutarku fahmaayo nambarada 0 iyo 1 oo loo yaqaano *Bits*, nooc walba oo asal ah (primitive) waa in qaadkiisa lagu qeexo bits sidaad ka arkeyso miiska 5.2. Nooca Person waxaynu ku isticmaalay saddex sifo oo laba tahay nooca String (magaca iyo telefoonka), midna yahay **int** (ID-da) oo ka mid ah PDT-da. String waa nooc la socda luuqadda Java oo loo isticmaalo wixii qoraal ama meeris (text) ah tusaale ahaan magaca, adarayska, telefoonka (oo ka kooban inta badan nambaro iyo calaamadda plus (+)), email-ka, iwm. Wixii tiro ah waxad u adeegsan kartaa noocyada **int**, **long**, **float** ama **double** kolba kii aad istidhaahdo wuu ku haboonyahay. Noocyada PDT waxaynu illaa hadda ka isticmaalay **int** oo kaliya, taas oo aynu u isticmaalay ID-da (id) qofka sidaan soo sheegayba.

Intaynaan hore u socon, waxad arkaysaa inaan ereyga af-soomaaliga ee *nooc* u isticmaalay halkan laba arimood, taas oo keeni karta khalkhal. Mar waxan u isticmaaley ereyga af-ingiriisiga ee *class* oo waa markaan ka hadlaayo noocyada sida Person, PersonTest, Student, StudentTest iwm. Marna waxan u isticmaaley nooca sifada oo noqonkara nooc class ah ama mid ka mid ah noocyada PDT-da, ee aad kor kusoo martay. Markaas hadba arinta aan ka hadlaayo ayey ku xidhantahay waxaan ula jeedo ereyga nooc. Markaas arintaas fiiro gaar ah sii.

Sifooyinku kaligood waxba ma tarayaan, haddii aan lagu isticmaali karin ficilada calooshooda. Waxaynu odhankarnaa macluumaadka kaydintiisu waxba tarimayso haddii aynaan waxku darikarin ama ka dhimikarin ama wax ku xisaabinkarin. Sidaas awgeed ficiladu waa inay isticmaalikaraan macluumaadka ku kaydsan sifooyinka. Aan u gudbo sharaxaad ku saabsan ficilada.

5.4 Ficilada

Ficiladu ama hababku (methods) waxay ka mid yihiin waxyaalaha ku jira ama ku qoran caloosha noocyada Java, waxaanay u qoranyihiin sida ka muuqata tusaalaha 5.3.

```

1 public class ClassName {
2     ...
3     AccessModifier [static] ReturnType MethodName(
4         ParamType paramName1, ParamType2 paramName2, ...)
5     [throws ExceptionName] {
6         ... // caloosha ficilka
7     }
8     ...
9 }

```

Tusaale 5.3: Ficilada nooca qaabka ay u qoranyihiin

Aan sharaxaad yar ka bixiyo qaabka ficilada noocu u qoranyihiin aniga oo tixraacaaya tusaalaha 5.3:

1. Laynka 3-aad *AccessModifier* waxay noqon kartaa **public**, **private**, **protected** ama way madhnaan kartaa. Innagu waxaynu inta badan buuggan ku isticmaaleynaa **public** iyo **private**. Ficilada **private**-ka ah lagama isticmaalikaro nooca dibeddiisa, kaliya waxa isticmaali kara ficilada kale ee nooca. Ficilada **public**-ka ah waxa laga isticmaali karaa nooca dibeddiisa (iyo calooshiisa), oo waxa isticmaali kara noocayada kale. Waxa kale oo jiri kara ficillo **protected** ah, kuwaas oo loogu talagalay inay u muuqdaan kaliya noocyada subclass-ka ah ama laga soo dhiraandhiriyey noocan
2. *AccessModifier* kadib waxa imankara ereyga **static**, kaas oo aan qasab ahayn, loona isticmaalo ficilada aan ku xidhnayn walxaha. Tusaale ahaan ficilka **public static void** main(...) ayaad ku arkaysaa ereygaas oo la isticmaalay. Ficilkaas si loo waco looma baahna in walax noocaas ah la sameeyo ee toos ayaa loo wici karaa. Inta badan ficilada **static**-ka ah waxa loo yaqaan kalkaaliyeyaal ama *Utilities* . Ereygaas waxa kale oo loo isticmaali karaa sifooyinka
3. *ReturnType* waa nooca qiimaha uu soo celinaayo ficilkaasi. Haddii ficilkaasi aanu qiime soo celinaynin *ReturnType*-kiisu waa **void**. *ReturnType* wuxu noqon karaa nooc (class) sida String, Door, Person iwm ama mid ka mid ah noocyada Primitive Data Types-ka Java ee miiska 5.2
4. *MethodName* waa magaca ficilka, kaas oo inta badan xaraf yar ku bilaabma. Kadib waxa imanaysa liiska *parameters*-ka oo xero ama qaws (...) ku jirta. Parameter kasta waxay leedahay nooc *ParamType*, kaas oo u dhigma *ReturnType* iyo magac *ParamName*, kaas oo loo qoraayo sida sifooyinka. Parameters waa qaabka ficilka loogu gudbinaayo qiimeyaasha. Parameters way madhnaan karaan, haddii aan ficilka wax qiime ah loo gudbinaynin. Tusaale ahaan ficilka *getName* ee nooca Person ma laha wax parameters ah, laakiin ficilka *setName* wuxu leeyahay hal parameter oo loogu talagalay in loogu gudbiyo qiimaha magaca cusub ee qofka
5. Qaybta xerada ku xidhan ee **[throws Exception]** waxay la xidhiidhaa waxa loo yaqaano **Exceptions**, waxaynu kaga hadlidoonaa dib oo looma baahna inaynu halkan kaga hadalo, sababtoo ah ma aha qasab
6. Haddii ficilkaasi leeyahay *ReturnType* ka duwan **void**, waxa shuruud ah in ficilkaasi soo celi qiime. Tusaale ahaan eeg ficilada *getName*, *getId* iyo *getPhone* ee nooca Person. Soo celista qiimuhu waxay ku imanaysaa isticmaalista ereyga **return** ee u gaarka ah Java. Inta badan ficilada qiimaha soo celinaaya magacooda waxa ku jira ereyga ***get*** oo ka dhigan isii
7. Intaas kadib waxa imanaysa calooshii ficilka oo ku jirta calaamadda { iyo } dhexdooda. Waa qasab labadaas calaamadood, sababtoo ah waxay qeexayaan bilawga iyo dhamaadka ficilka.

Waxan carabka ku dhuftay inay muhiim tahay in sifooyinka nooca lagu isticmaali karo caloosha ficilka. Waxan soo qaadnayaa laba tusaale. Tusaalaha 5.4 waxan kasoo goostay nooca Person ee aynu soo aragnay hore. Waxan kasoo goostay laba ficil oo kala *setPhone* iyo *getPhone*. Ficilka *getPhone* wuxu soo celinayaa sifada phone. Ficilka *getPhone* qiimaha uuso celinaayo noociisu (*ReturnType*) iyo sifada phone ee laynka 7-aad nooceedu waa isku mid. Ficilka labaad ee *setPhone*, waxaynu rabnaa inuu qiimaha parameter-ka phone loogusoo gudbiyey inuu ku kaydiyo sifada phone. Maadaama ay parameter-ka iyo sifada isku magac yihiin, sifada nooca waxaynu ka horaysiinay ereyga **this** oo aynu dib kaga hadlidoono.

```

1 package nimi.person;
2
3 public class Person {
4

```

```

    private int id;
6   private String name;
    private String phone;
8
    ...
10
    public void setPhone(String phone) {
12         this.phone = phone;
    }
14
    public String getPhone() {
16         return phone;
    }
18
    ...
20 }

```

Tusaale 5.4: Ku isticmaalka sifooyinka ficilka calooshiisa 1

Tusaalaha labaad ee 5.5 waxaynu kasoo goosanay nooca Student ee isna aynu hore u soo aragnay. Gaar ahaan waxaynu soo goosanay ficilka `getGpa`. Haddii aad caloosha ficilkan u fiirsato meelkasta oo aad ku aragto grades, waxaynu ula jeednaa sifada grades ee nooca Student. Haddii aynu rabno meelkasta oo ficilka calooshiisa ah in aynu ku isticmaalo grades waxaynu qoraynaa `this.grades` ama grades oo kaliya.

```

package nimi.student;
2
    ...
4
    public class Student extends Person {
6         private ArrayList<Grade> grades;
8
        ...
10
        /**
         * Xisaabi qiimaha iskucelceliska ee dhibcaha ardayga
12         * @return iskucelceliska ardayga
         */
14         public double getGpa() {
            if (grades.isEmpty()) {
16                 return 0.0;
            }
18             double total = 0;
            for (Grade grade : grades) {
20                 total += grade.getPoints();
            }
22             return total / grades.size();
        }
24
        ...
26 }

```

Tusaale 5.5: Ku isticmaalka sifooyinka ficilka calooshiisa 2

Sifada grades ee nooca Student haddii aad u fiirsato waxad iswaydiinaysaa waa maxay `ArrayList<Grade>`? `ArrayList` waa nooc la socda luuqadda Java oo loo isticmaalo in lagu qabto ama kaydiyo guruub walxo ah oo isku nooc ah, tusaale ahaan guruub riyo ah, guruub baabuur ah, guruub ciyaal ah, iwm. Nooca calaamadaha `<...>` u dhexeeya wuxu sheegayaa guruubkaas walxaha ah noocooda. Noocyada sidan oo kale ah waxa loo yaqaan **Generics** oo waxaynu ku dulmari gadaal buuggan.

Iskusoo duub, markaad rabto ficilka dhexdiisa inaad ku isticmaasho sifooyinka nooca waxad isticmaalaysaa magaca sifada oo laga horaysiiyey ereyga `this` iyo dhibic ama magaca sifada oo kaliya.

5.4.1 Constructor

Constructor mar ayaynu waxyar ka sheegnay, gaar ahaan markaynu ka hadlaynay qofka ama shakhsiga iyo sifooyinkiisa. Maadaama aynu ficilada wax ka sheegnay, waxa fiican inaynu mar labaad waxoogaa ka sheegno ficilka constructor iyo shaqadiisa. Xaqiiqadu waxay tahay noocu wuxu yeelan karaa in ka badan hal constructor, laakiin inta badan wuxu leeyahay hal constructor oo kaliya. Ficilka constructor magaciisu wuxu la mid yahay magaca nooca, taas oo aad kaga garanayso ficilada kale ee nooca. Waxaana la adeegsadaa marka la samaynaayo walax cusub oo noocaas ah. Af-soomaaliga waxaynu odhan karnaa waa ficilka dhisaha. Qasab ma ah inaad nooca u samayso ficil dhise ah, laakiin way ficantahay. Aynu soo qaadanno tusaalooyin, si aynu daaha waxyar arintan uga faydno. Tusaalaha 5.6 waxad ka arkaysaa labadii nooc ee aynu soo marnay iyo ficiladooda dhisaha ah.

```

1 public class Person {
2     private int id;
4     private String name;
5     ...
6     public Person(int id, String name) {
8         this.id = id;
9         this.name = name;
10    }
11    ...
12 }

14 public class Student extends Person {
15     private List<Grade> grades;
16
17     public Student(int id, String name) {
18         super(id, name);
19         this.grades = new ArrayList<Grade>();
20    }
21    ...
22 }

```

Tusaale 5.6: Ficilka dhisaha (Constructor) ee noocyada Person iyo Student

Aynu sharaxaad ka bixinno tusaalaha 5.6. Nooca Person ee sare ficilkiisa dhisaha ah waxad ka arkaysaa laymanka 7-10, sidoo kale nooca Student waxad ka arkaysaa laymanka 17-20. Labada ficilba magacoodu wuxu la midyahay magaca nooca, taasina waxay kuu cadaynaysaa inay yihiin constructor ama ficillo dhiseyaal ah. Access modifier-koodu waa inay noqoto **public**. Su'aasha xigtaa waxay tahay goorma ayaa la adeegsanayaa constructor-ka nooca? Aynu tusaale soo qaadano arintaas inoo muujinaaya. Tusaalaha 5.7 ayaa kuu iftiiminaaya arintaas.

```

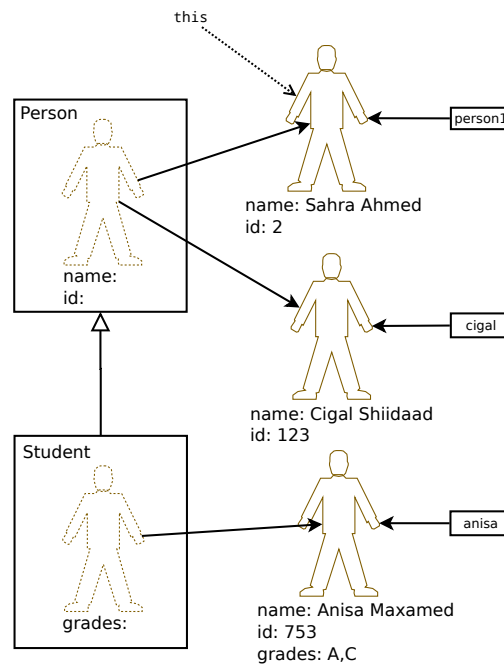
...
2 Person person1 = new Person(2, "Sahra Ahmed");
3 Person cigaal = new Person(123, "Cigaal Shiidaad");
4
5 //waydii cigaal telefoonkiisa si aynu u wacno
6 cigaal.getPhone();
7 ...
8
9 Student anisa = new Student(753, "Anisa Maxamed");
10 anisa.addGrade(Grade.A);
11 anisa.addGrade(Grade.C);
12
13 //waydii Anisa ID-deeda
14 anisa.getId();
15 ...

```

Tusaale 5.7: Isticmaalka ficilka dhisaha ah

Luuqadda Java ereyga **new** ayaa loo adeegsadaa samaynta walax cusub. Laymanka 2-3 ee tusaalaha 5.7 waxaynu samaynaynaa laba walxood oo noocu yahay Person. Tusaalaha sare ee 5.6 (gaar ahaan laymanka 7-10) waxad ka arkaysaa in ficilka dhisaha ah ee nooca Person uu leeyahay laba parameter oo kala ah *id* iyo *name*, kuwaas oo loogu talagalay in qiime loogu gudbiyo sifooyinka nooca ee iyana magacoodu yahay *id* iyo *name* ee ka muuqda laymanka 3 iyo 4 ee isla tusaalahaas. Labadaas sifo oo aan qiime loo soo gudbin la'aantood lama samaynkaro walxo noocoodu yahay Person. Sidaas awgeed markasta oo aynu rabno inaynu samayno walax Person ah waa inaynu qiimeyaal siino labadaas sifo. Laynka 2-aad ee 5.7 waxaynu siinaynaa qiimeyaasha (*id*=2, *name*="Sahra Ahmed"), laynka xigana (*id*=123, *name*="Cigaal Shiidaad").

Sawirka 5.1 ayaa kuu muujinaaya tusaalaha 5.7 iyo guud ahaan xidhiidhka ka dhexeeya nooca iyo walxaha. Sidaad sawirka ka arkayso noocu (class) wuxu qeexayaa caynka ay u eegyihiin iyo sifooyinka ay leeyihiin walxaha noocan laga sameeyaa. Sida sawirka ka muuqada noocyada waxa lagu sawiray xariijin goo'go'an, taas oo ku tusaalaynaysa inaanu noocu wax faa'iido ah oo badan kuu lahayn wakhtiga borogaraamka la isticmaalaayo. Sida sawirka ka muuqata nooca Person waxaynu ka samaynaa laba walxood, nooca Student-na hal walax. Sida sawirka ka muuqada walax kastaba sifooyinkeeda waxa lagu buuxinayaa qiimayaashii aynu siinay markii aynu samaynaynay walaxdaas. Waxa kale oo muhiim ah inaad fahamto magacyada *person1* iyo *cigaal* ee laymanka 2-3 ee borogaraamka 5.7 iyo qaabka ay sawirka uga muuqdaan. Magacyadaas oo layidhaahdo *variables* ama *references* waa hogaan ama tilmaan lagu garto walaxdaas, si tusaale ahaan meel kale borogaraamka looga isticmaalo walaxdaas sida ka muuqata laynka 6-aad ee tusaalaha 5.7. Waxa kale oo hogaankaas loo gudbin karaa ficilada. Variables-ka ama doorsoomeyaasha waxaynu kaga hadlaynaa isla qaybtan.



Sawirka 5.1: Tusaalaha 5.7 oo sawir ahaan loo muujiyey iyo ereyga this.

Intaynaan hore u socon waxa muhiim ah inaad aragto ama fahamto in walaxdu leedahay dhammaan ficilada nooceda.

5.5 This

Ereyga **this** waxad ku arkaysaa meelo badan oo ka mid borogaraamyada Java. Gaar ahaan waxad aad ugu arkaysaa ficilada calooshooda. Ereyga **this** waxad ka arkaysaa sawirka 5.1. Sida sawirka ka muuqata ereyga **this** wuxu tilmaamayaa kolba walaxda wakhtigaas ficiladeeda la kaxaynaayo. Aan tusaale yar soo qaato, kaas oo aan kasoo goostay nooca Person, waxaanad ka arkaysaa tusaalaha 5.8.

Ereyga **this** wuxu gaar u yahay luuqadda Java, waxaanu ka dhiganyahay *kan* ama walaxdan, waxaana inta badan loo isticmaalaa meesha uu shaki ka imankaro. Ereyga **this** ma aha qasab, laakiin sababta innagu kaliftay inaynu isticmaalo waa inaynu ku kala saarno sifooyinka nooca Person iyo parameters-ka ficilka. Si aynu haddaba labadaas u kala saarno waa inaynu sifooyinka nooca ka horaysiino ereyga **this** iyo dhibic. Aynu kasii socono.

```

1 public class Person {
2
3     private int id;
4     private String name;
5     ...
6
7     public Person(int id, String name) {
8         this.id = id;
9         this.name = name;
10    }
11    ...
12 }

```

Tusaale 5.8: Ereyga **this** iyo isticmaalkiisa. Ereygaasi wuxu tilmaamayaa kolba walaxda markaas la samaynaayo ama ficilkeeda la kaxaynaayo, tusaale ahaan sawirka 5.1 marka person1 la samaynaayo wuxu tilmaamayaa isaga, marka cigal la samaynaayana wuxu tilmaamayaa walaxda cigal.

5.6 Sharaxista

Sharaxista ama *comment* waxad ku arki kartaa meelo badan oo ka mid ah qoraalka borogaraamka. Ujeedadoodu waa iney sharaxaad ka bixiyaan dhismaha borogaraamka si fahamkiisu u fududaado. Waxa muhiim ah inaad ogaato inaaney comments-ku ka qaybqaadan habka borogaraamku u shaqaynaayo ee kaliya ay yihiin sharaxaad.

```

1 package com.example;
2
3 /**
4  *
5  * Noocani wuxu matalayaa baabuurta
6  *
7  */
8 public class Car {
9     // model-ka gaadhiga
10    private String model;
11
12    /**
13     * Shirkadda sameysa
14     * gaariga
15     */
16    private String manufacturer;
17
18    /**
19     * ficil ku siinaaya model-ka gaadhiga
20     * @return model-ka gaadhiga
21     */
22    public String getModel() {
23        return this.model;
24    }
25
26    /**
27     * This method returns the speed of the car
28     * @return the speed of the car
29     */
30    public float getSpeed() {
31        float length = 120.0f; // km
32        float time = 1.2f; // 1.2 hours
33        // xisaabi xawaaraha
34        float speed = length/time;
35        return speed;
36    }
37 }

```

Tusaale 5.9: Sharaxaadda qoraalka borogaraamka (Java comments)

Waxaynu tusaale u soo qaadaneynaa borogaraamka tusaalaha 5.9 si aynu u aragno qaabka ay comments-ku u eegyihiin. Dhawr siyaabood ayaa loo qori karaa comments-ka, oo ay ka midyihiin kuwan:

1. Laymanka 9 iyo 33 waxa ku qoran sida inta badan comments-ka loo qoro. Comments-ka noocan oo kale ah waxay ku bilaabmaan calaamadda `//` illaa laynka dhammaadkiisa ayeyna ku eegyihiin
2. Laymanka 31 iyo 32 iyana waxa ku qoran comments, laakiin waxay kaga duwanyihiin kuwii hore in kuwani ay laynka kamid ah borogaraamka ay dhammaadkiisa ku qoranyihiin
3. Waxa jira laba siyaabood oo kale oo comments-ka loo qoro, gaar ahaan marka sharaxaaddaasi ka koobantahay inka badan hal layn oo qoraal ah. Waxad ka arkaysaa laymanka 3 illaa 7 oo sharaxaysa nooca Car, 12-15 oo sharaxaysa sifada manufacturer, 18-21 oo sharaxaysa ficilka `getModel` iyo 26-29 oo sharaxaad ka bixinaysa ficilka `getSpeed`.

Waxad halkan ku baratey siyaabooyinka kala duwan ee sharaxaadda borogaraam loo qorikaro. Waxa kale oo aad fahamtay in aaney sharaxaaddu ka qaybqaadaneynin habka borogaraamku u shaqaynaayo ee kaliya ay sharaxaad ka bixinayaan qaabka loo qoray si fahamkiisu u fududaado. Sharaxaadda ficilada tusaale ahaan ficilka `getSpeed` waxad ku arki kartaa `@return`, `@param` iyo qaar kale. Sharaxaadda noocan oo kale ah waxa loo yaqaan *javadoc*. Waxa jira borogaraamyo loogu talagalay inay fahmaan calaamada-haas iyo sharaxaadda qaabkaas u eeg, kuwaas oo sharaxaadda borogaraamka tusaale ahaan ka dhigaaya HTML si gooni loogu akhristo.

5.7 Doorsoomeyaasha

Waxaynu ku jirnaa calooshii luuqadda Java. Si ay inoogu fududaato cutub hoosaadkan iyo kuwa soo socdaba, waxaynu samaynaanaa talaabo yar oo aasaas u ah. Waxad samaysaa nooc cusub adiga oo raacaya talaabada (**NewClass**). Waxad u buuxbuuxisaa sidan: Source folder=nimi/src, Package=nimi, Name=IsbarJava. Kadibna waxad u eekaysiisaa qoraalka noocan tusaalaha 5.10.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {
6         // halkan ku qor wixii aad rabto inaad tijaabiso
7     }
8 }

```

Tusaale 5.10: Nooca IsbarJava oo aynu ugu talagalay ineynu wax ku tijaabino

Waxan rabaa inaan sharaxaad yar ka bixiyo ficilka **public static void main(String[] args)** ee laynka 5-aad ee nooca IsbarJava ee tusaalaha 5.10. Ficilkan waa mar labaadkii aynu aragno. Markii hore waxaynu kusoo aragnay borogaraamkii Hello World ee cutubka 2.3. Ficilkani waa albaabka laga soo galo borogaraam kasta oo Java ah. Waa meesha ay borogaraamkaas kaxayntiisu ka bilaabanto.

Su'aashu waxay tahay maxaynu u arkiwaynay ficilkan markii aynu qoraynay noocyada Person, PersonTest, Student iyo StudentTest? Maxaynu u arkiwaynay markii aynu kaxaynaynay noocyada tijaabada ah ee PersonTest iyo StudentTest? Jawaabtu waxay tahay noocyada tijaabada ah looma baahna in loo sameeyo ficilkaas albaabka ah, sababtoo ah JUnit ayaa kaxaynaaya, waxaanu ku garanayaa wixii ficilo ee tijaabo ah sumadda @Test.

Aan ka socdo. Waxad soo baratay sifooyinka nooca, kuwaas oo loogu talagalay in lagu kaydiyo qiimeyaasha kala duwan ee walxaha laga sameeyo noocaas. Doorsoomeyaasha (*variables*) inta badan waxa lagu isticmaalaa ficilada calooshooda, waana meel wax lagu kaydinkaro, inta ficilkaasi uu socdo. Dadka qaar ayaa sifooyinka nooca u yaqaan iyagana iney yihiin variables, laakiin inta badan iyaga waxa loo yaqaan fields ama attributes. Si haddaba aynu u fahamno doorsoomeyaashu waxay yihiin waxaynu soo qaadaneynaa laba siyaabood oo ay noqon karaan:

- local variables
- parameters

5.7.1 Local variables

Local variables-ku sidoodaba waxay ku bilaabmaan sidan oo kale:

```
TypeName variableName[ = VariableValue];
```

Waliba inta xerada [...] ku jirtaa qasab ma aha. Marka hore *variableName* waxay noqon kartaa magacaad rabto, waana inaad xaraf yar ku bilawdo sida sifooyinka oo kale. *TypeName* waa inuu sidii sifooyinka oo kale laba mid noqdo: Nooc sida: String, Object, Door, Person, iwm ama mid ka mid PDT-ga Java. Kadib waxa imanaysa calaamadda = oo loo yaqaano *assignment operator* oo aynu gadaal kaga hadlidoono. Intaas kadib waxa imanaaya *VariableValue* oo ah qiimihii doorsoomehaas. Borogaraamka 5.11 ayaa ku tusaaya isticmaalka local variables-ka.

```

1 package nimi;
2
3 import nimi.person.Person;
4
5 public class IsbarJava {
6
7     public static void main(String[] args) {
8         // tusaalooyin local variables ah
9         byte bt = 23;
10        short sh = 7;
11        int weight = 225;
12        float speed = 1.10f;
13        boolean gameOver = false;
14        String message = "Hello, Java!";
15        Person person1 = new Person(2, "Sahra Ahmed");
16        Person cigal = new Person(123, "Cigal Shiidaad");
17
18        System.out.println("bt=" + bt);
19        System.out.println("sh=" + sh);
20    }
21 }

```

```

20     System.out.println("weight=" + weight);
    System.out.println("speed=" + speed);
22     System.out.println("gameOver=" + gameOver);
    System.out.println("message=" + message);
24     System.out.println("Magaca person1 waa: "
        + person1.getName());
26     System.out.println("ID nambarka Cigal waa: "
        + cigal.getId());
28 }
}

```

Tusaale 5.11: Borogaraam ku tusaaya isticmaalka local variables-ka

Tusaalaha 5.11 waxad ku arkeysaa siddeed tusaale oo local variables ah iyo isticmaalkooda. Markaad rabto inaad variable cusub sameyso, waa inaad qeexdo magaceeda (sida: *bt*, *sh*, *weight*, iwm eeg laymanka 9-16). Waxa fiican inaad magac fiican oo la fahmikaro isticmaasho, oo aanad sida tusaalahan magacyo gaagaaban oo laba xaraf ka kooban isticmaalin. Magaca kadib waa inaad qeexdo *TypeName*-keeda oo ah nooca variable-ka (sida: *byte*, *short*, *int*, *Person*, iwm eeg laymanka 9 illaa 16). Magaca waxa ka horeeya *TypeName*-ka. *TypeName*-ku wuxu sheegayaa nooca macluumaadka ay kaydinkarto variable-ku ama doorsoomuhu. Haddii aad kaxeeyso (RunJava) borogaraamka 5.11 waxa kuusoo baxaaya (oo aad ka arkaysa Console-ka):

```

bt=23
sh=7
weight=225
speed=1.1
gameOver=false
message=Hello, Java!
Magaca person1 waa: Sahra Ahmed
ID nambarka Cigal waa: 123

```

Laymanka 18 illaa 27 waxad ka arkaysaa `System.out.println(...)` oo ah ficil loo isticmaalo marka la rabo in isticmaalaha la tuso fariin.

Intaynaan hore u socon doorsoomayaasha *TypeName*-koodu yahay mid ka mid ah PDT iyo kuwa uu yahay noocyada (class) mid, waxay ku kala duwanyihiin waxyar. Arintaasi waxay u baahantahay in sharaxaad laga bixiyo. Sawirka 5.2 ayaan isku dayayaa inaan arintan ku muujiyo. Sida sawirka ka muuqata doorsoomayaasha PDT-ga ah waxa la odhankaraa qiimahooda waxay ku haystaan caloosha, laakiin kuwa kale qiimuhu wuxu ka yaal banaanka oo waa nooc minal tilmaan ama hogaan ah. Kaladuwanaanshahani inta badan wuxu soo ifbaxayaa marka laga hadlaayo *parameters* oo aynu kaga hadlayno hoos.

5.7.2 Parameters

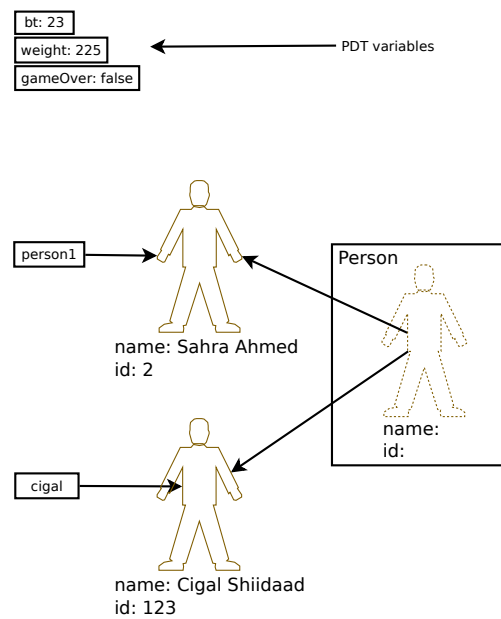
Parameters-ka haddii aan iyana tusaale ka bixiyo, waxad ka arkaysaa borogaraamka 5.12.

```

package nimi;
2
import nimi.person.Person;
4
public class IsbarJava {
6
    public static void main(String[] args) {
8        tusAkhbaartaQofka("Muuse", 200);
        tusAkhbaartaQofka("Maxamuud", 180);
10        System.out.println("-----");
        int dhererka = 200;
12        Person cigal = new Person(123, "Cigal Shiidaad");
        System.out.println("Ka HOR wicista ficilka badal(...)!");
14        System.out.println("dhererka = " + dhererka);
        System.out.println("cigal.id = " + cigal.getId());
16        System.out.println("cigal.name = " + cigal.getName());
        badal(dhererka, cigal);
18        System.out.println("Ka DIB wicista ficilka badal(...)!");
        System.out.println("dhererka = " + dhererka);
20        System.out.println("cigal.id = " + cigal.getId());
        System.out.println("cigal.name = " + cigal.getName());
22    }

24    /**
     * Ficil magaciisu yahay af-soomaali ah oo leh laba parameter
26    */
    private static void tusAkhbaartaQofka(String magacaQofka,

```



Sawirka 5.2: Kaladuwanaanshaha doorsoomeyaasha PDT-ga ah iyo kuwa kale

```

28     int dhererkaQof) {
29         System.out.println(magacaQofka + " dhererkiisu waa "
30             + dhererkaQof + "cm.");
31     }
32
33     /**
34      * Ficil kale oo isna leh laba parameter
35      */
36     private static void badal(int dhererkaQofka, Person qof) {
37         dhererkaQofka = 1000;
38         System.out.println("**Caloosha badal(...) dhererkaQof = "
39             + dhererkaQofka);
40         qof.setId(100002);
41         qof.setName("Nin la yaqaan");
42     }
43 }

```

Tusaale 5.12: Borogaraam ku tusaaya isticmaalka parameters-ka

Borogaraamka 5.12 markaad kaxayso waxa kuusoo baxeysa:

```

Muuse dhererkiisu waa 200cm.
Maxamuud dhererkiisu waa 180cm.
-----
Ka HOR wicista ficilka badal(...)!
dhererka = 200
cigal.id = 123
cigal.name = Cigal Shiidaad
**Caloosha badal(...) dhererkaQof = 1000
Ka DIB wicista ficilka badal(...)!
dhererka = 200
cigal.id = 100002
cigal.name = Nin la yaqaan

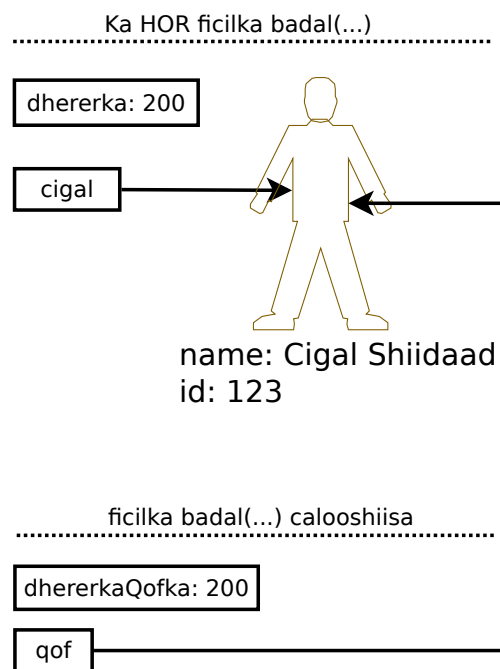
```

Tusaalaha 5.12 wuu yara dheeryahay, laakiin wuu fududyahay. Ficilka main() waynu naqaan, sababtoo ah waynu kasoo hadalay. Tusaalaha waxad ka arkaysaa laba ficil oo kale oo kala ah tusAkhbaartaQofka

iyo badal. Labada ficilba magacoodu waa af-soomaali taas oo ku tusaysa inaad borogaraamkaaga ku qori karto af-soomaali. Laymanka 24 illaa 31, waxad ka arkaysaa ficilka tusAkhbaartaQofka. Ficilkaasi wuxu leeyahay laba parameter, oo magacyadoodu kala yihiin magacaQofka (oo noocedu yahay String) iyo dhererkaQof (oo noocedu yahay int). Ficilkaas tusAkhbaartaQofka calooshiisa waxaynu isticmaalaha tusaynaa fariin ka kooban qoraal iyo qiimeyaasha labada parameter.

Haddaba markasta oo aynu rabno inaynu kaxayno ama isticmaalo ficilka tusAkhbaartaQofka waa inaynu qiime siino labadaas parameter. Laynka 8-aad waxaynu ficilka u gudbinaynaa qiimeyaasha "Muuse" iyo 200 oo ku kalabeegmaaya labada parameter. Laynka xiga ee 9-aad waxaynu sidoo kale ficilka u gudbinaynaa laba qiime oo kala ah "Maxamuud" iyo 180. Ficilka tusAkhbaartaQofka wuxu inna tusayaa hal layn oo qoraal ah oo uu ku jiro qiimeyaashii loo soo gudbiyey. Inta badan qiimeyaasha loo gudbinaayo ficilka ee sida kuwa laymanka 8 iyo 9 waxa loo yaqaan *arguments*.

Laga bilaabo laynka 11-aad ee borogaraamka 5.12 wuxu ku tusayaa waxay ku kala duwanyihiin doorsoomeyaasha ah PDT-ga iyo kuwa aan ahayn. Waa arintaynu qaybtii hore ku dulmarnay. Waxa kuu iftiiminaaya sawirka 5.3.



Sawirka 5.3: Kaladuwanaanshaha doorsoomeyaasha PDT-ga ah iyo kuwa kale markay timaado ficilada

Aan sii sharaxo borogaraamka 5.12. Laymanka 13-16 waxay ku tusayaan qiimeyaasha labada doorsoome ee laymanka 11-12, kahor intaan la wicin ficil badal ee laynka 17-aad. Laymanka 18-21 waxay ku tusayaan isla qiimeyaasha labadii doorsoome kadib wicista ficilka badal. Halkaas waxad ka arkaysaa in qiime badalka uu ficilka badal ku sameeyey doorsoomaha dhererka aaney muuqanaynin, laakiin badalka uu ku sameeyay doorsoomaha cigal ay muuqanayso. Halkaas waxaynu ka arkaynaa in wixii argument ee PDT ah in ficilka loo gudbinaayo qiimaha taas oo luuqadda Java loo yaqaano *pass by value*, laakiin arguments-ka noocyada ah waxa ficilka loo gudbinayaa hogaanka ama tilmaanta taas oo loo yaqaano *pass by reference* ee sawirka 5.3.

Muhiimadda uu tusaalaha sare is leeyahay fahamsii akhristaha waxa weeye caloosha ficilka wixii ka dhacaaya iyo sida uu ugu muuqanaayo dibedda ficilka. Ficilka waxad u sawiran kartaa sida qol madaw oo kale. Meesha kaliya ee qolkaas laga gali karo waa albaabka oo u dhigmaaya parameters-ka ficilka. Luuqadda Java waxa suurtoagal ah in wixii caloosha ficilka ka dhacaa ay u muuqdaan borogaraamka wacaaya ficilkaas. Haddii ficilka loo gudbiyo qiimeyaal PDT ah, dibedda ficilka uma muuqanayso wax kasta oo badal ah uu ficilku ku sameeyo qiimeyaashaas, laakiin haddii ficilka loo gudbiyo walxo wixii badal ah ee uu ficilkaasi ku sameeyo walxahaas waxay ka muuqanaysaa dibedda ficilkaas. Inta badan waa arin loo baahanyahay in ficiladu wax ka badalaan walxaha loo soo gudbiyo.

5.8 Qiimaha null

Waxa jira qiime gaar ah oo luuqadda Java leedahay, kaas oo la yidhaahdo **null**. Qiimahaas waxa heli kara doorsoomeyaasha iyo sifooyinka aan ahayn PDT.

```

1 package nimi;
2
3 import nimi.person.Person;
4
5 public class IsbarJava {
6
7     public static void main(String[] args) {
8         ...
9         int tijaabo = null; // Waa qalad!
10        Person person1 = null; // Waa sax!
11        ...
12    }
13 }

```

Tusaale 5.13: Qiimaha **null**

Sida ka muuqata 5.13 doorsoomaha nooceedu yahay PDT ma siinkartid qiimaha **null**. Borogaraamka Eclipse tusaale ahaan wuxu ku tusayaa casaan laynkaas ah. Laakiin doorsoomeyaasha aan ahayn PDT waxad siin kartaa qiimaha **null**, taas oo ka dhigan inaanu jirin walax ay doorsoomahani ama sifadani tilmaamayso.

5.9 Operators

Doorsoomeyaashu kaligood innaguma filla. Waxa loo baahanyahay in macluumaadka ku kaydsan doorsoomeyaasha wax lagu xisaabin karo. Waxa hawshan qabanaaya *operators*-ka. Waxaynu bilaabaynaa inaynu ka hadallo kuwa ugu muhiimsan.

5.9.1 Assignment operator

Operators-ka waxa ugu horeysa *Assignment operator* oo aad loo isticmaalo badaleysana qiimaha doorsoomaha ama sifada ku kaydsan, waxaanad ka arkaysaa miiska 5.3. Meelo badan oo hore ayaynu kusoo aragnay isticmaalista operator-kan, laakiin illaa hadda wax sharaax ah kamaynaan bixin.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {
6         int speed = 0;
7         int gear = 1;
8         System.out.println("Speed = "+speed);
9         System.out.println("Gear = "+gear);
10        gear = 4;
11        System.out.println("Gear = "+gear);
12    }
13 }

```

Tusaale 5.14: Borogaraam ku tusaaya isticmaalka assignment operator-ka

Miiska 5.3: Assignment operator

Calaamadda operator-ka	Sharaxaad
=	Operator-kan ama calaamad-dan magaceeda waxa la yidhaahdaa assignment operator, waxaana loogu tala-galay in qiimaha ku kaydsan variable-ka ama sifada lagu badalo

Borogaraamka 5.14 ayaa ku tusaaya isticmaalka assignment operator-ka. Sharaxaad ku saabsan borogaraamka 5.14:

1. Laynka 6-aad waxaynu badalaynaa qiimaha ku kaydsan doorsoomaha ama variable-ka speed oo aynu ku badalaynaa 0. Variable-kani hore umay jirin, sidaas awgeed waa qiimaheedii ugu horeeyey
2. Laynka 7-aad sidoo kale waxaynu variable-ka gear siinaynaa qiimaha 1
3. Intaas kadib waxaynu isticmaalaha tusaynaa qiimayaasha ku kala kaydsan labada variable ee speed iyo gear
4. Laynka 10 waxaynu badalaynaa qiimaha ku kaydsan gear oo aynu ka dhigaynaa 4. Intaas kadib waxaynu isticmaalaha tusaynaa qiimaha ku kaydsan gear.

Haddii aad kaxayso borogaraamka 5.14 waxa kuusoo baxayaa:

```
Speed = 0
Gear = 1
Gear = 4
```

Waxa kale oo calaamaddan iyo kuwa soosocda loo isticmaali karaa sifooyinka nooca. Waxad tusaale u soo qaadankartaa ficilada setId, setName iyo setPhone ee nooca Person, kuwaas oo aad ka arkayso sidaad u badali karto qiimaha ku kaydsan sifooyinka.

5.9.2 Arithmetic operators

Waxa ku xiga operators-ka nooca ugu fudud uguna badan dhanka isticmaalka oo loo yaqaano *Arithmetic operators*, waxaanad ka arkaysaa miiska 5.4.

Miiska 5.4: Arithmetic operators

Calaamadda operator-ka	Sharaxaad
+	additive operator - isugeyso
-	subtraction operator - kalagoyso
*	multiplication operator - iskudhufasho
/	division operator - qaybiso
%	remainder operator - hadhaaga qaybiska kadib

Arithmetic operators-ka waxa loogu talagalay xisaabta sida isugaynta, kalagoynta, iwm. Arithmetic operators-ku dhammaantood waxay u baahanyihiin laba qiime oo loo yaqaanno *operands*. Aynu tusaale u soo qaadano borogaraam yar oo fudud oo aad ka arkayso 5.15, kaas oo ku tusaaya isticmaalka operators-kan.

```
1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {
6         int jawaab = 1 + 2;
7         System.out.println("jawaab = " + jawaab);
8         int jawaab2 = jawaab - 1;
9         System.out.println("jawaab2 = " + jawaab2);
10        System.out.println("jawaab * jawaab2 = "
11                               + jawaab * jawaab2);
12        System.out.println("jawaab % jawaab2 = "
13                               + jawaab % jawaab2);
14        System.out.println("(jawaab * jawaab2)/jawaab2 = "
15                               + (jawaab * jawaab2) / jawaab2);
16    }
17 }
```

Tusaale 5.15: Borogaraam yar oo ku tusaaya isticmaalka arithmetic operators-ka. Laynka 7-aad calaamadda lagu isticmaalay ee + waxay isku dhajinaysaa laba qoraal ee ma aha iskugeynta xisaabta ee miiska sare ku jirta oo dib ayaynu kaga hadlaynaa.

Borogaraamka sare ee 5.15 haddii aad kaxayso waxa kuusoo baxayaa:

```

jawaab = 3
jawaab2 = 2
jawaab * jawaab2 = 6
jawaab % jawaab2 = 1
(jawaab * jawaab2)/jawaab2 = 3

```

Aan sharaxaad yar ka bixiyo borogaraamka 5.15:

1. Laynka 6-aad waxaynu samaynaynaa doorsoome cusub oo af-soomaali ah oo magaceeda la yidhaahdo jawaab, iyada oo aynu ku kaydinayno qiimaha jawaabta 1+3 oo ah 4
2. Laynka xiga ee 7-aad waxaynu isticmaalaha tusaynaa qiimaha ku kaydsan doorsoomaha jawaab
3. Intaas kadib laynka 8-aad waxaynu samaynaynaa doorsoome cusub oo iyana af-soomaali ah oo magaceeda la yidhaahdo jawaab2. Iyana waxaynu ku kaydinayaa qiimaha kasoo baxa marka qiimaha ku kaydsan jawaab laga jaro 1
4. Intaas kadib waxaynu isticmaalaha tusaynaa qiimaha ku kaydsan jawaab2
5. Laynka 10-aad (ee ku dhammaanaaya laynka 11-aad) waxaynu isticmaalaha tusaynaa qiimaha jawaabta kasoo baxda marka laysku dhufto qiimaha ku kaydsan jawaab iyo jawaab2
6. Intaas kadib waxaynu isticmaalaha tusaynaa isticmaalka operator-ka %, taas oo noqonaysa jawaab - 1 x jawaab2=1
7. Laymanka 14-15 waxaynu isticmaalaha tusaynaa ka shaqaynta xisaab yara adag oo waxaynu tusaynaa jawaabta kasoo baxda xisaabta $\frac{jawaab+jawaab2}{jawaab2}$. Si isugeyntu u horayso, waa inaad isticmaasho xerooyinka ().

5.9.3 Unary Operators

Waxa xiga *Unary Operators*-ka oo ka duwan kuwii hore. Halka kuwii hore ay u baahanyihiin laba qiime, dhammaan kuwani waxa la siinayaa qiime kaliya. Ereyga Unary wuxu ka dhiganyahay qur ama mid.

Miiska 5.5: Unary operators

Calaamadda operator-ka	Sharaxaad
+	unary plus operator - waxay sheegaysaa qiime positive ah (zero ama ka weyn)
-	unary minus operator - waxay sheegaysaa qiime negative ah
++	increment operator - midh ku dar
--	decrement operator - midh ka jar
!	logical compliment operator - waxay gadiyeysaa qiimaha <i>boolean</i> -ka

Borogaraamka 5.16 ayaa ku tusaaya isticmaalka calaamadaha Unary opertors-ka ee sare.

```

1 package nimi;
3 public class IsbarJava {
5     public static void main(String[] args) {
6         int value1 = +100;
7         int value2 = -10;
8         System.out.println("value1 + value2 = "
9             + (value1 + value2));
10        value1--;
11        value2++;
12        System.out.println("value1 = " + value1);
13        System.out.println("value2 = " + value2);
14        boolean ON = true;
15        System.out.println("ON = " + ON);
16        System.out.println("!ON = " + !ON);

```

```
17 }
}
```

Tusaale 5.16: Borogaraam ku tusaaya isticmaalka unary operators-ka

Borogaraamka 5.16 haddii aad kaxayso waxa kuusoo baxaaya:

```
value1 + value2 = 90
value1 = 99
value2 = -9
ON = true
!ON = false
```

Sharaxaad ku saabsan borogaraamka 5.16:

1. Laymanka 6 iyo 7 waxaynu samaynaanaa laba doorsoome ama variable oo magacyadooda la kala yidhaahdo value1 iyo value2, oo labadoodaba noocodu yahay **int**
2. Laynka 8-aad (oo ku dhammaanaaya laynka 9-aad) waxaynu isticmaalaha tusaynaa qoraalka "**value1 + value2 =** " oo lagu dabaqabtay ama lagu nuday natiijada kasoo baxda marka laysu geeyo qiimeyaasha ku kala kaydsan labada doorsoomeyaal ee value1 iyo value2
3. Laynka 10-aad waxaynu mid ka jaraynaa ama ka dhimaynaa qiimaha ku kaydsan value1, kadib wixii soo baxay ayaynu dib ugu kaydinaynaa variable-ka value1 (waxa inoo fulinaysa operator-ka - -)
4. Kadib laynka 11, waxaynu isticmaaley operator-ka ++ taas oo keenaysa iney qiimaha ku jira value2 mid ku darto, kadibna natiijada soo baxda dib ugu kaydiso isla value2
5. Laynka 12 waxaynu isticmaalaha tusaynaa qoraalka "**value1 =** " oo lagu nuday qiimaha ku jira value1
6. Laynka 13 sidoo kale waxaynu isticmaalaha tusaynaa qoraalka "**value2 =** " oo lagu nuday qiimaha ku kaydsan value2
7. Laynka 16 waxad ka arkaysaa isticmaalka calaamadda ! ee luuqadda Java.

5.9.4 Equality and Relational Operators

Intaa waxa xiga *Equality and Relational Operators* oo aad ka arkayso miiska 5.6, oo iyagu isbarbardhi-gaaya laba qiime (ama laba doorsoome), kadibna soo celinaaya qiime boolean ah sida **true** ama **false**. Operators-ka noocan ah inta badan waxa lala isticmaalaa **if**-statement-ka oo aynu dhakso u barandoona.

```
package nimi;
2
public class IsbarJava {
4
    public static void main(String[] args) {
6        int value1 = 1;
        int value2 = 2;
8        if (value1 == value2) {
            System.out.println("Waa isku mid labada qiime");
10        }
        if (value1 != value2) {
12            System.out.println("Ma aha isku mid labada qiime");
        }
14        if (value1 < value2) {
            System.out.println("Waxa yar qiimaha ku kaydsan value1");
16        }
        if (value1 > value2) {
18            System.out.println("Waxa yar qiimaha ku kaydsan value2");
        }
20    }
}
```

Tusaale 5.17: Borogaraam ku tusaaya isticmaalka Equality and Relational Operators-ka

Aynu soo qaadano borogaraam yar oo inna tusaaleynaaya isticmaalka operators-ka noocan ah. Waxad ka arkaysaa borogaraamkan tusaalaha 5.17.

Borogaraamka 5.17 haddii aad kaxayso waxa kuusoo baxaysa:

```
Ma aha isku mid labada qiime
Waxa yar qiimaha ku kaydsan value1
```

Miiska 5.6: Equality and Relational Operators

Calaamadda operator-ka	Sharaxaad
==	equal to - iskumid
!=	not equal to - aan iskumid ahayn
>	greater than - kaweyn
>=	greater than or equal to - kaweyn ama lamid ah
<	less than - kayar
<=	less than or equal to - kayar ama lamid ah

Aynu sharaxaad yar oo kooban ka bixino borogaraamka 5.17 intaynaan hore u socon:

1. Laymanka 6 iyo 7 waa sidoo borogaraamkii hore ee 5.16, kaliya waxa ka duwan qiimaha labada doorsoome
2. Laynka 8-aad wuxu leeyahay haddii qiimeyaasha value1 iyo value2 ay isku mid yihiin, tus fariinta laynka 9-aad isticmaalaha
3. Laynka 11-aad wuxu leeyahay haddii qiimeyaasha value1 iyo value2 ay kala duwanyihiin, tus fariinta laynka 12-aad isticmaalaha
4. Sidoo kale laynka 14-aad wuxu leeyahay haddii qiimaha value1 ka yartahay qiimaha value2, tus fariinta laynka 15-aad isticmaalaha
5. Isla sidoo kale laynka 17-aad wuxu leeyahay haddii qiimaha value1 ka wayntahay qiimaha value2, tus fariinta laynka 18-aad isticmaalaha.

5.9.5 Conditional Operators

Waa qaybtii ugu dambaysay ee operators-ka waxaana loo yaqaan guruubkan *Conditional operators*, waxaanad ka arkaysaa miiska 5.7.

Miiska 5.7: Conditional operators

Calaamadda operator-ka	Sharaxaad
&&	conditional-AND - AND shardiyaaysan
	conditional-OR - OR shardiyaaysan

Borogaraamka 5.18 wuxu ku tusayaa isticmaalka conditional operators-ka.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {
6         int value1 = 1;
7         int value2 = 2;
8         if ((value1 == 1) && (value2 == 2)) {
9             System.out.println("value1 is 1 AND value2 is 2");
10        }
11        if ((value1 == 1) || (value2 == 1)) {
12            System.out.println("value1 is 1 OR value2 is 1");
13        }
14    }
15 }
```

Tusaale 5.18: Borogaraam ku tusaaya isticmaalka Conditional operators-ka

Haddii aad kaxayso borogaraamka 5.18 waxa kuusoo baxaaya:

```

value1 is 1 AND value2 is 2
value1 is 1 OR value2 is 1
```

Aan sharaxaad yar ka bixiyo borogaraamka 5.18 intaynaan hore u socon:

1. Laymanka 6 iyo 7 waa sidii hore
2. Laynka 8-aad wuxu leeyahay haddii qiimaha value1 yahay 1 IYO qiimaha value2 tahay 2, tus fariinta laynka 9-aad isticmaalaha
3. Laynka 11-aad wuxu leeyahay haddii qiimaha value1 yahay 1 AMA qiimaha value2 yahay 1, tus fariinta laynka 12-aad isticmaalaha.

Sababta loo leeyahay way shardiyeysanyihiin labadan calaamadood waa maxay? Sidaad arkaysaba laba calaamadoodba waxa la siinayaa laba qiime. Si haddaba loo ogaado natiijada calaamadan waxa marmar innagu filan kaliya inaynu ogaano qiimaha hore wuxu yahay. Haddii tusaale ahaan qiimaha hore noqdo **true** (sida **true** || ...) natiijada calaamaddu waxay noqonaysaa **true**. Intaas ayaynu kaga gudbaynaa operators-ka.

5.10 Expressions, Statements iyo Blocks

Mar haddii aad baratey doorsoomeyaasha iyo operators-ka, waxad halkan ku barandoontaa saddex ari-mood oo kala ah *expressions*, *statements* iyo *blocks*. Waxaynu ka bilaabeynaa sharaxaad ku saabsan statements-ka iyo waxay yihiin. Kadib waxaynu ku xigsiinaynaa kuwa kale. Haddaba waa maxay statement? Statement-ku wuxu u dhigmaa inta badan jumladaha ama weedha luuqadda caadiga ah (luuqadda ingiriisiga waa sentence). Statement kastaaba waa unug ama talaabo dhamaystiran, waxaana u calaamad ah inuu ku dhamaado calaamadda ';' (oo loo yaqaano *semi colon*). Statement-ka waxa la odhan karaa waa talaabooyinka uu borogaraamka ka koobanyahay, sidaas awgeed waxan marmarka qaar adeegsanayaa ereyga talaabo aniga oo uga jeeda statement. Borogaraamka 5.19 ayaa ku tusaaya tusaalooyin ku saabsan statements-ka.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {
6         int qiime = 23;
7         qiime--;
8         tusQiimaha(qiime);
9         tusQiimaha(0);
10        tusQiimaha(-10);
11    }
12
13    private static void tusQiimaha(int qiime) {
14        if (qiime > 0) {
15            System.out.println("Qiimuhu waa " + qiime + " shilin.");
16        } else if (qiime == 0) {
17            System.out.println("Waa bilaash!");
18        } else {
19            System.out.println(
20                "Aduunka wali laguma arag " +
21                "shay ama arin qiimihiisu " +
22                "yahay negative. Waa inaad siiso " +
23                "-qiime + " shilin qofka wax kaa iibsanaaya!");
24        }
25    }
26 }

```

Tusaale 5.19: Borogaraam ku tusaaya statements-ka

Imisa statement ayaad kasoo helaynaa borogaraamka 5.19?

Jawaabtu waa sideed. Laymanka 6 illaa 10, laynka 15, laynka 17 iyo laynka 19 (oo ku dhamaada laynka 23-aad).

Guruub statement-yo ah haddii la rabo in laysu uruuriyo oo tusaale ahaan la kaxeeyo haddii ay xaalad timaado ama laga boodo dhammaantood, waxa la isticmaalaa calaamadaha {...}. Block waxa loo yaqaan calaamadahaas iyo wax kasta oo ay caloosha ku haystaan. Imisa block ayaad ka soo helaynaa borogaraamka 5.19? Jawaabtu waa saddex. Laynka 15 waa block ka kooban hal statement, sidoo kale laynka 17 iyo 19. Inta badan ficilada iyo noocyada laguma tiriyo blocks-ka, inkastoo ay buuxinayaan shuruudihii block-ka.

Borogaraamka 5.19 haddii aad kaxayso waa inay kuusoo baxdo fariintan:

Qiimuhu waa 22 shilin.
Waa bilaash!
Aduunka wali laguma arag shay ama arin qiimihiisu ...

Statement-ka haddii lasii kala qaado wuxu u kala baxayaa waxa loo yaqaano *expressions*. Expression-ku isna wuxu ka koobanyahay doorsoomeyaal (ama sifooyin) iyo operators-ka aynu kor kusoo baranay. Expressions-ka inta badan sidaas uma muhiim aha, oo looma baahna inaad gooni u barato. Intaas ayaa kaaga filan inaad ka ogaato.

5.11 Control Flow Statements

Kumbuyuutarku marka uu kaxeynaayo borogaraamka Java, wuxu kaga bilaabayaa statement-ka ugu horeeya ee ficilka **public static void** main(String[] args), kadibna ka ku xiga, kadibna ka ku xiga ..., sidaas ayay kaxayntu ku soconaysaa illaa inta uu ka dhammaanaayo borogaraamku. Shayga ugu muhiimsan kumbuyuutarka ee loogu talagalay inuu kaxeeyo borogaraamyada waa processor-ka. Waxa dhicikarta in marmarka qaar loo baahdo in statement-ka qaar laga boodo ama qaar lagu celceliyo, waxaana arintan loo adeegsadaa *Control Flow Statements*, kuwaas oo keenaya in kumbuyuutarku talaabooyinka qaar ka boodo ama qaar ku laablaabto, isaga oo eegaaya xaalad ama shardi (condition).

Qaybtan waxad ku barandoontaa control flow statements-ka shardiga leh (*if*, *if-else*, *switch*) iyo kuwa laablaabta (*for*, *while*, *do-while*).

5.11.1 If statement

If waxaynu kusoo aragnay meelo badan, oo ka mid ah borogaraamyadii aynu illaa hadda soo qornay. *If* aad ayey u fududahay fahamkeedu. Waxa loogu talagalay in guruub talaabooyin ah shardi lagu xidho, shardigaas oo qiime boolean (**true** ama **false**) ah keenaaya marka la cabiro. Boolean waxad kusoo aragtay 5.3 markaynu ka hadlaynay PDT-ga Java. Haddii shardigu noqdo **true**, waxa la kaxaynayaa dhammaan talaabooyinka ku jira caloosha if-ka, haddii kale waa laga boodayaa dhammantood.

```

2  if (condition ama shardi) {
    //haddii shardigu noqdo true wixii
    //statement ah ee block-kan ku jira kaxee (run)
4  }
```

Tusaale 5.20: If statement

Isticmaalka *if* waxad ka arki kartaa borogaraamkii ugu dambeeyey ee 5.19, gaar ahaan laymanka 14-16, kaas oo shuruuddiisu tahay qiime > 0 oo ka dhigan haddii qiimuhu ka waynyahay eber. Haddii shuruuddaasi rumowdo (oo **true** noqoto), waxa isticmaalaha la tusayaa fariinta laynka 15-aad. Sidoo kale borogaraamka 5.17 ayaad kusoo aragtay tusaalooyin dhawr ah oo ku tusaaya if-ka iyo isticmaalkiisa.

5.11.2 If-else statement

If-else waa if-kii oo lagu korodhiyey qayb kale oo *else* la yidhaahdo. Qaybta *else* waxa loogu talagalay in la kaxeeyo haddii shardigu aanu rumoobin oo **false** noqdo. *If-else* qaabka uu u eegyahay waxad ka arkaysaa tusaalaha 5.21.

```

2  if (condition ama shardi) {
    //haddii shardigu noqdo true wixii
    //statement ah ee block-kan ku jira kaxee
4  } else {
    //haddii shardigu noqdo false wixii
6    //statement ah ee block-kan ku jira kaxee
    }
```

Tusaale 5.21: If-else statement

If-else aad ayuu fududahay markaas sharaxaad dheeraad ah inoogama baahna.

5.11.3 If-else if-else statement

If-else if-else waxay kusii dhismaysaa qaybtii hore ee *if-else*. *If-else if-else* waxay inoo saamaxaysaa inaynu isticmaalo shardiyo ka badan mid oo laba ah ama ka badan.

```

1  if (shardi1) {
    //haddii shardi1 noqdo true wixii
3    //statement ah ee block-kan ku jira kaxe
    } else if (shardi2) {
    //haddii shardi2 noqdo true wixii
5    //statement ah ee block-kan ku jira kaxe
    } else {
    //haddii kale oo aanu shardiyada midna true noqon
9    //wixii statement ah ee block-kan ku jira kaxe
    }

```

Tusaale 5.22: If-else if-else statement

If-else if-else waxad ka arkaysaa tusaalaha 5.22. Haddii shardiyadu ka bataan laba waa in la kordhiyo qaybta else if oo shardi kasta loo sameeyo else if u gaar ah. Isticmaalka if-else if-else waxad ka arkaysaa borogaraamka 5.19 gaar ahaan laymanka 14 illaa 24.

5.11.4 Switch statement

Switch wuxu u dhigmaa if-else if-else, laakiin waxooga ayuu ka hormarsanyahay. Waxaynu samaynaynaa borogaraam yar oo inna tusaaya isticmaalkiisa. Waxaynu samaynaynaa borogaraam yar kaas oo isticmaalaha waydiinaaya tiro u dhaxaysa 1 illaa 12, tiradaas oo kadhigan bilaha sanadka. Kadibna marka uu isticmaaluhu siiyo tiro, borogaraamku wuxu sheegayaa magaca bisha tiradaas leh oo af-ingiriisi ah. Haddii isticmaaluhu siiyo borogaraamka tiro ka baxsan 1-12, wuxuu borogaraamku tusayaa fariinta "Tirada bisha ee x waa qalad, waa inay u dhaxayso 1-12.", x wuxu ku badalayaa tirada uu isticmaaluhu siiyey borogaraamka. Xarafka 'Q' ayuu isticmaaluhu siinayaa borogaraamka marka uu rabo inuu ka baxo ama joojiyo borogaraamka, markaas oo borogaraamku tusaayo isticmaalaha qoraalka "Bye!" kadibna istaagaayo.

```

package nimi;

2  import java.util.Scanner;

4  public class IsbarJava {

6      public static void main(String[] args) {
9          String monthNumber = "";
          while (true) {
10             monthNumber = getMonthNumber();
             if ("q".equalsIgnoreCase(monthNumber)) {
12                 System.out.println("Bye!");
                 break;
14             } else {
                 displayMonthName(monthNumber);
16             }
          }
18     }

20     private static String getMonthNumber() {
        System.out.print("Isii tirada bisha > ");
22         Scanner sc = new Scanner(System.in);
        return sc.nextLine();
24     }

26     private static void displayMonthName(String mountNumber) {
        int number = 0;
28         String msgFormat = "Tirada bisha ee %s" +
            " waa qalad, waa inay " +
30             "u dhaxayso 1-12.";

        try {
32             number = Integer.parseInt(mountNumber);
        } catch (NumberFormatException nfe) {
34             System.out.println(String.format(msgFormat, mountNumber));
            return;
36         }

        switch (number) {
38         case 1:
            System.out.println("January");
            break;
40         case 2:
            System.out.println("February");
            break;
42         case 3:

```

```

    System.out.println("March");
46     break;
    case 4:
48     System.out.println("April");
        break;
    case 5:
50     System.out.println("May");
        break;
    case 6:
52     System.out.println("June");
        break;
    case 7:
54     System.out.println("July");
        break;
    case 8:
56     System.out.println("August");
        break;
    case 9:
60     System.out.println("September");
        break;
    case 10:
62     System.out.println("October");
        break;
    case 11:
64     System.out.println("November");
        break;
    case 12:
66     System.out.println("December");
        break;
    default:
68     System.out.println(String.format(msgFormat, mountNumber));
70     break;
72 }
74 }
76 }
78 }

```

Tusaale 5.23: Isticmaalka switch statement

Aynu waxyar oo sharaxaad ah ka bixino borogaraamka 5.23, kaas oo ay kusoo kordheen waxyaalo cusub.

1. Laymanka 7-18 waa ficilkii caanka ahaa ee albaabka borogaraamka ee main
2. Laynka 8 waxaynu samaynaynaa doorsoome cusub oo aynu u bixinay monthNumber, kaas oo noociisu yahay String
3. Laymanka 9-17 waxaynu galaynaa wareeg (**while**) aan dhammaanaynin (forever loop). Isticmaalka **while** waxaynu kaga hadlidoonnaa dib
4. Laynka 10-aad waxaynu isticmaalaynaa ficilka getMonthNumber, kaas oo qiimaha uu soo celiyo aynu ku kaydinayno doorsoomaha monthNumber. Ficilka getMonthNumber oo ku eeg laymanka 20-24, waxa loogu talagalay inuu isticmaalaha waydiiyo tirada bisha, kadib soo celiyo qiimaha uu isticmaaluhu siiyo oo noociisu yahay String
5. Laymanka 11-16 waa if-else statement oo aynu soo marnay, kaas oo shardiigiisu yahay laynka 11-aad, kaas oo sheegaaya haddii qiimaha variable-ka monthNumber yahay xarafka q ama Q (laynka 10-aad ayaynu qiimihii isticmaaluhu inna siiyey ku kaydinay variable-ka monthNumber) waxaynu kaxaynaynaa laymanka 12-13, haddii kale waxaynu kaxaynaynaa qaybta else oo caloosha ku haysata laynka 15 oo kaliya
6. Haddii isticmaaluhu borogaraamka siiyo xarfaha q ama Q, laynka 13-aad ee **break** wuxu keenayaa in **while**-ka laga soo boodo oo borogaraamku kasii socdo laynka 18-kaas kaas oo ah dhamaadkii, oo sidaas ku istaago borogaraamku
7. Haddii isticmaaluhu siiyo borogaraamka qiime ka duwan xarafka lagaga baxaayo ee q ama Q, waxa la wacayaa ama isticmaalayaa ficilka displayMonthName ee laynka 15, kaas oo la siinaayo qiimihii isticmaaluhu inna siiyey
8. Ficilka displayMonthName waxa loogu talagalay inuu isticmaalaha tuso magacaga bisha uu nambarkooda siiyey. Haddii nambarka bishu qaladyahay oo ka baxsan yahay 1-12, ficilkaasi wuxu isticmaalaha tusayaa fariin ku haboon. Ficilka displayMonthName laymanka 31-36 wuxu isku dayayaa inuu qoraalka isticmaaluhu siiyey oo noociisu yahay String u badalo nambar ah **int**, kaas oo uu ku kaydinaayo doorsoomaha number eeg gaarahaan laynka 32

9. Ficilka `displayMonthName` waxa ugu muhiimsan laymanka 37-77, halkaas oo aad ka arkayso isticmaalka **switch**. Qaabka uu u shaqaynaayo **switch** way fududahay. Wuxu eegayaa qiimaha variable-ka la siiyey ee laynka ugu horeeya ee 37, kadib wuxu ku boodayaa **case**-ka qiimahaas. Tusaale ahaan haddii variable-ka number qiimaheedu noqdo 5, wuxu ku boodayaa laymanka 50-52, halkaas oo isticmaalaha la tusaayo qoraalka "May". Haddii qiimaha variable-ka number loo waayo **case** ku haboon, waxa lagu boodayaa qaybta **default** ee laymanka 74-76, halkaas oo isticmaalaha la tusaayo fariinta laynka 28 oo meesha ay ku qorantahay "%s" lagu buuxiyey qiimihii uu isticmaaluhu siiyey borogaraamka
10. Qaabka guud ahaan **switch** u taalo waxad ka arkaysaa tusaalaha 5.24. Sawirka 5.4 waxad ka arkaysaa borogaraamka 5.23 oo la kaxeeyey.

```

switch (switchVariable) {
2  case X:
    // wax samee halkan haddii qiimaha
4    // switchVariable noqdo X
    break;
6  case Y:
    // wax samee halkan haddii qiimaha
8    // switchVariable noqdo Y
    break;
10 case Z:
    // wax samee halkan haddii qiimaha
12    // switchVariable noqdo Z
    break;
14 case ..
15 default:
    // wax samee halkan haddii qiimaha
16    // switchVariable uu noqonwaayo
18    // mid ka mid ah case-ka
    break;
20 }

```

Tusaale 5.24: Switch statement qaabka uu u yaalo guud ahaan. `switchVariable` nooceedu waa inuu noqdo `int` ama `enum`. Java 7 waxay soo kordhisay isticmaalka nooca `String`.

```

53  .... case 6:
54  .... System.out.println("June");
55  .... break;

```

Problems @ Javadoc Declaration Console

<terminated> IsbarJava [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2.5.2012 20.13.07)

```

Isii tirada bisha > 1
January
Isii tirada bisha > 12
December
Isii tirada bisha > 7
July
Isii tirada bisha > 19
Tirada bisha ee 19 waa qalad, waa inay u dhaxayso 1-12.
Isii tirada bisha > ABC
Tirada bisha ee ABC waa qalad, waa inay u dhaxayso 1-12.
Isii tirada bisha > q
Bye!

```

Sawirka 5.4: Borogaraamka 5.23 oo la kaxeeyey

5.11.5 While statement

Isticmaalka **while** waynu soo aragnay. Waxa loogu talagalay in guruub hal ama ka badan oo statement-yo ah lagu laalaabto inta shardigu yahay **true**. Marka shardiga **while** noqdo **false** ayaa laga baxayaa wareegga oo borogaraamku kasii soconayaa laynka ka hooseeya **while**-ka. Tusaalaha 5.25 ayaad ka arkaysaa qaabka uu yaalo **while**. Borogaraamka 5.26 ayaa ku tusaaya borogaraam yar oo isticmaalaaya **while**.

```

1 while (shardi) { // shardigu waa inuu qiime boolean ah keeno
2   // illaa iyo inta shardigu tahay true
3   // talaabooyinkan ku laablaabo
4 }
halkan ayuu borogaraamku kasii soconayaa marka wareegga laga baxo

```

Tusaale 5.25: While statement qaabka uu u yaalo guud ahaan

```

1 package nimi;
2
3 public class IsbarJava {
4
5   public static void main(String[] args) {
6     int count = 1;
7     while (count < 11) {
8       System.out.println("count is: " + count);
9       count++;
10    }
11  }
12 }

```

Tusaale 5.26: Borogaraam yar oo ku tusaaya isticmaalka while

Borogaraamka 5.26 markaad kaxayso waxa kuusoo baxaaya:

```

count is: 1
count is: 2
count is: 3
count is: 4
count is: 5
count is: 6
count is: 7
count is: 8
count is: 9
count is: 10

```

5.11.6 For statement

for waxay lamid tahay while, laakiin wax yar ayay dhisme ahaan ka duwantahay, waxaanad isticmaalkeeda ka arkaysaa borogaraamka 5.27.

```

1 package nimi;
2
3 public class IsbarJava {
4
5   public static void main(String[] args) {
6     for (int count = 1; count < 11; count++) {
7       System.out.println("count is: " + count);
8     }
9   }
10 }

```

Tusaale 5.27: Borogaraam yar oo ku tusaaya isticmaalka for

Borogaraamka 5.26 markaad kaxayso waxa kuusoo baxaaya:

```

count is: 1
count is: 2
count is: 3
count is: 4
count is: 5
count is: 6
count is: 7
count is: 8
count is: 9
count is: 10

```

Guud ahaan qaabka uu u yaallo for statement-ku oo sidoo kale loo yaqaano *for loop*, halka while-ka ka horeeya loo yaqaano *while loop* waa sidan:

```

1 for(bilawga;shardiga;talaabada) {
2     //illaa iyo inta shardigu tahay true halkan ku laablaabo
3 }
4 halkan ayuu borogaraamku kasii soconayaa marka wareegga laga baxo

```

Tusaale 5.28: For statement qaabka uu u yaalo guud ahaan

5.11.7 Foreach statement

foreach (**for** lasoo gaabiyey) waxa lagusoo daray luuqadda Java soosaarkeedii 5-aad. Waxaana loo isticmaalaa inta badan guruubka walxaha. Waxaanu u eegyahay sida tusaalaha 5.29.

```

1 for (Nooca obj: objects) {
2     // wax ku samee walaxda obj
3 }

```

Tusaale 5.29: Foreach qaabka uu u yaalo

Aynu borogaraam yar oo tusaale ah ku xigsiino, kaas oo aad ka arkayso 5.30.

```

1 package nimi;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import nimi.person.Person;
7
8 public class IsbarJava {
9
10     public static void main(String[] args) {
11         List<Person> dad = new ArrayList<Person>();
12         dad.add(new Person(1, "Cigal"));
13         dad.add(new Person(2, "Sahra"));
14
15         for (Person qof : dad) {
16             System.out.println(qof.getName());
17         }
18     }
19 }

```

Tusaale 5.30: borogaraam ku tusaaya isticmaalka for each

Laymanka 15-17 ayaad ka arkaysaa isticmaalka wareegga foreach.

5.11.8 Break iyo Continue statements

Haddii aad dooneyso inaad joojiso ama ka baxdo wareegga **while** ama **for**, waxad isticmaali kartaa **break**, taas oo keenaysa in kaxaynta borogaraamku kasii socota laynka ka hooseeya. Haddii aad dooneyso inaad wareegga soo bilawdo, oo talaabada ugu horaysa kasoo bilawdo, waxad isticmaali kartaa **continue**.

5.12 Strings

Waxaynu meelo badan kusoo aragnay System.out.println(<fariin>), taas oo aynu u isticmaalaynay markasta oo aynu rabno inaynu isticmaalaha borogaraamka tusno qoraal ama fariin (ama meeris). Fariinta aynu rabno inaynu tusno isticmaalaha waxaynu ku qoraynay meesha ay ku qorantahay <fariin>. Luuqadda Java xarafka kaliya waxa loo isticmaalaa nooca **char**, kaas oo ka tirsan noocyada PDT-ga Java. Haddii sheekadu ka badato hal xaraf, waxa imanaaya meeris ama qoraal, waxaana loo yaqaan string. Qoraalka ka badan hal xaraf waxa loo isticmaalaa nooc u gaar ah oo loo yaqaano String (magaciisa oo dhamaystiran waa java.lang.String). Maxaad ku samayn kartaa qoraalada? Waxyaalo badan ayaad ku samaynkartaa qoraalka ama strings-ka, oo ay ka mid yihiin inaad isticmaalaha tusto, inaad isku dhadhajiso ama nuddo qaybo qoraal ah, inaad tiriso inta xafar ee qoraalku ka koobanyahay, inaad qayb kamid ah qoraalka soo goosato, iyo waxyaale kale oo badan. Borogaraamka 5.31 ayaad ka arkaysaa tusaalooyin kala duwan oo ku saabsan isticmaalka qoraalka ama fariimaha.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static void main(String[] args) {

```

```

6      System.out.println("Waan ku salaamay aduunka!");
      byte bt = 23;
8      System.out.println("bt=" + bt);
      int jawaab = 1 + 2;
10     System.out.println("jawaab = " + jawaab);
      System.out.println("Waa " + "fariin " + "ka_kooban " + 5
12         + " qaybood");
      System.out.println("-----");
14     String fariin1 = new String("Waa meel ");
      String fariin2 = new String("dheer!");
16     System.out.println(fariin1 + fariin2);
      System.out.println("-----");
18     System.out.println("Waan ku salaamay aduunka!".length());
      System.out.println(fariin1.length());
20     System.out.println(fariin2.length());
      System.out.println("-----");
22     System.out.println(fariin1.substring(0, 3));
      System.out.println(fariin1.substring(4));
24     System.out.println("-----");
      System.out.println(fariin1.charAt(0));
26     System.out.println(fariin1.indexOf('m'));
      System.out.println(fariin1.lastIndexOf('e'));
28 }
}

```

Tusaale 5.31: Tusaalooyin ku saabsan isticmaalka qoraalka (strings)

Haddii aad kaxayso borogaraamka 5.31 waxa kuusoo baxaysa natijadan:

```

Waan ku salaamay aduunka!
bt=23
jawaab = 3
Waa fariin ka_kooban 5 qaybood
-----
Waa meel dheer!
-----
25
9
6
-----
Waa
meel
-----
W
4
6

```

Haddaba, aynu sharaxaad ka bixiyo borogaraamka 5.31:

1. Laynka 6-aad waxaynu tusaynaa isticmaalaha fariinta: "Waan ku salaamay aduunka!". Qoraal kasta oo la dhexgaliyey calaamadaha "..." waa string
2. Laynka 8-aad waxaynu tusaynaa isticmaalaha fariinta "bt=" oo lagu nuday qiimaha ku kaydsan doorsoomaha bt, innaga oo adeegsanayna calaamadaha + (plus). Isku dabaqabashada qoraalka ee sidan oo kale ah waxa loo yaqaan luuqadda ingiriisiga *concatenation*. Laynka 10-aad waa lamid
3. Laynka 11-aad wuxu inna tusayaa qoraal ka kooban qaybo dhawr ah oo laysku nuday
4. Laymanka 14 iyo 15 waxay inna tusayaan isticmaalka nooca String ee luuqadda Java, innaga oo samaynayna laba walxood oo kala ah fariin1 iyo fariin2. Laynka 16 wuxu inna tusayaa in walxaha noocoodu yahay String laysku nuddi karo iyada oo la adeegsanaayo calaamadaha + (plus)
5. Intaas kadib waxaynu tusaale u soo qaadanay ficilada strings-ka. Laymanka 18-20 waxad ka arkaysaa in qoraal kasta in la waydiinkaro dhererkiisa (inta xafar ee uu ka koobanyahay), iyada oo la adeegsanaayo ficilka length(), tusaale ahaan fariin2 waxay ka koobantahay lix xafar
6. Labada layn ee 22-23 waxad ka arkaysaa sida qoraalka qayb looga soo goosankaro, iyada oo la adeegsanaayo ficilka substring() ee nooca String, tirooyinka lasiiyey ficilkaas waxad sharaxaad ku saabsan ka helayaas hoos

7. Laymanka 25-27 waxad ka arkaysaa sida xaraf ku jira qoraalka booskiisa loo raadinkaro ama boos la garaano xarafka ku jira loo soo qaadan karo. Sharaxaad ku saabsan arintan waxad ka helaya hoos.

Haddii qoraalka aad u baahantahay mar kaliya waxad isticmaashaa sida laymanka 6,8,10 ama 11 ee borogaraamka 5.31, laakiin haddii aad u baahanayso in ka badan hal jeer waxad qoraalka ka samaysaa walax sida laymanka 14 iyo 15.

Xafar kasta oo qoraalka ku jira wuxu leeyahay nambar ama *position* kaas oo sheegaaya booska xarafkaasi kaga jiro qoraalka. Sawirka 5.5 ayaa arintan kuu iftiiminaaya. Waxaynu tusaale u soo qaadanay qoraalka laynka 6 iyo 14 ee borogaraamka 5.31. Laynka 25 waxaynu adeegsanay ficilka `charAt` ee nooca `String`, kaas oo kuusoo celinaaya xarafka ku jira booska aad siisay. Laynka 26 waxaynu adeegsannay ficilka `indexOf`, kaas oo kuusoo celinaaya booska xarafka aad siisay uu kaga jiro qoraalka. Ugu dambeyntii waxaynu adeegsannay ficilka `lastIndexOf`, kaas oo kuusoo celinaaya booska ugu dambeeya ee xarafka aad siiyey kaga jiro qoraalka.

W	a	a		m	e	e	l	
0	1	2	3	4	5	6	7	8

W	a	a	n		k	u		s	a	l	a	a	m	a	y		a	d	u	u	n	k	a	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Sawirka 5.5: Qoraalka xaraf waliba wuxu leeyahay boos ama position. Ficilada `indexOf` iyo `lastIndexOf` ayaa ku siinaaya booska uu xarafkaasi ku jiro. Haddii aad garanayso booskii oo aad raadinayso xarafka booskaas ku jira, waxad isticmaalaysaa ficilka `charAt`. Tusaale ahaan qoraalka sare meesha ugu dambeysa ee uu xarafka 'e' kaga jiro waa booska 6-aad, sida uu inoo sheegay borogaraamka 5.31. Xafarka ugu horeeya booskiisu waa eber.

5.13 Packages

Waxaan halkan ku sharaxayaa arin kale oo muhiim ah oo qayb libaaxle ka ciyaarta borogaraamyada luuqadda Java. Arintani ma aha mid Java oo kaliya ku kooban ee waa mid laga helaayo luuqadaha kale ee waaweyn. Luuqadda Java inta badan nooc walba waxa lagu kaydiyaa fayl ama gal u gaar ah, kaas oo magaciisu yahay ka nooca oo ku dhamaada `.java`. Waa arintaynu hore ugasoo hadalay qaybtii ugu horaysay ee buuggan. Tusaale ahaan nooca `Person` waxaynu ku kaydinay fayl magaciisa la yidhaahdo `Person.java`, nooca `Student` waxaynu ku kaydinay `Student.java`, iwm. Haddii tirada noocyada (iyo faylalka) borogaraamka ku jira bataan waxa imanaysa xaalad adag. Waxa kale oo iyana dhibaato imanaysaa haddii ay jiraan noocyo isku magac ah. Tirada noocyada ee borogaraamka Java ee caadiga ah ku jirta waxay gaadhi kartaan boqolaal, kumanaan ama boqolaal kun, iyada oo ay ku xidhantahay hadba sida system-kaasi u adagyahay.

Waxa loo baahanayaa hab ama nidaam noocyada lagu qayb-qaybiyo ama lagu kala xareeyo. Noocyadii wada xidhiidha waa in meel gooni ah lagu xareeyo. Kadibna ciddii u baahata ay meeshaas ka adeegsato. Luuqadda Java arintan waxa lagu xaliyaa *package*-yaga. Waad xasuusataa, markasta oo aynu nooc cusub samaynayno (`NewClass`) waa inaynu qeexo package-kiisa, tusaale ahaan nooca `Person`, package-kiisa waxaynu ka dhignay `nimi.person`. Waxad aragtay in noocyada package-koodu kala duwanyahay ay galayaan kala meel. Sida caadiga ah package-ku waa folder ama directory (gol) kumbuyuutarka ku jira. Package-yadu waa *hierarchical*, taas oo ka dhigan in package-ku package-yo kale ayuu caloosha ku haysan karaa.

Illaa hadda meelo dhawr ah ayaynu ku isticmaalay package-ka. Nooc walba laynka ugu horeeya ayaa sheegaayaa package-ka uu noocaasi ku jiro. Tusaale ahaan nooca `Isbarjava` ee borogaraamyadii aynu soo

Magacaga package-ka	Sharaxaad
java.lang	Noocyada aasaaska u ah luuqadda Java sida nooca String. Noocyada ku jira package-kan looma baahna inaad soo import -garayso
java.util	Noocyada walxaha lagu aruuriyo sida List iyo ArrayList
java.io	Ku qorista iyo akhriska faylasha
javax.swing	Noocyada Swing oo aynu qaybta saddexaad ee buuggan ku isticmaali doono

Miiska 5.8: Package-yada luuqadda Java kuwa ugu muhiimsan

dhaaftay ku isticmaalay wuxu ku jiraa package-ka nimi, Nooca Person wuxu ku jiraa package-ka nimi.person sidaan soo sheegay. Nooca Student wuxu ku jiraa package-ka nimi.student. Package-yada nimi.person iyo nimi.student waxay ku hoosjiraan package-ka nimi. Package-ka nimi waa folder ku hoos jirta folder-ka src ee qoraalka borogaraamku ku jiro.

Markaad rabto inaad nooc adeegsato, waa inaad ogaato meesha ama package-ka noocaasi uu ku jiro. Tusaale ahaan haddii aan rabo inaan nooca IsbarJava calooshiisa ku adeegsado nooca Person, waa inaan noocaas soo *import*-gareeyo aniga oo adeegsanaaya **import** nimi.person.Person;, taas oo aad ku qorayso magaca nooca meesha ka saraysa. Miiska 5.8 waxad ka arkaysaa package-yada Java kuwa ugu caansan qaar ka mid ah.

5.14 Casting iyo Instanceof

Waxaynu cutubkan kusoo gabagabaynaynaa laba mawduuc oo gaagaaban, kala ahna *Casting iyo Instanceof*. Waxaynu isku dayaynaa inaynu sharaxaad kooban ka bixino labadaas arimood. Borogaraamka 5.32 ee hoose ayaa arintan inoo iftiiminaaya.

```

1 package nimi;
2
3 import nimi.person.Person;
4 import nimi.student.Grade;
5 import nimi.student.Student;
6
7 public class IsbarJava {
8
9     public static void tusMacluumaadkaQofka(Person qof) {
10         System.out.println("-----");
11         System.out.println("ID: " + qof.getId());
12         System.out.println("Name: " + qof.getName());
13         if (qof instanceof Student) {
14             Student student = (Student)qof;
15             System.out.println("GPA: " + student.getGpa());
16         }
17     }
18
19     public static void main (String args[]) {
20         Person qof1 = new Person(1, "Cali");
21         Student qof2 = new Student(2, "Sahra");
22         qof2.addGrade(Grade.A);
23         qof2.addGrade(Grade.B);
24         qof2.addGrade(Grade.C);
25
26         tusMacluumaadkaQofka(qof1);
27         tusMacluumaadkaQofka(qof2);
28     }
29 }

```

Tusaale 5.32: Casting iyo Instanceof

Intaynaan sharaxaad ka bixin, haddii aad kaxayso borogaraamka sare ee 5.32 waa inay kuusoo baxdo:

ID: 1

Name: Cali

 ID: 2
 Name: Sahra
 GPA: 3.0

Aynku ku xigsiino sharaxaad ku saabsan borogaraamkaas:

1. Ficilka main, gaar ahaan laymanka 20-24, waxaynu samaynaynaa laba qof oo midkood arday yahay. Waxaynu kala siinaynaynaa laba ID iyo magac, kadibna waxaynu ardayga siinaynaa saddex dhibcood oo kala ah Grade.A, Grade.B iyo Grade.C
2. Intaas kadib, waxaynu laba jeer wacaynaa ficilka `tusMacluumaadkaQofka`, innaga oo kala siinaynaa labadii qof ee midkood ardayga ahaa
3. Ficilka `tusMacluumaadkaQofka` ee laymanka 9-17, haddii aynu u fiirsano, waxad arkaysaa in ficilkaas uu leeyahay hal parameter, oo noocedu yahay `Person`. Waxad markiiba halkaas ka arkaysaa in ficilka lasiinkaro walxo noocoodu yahay noocyada laga soo dhiraandhiriyeey nooca uu ficilkaasi qaadanaayo, sida halkan aynu ugu gudbinay walax `Student`
4. Caloosha ficilka haddii aynu u fiirsano, waxaynu arkaynaa in ficilkaasi marka hore isticmaalaha tusaayo ID-da iyo magaca qofka waa laymanka 11-12, arintaas wax cusub uma baahna
5. Haddaba maadaama ficilkan lasiinkaro walxo noocoodu yahay `Student`, waxaynu isticmaalaynaa operator-ka **`instanceof`**, si aynu ku ogaano in walaxdaasi tahay `Student`, waa laynka 13-aad. Haddii walax ficilka loo soo gudbiyeey noqonwaydo `Student`, operator-ka **`instanceof`** waxay soo celinaysaa qiimaha **`false`**, taas oo keenaysa in caloosha **`if`**-ka laga boodo, haddiise walaxdu tahay `Student`, operator-ka **`instanceof`** waxaysoo celinaysaa qiimaha **`true`**, taas oo keenaysa in caloosha **`if`**-ka la kaxeeyo
6. Haddii walaxda ficilka loo soo gudbiyeey tahay `Student`, waxaynu rabnaa inaynu isticmaalo ficilada u gaarka ah nooca `Student`. Sidaas awgeed markaynu ogaano in walaxdu tahay `Student`, waxaynu ku samaynaynaa waxa loo yaqaano *Casting* oo waxaynu u badalaynaa `Student`. Arintaas waxad ka arkaysaa laynka 14-aad. Arintaas waxay ku imanaysaa inaynu samayno doorsoome cusub oo ah `Student`, kadibna qiimehaan u siino qiimihii parameter-ka oo loo badalay `Student`, iyada oo la isticmaalay qaabka `(Student)qof`. Sidaad ka arkayso halkaas nooca aad u badalayso waxad galinaysaa xero, waxaad ka horaysiinaysaa doorsoomaha ama parameter-ka la badalaayo. Arintan waxaynu halkan ugusoo qaadanaay si aanad ulayaabin, markaynu qaybaha soo socda ee buuggan ku isticmaalo.

Cutub 6

Macalimiinta

Aynu borogaraamkeenii ka sii wadno meeshii uu inoo joogay. Macalinka iyo ardaygu waxa la odhan karaa waa isku mid, sababtoo ah labadooduba waa shakhsi, laakiin waxay ku kala duwanyihiin dhanka sifooyinka. Labadooduba waxay leeyihiin dhammaan sifooyinkii shakhsiga sida magacaga, telefoonka, email-ka, iwm. Halka ardaygu leeyahay dhibco uu ka helay koorsooyinkii uu qaatay oo isaga u gaar, macalinka maxaa u gaar ah ayey kula tahay? Halkan waxaynu kusoo qaadaneynaa hal sifo oo kaliya oo ah:

1. Mushaharka

Waxyaale kale oo macalinka u gaar ah halkan kusoo qaadanmayno. Mushaharku wuxu gaar u yahay macalimiinta, sababtoo ah sida caadiga ah ardayda mushahar laguma siiyo tacliinta. Waxaynu ku bilaabaynaa nooc cusub oo tijaabiye ah. Maxaynu u bixinaynaa? Waxaad rabto ayaad u bixin kartaa, laakiin qaabka aynu innagu noocyada tijaabada u magacawno waa magaca nooca ay tijaabinayaan oo *Test xagga dambe laga raaciyey, sidaas awgeed waxaynu u bixinaynaa `TeacherTest`, kaas oo loogu talagalay inuu tijaabiyo nooca `Teacher` oo matalaaya macalinka, waxaanad ka arkaysaa sawirka 6.1. Sida sawirka ka muuqata waa markii labaad ee aynu isticmaalo takhasuska, maadaama macalinku yahay qof. Waxa kale oo aad sawirka ka arkaysaa inuu nooca `TeacherTest` leeyahay laba ficil oo tijaabiyeyaal ah (waxa la odhankaraa wuxu leeyahay laba tijaabo): `testCreateTeacher` iyo `testSalary`.

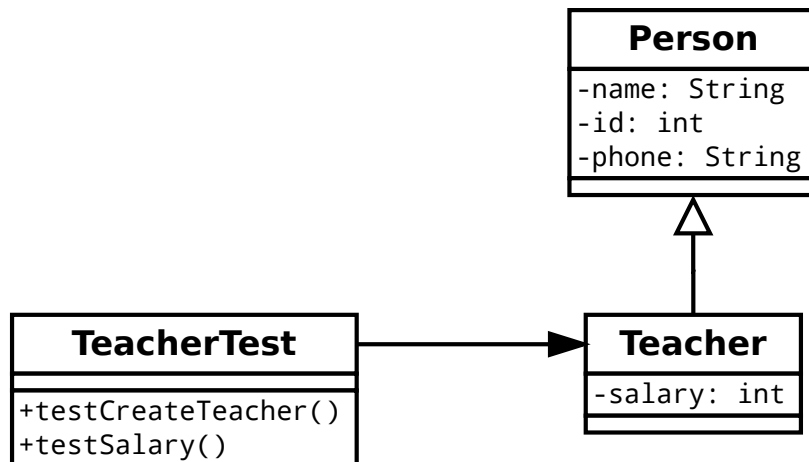
Waxad samaysaa nooc cusub oo tijaabiye ah adiga oo raacaya talaabada **NewTest**, waxad u buuxbuxisaa sidan: `Source folder=nimi/tst, Package=nimi.teacher, Name=TeacherTest`. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo `TeacherTest`, laakiin madhan. `TeacherTest` waxad ku bilawdaa sida borogaraamka 6.1.

```
1 package nimi.teacher;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class TeacherTest {
8
9     @Test
10    public void testCreateTeacher() {
11        String name = "Macalin Muuse";
12        String phone = "101010";
13        int id = 5050;
14        Teacher teacher = new Teacher(id, name);
15        assertEquals(name, teacher.getName());
16        assertEquals(id, teacher.getId());
17        teacher.setPhone(phone);
18        assertEquals(phone, teacher.getPhone());
19    }
20 }
```

Tusaale 6.1: Qoraalka `TeacherTest` bilawgiisa

Aynu sharaxaad yar ka bixiyo borogaraamka 6.1 intaynaa hore u socon:

1. Laymanka 10-19 waa ficil magaciisa la yidhaahdo `testCreateTeacher`, kaas oo loogu talagalay in lagu tijaabiyo samaynta walax (ama walxo) cusub oo `Teacher` ah
2. Laynka 11 waxaynu samaynaynaa doorsooma cusub, taas oo magaceeda la yidhaahdo `name`, nooceedu yahay `String` qiimaheeduna yahay `"Macalin Muuse"`. Sidoo kale labada layn ee ku xiga waxaynu samaynaynaa laba doorsooma oo kala ah `phone` iyo `id`



Sawirka 6.1: Nooca Teacher iyo tijaabihiisa TeacherTest. Teacher waa qof, sidaas awgeed wuxu isticmaalayaa takhasuska Person.

3. Laynka 14 waxaynu samaynaynaa macalin cusub, kaas oo aynu siinayno ID-ga iyo magaca ku kaydsan doorsoomeyaasha id iyo name
4. Laynka 15 waxaynu hubinaynaa magaca macalinkaas aynu samaynay marka la waydiiyo inuu qiimaha uu soo celiyo la midyahay ka ku kaydsan doorsoomaha name. Laynka ka hoose waa la mid, laakiin wuxu eegayaa ID-ga macalinka
5. Laynka 17 waxaynu badalaynaa telefoonka macalinka oo aynu ka dhigaynaa ka ku kaydsan phone
6. Laynka 18 waxaynu hubinaynaa telefoonka macalinka.

Kaxee nooca TeacherTest adiga oo raacaya talaabada **RunTest**, si aad u aragto inuu guuldarraysanaayo. Eclipse wuxu ka cabanayaa laynka 14, oo uu leeyahay:

Teacher cannot be resolved to a type

Taas oo macnaheedu tahay inaanu garanaynin nooca Teacher. Waxad samaysaa nooc cusub adiga oo raacaya talaabada **NewClass**. Waxad ku buuxbuuxisaa sidan: Source folder=nimi/src, Package=nimi.teacher, Name=Teacher. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo Teacher, laakiin madhan. Teacher wuxu ku bilaabmayaa sida tusaalaha 6.2.

```

1 package nimi.teacher;
2
3 import nimi.person.Person;
4
5 public class Teacher extends Person {
6
7     public Teacher(int id, String name) {
8         super(id, name);
9     }
10 }
  
```

Tusaale 6.2: Qoraalka nooca Teacher bilawgiisa. Wuxu caloosha ku haystaa kaliya ficilka dhisaha, kaas oo qiimeyaasha la siiyey u gudbinaaya nooca sare ee Person, isaga oo adeegsanaaya ereyga super.

Intaas kadib mar kale kaxee nooca TeacherTest si aad u hesho cagaar. Waxaynu ku xigsiinaynaa tijaabadii labaad oo eegaysa mushaharka ama salary-ga macalinka. Nooca TeacherTest ku kordhi ficilka testSalary sida ka muuqata qoraalka 6.3.

```

1 package nimi.teacher;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class TeacherTest {
8
9     @Test
10    public void testCreateTeacher() {
11        ...
12    }
13
14    @Test
15    public void testSalary() {
16        Teacher teacher = new Teacher(1, "Cigal Shiidaad");
17        int salary = 100; // dollar
18        teacher.setSalary(salary);
19        assertEquals(salary, teacher.getSalary());
20    }
21 }

```

Tusaale 6.3: Ficilka sare ee testCreateTeacher calooshiisa waxba iskamay badalin. Waxa soo kordhay ficilka testSalary.

Aynu sharaxaad yar ka bixino tusaalaha 6.3, intaynaan hore u socon:

1. Laynka 16 waxaynu samaynaynaa macalin cusub, innaga oo siinayna ID iyo magac
2. Kadib laynka xiga ee 17 waxaynu samaynaynaa doorsoome cusub oo nooceedu yahay `int`, iyada oo aynu ku kaydinaayno qiimaha 100
3. Kadib laynka xiga macalinka ayaynu mushahar uga dhigaynaa innaga oo adeegsanayna ficilka `setSalary`
4. Intaas kadib laynka 19 waxaynu hubinaynaa in qiimaha macalinka markaynu waydiino mushaharkiisa innaga oo adeegsanayna ficilka `getSalary` inoosoo noqdo iyo qiimaha ku kaydsan doorsoomaha salary inay isku midyihiin. Mar kasta ogow inaanay wali jirin labada ficil ee `setSalary` iyo `getSalary` ee aynu ku fikirnay in loo baahanyahay.

Eclipse wuxu ka cabanayaa laymanka 18 iyo 19, gaar ahaan ficilada `setSalary` iyo `getSalary` isaga oo leh, markaad jiirka dulgayso:

The method `setSalary(int)` is undefined for the type `Teacher`

The method `getSalary()` is undefined for the type `Teacher`

Taas oo sheegaysa in aanu nooca Teacher lahayn labadaas ficil midna. Arintani waa arimaha ugu muhiimsan ee nidaamka TDD uu keenaayo. Waxad ku fikiraysaa oo borogaraamka tijaabada ku qoraysaa noocyada iyo ficilada aad is leedahay waa loo baahanyahay. Haddhow marka tijaabadu casaan keento oo Eclipse sheego inaanay jirin ayaad bilaabaysaa inaad qorto. Marka tijaabadu cagaar keento ayaad ogaanaysaa inaad qortay borogaraamkii loo baahnaa. Haddaba, haddii aad kaxayso nooca TeacherTest, waxa guulaysanaysa hal tijaabo ta kalena ee testSalary way guuldaraysanaysaa. Waa yahay, aynu nooca Teacher ku darno labadaas ficil ee ka dhiman, waxaanad ka arkaysaa tusaalahan:

```

1 package nimi.teacher;
2
3 import nimi.person.Person;
4
5 public class Teacher extends Person {
6
7     private int salary;
8
9     public Teacher(int id, String name) {
10        super(id, name);
11    }
12
13    public void setSalary(int salary) {
14        this.salary = salary;
15    }
16 }

```

```

16 public int getSalary() {
18     return this.salary;
20 }

```

Tusaale 6.4: Qoraalka nooca Teacher marka lagu daray ficilada ka dhiman iyo sifada salary oo kaydinaaya mushaharka macalinka

Intaas kadib haddii aad kaxayso nooca TeacherTest waa inaad hesho cagaar. Ku dar nooca AllTests nooca TeacherTest, sida ka muuqadata 6.5.

```

package nimi;

2 import org.junit.runner.RunWith;
4 import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

6 @RunWith(Suite.class)
8 @SuiteClasses({
    nimi.person.PersonTest.class, // PersonTest
10    nimi.student.StudentTest.class, // StudentTest
    nimi.teacher.TeacherTest.class // TeacherTest
12 })
public class AllTests {}

```

Tusaale 6.5: AllTests oo lagu daray TeacherTest

AllTests waxaynu iskugu uruurinay dhammaan tijaabooyinkii aynu illaa hadda qornay. Haddii aad kaxayso isna waa inuu cagaar bixiyo.

6.1 Exceptions

Waxan rabaa inaan halkan kusoo qaato arin aad muhiim uga ah luuqadda Java iyo guud ahaan luuqadaha sarsare ee kumbuyuutarka. Arintaas oo loo yaqaano *Exceptions*. Waxan rabaa inaan sharaxaad ka bixiyo exceptions-ka iyo ujeedada ka dambaysa, intaynaan gaadhin tusaalaha arintan ku saabsan. Laakiinse uga hadlimayno arintan si qoto dheer halkan, ee waa shaqo akhristaha u taalla inuu markuu u baahdo macluumaad dheeraad oo arintan ku saabsan raadsado. Laakiin maadaama ay tahay arin aynu la kulmidoono, oo meelo badan ku arkidoono, gaar ahaan qaybta saddexaad ee buuggan, waxa fiican inaynu wakhti hore wax ka sheegno.

6.1.1 Waa maxay exceptions

Horta ereyga exception waxa laga soo gaabiyey *exceptional event*, waxaanu ka dhiganyahay xaalad aan caadi ahayn ama dhacdo gaar ah. Xaaladaas aan caadiga ahayn waxay imanaysaa inta badan marka borogaraamka ama qayb kamid ah borogaraamka ciladi ku timaado, taas oo borogaraamka ka leexinaysa waddadii saxda ahayd. Ciladaasi waxay noqon karta mid laga gudbi karo ama mid aan laga gudbi karin. Haddii ciladaasi tahay mid laga gudbi karo, borogaraamku intuu isticmaalaha tuso inay ciladi timid ayuu halkiisii ka sii soconayaa. Xitaa shardi ma aha inuu isticmaalaha tuso. Haddiise ciladdu tahay mid aan laga soo kaban karin, borogaraamku wuu istaagayaa ciladdaas awgeed. Marka Java laga hadlaayo waxa ku dheehan habkii loola dhiilgarayn lahaa markay xaaladdaasi timaado.

Exception-ku inta badan waxay keenaysaa in borogaraamku ka boodo meeshii uu maraayay oo meel kale u boodo. Meeshaas oo ah meeshii loogu talagalay in lagula dhiilgareeyo ciladdaas timid, loona yaqaano *Exception handler*. Haddii la waayo meeshaas, waxa imanaysa in borogaraamku istaago ciladdaas awgeed. Haddaba sidee ayay boodistu ku imanaysaa?

Sidaad aragtayba borogaraamka waxa wax qabanayaa ama wax tarayaa waa ficilada noocyada. Marka ficilka ay ciladi ku timaado, wuxu ficilkaasi samaynayaa walax mataleysa ciladdaas, oo loo yaqaano *exception object*, kadib wuxu u gudbinayaa ama u dhiibayaa JVM-ka kaxaynaysa borogaraamka (oo inta badan loo isticmaalo magaca *runtime system*), taas oo bilaabmayaa raadinta exception handler-kii ku haboonayd walaxdaas, waxaana loo yaqaan arinta *throwing an exception* ama tuurista walaxda exception-ka ah. Walaxdaasi inta badan nooceeda ayaa laga garanayaa ciladda timid waxay tahay, inkastoo inta badan walaxdaasi xambaarsantahay macluumaad dheeraad ah oo ku saabsan ciladdaas.

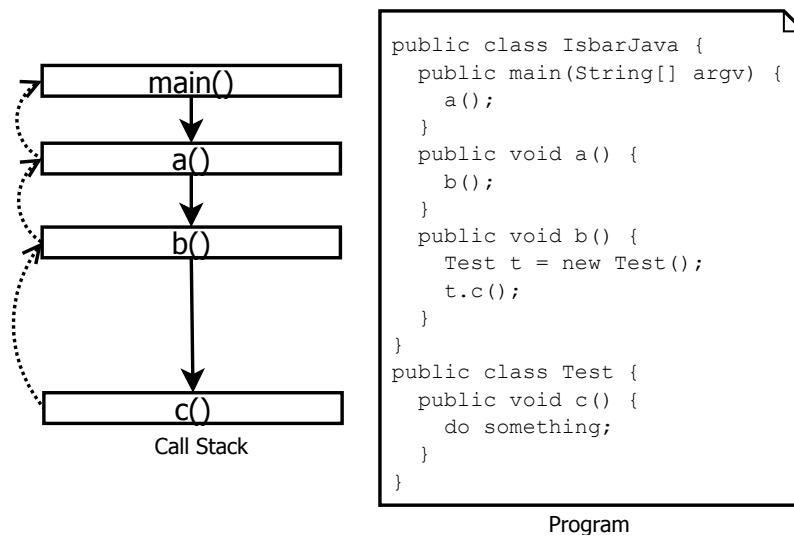
6.1.2 Waa maxay call stack

Haddaba, intaynaan tusaale soo qaadan aynu horta sharaxaad ka bixino waxa loo yaqaano *call stack*, oo aad ka arkayso sawirka 6.2. Sida sawirka ka muuqata call stack waa waddada ama ficilada uu soo maray

borogaraamku si uu u soo gaadho meesha uu imika maraayo.

Sideedaba ciladdu waxay ku iman kartaa borogaraamkaaga qayb kamid ah ama qayb kamid ah luuqadda Java oo borogaraamkaagu adeegsanaayo. Tusaale ahaan furista iyo akhriska faylasha (oo aynu dib kaga hadlidoono) ficilada aad u adeegsanayso waxay ka midyihiin luuqadda Java. Haddii ficiladaas ciladi ku timaado, tusaale ahaan haddii faylka aad doonayso inaad akhrido aanu jirin, waxay soo tuurayaan exception object arintaas sharaxaad ka bixinaaya, sidaas awgeed waa inuu borogaraamkaagu qabto wixii exception object ah ee lasoo tuuro, taas oo loo yaqaano *catch an exception*, kadibna la dhiilgareeyo.

Call stack waa qayb muhiim ah oo kamid ah exceptions-ka. Sidaas awgeed markasta oo exception timaado waxa la socda call stack, taas oo aad markiiba ka arkayso meesha ay ciladdu ku timid borogaraamka.



Sawirka 6.2: Haddii borogaraamka midigtu maraayo ficilka c() ee nooca Test, call stack-kiisu wuxu u eekaanayaa sawirka bidixda. Falaadhaha hoos u jeeda waxay sheegayaan ficilba ficilka uu sii wacay. Falaadhaha kor u jeeda ee googo'an waxay sheegayaan raadinta exception handler-ka, markay ciladi timaado. Raadintu waxay ka bilaabmaysaa ficilka la maraayo laftiisa, kadibna waxa dib loo raacayaa call stack-ka. Haddii ficilka main laga waayo exception handler ku haboon exception object-ka lasoo tuuray, borogaraamku wuu istaagayaa ciladdaas awgeed, maadaama ay tahay meeshii ugu dambaysay ee laga raadinaayey exception handler-ka.

6.1.3 Tuurista walxaha exception-ka

Arinta ugu muhiimsan ee exceptions-ku waa tuurista walaxda exception-ka. Luuqadda Java (iyo luuqadaha kaleba) waxad tuuri kartaa kaliya hal walax oo exception ah. Suurtogal ma aha inaad tuurto laba ama wax ka badan. Laakiin, waxa dhici karta in walaxdaas aad tuurtay ay caloosha ku haysato walxo kale. Sidee ayaad u tuuri kartaa walaxda exception-ka?

Si aad u tuurto walax exception ah, waa inaad marka hore samayso ama instantiate-garayso walaxdaas. Laakiin si aad u samayso walaxdaas, waa inaad ogaado ama haysato nooceda. Luuqadda Java sideedaba suurtoogal ma aha inaad samayso walax aan nooceda la garanaynin. Aan tusaale yar soo qaato, kaas oo aad ka arkayso borogaraamka 6.6.

```

package nimi;

public class IsbarJava {

```

```

4  /**
   * Ficil isuqaybinaaya laba tiro , kadibna soo celinaaya
6  * jawaabta
   *
8  * @param a
   *      qiimaha la qaybinaayo
10 * @param b
   *      qiimaha loo qaybinaayo
12 * @return jawaabta
   * @throws QiimeQaladAhException
14 *      haddii qiimaha loo qaybinaayo noqdo eber
   */
16 public static int isuqaybi(int a, int b)
   throws QiimeQaladAhException {
18     if (b == 0) {
   throws new QiimeQaladAhException(
20         "Suurtogal ma aha in tiro eber loo qaybiyo!");
   }
22     return a / b;
   }
24 ...
}

```

Tusaale 6.6: Tuurista walaxda exception-ka. Samaynta walaxdu waa sidii hore aad u soo baratay (waxad u adeegsanaysaa ereyga new), kaliya waxa cusub ereyga throw.

Marka hore borogaraamka sare ee 6.6 ha isku dayin inaad kaxayso, waayo ma dhammaystirra wali. Aan sharaxaad ka bixiyo borogaraamkaas.

1. Sidaad ka arkayso tusaalaha waxa ugu muhiimsan waa ficilka isuqaybi ee laymanka 16-23
2. Laymanka 4-15 waa sharaxaad ku saabsan waxa uu ficilku qabanaayo. Sharaxaadda noocan oo kale ah waxa loo yaqaan Javadoc, sidaan hore u soo sheegay
3. Caloosha ficilka waxa ugu muhiimsan laynka 19 (oo ku dhammaanaaya laynka 20), halkaas oo aynu samaynayno walax nooceedu yahay QiimeQaladAhException, kadibna waxaynu adeegsanay ereyga **throw** si aynu u tuuro. Walaxdaas waxaynu tuuraynaa haddii qiimaha loo qaybinaayo yahay eber, taas oo ka baxsan shuruudca xisaabta. Walaxdaas aynu tuurayno waxaynu ku raraynaa fariinta "Suurtogal ma aha in tiro eber loo qaybiyo!"
4. Waxa kale oo lagama maarmaan ah in ficilka isuqaybi uu sheego noocyada wixii walxo exception ah ee uu soo tuuri karo, si ay isticmaalaha ficilkani ugu diyaargaroobaan. Taasi waxay ku imanaysaa isticmaalka ereyga **throws**. Magaca ficilka kadib ayaa lagu qeexayaa noocyada sida isuqaybi(**int** a, **int** b) **throws** QiimeQaladAhException ee laynka 17.

Tuurista kadib waxa bilaabmaysa raadinta exception handler-kii ku haboonayd walxaha noocan oo kale ah. Sidaan soo sheegayba raadintu waxay ka bilaabmaysaa ficilkan laftiisa kadibna waxay kor u raacaysaa call stack-ka. Intaas kadib waxan wax ka sheegayaa nooca QiimeQaladAhException iyo shuruudaha looga baahanyahay, kadibna waxan dhamaystirayaa tusaalahan.

6.1.4 Noocyada walxaha exception-ka

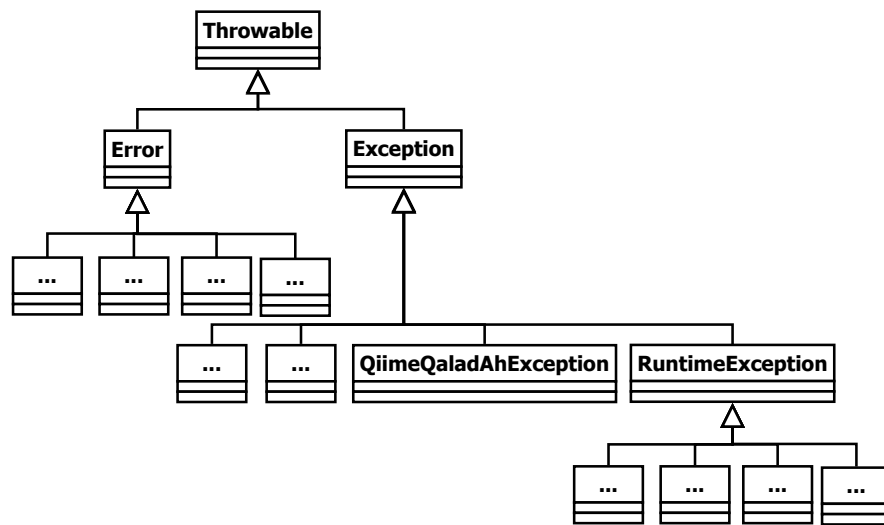
Walax kasta miyaad tuuri kartaa, mise waxa jira noocyo gaar ah oo walxaha laga sameeyo kaliya la tuuri karo? Nooca Java ee Throwable ayaa ah aabaha noocyada walxaha exception-ka. Laakiin, inta badan isaga toos looma isticmaalo ee waxa la isticmaalaa nooc-hoosaadkiisa Exception. Waxa jira noocyo aad u badan oo luuqadda Java kamid ah, oo laga soo dhiraandhiriyeey noocaas. Haddii aad rabto waxad adeegsan kartaa noocyadaas. Haddii aad rabto inaad walxahaaga exception-ka ah nooc gooni ah u samayso, waxad adna kasoo dhiraandhirinaysaa noocaas. Tusaale ahaan nooca QiimeQaladAhException ee sare, waa in laga soo dhiraandhiriyo Exception, si loo tuurikaro oo ereyga **throw** uu u qaato. Sawirka 6.3 ayaad ka arkaysaa sida ay noocyada exception-ku isugu xidhanyihiin.

```

public class QiimeQaladAhException extends Exception {
2  public QiimeQaladAhException(String msg) {
   super(msg);
4  }
}

```

Tusaale 6.7: Nooca QiimeQaladAhException caynka uu u eekaan karo qoraalkiisu. Ficilkiisa dhisaha ah waxa la siinayaa fariintii loo xambaarin lahaa walxaha noocan laga sameeyo, taas oo uu u gudbinaayo nooca sare isaga oo adeegsanaayoa ereyga **super**



Sawirka 6.3: Noocyada exception-ka iyo qaabka ay isugu xidhanyihiin. Nooca Error waxa loogu talagalay JVM-ka oo looma baahna inaynu borogaraamyadeena ku isticmaalo, waxaanu ka turjumayaa ciladaha aan laga soo kaban karin. Noocyadaada waxad kasoo dhiraandhirinaysaa nooca Exception, sida nooca QiimeQaladAhException oo kale. Nooca RuntimeException ee laga soo dhiraandhiriyey nooca Exception, isna wuxu leeyahay macno gaar ah, laakiin isna looma baahna inaynu buuggan kaga hadalo, waayo waa laga maarmi karaa, kaliya waxa kugu filan inaad ogaato inuu jiro.

Haddaba nooceena QiimeQaladAhException sidee ayuu u eekaan karaa marka la eego qoraalkiisa? Shuruudda kaliya ee lagu xidhay waa in laga soo dhiraandhiriyo nooca Exception. Wuxu u eekaan karaa sida tusaalaha 6.7, iyada oo aan wax kale lagu darin.

6.1.5 Qabashada walxaha exception-ka

Wixii walxo ah ee lasoo tuuro ficilku wuu qaban karaa oo wuu la dhiilgarayn karaa ama wuu isdhaafin karaa oo wuxu u gudbin karaa ficilkii soo wacay. Arintan luuqadda Java waxa loo yaqaan *Catch or Specify requirement*. Haddii uu ficilku doorto inuu isdhaafiyo wixii walxo lasoo tuuro, waa inuu magaciisa kadib ku qeexo noocyada walxaha uu isdhaafinaayo, isaga oo adeegsanaaya ereyga **throws**. Aan tusaale yar soo qaato oo ku saabsan isdhaafinta, kaas oo aad ka arkayso tusaalaha 6.8.

```

1 package nimi;
2
3 public class IsbarJava {
4
5     public static int isuqaybi(int a, int b)
6         throws QiimeQaladAhException {
7         ...
8     }
9
10    public static int xisaabi(int a, int b, String nooca)
11        throws QiimeQaladAhException {
12        if(nooca.equals("isuqaybi")) {
13            return isuqaybi(a,b);
14        } else if (nooca.equals("kalagoo")) {
15            return kalagoo(a,b);
16        }
17        ...
18    }
19    ...
20 }

```

Tusaale 6.8: Isdhaafinta walaxda exception-ka ah. Ficilka xisaabi wuxu adeegsanayaa ficilka isuqaybi sida ka muuqata laynka 13-aad. Maadaama ficilka isuqaybi uu soo tuuraayo walax exception ah oo nooceedu yahay QiimeQaladAhException, ficilka xisaabi wuxu ku qasbanyahay inuu qabto oo la dhiilgareeyo ama uu isdhaafiyo. Imika wuxu doortay inuu isdhaafiyo, sidaas awgeed waa inuu magaciisa kadib ku sheego noocyada uu isdhaafinaayo isaga oo adeegsanaaya ereyga **throws**

Suurtogal ma aha in ficil kastaaba isdhaafiyo walxaha exception. Waa in meeluun lagu qabto oo lagula dhiilgareeyo, maadaama ay walxahaasi ka turjumayaan cilado ku yimid borogaraamka. Sidaas awgeed sidaan hore u soo sheegay, haddii ciladdu tahay mid laga soo kaban karo, waa in la qabto oo lala dhiilgareeyo walxaha exception-ka ee ciladdaas matalaaya. Waxan rabaa inaan soo qaato borogaraam yar, laakiin dhamaystiran oo arintan ku tusaaya. Borogaraamkaasi wuxu adeegsanayaa ficilka isuqaybi ee aynu kor kusoo aragnay. Borogaraamku wuxu akhristaha waydiinayaa laba tiro, kadibna wuxu u soo celinayaa jawaabta tirada hore oo loo qaybiyey tirada dambe. Tusidda jawaabta kadib, borogaraamku wuxu isticmaalaha waydiinayaa laba tiro oo cusub. Wareeggaas ayuu ku jirayaa illaa inta uu ka damaayo. Borogaraamkan yar waxad ka arkaysaa 6.9.

```

1 package nimi;
2
3 import java.util.Scanner;
4
5 class QiimeQaladAhException extends Exception {
6     public QiimeQaladAhException(String msg) {
7         super(msg);
8     }
9 }
10
11 public class IsbarJava {
12
13     /**
14      * Ficil isuqaybinaaya laba tiro, kadibna soo celinaaya jawaabta
15      *
16      * @param a
17      *         qiimaha la qaybinaayo
18      * @param b
19      *         qiimaha loo qaybinaayo
20      * @return jawaabta
21      * @throws QiimeQaladAhException
22      *         haddii qiimaha loo qaybinaayo noqdo eber
23      */

```

```

24 public static int isuqaybi(int a, int b)
    throws QiimeQaladAhException {
26     if (b == 0) {
        throw new QiimeQaladAhException(
28         "Suurtogal ma aha in eber tiro loo qaybiyo!");
    }
30     return a / b;
    }
32
33 public static void main(String[] args) {
34     Scanner scan = new Scanner(System.in);
    while (true) {
36         System.out.print("Isii tirada la qaybinaayo: ");
        int a = scan.nextInt();
38         System.out.print("Isii tirada la qaybinaayo: ");
        int b = scan.nextInt();
40         System.out.println("Jawaabtu waa: " + isuqaybi(a, b));
    }
42 }
}

```

Tusaale 6.9: Qabashada walaxda exception-ka

Nasiibdarro borogaraamka sare ee 6.9 sidaas kuma shaqaynaayo, sababtoo ah wax exception handler ah kuma jiraan. Ma dhaafaayo compiler-ka Java, maadaama ay ku jiraan cilado ka baxsan shuruuca luuqadda Java. Ciladdu waxay ka taagan tahay laynka 40, kaas oo markaad jiirka dulgayso Eclipse ku tusaayo fariinta:

Unhandled exception type QiimeQaladAhException

Taas oo uu ula jeedo in aan exception-kaas la qabanin, loona baahanyahay in la qabto. Badal laynkaas oo waxad ku badashaa laymanka tusaalaha hoose ee 6.10.

```

1 ...
2 try {
3     System.out.println("Jawaabtu waa: " + isuqaybi(a, b));
    } catch (QiimeQaladAhException qiimeQaladAh) {
5         System.out.println(qiimeQaladAh);
    }
7 ...

```

Tusaale 6.10: Qabashada exception-ka nooceedu yahay QiimeQaladAhException

Intaas kadib borogaraamkeenii waa diyaar oo waa la kaxaynkaraa, maadaama aanu meelo kale oo casaan ka muuqan. Kaxee borogaraamka adiga oo raacaya `RunJava`, kadibna tirooyin kala duwan sii oo eeg waxa uu kuugu jawaabo (view-ga `Console`). Ugu dambayntii sii qiime aan tiro ahayn, kadibna eeg waxa dhaca. Sawirka 6.4 ayaad ka arkaysaa hal jeer oo aan anigu kaxeeyey.

Sidaad sawirka ka arkayso borogaraamku caadi ayuu u shaqaynayaa illaa inta uu isticmaaluhu ka siinaayo qiime qalad ah oo aan tiro ahayn. Haddii isticmaaluhu siiyo qiime qalad ah laynka 37-aad ee `int a = scan.nextInt();` ayaa wuxu soo tuurayaa walax exception ah oo nooceedu ayahay `java.util.InputMismatchException` (laynka ugu sareeya call stack-ka), taas oo maadaama aan la qabanaynin keenaysa in borogaraamku istaago. Marka borogaraam sidan oo kale iyada oo aan laga filanaynin istaago ama kufo waxa loo yaqaan *Crash*. Waxan filayaa inaad aragtay *Blue Screen of Death*, kaas oo ah screen-ka buluugga ah ee marka OS-ka Windows crash-gareeyo ku tusaayo.

Ugu dambayntii waa maxay qaybaha kala duwan ee ay ka koobantahay qabashada walxaha exception-ku? La dhiilgaraynta exceptions-ku sideedaba waxay ka koobantahay saddex qaybood `try`, `catch` iyo `finally` oo aad ka arkayso tusaalaha 6.11

```

1 try {
2     //halkan ku qor ficilada aad wacayso
    //ee soo tuuri kara walxo exception ah
4 } catch (ExceptionType1 name1) {
    //halkan ku qor talaabooyinkii aad kula
6     //dhiilgaraynayso haddii lasoo tuuro walxo noocoodu
    //yahay ExceptionType1
8 } catch (ExceptionType2 name2) {
    //halkan kuwa ah ExceptionType2
10 } catch (ExceptionTypeX nameX) {
    //halkan kuwa ah ExceptionTypeX
12 } finally {
    //Wax kasta oo dhaca qaybtan waa la
14 //kaxaynayaa ugu dambeyntii
    }

```

```

<terminated> IsbarJava [Java Application] C:\Program Files\Java\jdk1.7.0_11\bin\javaw.exe (9.2.2013 7.18.49)
Isii tirada la qaybinaayo: 200
Isii tirada la qaybinaayo: 10
Jawaabtu waa: 20
Isii tirada la qaybinaayo: 225
Isii tirada loo qaybinaayo: 10
Jawaabtu waa: 22
Isii tirada la qaybinaayo: abc
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor (Scanner.java:909)
    at java.util.Scanner.next (Scanner.java:1530)
    at java.util.Scanner.nextInt (Scanner.java:2160)
    at java.util.Scanner.nextInt (Scanner.java:2119)
    at nimi.IsbarJava.main (IsbarJava.java:37)
Call Stack

```

Sawirka 6.4: Call Stack

Tusaale 6.11: Qaybaha ay ka koobantahay qabashada walxaha exception-ku

Waxa fiican inaan sharaxaad kooban ka bixiyo tusaalaha 6.11, intaynaan hore u socon:

1. Marka hore **catch** kastaaba waa exception handler, waxaana loogu talagalay in lagula dhiilgareeyo walxaha dhammaan walxaha noocoodu yahay nooca la siiyey. Tusaale ahaan **catch**-ka ugu sareeya waxa loogu talagalay in lagula dhiilgareeyo walxaha noocoodu yahay `ExceptionType1`
2. Qaybta **try** caloosheeda waxa lagu qorayaa borogaraamka laga baqaayo inuu soo tuuro walxo exception ah
3. Marka caloosha **try** laga soo tuuro walax exception ah, runtime system-ka ayaa u raadinaaya **catch**-kii ku haboonayd isaga oo eegaaya nooca walaxdaas
4. Ugu dambayntii qaybta **finally** waxa loogu talagalay in la kaxeeyo haddii ay exception timaado iyo haddii aanay imanba. Waxaana loogu talagalay waxyaalaha nadiifinta ah sida tusaale ahaan xidhista faylashay. Qaybtani waxay u dhigantaa tusaale ahaan qofka ugu dambeeya marka uu guriga ka baxo waxyaalaha uu qabanaayo, sida daminta laydhadhka, xidhista albaabada iyo daaqaadaha, hubinta in aan wax dab ah oo baxayaa jirin, iwm.

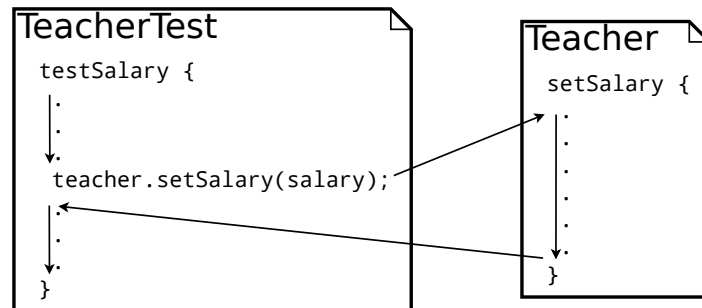
Intaanad qaybtan ka gudbin, cashar ahaan, badal borogaraamka 6.9, si uu ugu shaqeeyo tirooyinka decimal-ka ah sida 123.50. Haddii aan ku siiyo inyar oo caawimo ah meel kasta oo ay ku jirto **int** ku badal **double**. Si kastaba, waa inaanay marna macluumaadka uu isticmaaluhu siinaayo borogaraamkaaga keenin in borogaraamku kufo ama crash-gareeyo.

6.1.6 Tijaabinta exceptions-ka

Waxaynu soo qaadaneynaa tusaale kale oo kamid ah borogaraamkeenii aynu buuggan ku dhisayno. Waa yahay, haddii tusaale ahaan aad u fiirsato ficilka `setSalary` ee nooca `Teacher` ee borogaraamka 6.4, waxad arkaysaa marka hore inuu yahay ficil aad runtii u fudud. Waxa kaliya ee uu qabanayaa waa inuu qiimaha la siiyey ee parameter-ka `salary` loogusoo gudbiyey uu ku kaydiyo sifada `salary` ee nooca `Teacher` (`this.salary`). Haddaba su'aashu waxay tahay ma noqonkara mushaharka macalinku negative ama qiime ka yar eber, tusaale ahaan -50? Haddii mushaharka macalinku aanu noqonkarin qiime ka yar eber, miyaanay haboonayn in ficilka `setSalary` marka la siiyo qiime negative ah inuu arintaas wax ka yidhaahdo?

Ficilka `setSalary` waxaynu isticmaalaynaa laynka 18-aad ee tusaalaha 6.3 ee nooca `TeacherTest`, si aynu u tijaabino, laakiin waxad ka isticmaali kartaa meel kasta oo aad u baahato inaad ficilkaas adeegsato. Sideedaba isticmaalaha ama meesha ficilka laga isticmaalaayo waxa loo yaqaan *caller*. Ficilka `setSalary` wax qiimaha masoo celinaayo sababtoo ah `ReturnType`-kiisu waa `void`. Sawirka 6.5 ayaad ka arkaysaa kaxaynta tijaabada `testSalary` ee nooca `TeacherTest` waddada ay marayso. Kaxayntu waxay ka bilaabmaysaa

talaabada ama statement-ka ugu horeeya ee ficilka `testSalary` kadib hoos ayey u soconaysaa. Marka lasoo gaadho meesha laga isticmaalaayo ficilka `setSalary`, ayey kaxayntu ku boodayso ficilka `setSalary` calooshiisa oo ka mid ah nooca `Teacher`, halkaas oo ay ka bilaabayso talaabada ugu horaysa. Ereyga **this** wuxu tilmaamayaa macalinka aynu ku samaynay laynka 16-aad ee ficilka `testSalary`, inta ay kaxaynta ficilka `setSalary` socoto. Kadib marka kaxaynta ficilka `setSalary` ee nooca `Teacher` soo dhamaato ayaa kaxaynta borogaraamku kusoo laabanaysaa ficilka `testSalary` (caller), halkaas oo kaxayntu kasii soconayso talaabada ka hoosaysa sida sawirka ka muuqata. Sidani waa qaabka borogaraamku u soconaa xaaladda caadiga ah, ee aanay ku jirin u diyaargarawga ciladaha.



Sawirka 6.5: Ficilka `testSalary` ee nooca `TeacherTest` siduu u soconaa marka la kaxeeyo sida caadiga ah

Si haddaba aynu su'aalihii uga jawaabno, waxa la rabaa in kaxaynta ficilka `setSalary` ay istaagto marka ciladi timaado, kadibna kaxayntu ku laabato wicihii ama meeshii ficilkan laga soo wacay iyada oo warbixin laga siinaayo ciladdii timid. Arintaas laba siyaabood ayaa lagu gaadhi karaa. Kow, waa in ficilka `setSalary` soo celiso qiime oo tusaale ahaan markay ciladi timaado qiime **boolean** ah oo **false** ah uu soo celiso, haddii kalena oo kaxayntiisu caadi ku dhamaatayna soo celiyo **true**. Laba, inuu exception object soo tuuro marka ciladdi timaado. Sideedaba qiimaha lasoo celinaayo inta badan kuma haboona ka warbixinta ciladaha, sidaas awgeed arimaha noocan oo kale ah waxa loogu talagalay sida labaad oo ah exceptions-ka. Sawirka 6.6 ayaad ka arkaysaa marka exceptions-ka lagusoo daro sawirka iyo sida ay wax uga badalayso kaxaynta ficilka `setSalary`.

Kaxaynta ficilka `setSalary` waa sidii hore, laakiin imika marka la arko in qiime qalad ah lasiiyey ama ay timaado cilad kale wuxu ficilku tuurayaa walax loo yaqaano exception object, sidaan hore u soo sheegayba, isaga oo adeegsanaayo ereyga **throw**, walaxdaas oo sharaxaad ka bixinaysa ciladda timid. Tuurista kadib, waxa bilaabmayna raadinta exception handler-kii ku haboonayd, taas oo ka bilaabmayna ficilkan kadibna kor u raacaysa silsiladda call stack-ka.

Tusaalahan kaxaynta borogaraamku waxay ku boodayso ficilka `testSalary` oo isticmaalayay ah. Ficilkaas waxa u banaan laba arimood, oo aad ka arkayso isla sawirka 6.6, inuu u diyaargarooobo oo qabto wixii walxo exception ah ee ficilka `setSalary` soo tuuro, oo ah qaybta sare ee sawirka (ee kow ku qorantahay) iyo inuu is dhaafiyo wixii walxo exception ah ee ficilkaasi soo tuuro oo ah qaybta hoose ee sawirka (ee laba ku qorantahay).

Haddii aad rabto inaad qabato wixii exception ah ee lasoo tuuro noocay rabto ha noqotee, waxad isticmaali kartaa **try-catch** u eeg tusaalaha 6.12.

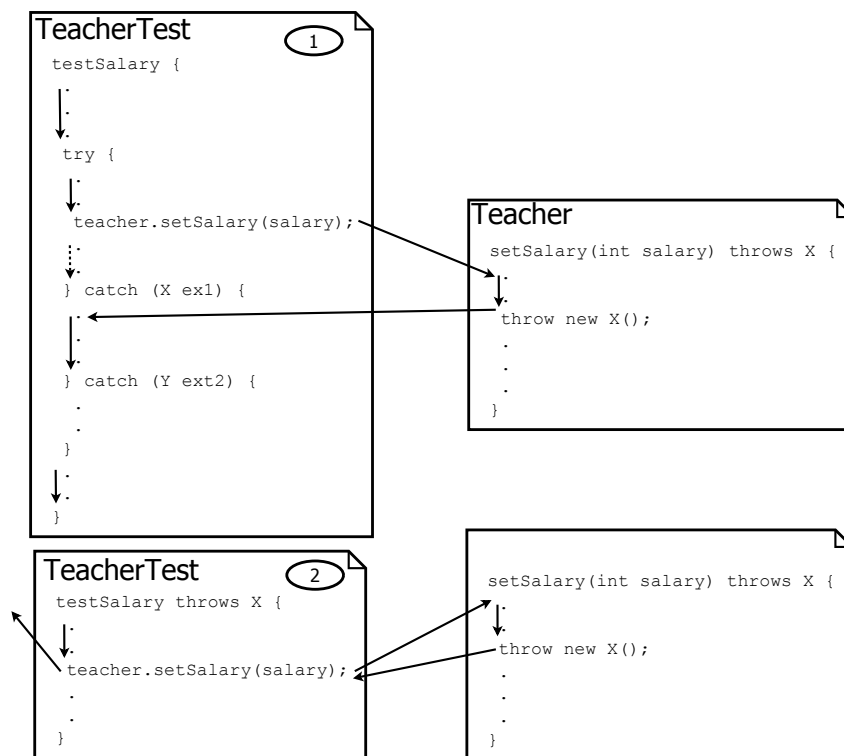
```

1 try {
2   //halkan ku qor ficilada aad wacayso
3   //ee soo tuuri kara walxo exception ah
4 } catch (Exception name) {
5   //exception kasta oo lasoo tuuro noocay rabto
6   //ha noqotee halkan kula dhiilgaree
7 }
  
```

Tusaale 6.12: Try-catch-all

Tusaalaha 6.12 inta badan ficilka main ayaa lagu isticmaalaa sidan oo kale si loo qabto dhammaan wixii walxo lasoo tuuro. Laakiin ogow haddii aad ku isticmaasho ficilada calooshooda in exception handler-ka noocan oo kale ah qabanayso dhammaan wixii walxo lasoo tuuro, taas oo keeni karta marmar qaar inay qabato qaar la rabay inay is dhaafiso. Sideedaba ficilka walxaha lasoo tuuro qaarna wuu qabankaraa, qaarna wuu is dhaafinkaraa.

Si haddaba arintani inoogu furanto oo aynu u fahamno waxaynu ficilka `setSalary` ku daraynaa tuurista exception object markay ciladi ku timaado. Maadaama aynu adeegsanayno **TDD**, waa inaynu ku horayno qorista tijaabada. Waxaynu tijaabinaynaa in marka ficilka `setSalary` la siiyo qiime negative ah inuu soo tuuraayo walax exception ah. Walax kastaaba waxay leedahay nooc, sidaas awgeed walaxdaas nooceeda waxaynu u isticmaali karnaa nooc aynu iska leenahay oo aynu u bixinkarno tusaale ahaan



Sawirka 6.6: Ficilada `testSalary` iyo `setSalary` iyo saameynta exceptions-ku ku leeyihiin kaxayntooda. Waxad arkaysaa in labada ficilba wax iska badalayaan.

`InvalidSalaryException` ama waxaynu adeegsan karnaa nooca `IllegalArgumentException` oo ku jira luuqadda Java (ama noocyada kale ee lamid ka ah mid ku haboon). Qaabkani waa sida aad u tijaabin karto in ficiladaadu u shaqaynayaan sidii la rabay, markay ciladi ku timaado ama la siiyo macluumaad qalad ah.

Waxaynu badalka ku bilaabaynaa nooca `TeacherTest`, innaga oo u eekaysiinayna tusaalaha 6.13, ee hoose.

```

1 package nimi.teacher;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class TeacherTest {
8
9     ...
10
11     @Test
12     public void testSalary() throws InvalidSalaryException {
13         Teacher teacher = new Teacher(1, "Cigal Shiidaad");
14         int salary = 100; // dollar
15         teacher.setSalary(salary);
16         assertEquals(salary, teacher.getSalary());
17     }
18
19     @Test
20     public void testSalaryInvalid() {
21         Teacher teacher = new Teacher(1, "Cigal Shiidaad");
22         try {
23             teacher.setSalary(-50);
24             fail("Exception expected, but was not thrown!");
25         } catch (InvalidSalaryException expected) {
26             assertEquals("-50 is invalid salary for teacher!",
27                         expected.getMessage());
28         }
29     }
30 }

```

Tusaale 6.13: Tijaabinta xaaladaha aan caadiga ahayn. Ficilkii hore ee `testSalary` calooshiisu waa sidii hore, laakiin maadaama ay tahay inuu qabto ama isdhaafiyo wixii walxo ee `setSalary` soo tuuro, wuxu doortay inuu isdhaafiyo. Waxa kale oo soo kordhay ficilka `testSalaryInvalid`.

Sida ka muuqata 6.13 ficilkii `testSalary` waxyar ayaa iska badalay, waxa kale oo soo kordhay ficil cusub `testSalaryInvalid`. Ficilka `testSalary` calooshiisu waa sidii hore, laakiin maadaama uu adeegsanaayo ficil soo tuuraaya walax exception ah, waa inuu laba mid doorto: inuu qabto oo la dhiilgareeyo iyo inuu isdhaafiyo. Maadaama aynu qiime la ogolyahay (100) siinay ficilka `setSalary`, waxaynu ku kalsoonay inaanu soo tuuraynin exception, sidaas awgeed waxaynu dooranay inaynu isdhaafino, oo waxaynu ficilka magaciisa ku darnay **throws** `InvalidSalaryException`. Haddii ay dhacdo in ficil `setSalary` soo tuuro walax exception ah innaga oo siinay qiime la ogolyahay, waxa imanaysa in tijaabadu guuldarraysato oo casaan keento, kadibna waa in loo fiirsado sababta keentay.

Tijaabada labaad ee `testSalaryInvalid`, waxaynu tijaabinaynaa marka macalinka la siiyo mushahar qalad ah. Waxaynu rabnaa inaynu hubsanno, marka ficilka `setSalary` la siiyo qiime qalad ah, inuu soo tuuraayo walax exception ah, taas oo nooceedu yahay `InvalidSalaryException`. Haddii uu soo tuuriwaayo waa in tijaabadu guuldaraysato oo casaan keento, taas oo sheegaysa inaanu ficilku sidii la rabay u shaqaynaynin.

1. Laynka 23 waxaynu wacaynaa ficilka `setSalary`, innaga oo siinayna qiime qalad ah, oo ah -50. Waxaynu sugaynaa in ficilkaasi soo tuuro walax exception ah, kadib borogaraamku u socdo sida sawirka 6.6, gaar ahaan qaybta sare
2. Haddii uu soo tuuriwaayo wax exception ah, borogaraamku wuxu u gudbayaa laynka ku xiga 24, kaas oo aynu isticmaalayno ficilka `fail` ee JUnit, kaas oo keenaaya in tijaabadu guuldaraysato oo casaan keento
3. Haddiise uu ficilka `setSalary` soo tuuro walax exception ah, taas oo sidii laga sugaayey ah, borogaraamku wuxu ku boodayaa exception handler-kii ku haboonayd, taas ka mid ah ficilka `testSalary`, maadaama ficilka `setSalary` aanay ku jirin, waana laymanka 25-28. Exception-ka waxaynu u bixinay `expected`, maadaama ay tahay mid la sugaayey
4. Kadib laynka 26 (oo ku dhammaanaaya 27), waxay walaxda exception-ka ee lasoo tuuray isticmaalaynaa ficilkeeda `getMessage`, kaas oo inoosoo celinaaya fariinta uu xambaarsanyahay, kadib waxaynu hubinaynaa in fariinta uu xambaarsanyahay tahay `"-50 is invalid salary for teacher!"`.

Intaas kadib haddii aad kaxayso nooca `TeacherTest`, waa inuu casaan keeno, oo Eclipse wuxu ka cabanayaa nooca `InvalidSalaryException`, isaga oo leh:

`InvalidSalaryException cannot be resolved to a type`

Samee noocaas, waxaanad ku tuurtaa isla package-ka nooca `Teacher`, waxaanad ka arkaysaa tusaalaha 6.14.

```

1 package nimi.teacher;
2
3 public class InvalidSalaryException extends Exception {
4     public InvalidSalary(String msg) {
5         super(msg);
6     }
7 }

```

Tusaale 6.14: Nooca `InvalidSalaryException` waa in laga soo dhiraandhiriyo nooca `Exception`

Nooca `InvalidSalaryException` waxa laga soo dhiraandhriyey nooca `Exception`, kaas oo ah nooca aad inta badan kasoo dhiraandhirinayso noocyada walxaha exception-ka. Ficilkiisa dhisaha ah waxaynu u gudbinaynaa fariinta uu walaxda laga sameeyey noocan ay xambaaridoonto, isna wuxu usii gudbinayaa ficilka dhisaha ee nooca `Exception`.

Intaas kadib, Eclipse wuxu ka cabanaa laynka 35, isaga oo leh:

`Unreachable catch block for InvalidSalaryException.`
`This exception is never thrown from the try statement body`

Taas oo ka dhigan inaan caloosha **try** borogaraamka ku jira aanu soo tuuraynin walxo exception ah oo noocoodu yahay `InvalidSalaryException`, sidaas awgeed aan loo baahnayn. Si aynu ciladdan uga gudubno waa inaynu ficilka `setSalary` wax ka badalo, oo ku darno inuu walax exception ah soo tuuro, haddii qiime qalad ah la siiyo, waxaanad ka arkaysaa tusaalaha 6.15 ee hoose.

```

1 package nimi.teacher;
2
3 import nimi.person.Person;
4
5 public class Teacher extends Person {
6     ...
7
8     public void setSalary(int salary) throws InvalidSalaryException {
9         if (salary <= 0) {
10             throw new InvalidSalaryException(salary
11                 + " is invalid salary for teacher!");
12         }
13         this.salary = salary;
14     }
15     ...
16 }
17
18 }

```

Tusaale 6.15: Ficilka `setSalary` oo lagu daray inuu wax ka yidhaahdo haddii qiime qalad ah la siiyo

Aan sharaxaad yar oo kooban ka bixiyo tusaalaha sare ee 6.15:

1. Marka hore laymanka 10-12 waxaynu hubinaynaa qiimaha parameter-ka `salary`, haddii uu ka yaraado ama la mid noqdo eber, waxaynu samaynaynaa walax cusub oo ah `InvalidSalaryException`, taas oo aynu siinayno fariintii ay xambaari lahayd, kadib waxay adeegsanaynaa ereyga **throw** si aynu u tuurno oo ugu gudbino JVM-ka, taas oo bilaabi doonto raadinta exception handler-kii ku haboonayd
2. Ficilku haddii uu tuuraayo walxo exception ah waa inuu sheego noocyadooda. Ficilka `setSalary` wuxu tuuri karaa kaliya walxo `InvalidSalaryException` ah, sidaas awgeed waa inaynu magaciisa ku dabaqabano **throws** `InvalidSalaryException`.

Intaas kadib haddii aad kaxayso tijaabada `TeacherTest` waa inuu dhammaan cagaar bixiyo.

6.2 Gunaanad

Cutubkan wixii aynu ku baranay waxa ugu muhiimsan exceptions-ka. Exceptions-ku waa qaab borogaraamka Java uu ula dhiilgareeyo ciladaha iyo xaaladaha aan caadiga ahayn. Waxaynu odhan karnaa waa qaabka uu borogaraamku isugu gudbiyo dhammaan wixii ciladda la xidhiidha.

Cutub 7

Koorsooyinka

Dhagaxii ugu waynaa waynu iska hor qaadnay, maadaama aynu ka hadalay naxwaha ama qodobada uu ka kooban yahay borogaraam kasta oo Java ah. Cutubkan waxaynu sii wadaynaa borogaraamkii **nimi** ee aynu buuggan ku dhisaynay. Cutubkan waxaynu kaga hadlaynaa koorsooyinka oo kaalin muhiim ah kaga jira nidaamka macluumaadka iskuulka (nimi). Maxaynu ula jeednaa koorsooyinka? Maxay ka kooban yihiin koorsooyinku?

Haddii aan waxyar dib ugu laabto bilawgii qaybtan. Waxaynu soo sheegnay in koorsooyinku noqon karaan tusaale ahaan: xisaab, juqraafi, af-ingiriisi, af-soomaali, iwm. Koorsadu waxay yeelan kartaa dhawr fadhi sanadkii. Fadhiyadu waxay dhici karaan isku mar haddii tusaale ahaan ardaydu badantahay. Fadhi kastaaba wuxu leeyahay wakhti uu bilawdo, koorso uu la xiriiro, macalin dhiga iyo dhibco laga helaayo marka ardaygu dhameeyo koorsadaas. Waxaynu si fiican uga hadlaynaa marka aynu gaadhno cutubyada soo socda.

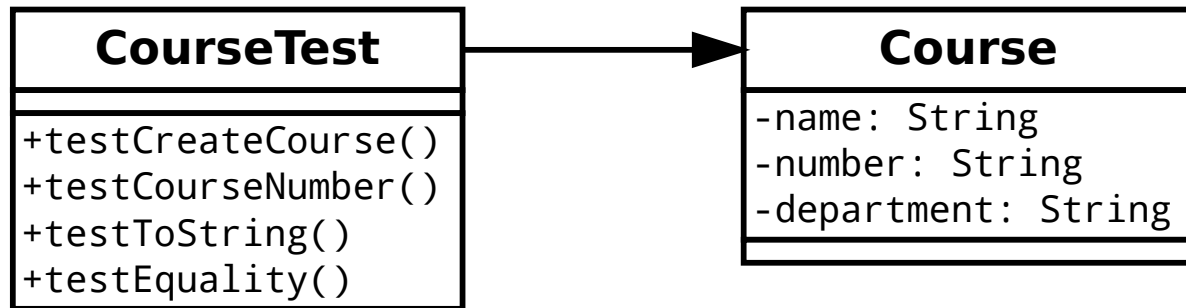
Macluumaadka koorsooyinka looga baahanyahay waxa ka mid ah:

1. Magaca koorsada (tusaale ahaa: "Af-soomaali 1", "Af-soomaali 2", "Ingiriisi bilaw", "Xisaab bilaw", "Geometry", "Programming 1", ...)
2. Nambarka koorsada (sida: "MAT-01", "MAT-02", "ENG-01", "SOM-01", "COM-02", ...)
3. Qaybta ama *department*-ka koorsada (sida: "Xisaabta", "Luqadaha", "Fisigiska", "Kumbuyuutar saynis", ...)

Intaasi waa inta aynu ku soo qaadanayno buuggan, laakiin inta badan way ka badan yihiin macluumaadka koorsooyinka laga kaydiyo marka la eego jaamacadaha waaweyn ee adduunka. Aan sharaxno sifooyinka koorsada. Magacu waa caadi oo waa qoraal (string) u gaar ah koorsada. Nambarku wuxuu ka koobanyahay saddex xaraf iyo jiitin, kadibna hal nambar oo ka kooban laba boos. Ujeedada aynu ka leenahay waa iney dhammaan nambarada koorsooyinku isu eekaadaan. Saddexda xaraf waxa laga soo gaabiyey magaca koorsada ama magaca department-ga koorsadaas. Laakiin qasab ma aha oo waxay noqon karaan saddex xaraf ee aad rabto. Nooca Course iyo tijaabihiisa CourseTest oo u taagan koorsooyinka, waxa kuu muujinaaya sawirka [7.1](#).

Waxaan ku bilaabaynaa nooc cusub oo tijaabiye ah oo magaciisa la yiraa CourseTest. Samee noocan cusub adiga oo raacaya talaabada **NewTest**, u buuxbuux sidan: Source Folder=**nimi/tst**, Package=**nimi.course** iyo Name=**CourseTest**. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo CourseTest, laakiin madhan. CourseTest waxad ku bilawdaa sida tusaalaha [7.1](#).

```
1 package nimi.course;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class CourseTest {
8
9     @Test
10     public void testCreateCourse() throws Exception {
11         String courseName = "Af-soomaali 1";
12         String courseNumber = "SOM-01";
13         String department = "Luqadaha";
14         Course course = new Course(courseName, courseNumber,
15                                     department);
16         assertEquals(courseName, course.getName());
17         assertEquals(courseNumber, course.getNumber());
18         assertEquals(department, course.getDepartment());
19     }
20 }
```



Sawirka 7.1: Nooca Course iyo Tijaabihiisa CourseTest.

```

    }
20 }

```

Tusaale 7.1: Qoraalka CourseTest bilawgiisa

Aan sharaxno borogaraamka 7.1:

1. Laymanka 10-19 waxaynu ku tijaabinaynaa samaynta nooc cusub oo Course ah, kaas oo magaciisu yahay "Af-soomaali 1", nambarkiisu "SOM-01" iyo qaybtiisu "Luqadaha". Laynka 14 ayaynu samaynaynaa noocan cusub ee koorsada (Course) ah
2. Laynka 16 waxaynu hubinaynaa koorsada marka la waydiiyo magaceeda inuu la mid yahay magacii aynu siinay
3. Laynka 17 waxaynu hubinaynaa nambarka koorsada
4. Laynka 18 waxaynu hubinaynaa qaybta (department) koorsada.

Haddii aad kaxayso borogaraamka 7.1 adiga oo raacaya talaabada **RunTest**, waxaad la kulmaysaa cilad. Marka ugu horaysa **Eclipse** wuxuu ka cabanayaa laynka 14, isaga oo sheegaya:

`Course cannot be resolved to a type`

Hore ayaad u soo aragtay ciladan, markaas waad garanaysaa sababta keentay. Nooca Course wali ma jiro, sidaas awgeed **Eclipse** wuxuu hoosta ka marinayaa casaan/guduud. Si aynu ciladan uga gudubno samee nooc cusub adiga oo raacaya talaabada **NewClass**, una buuxbuuxi sidan: Source Folder=nimi/src, Package=nimi.course iyo Name=Course. Intaas kadib waxaad helayaa nooc cusub, laakiin madhan. Nooca Course waxaad u ekaysiisaa oo ku bilawdaa tusaalaha 7.2.

```

1 package nimi.course;
2
3
4 public class Course {
5     public Course(String name, String number,
6         String department) {
7     }
8 }

```

Tusaale 7.2: Nooca Course bilawgiisa

Intaas kadib **Eclipse** wuxuu ka cabanayaa ficilada `getName`, `getNumber` iyo `getDepartment` ee laymanka 16-18 ee tusaalaha 7.1, isaga oo ku tusaaya fariimahan midkood markaad jiirka dulsaarto:

The method `getName()` is undefined for the type `Course`
 The method `getNumber()` is undefined for the type `Course`
 The method `getDepartment()` is undefined for the type `Course`

Ku kordhi ficiladaas nooca Course sida ka muuqata tusaalaha 7.3:

```

package nimi.course;
2
public class Course {
4
    ...
6
    public String getName() {
8        return null;
    }
10
    public String getNumber() {
12        return null;
    }
14
    public String getDepartment() {
16        return null;
    }
18 }

```

Tusaale 7.3: Nooca Course oo lagu kordhiyey ficilada getName, getNumber iyo getDepartment

Intaas kadib tijaabada 7.1 haddii aad kaxayso cilad (error) ma imanayso, laakiin tijaabadu way guuldaraysanaysaa, sababtoo ah nooca Course uma shaqaynaayo sida tijaabadu ka sugayso, tusaale ahaan ficilada getName, getNumber iyo getDepartment ma soo celinayaan qiimayaashii laga rabay. Si ay tijaabadu u guulaysato, waa inaynu wax ka badalo nooca Course oo ku darno wixii ka dhiman. Ku dar nooca Course sifooyinka ka dhiman oo u eekaysii tusaalaha 7.4.

```

package nimi.course;
2
public class Course {
4
    private String name;
6    private String number;
    private String department;
8
    public Course(String name, String number,
10        String department) {
        this.name = name;
12        this.number = number;
        this.department = department;
14    }

16    public String getName() {
        return this.name;
18    }

20    public String getNumber() {
        return this.number;
22    }

24    public String getDepartment() {
        return this.department;
26    }
}

```

Tusaale 7.4: Nooca Course oo lagu kordhiyey sifooyinka name, number iyo department

Intaas kadib haddii aad kaxayso tijaabada CourseTest waa inay cagaar noqoto, taas oo markhaati ka ah in nooca Course uu u shaqaynaayo sidii tijaabadu ka sugaysey. Aynu sii wadno tijaabinta nooca Course, gaar ahaan nambarka koorsada. Waxaynu rabnaa haddii la rabo in la sameeyo koorso cusub, laakiin la siiyo nambar aan buuxinaynin shuruuddii aynu u dhignay nambarka koorsada, inay guuldaraysato samaynta koorsadaas cusub. Waxaad nooca CourseTest ku dartaa ficilka testCourseNumber sida ka muuqata tusaalaha 7.5.

```

package nimi.course;
2
import static org.junit.Assert.*;
4
import org.junit.Test;
6
public class CourseTest {
8
    ...
10
    @Test

```

```

12 public void testCourseNumber() throws Exception {
13     try {
14         Course course = new Course("Af-soomaali 1", "abc",
15             "Luqadaha");
16         fail("Waxaynu filanaynay exception, oo in halkan "
17             + "borogaraamku soo gaadho may ahayn!");
18     } catch (IllegalCourseNumberException expected) {
19         String message = expected.getMessage();
20         assertEquals("'abc' waa lambar koorso oo qalad ah!",
21             message);
22     }
23 }
24 }

```

Tusaale 7.5: Qoraalka CourseTest oo lagu daray ficilka testCourseNumber oo tijaabinaysa nambarka koorsada

Aan sharaxaad ka bixiyo tusaalaha 7.5:

1. Marka hore exceptions-ka waxaynu kagasoo hadalay qaybta 6.1. Waa markii ugu horaysey ee aynu borogaraamkeena ku isticmaalo. Ereyada **try** iyo **catch** waxaad ku soo martay qaybtaas. Laynka 14 (oo ku dhammaanaaya 15) waxaynu samaynaynaa walax cusub oo noocedu yahay Course. Walaxdaas koorsada ah oo aynu siinay qiimayaasha magaca koorsada oo ah "Af-soomaali 1", nambarka koorsada oo ah "abc" iyo qaybta koorsada oo ah "Luqadaha"
2. Laynka ka hooseeya ee 16 waxaynu isticmaalaynaa ficilka *fail()* oo qayb ka ah JUnit. Ficilkaasi wuxuu cadaynayaa in haddii la soo gaadho laynkaas inay tijaabadu guuldaraysato oo keento casaan. Taas oo sheegaysa in borogaraamku soo gaadhey meel qalad ah oo ay ahayd inaanu soo gaadhin. Waxaynu rabnay in borogaraamku laynkaas ka boodo, maadaama aynu siinay qiimaha nambarka koorsada oo qalad ah. Haddii uu laynkaasi ka boodi waayo, waxay ka dhigantahay in borogaraamkeenu ku shaqaynaayo qiimaha qalad ka ah, sidaas awgeed uu qalad u shaqaynaayo
3. Haddii talaabooyinka ku qoran caloosha **try** midkood soo tuuro walax exception ah oo noocedu yahay IllegalCourseNumberException, waxaynu ku qabanaynaa oo ku xalinaynaa laymanka 18-22 ee isla tusaalaha. Magaca ama reference-ka walaxdaas waxaynu ka dhignay expected, laakiin waxaad u bixin kartaa magaca aad rabto. expected waxaynu ugu bixinay maadaama ay tahay mid la sugaayey. Maadaama aynu siinay qiime qaldan nambarka koorsada laynka 14-aad, waxaynu sugaynaa in walax exception ah oo noocedu yahay IllegalCourseNumberException lasoo tuuro. Haddii taas ay dhiciwaydo, borogaraamku qalad ayuu u shaqaynayaa oo wuxuu ku dhacayaa ficilka fail, ee laynka 16-aad

Eclipse wuxu ka cabanayaa nooca IllegalCourseNumberException, oo uu leeyahay noocaasi ma jiro. Sidaas awgeed, samee nooc cusub adiga oo raacaya talaabada **NewClass**, waxaad u buuxisaa sidan: Source Folder=nimi/src, Package=nimi.course iyo Name=IllegalCourseNumberException. Kadib, waxaad u eekaysiisaa tusaalaha 7.6. Waxa isna u baahan in waxyar laga badalo nooca Course gaar ahaan ficilka dhisaha (constructor) sida ka muuqata tusaalaha 7.7. Ku kordhi qoraalka **throws** IllegalCourseNumberException dhamaadka magaca ficilka. Taasi waxay sheegaysaa in ficilkaasi walxo exception ah oo noocodu yahay IllegalCourseNumberException uu tuuri karo.

```

package nimi.course;

2 public class IllegalCourseNumberException extends Exception {
3
4     public IllegalCourseNumberException(String msg) {
5         super(msg);
6     }
7
8 }

```

Tusaale 7.6: Nooca exception-ka ee IllegalCourseNumberException

```

1 package nimi.course;
2
3 public class Course {
4
5     ...
6
7     public Course(String name, String number,
8         String department)
9         throws IllegalCourseNumberException {

```

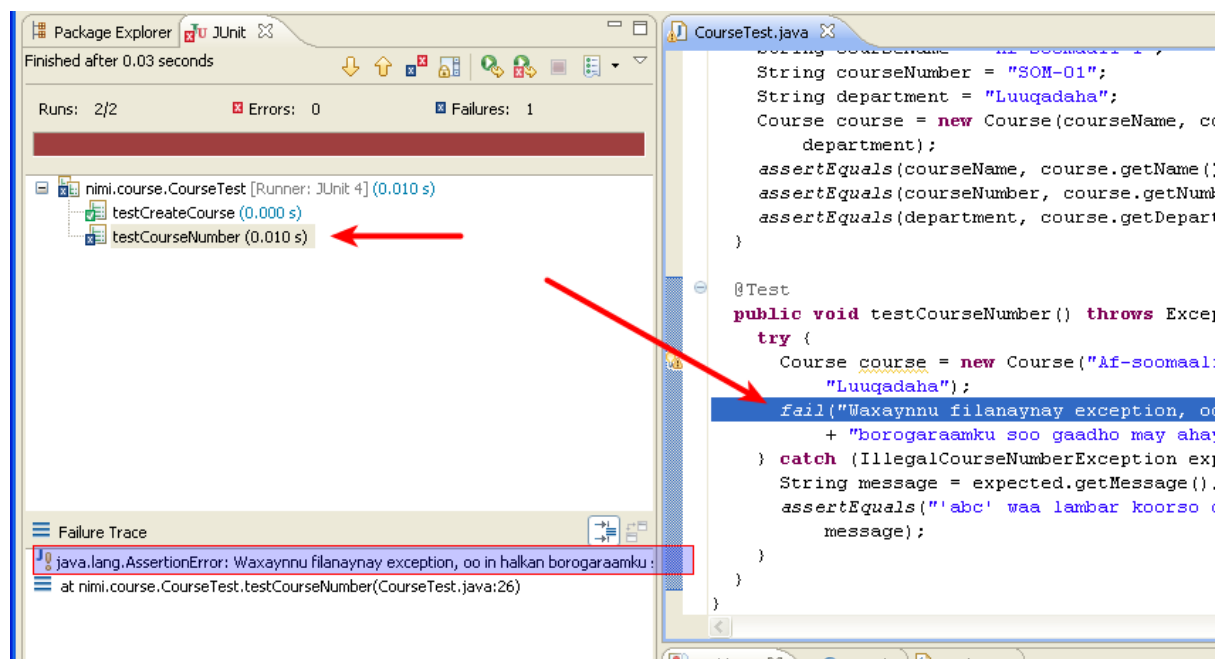
```

11 }
13 ...
}

```

Tusaale 7.7: Ficilka dhisaha oo lagu daray `IllegalCourseNumberException`

Intaas kadib, haddii aad kaxayso nooca `CourseTest` waa inay kuu soo baxdo sawirka 7.2 xaaalad la mid ah. Sawirka waxaad ku arkaysaa in tijaabada `testCreateCourse` guulaysatay, laakiin `testCourseNumber` guuldaraysatay, sababtana waxaad ka arkaysaa sawirka.



Sawirka 7.2: Nooca `TestCourse` oo ka kooban laba tijaabo `testCreateCourse` iyo `testCourseNumber` oo la kaxeeyey

Si haddaba ay tijaabada `testCourseNumber` u guulaysato waa inaynu nooca `Course` logic-ga ka maqan ku darno, gaar ahaan ficilka dhisaha ah ee `Course`. Tusaalaha 7.8 ayaad ka arkaysaa isbadalka lagu sameeyey nooca `Course`.

```

1 package nimi.course;
2
3 public class Course {
4
5     ...
6
7     public Course(String name, String number,
8         String department)
9         throws IllegalArgumentException {
10         if (number == null || number.length() != 6
11             || number.charAt(3) != '-'
12             || !Character.isDigit(number.charAt(4))
13             || !Character.isDigit(number.charAt(5))) {
14             throw new IllegalArgumentException("'" + number
15                 + "' waa lambar koorso oo qalad ah!");
16         }
17         this.name = name;
18         this.number = number;
19         this.department = department;
20     }
21
22     ...
23 }

```

Tusaale 7.8: Ficilka dhisaha oo lagu daray tuurista exception-ka `IllegalArgumentException`, haddii nambar qalad ah la siiyo koorlada

Intaas kadib haddii aad mar kale kaxayso nooca `CourseTest`, waa inay labada tijaabo ka gudbaan oo cagaar noqdaan.

7.1 ToString

Ficilka `toString` waxa loogu talagalay inuu soo celiyo qoraal (string) matalaaya walaxdan. Waxaad nooca `CourseTest` ku dartaa ficilka `testToString` sida ka muuqata tusaalaha 7.9.

```

1 package nimi.course;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class CourseTest {
8
9     ...
10
11     @Test
12     public void testToString() throws Exception {
13         Course course = new Course("Af-soomaali 1", "SOM-01",
14             "Luuqadaha");
15         System.out.println(course);
16         assertEquals("Af-soomaali 1 (SOM-01)", course.toString());
17     }
18 }

```

Tusaale 7.9: Qoraalka `CourseTest` oo lagu daray ficilka `testToString` oo tijaabinaysa u badalista walaxda `Course`-ka ah qoraal

Haddii aad kaxayso nooca `CourseTest`, tijaabada `testToString` way guuldaraysanaysaa, laakiin labada tijaabo ee kale way guulaysanayaan. Laynka 15 ee tusaalaha 7.9 wuxuu keenayaa in console-ka aad ka aragto fariin u eg sidan:

```
nimi.course.Course@77c99159
```

Si haddaba tijaabadani u pass-garayso, waa in marka walaxda koorlada ah la isticmaalo ficilkeeda `toString` sida ka muuqata laynka 16 ay soo celiso qoraal la fahmi karo oo ka turjumaaya koorladaas. Tijaabadan waxaynu sugaynaa in koorlada aynu samaynay soo celiso "Af-soomaali 1 (SOM-01)" oo ka kooban magaca iyo nambarka koorlada. Si tijaabada `testToString` u pass-garayso nooca `Course` dhammaadkiisa ku dar ficilka `toString` ee ka muuqda tusaalaha 7.10. Kadib kaxee nooca `CourseTest` si aad u aragto in dhammaan tijaabooyinku pass-garaynayaan oo cagaar yihiin.

```

1 package nimi.course;
2

```

```

1 public class Course {
2     ...
3
4     public String toString() {
5         return this.name+" (" + this.number + ")";
6     }
7 }
8
9
10

```

Tusaale 7.10: Ficilka toString ee nooca Course

Nooca CourseTest markaad kaxayso waxaad console-ka ka arkaysaa

Af-soomaali 1 (SOM-01)

oo ah wixii aynu ka filanaynay. Ficilka toString waxa loogu talagalay inuu walaxda qoraal u badalo. Qoraalkaas oo ama fayl lagu qori karo ama isticmaalaha la tusi karo.

7.2 Object

Haddii aanad ficilka toString u samayn nooca, wuxuu ka dhaxlayaa nooca Object oo ah aabaha dhammaan noocyada Java. Aan si kale u sheegno, dhammaan noocyada Java waxay subclass u yihiin nooca Object, xitaa haddii aanad si cad u qeexin ama u sheegin. Aan tusaale u soo qaato nooca Person oo aynu hore u soo aragnay. Nooca Person wuxu u eekaa sida 7.11.

```

1 package nimi.person;
2
3 public class Person {
4     //caloosha nooca Person
5 }

```

Tusaale 7.11: Noocii Person oo waxyar laga soo qaatay

Laakiin xaqiiq ahaan Java wuxu ula mid yahay sida tusaalaha 7.12

```

1 package nimi.person;
2
3 public class Person extends Object {
4     //caloosha nooca Person
5 }

```

Tusaale 7.12: Nooca Person xaqiiqo ahaan sida uu Java uga dhiganyahay

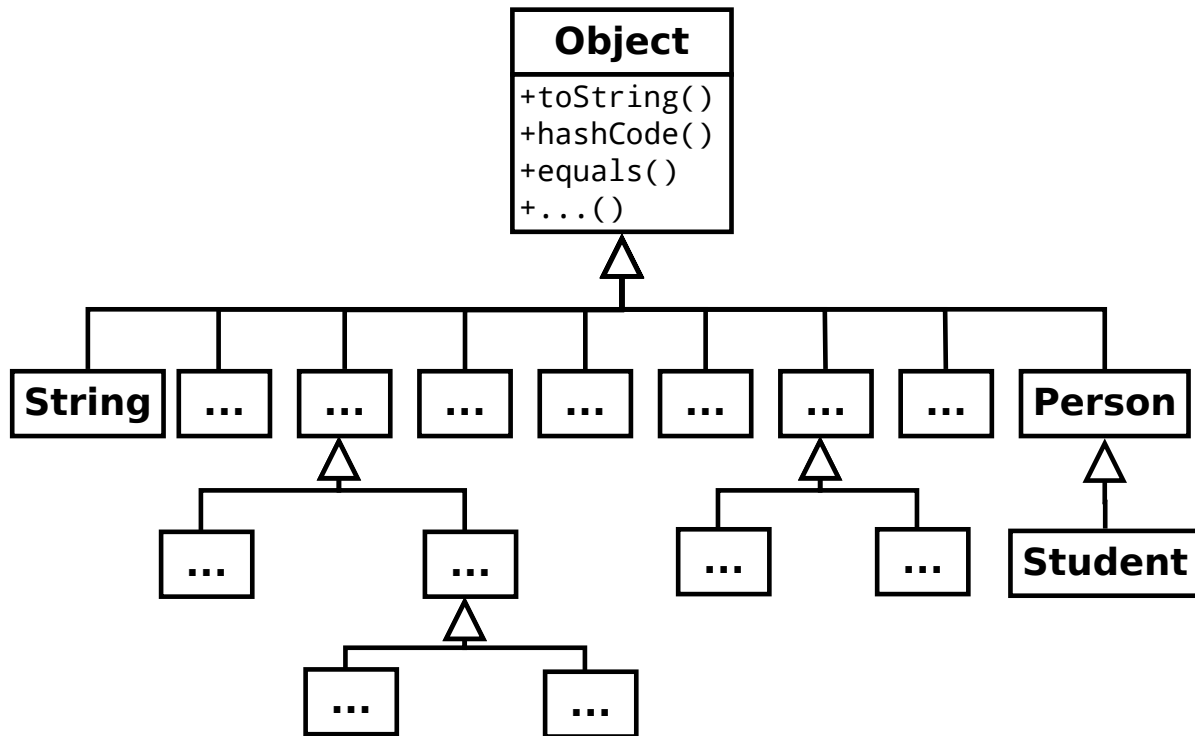
Maadaama dhammaantood noocyada Java ay ka simanyihiin 'extends Object', qasab ma aha in la qoro oo waa laga tegikaraa. Tusaale ahaan nooca Student ee aynu kaga hadalay qaybta ku saabsan ardayda, wuxu subclass u yahay nooca Person, kaas oo subclass u ah nooca Object. Markaas nooc kasta oo Java ah si toos ah ama aan toos ahayn (dadban) ayaa looga soo dhiraandhiriyeey nooca Object. Sawirka 7.3 ayaa ku tusaaya qaabka ay noocyada Java isugu xidhanyihiin iyo in dhammaantood ay nooca Object subclass u yihiin. Haddaba nooca Object ficillo sidee ah ayuu leeyahay oo laga dhaxlayaa? Wuxuu leeyahay ficillo dhawr ah.

Ficilka ugu muhiimsan ee aynu halkan ku soo qaadanay waa ficilka toString. Laakiin ficilka toString ee nooca Object wuxuu kuusoo celinayaa magaca nooca walaxdan oo dhammaystiran oo lagu dabaqabtay calaamadda @ iyo tiro. Inta badan taasi innama anfacyso, oo waxaynu rabnaa qoraal la fahmi karo, sidaas awgeed isla ficilkii toString ayeynu nooca Course u samaynay, calooshiisana ama qaabka uu u shaqeeyana ficilkaasi waxaynu ka dhignay sidii aynu rabnay oo ah inuu soo celiyo magaca koorsada oo lagu dabaqabtay nambarka koorsada. Su'aal ayaa meesha taalla, laynka 16-aad ee tusaalaha 7.9, waxaynu isticmaalay ficilka toString (course.toString()), halkaas marka borogaraamku maraayo ma waxa la isticmaalayaa ficilka toString ee nooca Object mise ka nooca Course (maadaama ay labada noocba leeyihiin ficilka toString)? Waxa la isticmaalayaa ficilka toString ee nooca Course. Aan tusaale yar soo qaato oo aad ka arkayso 7.13.

```

1 package nimi;
2
3 import nimi.course.Course;
4 import nimi.course.IllegalCourseNumberException;
5
6 public class IsbarJava {
7
8     public static void main(String[] args) {
9         try {

```



Sawirka 7.3: Noocyada Java kulligood (kuwaad adigu qorto iyo kuwa diyaarka ahba) waxa laga soo dhiraandhiriyeey nooca Object, kaas oo ay ka dhaxlayaan ficillo dhawr ah oo uu ka midyahay ficilka toString

```

10      Object ob = new Object();
11      Course eng1 = new Course("English 1", "ENG-01",
12          "Luuqadaha");
13      System.out.println(ob.toString());
14      System.out.println(eng1.toString());
15  } catch (IllegalCourseNumberException e) {
16      e.printStackTrace();
17  }
18  }

```

Tusaale 7.13: Tusaale ku saabsan method override-ka (habka kukor qoridda)

Haddii aad kaxayso tusaalaha 7.13, waxaa kuu soo baxaaya (console-ka):

```

java.lang.Object@527c6768
English 1 (ENG-01)

```

Qoraalka sare waxa keenaaya laymanka 13 iyo 14. Waxaad tusaalahan ka arkaysaa marka walaxda noocadu yahay Course ficilkeeda toString la isticmaalo ay kuusoo celinayso qoraal ka duwan, markaad isticmaasho ficil toString walax noocadu yahay Object. Waxa halkan inooga soo baxay astaan kale oo muhiim ah oo uu leeyahay takhasusku. Haddii noocan iyo nooca laga soo dhiraandhiriyeey labadooduba leeyihiin ficil isku magac ah, waxa la isticmaalayaa ficilka noocan, marka ficilkaas la waco. Cashar ama laylis ahaan u samee noocyada Person, Student iyo Teacher ficilka toString.

7.3 Equals

Intaynaan hore u socon, waxa inoo hadhsan hal tijaabo oo ka mid ah nooca CourseTest, sida ka muuqata sawirka 7.1, waa testEquality. testEquality (isl'egta-tijaabada) waxa loogu talagalay inay tijaabiso isku-mid ahaanshaha laba walxood oo koorso ah. Waxaad ka fahmidoontaa borogaraamka tusaalaha waxaan ula jeedo arintaas. Kudar nooca CourseTest ficilka testEquality ee ka muuqda tusaalaha 7.14:

```

1 package nimi.course;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class CourseTest {
8
9     ...
10
11     @Test
12     public void testEquality() throws Exception {
13         Course course1 = new Course("A", "AAA-01", "Dep1");
14         Course course2 = new Course("A", "AAA-01", "Dep1");
15         Course course3 = new Course("B", "BBB-12", "Dep2");
16
17         assertTrue(course1.equals(course1)); //course1==course1?
18         assertEquals(course1, course2); //course1==course2?
19         assertFalse(course1.equals(course3)); //course1==course3?
20
21         assertFalse(course1.equals(null)); //course1==null?
22         assertFalse(course1.equals("Hello")); //course1=="Hello"?
23     }
24 }

```

Tusaale 7.14: CourseTest oo lagu daray ficilka testEquality oo tijaabinaaya iskumid ahaanshaha walxaha noocoodu yahay Course iyo kuwa kale

Aan sharaxaad ka bixiyo borogaraamka 7.14:

1. Laymanka 13-15 waxaynu samaynaynaa saddex walxood oo Course ah oo kala ah: course1, course2 iyo course3. Mid walba waxaynu siinaynaa magac ka kooban hal xaraf, nambar iyo department (qayb). Waxaad halkaas ka arkaysaa in labada walxood ee hore aynu isku qiimo siinay, laakiin ta saddexaad ka duwantahay
2. Laynka 17-aad waxaynu isticmaalaynaa ficilka equals (isle'eg) ee walaxda course1 (oo noocedu yahay Course). Maadaama aanu nooca Course lahayn ficil layidhaahdo equals, wuxuu ka dhaxlayaa nooca Object. Waxaynu hubinaynaa in ficilkaasi soo celinaayo qiimo boolean ah oo ah **true**, inaga oo isticmaalayna ficilka assertTrue ee JUnit. Haddii ficilka equals uu keeno qiime aan ahayn **true**, assertTrue waxay keenaysaa in tijaabadaasi guuldaraysato
3. Laynka 18-aad waxaynu adeegsanaynaa assertEquals oo aynu hore u soo aragnay, kaas oo hubinaaya in course1 iyo course2 isku mid yihiin
4. Laynka 19-aad waxaynu isticmaalaynaa ficilka equals ee nooca Course mar kale. Waxaynu hubinaynaa in ficilkaasi soo celinaayo boolean **false**, marka la siiyo qiime aan la mid ahayn walaxdan. Waxaynu isticmaalayna ficilka assertFalse ee JUnit
5. Laymanka 21 iyo 22 waxaynu adeegsanaynaa ficilka assertFalse ee JUnit, kaas oo aynu ku hubinayno in ficil equals ee nooca Course uu soo celinaayo **false**, marka la siiyo qiima **null** ama qoraalka "Hello". Sidaasi waa habka la rabo inuu ficilka equals u shaqeeyo, waayo labadaas qiime midkoodna lama mid aha walaxda noocedu yahay Course, sidaas awgeed ficilka equals marka la siiyo qiime aan ahayn walax Course ah, waa inuu **false** soo celiyo.

Haddii aad kaxayso nooca CourseTest, tijaabada testEquality oo kaliyaa guuldaraysanaysa, inta kale waa cagaar, maadaama aynu hore u soo qornay. Haddaba aynu cagaar ka dhigno ficilkaasna. Waa yahay. Goorma ayay laba walxood oo noocoodu yahay Course isku mid yihiin? Waxay isku mid yihiin, haddii ay isku koorso matalayaan. Waa haddii dhammaan sifooyinkooda name, number iyo department isku mid yihiin.

Nooca Course waxaad ku dartaa ficilka equals ee ka muuqda tusaalaha 7.15:

```

1 package nimi.course;
2
3 public class Course {
4
5     ...
6
7     public boolean equals(Object object) {
8         if (object == null) {
9             return false;
10        }

```

```

12     if (this.getClass() != object.getClass()) {
        // noociisu ma aha Course
        return false;
14     }
    Course that = (Course) object;
16     if (this.name.equals(that.name)
        && this.number.equals(that.number)
18         && this.department.equals(that.department)) {
        return true;
20     } else {
        return false;
22     }
24 }

```

Tusaale 7.15: Ficilka equals ee nooca Course

Ficilka equals waxa loogu talagalay in lagu tijaabiyo in walaxdan iyo mid kale (that) ay isku midyihiin iyo in kale. Walaxdaas kale waxay noqon kartaa mid ay tan isku noocyihiin ama mid ka duwan, sidaas awgeed laymanka 11-14 waxaynu hubsanaynaa inay isku qolo yihiin walaxdan iyo taa kale. Waxa kale oo dhici karta in walaxda kale tahay mid aan jirin **null**, taasna waxaynu hubinaynaa laymanka 8-10 ee 7.15.

Intaynaan hore u socon kaxee nooca CourseTest is aad u aragto inuu cagaar bixinaayo. Waxaad nooca AllTests ku dartaa nooca CourseTest sida ka muuqata 7.16.

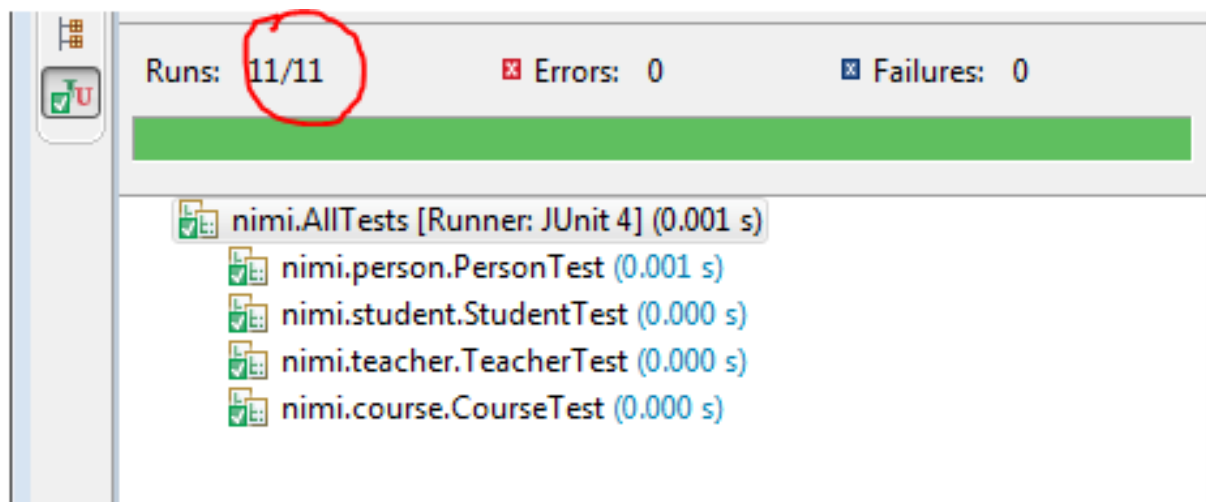
```

package nimi;
2
import org.junit.runner.RunWith;
4 import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;
6
@RunWith(Suite.class)
8 @SuiteClasses({
    nimi.person.PersonTest.class, // PersonTest
10    nimi.student.StudentTest.class, // StudentTest
    nimi.teacher.TeacherTest.class, // TeacherTest
12    nimi.course.CourseTest.class // CourseTest
})
14 public class AllTests {}

```

Tusaale 7.16: AllTests oo lagu daray CourseTest

Kaxee nooca AllTests, si aad u aragto in wixii aynu tijaabooyin soo qorno illaa hadda dhammaantood cagaar yihiin. Sida sawirka 7.4 ka muuqada waxaynu haysanaa 11 tijaabo oo dhammaantood guulaysanayaan, taasi waxay sheegaysaa in borogaarka qaybihiisa kale duwan ee aynu tijaabinay sidii loogu talagalay u shaqaynayaan.



Sawirka 7.4: Nooca AllTests oo lagu kordhiyey nooca CourseTest. Dhammaan tijaabooyinka aynu illaa hadda haysano waa 11 sida ka muuqata sawirkan.

7.4 Gunaanad

Intaynaan ka gudbin qaybtan waxaad nooca Course ku dartaa saddexda ficil ee tusaalaha 7.17 ee fudud ee badalaaya qiimayaashiisa kala duwan. Ficilada noocan oo kale ah waxa loo yaqaan *setters*. Qaybtan waxyaalaha ugu muhiimsan ee aad ku baratay waxa kamid ah ficilada `toString` iyo `equals` iyo kaalinta uu ku jiro nooca Object.

```
1 public class Course {  
2     ...  
4     public void setName(String name) {  
5         this.name = name;  
6     }  
8     public void setNumber(String number) {  
9         this.number = number;  
10    }  
12    public void setDepartment(String department) {  
13        this.department = department;  
14    }  
15    ...  
16 }
```

Tusaale 7.17: Ficilada setters-ka oo lagu diray nooca Course

Cutub 8

Fadhiyada Koorsooyinka

Macluumaadka koorradu sida magaca, nambarka, qaybta ama department-ka, iwm inta badan isma badalaan sanad walba. Tusaale ahaan koorrada "Af-soomaali 1" oo nambarkedu yahay "SOM-01" waxa suurto ah inaanay is badalin inta iskuulkaasi jiro.

Sanadka waxbarasho ama sanad-dugsiyeedka (academic year) waxa loo qaybiyaa laba, saddex ama afar qaybood. Wadamada adduunka way ku kala duwanyihiin xilli sanadeedka. Wadamada isticmaala semesterka (saddex bilood ama afar bilood) inta badan waxay sanadka u qaybiyaan laba qaybood.

Fadhi (course session ama session) waa qabsomidda koorrada semesterkiiba. Waxa jiri kara koorsooyin la qabto semester walba ama qaar la qabto sanad walba semesterka ugu horeeya. Halkaas waxaad ka arkaysaa in koorradu yeelankarto hal ama dhawr fadhi sanadkiiba. Xitaa waxa suurto ah inaanay yeelan wax fadhi ahba, waa haddii koorradaas aan la dhiganaynin sanadkaas.

Sifooyinka fadhiga waxa ka mid ah:

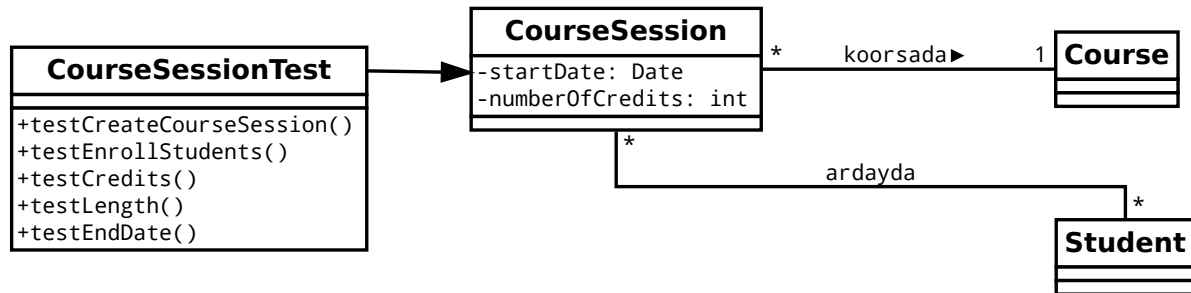
1. Taariikhda ama maalinta la bilaabaayo
2. Taariikhda ama maalinta uu dhammaanaayo
3. Meesha ama fasalka lagu qabanaayo fadhigan
4. Ardayda ka qaybgalaysa fadhigaa
5. Macalinka dhigaaya
6. Inta dhibcodd ama credit ee laga helaayo koorradaas marka ardaygu dhameeyo

Intaas waa inta aynu halkan ku xasuusanay, laakiin way ka badnaan karaan. Iskuulka aynu nidaamkan u samaynaynaa wuxuu inoo sheegay in fadhiyadiisa aan la qeexin wakhtiga ay dhammaanayaan, kaliya la sheegay taariikhda ay bilaabmayaan iyo in fadhi walba dhererkiisu 16 usbuuc ama wiig yahay. Waxa kale oo aynaan halkan ku soo qaadnaynin meesha lagu qabanaayo fadhiga iyo macalinka dhigaaya. Taas layli ahaan ayaa ardayga loogu daayey. Sifooyinka uu fadhigu yeelanaayo waxay iskugu soo hadheen:

1. Taariikhda ama maalinta la bilaabaayo
2. Ardayda ka qaybgalaysa fadhigaa
3. Inta dhibcodd ama credit ee laga helaayo koorradaas marka ardaygu dhameeyo

Waxaynu bilaabaynaa qoraalkii fadhiyada. Fadhiiga waxaynu u isticmaalaynaa magaca CourseSession. Nooca CourseSession iyo xidhiidhka uu la leeyahay noocyada kale waxad ka arkaysaa sawirka 8.1.

Haddii aan sharaxaad ka bixiyo sawirka 8.1 intaynaan hore u socon. Sidi horeba, waxaad sawirka ka arkaysaa nooca CourseSession iyo noociisa tijaabiyaha ah ee CourseSessionTest. Waxa kale oo aad sawirka ka arkaysaa in nooca CourseSession uu xidhiidh la leeyahay noocyada Course iyo Student oo aynu hore u soo qaadnay. Xidhiidhka noocyada ka dhexeeya waa wax innagu cusub, sidaas awgeed waxa loo baahanyahay inaynu sharaxaad yar ka bixiyo intaynaan hore u socon. Xidhiidhka labada nooc ama class ka dhexeeya waxa loo yaqaan *association*. Xidhiidhku wuxuu yeelan karaa magac sida ka muuqata sawirka, jaho (direction) iyo nambar (multiplicity). Magacu waa qoraal caadi ah oo sharaxaaya xidhiidhkaas. Jahadu waxay sheegaysaa dhinaca loo akhriyaayo xidhiidhkaas, laakiin qasab ma aha. Tirada ama multiplicity-ga xidhiidhka waxay sheegaysaa inta walxood ee labadaas nooc ah ee isku xidhnaan karta. Xidigtu "*" waxay ka dhigantahay eber ama in ka badan. Xidhiidhka u dhexeeya CourseSession iyo Course wuxuu sheegayaa in walax kasta oo noocedu yahay CourseSession ay ku xidhantahay hal ama mid koorso, laakiin dhanka kale walaxda noocedu yahay Course waxa ku xidhnaan kara eber ama in ka badan



Sawirka 8.1: Nooca CourseSession iyo noocyada kale ee uu xidhiidhka la leeyahay

oo CourseSession ah. Halkaas waxad ka arkaysaa in aanu jiri karin walax CourseSession ah oo aan ku xidhnayn hal koorso.

Sidoo kale xidhiidhka u dhexeeya nooca CourseSession iyo Student wuxu sheegayaa in CourseSession-kii oo matalaaya fadhiga koorso ay kasoo qaybgali karaan eber ama in ka badan oo arday ah, ardayguna uu ka qaybgali karo eber ama in ka badan oo CourseSession ama fadhi ah.

Sawirada sidan oo kale ah, sidaan hore u soo sheegnayba waxay qayb ka yihiin standard-ka UML. Haddii aad rabto inaad si qoto dheer u barato standard-ka UML waxad gashaa bogga internet-ka ee <http://www.uml.org/>.

Waxaynu bilaabaynaa qoristii nooca CourseSession oo matalaaya fadhiyada koorsooyinka. Waxaynu ku bilaabaynaa qorista nooca tijaabihiisa ah ee CourseSessionTest. Samee noocan cusub adiga oo raacaya talaabada **NewTest**, una buuxbuux sidan: Source Folder=nimi/tst, Package=nimi.course iyo Name=CourseSessionTest. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo CourseSessionTest, laakiin madhan. CourseSessionTest waxaad ku bilawdaa sida tusaalaha 8.1.

```

1 package nimi.course;
2
3 import static org.junit.Assert.*;
4
5 import java.util.Date;
6 import java.util.GregorianCalendar;
7
8 import org.junit.Before;
9 import org.junit.Test;
10
11 public class CourseSessionTest {
12
13     private Course course;
14     private Date timeNow;
15     private int numberOfCredits;
16     private CourseSession courseSession;
17
18     @Before
19     public void before() throws Exception {
20         String courseName = "Af-soomaali 1";
21         String courseNumber = "SOM-01";
22         String department = "Luuqadaha";
23         this.course = new Course(courseName, courseNumber,
24             department);
25         GregorianCalendar calendar = new GregorianCalendar();
26         this.timeNow = calendar.getTime();
27         this.numberOfCredits = 3;
28         this.courseSession = new CourseSession(this.course,
29             this.timeNow, this.numberOfCredits);
30     }
31
32     @Test
33     public void testCreateCourseSession() throws Exception {
34         assertNotNull(this.courseSession);
35         assertNotNull(this.courseSession.getCourse());
36     }
37 }
  
```

```

36 assertEquals(3, this.courseSession.getNumberOfCredits());
38 assertEquals(this.timeNow, this.courseSession.getStartDate());
}
}

```

Tusaale 8.1: Qoraalka CourseSessionTest bilawgiisa

Aynu sharaxaad ka bixino borogaraamka 8.1:

1. Waa markii ugu horaysay ee nooc tijaabiye ah aynu u samayno sifooyin. Laymanka 13-16 waxad ka arkaysaa in nooca CourseSessionTest leeyahay afar sifood. Sifada ugu horaysa noocedu waa Course oo aynu hore u soo aragnay, waxaanay matalaysaa koorsada uu fadhigani la xidhiidho. Sifada xigta noocedu waa Date, kaas oo ka mid ah luuqadda Java, kuna jira package-ka *java.util*. Date wuxu midaynayaa maalinta, bisha, sanadka iyo saacadda oo wuxu tilmaamayaa wakhtiga koorsadu bilaabmayso. Sifada numberOfCredits, waxay sheegaysaa inta dhibcood ama credit ee koorsadaas ardaygu ka helaayo marka uu dhameeyo. Ugu dambayntii nooca CourseSession wuxu matalayaa qabsoomidda koorsada
2. Waxa kale oo innagu cusub ficilka sumadda @Before leh ee before ee laymanka 18-30. Ficilkaas magaciisa iyo calooshiisa waxaad rabto ayaad ka dhigi kartaa, laakiin sumadda @Before waxay leedahay ujeeddo kale oo JUnit fahmaayo. Ficilka @Before waxa loogu talagalay in laga horaysiiyo kaxaynta ficil kasta oo sumadda @Test leh. Ujeedadiisuna waa inuu ficilada tijaabada ah ee sumadda @Test leh u diyaariyo waxyaalaha ay u baahanyihiin. Halkan ficilka @Before wuxu qiimeyaal siinaayo sifooyinka sare oo wuxu samaynayaa koorso cusub, waxa kale oo uu adeegsanayaa ficilka getTime ee nooca GregorianCalendar, si uu u helo taariikhda iyo saacadda imika. Ugu dambayntii wuxu samaynayaa nooc cusub oo ah CourseSession, kaas oo uu u gudbinaayo qiimeyaasha ku kaydsan sifooyinka nooca
3. Tijaabada ugu horaysa waa ficilka testCreateCourseSession ee laymanka 32-38. Ficilkaasi laynkiisa 34 wuxu hubinaaya in qiimaha courseSession aanu ahayn **null** (taas oo kadhigan inuu yahay walax jirta). Laynka 35 waxaynu hubinaynaa in walaxda courseSession marka la isticmaalo ficilkeeda getCourse ay soo celinayso qiime aan ahayn **null**. Laynka 36 waxaynu hubinaynayaa in ficilka getNumberOfCredits ee walaxda courseSession uu soo celinaayo qiimaha 3, maadaama aynu qiimahaas siinay ficilka dhisaha ee laynka 28. Laynka ugu dambeeyana waxaynu hubinaynaa in ficil getStartDate soo celinaayo maalinta aynu siinay.

Haddii aad kaxayso nooca CourseSessionTest wuxu ku istaagayaa cilad. Eclipse wuxu ka cabanayaa laynka 16-aad, taas oo uu leehayahay:

CourseSession cannot be resolved to a type

Taas oo sheegaysa inaanu nooca CourseSession jirin. Si aynu uga gudubno ciladan samee nooc cusub, una buuxbuuxi sidan: Source Folder=nimi/src, Package=nimi.course iyo Name=CourseSession. Waa inaad hesho nooc cusub oo magaciisa la yidhaahdo CourseSession, laakiin madhan una eeg tusaalaha 8.2.

```

1 package nimi.course;
2
3 public class CourseSession {
4
5 }

```

Tusaale 8.2: Bilawga nooca CourseSession

Intaas kadib, Eclipse wuxu ku tusayaa in laynka 28-aad ee CourseSessionTest ciladdi ka jirto, isaga oo ku tusaaya fariinta markaad jiirka dulgayso laynka ciladdu ka jirto:

The constructor CourseSession(Course, Date, int) is undefined

Taas oo uu leehayahay in aanu nooca CourseSession lahayn ficil constructor-ka ama dhisaha ah ee sidaas u eeg. Talaabada xigta waa inaad ficilka constructor-ka ku darto nooca CourseSession, una eekaysiiso 8.3:

```

1 package nimi.course;
2
3 import java.util.Date;
4
5 public class CourseSession {

```

```

7  private Date startDate;
   private Course course;
9  private int numberOfCredits;

11 public CourseSession(Course course, Date startDate,
   int numberOfCredits) {
13     this.course = course;
   this.startDate = startDate;
15     this.numberOfCredits = numberOfCredits;
   }
17 }

```

Tusaale 8.3: Nooca CourseSession oo lagu daray ficilka constructor-ka ama dhisaha ah, laguna daray sifooyinkii

Intaas kadib, Eclipse wuxu ka cabanayaa laymanka 35 iyo 36 isaga oo leh:

```

The method getCourse() is undefined ...
The method getNumberOfCredits() is undefined ...
The method getStartDate() is undefined ...

```

oo ka dhigan in aanu nooca CourseSession lahayn ficilada getCourse, getNumberOfCredits iyo getStartDate. Kudar labadaas ficil nooca CourseSession si uu ugu eekaado tusaalaha 8.4.

```

1  package nimi.course;

3  import java.util.Date;

5  public class CourseSession {

7      private Date startDate;
       private Course course;
9      private int numberOfCredits;

11     public CourseSession(Course course, Date startDate,
        int numberOfCredits) {
13         this.course = course;
        this.startDate = startDate;
15         this.numberOfCredits = numberOfCredits;
        }

17     public Course getCourse() {
19         return this.course;
        }

21     public int getNumberOfCredits() {
23         return this.numberOfCredits;
        }

25     public Date getStartDate() {
27         return this.startDate;
        }

29 }

```

Tusaale 8.4: Nooca CourseSession oo lagu daray ficilo cusub

Intaas kadib haddii aad kaxayso nooca CourseSessionTest adiga oo raacaya talaabada RunTest, waa inaad hesho cagaar oo tijaabadii ugu horaysay guulaysato. Waa arin fiican. Aynu hore uga socono. Tijaabada xigta ee aynu ku darayno nooca CourseSessionTest waa maxay ayey kula tahay? Waxaynu samaynaynaa tijaabo fudud oo tijaabinaysa in dhibcaha ama credits-ka nooca CourseSession la badali karo. Tijaabadaas waxaynu u bixinay testCredits. Ku dar ficilkaas tijaabada ah nooca CourseSessionTest sida ka muuqata tusaalah 8.5.

```

1  package nimi.course;

3  import static org.junit.Assert.*;

5  import java.util.Date;
   import java.util.GregorianCalendar;

7  import org.junit.Before;
   import org.junit.Test;

11 public class CourseSessionTest {

```

```

13  ...
15  @Test
16  public void testCredits() throws Exception {
17      assertEquals(3, this.courseSession.getNumberOfCredits());
18      this.courseSession.setNumberOfCredits(4);
19      assertEquals(4, this.courseSession.getNumberOfCredits());
20  }
21  }

```

Tusaale 8.5: Ficilka testCredits ee nooca CourseSessionTest

Haddii aan sharaxaad yar ka bixiyo tusaalaha 8.5 gaar ahaan ficilkiisa testCredits. Marka hore waxaynu hubinaynaa in walaxda courseSession marka la waydiiyo tiradeeda credit inay tahay saddex (maadaama aynu intaas ka dhignay markii aynu walaxdan samaynaynay ficil before). Kadib, waynu badalaynaa tirada credit-ka, innaga oo adeegsanayna ficilka setNumberOfCredits oo waxaynu ka dhigaynaa afar. Intaas kadib waxaynu hubinaynaa inay afar tahay qiimaha uu inoosoo celiyo ficil getNumberOfCredits ee walaxda courseSession. Eclipse markiiba wuxu ka cabanayaa ficilka setNumberOfCredits oo uu leeyahay ma jiro, sidaas awgeed ku dar nooca CourseSession ficilkaas sida ka muuqata 8.6.

```

1  package nimi.course;
2
3  import java.util.Date;
4
5  public class CourseSession {
6
7      ...
8
9      public void setNumberOfCredits(int credits) {
10         this.numberofCredits = credits;
11     }
12 }

```

Tusaale 8.6: Nooca CourseSession oo lagu daray ficilka badalaaya dhibcaha

Xawaare aan caadi ahayn ayaynu ku soconaa. Maxaa loo baahanyahay inaynu tijaabino oo xiga? Waxaynu samaynaynaa ficil tijaabinaaya in nooca CourseSession sax u xisaabinaayo maalinta uu fadhigu dhammaanaayo. Maadaama aynu siinay maalintii fadhigu bilaabmaayey, aynu ognahayna in dhammaan fadhiyada dhererkoodu yahay 16 wiig/usbuc, waa in nooca CourseSession sax u xisaabiyo maalinta uu fadhigu dhammaanaayo. Tusaale ahaan haddii fadhigu bilaabmo 2.7.2012, waa inuu dhamaadka inoo xisaabiyo 22.10.2012. Sidoo kale haddii uu bilaabmo 13.8.2012, waa inuu dhamaadka xisaabiyo 3.12.2012. Kudar ficilka testEndDate nooca CourseSessionTest sida ka muuqata tusaalaha 8.7.

```

1  package nimi.course;
2
3  import static org.junit.Assert.*;
4
5  import java.util.Date;
6  import java.util.GregorianCalendar;
7
8  import org.junit.Before;
9  import org.junit.Test;
10
11 public class CourseSessionTest {
12
13     ...
14
15     @Test
16     public void testEndDate() throws Exception {
17         Date startDate = getDate(2, 7, 2012);
18         CourseSession session = new CourseSession(this.course,
19             startDate, 5);
20         Date endDate = getDate(22, 10, 2012);
21         assertEquals(endDate, session.getEndDate());
22         startDate = getDate(13, 8, 2012);
23         session = new CourseSession(this.course, startDate, 3);
24         endDate = getDate(3, 12, 2012);
25         assertEquals(endDate, session.getEndDate());
26     }
27
28     private Date getDate(int date, int month, int year) {
29         GregorianCalendar calendar = new GregorianCalendar();
30         calendar.clear();
31         calendar.set(year, month - 1, date);
32         return calendar.getTime();
33     }
34 }

```

```

    }
34 }

```

Tusaale 8.7: Ficilka testEndDate ee nooca CourseSessionTest

Aan sharaxaad yar ka bixiyo 8.7 intaynaan hore u socon.

1. Ficilka testEndDate ee laymanka 15-26, wuxu eegayaa in nooca CourseSession sax u xisaabinaayo maalinta dhammaadka fadhiga. testEndDate wuxu adeegsanayaa ficilka getDate ee laymanka 28-33. getDate wuxu soo celinayaa object ama walax noocadu yahay Date, taas oo matalaysa maalinta, bisha iyo sanadka la siiyey. Nooca GregorianCalendar ee aynu ku adeegsanay laynka 29, wuxu matalayaa nidaamka calendar-ka ee aduunka intiisa badan laga isticmaalo. Macluumaad dheeraad oo noocan ku saabsan waxad ka helayaa *J2SE API Documentation*, kaas oo aynu dhawaan waxyar ka sheegi doono
2. Laynka 17-aad waxaynu samaynaynaa taariikhda 2.7.2012 oo ah maalinta fadhigu bilaabmaayo. Kadib laynka 18-aad waxaynu samaynaynaa CourseSession cusub innaga oo siinayna koorsadii uu ku xidhnaa, maalintii bilawga iyo inta credit ee laga helaayo
3. Laynka 20-aad waxaynu samaynaynaa taariikhda 22.10.2012, taas oo 16 usbuuc kadib 2.7.2012. Kadib, laynka 21-aad waxaynu hubinaynaa in maalinta uu ficilka getEndDate ee CourseSession soo celiyo iyo maalinta aynu filanaynay isku mid yihiin
4. Laymanka 22-25 waxaynu hubinaynaa dhamaadka fadhi labaad oo bilaabmaaya 13.8.2012.

Intaas kadib Eclipse wuxu ka cabanayaa laymanka 21 iyo 25 oo uu leeyahay

The method getEndDate() is undefined for the type ...

maadaama uu ficil getEndDate ka maqanyahay nooca CourseSession. Talaabada xigta waa inaynu ficilkaas ku darno nooca CourseSession sidaad ka arkayso 8.8

```

package nimi.course;
2
import java.util.Date;
4
public class CourseSession {
6
    ...
8
    public Date getEndDate() {
10        return null;
    }
12 }

```

Tusaale 8.8: Ficil getEndDate oo lagu daray nooca CourseSession

Intaas kadib haddii aad kaxayso tijaabada CourseSessionTest, waxad arkaysaa in labada tijaabo ee hore guulaysanayaan, laakiin tijaabada testEndDate ay guuldaraysanayso, maadaama ficil getEndDate ee CourseSession aanu u shaqaynaynin sidii laga rabay. Talaabada xigta waa inaynu kaga shaqaysiino ficilka getEndDate illaa iyo inta ay tijaabada testEndDate cagaar ka keenayso. Tusaalaha 8.9 ayaad ka arkaysaa getEndDate oo sidii la rabay loo qoray.

```

package nimi.course;
2
import java.util.Calendar;
import java.util.Date;
4 import java.util.GregorianCalendar;
6
public class CourseSession {
8
    ...
10
    public Date getEndDate() {
12        GregorianCalendar calendar = new GregorianCalendar();
        calendar.clear();
14        calendar.setTime(this.startDate);
        int numberOfDays = 16 * 7;
16        calendar.add(Calendar.DAY_OF_YEAR, numberOfDays);
        return calendar.getTime();
18    }
}

```

Tusaale 8.9: Ficilka getEndDate ee CourseSession oo sax loo qoray

Tusaalaha 8.9 wuxu nooca CourseSession kusoo kordhiyey ficilka getEndDate iyo laymanka 3 iyo 5. Imika kaxee CourseSessionTest si aad u aragto in dhammaan tijaabooyinku cagaar yihiin. Intaynaan hore u socon, waxan rabaa inaynu markii ugu horaysay adeegsano talaabada ugu dambaysa ee nidaamka TDD oo refactor. Illaa hadda waxaynu ku wareegaynay labada talaabo ee hore oo kala Test iyo Code (Marka hore qor tijaabada kadibna qor borogaraamka si tijaabadaasi cagaar u noqoto). Ujeedada talaabada refactor waa inay borogaraamka fahamkiisu iyo design-kiisaba wax ka badasho oo tusaale ahaan fududayso, iyada oo aan tijaabooyinku jabin ama casaan keenin. Aan ka socdo. Inta badan ma fiicna in borogaraam ay ka muuqdaan nambarro aan la garanaynin waxay ka dhiganyihiin. Nambaradaas waxa loo yaqaan *Magic numbers*. Haddii aad eegto laynka 15 ee tusaalaha 8.9, waxad ka arkaysaa laba tiro oo kala ah 16 iyo 7. 16-ku waxay ka dhigantahay dhererka fadhiga, 7-duna waxay ka dhigantahay inta maalmood ee usbuucii ku jirta. Waxa loo baahanyahay inaynu muujino arintaas. Haddaba badal ficilka getEndDate ee CourseSession oo u eekaysii tusaalaha 8.10.

```

1 package nimi.course;

3 import java.util.Calendar;
  import java.util.Date;
5 import java.util.GregorianCalendar;

7 public class CourseSession {

9     public static final int SESSION_LENGTH = 16; // in weeks
      ...

11     public Date getEndDate() {
12         GregorianCalendar calendar = new GregorianCalendar();
13         calendar.clear();
14         calendar.setTime(this.startDate);
15         int daysInWeek = 7;
16         int numberOfDays = SESSION_LENGTH * daysInWeek;
17         calendar.add(Calendar.DAY_OF_YEAR, numberOfDays);
18         return calendar.getTime();
19     }
20 }
21 }

```

Tusaale 8.10: Ficil getEndDate oo refactor lagu sameeyey

Intaas kadib kaxee tijaabada CourseSessionTest si aad u aragto in wali wax waliba shaqaynayaan oo dhammaan tijaabooyinku cagaar yihiin.

Waxa halkan inooga soo baxday mid ka mid ujeedooyinka ugu waaweyn ee loogu talagalay tijaabooyinka ama loogu baahanyahay talaabada Test ee nidaamka TDD. Marba marka ka dambaysa waxa sii kordhaaya borogaraamka source code-kiisa oo waxa sii kordhaaya tirada noocyada. Haddii mustaqbalka ama mar dambe borogaraamka waxyar meel laga badalo, waxa laga yaabaa in badalkaasi keeno in meel kale oo borogaraamka kamid ah cilad ku tijaado oo shaqayn waydo. Arintaas waxa loo yaqaan *regression*. Si haddaba looga hortago regression-ka, waa in markasta oo borogaraamka waxyar laga badalo la kexeeyo dhammaan wixii tijaabooyin la qoray, si loo ogaado in meel kale ciladi ku timid iyo in kale. Halkaas waxad ka arkaysaa mid ka mid ah faa'idooyinka ugu waaweyn ee nidaamka TDD iyo tijaabooyinku ay keenayaan.

Ardaygu markuu rabo inuu koorso ka qaybgalo ama qaato waa inuu isku qoro fadhiyada koorsadaas midkood. Sidaas awgeed waa in nooca CourseSession uu support-gareeyo ardayda. Waxaynu qoraynaa ficilkii tijaabin lahaa ardayda iyo ka qaybgalka fadhiga. Ku dar ficilka testEnrollStudents nooca CourseSession sida ka muuqata tusaalaha 8.11.

```

1 package nimi.course;

3 import static org.junit.Assert.*;

5 import java.util.Date;
  import java.util.GregorianCalendar;

7 import nimi.student.Student;

9 import org.junit.Before;
10 import org.junit.Test;

12 public class CourseSessionTest {

14     ...

16     @Test
17     public void testEnrollStudents() throws Exception {

```

```

19 assertEquals(0, this.courseSession.getNumberOfStudents());
   Student s1 = new Student(123, "Ahmed");
21 this.courseSession.enroll(s1);
   assertEquals(1, this.courseSession.getNumberOfStudents());
23 Student s2 = new Student(456, "Anisa");
   this.courseSession.enroll(s2);
25 assertEquals(2, this.courseSession.getNumberOfStudents());
   assertEquals(2, this.courseSession.getAllStudents().size());
27 }
}

```

Tusaale 8.11: Ficilka testEnrollStudents oo lagu daray nooca CourseSessionTest

Haddii aan sharaxaad gaaban ka bixiyo **8.11**:

1. Laynka 19 waxaynu wacaynaa ficilka `getNumberOfStudents` oo ah ficil cusub kaas oo aynu rabno inuu soo celiyo tirada ardayda ee ku jirta fadhigan, maadaama aanay fadhiga `courseSession` wax arday ah is qorin, waa inuu ficilkaasi inoosoo celiyo eber tirada ardayda
2. Kadib waxaynu samaynaynaa arday cusub `s1`, kadibna waxaynu qoraynaa fadhiga innaga oo adeegsanaynaa ficilka cusub ee `enroll` laynka 21-aad
3. Kadib, waxaynu hubinaynaa in tirada ardaydu tahay kow
4. Laymanka 23-24 waxaynu fadhiga qoraynaa arday cusub `s2`
5. Kadibna waxaynu hubinaaya in tirada ardaydu tahay laba
6. Ugu dambayntii laynka 26 waxaynu adeegsanaynaa ficilka cusub ee `getAllStudents`. Qiimaha uu ficilka `getAllStudents` soo celinayaa waa `List` arday ah, kadib waxaynu isticmaalaynaa ficilka `size` ee si aynu u helo tirada ardayda ee kujira, taas oo la rabo inay ahaato laba.

Wuxu Eclipse ka cabanayaa saddexda ficil oo wuxu ku tusayaa cilado u eeg:

The method `getNumberOfStudents()` is undefined ...

The method `enroll(Student)` is undefined ...

The method `getAllStudents()` is undefined ...

Mar kaliya ayaan saddexdaas ficil ku darayaa nooca `CourseSession` si ay tijaabadu u guulaysato, waxaanad ka arkaysaa tusaalaha **8.12**.

```

package nimi.course;
2
import java.util.ArrayList;
4 import java.util.Calendar;
import java.util.Date;
6 import java.util.GregorianCalendar;

8 import nimi.student.Student;

10 public class CourseSession {

12     public static final int SESSION_LENGTH = 16; // in weeks
    private Date startDate;
14     private Course course;
    private int numberOfCredits;
16     private ArrayList<Student> students;

18     public CourseSession(Course course, Date startDate,
        int numberOfCredits) {
20         this.course = course;
        this.startDate = startDate;
22         this.numberOfCredits = numberOfCredits;
        this.students = new ArrayList<Student>();
24     }

26     public Course getCourse() {
        return this.course;
28     }

30     public int getNumberOfCredits() {
        return this.numberOfCredits;
32     }
}

```

```

34 public Date getStartDate() {
35     return this.startDate;
36 }
37
38 public void setNumberOfCredits(int credits) {
39     this.numberOfCredits = credits;
40 }
41
42 public Date getEndDate() {
43     GregorianCalendar calendar = new GregorianCalendar();
44     calendar.clear();
45     calendar.setTime(this.startDate);
46     int daysInWeek = 7;
47     int numberOfDays = SESSION_LENGTH * daysInWeek;
48     calendar.add(Calendar.DAY_OF_YEAR, numberOfDays);
49     return calendar.getTime();
50 }
51
52 public int getNumberOfStudents() {
53     return this.students.size();
54 }
55
56 public void enroll(Student student) {
57     this.students.add(student);
58 }
59
60 public ArrayList<Student> getAllStudents() {
61     return this.students;
62 }
63
64 public void setCourse(Course course) {
65     this.course = course;
66 }
67
68 public void setStartDate(Date startDate) {
69     this.startDate = startDate;
70 }
71 }

```

Tusaale 8.12: Ficilada sare oo lagu daray CourseSession (dhammaystiran)

Intaas kadib kaxee tijaabada CourseSessionTest si aad u aragto in dhammaan ay cagaar bixinayso. Kudar nooca CourseSessionTest, nooca AllTests oo loo yaqaano *Test Suite*, maadaama uu uruurinaayo noocyada tijaabooyinka ah oo dhan, sidaad ka arkayso 8.13.

```

1 package nimi;
2
3 import org.junit.runner.RunWith;
4 import org.junit.runners.Suite;
5 import org.junit.runners.Suite.SuiteClasses;
6
7 @RunWith(Suite.class)
8 @SuiteClasses({
9     nimi.person.PersonTest.class, // PersonTest
10    nimi.student.StudentTest.class, // StudentTest
11    nimi.teacher.TeacherTest.class, // TeacherTest
12    nimi.course.CourseTest.class, // CourseTest
13    nimi.course.CourseSessionTest.class // CourseSessionTest
14 })
15 public class AllTests {}

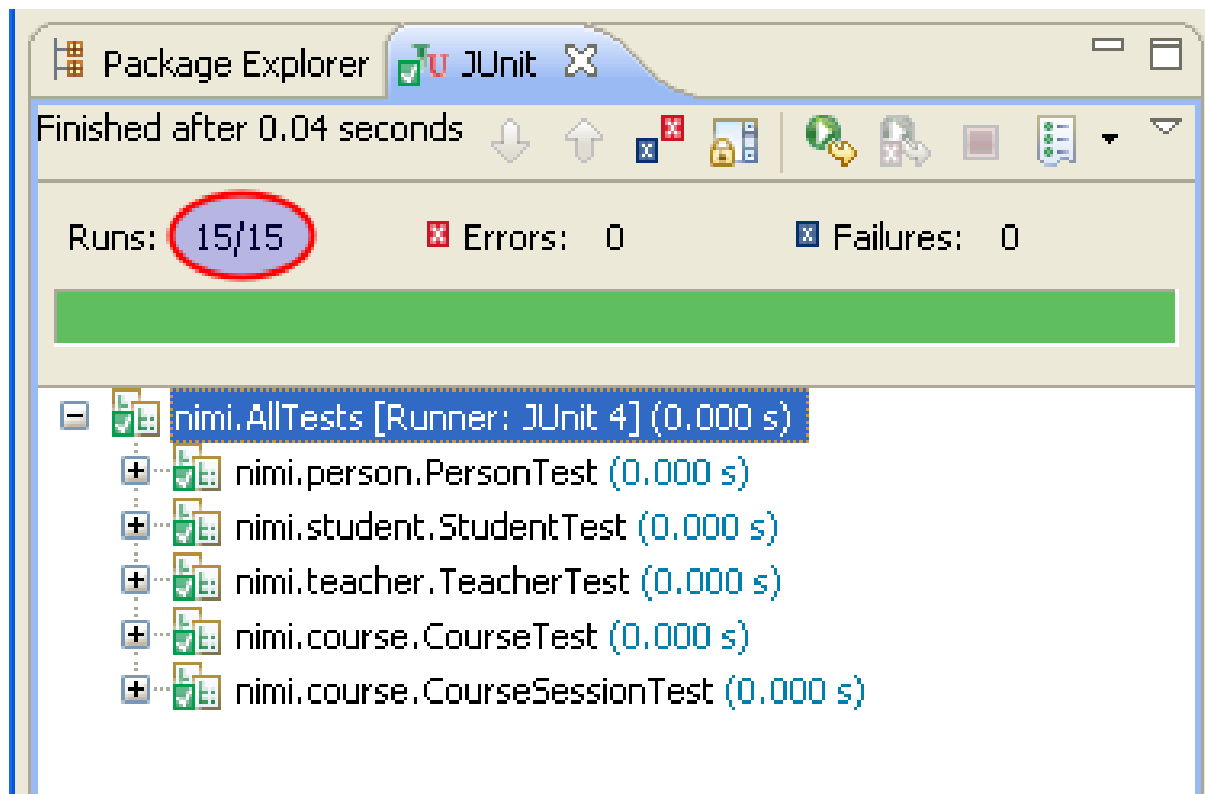
```

Tusaale 8.13: Nooca AllTests oo lagu kordhiyey nooca CourseSessionTest

Kaxee nooca AllTests si aad u aragto in illaa hadda wixii tijaabo aynu qornay dhammaantood cagaar yihiin, taas oo ka dhigan in borogaraamku sidii aynu ka rabnay u shaqaynaayo, sidaad ka arkayso sawirka 8.2.

8.1 Java SE API Documentation

Luuqadda Java waxa loo qaybiyaa saddex qaybood. kumbuyuutarada caadiga inta badan waxa lagu isticmaalaa *Java Platform Standard Edition* oo loo soo gaabiyo *Java SE*. Kumbuyuutarada server-ka ah waxa lagu isticmaalaa *Java Platform Enterprise Edition* oo loo soo gaabiyo *Java EE*. Java EE waa Java SE oo lagu kordhiyey waxyaalo mishiinada server-ka u gaar ah. Luuqadda Java waxa kale oo loo isticmaali karaa in borogaraamo loo sameeyo telefoonada iyo tablet-yada, markaas waxa la adeegsanayaa *Java Platform*



Sawirka 8.2: Dhammaan tijaabooyinka aynu illaa hadda haysano oo dhammaantood cagaar ah

Micro Edition oo loo soo gaabiyo *Java ME*. Java ME waa Java SE oo lasii fududeeyey. Sidaad aragtayba borogaraamkaagu wuxu ka koobanyahay noocyo aad adigu qortay iyo noocyo aad ka adeegsanayso luuqadda Java. Qaybahan Java waxay ku kala duwanyihiin noocyada ay caloosha ku haystaan. Sawirka 8.3 waxad ka arkaysaa dhammaan noocyada Java SE (gaar ahaan soosaarkeeda 6-aad oo aynu buugan ku isticmaalayno). Waxaanad ka haysaa adarayska <http://docs.oracle.com/javase/6/docs/api/>.

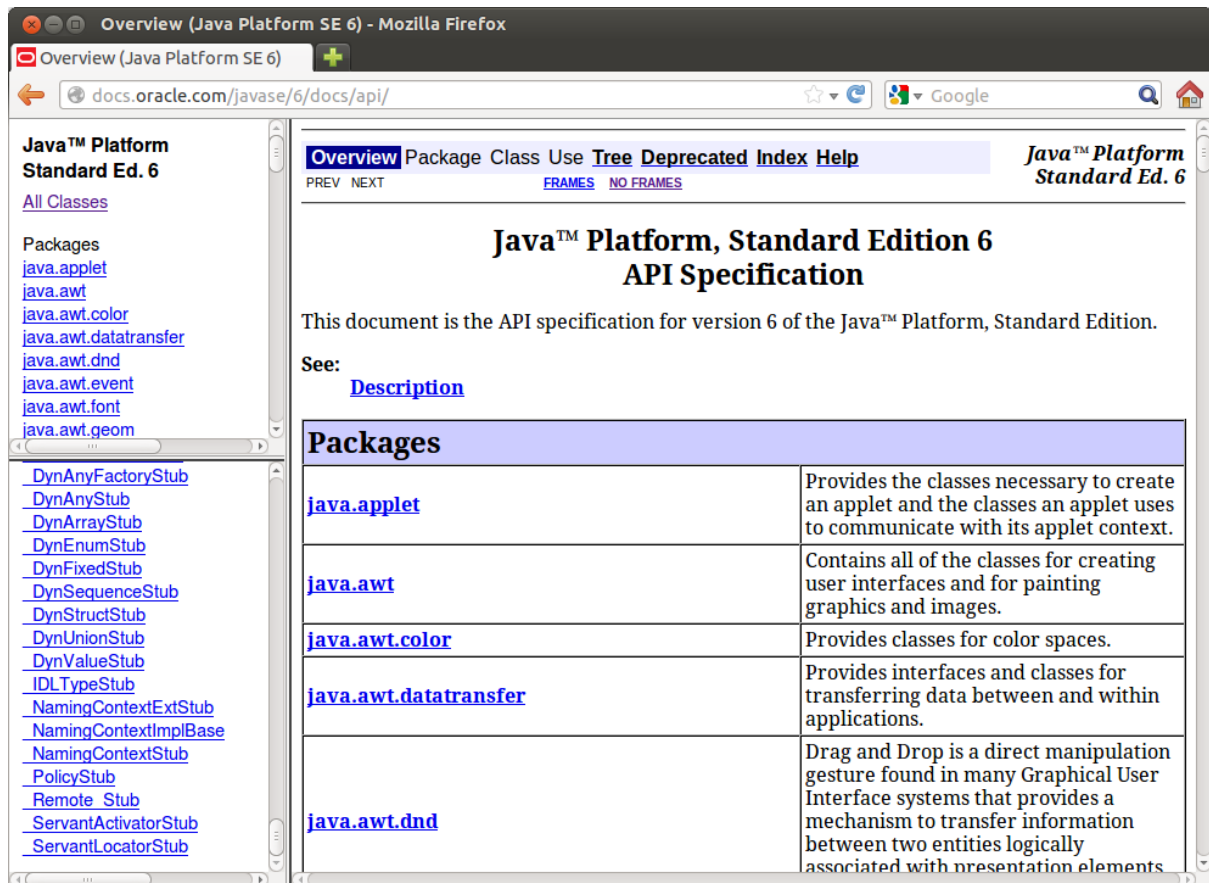
API Specification-ka waxad ka haysaa dhammaan package-yada iyo class-yada luuqadda Java iyo sharaxaad ku saabsan class-walba waxa loogu talagalay. Laylis ahaan waxad ka raadisaa noocyada String, Date, Calendar, GregorianCalendar iyo ArrayList. Waa maxay package-yada noocyadaasi?

8.2 Interfaces

Intaynaan hore u socon, waxa muhiim ah inaynu wax ka sheegno *interfaces*-ka iyo waxay yihiin. Waxad soo baratay waxyaalo badan illaa hadda. Waxyaalaha aad soo baratay illaa hadda waxa kamid ah in walxuhu ama objects-ku ay ku wada xidhiidhaan ficilada ay leeyihiin. Ficiladaas oo ka imayaana walaxda nooceeda iyo noocyada laga soo dhiraandhiriyeey. Tusaale ahaan, haddii aad haysato walax nooceedu yahay Student, walaxdaasi waxay leedahay dhammaan ficilada noocyada Student, Person iyo Object (Object waa nooca dhammaan noocyada Java laga soo dhiraandhiriyeey, sidaan hore u soo sheegnayba). Haddaba, ficiladaas walaxdu leedahay ayaa loo yaqaan *interface*. Ereyga interface runtii macno badan ayuu leeyahay, oo tusaale ahaan badhannada moobilka, radio-ga, tv-ga, saacadda, iwm dhammaantood waxa la odhan karaa waa interface-ka u dhaxaysa shaygaas iyo qofka isticmaalaaya. Ficilada walxaha waxad u sawiran kartaa inay yihiin interface-ka u dhaxaysa walaxdaas iyo walxaha kale. Waxa dhici karta in laba walxood oo kala nooc ah inay yeeshaan ficillo isku mid ah, haddii ay taasi dhacdo inta badan ficiladaas waxa lagu uruuriyaa interface gooni ah, taas oo sumad midaysa u noqota labadaas walxood.

Tusaale ahaan dhammaan walxaha magaca leh sida dadka, gaadiidka, magaalooyinka, xayawaanka, buugaagta, iwm, waxay ka midaysanyihiin inay magac leeyihiin. Sidaas awgeed waxaynu samaynkarnaa interface ka kooban hal ficil kaas oo ah kii magaca lagu waydiin lahaa sida ka muuqata tusaalaha 8.14. Tusaalahaas waxad ka arkaysaa in nooca iyo interface-ku isku dhawyihiin dhanka qoraalka. Halka noocu caloosha ku haysto sifooyin iyo ficillo, interface-ku waxay ka koobantahay kaliya ficillo aan calool lahayn.

```
1 package nimi.person;
3 public interface Named {
```



Sawirka 8.3: Java™ Platform, Standard Edition 6 API Specification, oo aad ka helayso sharaxaad ku saabsan dhammaan wixii noocyo ah ee ku jira

```

5 public String getName();
}

```

Tusaale 8.14: Tusaale interface

Innagu waxaynu haysanaa nooca Person oo matalaaya shakhsiga ama qofka. Shakhsi kastaaba wuxuu leeyahay magac. Sidee ayaynu haddaba u adeegsan karnaa interface-ka Named (MagacLeh) ee tusaalaha 8.14? Marka hore qor interface-ka 8.14, adiga oo raacaya talaabada NewClass, una buuxbuuxi sidan: Source folder=nimi/src, Package=nimi.person, Name=Named. Kadib u eekaysii sida 8.14. Intaas kadib, maadaama nooca Person la waydiinkaro magaciisa badal oo kudar [implements Named] magaciisa kadib, sida ka muuqata 8.15.

```

2 package nimi.person;
4 public class Person implements Named {
    ...
}

```

Tusaale 8.15: Interface implementation. Nooca Person oo implement-garaynaaya interface-ka Named

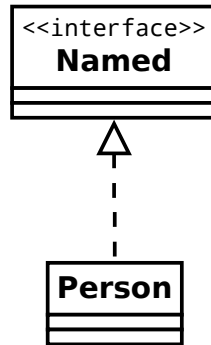
Tusaalaha 8.15 marka lagu muujinaayo UML waxa loo sawirayaa sida sawirka 8.4.

Marka nooca Person laga hadlaayo waxa la yidhaahdaa "wuxuu implement-garaynaa interface-ka Named", maadaama uu implement-garaynaayo interface-kaas. Halkaas waxad ka arkaysaa in interface-ka kaligeed aanay waxba taraynin, haddii aanu jirin nooc ama class implement-garaynaaya. Sidee loo adeegsan karaa interface-ka borogaraamka dhexdiisa? Waxad ka arkaysaa borogaraamka yar ee 8.16

```

1 package nimi;
3 import nimi.person.Named;
import nimi.person.Person;
5 public class IsbarJava {
7     public static void main(String[] args) {
9         Named magacLeh = new Person(2, "Cigaal Shiidaad");
    }
}

```



Sawirka 8.4: Nooca Person oo implement-garaynaaya interface Named

```

11     showName(magacLeh);
12 }
13 private static void showName(Named named) {
14     System.out.println("Magacu waa: " + named.getName());
15 }
16 }

```

Tusaale 8.16: Isticmaalka interface-ka borogaraamka dhexdiisa

Markaad kaxayso borogaraamka 8.16 adiga oo raacaya talaabada `RunJava`, waa inay kuusoo baxdo:

Magacu waa: Cigaal Shiidaad

Aan in yar oo sharaxaad ah ka bixiyo borogaraamka 8.16 intaynaan hore u socon.

1. Laynka 9-aad waxaynu samaynaynaa walax cusub oo Person ah, taas oo aynu siinayno magac iyo ID. Laakiin doorsoomaha aynu ku kaydinaayo ee magacLeh, noociisu waa interface-ka nooceda. Awal hore markasta oo aynu walax Person ah samayno waxaynu ku kaydinaaynay doorsoome noocedu yahay Person, laakiin sidaad halkan ka arkayso doorsoomaha noociisu wuxu noqonkaraa interface-ka magaceeda
2. Laynka 10-aad waxaynu wacaynaa ficil showName, kaas oo aynu u gudbinayno doorsoomaha magacLeh. Ficil showName calooshiisa gaarahaan laynka 14-aad, waxaynu isicmaalaynaa ficilkii interface-ka ee getName, maadaama walax kasta oo noocedu implement-garaynaayo interface-kaas uu leeyahay ficilkaas.

8.3 Collections

Nooca `CourseSession` ee 8.12 gaar ahaan laymanka 16 iyo 23 waxaynu isticmaalay nooca `ArrayList` oo kamid luuqadda Java. Haddaba muxuu yahay nooca `ArrayList`, muxuuse qabtaa? Ma aha markii ugu horaysay ee aynu adeegsano nooca `ArrayList`. Markii ugu horaysay waxaynu u adeegsanay nooca `Student` si aynu ugu kaydino dhibcaha ardayga. Imikana waxaynu u adeegsanay in aynu ku kaydino ardayda isqortay fadhiga ama `CourseSession`-ka. Halkaas waxad ka arkaysaa in ujeedada `ArrayList` tahay in lagu uruuriyo guruub walxo ah oo isku nooc ah. Noocyada sidan oo kale ah luuqadda Java waxa loo yaqaan *Collections*. Nooca `ArrayList` waa inaad u sheegto walxaha aad ku kaydinaayo noocooda, adiga oo dhexgalinaaya calaamadaha `<...>` sida `ArrayList<Student>` ama `ArrayList<Grade>` ama `ArrayList<Person>`, ... Haddii aad tusaale ahaan u sheegto inaad ku kaydinaayo walxaha noocoodu yahay Person, waxad ku kaydinkartaa wixii Person ah iyo wixii walxo ah ee noocoodu yahay subclass-yada ama nooc hoosaadka Person, sida `Student` iyo `Teacher`, laakiin kuma kaydintid kartid walax noocedu yahay tusaale ahaan Door.

Waxa jirta interface layidhaahdo `List` oo nooca `ArrayList` iyo qaar kale ay implement-garaynaayan. Sidaas awgeed inta badan doorsoomeyaasha iyo sifooyinka noocooda waxa laga dhigaa interface-kaas. Markaas nooca `CourseSession` waxyar ka badal sida ka muuqata tusaalaha 8.17, laymanka 7, 17 iyo 21. Intaas kadib, kaxee tijaabooyinka `AllTests`, si aad u hubiso in borogaraamku u shaqaynaayo wali sidii la rabay badalka kadib.

```

package nimi.course;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List; // laynkan ku dar

import nimi.student.Student;

public class CourseSession {

    public static final int SESSION_LENGTH = 16; // in weeks
    private Date startDate;
    private Course course;
    private int numberOfCredits;
    private List<Student> students; // Laynkan badal

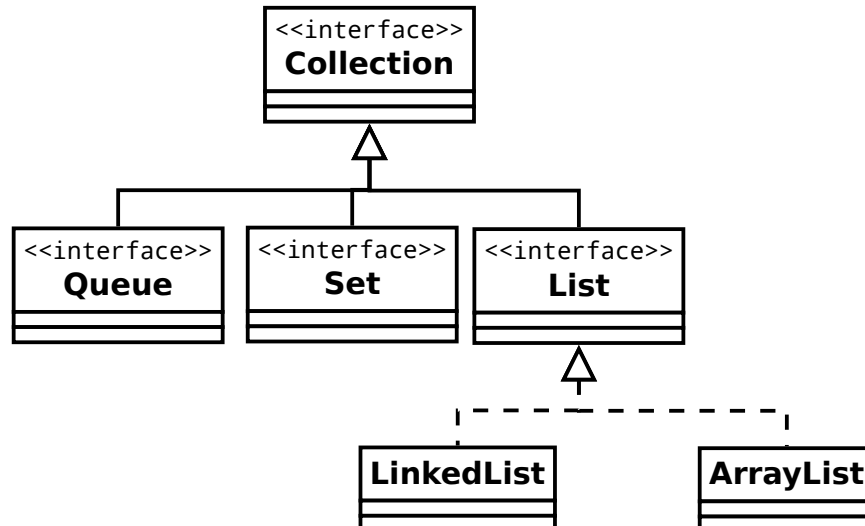
    ... // wixii halkan ku qarsoon looma baahna inaad badasho

    public List<Student> getAllStudents() {
        return this.students;
    }
}

```

Tusaale 8.17: Isticmaalka interface-ka List ee nooca CourseSession

Sawirka 8.5 waxad ka arkaysaa in sida noocyada looga soo dhiraandhirinkaro noocyo kale in interface-kana laga soo dhiraandhirinkaro interface kale. Waxa kale oo aad sawirka ka arkaysaa in nooca ArrayList uu implement-garaynaayo interface-ka List iyo interface-keeda sare ee Collection.



Sawirka 8.5: Interface-yada collections-ka ee Java. Waxad sawirkan ka arkaysaa in noocyada ArrayList iyo LinkedList, labadooduba yihiin List, maadaama ay implement-garaynayaan interface-kaas.

8.4 Modifier-ka final

Ereyga u gaarka ah luuqadda Java ee **final** waxaynu ku aragnay cutubkan. Ereygaasi wuxu u baahanyahay faahfaahin dheeraad ah intaynaan hore u socon. Af-soomaaliga ereygaas waxaynu odhan karnaa

kama dambays. Ereygaas waxa lagu isticmaalaa siyaabo kala duwan. Waxan halkan ku sharaxayaa siyaabooyinku ugu badan ama ugu isticmaalka badan. Waana kuwan:

1. Marka ugu horaysa ereyga **final** waxa laga horaysiin karaa magaca nooca, sidaan kusoo sheegay qayb hoosaadka noocada ee 5.2. Haddii noocu uu yahay **final**, wuxu ka dhiganyahay noocaas lagamasoo dhiraandhirinkaro noocyo cusub. Illaa hadda innamuu soo marin nooc **final** ah. **final**-ka noocan oo kale ah wuxu la xidhiidhaa inheritance-ka ama takhasuska
2. Marka xigta waxa jiri kara ficil **final** ah, taas oo ka dhigan inaan ficilkaas ficil la magac iyo parameters ah aanay yeelankarin noocyada noocan laga soo dhiraandhiriyo. Kan laftiisu wuxu la xidhiidhaa takhasuska
3. Istimmaalka ugu dambeeya ee **final** oo runtii la odhan karo waa ka ugu muhiimsan waa marka loo isticmaalo sifooyinka. Waxad tusaale u soo qaadataa sifada `SESSION_LENGTH` ee nooca `CourseSession`. Sifooyinka noocaas oo kale ah ee qiimahooda isla meeshaas lagu siiyo waxa loo yaqaan *constants*. Af-soomaaliga waxaynu u isticmaali karnaa ismadooriyeyaal. Sifooyinka kale qiimahooda waxad ku badali kartaa tusaale ahaan ficilka dhexdiisa, laakiin sifooyinka **final**-ka qiimehooda lama badali karo. Sidaas awgeed waxa loo isticmaalaa inta badan qiimeyaasha aan is badalin. Marka aad rabto inaad isticmaasho qiime **final** ah waa inaad ka horaysaysiiyo inta badan magaca nooca tusaale ahaan `CourseSession.SESSION_LENGTH`, waa markaad qiimahaas ku isticmaalayso nooc kale dhexdiisa.

8.5 Gunaanad

Wuxu ahaa cutubkii ugu dambeeyay ee buugan qaybtiisa labaad. Cutubkan waxyaala ugu muhiimsan ee aynu kaga hadalay waxa ka mid ah interface-ka, Collections-ka iyo documentation-ka luuqadda Java.

Intaynaan ka gudbin waxaad nooca `CourseSession` ku dartaa ficilada setters-ka ah ee tusaalaha 8.18, kadibna kaxee tijaabooyinka oo dhan.

```

1 public class CourseSession {
2     ...
3
4     public void setCourse(Course course) {
5         this.course = course;
6     }
7
8     public void setStartDate(Date startDate) {
9         this.startDate = startDate;
10    }
11    ...
12 }

```

Tusaale 8.18: Ficilada setters-ka oo lagu diray nooca `Course`

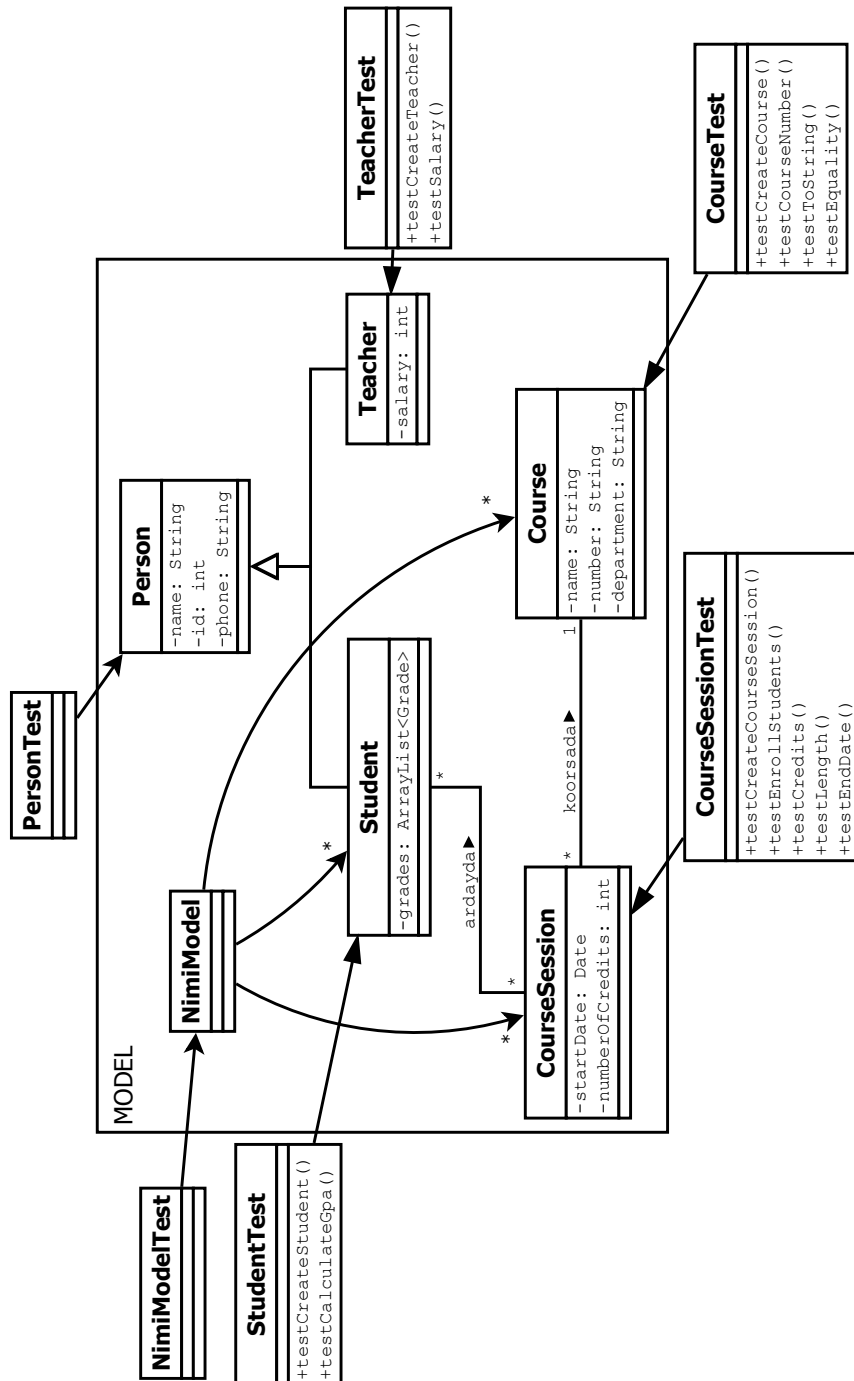
Cutub 9

Model-ka Borogaraamka

Illaa hadda waxaynu ka soo hadalay qaybihii ugu muhiimsanaa ee uu ka koobnaa borogaraamka aynu buuggan ku dhisayno ee **nimi**. Arimaha kala duwan ee aynu kasoo hadalay waxa ka mid ah qofka ama shakhsiga, ardayda, naxwaha luuqadda Java, macalimiinta, koorsooyinka iyo fadhiyada. Qayb walba waxaynu ku soo qaadanay cutub iyada u gaar ah oo aynu ku falanqaynay. Haddaba, waxa lagama maarmaan ah in qaybahaas kala duwan laysu geeyo, maadaama ay qayb ka yihiin sawirka wayn oo ah borogaraamka oo dhammaystiran. Qayb walba sawirkeeda ama UML-keeda gooni ayaan u soo qaatay, laakiin waxa loo baahanyahay hal sawir oo dhammaystiran oo laga arki karo dhammaan qaybaha kala duwan ee uu borogaraamku ka kooban yahay iyo noocyada ku jira borogaraamkan, waxaanad ka arkaysaa sawirka **9.1**. Sidaad sawirkaas ka arkaysaba waxaynu hal sawir iskugu keennay dhammaan noocyadii borogaraamka ku jiray. Sawirka noocan oo kale ah waxa loo yaqaan **model**-ka borogaraamka. Aan si kale u dhigo, model-ka borogaraamka waxa loo yaqaan dhammaan noocyada borogaraamkaas ku jira ee macluumaadka kaydinaaya iyo xidhiidhka ka dhexeeya (noocyada wajigu ee soo socda kuma jiraan model-ka).

Sidaad sawirka **9.1** ka arkaysaba, waxaan xero ku wada xareeyey dhammaan noocyada model-ka borogaraamku uu ka koobnaa. Waxan banaanka u saaray dhammaan noocyadii tijaabooyinka ahaa, sababtoo ah iyaga ujeedadoodu waa inay hubiyaan in borogaraamku sax u shaqaynaayo oo wixii loogu talagalay qabanaayo iyo in kale. Waxa kale oo aad sawirka ka arkaysaa oo kusoo biiray laba nooc oo cusub oo kala ah **NimiModel** iyo tijaabiyehiisa **NimiModelTest**. Waxaynu guda galaynaa ka hadalkooda iyo qoristooda. Nooca **NimiModel** wuxu matalayaa model-ka borogaraamka aynu dhisayno, oo waa noocii isu keenaayey dhammaan noocyada kale ee borogaraamka ku jira sida **Person**, **Student**, **Teacher**, iwm.

Waxaynu ku bilaabaynaa qorista nooca **NimiModelTest**. Waxad samaysaa nooc cusub adiga oo raacayaa talaabadii **NewTest**, waxaanad u buuxbuuxisaa sidan: **Source folder=nimi/tst**, **Package=nimi**, **Name=NimiModelTest**. Ku bilaw tusaalaha **9.1** nooca **NimiModelTest**.



Sawirka 9.1: Model-ka borogaraamka nimi

```

1 package nimi;
2
3 import static org.junit.Assert.*;
4
5 import java.util.List;
6
7 import nimi.student.Student;
8
9 import org.junit.*;
10
11 public class NimiModelTest {
12
13     private NimiModel model;
14
15     @Before
16     public void setUp() throws Exception {
17         // create nimi model
18         model = new NimiModel();
19     }
20
21     @Test
22     public void testStudents() throws Exception {
23         assertEquals(model.getNumberOfStudents(), 0);
24         Student s1 = new Student(1, "Muuse Jaamac");
25         model.addStudent(s1);
26         assertEquals(model.getNumberOfStudents(), 1);
27         List<Student> students = model.getStudents();
28         assertEquals(students.get(0).getName(), "Muuse Jaamac");
29     }
30 }

```

Tusaale 9.1: Bilawga nooca NimiModelTest, oo tijaabinaaya nooca NimiModel oo soo socda

Aan sharaxaad yar ka bixiyo tusaalaha 9.1, intaynaan hore u socon.

1. Ficilka setUp ee laymanka 16-19 ee sumadda @Before leh wuu innagu cusub yahay. Ficilkan wuxu JUnit kaxaynayaa ficil kasta oo tijaabo ah ka hor. Markaas, imika wuxu kaxaynayaa intaanu kaxaynin ficilka tijaabada ah ee testStudents. Ficilka setUp wuxu samaynayaa walax cusub oo nooceedu yahay NimiModel, kadibna wuxu ku kaydinayaa sifada model ee noocan NimiModelTest ee laynka 13-aad. Xaqiiqadii intaad doonto oo ficil oo sumadda @Before leh ayaad nooca u samayn kartaa. Haddii ay taasi dhacdo JUnit dhammaan ficilada @Before leh wuxu kaxaynayaa ficil kasta oo tijaabo ah ka hor
2. Ficilka testStudents ee laymanka 22-29 wuxu tijaabinayaa ardayda model-ka ku jirta. Laynka 23-aad waxaynu isticmaalaynaa ficilka getNumberOfStudents ee walaxda model, kaas oo u dhigma ficilka fadhiga ee la magaca ah, shaqadiisuna tahay inuu inna siiyo tirada ardayda ee ku jirta model-ka, kadib waxaynu hubinaynaa inay tahay eber, maadaama aynaan wali wax arday ah ku darin model-ka
3. Labada layn ee ka hooseeya waxaynu samaynaynaa arday cusub, kaas oo aynu ku darayno model innaga oo adeegsanayna ficil addStudent. Kadib laynka saddexaad waxaynu hubinaynaa in tirada ardaydu tahay mid, maadaama aynu imika hal arday ku darnay
4. Laymanka 27 iyo 28 waxaynu model-ka waydiisanaynaa dhammaan ardayda innaga oo adeegsanayna ficilka getStudents. Kadib waxaynu hubinaynaa in ardayga booska ugu horeeya oo nambarkiisu yahay 0 kujira uu yahay ardaygii aynu ku darnay model.

Haddii aad isku daydo inaad kaxayso tusaalaha 9.1, waxa kaa hor imanaysa cilad, maadaama aanu jirin wali nooca NimiModel. Waxad qortaa nooca NimiModel, waxaad u eekaysiisaa tusaalaha 9.2 ee hoose.

```

1 package nimi;
2
3 import java.util.List;
4
5 import nimi.student.Student;
6
7 public class NimiModel {
8
9     public int getNumberOfStudents() {
10         return 0;
11     }
12 }

```

```

12 public void addStudent(Student student) {
14 }
16 public List<Student> getStudents() {
18     return null;
19 }
20 }

```

Tusaale 9.2: Bilawga nooca NimiModel

Intaas kadib nooca NimiModelTest casaankii iyo ciladihii way ka tagayaan, laakiin markaad kaxayso ficilka testStudents wuu guuldaraysanayaa, maadaama aanu nooca NimiModel sidii laga sugaayey aanu u shaqaynaynin. Intaas kadib waxaynu wax ka badalaynaa nooca NimiModel illaa inta nooca NimiModelTest uu gacaar ka bixiyaano oo inna tusaayo inuu sax u shaqaynaayo. Badal oo u eekaysii nooca NimiModel tusaalaha 9.3 ee hoose.

```

1 package nimi;
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import nimi.student.Student;
7
8 public class NimiModel {
9     private ArrayList<Student> students;
10
11     public NimiModel() {
12         this.students = new ArrayList<Student>();
13     }
14
15     public int getNumberOfStudents() {
16         return this.students.size();
17     }
18
19     public void addStudent(Student student) {
20         this.students.add(student);
21     }
22
23     public List<Student> getStudents() {
24         return this.students;
25     }
26 }

```

Tusaale 9.3: Nooca NimiModel oo la implement-gareeyey ficiladiisa getNumberOfStudents, addStudent iyo getStudents. Sidoo kale lagu daray ficilka dhisaha ama constructor-ka

Intaas kadib tijaabada testStudents ee nooca NimiModelTest waa inay gacaar bixiso marka la kaxeeyo. Waxa inoo xigtay koorsooyinkii. Kudar ficilka testCourses nooca NimiModelTest sida ka muuqata tusaalaha 9.4 ee hoose.

```

1 package nimi;
2
3 import static org.junit.Assert.*;
4
5 import java.util.List;
6
7 import nimi.course.Course;
8 import nimi.student.Student;
9
10 import org.junit.*;
11
12 public class NimiModelTest {
13
14     ...
15
16     @Test
17     public void testCourses() throws Exception {
18         assertEquals(model.getNumberOfCourses(), 0);
19         Course c1 = new Course("A", "ABC-01", "ABC");
20         model.addCourse(c1);
21         assertEquals(model.getNumberOfCourses(), 1);
22         List<Course> courses = model.getCourses();
23         assertEquals(courses.get(0).getName(), "A");
24     }
25 }

```

Tusaale 9.4: Ficilka testCourses ee nooca NimiModelTest

Ficilka testCourses sharaxaad badan inoogama baahna, sababtoo ah wuxu la midyahay ficilka sare ee testStudents, kaliya ardaydii ayaa isu badashay koorsooyin. Tusaalaha 9.4 waxad ka arkaysaa in loo baahanyahay saddex ficil oo cusub oo kala ah getNumberOfCourses, addCourse iyo getCourses kusoo kordhayaan nooca NimiModel. Ku dar ficiladaas cusub nooca NimiModel si tijaabada testCourses u pass-garayso. Hoose 9.5 ayaad ka arkaysaa nooca NimiModel oo lagu daray ficiladaas.

```

1 package nimi;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import nimi.course.Course;
7 import nimi.student.Student;
8
9 public class NimiModel {
10     private ArrayList<Student> students;
11     private ArrayList<Course> courses;
12
13     public NimiModel() {
14         this.students = new ArrayList<Student>();
15         this.courses = new ArrayList<Course>();
16     }
17
18     public int getNumberOfStudents() {
19         return this.students.size();
20     }
21
22     public void addStudent(Student student) {
23         this.students.add(student);
24     }
25
26     public List<Student> getStudents() {
27         return this.students;
28     }
29
30     public int getNumberOfCourses() {
31         return this.courses.size();
32     }
33
34     public void addCourse(Course course) {
35         this.courses.add(course);
36     }
37
38     public List<Course> getCourses() {
39         return this.courses;
40     }
41 }

```

Tusaale 9.5: Nooca NimiModel oo lagu daray ficilada sare

Intaas kadib, haddii aad kaxayso NimiModelTest waa in labada tijaabo ay guuleystaan oo cagaar keenaan. Taas oo ku tusaysa in nooca NimiModel sidii aynu rabnay uu u shaqaynaayo. Waxa inoo dhiman fadhiyadii. Kudar tijaabada testSessions nooca NimiModelTest sida tusaalaha 9.6 ka muuqata.

```

1 package nimi;
2
3 import static org.junit.Assert.*;
4
5 import java.util.Date;
6 import java.util.GregorianCalendar;
7 import java.util.List;
8
9 import nimi.course.Course;
10 import nimi.course.CourseSession;
11 import nimi.student.Student;
12
13 import org.junit.*;
14
15 public class NimiModelTest {
16     ...
17 }

```

```

19  @Test
    public void testSessions() throws Exception {
21      assertEquals(model.getNumberOfSessions(), 0);
      Course c1 = new Course("A", "ABC-01", "ABC");
23      GregorianCalendar calendar = new GregorianCalendar();
      Date now = calendar.getTime();
25      CourseSession s1 = new CourseSession(c1, now, 3);
      model.addSession(s1);
27      assertEquals(model.getNumberOfSessions(), 1);
      List<CourseSession> sessions = model.getSessions();
29      assertEquals(sessions.get(0).getCourse().getName(), "A");
      assertEquals(sessions.get(0).getNumberOfCredits(), 3);
31  }
}

```

Tusaale 9.6: Ficilka testSessions oo lagu daray nooca NimiModelTest

Ficilka testSessions ee 9.6 wuxu la mid yahay ficiladii hore ee testCourses iyo testStudents, kaliya laynka 24-aad waxaynu nooca GregorianCalendar waydiisanaynaa taariikhda imika oo noqonaysa maal-inta fadhigu bilaabmaayo. Si tijaabada testSessions u pass-garayso ku dar ficilada getNumberOfSessions, addSession iyo getSessions nooca NimiModel sida ka muuqata tusaalaha 9.7.

```

package nimi;
2
import java.util.ArrayList;
4 import java.util.List;

6 import nimi.course.Course;
import nimi.course.CourseSession;
8 import nimi.student.Student;

10 public class NimiModel {
    ...
12     private ArrayList<CourseSession> sessions;

14     public NimiModel() {
        ...
16         this.sessions = new ArrayList<CourseSession>();
    }

    ...

20     public int getNumberOfSessions() {
22         return this.sessions.size();
    }

24     public void addSession(CourseSession session) {
26         this.sessions.add(session);
    }

28     public List<CourseSession> getSessions() {
30         return this.sessions;
    }
32 }

```

Tusaale 9.7: Nooca NimiModel oo lagu daray ficilada sare

Intaas kadib, kaxee nooca NimiModelTest si aad u aragto in saddexda tijaababa cagaar yihiin. Cutubkan intaas aynu ku joojinaynaa. Qaybta saddexaad ee buuggan ayaynu modelka wax ku kordhinaynaa, laakiin imika halkaas ayaynu ku joojinaynaa. Intaynaan laakiin ka gudbin cutubkan waxan qeexayaa qodobada ugu muhiimsan ee model-ku borogaraamka u leedahay:

1. Marka ugu horaysa model-ku waxay isu uruurinaysaa dhammaan noocyada ku jira borogaraamka, gaar ahaan kuwa khuseeya macluumaadka borogaraamka
2. Modelka waxa lagu kaydin karaa xasuusta kumbuyuutarka ama database, si marka kumbuyuutarka la damiyo aan data-du u lumin. Qodobkani waa arimaha ugu muhiimsan ee model loogu baahanyahay
3. Model-ka waxa laga samaynkaraa report-yo ama warbixin kala duwan.

9.1 Gunaanad

Si aynu u soo koobno model-ka kumaynaan darin tusaale ahaan macalimiintii. Model-ku waa meesha macluumaadka borogaraamku tusaale ahaan ardaydu, macalimiintu, koorsooyinku, iwm. ku kaydsan yihiin. Model-ka waxa lagu kaydiyaa inta badan xasuusta kumbuyuutarka ama database-ka si aanay u lumin macluumaadka borogaraamka lagu diyaariyey. Qaybta ugu dambaysa ee buuggan waxaynu model-ka ku xidhaynaa wajiga ama muuqaalka oo loo yaqaano *View*.

Cutub 10

Java IO

Waxaynu si guud halkan ugaga hadlaynaa mawduuc runtii aad muhiim u ah, laguna magacaabo **Input-Output**, oo loo soo gaabiyo *I/O* ama *IO* inta badan. Ma aha mawduucani mid u gaar ah luuqadda kumbuyuutarada ee Java. Waa qayb muhiim ah oo kamid ah kumbuyuutarada sideedaba. Runtii waa mawduuc u baahan in si dheer looga hadlo, laakiin maadaama uu buuggani yahay hordhac ama introduction, waxaynu halkan ugaga hadlaynaa si dulmar ah. Waxan isku dayayaa inaan halkan kaga jawaabo laba su'aalood oo kala ah: waa maxay IO? Sidee ayaan luuqadda Java ugu adeegsan karaa IO-da?

Waxan ku bilaabayaa inaan su'aasha hore jawaab u raadiyo. IO sideedaba waa qaybta u qaabilsan kumbuyuutarku inuu la hadlo ama macluumaad waydaarsado aduunka kale ee ka baxsan. Kumbuyuutarka waxaynu kasoo qaadaynaa labada unug ee ugu muhiimsan ee kala ah xasuusta dhexe (main memory) iyo CPU-ga. Labadaas oo la odhan karo waa maskaxda ama lafdhabarka kumbuyuutarka, wixii ka baxsan, sida xasuusta weyn ee **hard disk**-ka, jiirka, keyboard-ka, monitor-ka, web camera-da, printer-ka, modem-ka, iwm. dhammaantood waxay u baahanyihiin IO si loo adeegsado, waxaana qalabkaas loogu yeedhikaraa magaca guud ee *IO Devices*. Dhammaan qalabkaas kuwa wax laga soo akhriyaayo sida jiirka, keyboard-ka, **hard disk**-ka, iwm. waxa loo yaqaan *Input Devices*, maadaama macluumaad laga soo akhriyikaro. Qalabka wax lagu qorikaro sida mishiinka daabacaadda, shaashadda, **hard disk**-ka, iwm. waxa loo yaqaan *Output Devices*, maadaama macluumaad lagu qorikaro. Qalabka qaar sida **hard disk**, waxay noqon karaan Input Device iyo Output Device labadaba, maadaama waxna lagu qorikaro, waxna laga soo akhriyi karo. Marka kumbuyuutarku rabo inuu la hadlo kumbuyuutarada kale ee meel-kale yaalla, iyana waxay u baahantahay IO, waa marka tusaale ahaan aad email kumbuyuutarkaaga ka dirayso ama aad qof la sheekaysanayso. Haddii aynu iskusoo duubno IO waa macluumaad ku qorista ama kasoo akhriska meelaha kala duwan.

Luuqadaha kumbuyuutarada sida Java waxay kuu saamaxayaan inaad adeegsato IO-ga. Taas oo ka dhigan inaad macluumaad meelaha kala duwan ku qorto ama u dirto iyo inaad kasoo akhrido. Waxyaalaha ugu muhiimsan ee aad u isticmaali karto Java IO waxa ka mid ah tusaale ahaan inaad macluumaad ku qorto ama kasoo akhrido fayl, taas oo aynu tusaale ku saabsan halkan kusoo qaadan-doono. Waxa kale oo aynu tusaale u soo qaadanayaa sida aad isticmaalaha macluumaad u waydiinkarto oo ugasoo akhriyi karto keyboard-ka waxa uu ku qoro, tusaale ahaan magaciisa, da'diisa, telefoonkiisa, adarayskiisa, iwm.

Haddii aynu intaas kaga gudubno hordhac ahaan oo dhexgallo luuqadda Java, si aynu uga jawaabno su'aasha labaad. Noocyada Java ee ku saabsan ama loogu talagalay IO-ga (oo loo yaqaano Java IO) waxay ku jiraan package-ka `java.io` iyo package-hoosaadkeeda. Java IO sideedaba waxa aasaas u ah fikirka ama aragtida lagu magacaabay *streams*, taas oo runtii fududaynaysa fahamka iyo isticmaalka Java IO. Streams-ka Java waxay u shaqaynayaan isku qaab, laakiin waxay ku kala duwanyihiin tusaale ahaan macluumaadka ay qaadayaan noociisa. Waxan halkan isku dayay inaan sawir ka bixiyo streams-ka Java, waxaanad ka arkaysaa sawirkaas **10.1**.

Sidaad ka arkayso sawirka **10.1**, stream-ka waxa la odhan karaa waa "dhuun" ama "tuubo" isku xidhaysa source-ka iyo destination-ka. Source waa isha uu macluumaadku ka imanaayo. Tusaale ahaan haddii borogaraamku uu macluumaad ku qoraayo xasuusta wayn (**hard disk**-ka) wuxu noqonayaa source, xasuustuna waxay noqonaysaa destination. Haddiise uu macluumaad kasoo akhriyaayo xasuusta wayn wuxu noqonayaa destination, xasuustiina waxay noqonaysaa markaas source. Source waxa kale oo tusaale ahaan noqonkara keyboard-ka ama Internetka, waa haddii aad tusaale ahaan website macluumaad kasoo akhriyayso. Waxa kale oo noqon kara source borogaraam kale. Qalabka kumbuyuutarka wixii input ah waa source, wixii output ahna waa destination. Laakiin sidaan soo sheegayba qalabka qaar ayaa noqon kara input iyo output labadaba. Stream-ku wuxu qaadikaraa qoraal caadiha oo dadku fahmaayo ama macluumaad *binary* ah oo noqon kara sawir, fiidiyaw, iwm kaas oo dadku akhriyikarin. Stream-ka



Sawirka 10.1: Stream-ku waa sida tuubo oo kale, tuubadaas oo isku xidhaysa borogaraamkaaga iyo qalabka kala duwan sida: Internet-ka, shaashadda, keyboard-ka, jiirka, iwm.

source-ka leh waxa waxa loo yaqaan *Input Stream*, ka destination-ka lehna *Output Stream*. Ma doonaayo inaynu mawduucan ku fogaado, sidaas awgeed waxan u gudbayaa tusaalooyin kooban. Laba arimood ayaa inta badan u baahan inaad xasuusato markaad la dhiilgaraynayso streams-ka luuqadda Java:

1. Inaad furto, kuna xidho source-kii ama destination-kii
2. Inaad xidho markaad ka dhamaysato

Qodobka koobaad inta badan waa automatic, oo looma baahna inaad xasuusato, laakiin qodobka labaad ayaa aad u muhiim ah. Waa inaad xidho stream-ka markaad ka dhamaysato.

10.1 Write to file - Kuqorista faylka

Waxyaalaha ugu muhiimsan ee aad IO-ga uga baahanayso inta badan waxa kamid ah inaad fayl macluumaad ku qorto. Macluumaadkaas oo qoraal ama binary ah. Binary waxa noqon kara tusaale ahaan sawirrada ama fiidiyawyada sidaan soo sheegayba. Waxaynu halkan kusoo qaadanaynaa sidaad fayl qoraal ugu qorikarto. Waxaynu adeegsanaynaa nooca *IsbarJava* ee hore (haddii aad tuurtay noocaas, samee mid cusub, kuna tuur package-na nimi), laakiin calooshiisa ayaynu badalaynaa. Waxad noocaas u eekaysiisaa sida borogaraamka [10.1](#).

```

1 package nimi;
2
3 import java.io.FileWriter;
4 import java.io.IOException;
5
6 public class IsbarJava {
7
8     /**
9      * Ku qor faylka magaciisa lagu siiyey macluumaadka lagu siiyey
10     */
11     * @param magacaFaylka
12     *      magaca faylka
13     * @param macluumaadka
14     *      macluumaadka lagu qoraayo
15     * @throws IOException
16     *      haddii ciladi timaado, tusaale ahaan
17     *      xasuustu buuxsanto ama hard
18     *      disk-ku halawdo
19     */
20     private static void kuQorFaylka(String magacaFaylka,
21                                     String macluumaadka) throws IOException {
22         FileWriter fw = new FileWriter(magacaFaylka);
23         try {

```

```

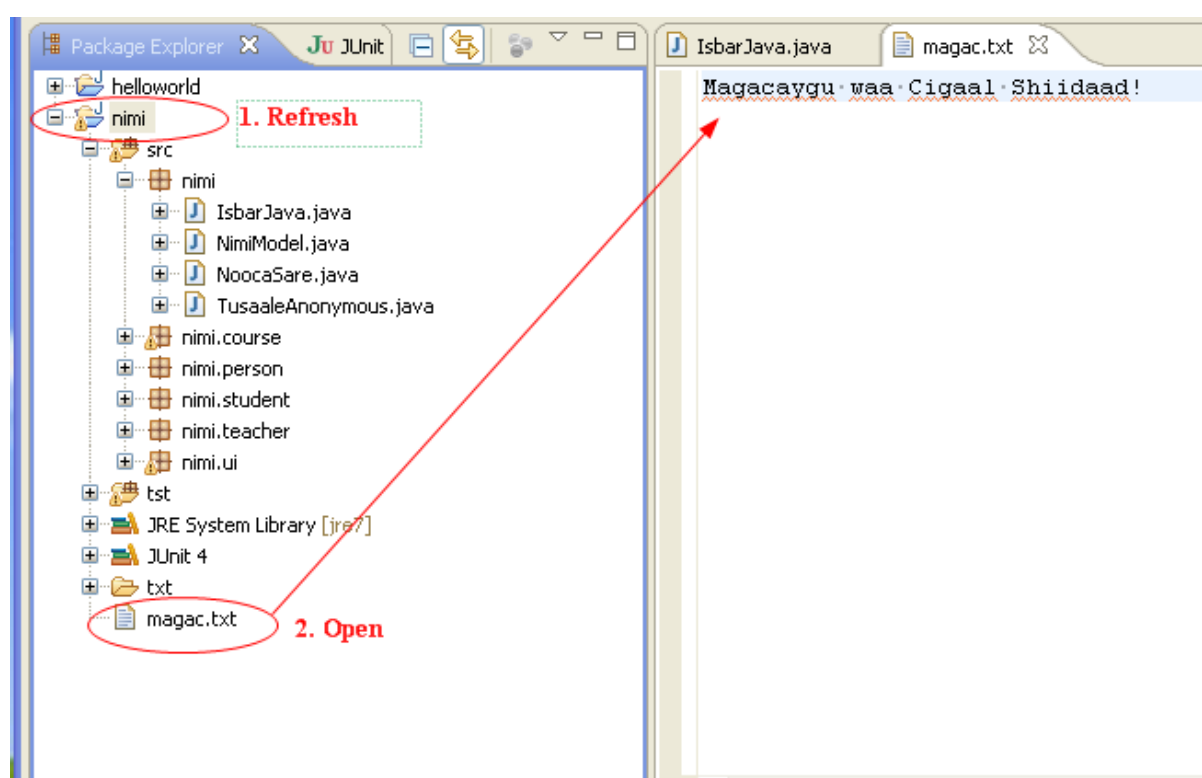
24     fw.write(macluumaadka);
    } finally {
26         fw.close();
    }
28 }

30 public static void main(String[] args) {
    try {
32         kuQorFaylka("magac.txt",
            "Magacaygu waa Cigaal Shiidaad!");
34     } catch (IOException e) {
        System.out.println("Waxa timid cilad!");
36         e.printStackTrace();
    }
38 }
}

```

Tusaale 10.1: Sidee ayaad fayl ugu qorikartaa macluumaad

Intaanan sharaxaad ka bixin borogaraamka sare ee 10.1 kaxee si aad u aragto wuxu samaynaayo. Sawirka 10.2 ayaa ku tusaaya waxa kuusoo bixidoona, kaxaynta kadib.



Sawirka 10.2: Borogaraamka 10.1 natiijadiisa. Si aad u aragto faylka uu borogaraamku sameeyey, dooro marka koobaad dooro project-ka kadibna refresh-garee (badhanka f5 guji), kadib waa inaad aragto faylka cusub. Intaas kadib laba jeer guji faylkaas cusub si aad u furto.

Intaas kadib, aan iskudayo inaan sharaxaad ka bixiyo borogaraamka sare ee 10.1.

1. Dhammaan noocyada ama ficilada la xidhiidha ku qorista faylasha ama akhriskooda waxay soo tuurayaan IOException, sidaas awgeed waa inaad isticmaasho **try ... catch...finally**
2. Inta ugu muhiimsan borogaraamka sare waa ficilka kuQorFaylka, kaas oo faylka aad magaciisa siisay macluumaadka aad siisay ku qoraaya. Ficilkaas waxaynu wacaynaa laynka 32-aad
3. Haddii caloosha ficilkaas ciladi ku timaado wuxu soo tuurayaa IOException, sidaas awgeed waa inuu isticmaaluhu ama wucuhu u diyaargarooobo inuu la dhiilgareeyo. Exception-kaas waxa lasoo tuurayaa haddii sababo kala duwan awgeed ay u guuldaraysato kuqorista faylku
4. Haddii aan sharaxaad ka bixiyo caloosha ficilkaas. Laynka 22-aad waxad ka arkaysaa inaynu adeegsanayno nooca FileWriter (oo ku jira package-ka java.io) si aynu qoraalka laynoosoo gudbiyey faylka ugu qorno. Noocaas ficilkiisa dhisaha ah waxa la siinayaa magaca faylka (ama magaca iyo

wadadda (path) oo dhamaystiran sida kuQorFaylka("c:\magac.txt", "Cigaal")). Intaas kadib, caloosha **try** (waa laymanka 23-25), waxaynu adeegsanaynaa ficilka write ee nooca FileWriter si aynu qoraalka ugu qorno faylkaas. Caloosha **try** kadib, wax kasta oo dhaca, waxa la kaxaynayaa caloosha **finally**, halkaas oo aynu adeegsanayno ficilka close, si aynu u xidhno stream-ka. Inkastoo aan la arkaynin, laynka 22-aad, wuxu furayaa stream-kii, laynka 26-aadna wuu xidhayaa marka laga dhamaysto. Labadaas qodob, haddii aad xasuusato, waxay ku gaadhsiinayaan meel fog. Haddii sababo jira awgeed ficilka kuQorFaylka uu soo tuuro exception (taas oo noqonkarta kaliya IOException), waxa qabanaaya **catch**-ka laynka 34-aad, kaas oo aynu isticmaalaha tusayno fariinta "Waxa timid cilad!".

Haddii aad adeegsanayso Java 7 iyo wixii ka dambeeyey, tusaalaha sare, gaar ahaan caloosha ficilka kuQorFaylka waxad u soo gaabinkartaa sida tusaalaha 10.2 ee hoose. Buuggani wuxu target-garaynayaa Java soosaarkeedii 6-aad iyo wixii ka dambeeyey.

```

1 package nimi;
2
3 ...
4
5 public class IsbarJava {
6
7     ...
8     private static void kuQorFaylka(String magacaFaylka,
9         String macluumaadka) throws IOException {
10         // FileWriter fw = new FileWriter("c:\\temp\\magac.txt");
11         try (FileWriter fw = new FileWriter(magacaFaylka)) {
12             fw.write("Magacaygu waa Cigaal Shiidaad!");
13         }
14     }
15     ...
16 }

```

Tusaale 10.2: Tusaalaha 10.1 oo lagu qoray Java 7

10.2 Read a file - Akhrista fayl

Arinta labaad ee aad loogu baahanaayo ee lagama maarmaanka ah waa in macluumaad fayl ku kaydshan la soo akhriyo, si tusaale ahaan wax looga badalo ama loo cusboonaysiiyo. Waxaynu ku xigsiinaynaa tusaale ku saabsan arintaas. Borogaraamka 10.3 ee hoose, wuxu ku tusayaa sidaad macluumaadka ku kaydshan faylkii uu borogaraamka hore sameeyey aad u soo akhriyi karto. Sidaas awgeed waxa shuruud ah inaad borogaraamka hore kaxayso si faylkaasi u dhasho, kadibna aad borogaraamkan isticmaasho si aad macluumaadka ku kaydshan faylkaas u soo akhrido.

```

1 package nimi;
2
3 import java.io.BufferedReader;
4 import java.io.FileReader;
5 import java.io.IOException;
6
7 public class IsbarJava {
8
9     /**
10      * Iisoo akhri macluumaad ku jira faylka, kadibna iisoo celi
11      * isaga oo String
12      * ah
13      *
14      * @param magacaFaylka
15      * @return String matalaaya waza faylka ku jira
16      * @throws IOException
17      *         haddii ciladi timaado, tusaale ahaan
18      *         faylkaasi aanu jirin
19      */
20     private static String isiiWaxaKujiraFaylka(
21         String magacaFaylka) throws IOException {
22         BufferedReader br = new BufferedReader(new FileReader(
23             magacaFaylka));
24         try {
25             StringBuilder sb = new StringBuilder();
26             String line = null;
27             while ((line = br.readLine()) != null) {
28                 sb.append(line);
29             }
30         }
31     }
32 }

```

```

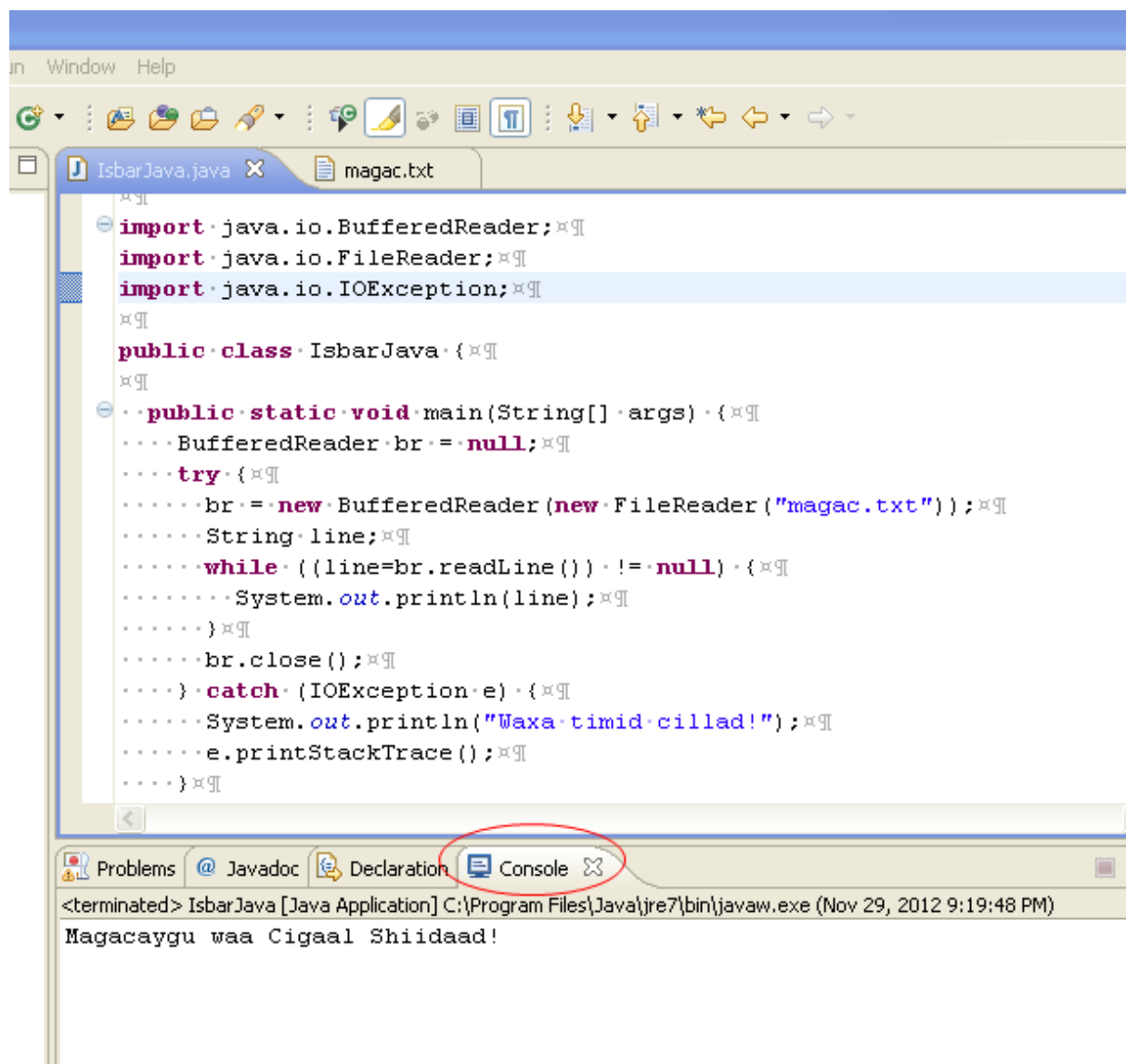
    return sb.toString();
31 } finally {
    br.close();
33 }
}

35 public static void main(String[] args) {
37     try {
        String waxaKujiraFaylka = isiiWaxaKujiraFaylka("magac.txt");
39     System.out.println(waxaKujiraFaylka);
    } catch (IOException e) {
41     System.out.println("Waxa timid cilad!");
        e.printStackTrace();
43     }
    }
45 }

```

Tusaale 10.3: Borogaraam ku tusaaya sida aad fayl u akhriyi karto macluumaadka ku jira

Aynu kaxaynno intaynaan wax sharaxaad ka bixin borogaraamka 10.3. Waa inuu kuusoo baxo sida sawirka 10.3 ee hoose.



Sawirka 10.3: Borogaraamka 10.3 natiijadiisa. Wixii faylka ku qornaa waxad ka arkaysaa caleenta console-ka

Aan sharaxaad kooban ka bixiyo isna borogaraamka sare ee 10.3:

1. Borogaraamka 10.3 wuxu aad ugu dhawyahay kii hore, laakiin waxaynu halkan ku isticmaalaynaa `FileReader` iyo `BufferedReader` (iyo nooca kalkaaliyaha ah ee `StringBuilder`), maadaama aynu rabno

inaynu soo akhrino macluumaadka ku kaydsan faylka. Laakiin sidaad ka arkayso laynka 22-aad `FileReader` waa in lagu wareejiyo nooca `BufferedReader`, kaas oo inoo saamaxaaya inaynu faylka layn-layn u akhrino, waxaana inoo saamaxaaya ficilkiisa `readLine`. Markuu faylku dhamaado ficilkaasi wuxu soo celinayaa `null`, sidaas awgeed waa inaynu markaas joojino wareegga akhriska ee `while`

2. Ficilka `isiiWaxaKujiraFaylka` ee aynu wacayno laynka 38-aad wuxu soo celinayaa caloosha faylka. Kadib, laynka 39 ayaynu isticmaalaha tusaynaa wixii faylka laga soo akhriyey. Isla sidii boroga-raamkii hore waa inaad xasuusato inaad stream-ka xidho markaad ka dhamaysato sida ka muuqata laynka 32-aad. Ficilka `toString` ee nooca `StringBuilder` ayaa soo celiyaana waxa ku jira `StringBuilder`-ka, sida ka muuqata laynka 25-aad
3. Bal intaanad ka gudbin tusaalahan delete-garee ama tuur faylka "magac.txt", kadibna kaxee borogaraamka sare ee 10.3.

Intaynaan qaybtan ka gudbin waxan rabaa inaan sheego in marka aad qoraalka caadiga ah fayl ku kaydinaayso ama aad kasoo akhriyayso inaad isticmaasho noocyada `FileWriter`, `FileReader` iyo `BufferedReader`. Luuqadda Java waxa kale oo ay kuu saamaxaysaa inaad walxo ku qorto fayl, kadibna ka soo akhrido iyo inaad faylal aan qoraal ahayn oo tusaale ahaan fiidiyaw ama sawirro ah qorto ama akhrido, sidaan soo sheegayba. Qaabka aad walxaha fayl ugu kaydinkarto, kadibna aad ugasoo akhriyikarto waxaynu kaga hadlaynaa isla buuggan, laakiin qorista iyo akhriska sawirrada ama fiidiyaw-ga ama wixii u dhigma halkan kaga hadlimayno. Noocyada aad magacooda ku aragto *Reader* iyo *Writer* waxad u adeegsan kartaa qoraalka caadiga ee xarfaha iyo nambarada ka kooban markaad qorayso ama akhriyayso.

10.3 User Input- Waydiinta macluumaad isticmaalaha

Waxa jira xaalado badan oo loo baahanyahay in isticmaalaha macluumaad la waydiiyo sida magaciisa, username-kiisa, password-kiisa, adarayskiisa, telefoonkiisa, iwm. Sida caadiga arintan oo kale haddii ay inasoo wajahdo, waxa borogaraamka loo sameeyaa waji ama arag, taas oo aynu kaga hadlayno qaybta ugu dambaysa ee buuggan. Laakiin haddii aanay suurtoagal ahayn in borogaraamka loo sameeyo waji ama arag, laakiin loo baahanyahay in macluumaad isticmaalaha laga soo akhriyo, waa inay suurtoagal noqoto. Nasiibwanaag luuqadda Java kugu qasbimayso inaad borogaraamka waji u samayso si aad macluumaad isticmaalaha ugasoo akhrido. Waxaynu halkan kusoo qaadanaynaa borogaraam yar oo ku tusaaya sida aad tusaale ahaan isticmaalaha u waydiinkarto magaciisa iyo da'diisa. Borogaraamkii oo diyaar ah waxad ka arkaysaa tusaalaha 10.4 ee hoose.

```

1 package nimi;
2
3 import java.io.Console;
4
5 public class IsbarJava {
6
7     public static void main(String args[]) {
8
9         Console c = System.console();
10        if (c == null) {
11            System.err.println("Console lama helin!");
12            System.exit(1);
13        }
14
15        String magac = c.readLine("Magaca: ");
16        String da = c.readLine("Da'daada: ");
17
18        // Number u badal da'da
19        int daTiroAh = Integer.parseInt(da);
20        if (daTiroAh > 30) {
21            c.printf("%s, waxad tahay oday!", magac);
22        } else if (daTiroAh < 15) {
23            c.printf("%s, waxad tahay ciyaal wali!", magac);
24        } else {
25            c.printf("%s, wali dhallinyarada ayaad ku jirtaa!",
26                    magac);
27        }
28    }
29 }

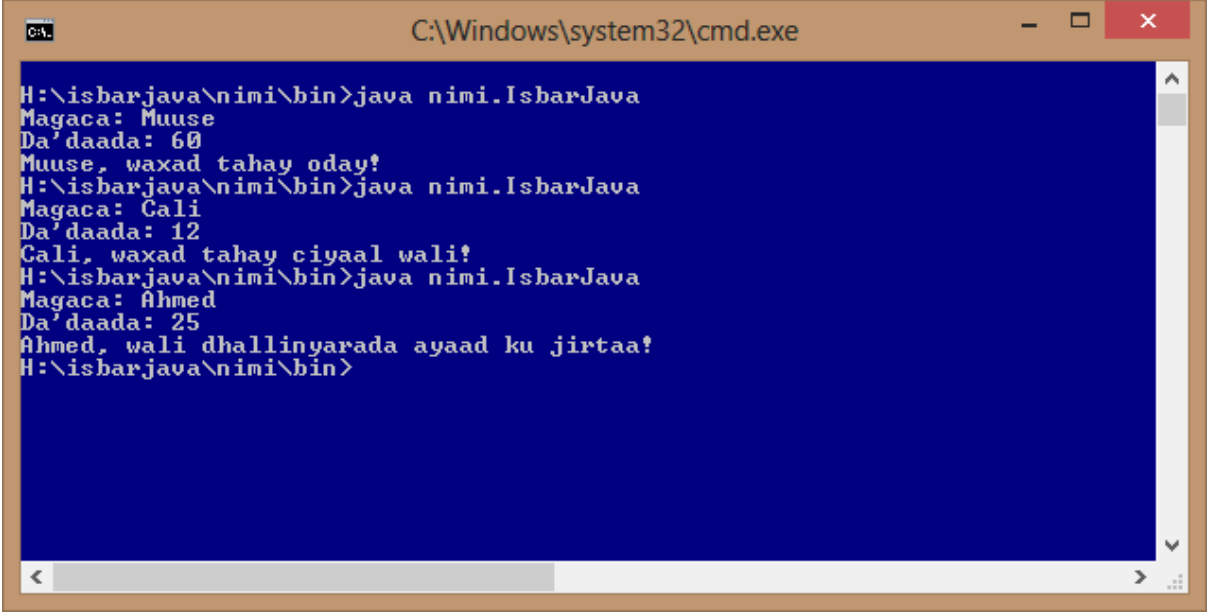
```

Tusaale 10.4: Isticmaalka Console-ka si aad isticmaalaha macluumaad u waydiiso

Borogaraam sare ee 10.4 haddii aad borogaraamka Eclipse ka kaxayso adiga oo adeegsanaaya `RunJava`, waxa kuusoo baxaysa:

Console lama helin!

Taasna waxa keenaaya cilad ku jirta borogaraamka **Eclipse**. Marka borogaraamyada noocan oo kale ah laga kaxeeyo **Eclipse**, ficilka `System.console()` ee laynka 10-aad wuxu soo celinayaa **null**, halkii laga sugaayey walax Console ah. Si aad uga waraagto arintan, waa inaad borogaraamka toos command line-ka uga kaxayso sida ka muuqata sawirka 10.4.



```
C:\Windows\system32\cmd.exe
H:\isbarjava\nimi\bin>java nimi.IsbarJava
Magaca: Muuse
Da'daada: 60
Muuse, waxad tahay oday!
H:\isbarjava\nimi\bin>java nimi.IsbarJava
Magaca: Cali
Da'daada: 12
Cali, waxad tahay ciyaal wali!
H:\isbarjava\nimi\bin>java nimi.IsbarJava
Magaca: Ahmed
Da'daada: 25
Ahmed, wali dhallinyarada ayaad ku jirtaa!
H:\isbarjava\nimi\bin>
```

Sawirka 10.4: Borogaraamka sare oo laga isticmaalay command line-ka. Marka hore waa inaad tagto folder-ka `bin` oo ku hoos jirta meesha uu project-ka `nimi` kumbuyuutarkaaga kaga kaydsanyahay.

Haddii aanad rabin inaad isticmaasho nooca `java.io.Console` sida borogaraamka sare ee 10.4, waxad toos u adeegsan kartaa `System.in`, `System.out` iyo `java.util.Scanner`, kuwaas oo aad kusoo aragtay cutubka 5. Waxan kaga gudbayaa Java IO intaas.

QAYBTA III

DHISIDDA WAJIGA AMA
ARAGGA (GUI)

Qaybtani waa qaybtii ugu dambeysay ee buuggan. Waxaanay ku saabsantahay sidaynu hore u soo sheegnayba sameynta wajiga/aragga/muuqaalka borogaraamka. Muuqaalka borogaraamka waxa loo yaqaan *Graphical User Interface* oo loo soo gaabiyo (*GUI*) ama kaliya *User Inteface (UI)*. Haddii borogaraamku aanu lahayn GUI, waxa loo yaqaan borogaraamkaas *Command line program* ama *Console program*. Halka borogaraamyada Console-ka ahi adeegsadaan inta badan kaliya qoraal (*text*), borogaraamyada leh User Interface-ku, waxa kale oo ay adeegsadaan sawirro (*images*), cod (*audio*) iyo fiidiyawba (*video*).

Waxan hubaa in qaybtani tahay qaybta ugu xiisaha badan buuggan. Waayo dadka badankoodu waxay jecelyihiin inay arkaan borogaraamka ay sameeyeen caynka uu u eegyahay iyo inay dadka kale tusaan. Mida kale, borogaraamyada UI-ga leh wuu ka fududyahay isticmaalkoodu kuwa aan lahayn, sidaas awgeed intabadan waxaynu ku qasbannahay inaynu borogaraamka u samaynno UI ama waji. Laga bilaabo halkan inta badan waxan adeegsanayaa ereyga waji ama arag markaan ka hadlaayo UI-ga borogaraamka.

Borogaraamka aynu halkan ku samaynayno waa mid kumbuyuutar kaliya ku eeg. Halka *distributed system* uu ka yahay borogaraam ama nidaam daadsan oo ku faafsan kumbuyuutarro laba ama ka badan, kuna wada hadlaaya *network*-ka. Sideedaba borogaraamyada distributed-ka ah way ka adagyihiin, kuwa aan ahayn distributed qoristoodu. Sidaas awgeed, midhaha waxa la gurtaa kuwa kuu dhaw horta, markaas waxaynu halkan ku baranaynaa sidii aad u samaynlahayd borogaraam ku eeg kumbuyuutar kaliya, adiga oo adeegsanaaya luuqadda Java.

Magaca *Desktop application* ayaa inta badan loo adeegsadaa borogaraamyada kumbuyuutarka ku kaydsan ee aan ahayn distributed system-ka. Waxaana ka mid ah tusaale ahaan Internet Explorer, Firefox, Notepad, Wordpad, Microsoft Word, Eclipse, iwm.

Ilaa hadda waxaynu adeegsanaynay nidaamka **TDD** oo waxaynu ku horaynaynay inaynu nooc walba qoristiisa ku bilawno noociisii tijaabiyaha ahaa. Taasi waxay keentay inaynu hello noocyo badan oo tijaabiyeyaal ah, oo shaqadoodu tahay inay tijaabiyaan noocyada kale inay sax u shaqaynayaan iyo in kale. Qaybtii labaad waxay ahayd *logic*-kii ama maskaxdii (waxa kaloo loo yaqaan *model*) borogaraamka, waxaana fiican inaad si fiican u tijaabiso oo **TDD** u adeegsato qoristiisa. Aragga ama UI-ga borogaraamka laftiisa waxa fiican inaad tijaabiso haddii aad kari karto. Tijaabinta araggu way ka adagtahay tijaabinta model-ka borogaraamka. Taasi waxay keentay in dad badan iska daayaan tijaabinta aragga borogaraamka, laakiin ay ku dadaalaan inay ka dhigaan khafiif. Way jiraan *library*-yo loogu talagalay in lagu tijaabiyo UI-ga borogaraamka waxaana ka mid ah tusaale ahaan Abbot iyo Jemmy. Library sideedaba waxa loo yaqaan borogaraamyo yaryar oo loogu talagalay in lagu adeegsado borogaraamyada kale, kuwaas oo qabanaaya hawl khaas ah. Library-gu kalidii waxba ma taro inta badan ee wuxu qayb ka yahay borogaraamka kale. Maadaama tijaabinta wajiga borogaraamku aanay fududayn, waxan qabaa inaad logic-ka borogaraamka si fiican u tijaabiso, laakiin aanad isku dhibin tijaabinta wajiga. Sidaas awgeed waxan go'aansaday inaan tijaabinta wajiga ka daayo daabacaadda hore ee buugga. Sidaas awgeed qaybtan ma adeegsandoonno **TDD** oo ma tijaabinayno noocyada aynu qoridoono.

Cutub 11

Java Swing

UI-ga borogaraamyada Java inta badan waxa lagu qoraa Swing. Swing wuxu kamid yahay luuqadda Java, sidaas awgeed looma baahna inaad meel kale ka raadiso. Swing waxba taariikhdiisa iyo ka sheekayntiisa wakhti iskaga luminmayno, oo waxad ka helaya Internet-ka, laakiin saddex qodob oo kaliya ayaynu ka sheegaynaa intaynaan u gudbin isticmaalkiisa.

1. Swing waa library kamid ah luuqadda Java, waxaanu ka dambeeyey library-gii hore ee AWT - Abstract Window Toolkit, kaas oo hadda laga baxay. Swing shaqada koobaad ee loogutalogay waa dhisidda aragga ama muuqaalka borogaraamyada Java
2. Swing wuxu leeyahay *look and feel* la badali karo, taas oo saamaxaysa in borogaraamka Java loo eekaysiiyo OS-ka lagu isticmaalaayo
3. Swing waa *platform-independent*, taas oo ka dhigan in borogaraamyada Swing lagu sameeyey lagu isticmaali karo OS-yo kala duwan, iyada oo aan qoraalkii borogaraamka waxba laga badalin.

11.1 Hello Swing

Sidaan hore u soo sheegayba midhaha waxa la gurtaa oo lagu horeeyaa kuwa kuu dhaw marka hore. Waxaynu samaynaynaa borogaraamkii ugu fududaa ee Swing lagu samaynkaro, si aynu qiyaas uga hello waxa innaga horeeya. Borogaraamkani wuxu u dhigmaa kii hore ee Hello World ee cutubka 2.3.

Waxad samaysaa nooc cusub, adiga oo raacaya talaabada **NewClass**, una buuxbuuxi sidan: Source folder=nimi/src, Package=nimi.ui, Name=HelloSwing, una eekaysii tusaalaha 11.1 ee hoose.

```
1 package nimi.ui;
2
3 import javax.swing.JFrame;
4 import javax.swing.JLabel;
5 import javax.swing.SwingUtilities;
6
7 public class HelloSwing {
8
9     public static void createAndShowGUI() {
10         // JFrame waa daaqadda ugu saraysa borogaraamka
11         JFrame frame = new JFrame("HelloSwing");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         // JLabel waxa loo isticmaalaa qoraalka ama text-ka
14         JLabel label = new JLabel("Hello World");
15
16         // dhexgali qoraalka daaqadda kadibna uruuri
17         frame.getContentPane().add(label);
18         frame.pack();
19
20         // tus daaqadda
21         frame.setVisible(true);
22     }
23     public static void main(String[] args) {
24         SwingUtilities.invokeLater(new Runnable() {
25             public void run() {
26                 createAndShowGUI();
27             }
28         });
29     }
30 }
```

Tusaale 11.1: HelloSwing - waa noocii ugu horeeyay ee aynu Swing ku isticmaalo

Sawirka 11.1 waxad ka arkaysaa borogaraamka 11.1 sida uu u eegyahay marka lagu isticmaalo kumbuyuutar uu ku jiro OS-ka Windows XP iyo mid uu ku jiro Linux Ubuntu. Waxad markiiba arkaysaa in borogaraamkii Java dhanka muuqaalka uu isbadalay, una eekaaday OS-ka lagu isticmaalaayo.



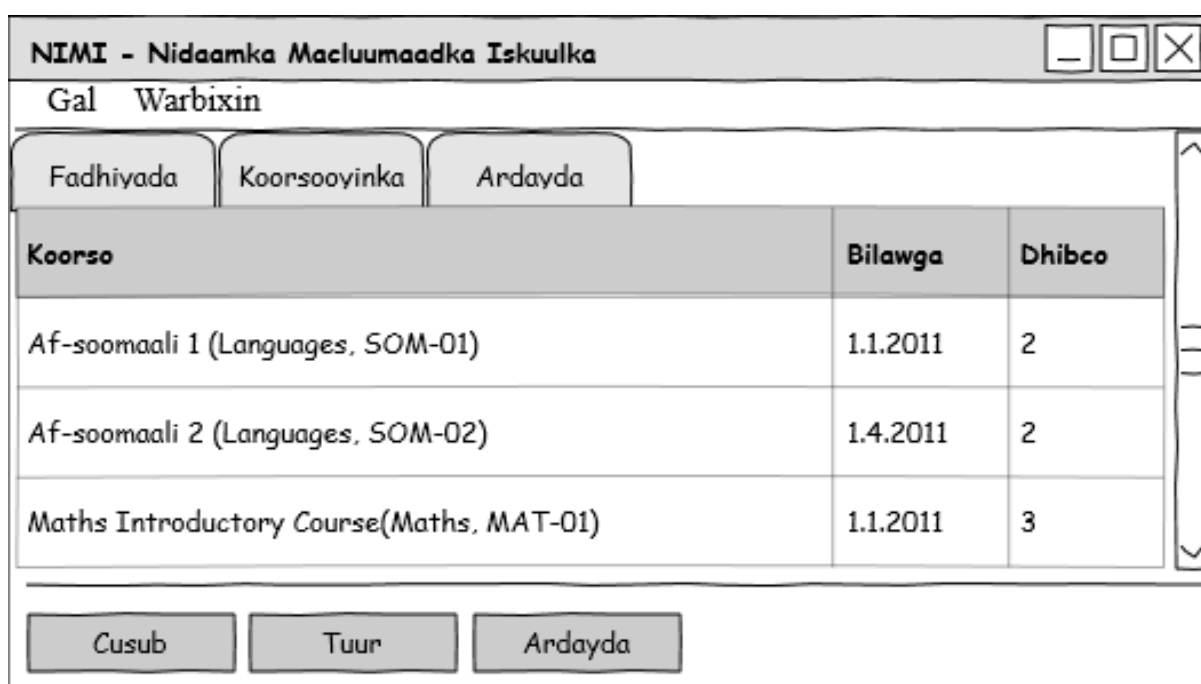
Sawirka 11.1: Borogaraamka 11.1 caynka uu eegyahay marka lagu isticmaalo OS-yo kala duwan

Waxba sharaxaad borogaraamka 11.1 ka bixinmayno, maadaama aanu faa'iido badan inoo lahayn. Waxaynu sharaxaad fiican ka bixinaynaa borogaraamka aynu samaynayno. Cutubkan intaas ayaynu kaga gudbaynaa oo waxaynu bilaabaynaa inaynu borogaraamkii nimi u samayno arag si fiicanna u sharaxno talaabo kasta oo aynu qaadno.

Cutub 12

Qorshaha

Waxaynu gacanta ku sawiraynaa qaabka aynu rabno inuu u eekaado borogaraamka **nimi** araggiisu. Waxa laga yaabaa in dadka kale u sawiraan qaab kale oo ay rabaan in borogaraamku u eekaado, laakiin anigu waxan ku fikiray inaynu borogaraamka wajigiisa u eekaysiino sida sawirka 12.1.



Sawirka 12.1: Qaabka aynu isleenahay u eekaysiia wajiga borogaraamka

Sawirka 12.1 waxad ka arkaysaa borogaraamka **nimi** qaabka aynu is leenahay u eekaysiia (sawirkan waxan ku sawiray borogaraam bilaash ah oo la yidhaahdo Pencil¹). Marka hore borogaraamku waa inuu yeesho daaqad u gaar ah, taas oo uu madaxa kaga qoranyahay magaca borogaraamku. Daaqaddu waa inay yeelato meeshii laga yarayn, waynayn ama xidhi lahaa (giftinka midigta sare). Intaas kadib, qaybta Menu-da oo ka hoosaysa madaxa sare, waxa laga helayaa *Gal* iyo *Warbixin*. *Gal* (File [1]) waxa ku hoos jira tusaale ahaan furista (Open File...), kaydinta (Save) iyo u kaydinta (Save As...) iyo xidhista (Exit) ama ka bixista borogaraamka. *Warbixinta* hoosteeda waxad ka helayaa soo saarista qoraal warbixin ah oo dhamaystiran. Menu-da kadib waxa imanaysa calooshii borogaraamka, taas oo ka kooban caleemo is baalyaalla oo soo bandhigaaya macluumaad kala duwan, sida fadhiyada, koorsooyinka iyo ardayda (hadaad rabto waxad ku dari kartaa caleen macalimiinta ah). Caleen kasta hoosteeda waxad ka helayaa badhanno kuu saamaxaaya inaad macluumaadka caleentaas wax ka badasho, tusaale ahaan fadhi cusub ku darto ama ka tuurto borogaraamka. Intaas waxaynu ku dhaafaynaa sawirkii guud ee ku tusaayey ujeedada aynu u socono. Laga bilaabo imika waxaynu faraha la galaynaa qooshka oo waxaynu bilaabaynaa inaynu samayno wajigii borogaraamka.

¹<http://pencil.evolus.vn/>

Cutub 13

Daaqadda Borogaraamka

Waxaynu dhisidda wajiga borogaraamka ku bilaabaynaa samaynta daaqadda wayn oo af-ingiriisiga la yidhaahdo *main window* ama *main frame*, kadibna waxaynu bilaabaynaa inaynu wixii ka dhiman caloosheeda ku gurno. Waxad bilawdaa nooc cusub, kaas oo aad u bixiso Nimi, kuna rid package-ka nimi.ui, waxaanad u eekaysiisaa tusaalaha 13.1.

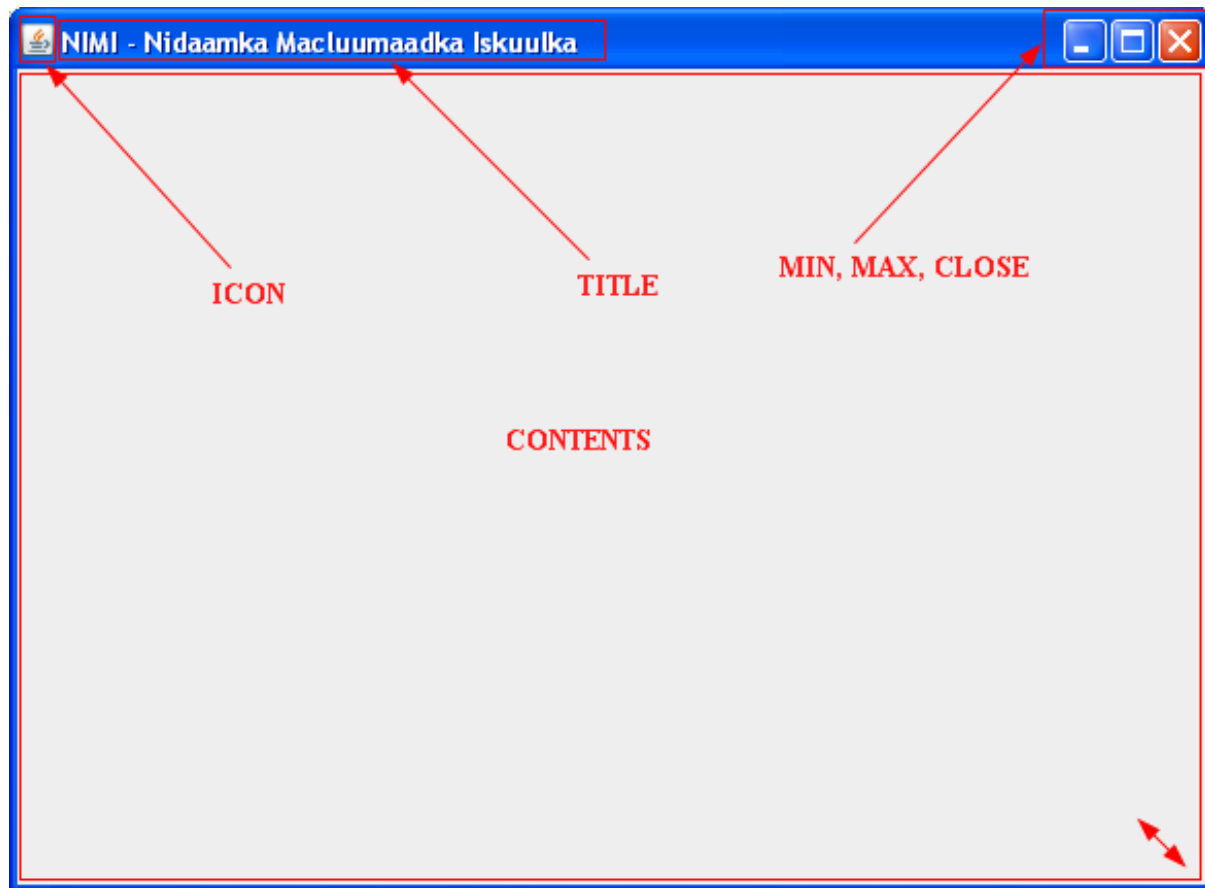
```
1 package nimi.ui;
2
3 import javax.swing.JFrame;
4 import javax.swing.SwingUtilities;
5
6 public class Nimi extends JFrame {
7
8     public Nimi() {
9         setTitle();
10        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11    }
12
13    /**
14     * Ficilka main waa albaabka borogaraamka Java sidaad ogtahayba
15     */
16    public static void main(String[] args) {
17        SwingUtilities.invokeLater(new Runnable() {
18            public void run() {
19                Nimi nimi = new Nimi();
20                nimi.setVisible();
21            }
22        });
23    }
24
25    private void setTitle() {
26        setTitle("NIMI - Nidaamka Macluumaadka Iskuulka");
27    }
28
29    private void setVisible() {
30        pack();
31        setVisible(true);
32    }
33 }
```

Tusaale 13.1: Main window-ga ama daaqadda sare borogaraamka nimi

Kadib, haddii aad 13.1 kaxayso adiga oo raacaya talaabada **RunJava**, waa inay kuusoo baxdo daaqadda sawirka 13.1 (maadaama daaqaddu madhantahay way yartahay, sidaas awgeed waa inaad waynayso, intaad giftinka midigta ee hoose jiirka ku jiido)

Haddii aad u fiirsato daaqadda sawirka 13.1 waxad arkaysaa inay u eegtay sawirka 12.1 ee qorshaha inoo ahaa, laakiin caloosheedu madhantahay. Haddii aan sharaxaad ka bixiyo borogaraamka 13.1, sawirka kadib:

1. Laynka koobaad waxad ka arkaysaa package-ka nimi.ui oo aynu ku uruurinayno noocyada la xidhiidha wajiga borogaraamka. Laga bilaabo halkan wixii nooc (class) ee cusub ee la xidhiidha wajiga borogaraamka waxad ku tuurtaa package-kaas
2. Nooca noqonaaya daaqadda borogaraamka waa inaad kasoo dhiraandhiriso nooca javax.swing.JFrame sida ka muuqata laynka 6-aad, laakiin si aad noocaas u isticmaasho waa inaad soo **import**-garayso, sida ka muuqata laynka 3-aad. Waxa kale oo aad u baahantahay nooca kalkaaliyaha ah ee



Sawirka 13.1: Daaqadda ama Frame-ka sare ee nimi

`javax.swing.SwingUtilities` ee laynka ka hooseeya. Xasuuso in package-ka nooca `JFrame` iyo noocyo badan oo soo socda uu tahay `javax.swing`. Package-kaas waxa ku kaydsan noocyada laf-dhabarka u ah `Swing`. Markaad noocyada lasoo **import**-gareeyey isha mariso, markiiba waxad arkaysaa inuu borogaraamkaasi isticmaalaayo `Swing` iyo in kale

- Intaas kadib ficilka dhisaha ama constructor-ka ee laymanka 8-11, waxaynu wacaynaa ficilka `setTitle` ee laymanka 25-27, kaas oo isticmaalaaya ficilka la magaca ah ee `setTitle` ee nooca `JFrame` si uu u badalo magaca sare ee daaqadda. Laynka xiga ee 10-aad waxaynu adeegsanaynaa ficilka `setDefaultCloseOperation`, kaas oo aynu siinayno qiimaha constant-ka ah ee `JFrame.EXIT_ON_CLOSE`, oo ujeedadiisu tahay in borogaraamku is xidho, marka uu isticmaaluhu gujiyo xiritaanka (calaamadda cross-ka ee giftinka midigta sare)
- Sidaynu hore u soo baranayba borogaraamka marka la istaado wuxu ka bilaabmayaa ficilka `main(String[] args)` ee laymanka 16-23, kaas oo samaynaaya walax cusub oo nooceedu yahay `Nimi` (waa laynka 19-aad), kadibna isticmaalaaya ficilkeeda `setVisible` ee laymanka 29-32. Ficilkaasi wuxu samaynayaa laba talaabo. Marka hore wuxu wacayaa ficilka `pack`, kaas oo ujeedadiisu tahay inuu daaqadda uruuriyo si ay u leekaato wixii caloosheeda ku jira, kadibna `setVisible(true)` (halka `setVisible(false)` ay ka qarin lahayd) oo muujinaysa daaqadda. Nooca `SwingUtilities`, gaar ahaan ficilkiisa `invokeLater` halkani waa meesha kaliya ee aynu ku adeegsanayno, markaas waxa kugu filan inaad xasuusato sidan aynu u adeegsanay, waayo sharaxaad dheeraad ah halkan kaga bixinmayno. Ficilka `main` sidaas ayuu ku dhanyahay oo mar dambe wax ka badalimayno
- Iskusoo duub: waxaynu samaynay walax nooceedu yahay `Nimi`, taas oo laga soo dhiraandhiriyeey nooca `JFrame`. Walaxdaasi waxay noqonaysaa daaqadda sare borogaraamka, kadibna waxaynu tusaynaa isticmaalaha.

Waxa jirta arin muhiim ah oo laga yaabo inuu akhristaha fiiro dheeri iswaydiiyo. Nooca `Nimi` iyo nooca `JFrame` waxay leeyihiin ficillo isku magac ah, tusaale ahaan ficilka `setTitle`, su'aashu waxay tahay sidee ayuu borogaraamku ku garanayaa ficilka uu isticmaalaayo, haddii ay jiraan ficillo isku magac ah (tusaale ahaan laynka 9-aad sidee ayuu ku garanayaa ficilka `setTitle` ee uu borogaraamku isticmaalaayo, ma ka noocan mise ka nooca laga soo dhiraandhiriyeey, midkee ayaa la wacayaa)? Luuqadda Java waxay

ogoshahay inay noocyadu yeeshaan ficillo isku magac ah, si kasta oo ay noocyadaasi isugu xidhanyihiin. Ficilka calooshiisa waxad ku isticmaali kartaa ama wici kartaa ficilada kale xitaa kuwa ka mid ka ah noocyada aad kasoo dhiraandhirisay noocaaga. Waxa jira laba mawduuc oo arintan la xidhiidha oo aan rabo inaan borogaraamka 13.2 ee hoose kugu tuso (waxad ku qortaa nooca IsbarJava ee aynu hore u soo isticmaalay).

```

package nimi;

2
class Xisaabiye {
    // isugee laba nambar
    public int isugee(int a, int b) {
        int natiiyo = a + b;
        System.out.println(String.format("Xisaabiye:%d+%d=%d", a,
8         b, natiiyo));
        return natiiyo;
10    }

    // isugee saddex nambar
    public int isugee(int a, int b, int c) {
        int natiiyo = a + b + c;
        System.out.println(String.format("Xisaabiye:%d+%d+%d=%d",
14         a, b, c, natiiyo));
        return natiiyo;
16    }
18 }

20
class Xisaabiye2 extends Xisaabiye {
    // isugee laba nambar
    public int isugee(int a, int b) {
        int natiiyo = super.isugee(a, b);
        System.out.println(String.format("Xisaabiye2:%d+%d=%d", a,
24         b, natiiyo));
        return natiiyo;
26    }
28 }

30
public class IsbarJava {
32
    public static void main(String[] args) {
        Xisaabiye xisaabiye = new Xisaabiye();
        xisaabiye.isugee(10, 20);
        xisaabiye.isugee(10, 20, 30);
        System.out.println("-----");
        Xisaabiye2 xisaabiye2 = new Xisaabiye2();
        xisaabiye2.isugee(10, 20);
38    }
40 }

```

Tusaale 13.2: Method overloading iyo method overriding

Haddii aad kaxayso borogaraamka sare waxa kuusoo baxayaa sidan:

```

Xisaabiye:10+20=30
Xisaabiye:10+20+30=60
-----
Xisaabiye:10+20=30
Xisaabiye2:10+20=30

```

Aan sharaxaad ka bixiyo arimaha innagu cusub ee borogaraamka 13.2:

- Hore umaynaan arag saddex nooc oo hal fayl ku wada jira. Borogaraamka sare waxa hal fayl (IsbarJava.java) aynu ku qornay saddex nooc Xisaabiye, Xisaabiye2 iyo IsbarJava. Halkaas waxad ka arkaysaa in luuqadda Java ay ogoshahay in noocyo badan hal fayl lagu uriiriyo, laakiin waxa jirta hal shuruud. Noocyadaas mid kaliya ayaa noqon kara Public, sidaas awgeed waxa halkan Public ah kaliya nooca Isbarjava
- Qodobka labaad ee innagu cusub ee tusaalaha sare aynu ku isticmaalay waa ficilkan String.format. Wali maynaan arag ficil toos nooca looga isticmaalo. Illaa hadda waxaynu marka hore samaynaynay walax, kadibna waxaynu wacaynay ficilada walaxdaas. Luuqadda Java waxay kuu saamaxaysaa inaad nooca u samayso ficilo aan ku xidhnayn walxaha laga sameeyo noocaas. Ficiladaas oo toos loo adeegsanaayo iyada oo la isticmaalaayo qaabka MagacaNooca.MagacaFicilka. Si aad u samayso ficillo noocaas oo kale ah oo aan ku xidhnayn walxaha laga sameeyo noocaas, waa inaad ficilka

u qorto sidan oo kale **public static** ReturnType methodName(parameters){...}. Ficilada sidan oo kale ah waxa loo yaqaan *Static methods*. Isticmaalka ficilka `String.format`, wuxu inoo saamaxayaa inaynu samayno qoraal aynu buuxbuuxinay, oo qiimeyaasha doorsoomeyaasha ku buuxbuuxinay, sidaad ka arkaysa kor.

13.1 Method overloading - rarista ficilka

Nooca Xisaabiye wuxu leeyahay laba ficil oo labadoodaba magacoodu yahay isugee. Labadooduba waxay soo celinayaan qiime **int** ah. Si haddaba lagu kala garto ficiladaas, ficil kastaaba wuxu leeyahay waxa loo yaqaano saxeex (*method signature*), kaas oo ka kooban magaca ficilka iyo noocyada parameters-kiisa. Sidaas awgeed ficilada isugee(**int**, **int**) ee laynka 5-aad iyo ficilka isugee(**int**, **int**, **int**) ee laynka 13-aad way kala duwanyihiin maadaama uu saxeexoodu kala duwanyahay. Arintan ayaa loo yaqaan *Method overloading* ama *rarista ficilka*. Halkaas waxad ka arkaysaa in labada ficil ee `setTitle()` iyo `setTitle(String)` ee tusaalaha 13.1 ay kala duwanyihiin.

13.2 Method overriding - burinta ficilka

Isla tusaalaha sare ee 13.2 waxad ka arkaysaa in labada nooc ee Xisaabiye iyo Xisaabiye2 ay labadooduba leeyihiin ficilka `isugee(int, int)`. Xaaladdan oo kale waxa la odhanayaa nooca hoose wuxu *burinayaa* ama *override*-garaynayaa ficilka nooca sare ee isla magacaas iyo parameters-kaas leh. Waxa kale oo aad nooca hoose ka arkaysaa in haddii aad rabto inaad wacdo ficilka nooca sare aad isticmaalayso qaabka **super**.`[magaca ficilka]`. Haddii aad **super**. ka tagto, ficilka ayaa laftiisa iswacaaya, taas oo lafteeda mar marka qaar loo baahanyahay. Haddii ficilku iskii uu is wacaayo waxa loo yaqaan arintaas *recursive call*. Bal kaxee borogaraamka intaad ka masaxdo **super**. laynka 38-aad. Aan ka gudbo.

13.3 Icon-ka daaqadda

Intaas kadib, waxaynu daaqadda u samaynayaa *icon* oo ah sawirka yar ee giftinka bidixda ee sare saaran sida ka muuqata sawirka 13.1, oo waxaynu badalaynaa icon-ka Java ee iskii u soo fuulay. Waxad nooca Nimi ku dartaa ficil `setIcon` sida ka muuqata tusaalaha 13.3.

```

1 package nimi.ui;
2
3 ...
4 import java.net.URL;
5 import javax.swing.ImageIcon;
6
7
8 public class Nimi extends JFrame {
9
10    public Nimi() {
11        ...
12        // baddal sawirka borogaraamka
13        setIcon();
14    }
15
16    ...
17
18    private void setIcon() {
19        URL iconUrl = Nimi.class.getResource("icon.png");
20        ImageIcon icon = new ImageIcon(iconUrl);
21        setIconImage(icon.getImage());
22    }
23 }

```

Tusaale 13.3: Badal icon-ka daaqadda

Aan sharaxaad ka bixiyo tusaalaha 13.3:

1. Marka hore soo **import**-garee noocyada `java.net.URL` iyo `javax.swing.ImageIcon` ee laymanka 4-5. Dhammaadka ku dar ficilka `setIcon`
2. Ficilka `setIcon` waxaynu wacaynaa laynka 13-aad
3. Ficilka `setIcon` wuxu laynka 19-aad wuxu isticmaayaa ficilka `getResource` ee nooca `Nimi`. Inta borogaraamku socdo nooca `Nimi` waxa matalaaya sifada `Nimi.class` (iyo ficilkaba), taas oo Java automatic

u samaynayso. Ficilkaas `getResource` wuxu raadinayaa faylka aad doonayso kadibna wuxu soo celinayaa waddada laga helaayo oo noqolaysa walax nooceedu yahay URL

4. Kadib waxaynu samaynaynaa walax cusub oo ah `ImageIcon`, taas oo aynu siinayno waddadii laga helaayey faylka `icon.png`
5. Sawirka `icon.png` waxad ka heli kartaa bogga <http://www.iconfinder.com> ama bogga buugga ee isbarjava.com ama adigaaba samaysan kara mid aad iska leedahay.

Intaas kadib, haddii aad mar kale kaxayso borogaraamka waa inuu isbadalo sawirka borogaraamku.

13.4 Menu-da daaqadda

Waynu dheeraynaynaa. Waxaynu ku xigsiinaynaa menu-da borogaraamka. Dhawr jeer oo hore ayaynu menu-da ka hadalay, markaas waad garanaysaa waxay tahay. Waxad si fiican uga arkaysaa sawirka 2.10. Soo eeg sawirkaas si aad maskaxdaada dib ugu soo xasuusiso qaybaha kala duwan ee daaqadda borogaraamku ka koobantahay. Ma aha sawirkaasi mid u gaar ah borogaraamka **Eclipse** oo kaliya, ee qaybahaas borogaraamyada badankooda waa laga helayaa. Menu-du sideedaba waxay ka koobantahay qaybo kala duwan sidaynu hoos ka arkidoonaba. Qaybahaas kala duwan waa inay dhammaantood ka muuqdaan qoraalka borogaraamka. Markaad u firsato borogaraamka qaybta qaabilsan menu-da, waxad arkaysaa inuu qoraalkaasi laba qaybood u qaybsamaayo. Qayb qaabilsan muuqaalka menu-da iyo qayb ku saabsan wixii dhici lahaa marka isticmaaluhu doorto menu-da mid ka mid ah. Qoraalka menu-da borogaraamkeenu wuxu ku bilaabmayaa tusaalaha 13.4:

```

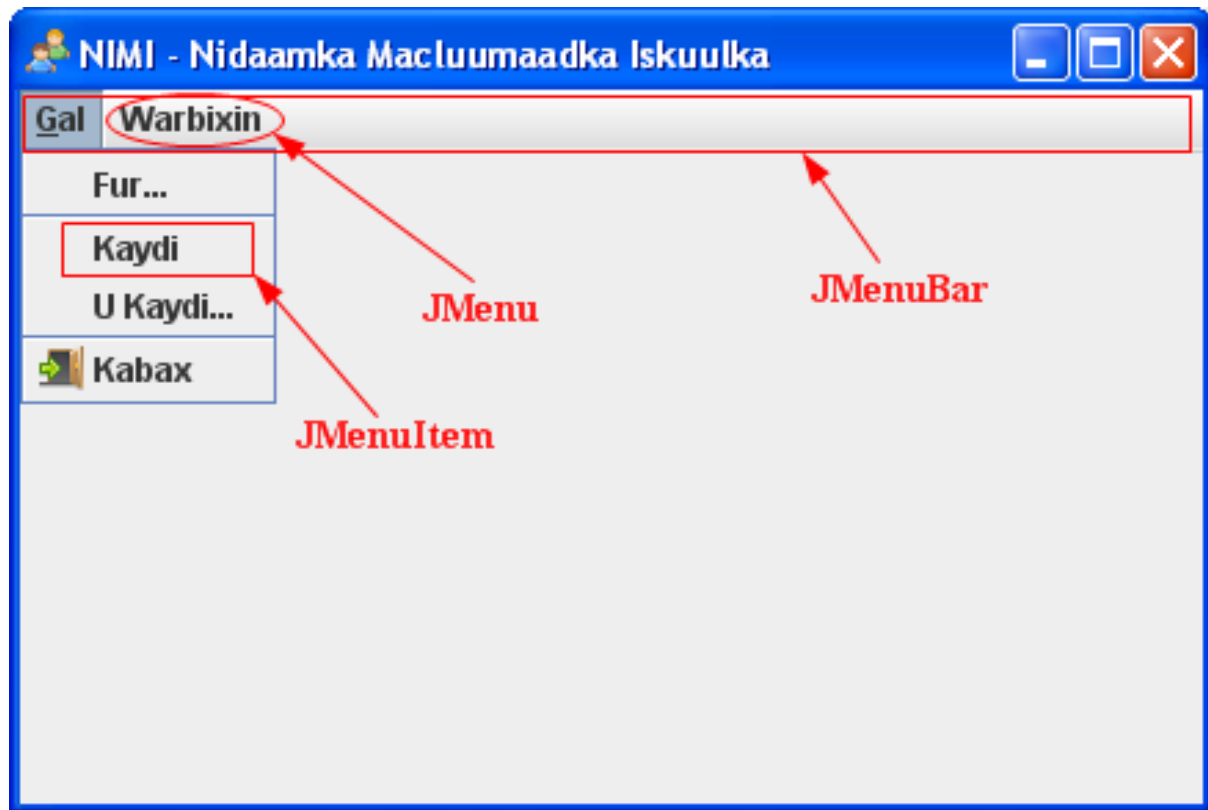
1 package nimi.ui;
2
3 ...
4 import javax.swing.JMenu;
5 import javax.swing.JMenuBar;
6 import javax.swing.JMenuItem;
7 import javax.swing.JSeparator;
8
9 public class Nimi extends JFrame {
10
11     public Nimi() {
12         ...
13         // samee menu-ga borogaamka nimi
14         createMenu();
15     }
16
17     ...
18
19     private void createMenu() {
20         JMenuBar menuBar = new JMenuBar();
21
22         JMenu fileMenu = addMenu(menuBar, "Gal");
23         JMenu reportMenu = addMenu(menuBar, "Warbixin");
24
25         JMenuItem open = addMenuItem(fileMenu, "Fur...");
26         fileMenu.add(new JSeparator()); // jiitin
27         JMenuItem save = addMenuItem(fileMenu, "Kaydi");
28         JMenuItem saveAs = addMenuItem(fileMenu, "U Kaydi...");
29         fileMenu.add(new JSeparator()); // jiitin
30         JMenuItem exit = addMenuItem(fileMenu, "Kabax");
31         URL iconUrl = Nimi.class.getResource("exit.png");
32         exit.setIcon(new ImageIcon(iconUrl));
33         JMenuItem fullReport = addMenuItem(reportMenu,
34             "Warbixin dhan...");
35         setJMenuBar(menuBar);
36     }
37
38     private JMenuItem addMenuItem(JMenu menu, String name) {
39         JMenuItem menuItem = new JMenuItem(name);
40         menu.add(menuItem);
41         return menuItem;
42     }
43
44     private JMenu addMenu(JMenuBar menuBar, String name) {
45         JMenu menu = new JMenu(name);
46         menuBar.add(menu);
47         return menu;
48     }

```

}

Tusaale 13.4: Menu-da borogaraamka nimi

Intaas kadib, intaynaan sharaxaadda u gudbin, markaad kaxayso borogaraamka waxa kuusoo baxaaya sawirka 13.2.



Sawirka 13.2: Menu-da borogaraamka iyo sharaxaad ku saabsan qaybaha kala duwan ee ay ka koobantahay. Waxan isticmaalidoonaa magacyada menubar (JMenuBar), menu (JMenu) iyo menuitem (JMenuItem). Waxa aan sawirka lagu calaamadin JSeparator oo ah xariijinta tusaale ahaan u dhaxaysa "Fur..." iyo "Kaydi".

Maadaama aad sawirka 13.2 aragtay way kuu fududahay fahamka tusaalaha sare ee 13.4. Waxaynu ka bixinaynaa sharaxaad kooban:

1. Marka hore xasuuso inaad soo **import**-garayso noocyada aad u baahantahay, kuwaas oo ku jira package-ka `javax.swing`, sida ka muuqata laymanka 4-7. Waxad u baahantahay noocyada `JMenu`, `JMenuBar`, `JMenuItem` iyo `JSeparator`
2. Samaynta menu-da waxaynu ugu talagalay ficil khaas ah oo ah `createMenu`, kaas oo aynu isticmaalayno laynka 14-aad oo aad kaga darayso ficilka dhisaha dhamaadkiisa. Ficilka `createMenu` wuxu ka bilaabmayaa laynka 19-aad
3. Ficilka `createMenu` wuxu isticmaalayaa laba ficil oo kalkaaliyeyaal ah oo kala ah `addMenu` iyo `addMenuItem`, kuwaas oo aad ka arkayso laymanka 38-48. Ficilka `addMenu` wuxu menubar-ka ku darayaa menu cusub, taas oo aad magaceeda siinayso, sidoo kale ficilka `addMenuItem` wuxu menu-da aad siisay hoosteeda ku darayaa menuitem cusub, kaas oo aad magaciisa siinayso ficilka
4. Haddii aad u fiirsato menu-da borogaraamka, waxad arkaysaa in kaliya menuitem-yadu wax qabanayaan oo menubar-ka, menu-da iyo separator-ka kaliya loogutalagay in lagu kala habeeyo menuitem-yada
5. Haddii aad gujiso mid kamid ah menuitem-yada "Fur..." ama "Kaydi", waxad arkaysaa inaanay waxba qabanaynin, maadaama aynaan wali la implement-garayn (ama la qorin) logic-kii ka dambeeyey ama ay qaban lahaayeen marka la gujiyo
6. Ficilka `createMenu` wuu fududyahay. Marka hore wuxu diyaarinayaa `JMenuBar` ee laynka 20-aad, kaas oo uu marka hore ku daraayo laba `JMenu`, kadibna midkoobda hoosta ayuu kaga darayaa

JMenuItem-yadii u gaarka ahaa. Ugu dambayntii wuxuu isticmaalayaa ficilka setJMenuBar ee laynka 35, kaas oo keenaaya in menubar-ka lagu dhajiso daaqadda.

Intaynaan hore u socon waxad arkaysaa in qayb kasta oo daaqadda ka mid ah aynu u samaynay ficil u gaar oo qaybtaas diyaariya.

13.5 Ka bax ama Exit

Borogaraamyada daaqadda leh inta badan waxa jira laba siyaabood oo loo xidho ama looga baxo. Sida koobaad waa inaad giftinka midigta ee sare calaamadda X-ka ah ee ku taalla aad gujiso. Sida labaad waa inaad inta badan menu-da Gal-ka ama File menuitem-ka ugu hooseeya oo ah exit/quit/kabax doorato. Laakiin labadoodaba ujeedadoodu waa isku mid, waana inaad ka baxdo borogaraamka. Qaabka aad uga baxayso borogaraamka waa inuu noqdo isku mid ama X-da guji ama menuitem-ka dooro (inkastoo tusaalahan aaney isku si u shaqaynaynin, maadaama aynu fududaynay). X-da markaad gujiso waa inoo diyaar oo daaqadda way is xidhaysaa, taasna waxa keenaysa setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE), oo aad hore u soo aragtay. Halkan waxaynu qoraynaa ficilkii ka dambeeyey menuitem-ka Kabax ee ku hoos jira Gal-ka. Waxaynu rabnaa inuu borogaraamku is xidho marka uu isticmaaluhu gujiyo "Kabax". Waxaanad ka arkaysaa tusaalaha gaaban ee 13.5.

```

1 package nimi.ui;
2
3 ...
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6
7 public class Nimi extends JFrame {
8     ...
9     private void createMenu() {
10         ...
11         JMenuItem exit = addMenuItem(fileMenu, "Kabax");
12         addExitListener(exit);
13         ...
14     }
15
16     ...
17
18     private void addExitListener(JMenuItem exit) {
19         exit.addActionListener(new ActionListener() {
20             ...
21             public void actionPerformed(ActionEvent e) {
22                 System.exit(0);
23             }
24         });
25     }
26 }

```

Tusaale 13.5: Ka bax ama xidh

Aan sharaxaad ka bixiyo tusaalaha 13.5:

1. Marka hore ficilka addExitListener ee laymanka 18-25 ku dar dhammaadka nooca Nimi. Ficilkani wuxuu u baahanyahay sharaxaad dheeraad ah ee haka warwarin, arimo badan ayaa kuu iftiimi doono laba cusub kadib
2. Kadib waxad ficilka createMenu ka raadisaa laynka 11-aad, halkaas oo aynu ku samaynay menuitem-ka "Kabax"
3. Waxad hoostiisa ku qortaa laynka xiga ee 12, halkaas oo aynu wacayno ama isticmaalayno ficilka addExitListener.

Waxa jira waxyaalo inta badan la qabto, marka isticmaaluhu damco inuu xidho ama ka baxo borogaraamka. Waxaynu tusaale u soo qaadan karnaa haddii uu isticmaaluhu qoraalka ama faylka badalo in la waydiiyo inuu rabo inuu kaydiyo iyo kale (waxad ku ogaan kartaa intaad tusaale ahaan shiddo borogaraamka Notepad, kadibna wax ku qorto, kadibna aad gujiso xidhitaanka). Arintani waa arin u baahan inaad garwaaqsato.

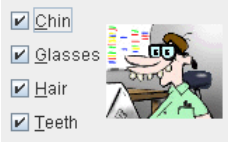
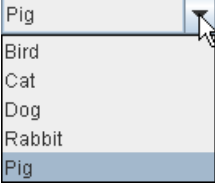
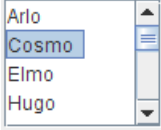
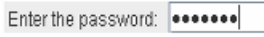
Cutub 14

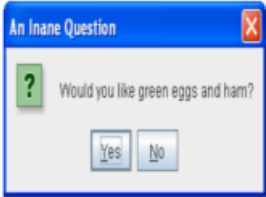
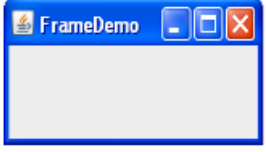
Unugyada Swing

Shay kasta oo aad samaynayso waa inay jiraan waxyaalo yaryar oo aad ka samayn karto. Tusaale ahaan haddii aad kumbuyuutar doonayso inaad samayso waxyaalaha yaryar ee aad u baahantahay waxa kamid ah jiir, keyboard ama badhannadii, shaashaddii, CPU-gii, iwm. Haddii kuwaas hoos looga sii dago oo la eego waxyaalaha ugu yaryar ee kumbuyuutarka aad ka samayn karto waxa kamid ah electronic-yada yaryar sida transistors-ka. Sidaas awgeed markaad samaynayso wajiga ama GUI-ga borogaraamka Java waxad u baahantahay waxyaalo aad ka samayn karto. Markaad Swing adeegsanayso, waxyaalahaas waxa loo yaqaan *Swing Controls*, *Swing Widgets* ama *Swing Components*. Miiska 14.1 waxaad ka arkaysaa qaar kamid¹. Waxa muhiim ah inaad ogaato inay controls-ka Swing ka badanyihiin inta ka muuqata miiska 14.1, laakiin kaliya intan aynu tusaale u soo qaadanay.

Sawir	Magaca nooca	Sharaxaad
	JLabel	Waxay mataleysa qoraalka aan la badali karin. Inta badan waa qoraalka ka horeeya meesha wax lagu qorikaro ama la dooran karo.
	JButton	Wuxu matalayaa badhanka la gujinkaro. Marka la gujiyo waa inay wax dhacaan. Waa inaad ku dhajiso waxa dhacaaya marka la gujiyo adiga oo adeegsanaaya ficilka addActionListener. Waxa kale oo aad badali kartaa qoraalka badhanka.
	JRadioButton	Inta badan waa guruub badhanno ah oo ka kooban mid ama in ka badan, oo mid kaliya laga dooran karo. Wax badan kagama duwana badhanka hore marka la eego qaabka ay u shaqaynayaan mid-kiiba. Markaad mid doorato waa inay wax dhacaan, sida sawirka oo is badala. Muuqaal ahaan waa goobo.

¹Dhammaantood waxad ka helayaa adarayska: <http://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

	JCheckBox	Guruub badhanno ah oo ka kooban mid ama dhawr, oo mid ama in ka badan laga dooran karo. Muuqaal ahaan wuu ka duwanyahay nooca hore.
	JComboBox	Guruub doorashooyin ah oo mid laga dooran karo. Wuxu ku fiicanyahay dhul ama space wayn ma qaadayaano. Falaadha ayaa ku tusaysa waxa uu caloosha ku haysto.
	JList	Guruub doorashooyin ah oo mid ama in ka badan laga dooran karo. Haddii doorashooyinku bataan, waxay giftinka midigta ku yeelanaysaa meel kor iyo hoos loogu qaado.
	JTextField	Waa meel aad wax ku qorikarto. Inta badan waxa lagu daraa inay xaqiijiso ama ansixiso qiimaha lagu qoray inuu yahay wixii la rabay, tusaale ahaan haddii tiro loo baahanyahay.
	JPasswordField	Waxa loogu talagalay qiimeyaasha sirta ah sida password-ka, kuwaas oo la rabo inaan la arkin markaad qorayso.
	JSpinner	Waxay soo bandhigaysaa doorashooyin aad falaadhaha kor iyo hoos ugu soconkarto si aad uga doorata kaad rabto. Waxay ku dhawdahay JComboBox-ka.
	JMenu	Waxa loo adeegsadaa menu-da borogaraamka, waxay caloosha ku haysataa JMenuItem-yo iyo JSeparator-yo, taasi waxay saamaxaysaa in markaad tusaale ahaan doorato menu-da <i>File</i>, in ay soo boodaan doorashooyin aad mid ka dooran karto .

	JTable	Waa miis kuu soo bandhigaaya macluumaad ka kooban safaf. Inta badan safkaad rabto ayaad wax ka badali kartaa. Tusaale ahaan miiska ardayda.
	JTabbedPane	Caleemaha isbaalyaalka ah ee mid waliba wax u gaar ah caloosha ku haysato, oo aad arkayso kolba midka aad doorato waxay caloosha ku haysato.
	JDialog	Daaqadaha yaryar ee inta badan soo booda si ay macluumaad ama cilad u tusaan isticmaalaha. Waxa kale oo isticmaalaha laga codsankaraa inuu tusaale ahaan hubo inuu arin fuliyo iyo in kale
	JFrame	Daaqadda ugu saraysa ee borogaraamka. Wax diidaaya in borogaraamku yeesho daaqado badan ma jiro.

Miiska 14.1: Qaar ka mid ah kuwa ugu muhiimsan unugyada Swing. Dhammaantood magacoodu wuxu ku bilaabmaa xarafka J-da, oo aad ku garanayso unugyada Swing.

Control-lada miiska 14.1 qaarkood waynu soo isticmaalay. Kuwa aynu soo isticmaalay waxa kamid ah JFrame (superclass-ka nooca Nimi), JMenu, JMenuItem iyo JSeparator. Qaarkoodna waxaynu ku isticmaalidoonaa qaybaha buuggan soo socda. Qaarkoodna adiga ayaa iskaa u baran oo buuggan kusoo qaadanmayno. Guud ahaan unugyada Swing waxa loo qaybiyaa laba: *Top-level components* iyo components. Dhammaantood waa wada components sidaan soo sheegayba, laakiin saddex kamid ah ayaa loo yaqaan *top-level components*: JFrame, JDialog iyo JApplet. Saddexdani waa kuwa kaliya ee noqon kara daaqad sare, laakiin dhammaan kuwa kale waxay gali karaan kaliya caloosha daaqadahaas ama caloosha unugyada kale. Waxa kale oo Swing kuu saamaxaysaa inaad samaysato unugyo aad adigu leedahay, haddii aanay sidooda kugu anfacaynin kuwa la hayaa.

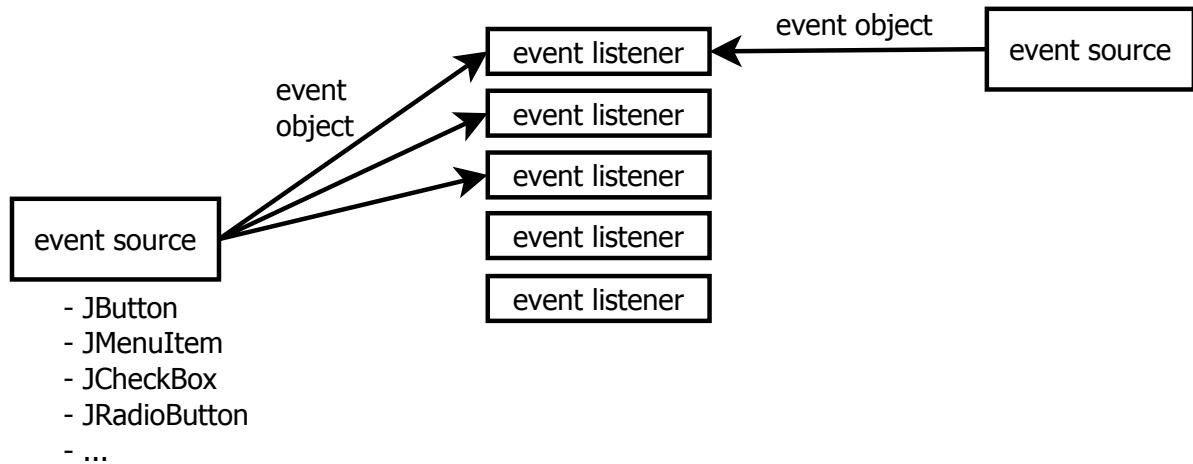
Unugyada noocan oo kale ah kuma koobna kaliya luuqadda Java iyo Swing ee waxad ka helayaa luuqadaha kale gaar ahaan kuwa lagu samaynkaro UI-ga borogaraamyada. Waxa kale aad ka helayaa tusaale ahaan luuqadda HTML.

Cutub 15

Dhagaysteyaasha dhacdooyinka

Unugyada Swing waxay leeyihiin noocyo badan. Qaar la dooran karo, qaar wax lagu qorikaro, qaar la gujinkaro iyo qaar kale. Hadaba, sidee ayaa laysugu xidhayaa unugyada Swing laftooda iyo borogaraamka intiisa kale? Arintan waxa loo yaqaan *dhagaysteyaasha dhacdooyinka* ama *event listeners*.

Waxan isku dayayaa inaan sharaxaad ka bixiyo arintaas. Arintani waxay ka koobantahay saddex qodob: dhagayste, dhacdo iyo ishii dhacdada (soo dirihii ay dhacdadu ama fariintu ka timid). Sawirka 15.1 ayaad ka arkaysaa siday saddexdaas arimood isugu xidhanyihiin.



Sawirka 15.1: Dhagaysteyaasha dhacdooyinka. Event source waa ishii dhacdada, event listener waa dhagaystihii, event object waa dhacdadii ama fariintii.

Dhawr arimood ayaan ka sheegayaa sawirka 15.1, intaynaan soo qaadan borogaraam kooban oo arintan ku saabsan.

1. Marka hore dhammaantood event source, event object iyo event listener waa walxo. Qaabkani wuxu saamaxayaa in walxaha laysku xidhiidhiyo oo ay fariimo isu diraan
2. Event source ama isha dhacdada sida sawirka ka muuqata waxa noqon kara unugyada Swing sida JButton, JMenuItem, iwm
3. Event object waa walax mataleysa fariinta ama dhacdada laysu diraayo sida gujinta, dulmarinta jiirka, wax ku qorista, doorashada, iwm
4. Event listener ama dhagaystaha dhacdada wuxu noqon karaa walxo borogaraamkaaga ku jira oo aad ugu talogashay inay dhageystaan unugyada Swing
5. Waxa kale oo aad sawirka ka arkaysaa in halkii event source uu yeelan karo eber, mid ama in ka badan oo dhagayste. Waxa kale oo suurtoagal ah in dhagayste uu dhageysto in ka badan hal event source ama ilo dhacdo
6. Qaabkani waa sida laysugu xidhiidhiyo qaybaha kala duwan ee borogaraamka Swing lagu sameeyey. Sidaas awgeed waa muhiim inaad fahanto.

Si ay arintani inoogusii furanto waxaynu samaynaynaa tijaabo yar. Waxaynu samaynaynaa borog-araam yar oo ka kooban hal badhan oo kaliya iyo qoraal tiro ku tusaaya. Mar kasta oo badhankaas la gujiyo, waxaynu mid ku daraynaa tirada qoraalka. borogaraamkaas yar waxad ka arkaysaa 15.1.

```

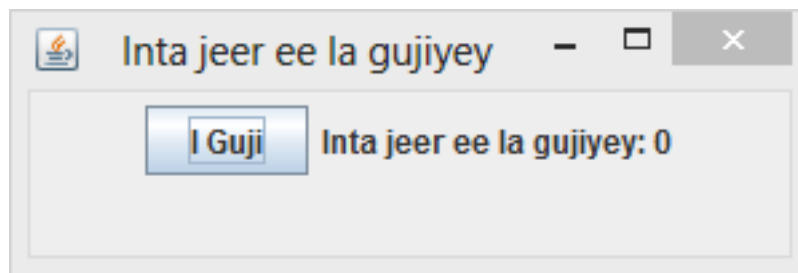
package nimi;
2
import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;

6 import javax.swing.JFrame;
import javax.swing.JButton;
8 import javax.swing.JLabel;
import javax.swing.JPanel;
10
import javax.swing.SwingUtilities;
12
public class IsbarJava extends JFrame {
14     private int tiro = 0;
    private JLabel label;
16     protected void createAndShow() {
        setTitle("Inta jeer ee la gujiyey");
18         JButton button = new JButton("I Guji");
        button.addActionListener(new ActionListener() {
20
            public void actionPerformed(ActionEvent e) {
22                 tiro++;
                label.setText("Inta jeer ee la gujiyey: " + tiro);
24             }
        });
26         label = new JLabel("Inta jeer ee la gujiyey: " + tiro);
        JPanel panel = new JPanel();
28         panel.add(button);
        panel.add(label);
30         setContentPane(panel);
        setSize(300, 100);
32         setVisible(true);
    }
34     public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
36
            public void run() {
38                 new IsbarJava().createAndShow();
            }
40         });
    }
42 }

```

Tusaale 15.1: borogaraam yar oo ku tusaaya dhagaysiga dhacdooyinka.

Haddii aad kaxayso borogaraamka sare ee 15.1 waxa kuusoo baxaysa daaqadda sawirka 15.2.



Sawirka 15.2: Mar kasta oo aad gujiso badhanka waxa kordhaaya tirada jeer ee la gujiyey (Sawirkan waa Windows 8)

Waxa jira qaab kale oo inyar ka duwan borogaraamka sare ee 15.1, oo isna aad loo isticmaalo, waxaanad ka arkaysaa tusaalaha 15.2.

```

package nimi;
2
...
4
public class IsbarJava extends JFrame implements ActionListener {
6     private int tiro = 0;
    private JLabel label;

```

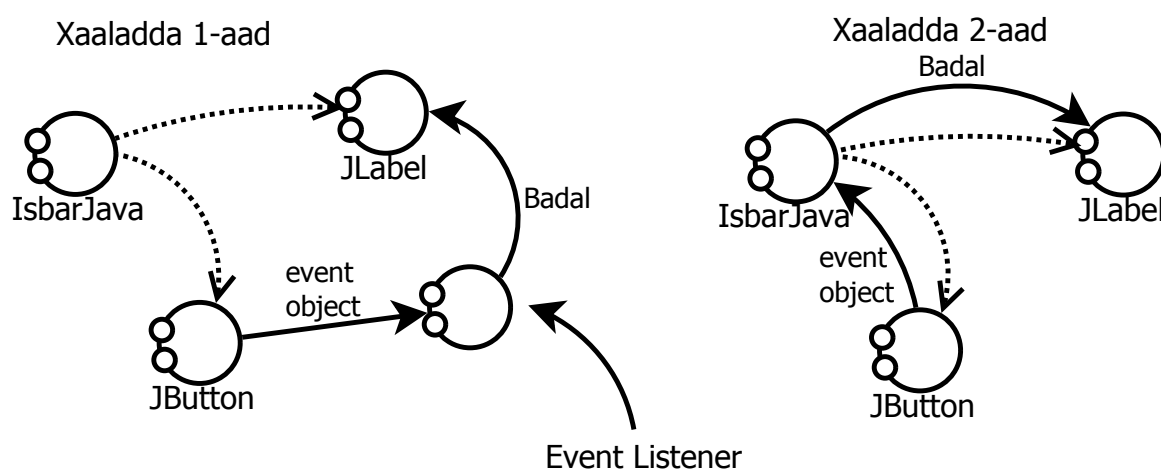
```

8   protected void createAndShow() {
9       ...
10      button.addActionListener(this);
11      ...
12  }
13  ...
14  public void actionPerformed(ActionEvent e) {
15      tiro++;
16      label.setText("Inta jeer ee la gujiyey: " + tiro);
17  }
18  }

```

Tusaale 15.2: Dhagaysiga dhacdooyinka oo si kale loo qoray. Sidan lafteeda aad ayaa loo isticmaalaa. Waxan soo qaatay kaliya meelaha ay wax iska badaleen

Labadan siyaaboodba waa sax oo kii aad rabto ayaad isticmaali kartaa. Waxan iskudayey inaan sawirka ka bixiyo labadan siyaabood waxay ku kala duwanyihiin, waxaanad ka arkaysaa sawirka 15.3.



Sawirka 15.3: Labada borogaraam ee 15.1 iyo 15.2 waxay ku kala duwanyihiin marka la eego dhanka walxaha. Falaadha goo'go'an waxay ku tusaysaa in walaxda IsbarJava caloosha ku haysato labada walxood ee JLabel iyo JButton. Xaaladda koobaad walaxda IsbarJava kama qayb qaadanayso dhagaysiga dhacdooyinka JButton, laakiin xaaladda labaad waa iyada walaxda dhageysanaysa badhanka, marka la gujiyo kadibna badalaysa qoraalka JLabel. Walax kasta nooceeda waan ku hoos qoray, laakiin xaaladda koobaad walaxda dhagaysenaysa badhanka, marka la gujiyo kadibna badalaysa qoraalka nooceeda waxan saaray su'aal, maadaama aan la garanaynin magaca nooceeda. Waxan ku sharaxayaa arintan hoos.

Waxa meesha ugu muhiimsan waa ficilka addActionListener ee nooca JButton. Ficilkaas waxa laga helayaa sidoo kale noocyada JMenuItem, JCheckBox iyo JRadioButton, waxaanu saamaxayaa in walxaha noocyadaas laga sameeyey laga diiwaangeliyo dhagayste (ama dhagaysteyaal). Dhagaystehaas oo la wacaayo markasta oo badhanka la gujiyo ama menuitem-ka la doorto. Ficilkaas isticmaalkiisa waxad ka arkaysaa laynka 19 ee borogaraamka sare ee 15.1 iyo laynka 10 ee tusaalaha 15.2

15.1 Nooc-hoosaadka iyo nooc-hoosaad aan magaca lahayn

Luuqadda Java waxay kuu saamaxaysaa inaad samayso walax aan nooceeda la garanaynin. Waxa kale oo ay kuu saamaxaysaa in noocu noocyada kale caloosha ku haysto. Waxan isku dayayaa inaan labadan arimood sharaxaad ka bixiyo. Nooca nooc kale ku hoos jira waxa loo yaqaan *inner class*. Aan soo qaato tusaale yar:

```

package nimi;

2  public class NoocaSare {
3      private NoocaHoose hoose;

4      public NoocaSare() {
5          this.hoose = new NoocaHoose();
6      }

7      public int isugee(int tiro1, int tiro2) {

```

```

    return this.hoose.sum(tiro1, tiro2);
12 }
    public class NoocaHoose {
14     public int sum(int i, int j) {
        return i+j;
16     }
    }
18
    public static void main(String[] args) {
20     NoocaSare sare = new NoocaSare();
        int jawaab = sare.isugee(123, 456);
22     System.out.println("Jawaabtu waa: " + jawaab);
    }
24 }

```

Tusaale 15.3: Nooc nooc kale hoose ama caloosha ku haysta (Inner Class)

Haddii aad kaxayso tusaalaha 15.3 (talaabada RunJava), waxa kuusoo baxaysa sidan:

Jawaabtu waa: 579

Laynka 11-aad ee 15.3 waxad arkaysaa inaynu adeegsanayno nooca hoose ficilkiisa sum (isugee).

Sidaad tusaalaha 15.3 ka arkayso luuqadda Java waxay kuu saamaxaysaa inaad nooca hoostiisa ama calooshiisa ku samayso noocyo kale, kuwaas oo uu nooca sare adeegsankaro. Noocyada nooca kale calooshiisa ku jira waxa loo yaqaan Inner Class, sidaan soo sheegayba. Waa maxay *Anonymous Inner class* hadaba? Ereyga af-ingiriisiga ee Anonymous, wuxu ka dhiganyahay nooc aan magac lahayn. Haddii aad eegto tusaalaha 15.3, waxad arkaysaa in nooca hoose (inner) magaciisu yahay NoocaHoose, sidaas awgeed wuxu leeyahay magac. Waxaynu ku xigsiinaynaa haddaba tusaale ku tusaysa isticmaalka *anonymous inner class* ama nooc-hoosaad aan magaca lahayn.

```

package nimi;
2
import nimi.person.Named;
4
public class TusaaleAnonymous {
6
    private Named sheegMagacayga;
8
    public TusaaleAnonymous(Named magacLeh) {
10     this.sheegMagacayga = magacLeh;
    }
12
    public String sheegMagacaaga() {
14     return sheegMagacayga.getName();
    }
16
    public static void main(String[] args) {
18     TusaaleAnonymous tusaale = new TusaaleAnonymous(new Named() {
        public String getName() {
20     return "SamaysoMagacAdigaKuuGaarAh";
        }
    });
22
    System.out.println("tusaale magaciisu waa: " +
24     tusaale.sheegMagacaaga());
    }
26 }

```

Tusaale 15.4: Istimmaalka nooc hoose oo aan magac lahayn.

Haddii aad kaxayso tusaalaha 15.4, waxa kuusoo baxaysa sidan:

tusaale magaciisu waa: SamaysoMagacAdigaKuuGaarAh

Inta ugu muhiimsan ee waa laymanka 18-22 ee 15.4. Named waa interface oo hore aynu u soo aragnay, laakiin maynaan arag interface sidan oo kale loo isticmaalo hore. Qaybta ...

```

1 new Named() {
    public String getName() {
3     return "SamaysoMagacAdigaKuuGaarAh";
    }
5 }

```

ee loo gudbiyey ficilka dhisaha ee nooca TusaaleAnonymous waxay keenaysaa inuu dhasho walax cusub, kaas oo laga sameeyey nooc aan magac lahayn, kaas oo implement-garaynaaya interface Named. Kadib walaxdaas waxaynu u gudbinaynaa ficilka dhisaha ee nooca TusaaleAnonymous, kaas oo ku kaydinaaya sifadiisa sheegMagacayga sida ka muuqata laynka 10-aad ee 15.4. Waxad arkaysaa in sidan oo kale walax cusub loo sameeyey laymanka 19-24 ee 13.5, kadibna loo gudbiyey ficilka addActionListener ee walaxda exit oo noocedu yahay JMenuItem. Aan intaas kaga gudbo.

Intaynaan hore u socon tusaale ahaan badal ficilka addExitListener ee nooca Nimi oo ka dhig sidan oo kale, kadib eeg waxa kuusoo baxa (xasuuso inaad soo **import**-garayso nooca javax.swing.JOptionPane):

```

...
2  private void addExitListener(JMenuItem exit) {
    exit.addActionListener(new ActionListener() {
4
        public void actionPerformed(ActionEvent e) {
6            int ret = JOptionPane.showOptionDialog(Nimi.this,
              "Ma hubtaa?", "Kabax", JOptionPane.YES_NO_OPTION,
8              JOptionPane.QUESTION_MESSAGE, null, new String[] {
                "Haa", "Maya" }, "Maya");
10           if (ret == JOptionPane.OK_OPTION) {
              System.exit(0);
12           }
        }
14     });
    }
16  ...

```


Cutub 16

Caloosha Daaqadda Borogaraamka

16.1 Ku xidhka Model-ka

Model-ka waxaynu kaga soo hadalay qaybtii hore ee buuggan. Sideedaba wajiga borogaraamku wuxu isticmaalaha tusayaa model-ka borogaraamka, waana araggii model-ka. Sidaas awgeed waxaynu samaynaynaa model tusaale ah oo aad u fudud, taas oo marka borogaraamka la shido ama istaadho aynu isticmaalaha tusno, waxaanad ka arkaysaa tusaalaha 16.1 ee hoose. Sida caadiga ah inta badan marka borogaraamka la shido wuxu isticmaalaha tusayaa model madhan ama model-kii iyo faylashii uu hore u sameeyey. Model-kaas tijaabada ah waxa samaynaaya nooca NimiModel, gaar ahaan ficilkiisa createSampleModel, oo aad halkan qoraalkiisa ka arkayso. Kudar ficilkaas nooca NimiModel.

```
package nimi;
2
...
4
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
8 import nimi.course.IllegalCourseNumberException;

10 public class NimiModel implements Serializable {
    ...
12     public static NimiModel createSampleModel() {
        NimiModel model = new NimiModel();
14         // Ardayda
        List<Student> students = model.getStudents();
16         students.add(new Student(1, "Cabdilaahi Maxamed Muuse"));
        students.add(new Student(2, "Khadar Cabdilaahi Muuse"));
18
        // Koorsooyinka
20         List<Course> courses = model.getCourses();
        try {
22             courses.add(new Course("Maths Introductory Course",
                                    "MAT-01", "Maths"));
24             courses.add(new Course("Functions and equations",
                                    "MAT-02", "Maths"));
26             courses
                .add(new Course("Polynomials", "MAT-03", "Maths"));
28             courses.add(new Course("Geometry", "MAT-04", "Maths"));
                courses.add(new Course("Probability and Statistics",
30                                     "MAT-05", "Maths"));
                courses.add(new Course("Writing Course", "ENG-02",
32                                     "Languages"));
                courses.add(new Course("Af-soomaali 1", "SOM-01",
34                                     "Languages"));
        } catch (IllegalCourseNumberException e) {
36             // Ma aha in halkan lasoo gaadho, waayo
            // dhammaan lambarada koorsooyinku waa sidii laga rabay
38             e.printStackTrace();
        }

40
        // Fadhiyada
42         List<CourseSession> sessions = model.getSessions();
        CourseSession session = new CourseSession(courses.get(0),
44             createDate(2011, 1, 1), 1);
        sessions.add(session);
46
```

```

48    // Xaadirinta ardayga
    sessions.get(0).enroll(students.get(0));
    return model;
50 }

52 /**
53  * Ficil kalkaaliye ah oo Date ku siinaaya
54  */
    public static Date createDate(int year, int month, int day) {
56         Calendar calendar = new GregorianCalendar();
        calendar.clear();

58         calendar.set(Calendar.YEAR, year);
60         calendar.set(Calendar.MONTH, month - 1);
        calendar.set(Calendar.DAY_OF_MONTH, day);
62         return calendar.getTime();
    }
64 }

```

Tusaale 16.1: Model-ka tusaalaha ah ee aynu isticmaalaha tusayno marka borogaraamka la shido

Haddii aan sharaxaad kooban ka bixiyo tusaalaha 16.1:

1. Waxa soo kordhay laba ficil oo cusub oo kala ah `createSampleModel` iyo `createDate`. Ficilka hore ee `createSampleModel` wuxu samaynayaa model cusub, sida ka muuqata laynka 13-aad, waxaanu model-kaas ku darayaa laba arday (laymanka 16-17), todoba koorso (laymanka 22-34) iyo hal fadhi (laymanka 42-45). Waxa kale oo uu ardayda midkood (ka hore) ka diiwaangelinayaa halkaas fadhi (laynka 48)
2. Ficilka `createDate`, waxaynu isticmaalaynaa markaynu samaynayno fadhiga, waxaanu qaadanayaa sanadka, bisha iyo maalinta oo dhammaantood `int` noocoodu yahay, kadibna wuxu soo celinayaa walax cusub oo noocedu yahay `Date`, taas oo mataleysa taariikhdaas.

Model-kii tusaalaha ahaa waa inoo diyaar. Waxa kale oo aynu nooca Nimi ku xidhaynaa model-kaas tusaalaha ah inuu isticmaalo. Laymanka kusoo kordhay noocaas waxad ka arkaysaa tusaalaha 16.2.

```

package nimi.ui;
2
...
4
import nimi.NimiModel;
6
public class Nimi extends JFrame {
8     private NimiModel model;

10    public Nimi() {
        ...
12        // model-ka tusaalaha ah ku xidh muuqaalka
        setSampleModel();
14    }

16    ...

18    private void setSampleModel() {
        this.model = NimiModel.createSampleModel();
20    }
}

```

Tusaale 16.2: Model-ka tusaalaha ah ee aynu isticmaalaha tusayno marka borogaraamka la shido oo lagu xidhay nooca Nimi

Nimi wuxu isticmaalayaa model tusaalaha ah ee aynu kor ku samaynay, sidaad ka arkaysaa tusaalaha 16.2. Kaliya wuxu sifadiisa model ku kaydinayaa model-kaas tijaabada ah ee uu soo celiyey ficilka static-ka ah ee `Nimi.createSampleModel()`.

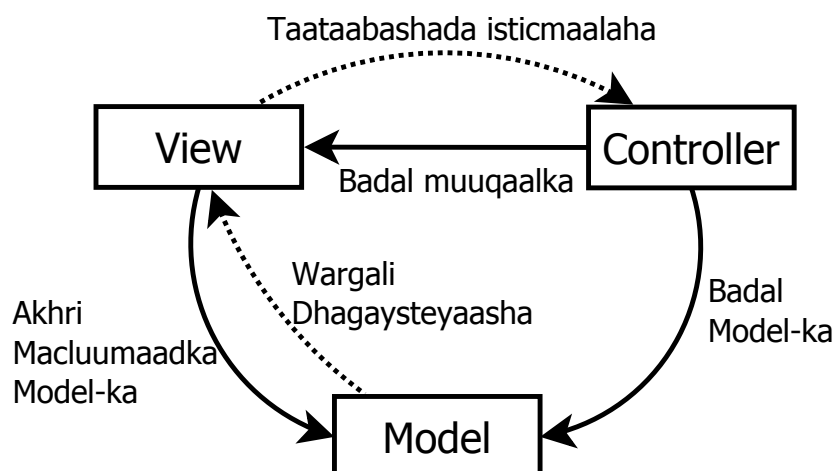
16.2 Model-View-Controller (MVC)

Waxa jira qaab caan ah oo inta badan la adeegsado marka la dhisaayo borogaraamyada wajiga leh, kaas oo loo yaqaano *Model-View-Controller (MVC)*. MVC waxay soo ifbaxday 1980-kii, waxaana wakhtigaas lagu adeegsaday luuqadda Smalltalk-80. Maadaama wakhtigaas xilli dheer laga joogo maanta, waxa jira qaabab badan oo laga soo dhiraandhiriyeey qaabkaas caanka ah oo maanta la isticmaalo. Ereyada

inta badan qaabkan loo isticmaalo waxa kamid *Architecture* ama *Pattern*, maadaama ay tahay qaab la raacaayo. Waxad arkaysaa in qaabkaas ka muuqdo borogaraamyada maanta la sameeyo. **MVC** waxay ka koobantahay saddex qaybood oo kala ah *Model*, *View* iyo *Controller*. Saddexdaas qaybood oo ah qaybaha qoraalka borogaraamka loo qaybinaayo. Ujeedada ugu muhiimsan ee ka dambaysa waa in borogaraamka qaybihiisa kala duwan xilka loo kala qaybiyo iyo in la fududeeyo fahamka borogaraamka guud ahaan. Waxa jira faa'idooyin kale oo qaabkani leeyahay, ee bal horta aynu sharaxaad ka bixiyo saddexdan qaybood waxay kala yihiin:

1. **Model** - Waa xogta borogaraamka (ardayda, macalimiinta, koorsooyinka, iwm). Waxay haysaa xaaladda uu borogaraamku maraayo. Waxa kale oo ay maamulaysaa wax ka badalka xogtaas. Xogtaasi tusaale ahaan waxay ku diiwaangashanaan kartaa **database**
2. **View** - Waa aragga ama muuqaalka borogaraamka
3. **Controller** - Waxay isku xidhaysaa labadaas qaybood ee sare. Waxay qaabilaysaa codsiga isticmaalaha, kadibna waxay badalaysaa aragga iyo xogta.

Sawirka 16.1 ayaad ka arkaysaa qaybaha kala duwan ee ay ka koobantahay **MVC** iyo sida ay isugu xidhanyihiin.



Sawirka 16.1: Sawir ku tusaaya qaybaha ay ka koobantahay MVC iyo sida ay isula xidhiidhayaan. Dhaqdhaqaaqa isticmaalaha sida gujinta badhanada waxa marka hore qaabilaaya View-ga. View-gu wuxu u gudbinayaa Controller-ka, kadibna Controller-ku waxay amraysaa oo badalaysaa Model-ka. View-gu wuxu iska diiwaangelinayaa Model-ka. Model-ku markasta oo wax iska badalaan waxay wargalinaysaa dhammaan wixii View iska diiwaangelisay. Model-ka waxa iska diiwaangelinkara dhagaysteyaalka kale oo aan ahayn View.

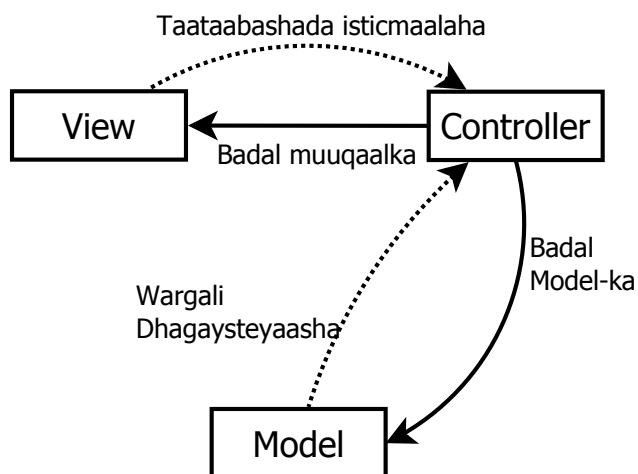
Waxan ku xigsiinayaa faa'iidooyinka ay **MVC** leedahay:

1. In aragga borogaraamka la badali karo ama borogaraamka loo samaynkaro muuqaallo kala duwan
2. In tusaale ahaan **model**-ka gooni loo adeegsankaro, haddii aan loo baahnayn muuqaal
3. In tijaabinta borogaraamku sidii hore ka fududaato, maadaama qayb walba goonideeda loo tiyaabinkaro

4. Shaqada oo la kala qaybsado, maadaama borogaraamka intii isku wax qabanaysaba gooni laysugu xidhay.

Sidaan soo sheegayba **MVC**-dii hore sideeda waa lagu isticmaalaa wali, laakiin waxa jira naqshado cusub oo laga soo dhiraandhiriyeey. Naqshadahaasi inta badan waxba kamay badalin qaybaha ay ka koobantahay **MVC** ee waxay inta badan badaleen siday isugu xidhanyihiin qaybahaasi. Inta badan waxa lagu dadaalaa in xidhiidha aan loo baahnayn la yareeyo. Fikirkan **MVC** waxa loo adeegsan karaa dhisidda borogaraamyada ama dhisidda unug ama waax ka mid ah borogaraamka wayn.

Naqshadda aynu buuggan ku adeegsandoono waxad ka arkaysaa sawirka 16.2. Kaliya waxay kaga duwantahay **MVC**-da sawirka 16.1, waa xidhiidhka u dhaxeeya qaybaha kala duwan.



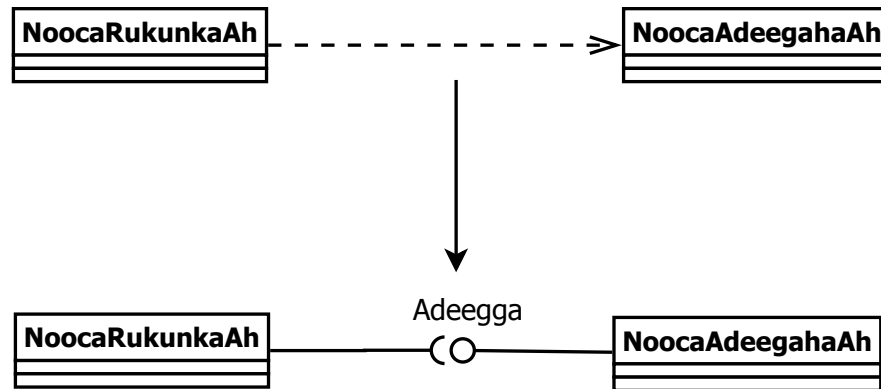
Sawirka 16.2: Naqshadan oo aynu buuggan ku adeegsandoono kaliya waxa la masaxay xidhiidhkii u dhaxeeyey View-ga iyo Model-ka. Model-ku imika marka ay is badasho waxay wargalinaysaa tusaale ahaan Controller-ka, haddii ay iska diiwaangeliso. Laakiin wargalinta buuggan kusoo qaadanmayno. Naqshadan waxa loo yaqaan *Passive View*.

Intaynaan ka gudbin qaybtan waxa muhiim ah inaad fahamto ujeedada ka dambaysa naqshadan **MVC** iyo faa'iidooyinka ay leedahay. Mar walba waxa lagu dadaalayaa in la khafiifiyo aragga ama View-ga borogaraamka.

16.3 Khafiifinta xidhiidhka noocyada

Intaynaan hore u socon waxan halkan kusoo qaadanayaa arin kale oo muhiim ah, taas oo aan ku magacaabay khafiifinta xidhiidhka noocyada. Sideeda arimaha ugu muhiimsan ee **interfaces**-yada loo isticmaalo waa khafiifinta xidhiidhka u dhexeeya noocyada iyo in la kala saaro adeegga loo baahanyahay noocuu yahay iyo fulinta ama qabashada adeegga. Sawirka 16.3 ayaad ka arkaysaa tusaale arintan ku saabsan.

Sida isla sawirkaas ka muuqata, halkii labada nooc ee NoocaRukunkaAh iyo NoocaAdeegahaAh toos laysugu xidhiidhin lahaa, waxa lagu dadaalayaa in la sameeyo **interface** qeexaysa adeegga uu u baahanyahay nooca rukunka ama customer-ka ah. Kadib nooca adeeg bixiyaha ah, waxa looga baahanyahay inuu interface-kaas implement-gareeyo. Waxa jiri kara noocyo kale oo interface-kaas implement-garaynaaya. Nooca adeegga u baahan wakhtigii uu rabo ayuu badalan karaa nooca adeeggaas qabanaaya. Halkaas waxa kasoo baxaysa in xidhiidhkii labada nooc yaraado oo nooca adeegga u baahan uu qanciyo noockasta oo uu adeeggaas ka helaayo [5]. Arintani waa faa'iidooyinka ugu waaweyn ee ay **interface**-yadu leeyihiin.



Sawirka 16.3: Isticmaalka Interfaces-ka si loogu khafiifiyo xidhiidhka ka dhexeeya laba nooc

Arintan waxaynu ku adeegsandoonaa buuggan. Waxa kale oo aad arkidoontaa sida **interface**-yada loo adeegsado guud ahaanba.

16.4 Adeegga u dhaxeeya Controller-ka iyo View-ga

Intaynaan bilaabin qorista qaybaha kala duwan ee wajiga, aynu qeexo adeegyada ka dhexeeya ama u dhexeeya Controller-ka iyo View-ga (midba midka kale wuxu uga baahanyahay).

16.4.1 Adeegga ay Controller-ku uga baahantahay View-ga

Interface-ka 16.3 ayaad ka arkaysaa adeegga ay Controller-ku uga baahantahay View-ga. Waxan **interface**-kan kusoo uruuriyey ficilada aan is leeyahay Controller-ku way uga baahantahay View-ga. Waxan isku dayey in aanan model-ka oo dhan mar kaliya u gudbin View-ga ee aan qayb walba gooni ugu gudbiyo. Taasi waxay ii saamaxaysaa in marka model-ka qaybtaasi is badasho, aan View-ga u gudbiyo qaybtaas oo kaliya si uu u cusboonaysiiyo qaybtaas.

```

1 package nimi.controller;
2
3 import java.util.List;
4
5 import nimi.course.Course;
6 import nimi.course.CourseSession;
7 import nimi.student.Student;
8
9 public interface NimiView {
10     // Register-garee dhagayste dhaqdhaqaaqa isticmaalaha
11     void registerUIListener(NimiUIListener listener);
12
13     // Cusboonaysii View-ga
14     void updateSessions(List<CourseSession> sessions,
15                         List<Course> courses);
16
17     void updateCourses(List<Course> courses);
18
19     void updateStudents(List<Student> students,

```

```

20     List<CourseSession> sessions);
22     // Tus ardayda fadhigan aan ku siiyey ku jirta
    void showStudentsForSession(CourseSession session);
24 }

```

Tusaale 16.3: Interface-ka qeexaysa adeegga ay Controller-ku uga baahantahay View-ga

Aan sharaxaad ka bixiyo **interface**-ka 16.3:

1. Marka hore **interface**-ka magaceeda waxa la yidhaahdaa NimiView, waxaanay caloosha ku haysataa shan ficil oo matalaaya adeegga ay Controller-ku uga baahantahay View-ga
2. Ficilka registerUIListener waxa loogu talagalay in View-ga uu iska diiwaangeliyo noockasta oo implement-garaynaaya **interface**-ka NimiUIListener. Taas oo saamaxaysa in View-gu uu dhaqdhaqaaqa isticmaalaha u gudbiyo dhagaystahaas. Way soo socotaa **interface**-kaasi caynka ay u eegtahay ee ha dagdagin
3. Saddexda ficil ee kala ah updateSessions, updateCourses iyo updateStudents waxa loogu talagalay in Controller-ku ku cusboonaysiiso liiska fadhiyada, koorsooyinka iyo ardayda
4. Ficilka showStudentsForSession waxa loogu talagalay in Controller-ku ku amarto View-ga inuu tuso isticmaalaha liiska ardayda ku jirta fadhiga la siiyey
5. Ugu dambayntii ficiladani waa kuwii aan anigu is idhi Controller-ku waxay uga baahan kartaa View-ga. Adigu waxa laga yaabaa inaad wax ku darto ama aad ka dhinto.

Samee **interface**-ka sare ee NimiView, kuna kaydi package-ka nimi.controller.

16.4.2 Adeegga uu View-gu uga baahanyahay Controller-ka

Interface-ka 16.4 ayaad ka arkaysaa adeegga uu View-gu uga baahanyahay Controller-ka.

```

package nimi.controller;
2
import nimi.course.Course;
4 import nimi.course.CourseSession;
import nimi.student.Student;
6
/**
8  * Interface for passing user gestures to the controller
9  *
10 * @author Hussein
11 *
12 */
public interface NimiUIListener {
14     // Walxo cusub kudar (marka la gujiyo badhanka 'Cusub')
    public void addNewSession();
16
    public void addNewCourse();
18
    public void addNewStudent();
20
    // Ka tuur walaxda model-ka (marka la gujiyo badhanka 'Tuur')
22     public void removeSession(CourseSession session);
24
    public void removeCourse(Course course);
26
    public void removeStudent(Student student);
28
    // Tus ardayda ku jirta fadhigan
    public void studentsInSession(CourseSession session);
30
    // Badal macluumaadka walaxda
32     public void changeSessionInformation(CourseSession session,
        Object newValue, int field);
34
    public void changeCourseInformation(Course course,
36         Object newValue, int field);
38
    public void changeStudentInformation(Student student,
        Object newValue, int field);
40
    // Ardayga ka diiwaangeli fadhiga

```

```

42 public void enrollStudent(CourseSession session ,
    Student student);
44
45 // Ardayga ka tuur fadhigan
46 public void removeStudent(CourseSession session ,
    Student student);
48
49 // Badal fadhigan koorsadiisa
50 public void changeSessionCourse(CourseSession session ,
    Course course);
52 }

```

Tusaale 16.4: Interface ku tusaysa adeegga uu View-gu uga baahanyahay Controller-ka

Sideedaba View-gu wuxu Controller-ka u gudbinayaa dhaqdhaqaaqa isticmaalaha tusaale ahaan, gujinta badhanada, garaacista keyboard-ka, dhaqdhaqaaqa jiirka, iwm. Sidaas awgeed ficillo ka badan kii hore ayaynu u baahannahay. Haddii aan sharaxaad ka bixiyo **interface 16.4**:

1. Ficilada addNewSession, addNewCourse iyo addNewStudent waxa loogu talagalay in View-gu ku wargaliyo Controller-ka, marka isticmaaluhu uu codsado in uu nidaamka ku daro walxo cusub. Waa marka uu gujiyo badhanada *Cusub* ee caleenta hoosteeda ku yaalla. Waa marka uu isticmaaluhu rabo inuu fadhi cusub, koorso cusub ama arday cusub nidaamka inuu ku kordhiyo
2. Intaas waxa ku xiga ficilada removeSession, removeCourse iyo removeStudent oo loogu talagalay in View-gu ku wargaliyo Controller-ka marka uu isticmaaluhu codsado inuu nidaamka ka tuuro fadhi, koorso ama arday, waa marka uu gujiyo badhanka *Tuur* ee caleen walba hoosteeda ku yaalla
3. Ficilka studentsInSession waxa loogu talagalay in View-gu waco ama isticmaalo marka uu isticmaaluhu rabo inuu arko ardayda ku jirta fadhiga uu doorto, waa marka uu jooga caleenta fadhiyada oo uu waliba doortay fadhiyada mid ka mid ah, kadib uu rabo ardayda fadhigaas iska xaadirisay liiskooda
4. Ficilada ka bilaabma change* waxa loogu talagalay in lagu badalo macluumaadka walxaha sida fadhiga, koorlada ama ardayga. Macluumaad badalista waxa ka mid ah tusaale ahaan badalista magaca ardayda
5. Ficilka enrollStudent waxa loogu talagalay in arday laga xaadiriyo fadhi
6. Ficilka removeStudent waxa loogutalogay in arday lagaga tuuro fadhiga, haddii aanu fadhigaas ka qaybqaadan karin sababo jira awgood
7. Ugu dambayntii ficilka changeSessionCourse waxa loogu talagalay in lagu badalo koorlada fadhiga.

Waxad arkaysaa in ficilada **interface**-ka NimiUIListener ay ka badanyihiin tii hore. Waxa suurto ah in qaarkood aynu akhristaha cashar ahaan ugu dayno. Qor **interface**-ka sare ee NimiUIListener, kuna kaydi package-ka nimi.controller.

16.5 Bilawga Controller-ka

Waxan soo bandhigayaa bilawga Controller-ka. Sideedaba waxa la rabaa in aanay Controller-ku ku xidhnaan aragga, waayo aragyo kala duwan ayaa adeegsan kara hal Controller. Samee nooc cusub, kaas oo aad u bixiso NimiController adiga oo raacaya talaabada **NewClass**. Intaanad xidhin daaqadda samaynta nooca cusub ee sawirka **2.20**, waxad meesha *Interfaces* ku kordhisaa (adiga oo gujinaaya badhanka 'Add...') nimi.controller.NimiUIListener, intaas kadib guji **finish**. Nooca cusub ee madhan ee kuusoo baxa waxad u eekaysiisaa sida tusaalaha **16.5**.

```

package nimi.controller;
2
import nimi.course.Course;
4 import nimi.course.CourseSession;
import nimi.student.Student;
6
import nimi.NimiModel;
8
public class NimiController implements NimiUIListener {
10 private final NimiView view;
    private final NimiModel model;
12
    public NimiController(final NimiView view ,
14                          final NimiModel model) {
        this.view = view;

```

```

16     this.model = model;
17     this.view.registerUIListener(this);
18 }
19 @Override
20 public void addNewSession() {
21     // TODO Auto-generated method stub
22 }
23
24 @Override
25 public void addNewCourse() {
26     // TODO Auto-generated method stub
27 }
28
29 @Override
30 public void addNewStudent() {
31     // TODO Auto-generated method stub
32 }
33
34 @Override
35 public void removeSession(CourseSession session) {
36     // TODO Auto-generated method stub
37 }
38
39 @Override
40 public void removeCourse(Course course) {
41     // TODO Auto-generated method stub
42 }
43
44 @Override
45 public void removeStudent(Student student) {
46     // TODO Auto-generated method stub
47 }
48
49 @Override
50 public void studentsInSession(CourseSession session) {
51     // TODO Auto-generated method stub
52 }
53
54 @Override
55 public void changeSessionInformation(CourseSession session ,
56     Object newValue, int field) {
57     // TODO Auto-generated method stub
58 }
59
60 @Override
61 public void changeCourseInformation(Course course ,
62     Object newValue, int field) {
63     // TODO Auto-generated method stub
64 }
65
66 @Override
67 public void changeStudentInformation(Student student ,
68     Object newValue, int field) {
69     // TODO Auto-generated method stub
70 }
71
72 @Override
73 public void enrollStudent(CourseSession session ,
74     Student student) {
75     // TODO Auto-generated method stub
76 }
77
78 @Override
79 public void removeStudent(CourseSession session ,
80     Student student) {
81     // TODO Auto-generated method stub
82 }
83
84 @Override
85 public void changeSessionCourse(CourseSession session ,
86     Course course) {
87     // TODO Auto-generated method stub
88 }
89 }

```

Tusaale 16.5: Bilawga Controller borogaraamka. Feciladii oo madhan, waxa inoo diyaariyey borogaraamka Eclipse. Qoraalka “TODO Auto-generated method stub” Eclipse ayaa automatic ugu

qoraaya caloosha ficilka madhan ee u baahan inaad adigu qofto.

Controller-ku waa inay ogaato View-ga iyo Model-ka, sidaas awgeed waxan ku kordhiyey labada sifood ee laymanka 10-11. Ficilka dhisaha ah ee laymanka 13-18, waxa la siinayaa View-gii iyo model-kii, kadibna wuxu ku kaydinayaa sifooyinkaas. Kadib, laynka 17-aad waxay Controller-ku iska diiwaangelinaysaa View-ga si uu u soo ogaysiiyo dhaqdhaqaaqa isticmaalaha. Sumadda `@Override` oo Eclipse automatic ugu qoray ficilada qaar korkooda, waa kalkaaliso aan qasab ahayn oo haddii aad rabto waad masaxi kartaa. Waxay sheegaysaa in ficilkaasi ka tirsanyahay **interface**-ka ama superclass-ka noocan. Haddii ficil aan shuruuddaas buuxin aad sumaddaas duldhigto, borogaraamka Eclipse wuxu ku tusayaa cilad ficilka dulsaaan oo layn cas ayuu hoosta ka marinayaa.

16.6 Bilawga View-ga iyo ku xidhka qaybaha kale

Intaas kadib, waxaynu u baahannahay unugyada Swing mid ku haboon caleemaha isbaalyaalka ah ee aynu isticmaalayno. Waxa markiiba aynu arkaynaa in unugta `JTabbedPane` hawshan ku haboon tahay. Waxaynu bilaabaynaa sidii aynu u adeegsan lahayn unugtaas.

Samee nooc cusub, kaas oo aad u bixiso `NimiViewImpl` adiga oo raacaya talaabada **NewClass**. Intaanad xidhin daaqadda samaynta nooca cusub ee sawirka 2.20, waxad meesha *Superclass* ku qortaa `javax.swing.JTabbedPane`, waxa kale oo aad meesha *Interfaces* ku kordhisaa (adiga oo gujinaaya badhanka 'Add...') `nimi.ui.NimiView`, intaas kadib guji **finish**. Intaas kadib, waa inuu kusoo baxo tusaalaha 16.6.

```

1 package nimi.ui;
2
3 import java.util.List;
4
5 import javax.swing.JTabbedPane;
6
7 import nimi.course.Course;
8 import nimi.course.CourseSession;
9 import nimi.student.Student;
10
11 public class NimiViewImpl extends JTabbedPane implements
12     NimiView {
13
14     @Override
15     public void registerUIListener(NimiUIListener listener) {
16         // TODO Auto-generated method stub
17     }
18
19     @Override
20     public void updateSessions(List<CourseSession> sessions) {
21         // TODO Auto-generated method stub
22     }
23
24     @Override
25     public void updateCourses(List<Course> courses) {
26         // TODO Auto-generated method stub
27     }
28
29     @Override
30     public void updateStudents(List<Student> students,
31         List<CourseSession> sessions) {
32         // TODO Auto-generated method stub
33     }
34
35     @Override
36     public void showStudentsForSession(CourseSession session) {
37         // TODO Auto-generated method stub
38     }
39 }

```

Tusaale 16.6: Bilawga nooca `NimiViewImpl` oo ah View-gii borogaraamka. Nooca `JTabbedPane` ayaynu u adeegsanaynaa caleemaha isbaalyaalka ah.

Maxay tahay sababta aynu noocan ugu bixinay `NimiViewImpl`? Luuqadda Java iyo luuqadaha kaleba inta badan, haddii **interface**-ka magaceedu yahay X, nooca implement-garaynaaya waxa loo bixiyaa `XImpl`. Laakiin qasab ma aha arintaasi. Noocani wuxu matalayaa View-ga borogaraamyada.

Waxad ku xigiisaa inaad nooca `Nimi` ku darto labada sifooyin ee view iyo controller iyo labada ficil ee `createView` iyo `createController` oo aad dhammaadka ku darto, sidaad ka arkayso tusaalaha 16.7.

```
package nimi.ui;
```

```

2  ...
4  import java.awt.Container;

6  public class Nimi extends JFrame {
    private NimiModel model;
    private NimiView view;
    private NimiController controller;

10     public Nimi() {
12         ...
13         // samee caloosha daaqadda
14         createView();
15         // samee controller-kii
16         createController();
17     }

18     ...

20     private void createView() {
22         view = new NimiViewImpl();
23         setContentPane((Container)view);
24     }

26     private void createController() {
28         controller = new NimiController(
29             this.view, this.model);
30     }

```

Tusaale 16.7: Ku kordhi nooca Nimi samayntii Controller-ka iyo View-ga iyo isku xidhkooda

Intaynaan cutubkan ka gudbin, waxad nooca NimiViewImpl ku dartaa laymanka tusaalaha 16.8.

```

package nimi.ui;

2  ...

4  import java.awt.Dimension;

6  public class NimiViewImpl extends JTabbedPane implements
8      NimiView {
    NimiUIListener uiListener;

10     public NimiViewImpl() {
12         initialize();
13     }

14     @Override
16     public void registerUIListener(NimiUIListener listener) {
17         this.uiListener = listener;
18     }

20     ...

22     private void initialize() {
23         Dimension d = new Dimension(700, 500);
24         setPreferredSize(d);
25         addTabs();
26     }

28     private void addTabs() {
29         // TODO wuu soo socdaa
30     }

```

Tusaale 16.8: NimiTabbedPane oo hore loo waday waxyar

Waxaynu nooca NimiViewImpl ku kordhinay ficilka dhisaha iyo laba ficil oo kale initialize iyo addTabs sida ka muuqata tusaalaha 16.8 iyo ficilka registerUIListener oo aynu calooshiisa ku darnay hal layn. Waxa kale oo aynu ku darnay noocan sifada uiListener ee laynka 9-aad. Ficilka registerUIListener waxa soo wacaysa Controller-ka, si ay isaga diiwaangeliso View-ga.

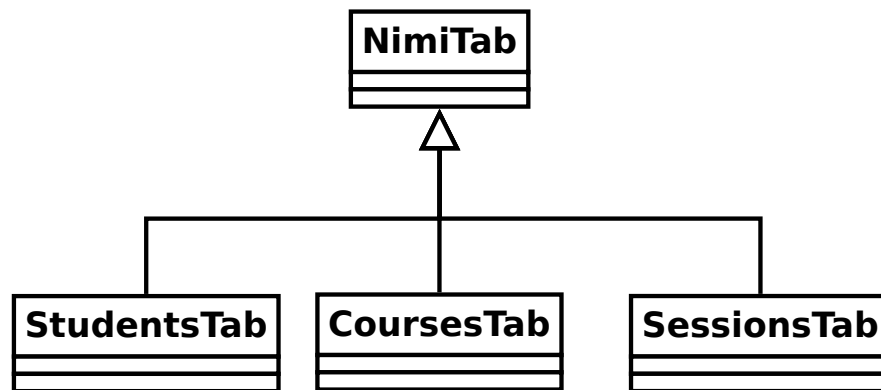
Haddii aan laba erey ka idhaahdo ficilka initialize, marka hore nooca Dimension wuxu kaydinayaa laba tiro oo kala matalaaya balaadhka iyo dhererka. Waxaynu samaynaynaa walax noocaas ah oo aynu siinayno balaadhka 700 iyo dhererka 500, kadibna waxay u gudbinaynaa ficilka setPreferredSize ee nooca

JTabbedPane, kaas oo loogutalogay in lagu siiyo cabirka uu doorbidaayo unugtaasi. Taasi waxay keenaysaa in cabirka caleemahan isbaalyaalla uu noqdo 700x500.

Intaas kadib kaxee nooca Nimi, si aad u aragto in uu balaadhkii iyo dhererkii daaqaddu isbadalay.

16.7 Aabaha caleemaha

Saddexda caleemoodba waxay ka simanyihiin waxyaalo badan, tusaale ahaan dhammaantood waxay leeyihiin layn hoose oo ay badhanno ku yaallaan, oo ah meeshii laga maamulaayey caleentaas, sidoo kale dhammaantood waxay leeyihiin miis soo bandhigaaya macluumaadka caleentaas. Halkii aynu caleen walba ku qorilahayn waxyaalaha ay ka simanyihiin (taas oo loo yaqaano *duplication*, ahna arin lagu dadaalo in laga fogaado), waxaan ku fikiray inaan wixii ay ka simanyihiin ku uruuriyo nooc sare, una bixiyo noocaas NimiTab, dhammaantood caleemahana laga soo dhiraandhiriyo noocaas. Qaabka ay noocaas iyo noocyada caleemuhu isugu xidhmayaan waxad ka arkaysaa diagram-ka 16.4.



Sawirka 16.4: Dhammaan caleemaha waxa laga soo dhiraandhiriyeey nooca NimiTab

Haddaba nooca NimiTab sidee ayuu u eekaanayaa? Samee nooc cusub oo aad u bixiso magacaas, kuna kaydi package-ka nimi.ui, una eekaysii sida tusaalaha 16.9 ee hoose.

```

1 package nimi.ui;
2
3 import javax.swing.BoxLayout;
4 import javax.swing.JPanel;
5
6 import nimi.controller.NimiUIListener;
7
8 abstract public class NimiTab extends JPanel {
9
10     protected NimiViewImpl nimiView;
11
12     public NimiTab(NimiViewImpl nimiView) {
13         this.nimiView = nimiView;
14         createContent();
15     }
16
17     /**
18      * This method creates the contents of the tab
  
```

```

20  */
    private void createContent() {
22      setLayout();
      addContent();
24      addControlButtons();
    }
26
    /**
28     * Ficilkani wuxu qeexayaa qaabka la rabo inay caleentani u
     * dhigto waxyaalaha ay caloosha ku haysato
30     */
    protected void setLayout() {
32      setLayout(new BorderLayout(this, BorderLayout.PAGE_AXIS));
    }
34
    /**
36     * Ficilkani wuxu samaynayaa caloosha caleenta wixii
     * ka sareeya badhannada hoose
38     */
    abstract protected void addContent();
40
    /**
42     * Ficilkani wuxu samaynayaa badhannada hoose ee caleenta
     */
44     abstract protected void addControlButtons();
  }

```

Tusaale 16.9: Nooca NimiTab

Aynu sharaxaad yar oo kooban ka bixinno noocan:

1. Laymanka 3 iyo 4 waxad ka arkaysaa inaynu laba nooc oo cusub oo ka tirsan Swing halkan ku isticmaalayno. Labadaas nooc waxay kala yihiin `BoxLayout` iyo `JPanel`
2. Bilawga nooca ee laynka 8-aad waxad ka arkaysaa isticmaalka ereyga **abstract** ee luuqadda Java u gaarka ah, kaas oo innagu cusub. Ereygaas waxa loo isticmaali karaa noocyada iyo ficilada. Marka nooca Java uu ereygaas ku bilaabmo, wuxu sheegayaa in aan noocaas laga samaynkariin walxo, oo tusaale ahaan qoraalka `NimiTab tab = new NimiTab(...)`; wuxu keenayaa cilad oo compiler-ka kama gudbaayo. Haddii noocaasi uu leeyahay ugu yaraan hal ficil oo ah **abstract**, waxa shuruud ah in noocana laga dhigo **abstract**
3. Ficilada **abstract** ku bilaabmaaya waxad arkaysaa in calooshoodu madhantahay. Waxa shuruud ah in nooc kasta oo laga soo dhiraandhiriyeey noocan, haddii aanu ahayn **abstract** inuu yeesho ficiladaas oo aan calooshoodu madhnayn
4. Ficilka `createContents` wuxu samaynayaa caloosha caleenta, waxaanu wacayaa saddexda ficil ee kale
5. Ficilka `setLayout` ee laynka 32 aynu ku isticmaalay, wuxu rakibayaa ama dajinayaa qaabka ay un-ugyada caleentani caloosha ku haysato u yaalaan. Ficilkaas waxa la siinayaa waxa loo yaqaano *Layout manager*, kaas oo ah walax maamulaysa dajinta unugyada. Halkan waxaynu ku isticmaalay-naa layout manager-ka `BoxLayout`, kaas oo is-dulsaaraaya ama is-baaldhigaaya, kolba sidii aad ku amarto (`BoxLayout.PAGE_AXIS` waxay keenaysaa in lays dulsaaro). Macluumaad fiican oo ku saabsan layout manager-yada Swing waxad ka haysaa Internet-ka.

Haddii aan iskuso duubno, nooca `NimiTab` (oo runtii magaciisu ku haboonaa `AbstractNimiTab`, maadaama uu yahay **abstract**), wuxu ka sugayaa noocyada laga soo dhiraandhiriyo in ay la yimaadaan ficilada `addContent` iyo `addControlButtons`. Waxay labadaas ficil qabanayaan waynu arki, markaynu samayno caleenta ugu horaysa.

16.8 JTable iyo isticmaalkiisa

Sawirka 16.5 ee hoose ayaad ka arkaysaa miiska (table) Java ee `JTable` ee aynu isticmaalidoono. Waxaynu sharaxaad yar ka bixinaynaa qaybaha kala duwan ee sawirkaas.

1. Marka hore waa laynka ugu sareeya oo loo yaqaano *header* ama madaxii. Madaxu wuxu sheegayaa magacyada tiirarka (columns)
2. Marka labaad miisku wuxu ka koobanyahay tiirar (columns). Tiir kastaaba wuxu isticmaalaha tusayaa macluumaad isku nooc ah

Fadhiyada	Koorsooyinka	Ardadya	Dhibco
	Koorso	Billawga	Dhibco
☐	Maths Introductory Course (Maths, MAT-01)	Jan 1, 2011	1
☐	Maths Introductory Course (Maths, MAT-01)	Apr 1, 2011	1
☐	Functions and equations (Maths, MAT-02)	Apr 1, 2011	3
☐	English Introductory Course (Languages, ENG-01)	Jan 1, 2011	1
☐	Af-soomaali 1 (Languages, SOM-01)	Jan 1, 2011	3
☐	Introduction to computer science (Computer Science, COM-01)	Jan 1, 2011	2
☐	Introduction to computer science (Computer Science, COM-01)	Apr 1, 2011	2
☐	Programming 1 (Computer Science, COM-02)	Apr 1, 2011	5

Sawirka 16.5: Swing JTable

3. Miisku wuxu ka koobanyahay unugyo (cells). Unug kastaaba waxay isticmaalaha tusaysaa hal macluumaad
4. Ugu dambayntii miisku wuxu ka koobanyahay sidoo kale safaf (rows) oo ah laymanka jiifa, halka columns ka yihiin laymanka taagan
5. Unug kastaaba waxay leedahay goob, kaas oo ka kooban qiime laba qaybood ah; nambarka safka iyo nambarka tiirka. Safka ugu horeeya (ee ugu sareeya) nambarkiisu waa eber, ka ku xiga waa kow, ka ku xiga waa laba, sidaas ayay ku soconayaan tirada safafku. Sidoo kale tiirka ugu horeeya (ee bidixda xiga) nambarkiisu waa eber, ka ku xiga waa kow, ka ku xiga laba, sidaas ayay ku soconayaan tirada tiirarku.

Waxaynu soo qaadanaaynaa borogaraam yar oo gooni u taagan oo ku tusaaya isticmaalka JTable. Waxaynu isticmaalaynaa noocii HelloSwing ee aynu hore u soo aragnay. Haddii aad rabto waxad isticmaali kartaa nooc cusub.

```

package nimi.ui;
2
import java.awt.Dimension;
4
import javax.swing.JFrame;
6
import javax.swing.JScrollPane;
import javax.swing.JTable;
8
import javax.swing.SwingUtilities;

10
public class HelloSwing {
    private static void createAndShowGUI() {
12
        // Header ama madaxa
        String[] columnNames = { "Magaca", "Magaca Aabaha", "Da'da" };

14

        // Data-dii miiska
16
        Object[][] data = {
            { "Caasha", "Maxamed", new Integer(5) },
18
            { "Faysa", "Maxamed", new Integer(30) },
            { "Amal", "Ismaaciil", new Integer(20) },
20
            { "Najma", "Xirsi", new Integer(12) },
            { "Ahmed", "Ismaaciil", new Integer(10) } };

22

        // Samee miis cusub adiga oo siinaaya data-da iyo madaxa
24
        JTable table = new JTable(data, columnNames);
        // balaadh iyo dhererka miiska inta muuqata
26
        table.setPreferredScrollableViewportSize(new Dimension(340, 80));

28

        // Samee daaqaddii
        JFrame frame = new JFrame("Tusaale JTable");
30
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

32

        // JTable inta badan waxa la dhexgaliyaa JScrollPane
        JScrollPane scrollPane = new JScrollPane(table);

34

        // dhexgali miiska daaqadda
36
        frame.getContentPane().add(scrollPane);
        frame.pack();

38

        // tus daaqadda
40
        frame.setVisible(true);
    }

42

    public static void main(String[] args) {
44
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {

```

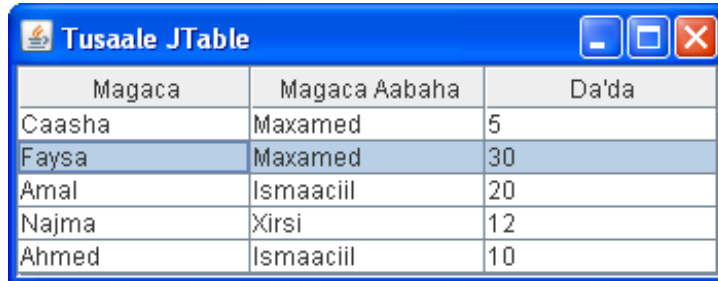
```

46         createAndShowGUI();
47     }
48 }
50 }

```

Tusaale 16.10: Tusaale ku tusaaya isticmaalka JTable

Haddii aad kaxayso tusaalaha 16.10 waa inuu kuusoo baxo sawirka 16.6.



Magaca	Magaca Aabaha	Da'da
Caasha	Maxamed	5
Faysa	Maxamed	30
Amal	Ismaaciil	20
Najma	Xirsi	12
Ahmed	Ismaaciil	10

Sawirka 16.6: Tusaale Swing JTable

Tusaalaha 16.10 wuxu ku tusayaa qaybaha kala duwan ee uu ka koobanyahay JTable. Laynka 13-aad waxad ka arkaysaa magacyada tiirarka. Laymanka 16-21 waxad ka arkaysaa macluumaadkii miiska uu soo bandhigaayey. Gaar ahaan laynka 24-aad waxad ka arkaysaa samaynta miiska iyo in la siinaayo data-dii iyo qoraalka madaxa. Inta dambe waa samaynta daaqadda iyo in miiska la dhexgaliyo. Halka aynu tusaalahan miiska ka siinay data-dii oo diyaar ah, borogaraamka aynu dhisayno, data-du waxay ka imanaysaa model-ka.

16.9 Miisaska caleemaha

Caleen kasta waxa ku hoos jira miis, kaas oo isticmaalaha tusaaya macluumaadka caleentaas, si uu u akhristo ama wax uga badalo. Waxyaalo badan ayaa ka dhexeeya miisaska caleemahaas sidaad arki-doontaba. Maadaama aynu si hordhac ah u aragnay ama aragti guud ka haysano isticmaalka nooca JTable, Waxaynu bilaabaynaa sidii aynu ugu adeegsan lahayn borogaraamkeena. Halka tusaalaha sare ee 16.10 aynu data-dii toos u siinay miiska, waxa loo baahanyahay inaynu model-ka borogaraamka qayb kamid ah miiska kusoo bandhigno.

Laynka 24-aad ee tusaalaha 16.10, waxad ka arkaysaa hal si oo aad u samayn karto walax cusub oo JTable ah. Qaabkaas waxad isticmaalaysaa markaad haysato data-dii oo diyaar ah oo aan is badalaynin, laakiin markaad rabto in data-da la badali karo ama wax cusub lagu darikaro, waa inaad qaab ka duwan u samayso walaxda JTable. Nasiibwanaag nooca JTable wuxu ku siinayaa siyaabo dhawr ah oo kala duwan oo aad u samayn karto walax cusub oo noocedu yahay JTable. Siyaabahaas waxa ka mid ah inaad JTable ficilkiisa dhisaha ah siiso model, taas oo uu miisku tusaayo isticmaalaha. Laakiin waxa jira shuruudo lagu xidhaayo model-kaas. Model-kaasi waa inuu noqdo TableModel (javax.swing.table.TableModel). TableModel waa **interface**, taasi oo ka dhigan inaan toos loo isticmaali karin, laakiin loo baahanyahay noocyo **implements**-garaynaaya interface-kaas.

UML diagram-ka 16.7, waxad ka arkaysaa qaabka model-lada JTable ee aynu adeegsandoonno u yaalaan iyo sida ay ugu xidhanyihiin interface-kaas aynu kor ku xusnay ee TableModel. Nooca AbstractTableModel (javax.swing.table.AbstractTableModel) wuxu ka midyahay Swing, laakiin magaciisa ayaad ka arkaysaa inaan toos loo isticmaali karin, sidaas awgeed waxaynu kasoo dhiraandhirinay nooca aynu leenahay oo aan isna toos loo isticmaali karin, laakiin noocyada ugu hoose toos ayaynu u isticmaali karnaa.

Waxyaalaha ay ka simanyihiin miisaskaasi waxaynu ku uruurinay nooca AbstractNimiTableModel. Kadib, miiska kasta model-kiisii waxaynu kasoo dhiraandhirinaynaa isaga. Sawirka 16.8 wuxu ku tusayaa sida ay isugu xidhanyihiin noocyada UML-ka sare iyo nooca JTable.

Waxaynu soo bandhigaynaa nooca AbstractNimiTableModel qoraalkiisii, waxaanad ka arkaysaa 16.11

```

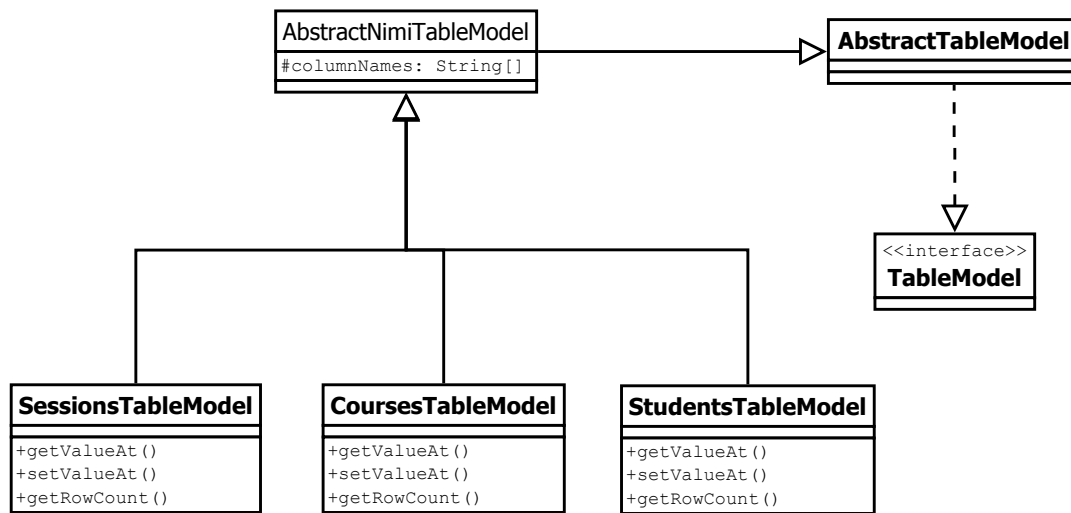
package nimi.ui;

import javax.swing.table.AbstractTableModel;

public abstract class AbstractNimiTableModel extends
    AbstractTableModel {
    protected String[] columnNames;

    public AbstractNimiTableModel(String[] columnNames) {

```



Sawirka 16.7: Nooca AbstractTableModel wuxu ka midyahay Java, waxaanu implement-garaynayaa interface-ka TableModel. Laakiin maadaama uu yahay abstract, lama isticmaali karo, waxa loo baahan-tahay in laga soo dhiraandhiriyo nooc aad leedahay.

```

10     this.columnNames = columnNames;
11 }
12
13 @Override
14 public String getColumnName(int col) {
15     return columnNames[col];
16 }
17
18 @Override
19 abstract public Object getValueAt(int row, int col);
20
21 @Override
22 public Class getColumnClass(int c) {
23     Object value = getValueAt(0, c);
24     return value.getClass();
25 }
26
27 @Override
28 public int getColumnCount() {
29     return columnNames.length;
30 }
31
32 @Override
33 public boolean isCellEditable(int row, int col) {
34     return true;
35 }
36
37 @Override
38 abstract public void setValueAt(Object value, int row,
39     int col);
40
41 @Override
42 abstract public int getRowCount();
43 }

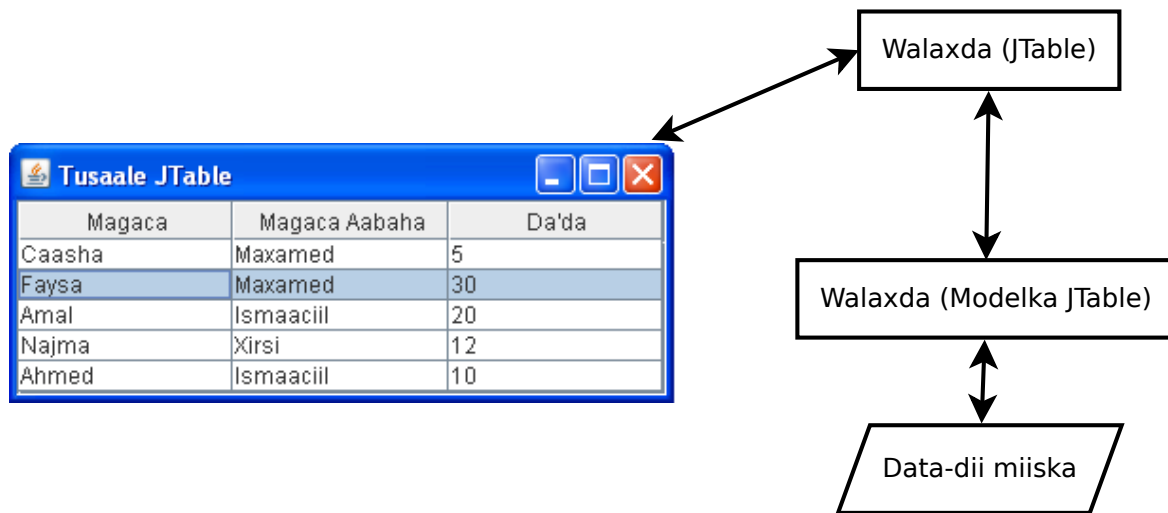
```

Tusaale 16.11: Nooca AbstractNimiTableModel

Dhammaan noocyada laga soo dhiraandhiriyo nooca AbstractNimiTableModel, waa inay la yimaadaan ficilada getValueAt, setValueAt iyo getRowCount. Waxa kale oo qasab ah inay ficilkiisa dhisaha ah u soo gudbiyaan magacyada tiirarka miiska.

16.10 Caleenta - Fadhiyada

Waxaynu bilaabaynaa inaynu fadhiisino caleentii ugu horaysay oo ah fadhiyada (course sessions). Caleentaasi waxay isticmaalaha tusaysaa macluumaadka fadhiyada nidaamka ku jira. Waxa kale oo ay isticmaalaha u saamaxaysaa inuu macluumaadka fadhiga badalo. Qaybta 16.1 waxaynu kusoo samaynay model tijaabo ah, oo aynu isticmaalaha tusayno marka borogaraamka la istaado. Waxaynu rabnaa



Sawirka 16.8: Walaxda jTable, Walaxda model-ka u ah iyo data-da xidhiidhka ka dhexeeya

inaynu fadhiyada model-kaas ku jira isticmaalaha tusno. Caleentan nooca matalaaya waxaynu u bixinay SessionsTab, waxaanu ku bilaabmayaa sida tusaalaha 16.12. Samee noocaas cusub, kuna bilaw sida tusaalahaas ka muuqata. Xasuuso inaad kasoo dhiraandhiriso nooca aabaha caleemaha ee NimiTab.

```

1 package nimi.ui;
2
3 public class SessionsTab extends NimiTab {
4
5     public SessionsTab(NimiView nimiView) {
6         super(nimiView);
7     }
8
9     @Override
10    protected void addContent() {
11
12    }
13
14    @Override
15    protected void addControlButtons() {
16
17    }
18 }

```

Tusaale 16.12: Bilawga caleenta Fadhiyada. Waxa laynooga baahanyahay inaynu qorno kaliya labada fiil ee qasabka ah ee aynu ka dhaxalay nooca sare ee NimiTab ee addContent iyo AddControlButtons

Waxad caleentaas ku dartaa nooca NimiViewImpl sidaad ka arkaysa tusaalaha 16.13.

```

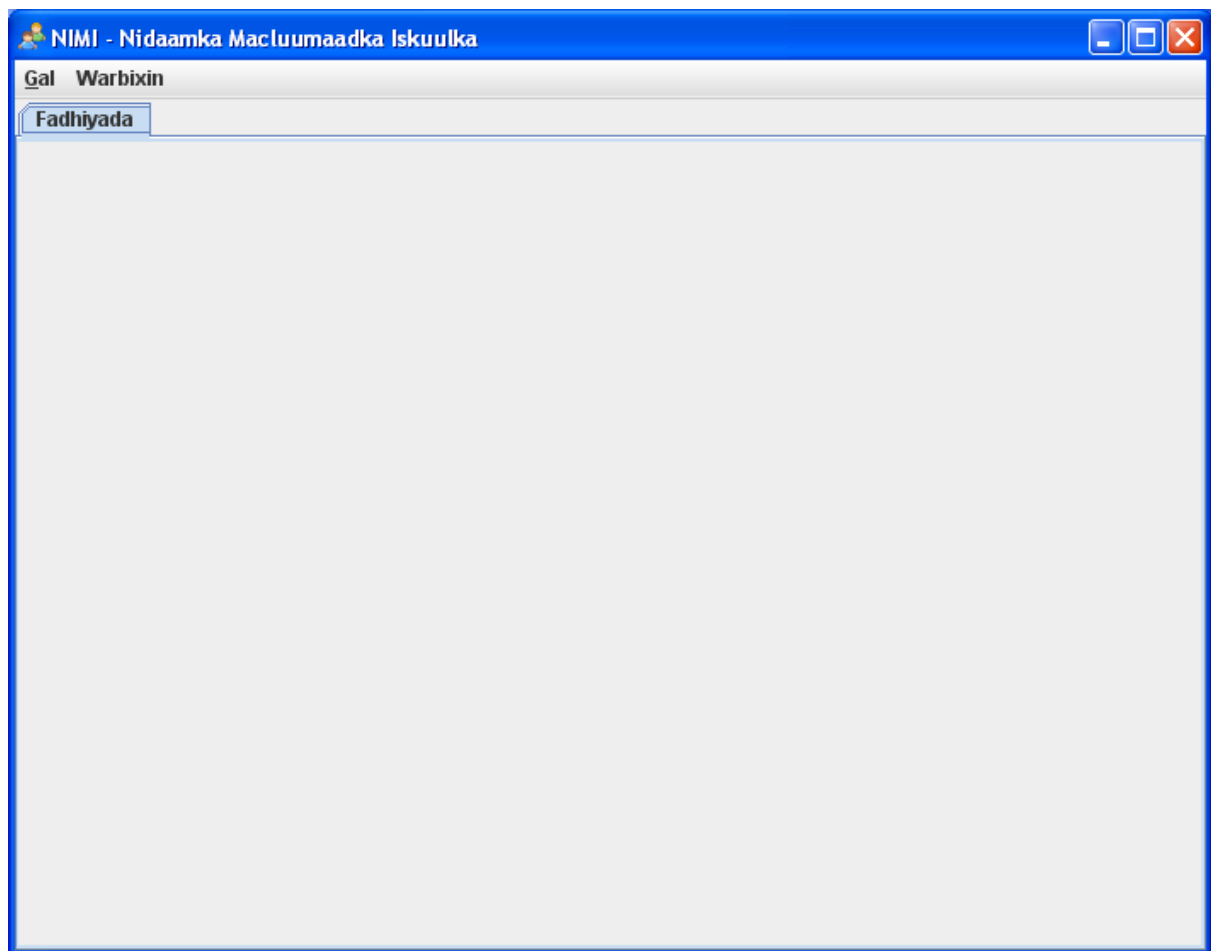
1 package nimi.ui;
2
3 ...
4
5 public class NimiViewImpl extends JTabbedPane implements
6     NimiView {
7     private SessionsTab sessionsTab;
8
9     ...
10
11    private void addTabs() {
12        this.sessionsTab = new SessionsTab(this);
13        addTab("Fadhiyada", this.sessionsTab);
14    }
15 }

```

Tusaale 16.13: Nooca NimiViewImpl oo lagu daray caleentii ugu horaysay ee fadhiyada. Waxa soo kordhay laynka 7 iyo laymanka 12-13

Intaas kadib, haddii aynu kaxaynno borogaraamka, waa in daaqadii borogaraamku u eekaato sawirka 16.9 oo kale. Sawirkaas waxad ka arkaysaa in caleentii ugu horaysay ee fadhiyada ay muuqato, laakiin ay madhantahay caloosheedu.

Aan sharaxaad kooban ka bixiyo tusaalaha sare 16.8:



Sawirka 16.9: Muuqaalka borogaraamka oo caleenta fadhiyada oo lagu kordhiyey

1. Laynka 7-aad sifada sessionsTab waxay tilmaamaysaa caleenta fadhiyada
2. Laynka 12-aad ayaynu samaynaynaa walaxda mataleysa caleentaas. Ficiilkeeda dhisaha ah waxa loo gudbinayaa walaxdan NimiViewImpl, taas oo ay ka simanyihiin dhammaan caleemuhu
3. Ugu dambayntii laynka 13-aad waxaynu adeegsanaynaa ficilka addTab ee ka yimid nooca JTabbedPane, kaas oo aynu u gudbinayno magaca caleenta iyo walaxda mataleysa caleentaas
4. Talaabooyinkan caleemaha oo dhan way ka simanyihiin.

Waxaynu bilaabaynaa inaynu buuxino caloosha caleentaas. Waxaynu marka hore buuxinaynaa ficilka addControlButtons oo aad ka arkayso tusaale 16.14 isaga oo badankiisa diyaar ah.

```

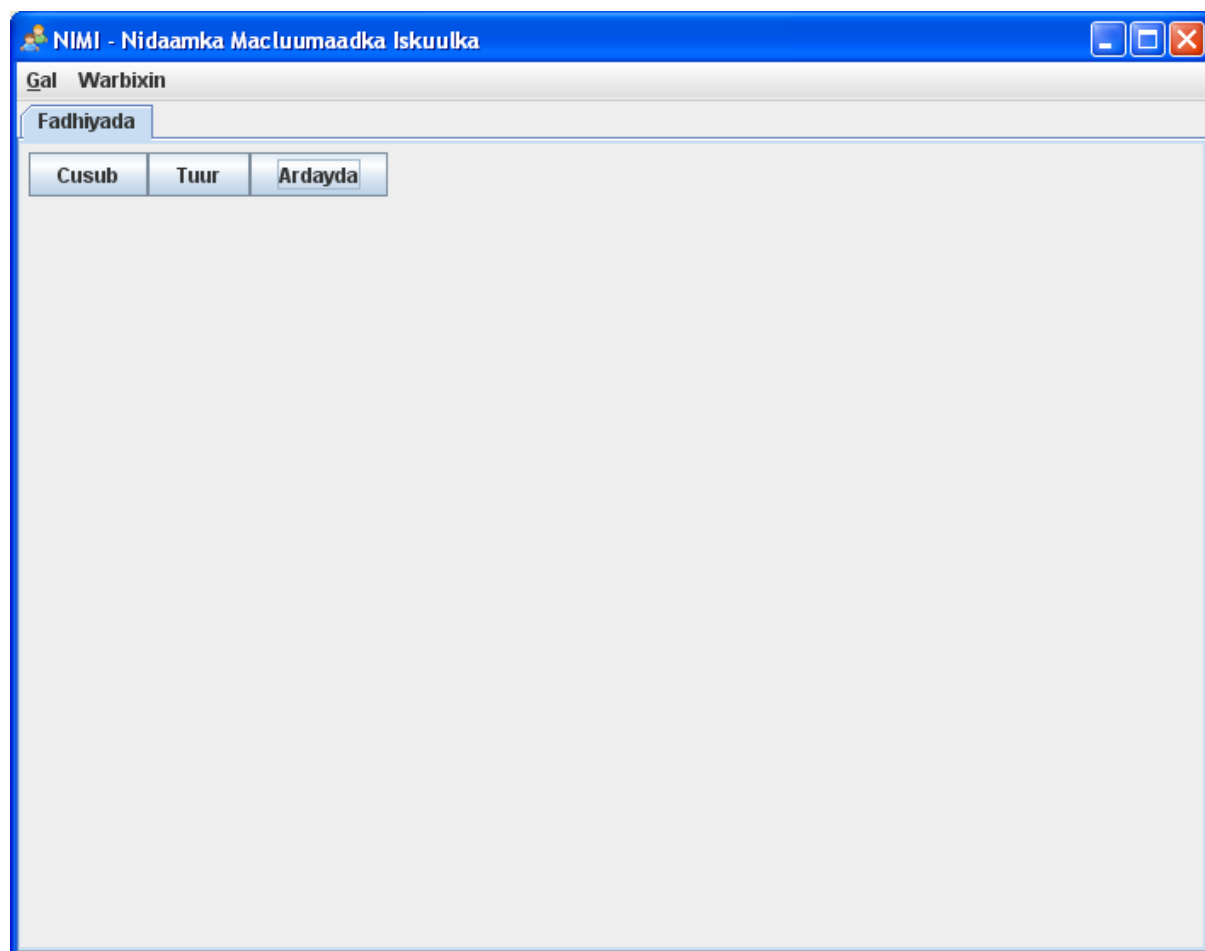
1 package nimi.ui;
2
3 import javax.swing.JPanel;
4 import javax.swing.JButton;
5 import javax.swing.BoxLayout;
6
7 import java.awt.event.ActionListener;
8 import java.awt.event.ActionEvent;
9
10 import javax.swing.Box;
11 import javax.swing.BorderFactory;
12
13 public class SessionsTab extends NimiTab {
14
15     ...
16
17     @Override
18     protected void addControlButtons() {
19         // JPanel waxa lagu uruurin karaa tusaale ahaan guruub badhanna ah
20         JPanel buttons = new JPanel();
21         buttons.setLayout(new BoxLayout(buttons, BoxLayout.LINE_AXIS));
22         JButton button = new JButton("Cusub"); // Cusub
23         button.addActionListener(new ActionListener() {
24             public void actionPerformed(ActionEvent e) {
25                 // Cusub
26             }
27         });
28         buttons.add(button);
29         button = new JButton("Tuur"); // Tuur
30         button.addActionListener(new ActionListener() {
31             public void actionPerformed(ActionEvent e) {
32                 // Tuur
33             }
34         });
35         buttons.add(button);
36         button = new JButton("Ardayda"); // Ardayda
37         button.addActionListener(new ActionListener() {
38             public void actionPerformed(ActionEvent e) {
39                 // Ardayda
40             }
41         });
42         buttons.add(button);
43         buttons.add(Box.createHorizontalGlue());
44         buttons.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
45         add(buttons);
46     }
47 }

```

Tusaale 16.14: Caleenta fadhiyada oo lagu daray badhannada hoose

Haddii aad mar kale kaxayso borogaraamka, waa inuu kusoo baxo sawirka 16.10 muuqaal u dhigma. Intaynaan sii socon aan sharaxaad kooban ka bixiyo tusaalaha 16.14:

1. Marka hore soo **import**-garee noocyada aad u baahantahay ee laymanka 3-11
2. Noocyada JPanel iyo BoxLayout waxaynu kusoo isticmaalay nooca NimiTab ee tusaalaha 16.9. Noocyada ActionListener iyo ActionEvent iyana waxaynu kusoo isticmaalay qaybtii dhagaysteyaasha dhacdooyinka. Waxa innagu cusub labada nooc ee Box iyo BorderFactory ee lasoo **import**-gareeyey laymanka 10-11. Si aad u ogaato labadaas nooc waxay qabanayaan waxad comment saartaa laymanka 43 iyo 44 (ku bilaw calaamadda //), kadibna kaxee borogaraamka oo eeg araggiisa waxa iska badala. Intaas kadib, laynka hore ee 43 comment-ka ka saar (ka masax calaamadda bilawga ee //), kadibna



Sawirka 16.10: Caleenta fadhiyada oo lagu kordhiyey badhanada kantaroolka ee hoose, kuwaas oo aan wali waxba qabanaynin marka la gujiyo

mar kale kaxee borogaraamka oo eeg araggiisa. Ugu dambayntii laynka 44 comment-ka ka saar, kadibna mar kale kaxee oo eeg sida uu borogaraamku u eekaado iyo waxa uu laynkaasi ka badalo muuqaalkiisa. Intaas ka dib waa inay kuu cadaato labadaas nooc waxay muuqaalka borogaraamka kusoo kordhinayaan

3. Laynka 20-aad waxaynu samaynaynaa walax JPanel ah, taas oo aynu laynka ku xiga layout-keeda ka dhignay BorderLayout. Qiimaha BorderLayout.LINE_AXIS wuxu layout manager-ka u sheegayaa inuu is-baaldhigo badhanada. Intaas kadib waxaynu badhanada ku uruurinaynaa walaxdaas JPanel-ka ah innaga oo isticmaalayna ficilkeeda add. Ugu dambayntii laynka 45-aad ayaynu caleentan ku daraynaa walaxdaas, oo caloosha ku haysata badhanadii. Arintan way ka simanyihiin caleemaha oo dhan.

Badhannadii waa diyaar, laakiin waxba ma qabanayaan marka la gujiyo. Waxaynu diyaarinaynaa miiska ka sareeya badhanada oo aad ka arkayso qorshaheenii, haddii aad dib u milicsato. Sidaad arki-doontaba, waa inaynu qaybaha kala duwan ee borogaraamka wax ka badalo ama ku darno, si aynu u aragno liiska fadhiyada ee caleenta aynu kusoo bandhigayno.

Waxaynu ka bilaabaynaa nooca Nimi ee matalaaya daaqadda. Waxad noocaas gaar ahaan ficilkiisa createController ku dartaa laynka 10-aad ee tusaalaha 16.15.

```

1 package nimi.ui;
2
3 ...
4
5 public class Nimi extends JFrame {
6     ...
7
8     private void createController() {
9         ...
10        controller.updateAllView();
11    }
12 }

```

Tusaale 16.15: Cusboonaysii muuqaalka oo dhan marka borogaraamku istaadhmo

Controller-ka samayntiisa kadib, waxaynu adeegsanaynaa ficilkiisa updateAllView, kaas oo aynu ku amrayno inay muuqaalka cusboonaysiiso. Waxa la rabaa inaynu ficilka updateAllView ku darno Controller-ka. Ujeedada ficilkaas waa in muuqaalka ama aragga lagu cusboonaysiiyo gidigii, waxaanad ka arkaysaa tusaalaha 16.16.

```

1 package nimi.ui;
2
3 ...
4
5 public class NimiController implements NimiUIListener {
6     ...
7
8     /**
9      * Aragga oo dhan cusboonaysii
10     */
11     public void updateAllView() {
12         this.view.updateSessions(this.model.getSessions(),
13                                 this.model.getCourses());
14         this.view.updateStudents(this.model.getStudents(),
15                                 this.model.getSessions());
16         this.view.updateCourses(this.model.getCourses());
17     }
18 }

```

Tusaale 16.16: Ficilka cusboonaysiinta muuqaalka oo dhan, oo lagu daray Controller-ka. Ficilkaasi wuxu amrayaa qaybaha kala duwan ee muuqaalka isaga oo u gudbinaaya model-ka qaybaheeda kala duwan, si ay isu cusboonaysiiyaan

Sideedaba Controller-ku waxay ka war haysaa model-ka iyo View-ga, maadaama aynu labadaasba u gudbinay markii aynu samaynaynay. Ficilkan cusub ee updateAllView wuxu isticmaalayaa saddexda ficil ee View-ga ee kala ah updateSessions, updateStudents iyo updateCourses. Ficil kasta wuxu u gudbinayaa model-ka qaybta khusaysa. Tusaale ahaan ficilka updateCourses ee laynka 16 lagu isticmaalay wuxu u gudbinayaa liiska koorsooyinka this.model.getCourses().

Xawaare fican ayaynu ku soconaa. Wali waxa laynooga baahanyahay inaynu noocyada NimiViewImpl iyo SessionsTab waxyar ku darno. Aynu ku bilawno nooca hore. Waxad noocaas ku dartaa laynka 11-aad ee caloosha ficilka updateSessions ee tusaalaha 16.17

```

package nimi.ui;
2
...
4
public class NimiViewImpl extends JTabbedPane implements NimiView {
6
...
8
@Override
public void updateSessions(List<CourseSession> sessions ,
10     List<Course> courses) {
    this.sessionsTab.updateSessions(sessions , courses);
12 }
...
14 }

```

Tusaale 16.17: Ficilka cusboonaysiinta fadhiyada

Ficilka `updateSessions` wuxu cusboonaysiinayaa caleenta fadhiyada. Si uu arintaas u gaadho wuxu adeegsanayaa isla ficilka `updateSessions` ee walaxda caleentaas matalaysa `this.sessionsTab`. Waxaynu isna waxa ka dhiman ku daraynaa nooca `SessionsTab`, kaas oo aad ka arkayso tusaalaha 16.18.

```

package nimi.ui;
2
...
4
import javax.swing.JScrollPane;
import javax.swing.JTable;
import java.util.List;
import nimi.course.CourseSession;
import nimi.course.Course;
10
public class SessionsTab extends NimiTab {
12     private List<CourseSession> sessions;
    private List<Course> courses;
14     private SessionsTableModel tableModel;
    private JTable sessionsTable;
16
    ...
18     @Override
    protected void addContent() {
20         this.tableModel = new SessionsTableModel();
        sessionsTable = new JTable(this.tableModel);
22         // Miiska u samee scrollpane kadib ku dhex tuur
        JScrollPane scrollPane = new JScrollPane(sessionsTable);
24         // Scrollpane-kii caleenta ku dhextuur
        add(scrollPane);
26     }
28
    ...
30     public void updateSessions(List<CourseSession> sessions ,
        List<Course> courses) {
32         this.sessions = sessions;
        this.courses = courses;
34         tableModel.fireTableDataChanged();
    }
36
    private class SessionsTableModel extends AbstractNimiTableModel {
38         public SessionsTableModel() {
            super(new String[] {"Koorso", "Bilawga", "Dhibco"});
40         }
42
        @Override
        public Object getValueAt(int row, int col) {
44             CourseSession session = sessions.get(row);
            switch (col) {
46                 case 0:
                    return session.getCourse();
48                 case 1:
                    return session.getStartDate();
50                 case 2:
                    return session.getNumberOfCredits();
52                 default:
                    break;
54             }
            return "";

```

```

56     }
57
58     @Override
59     public void setValueAt(Object value, int row, int col) {
60         // TODO
61     }
62
63     @Override
64     public int getRowCount() {
65         return sessions.size();
66     }
67 }
68 }

```

Tusaale 16.18: Caleenta fadhiyada oo lagu daray miiskii

Nooca SessionsTab waxa kusoo kordhay wax badan sidaad ka arkayso tusaalaha 16.18. Sidaas awgeed waa in sharaxaad laga bixiyo.

1. Marka hore waxaynu diyaarinay caloosha ficilka addContent. Ficilkaasi wuxu samaynayaa model-kii miiska iyo miiskii, kadibna wuxu ku kordhinayaa caloosha caleenta miiskaas cusub. Miiska waxaynu u adeegsanay nooca JTable iyo nooca JScrollPane oo inta badan lala isticmaalo. Way ka simanyihiin caleemuhu arintan
2. Nooca SessionsTableModel ee laymanka 37-67 ayaa matalaaya model-ka miiska. Waxaynu kasoo dhiiraandhirinay nooca AbstractNimTableModel oo ah aabaha model-ka miisaska. Noocaas waxaynu adeegsanaynaa laynka 20-aad, halkaas oo aynu model-kii miiska samaynayno kadibna ku kaydinayno sifada **this.tableModel**. Laynka xiga ee 21-aad waxaynu samaynaynaa miiskii isaga oo aynu u gudbinayno model-kii aynu laynka ka sareeya ku samaynay
3. Waxaynu noocan SessionsTableModel ku darnay ficiladii qasabka ahaa inuu la yimaado. Waxa kale oo uu ficilka dhisaha ahi u gudbinayaa nooca sare magacyada tiirarka (waa laynka 39-aad), taas oo iyana qasab ah
4. Ficilka getValueAt ee laymanka 43-56, wuxu soo celinayaa qiimihii lagusoo bandhigi lahaa unugta coordinate-keeda la siiyey. Laynka 44-aad waxaynu marka hore raadinaynaa fadhiga ku beegan qiimaha safka (row), kadibna waxaynu soo celinaynaa sifada ku beegan qiimaha tiirka (col). Tusaale ahaan (0,1) waxay soo celinaysaa fadhiga ugu horeeya taariikhda bilawgiisa
5. Ficilka getRowCount wuxu miiska u sheegayaa tirada guud ee fadhiyada
6. Waxa kale oo isna muhiim ah ficilka setValueAt, kaas oo uu miisku wacaayo marka isticmaaluhu damco inuu badalo macluumaadka fadhiga
7. Sideedaba model-ka miisku waxay isticmaalaha tusaysaa fadhiyada ku kaydsan sifada sessions ee laynka 12-aad
8. Ficilka updateSessions ee laymanka 30-35, waxa soo wacaaya nooca NimViewImpl isaga oo u soo gudbi-naaya liiska fadhiyada. Ficilkaas marka hore wuxu liiska fadhiyada ku kaydinayaa sifada **this.sessions** waa laynka 32, kadibna wuxu wacayaa ficilka fireTableDataChanged, kaas oo miiska wargalinaaya in data-diisii isbadashay loona baahanyahay inuu is cusboonaysiiyo.

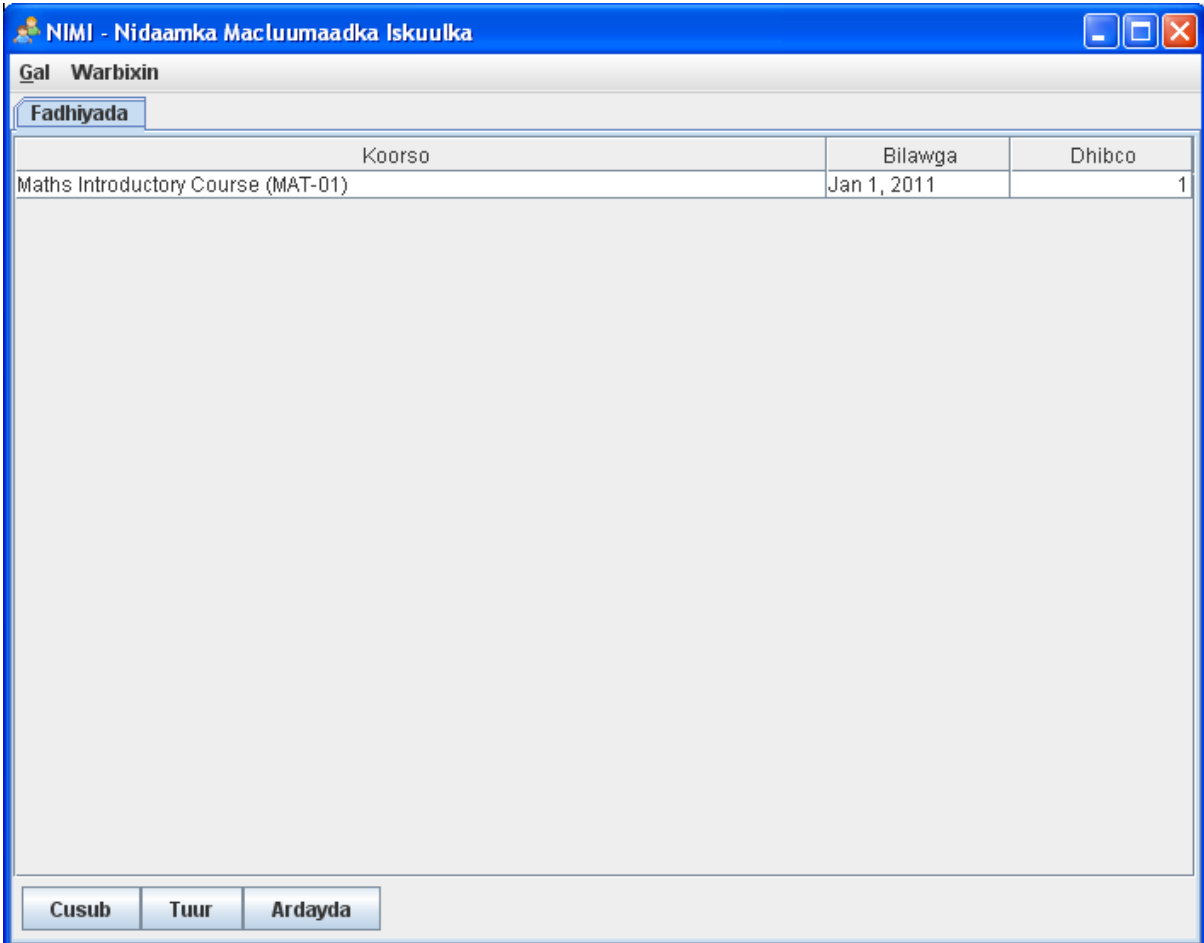
Intaas kadib, haddii aad kaxayso borogaraamka mar kale waa inuu kuusoo baxo sawirka 16.11 mid u dhigma. Waxaynu arkaynaa macluumaadkii halka fadhi ee model-ka ku jira.

Waxaynu kaga shaqaysiinaynaa badalista macluumaadka gaar ahaan taariikhda bilawga iyo dhibc-aha fadhiga. Waxad ficilka setValueAt ee nooca SessionsTableModel (ee ku dhexjira nooca SessionsTab) u eekaysiisaa sida tusaalaha 16.19 ee hoose.

```

package nimi.ui;
2
...
4
public class SessionsTab extends NimTable {
6
...
8
private class SessionsTableModel extends AbstractNimTableModel {
...
10
@Override
12
public void setValueAt(Object value, int row, int col) {
    CourseSession session = sessions.get(row);

```



NIMI - Nidaamka Macluumaadka Iskuulka

Gal Warbixin

Fadhiyada

Koorso	Bilawga	Dhibco
Maths Introductory Course (MAT-01)	Jan 1, 2011	1

Cusub Tuur Ardayda

Sawirka 16.11: Caleenta fadhiyada oo lagu kordhiyey miiska sare

```

14         uiListener.changeSessionInformation(session, value, col);
15     }
16     ...
17 }
18 }

```

Tusaale 16.19: Badalista macluumaadka fadhiga 1-aad

Sidaad tusaalahaas ka arkayso ficilka `setValueAt` wuxu adeegsanayaa ficilka `changeSessionInformation` ee sifada `uiListener` oo ah dhagaystihii dhaqdhaqaaqa isticmaalaha. Haddii aad dib u milicsato ficilka dhisaha ee Controller-ka `NimiController`, waxad laynka ugu hooseeya ka arkaysaa inay iska diiwaangelinayso `View`-ga si loogusoo gudbiyo dhaqdhaqaaqa isticmaalaha. Arintaas waxay ka mid tahay fikirka **MVC** ee aynu hore ugasoo hadalay.

Waxaynu diyaarinayaa ficilka `changeSessionInformation` ee nooca `NimiController`, kaas oo aad ka arkayso tusaalaha 16.20 ee hoose.

```

package nimi.ui;
2
...
4
import java.util.Date;
6
public class NimiController implements NimiUIListener {
8
    ...
10
    @Override
    public void changeSessionInformation(CourseSession session,
12        Object newValue, int field) {
        switch (field) {
14            case 0:
                session.setCourse((Course) newValue);
16            break;
            case 1:
                session.setStartDate((Date) newValue);
18            break;
            case 2:
                session.setNumberOfCredits((Integer) newValue);
20            break;
            default:
                break;
22        }
24    }
26
    ...
28
}

```

Tusaale 16.20: Badalista macluumaadka fadhiga 2-aad

Intaas kadib kaxee borogaraamka. Haddii aad istidhaahdo qiimeyaasha badal. Marka hore dhibcaha waad badali kartaa. Kaliya waxad siin kartaa nambar (waa cashar isticmaalaha u yaalla in aan dhibcuhu aqbalin nambar ka yar eber). Waxa kale oo aad badali kartaa taariikhda bilawga, laakiin sidaad rabto uma qori kartid. Waa inaad u qorto qaabka maraykanka ee `mm/dd/yyyy` (ama `mm/dd/yyyy hh:mm:ss` ama `mm/dd/yy`). Qaabka taariikhda uu kuugusoo bandhigaayo intaanad badalin waa Sun Jan 01 12:12:12 EET 2012. Badalka taariikhda waa inla fududeeyo, laakiin waa mawduuc ka baxsan buuggan. Waxad Internet-ka kaga raadin kartaa 'Swing Date Picker'.

Intaynaan ka gudbin caleentan waxaynu rabnaa inaynu qabano ugu yaraan laba arimood. Marka hore waa inaynu kaga shaqaysiino badhanada hoose. Marka xigta waa inaynu saamaxno in fadhiga koorsadiisa la badali karo. Waxaynu ku bilaabaynaa inaynu kaga shaqaysiino badannada hoose.

Badhanka ugu horeeya waa "Cusub". Marka isticmaaluhu gujiyo badhankaas waxa laynaga rabaa inaynu fadhi cusub ku darno miiska sare. Laynka 25 ee tusaalaha 16.14, meesha ay ku qorantahay "cusub" waxad ku badashaa qoraalka 16.21

```

package nimi.ui;
2
...
4
public class SessionsTab extends NimiTab {
6
    ...
    @Override
8    protected void addControlButtons() {
        ...
10        JButton button = new JButton("Cusub"); // Cusub

```

```

12     button.addActionListener(new ActionListener() {
13         public void actionPerformed(ActionEvent e) {
14             nimiView.uiListener.addNewSession();
15         }
16     });
17     ...
18 }

```

Tusaale 16.21: Marka la gujiyo badhanka cusub 1-aad

Marka badhanka “Cusub” la gujiyo waxaynu wacaynaa ficilka `addNewSession` ee Controller-ka. Ficilkaas wuu madhanyahay illaa hadda. Badal ficilkaas oo u eekaysii sida tusaalaha 16.22

```

package nimi.controller;
2
...
4
public class NimiController implements NimiUIListener {
6
...
8
    @Override
    public void addNewSession() {
10        CourseSession session = new CourseSession(model
11            .getCourses().get(0), new Date(), 1);
12        model.addSession(session);
13        this.view.updateSessions(this.model.getSessions(),
14            this.model.getCourses());
15    }
16
...
18 }

```

Tusaale 16.22: Marka la gujiyo badhanka cusub 2-aad

Intaas kadib bal guji badhanka “Cusub”. Wakaas! Markaad gujiso badhankaas, waxa miiska sare ku kordhaaya layn cusub, waxa kale oo aad arkaysaa in fadhiga cusubi ku xidhanyahay koorsada ugu horaysa, in uu bilaabmaayo maanta iyo in dhibcaha laga helaayo ay yihiin kow. Fadhigaas waxaynu samaynaynaa laymanka 10-11 ee tusaalaha sare ee 16.22. Samaynta kadib waxaynu ku daraynaa model-ka. Kadibna laynka ugu hooseeya ee 13-aad waxaynu ogaysiinaynaa aragga inuu iscusboonaysiiyo, gaar ahaan fadhiyada.

Waxaynu ku xigsiinaynaa badhanka “Tuur”. Waxa la rabaa in aynu tuurno dhammaan fadhiyada la doortay marka la gujiyo badhankaas. Waxad iyana ku bilawdaa sida tusaalaha 16.23:

```

package nimi.ui;
2
...
4
public class SessionsTab extends NimiTab {
6
...
8
    @Override
    protected void addControlButtons() {
10        button = new JButton("Tuur"); // Tuur
11        button.addActionListener(new ActionListener() {
12            public void actionPerformed(ActionEvent e) {
13                int sel = sessionsTable.getSelectedRow();
14                if (sel != -1) {
15                    nimiView.uiListener.removeSession(sessions.get(sel));
16                }
17            }
18        });
19        ...
20    }
}

```

Tusaale 16.23: Marka la gujiyo badhanka tuur 1-aad

Marka isticmaaluhu gujiyo “Tuur”, waxaynu wacaynaa ficilka `removeSession` ee Controller-ka. Waxa loo baahanyahay sida badhanka “Cusub” oo kale inaynu ficilkaas calooshiisa qorno. Illaa hadda wuu madhanyahay. Ficilkaas waxad u qortaa sida tusaalaha 16.24 ee hoose.

```

package nimi.controller;
2
...

```

```

4 public class NimiController implements NimiUIListener {
6     ...

8     @Override
9     public void removeSession(CourseSession session) {
10         model.getSessions().remove(session);
11         this.view.updateSessions(this.model.getSessions(),
12             this.model.getCourses());
13     }

14     ...
16 }

```

Tusaale 16.24: Marka la gujiyo badhanka tuur 2-aad

Intaas kadib, dooro fadhiga aad rabto inaad tuurto, kadibna guji "Tuur". Wakaas!. Dhawr fadhi oo cusub ku dar, kadibna qaarkood dooro oo tuur, si aad u hubiso inay shaqaynayaan labada badan ee cusub iyo tuur. Sida caadiga ah waa inaad isticmaalaha ka hubiso inuu rabo inuu tuuro fadhiyadaas iyo in kale, kadib haddii uu doorto "haa" aad nidaamka ka tuurto, haddii kale aanad tuurin. Waxaynu cashar ahaan u dhaafnay inaad tuuri karto in ka badan hal fadhi mar kaliya, oo intaad doorato fadhiyada aad rabto inaad tuurto, kadibna gujiyo badhanka tuurista.

Intaynaan ka gudbin caleenta, waxaynu saamaxaynaa in koorsada fadhiga la badali karo. Waxyar ayay ka adag tahay arintani kuwii hore, sababtoo ah waxaynu rabnaa inaynu isticmaalaha tusno JComboBox koorsooyinka ah oo uu ka dooran karo koorsada ay fadhigani ku xidhantahay. Aynu soo bandhigno marka hore waxyaalaha kusoo kordhay qoraalka nooca SessionsTab, oo aad ka arkayso tusaalaha 16.25:

```

package nimi.ui;

2     ...

4 import javax.swing.AbstractCellEditor;
5 import javax.swing.JComboBox;
6 import java.awt.Component;
7 import javax.swing.table.TableCellEditor;
8 import javax.swing.table.TableColumn;

10 public class SessionsTab extends NimiTab {
12     ...

14     @Override
15     protected void addContent() {
16         this.tableModel = new SessionsTableModel();
17         sessionsTable = new JTable(this.tableModel);

18         // badalista koorsada
19         TableColumn courseColumn = sessionsTable.getColumnModel()
20             .getColumn(0);
21         courseColumn.setCellEditor(new ComboBoxCellEditor());

22         ...
23     }

24     ...

28 private class ComboBoxCellEditor extends AbstractCellEditor
29     implements ActionListener, TableCellEditor {
30     private JComboBox combo;

32     @Override
33     public Object getCellEditorValue() {
34         return combo.getSelectedItem();
35     }

36     @Override
37     public Component getTableCellEditorComponent(JTable table,
38         Object value, boolean isSelected, int row, int column) {
39         combo = new JComboBox();
40         for (Course course: courses) {
41             combo.addItem(course);
42         }
43         combo.setSelectedItem(value);
44         return combo;
45     }
46 }
48

```

```

50     @Override
    public void actionPerformed(ActionEvent e) {}
52 }

```

Tusaale 16.25: Badalista koorsooda fadhiga

Aynu sharaxaad ka bixino tusaalaha 16.25:

- Marka hore soo **import**-garee noocyada ka muuqda laymanka 5-9. Waxa ugu muhiimsan nooca JComboBox oo inoo saamaxaaya inaynu liiska koorsooyinka mid ka doorano
- Intaas kadib waxad ficilka addContent ku dartaa laymanka 20-22, laynka 18 kadib. Waxaynu marka hore miiska waydiinaynaa tiirkiisa ugu horeeya
sessionsTable.getColumnModel().getColumn(0), kadibna waxay isticmaalaynaa ficilka setCellEditor, oo aynu u gudbinayno walax ah ComboBoxCellEditor, oo ah nooca aad ka arkayso laymanka 29-51.

16.11 Caleenta - Koorsooyinka

Caleemaha kale waxay aad ugu dhawyihiin caleentii hore ee fadhiyada, gaar ahaan caleenta koorsooyinka. Sidaas awgeed iyada sharaxaad dheer ka bixinmayno. Caleenta koorsooyinka waxad ku bilawdaa inaad ku darto laymanka tusaalaha 16.26 nooca NimiViewImpl.

```

package nimi.ui;
2
...
4
public class NimiViewImpl extends JTabbedPane implements
6     NimiView {
    ...
8     private CoursesTab coursesTab;
10
    ...
12
    @Override
    public void updateCourses(List<Course> courses) {
14         this.coursesTab.updateCourses(courses);
    }
16
    ...
18
    private void addTabs() {
20        ...
        this.coursesTab = new CoursesTab(this);
22        addTab("Koorsooyinka", this.coursesTab);
    }
24 }

```

Tusaale 16.26: Caleenta Koorsooyinka oo comment-kii laga qaaday

Intaas kadib wuxu Eclipse ka cabanayaa inaanu jirin nooca CoursesTab. Noocaas waxaynu rabnaa inuu matalo caleenta koorsooyinka. Waxad ku xigsiisaa inaad noocaas samayso adiga oo adeegsanaaya talaabada NewClass. Waxad ku kaydisaa isla package-ka ay ku jiraan noocyada kale ee aragga, waxaanad ku bilawdaa sida tusaalaha 16.27. Xasuuso inaad ku kordhiso ficilka updateCourses, taas oo loogu talagalay cusboonaysiinta koorsooyinka.

```

package nimi.ui;
2
import java.util.List;
4
import nimi.course.Course;
6
public class CoursesTab extends NimiTab {
8
    public CoursesTab(NimiViewImpl nimiView) {
10        super(nimiView);
        // TODO Auto-generated constructor stub
12    }

14    @Override
    protected void addContent() {
16        // TODO Auto-generated method stub
    }

```

```

18  @Override
20  protected void addControlButtons() {
    // TODO Auto-generated method stub
22  }

24  public void updateCourses(List<Course> courses) {
    // TODO
26  }
}

```

Tusaale 16.27: Bilawga caleenta koorsooyinka

Haddii aad borogaraamka kaxayso waxad arkaysaa inay soo korodhay caleen cusub oo madhan oo magaceedu yahay "Koorsooyinka". Waxaynu rabnaa inaynu saddexda ficil ee addContent, addControlButtons iyo updateCourses qorno, si ay caleentaas caloosheeda koorsooyinkii iyo badhannadii laga hagaayey uga muuqdaan. Waxaynu ku bilaabaynaa ficilka hore iyo ka dambe, waxaanad ka arkaysaa tusaalaha 16.28.

```

package nimi.ui;
2
...
4 import javax.swing.JScrollPane;
import javax.swing.JTable;
6
public class CoursesTab extends NimiTab {
8   private JTable coursesTable;
   private CoursesTableModel tableModel;
10  private List<Course> courses;
12
   ...
14  @Override
   protected void addContent() {
16     this.tableModel = new CoursesTableModel();
     coursesTable = new JTable(this.tableModel);
18
     // Miiska u samee scrollpane kadib ku dhex tuur
20     JScrollPane scrollPane = new JScrollPane(coursesTable);
     // Scrollpane-kii caleenta ku dhextuur
22     add(scrollPane);
   }
24
   ...
26
   public void updateCourses(List<Course> courses) {
28     this.courses = courses;
     tableModel.fireTableDataChanged();
30   }

32  private class CoursesTableModel extends AbstractNimiTableModel {

34     public CoursesTableModel() {
        super(new String[] {"Magac", "Nambar", "Qayb"});
36     }
     @Override
38     public Object getValueAt(int row, int col) {
        Course course = courses.get(row);
40         switch (col) {
            case 0:
42             return course.getName();
            case 1:
44             return course.getNumber();
            case 2:
46             return course.getDepartment();
            default:
48             break;
        }
50         return "";
     }

52     @Override
54     public void setValueAt(Object value, int row, int col) {
        Course course = courses.get(row);
56         nimiView.uiListener.changeCourseInformation(course,
            value, col);
58     }
}

```

```

60     @Override
61     public int getRowCount() {
62         return courses.size();
63     }
64 }

```

Tusaale 16.28: Caleenta koorsooyinka oo miiskii lagu daray

Intaas kadib waa in miiskii koorsooyinku kuusoo baxo, isaga oo ay ku jiraan koorsooyinka model-ku. Macluumaadka koorsada tusaale ahaan magaca, haddii aad badasho, markaad Enter gujiso ayuu sidiisii hore kusoo noqonayaa. Sida kale tusaalaha 16.28 wuxu aad ugu dhawyahay caleentii hore fadhiyada sidaas awgeed sharaxaad badan inoogama baahna. Waxaynu ku xigsiinaynaa inaynu kaga shaqaysiino badalista macluumaadka koorsada. Intii kale waa inoo diyaar ee waxa maqan qaybtii Controller-ka. Waxad Controller-ka ku dartaa caloosha ficilka `changeCourseInformation` sidaad ka arkayso tusaalaha 16.29 ee hoose.

```

package nimi.controller;

...

public class NimiController implements NimiUIListener {
    ...

    @Override
    public void changeCourseInformation(Course course,
        Object newValue, int field) {
        switch (field) {
            case 0:
                course.setName((String) newValue);
                break;
            case 1:
                course.setNumber((String) newValue);
                break;
            case 2:
                course.setDepartment((String) newValue);
                break;
            default:
                break;
        }
    }
}

...
}

```

Tusaale 16.29: Kaga shaqaysii badalista macluumaadka koorsada

Intaas kadib kaxee borogaraamyada oo eeg in macluumaadka koorsada la badali karaayo. Tijaabo ahaan intaad macluumaadka koorsada badasho, waxad tagtaa caleenta fadhiyada, kadibna eeg in macluumaadkii cusbaa ee koorsadu uu dhankaana ka muuqdo. Waxaynu ku xigsiinaynaa badhannada hoose, oo ah caloosha ficilka `addControlButtons`, waxaanad ka arkaysaa tusaalaha 16.30 ee hoose.

```

package nimi.ui;

...

import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.BoxLayout;
import javax.swing.Box;
import javax.swing.BorderFactory;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CoursesTab extends NimiTab {
    ...

    @Override
    protected void addControlButtons() {
        JPanel buttons = new JPanel();
        buttons.setLayout(new BoxLayout(buttons, BoxLayout.LINE_AXIS));
        JButton button = new JButton("Cusub");
        buttons.add(button);
        button.addActionListener(new ActionListener() {

```

```

24     public void actionPerformed(ActionEvent e) {
25         nimiView.uiListener.addNewCourse();
26     }
27 });
28 button = new JButton("Tuur");
29 buttons.add(button);
30 button.addActionListener(new ActionListener() {
31
32     public void actionPerformed(ActionEvent e) {
33         int sel = coursesTable.getSelectedRow();
34         if (sel != -1) {
35             nimiView.uiListener.removeCourse(courses.get(sel));
36         }
37     }
38 });
39 buttons.add(Box.createHorizontalGlue());
40 buttons.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
41 add(buttons);
42 }
43
44 ...
45 }

```

Tusaale 16.30: Badhannada oo lagu daray caleenta koorsooyinka

Markaad eegto noocyada Swing ee aynu adeegsanay ee laymanka 5-11 ee JPanel, JButton, BoxLayout, Box, BorderLayout, ActionListener iyo ActionEvent, waxad arkaysaa inay yihiin isla noocyadii aynu ku isticmaalay caleenta fadhiyada. Sidaas awgeed waxad arkaysaa inay waxbadan ka midaysanyihiin labada caleemood. Sidaas awgeed sharaxaadda waxaynu u dhaafnay caleenta hore.

Badhannada midkoona ma shaqaynaayo. Waxaynu bilaabaynaa inaynu kaga shaqaynsiino. Sabab-tuna waxa weeye Controller-ka ficiladeeda addNewCourse iyo removeCourse oo aan wali la qorin. Tusaalaha 16.31 ee hoose ayaad ka arkaysaa labadaas ficil oo la qoray. Ka raadi labadaas ficil Controller-ka kadibna qor labadaas ficil calooshooda.

```

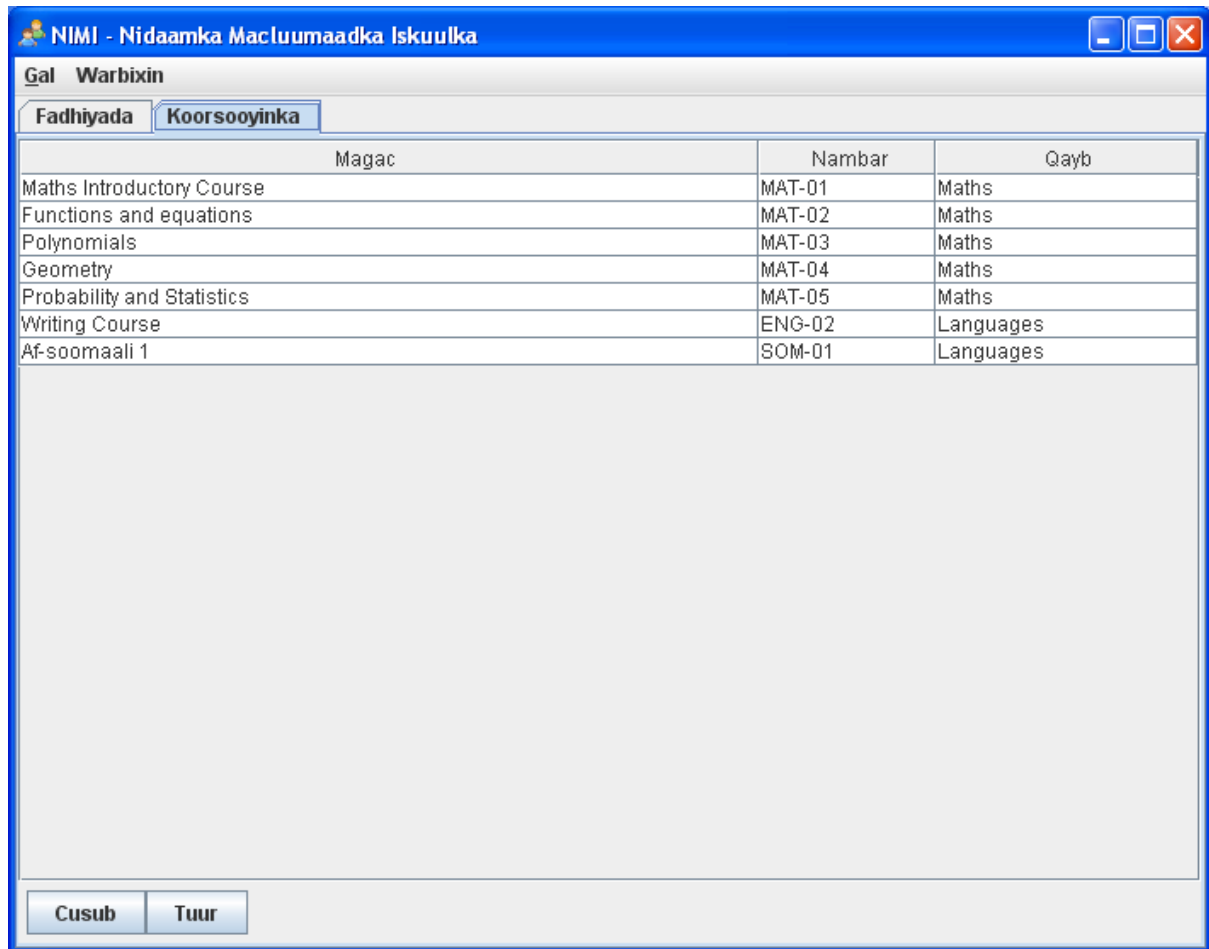
1 package nimi.controller;
2
3 ...
4 import nimi.course.IllegalCourseNumberException;
5
6 public class NimiController implements NimiUIListener {
7     ...
8
9     @Override
10    public void addNewCourse() {
11        Course course;
12        try {
13            course = new Course("<enter name>", "ABC-00",
14                                "<enter department>");
15            model.addCourse(course);
16        } catch (IllegalCourseNumberException e) {
17            e.printStackTrace();
18        }
19        this.view.updateCourses(this.model.getCourses());
20    }
21
22    ...
23
24    @Override
25    public void removeCourse(Course course) {
26        model.getCourses().remove(course);
27        this.view.updateCourses(this.model.getCourses());
28    }
29 }

```

Tusaale 16.31: Kaga shaqaysii badhannada caleenta koorsooyinka

Intaas kadib, kaxee borogaraam. Haddii aad gujiso badhanka cusub waa in koorso cusub kusoo biirto koorsooyinka. Ficilka addNewCourse uma baahna sharxaad sidaas u dheer waayo wuu iska cadyahay. Marka hore waxaynu samaynaynaa koorso cusub, taas oo macluumaadkeeda uu isticmaaluhu badali karo, kadibna waxaynu adeegsanaynaa ficilka addCourse ee model-ka si aynu ugu kordhino koorsooyinka model-ka. Kadib, waxaynu amraynaa aragga gaar ahaan qaybta koorsooyinka inuu is cusboonaysiiyo innaga oo adeegsanaynaa ficilka updateCourses. Marka la gujiyo badhanka tuurista araggu wuxu soo wacayaa ficilka removeCourse ee Controller-ka taas oo kooradii la siiyey marka hore ka tuuraysa model-ka, kadibna

cusboonaysiinaysa muuqaalka koorsooyinka. Xaaladda uu borogaraamku maraayo illaa hadda waa inay u eekaato sida sawirka 16.12 oo kale.



Sawirka 16.12: Caleenta koorsooyinka

16.12 Caleenta - Ardayda

Ugu dambayntii waxaynu diyaarinaynaa caleenta ardayda. Caleenta ardaydu guud ahaan waxay la midtahay caleemihii hore oo kale, laakiin waxyaalo yaryar ayay kaga duwantahay. Waxyaalaha ay caleemihii hore kala midtahay waxa kamid ah miiska. Sidiis caleemihii hore ayay miis leedahay. Miiskaas oo soo bandhigaaya macluumaadka ardayda. Waxyaalaha ay caleemihii hore kaga duwantahay waxa kamid ah, in aad laba kala dooran karto; dhammaan ardayda nidaamka ku jirta iyo kaliya ardayda ku jirta fadhiga aad dooratay. Taas oo kuu saamaxaysa inaad miiska ardayda ku aragto kaliya ardayda ku jirta fadhiga aad dooratay ama ardayda oo dhan. Caleenta ardaydu waxa kale oo ay saamaxaysaa inaad arday ka register-garayso ama ka diiwaangeliso fadhi koorso iyo inaad nidaamka ku darto arday cusub. Waxa kale oo aad ardayga ka tuuri kartaa fadhiga ama nidaamka oo dhan. Halkaas waxad ka arkaysaa inay caleentani ka adagtahay kuwii hore waxyar.

Qorista caleenta ardayda waxad ku bilawdaa inaad nooca caleemaha ee NimiViewImpl ku darto laymanka tusaalaha 16.32 sida caleemihii hore oo kale.

```

1 package nimi.ui;
2
3 ...
4
5 public class NimiViewImpl extends JTabbedPane implements
6     NimiView {
7     ...
8     private StudentsTab studentsTab;
9
10    ...
11
12    @Override

```

```

14     public void updateStudents(List<Student> students ,
        List<CourseSession> sessions) {
16         this.studentsTab.updateStudents(students , sessions);
18     }
19     ...
20     private void addTabs() {
21         ...
22         this.studentsTab = new StudentsTab(this);
23         addTab("Ardayda", this.studentsTab);
24     }
25 }

```

Tusaale 16.32: Bilawga caleenta ardayda

Intaas kadib, wuxu Eclipse ka cabanayaa inaanu jirin nooca StudentsTab, kaas oo aynu halkan ku diyaarindoono. Samee nooc cusub, una bixi StudentsTab, waxaanad ku kaydisaa isla package-ka caleemaha kale. Waxad noocaas ku bilawdaa sida tusaalaha 16.33.

```

1 package nimi.ui;
2
3 import java.util.List;
4 import nimi.student.Student;
5 import nimi.course.CourseSession;
6
7 public class StudentsTab extends NimiTab {
8
9     public StudentsTab(NimiViewImpl nimiView) {
10         super(nimiView);
11         // TODO Auto-generated constructor stub
12     }
13
14     @Override
15     protected void addContent() {
16         // TODO Auto-generated method stub
17     }
18
19     @Override
20     protected void addControlButtons() {
21         // TODO Auto-generated method stub
22     }
23
24     public void updateStudents(List<Student> students ,
        List<CourseSession> sessions) {
26     }
27 }

```

Tusaale 16.33: Bilawga nooca caleenta ardayda

Haddii aad kaxayso borogaraamka waa inaad aragto inay soo korodhay caleenta ardaydu, laakiin caloosheedu ay madhantahay. Waxaynu gudagalaynaa diyaarinta caloosheeda. Waxaynu ku bilaabaynaa badhanada hoose ee caleenta laga maamulaayo. Tusaalaha 16.34 ayaad ka arkaysaa qoraalka badhanada.

```

1 package nimi.ui;
2
3 ...
4
5 import javax.swing.JPanel;
6 import javax.swing.JButton;
7 import javax.swing.BoxLayout;
8 import javax.swing.Box;
9 import javax.swing.BorderFactory;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12
13 public class StudentsTab extends NimiTab {
14     private JButton buttonNewEnroll;
15     private JButton buttonDeleteDrop;
16
17     ...
18
19     @Override
20     protected void addControlButtons() {
21         JPanel buttons = new JPanel();
22         buttons.setLayout(new BoxLayout(buttons ,
23             BoxLayout.LINE_AXIS));

```

```

25     buttonNewEnroll = new JButton("Cusub");
    buttonNewEnroll.addActionListener(new ActionListener() {
27         public void actionPerformed(ActionEvent e) {
            //cusub ama kudar
        }
29     });
    buttons.add(buttonNewEnroll);
31     buttonDeleteDrop = new JButton("Tuur");
    buttonDeleteDrop.addActionListener(new ActionListener() {
33         public void actionPerformed(ActionEvent e) {
            // tuur ama ka tuur
35         }
        });
37     buttons.add(buttonDeleteDrop);
    buttons.add(Box.createHorizontalGlue());
39     buttons.setBorder(BorderFactory.createEmptyBorder(5, 5, 5,
        5));
41     add(buttons);
    }
43
45     ...
}

```

Tusaale 16.34: Bilawga nooca caleenta ardayda, oo lagu daray badhanadii hoose

Haddii aad u fiirsato tusaalaha 16.34 waxa soo kordhay caloosha ficilka addControlButtons iyo labada siifood ee buttonNewEnroll iyo buttonDeleteDrop, oo labadooduba ah JButton. Sidaad arkidoontaba labadan badhan waxay qabanayaan laba shaqo oo kala duwan. Badhanka hore ee buttonNewEnroll wuxu tusaale ahaan qabanayaa laba shaqo oo kala ah: kudar arday cusub nidaamka (waa marka la doortay liiska ardayda oo dhan) iyo fadhigan la doortay ka diiwaangeli arday ku jira ardayda aan illaa hadda ka diiwaangashanayn. Badhanka kale ee buttonDeleteDrop wuxu isna qabanayaa laba arimood oo kala ah: ka tuur nidaamka arday iyo ka tuur fadhiga arday (laakiin haka tuurin nidaamka oo dhan). Marka ugu horaysa waxay caleentani soo bandhigaysaa liiska ardayda oo dhan, sidaas awgeed labadaas badhan waxay ku tusayaan "Cusub" iyo "Tuur". Marka isticmaaluhu doorto liiska ardayda ku jirta fadhiga, waxay isu badalayaan "Kudar" iyo "Ka Tuur" sidaynu arkidoono.

Waxaynu ku xigsiinaynaa qorista ficilka addContent kaas oo diyaarinaaya qaybta badhanada ka saraysa. Waxaanu ku bilaabmayaa sida tusaalaha 16.35 ee hoose:

```

package nimi.ui;
2
...
4
import javax.swing.JRadioButton;
6 import javax.swing.JComboBox;
import javax.swing.ButtonGroup;
8 import javax.swing.JTable;
import javax.swing.JScrollPane;
10 import java.awt.GridLayout;

12 public class StudentsTab extends NimiTab {
    ...
14
    private JRadioButton allStudentsButton;
16     private JRadioButton sessionStudentsButton;
    private JComboBox sessionsCombo;
18
    private List<Student> allStudents;
20     private List<Student> displayedStudents;
    private List<CourseSession> sessions;
22
    private StudentsTableModel studentsTableModel;
24
    ...
26
    private void addRadioButtons() {
28         allStudentsButton = new JRadioButton("Dhammaan Ardayda");
        allStudentsButton.setSelected(true);
30         allStudentsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
32             buttonNewEnroll.setText("Cusub");
            buttonDeleteDrop.setText("Tuur");
34         }
        });
36         sessionStudentsButton = new JRadioButton("Kujirta Fadhiga");

```

```

        sessionStudentsButton
38         .addActionListener(new ActionListener() {
40             public void actionPerformed(ActionEvent e) {
42                 buttonNewEnroll.setText("Kudar");
43                 buttonDeleteDrop.setText("Ka Tuur");
44             }
45         });

46     ButtonGroup group = new ButtonGroup();
47     group.add(allStudentsButton);
48     group.add(sessionStudentsButton);

49     sessionsCombo = new JComboBox();
50     sessionsCombo.setEnabled(false);
51     JPanel bySessionPanel = new JPanel();
52     bySessionPanel.setLayout(new BoxLayout(bySessionPanel,
53         BoxLayout.X_AXIS));
54     bySessionPanel.add(sessionStudentsButton);
55     bySessionPanel.add(sessionsCombo);
56     bySessionPanel.add(Box.createHorizontalGlue());
57     // Put the radio buttons in a column in a panel.
58     JPanel radioPanel = new JPanel(new GridLayout(0, 1));
59     radioPanel.add(allStudentsButton);
60     radioPanel.add(bySessionPanel);
61     add(radioPanel);
62 }

63 private void addTable() {
64     this.studentsTableModel = new StudentsTableModel();
65     JTable studentsTable = new JTable(this.studentsTableModel);

66     // Miiska u samee scrollpane kadib ku dhex tuur
67     JScrollPane scrollPane = new JScrollPane(studentsTable);
68     // Scrollpane-kii caleenta ku dhextuur
69     add(scrollPane);
70 }

71 @Override
72 protected void addContent() {
73     addRadioButtons();
74     addTable();
75 }

76 ...

77 public void updateStudents(List<Student> students,
78     List<CourseSession> sessions) {
79     this.sessions = sessions;
80     this.allStudents = students;
81     this.displayedStudents = this.allStudents;
82     studentsTableModel.fireTableDataChanged();
83 }

84 ...

85 private class StudentsTableModel extends
86     AbstractNimiTableModel {

87     public StudentsTableModel() {
88         super(new String[] { "ID", "Magac", "Telefoon" });
89     }

90     @Override
91     public Object getValueAt(int row, int col) {
92         Student student = displayedStudents.get(row);
93         Object ret = null;
94         switch (col) {
95             case 0:
96                 ret = student.getId();
97                 break;
98             case 1:
99                 ret = student.getName();
100                 break;
101             case 2:
102                 ret = student.getPhone();
103                 break;

```

```

114     default :
115         break;
116     }
117     if (ret == null) {
118         return "";
119     } else {
120         return ret;
121     }
122 }

124 @Override
125 public void setValueAt(Object value, int row, int col) {
126     Student student = displayedStudents.get(row);
127     nimableView.uiListener.changeStudentInformation(student,
128         value, col);
129 }

130 @Override
131 public int getRowCount() {
132     return displayedStudents.size();
133 }
134 }
135 }

```

Tusaale 16.35: Nooca caleenta ardayda oo lagu daray qaybta ka saraysa badhanada maamulka

Tusaalaha sare ee 16.35 wuxu ka koobanyahay in ka badan 130 layn. Inkastoo uu aad u dheeryahay tusaalahaasi, waxaynu isku dayaynaa inaynu ka bixino sharaxaad fiican. *Lines of code* ama *source lines of code* waa qiyaas lagu qiyaaso inta uu leegyahay borogaraamka kumbuyuutarku. Waxa la tirinayaa laymanka dhammaan faylasha source code-ka borogaraamka oo laysu geeyey. Borogaraamkan yar ee aynu halkan ku dhisayno wuxu ka koobanyahay qiyaastii laba kun oo layn. Haddii aynu tusaale u soo qaadano borogaraamyada maanta la sameeyo qaarkood; Visual Studio 2012 (oo ah IDE-da lagu sameeyo borogaraamyada loogu talagalay Windows-ka) qiyaastii 50milyan oo layn, Microsoft Office (oo caloosha ku haysata borogaraamyada Excel, Word, PowerPoint, iwm) qiyaastii 30-40milyan oo layn oo C++ ah, Eclipse qiyaastii 2 milyan oo Java ah, LibreOffice qiyaastii 12 milyan oo layn, Windows XP qiyaastii 45milyan oo layn¹, Linux kernel 3.6 qiyaastii 16milyan oo layn². Halkaas waxad ka arkaysaa in qoraalka borogaraamku uu aad u waynaan karo oo uu ka koobnaankaro malaayiin layn. Luuqadda Java waxa dadka lagula taliyaa in aanu halkii fayl ka dheeraan 2000 oo layn³. Haddii aynu u soo noqono tusaalaha 16.35, kana bixino sharaxaad kooban:

- Laymanka 5-10 waxad ka arkaysaa noocyada aynu u baahanay ee aynu soo **import**-garaynay. Markaad isha mariso waxad arkaysaa inaynu adeegsanayno ugu yaraan JRadioButton, JComboBox, JTable iyo noocyo kale oo kalkaaliyaal ah oo uu ka mid yahay nooca ButtonGroup
- Ficilkii addContents oo aad ka arkayso laymanka 76-79, waxad arkaysaa inuu u kala jabay laba ficil oo kala ah addRadioButtons iyo addTable. Ficilka hore wuxu diyaarinayaa waxyaalaha ka sareeya miiska, ficilka dambena wuxu diyaarinayaa miiska ardayda
- Ficilka addRadioButtons oo aad ka arkayso laymanka 27-63, wuxu diyaarinayaa badhanada sare ee ah JRadioButton sidaan soo sheegayba. Ficilkani wuxu ku tusayaa sidaad u adeegsankarto noocyada JRadioButton iyo JComboBox. Badhanka dambe waxa ka dambeeya JComboBox, kaas oo aynu rabno inaynu kusoo bandhigno liiska fadhiyada si uu isticmaaluhu u doorto fadhiga uu rabo inuu arko ardayda ku jirta. Badhanadan radio-ga ah marka la doorto waxay badalayaan qoraalka badhanada hoose ee maamulka, waxaanay arintaasi ku imanaysaa inaynu badhan kasta ku dhajino dhagayste innaga oo isticmaalayna ficilka addActionListener
- Ardaydii waxad ka arkaysaa miiska. Waxaynu rabnaa inaynu miiskaas kusoo bandhigno laba nooc oo ardayda ah. Marka isticmaaluhu doorto "Dhammaan Ardayda" waxaynu kusoo bandhigaynaa dhammaan ardayda ku jirta nidaamka, laakiin marka isticmaaluhu doorto "Kujirta Fadhiga" waxaynu miiskaas kusoo bandhigaynaa kaliya ardayda ku jirta fadhigaas
- Maadaama doorashooyinku ka badanyihiin caleemihii hore waxaynu u baahanahay sifooyin kala duwan oo kolba aynu ku kaydino waxa uu isticmaaluhu doortay. Saddexda sifo ee laymanka 15-17 waxay kala matalayaan badhanada iyo combobox-ka. Saddexda sifo ee laymanka 19-21 waxay kala matalayaan liiska ardayda oo dhan (allStudents), ardayda isticmaalaha la tusaayo (displayedStudents)

¹<http://www.facebook.com/windows/posts/155741344475532>

²<http://www.h-online.com/open/features/What-s-new-in-Linux-3-6-1714690.html?page=3>

³<http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-141855.html>

iyo fadhiyada. Sifada laynka 23 ee `studentsTableModel` waa modelkii miiska. Nooca matalaaya model-kaas waxad ka arkaysaa laymanka 93-122. Model-kaasi wuxu u dhigmaa kuwii hore oo kale

- Ficilka `updateStudents` oo aad ka arkaysaa laymanka 83-89, wuxu cusboonaysiinayaa caleentan, waxaana soo wacaysa `Controller`-ka.

Kaxee borogaraamka si aad u aragto halka xaaladdu marayso. Waxad arkaysaa in badhanada sare ay badalayaan kaliya qoraalka badhanada hoose. Waxa la rabaa inay wax ka badalaan combobox-ka fadhiyada iyo miiska ardayda. Waxa kale oo aad arkaysaa in labadii arday ee model-ka tijaabada ah ku jiray ay muuqdaan. Waxaynu rabnaa inay kaga shaqaysiiyo badhanada sare. Waxa kale oo soo socda oo aad arki doontaa sida loo adeegsado dialog-yada, oo ah daaquadaha yaryar ee soo booda.

Waxaynu ku xigsiinaynaa badalista macluumaadka ardayda maadaama ay aad u fududahay. Waxad `Controller`-ka ku dartaa laymanka tusaalaha 16.36, oo ah caloosha ficilka `changeStudentInformation`.

```

1 package nimi.controller;
2
3 ...
4
5 public class NimiController implements NimiUIListener {
6     ...
7
8     @Override
9     public void changeStudentInformation(Student student,
10         Object newValue, int field) {
11         switch (field) {
12             case 0:
13                 student.setId((Integer) newValue);
14                 break;
15             case 1:
16                 student.setName((String) newValue);
17                 break;
18             case 2:
19                 student.setPhone((String) newValue);
20                 break;
21             default:
22                 break;
23         }
24     }
25
26     ...
27
28 }

```

Tusaale 16.36: Kaga shaqaysii badhannada caleenta koorsooyinka

Intaas kadib waa inaad badali karto macluumaadka ardayga. Ficilka `changeStudentInformation` wuxu eegayaa sifada aad rabto inaad badasho, kadibna qiimaha cusub ee la siiyey ayuu sifadaas ka dhigayaa isaga oo adeegsanaay ficilka lagu badalaayo sifadaas oo ka bilaabma `set*`. Intaas kadib araggu waa inuu u eekaado sida sawirka 16.13.

Sida sawirkaas caleenta ardayda ka muuqata, muuqaal ahaan caleentaasi waa inoo diyaar. Laakiinse qaybaheeda kala duwan ayay wax ka dhimanyihiin, gaar ahaan badhanada kuwa sare iyo kuwa hooseba.

Waxaynu ku xigsiinaynaa inaynu kaga shaqaysiino badhanada sare. Waa kuwa goobada ah oo loo yaqaano radio buttons. Badhanadaas caynka la rabo inay u shaqeeyaan waan soo sharaxay, oo ku laabanmaayo. Waxad nooca `StudentsTab` ku kordhisaa laymanka tusaalaha 16.37.

```

1 package nimi.ui;
2
3 ...
4
5 public class StudentsTab extends NimiTab {
6     ...
7
8     private void addRadioButtons() {
9         ...
10        allStudentsButton.addActionListener(new ActionListener() {
11            public void actionPerformed(ActionEvent e) {
12                ...
13                sessionsCombo.setEnabled(false);
14                displayedStudents = allStudents;
15                studentsTableModel.fireTableDataChanged();
16            }
17        });
18        sessionStudentsButton = new JRadioButton("Kujirta Fadhiga");

```

NIMI - Nidaamka Macluumaadka Iskuulka

Gal Warbixin

Fadhiyada Koorsooyinka **Ardayda**

☒ Dhammaan Ardayda
☐ Kujirta Fadhiga

ID	Magac	Telefoon
1	Cabdilaahi Maxamed Muuse	
2	Khadar Cabdilaahi Muuse	

Tuur Kudar

Sawirka 16.13: Caleenta ardayda

```

    sessionStudentsButton
20      .addActionListener(new ActionListener() {
22          public void actionPerformed(ActionEvent e) {
23              ...
24              if (sessionsCombo.getSelectedIndex() != -1) {
25                  sessionsCombo.setEnabled(true);
26                  CourseSession session = (CourseSession) sessionsCombo
27                      .getSelectedItem();
28                  displayedStudents = session.getAllStudents();
29                  studentsTableModel.fireTableDataChanged();
30              }
31          }
32      });
33
34      ...
35
36      ...
37
38      public void updateStudents(List<Student> students,
39          List<CourseSession> sessions) {
40          this.sessions = sessions;
41          this.allStudents = students;
42          updateSessions();
43          this.displayedStudents = this.allStudents;
44          studentsTableModel.fireTableDataChanged();
45      }
46
47      ...
48
49      private void updateSessions() {
50          sessionsCombo.removeAllItems();
51          for (CourseSession session : sessions) {
52              sessionsCombo.addItem(session);
53          }
54      }
55  }
56

```

Tusaale 16.37: Kaga shaqaysii badhannada sare ee caleenta ardayda

Aynu sharaxaad ka bixino waxyaalaha kusoo kordhay nooca StudentsTab ee tusaalaha sare ee 16.37:

1. Ficilka updateStudents ee ka bilaabmaaya laynka 39, waxad ku kordhinaysaa laynka 43 oo ah wicista ficilka hoose ee updateSessions
2. Ficilka updateSessions ee ka bilaabmaaya laynka 50, wuxu cusboonaysiinayaa fadhiyada ku jira combobox-ka sare ee sessionsCombo. Ficilkaasi wuxu marka hore ka daadinayaa fadhiyadii ku jirtay combobox-kaas, waxaanu ku badalyaa kuwa cusub ee la siiyey
3. Labada radio badhan ee sare ayaa iyana wax kusoo kordheen marka la gujiyo. Badhanka sare ee allStudentsButton marka la gujiyo waxa kusoo kordhay laymanka 13-15. Laynka 13-aad ee sessionsCombo.setEnabled(false) wuxu combobox-ka fadhiyada ka dhigayaa si aan la dooran karin, kadibna ardayda miisku soo bandhigaayo ee displayedStudents ayuu ka dhigayaa ardaydii oo dhan, kadib wuxu miiska u sheegayaa in macluumaadkii is badalay oo uu is cusboonaysiiyo studentsTableModel.fireTableDataChanged
4. Marka badhanka hoose ee "Kujirta Fadhiga" la gujiyo, waxaynu combobox-ka waydiinaynaa fadhiga la doortay waa laynka 26, kadibna waxaynu fadhigaas waydiinaynaa ardaydiisa, kuwaas oo aynu ka dhigayno kuwa uu miisku soo bandhigaayo, kadibna mar kale ayaynu miiska u sheegaynaa inuu is cusboonaysiiyo. Nooca JComboBox waxa lagu guri karaa nooc kasta, sidaas awgeed marka la waydiiyo qiimaha la doortay iyada oo la adeegsanaayo ficilkiisa getItem() sida laynka 26 aynu ku waydiinayno fadhiga la doortay, wuxu soo celinayaa Object. Maadaama aanu garanaynin walaxdan nooceeda saxda ah wuxu soo celinayaa Object (walax kastaaba waa Object luuqadda Java, maadaama uu yahay nooca laga soo dhiraandiriyey noockasta), sidaas awgeed waa inaad u badasho noocii aad rabtay ee aad ku gurtay nooca JComboBox, adiga oo sida tusaalaha sare xero galinayaa nooca aad u badalayso qiimaha uuso celiyey ficilkaasi sida (CourseSession). Arintan waxa loo yaqaan luuqadda Java *Type Casting*.

Haddii aad caleenta fadhiyada fadhi cusub ku darto, kama muuqanaayo fadhigaasi markiiba caleenta ardayda. Waxaynu ku xigsiinaynaa in marka fadhi cusub nidaamka lagu daro, inuu ka muuqdo markiiba caleenta ardaydana. Waxad nooca StudentsTab, ku dartaa ficilka updateSessions ee ka muuqada tusaalaha hoose ee 16.38.

```

package nimi.ui;
2
...
4
public class StudentsTab extends NimiTab {
6
...
8
    public void updateSessions(List<CourseSession> sessions) {
        this.sessions = sessions;
        sessionsCombo.removeAllItems();
        for (CourseSession session : sessions) {
10
            sessionsCombo.addItem(session);
12
        }
14
    }
}

```

Tusaale 16.38: Marka fadhi cusub nidaamka lagu daro waa inuu ka muuqdo caleenta ardaydana 1-aad

Intaas kadib, waxa isna loo baahanyahay in nooca NimiViewImpl hal layn lagu daro. Waxad laynka 12-aad ee tusaalaha 16.39, ku dartaa noocaas, gaar ahaan ficilka updateSessions.

```

package nimi.ui;
2
...
4
public class NimiViewImpl extends JTabbedPane implements
6
    NimiView {
    ...
8
    public void updateSessions(List<CourseSession> sessions,
10
        List<Course> courses) {
        this.sessionsTab.updateSessions(sessions, courses);
        this.studentsTab.updateSessions(sessions);
12
    }
14
    ...
16
}

```

Tusaale 16.39: Marka fadhi cusub nidaamka lagu daro waa inuu ka muuqdo caleenta ardaydana 2-aad

Intaas kadib, haddii aad fadhiyada ku darto fadhi cusub, wuxu isla markiiba soo galayaa caleenta ardayda. Aynu ka socono. Fadhiyadii way bateen, oo markasta oo aad fadhi cusub diiwaangeliso wuxu ka muuqanayaa caleenta ardayda iyo caleenta fadhiyada. Laakiin waxa jirta hal cilad. Caleenta ardayda haddii aad doorato "Kujirta Fadhiga", kadibna aad badasho fadhi oo doorato mid kale isma badalayso liiska ardaydu! Miyaad garanaysaa sababta? Waxa weeye combobox-ka fadhiyada dhagayste kumaynaan dhajin, oo marka la badalo fadhiga wuxu dhici lahaa maynaan qeexin. Ogow, sideedaba kumbuyuutarku ma qayaan "waxaas ayuu rabaa isticmaaluhu ee aan u qabto", iskii waxba uma qabankaro, wax walba waa in loo sheego. Waa inaynu u sheegno waxa dhacaaya marka la badalo fadhiga. Waxad nooca StudentsTab ku dartaa laymanka 12-23 ee tusaalaha hoose ee 16.40, waxad ku kordhisaa laymankaas meesha aynu ku samaynay combobox-ka fadhiyada gaar ahaan laynka 11 hoostiisa, ee ficilka addRadioButtons.

```

package nimi.ui;
2
...
4
public class StudentsTab extends NimiTab {
6
...
8
    private void addRadioButtons() {
        ...
10
        sessionsCombo = new JComboBox();
        sessionsCombo.setEnabled(false);
        sessionsCombo.addActionListener(new ActionListener() {
12
            public void actionPerformed(ActionEvent e) {
14
                if (sessionStudentsButton.isSelected()) {
                    CourseSession session = (CourseSession) sessionsCombo
16
                        .getSelectedItem();
                    if (session != null) {
18
                        displayedStudents = session.getAllStudents();
                        studentsTableModel.fireTableDataChanged();
20
                    }
                }
22
            }
        });
}

```

```

24
26 }
28 ...
}
```

Tusaale 16.40: Marka la badalo fadhiga waa inaynu cusboonaysiino miiska ardayda oo tusno ardayda ku jirta fadhigaas

Intaas kadib, haddii aad badasho fadhiga waxad arkaysaa in miiska ardaydu isbadalaayo. Kaliya fadhiga ugu horeeya ayuu hal arday ka diiwaangashanyahay. Kuwa kale wax arday ah kama diiwaangashana, sidaas awgeed markaad doorato miiska ardaydu wuu madhanyahay. Intaas kadib, waxaynu rabnaa inaynu kaga shaqaysiino labada badhan ee hoose.

16.12.1 Arday cusub

Markaynu arday cusub ku darayno nidaamka, waxaynu yeelaynaa laba arimood, oo ku xidhan kolba xaaladda aynu ku jirno. Haddii ay inoo doorantahay liiska ardayda oo dhan, way fududay ku darista arday cusub. Waxa loo baahanyahay kaliya inaynu ku darno miiska ardayda layn cusub. Kudarista kadib, waxaynu isticmaalaha u daynaynaa inuu badalo id-da, magaca iyo telefoonka ardaygaas. Laakiin, haddii ay inoo doorantahay ardayda ku jirta fadhiga, waxa loo baahanyahay inaynu isticmaalaha tusno liiska ardayda, kadibna uu isagu ka doorto ardayga uu rabo inuu ku daro ama ka diiwaangeliyo fadhigaas. Laakiin, waxa muhiim ah inaanay liiskaas ku jirin, ardayda hore ugusii jirtay ama uga diiwaangashanayd fadhigaas.

Waxay ku bilaabaynaa ku darista arday cusub nidaamka. Waxad dhagaystaha badhanka buttonNewEnroll, gaar ahaan ficilka actionPerformed ku dartaa laymanka 13-17 ee tusaalaha 16.41 ee hoose.

```

package nimi.ui;
2
...
4
public class StudentsTab extends NimiTab {
6
...
8
    private void addRadioButtons() {
        ...
10        buttonNewEnroll = new JButton("Cusub");
        buttonNewEnroll.addActionListener(new ActionListener() {
12            public void actionPerformed(ActionEvent e) {
                if (allStudentsButton.isSelected()) {
14                    nimiView.uiListener.addNewStudent();
                } else {
16                    // ka diiwaangeli fadhiga
                }
18            }
        });
20        ...
    }
22
    ...
24 }
```

Tusaale 16.41: Badhanka cusub ama kudar ee caleenta ardayda. Arday cusub nidaamka kudar 1-aad

Sidaad tusaalahaas sare ee 16.41 ka arkayso, marka badhanka Cusub/Kudar la gujiyo waxaynu eegaynaa in badhanka radio-ga ah ee allStudentsButton la doortay, innaga oo adeegsanayna ficilkiisa isSelected sida ka muuqata laynka 13-aad, haddii ficilkaasi soo celiyo true oo ka dhigan waa la doortay, waxaynu wacaynaa ficilka addNewStudent ee Controller-ka, sida ka muuqata laynka 14-aad. Haddii aad Controller-ka eegto qoraalka, waxad arkaysaa in ficilkaasi madhanyahay. Waxad ficilkaas calooshiisa ka dhigtaa sida tusaalaha 16.42 ee hoose.

```

package nimi.controller;
2
...
4
public class NimiController implements NimiUIListener {
6
...
8
    @Override
    public void addNewStudent() {
10        Student student = new Student(0, "<balad magaca>");
```

```

    model.addStudent(student);
12    this.view.updateStudents(this.model.getStudents(),
        this.model.getSessions());
14    }
16    ...
}

```

Tusaale 16.42: Badhanka cusub ama kudar ee caleenta ardayda. Arday cusub nidaamka kudar 2-aad

Tusaalaha 16.42 aad ayuu u fududyahay. Marka hore waxaynu samaynaynaa arday cusub, kadibna waxaynu ku daraynaa model-ka borogaraamka. Intaas kadib, waxaynu wacaynaa ficilka `updateStudents` ee muuqaalka si uu u cusboonaysiiyo liiska ardayda. Ficilkaas waxaynu u gudbinaynaa liiska ardayda iyo liiska fadhigada.

Waxaynu ku xigsiinaynaa ka diiwaangelinta ama ku darista ardayda fadhiga la doortay, waa marka la doortay badhanka "Kujirta Fadhiga" ee sare. Waxaynu rabnaa inaynu isticmaalaha u soo bandhigno liiska arday aan ku jirin fadhigan, kadibna uu isagu ka doorto kuu rabo inuu ku daro ama ka diiwaangeliyo fadhiga. Sidaad arkidoontaba waxay halkan ku arkidoonaa isticmaala *Dialogs*, oo daaqadaha soo booda ee wax laga doorto ama arin lagu yeelo ama lagu diido. Waxad ku bilawdaa sida tusaalaha 16.43.

```

package nimi.ui;
2
...
4 import java.util.ArrayList;
import javax.swing.JOptionPane;
6
public class StudentsTab extends NimiTab {
8
    ...
10 private void addRadioButtons() {
    ...
12    buttonNewEnroll = new JButton("Cusub");
    buttonNewEnroll.addActionListener(new ActionListener() {
14        public void actionPerformed(ActionEvent e) {
            if (allStudentsButton.isSelected()) {
16                nimiView.uiListener.addNewStudent();
            } else {
18                CourseSession selectedSession = (CourseSession) sessionsCombo
                    .getSelectedItem();
20                Student student = pickStudent(selectedSession);
                if (student!=null) {
22                    nimiView.uiListener.enrollStudent(selectedSession, student);
                    studentsTableModel.fireTableDataChanged();
24                }
            }
26        }
    });
28    ...
}
30
...
32 private Student pickStudent(CourseSession selectedSession) {
34     if (selectedSession != null) {
        List<Student> enrolledStudents = ((CourseSession) selectedSession)
36            .getAllStudents();
        List<Student> toShow = new ArrayList<Student>();
38        for (Student student : allStudents) {
            if (!enrolledStudents.contains(student)) {
40                toShow.add(student);
            }
42        }

        Student s = (Student) JOptionPane.showInputDialog(this,
44            "Dooro ardayga aad rabto inaad ku darto fadhiga.",
            "Dooro arday", JOptionPane.PLAIN_MESSAGE, null,
46            toShow.toArray(new Object[0]), null);
48        return s;
    }
50    return null;
}
52 }

```

Tusaale 16.43: Badhanka cusub ama kudar ee caleenta ardayda. Arday kudar ama ka diiwaangeli fadhiga la doortay 1-aad

Haddii aan sharaxaad ka bixiyo tusaalaha sare ee **16.43**:

1. Laynka 18-aad waxaynu marka hore combobox-ka waydiinaynaa fadhiga la doortay, kadibna qiimaha uu inoosoo celiyo ayaynu u gudbinaynaa ficilka pickStudent oo aad ka arkayso dhamaadka
2. Ficilka pickStudent haddii aan calooshiisa wax ka sheego. Marka hore haddii fadhiga loo soo gudbiyey yahay **null**, ficilkaas wuxu soo celinayaa qiimaha gaar ah ee **null**. Haddii kale, doorsoomaha toShow ee laynka 37-aad ayuu marka hore ku uruurinayaa ardayda la tusaayo isticmaalaha, si uu uga doorto mid, kadibna waxaynu adeegsanayanaa ficilka showInputDialog ee nooca JOptionPane innaga oo u gudbinayna qiimeyaasha uu u baahanyahay sida liiska ardayda, magaca dialog-ga, iwm. Ficilkaasi wuxu inoosoo celinayaa ardayga la doortay
3. Intaas kadib laynka 22-aad ayaynu Controller-ka ficilkeeda enrollStudent isticmaalaynaa si aynu ardayga uga diiwaangelino fadhigaas. Wicista kadib, waxaynu cusboonaysiinaynaa miiska ardayda.

Ficilka enrollStudent ee Controller-ku wuxu u eegyahay sida tusaalaha **16.44** ee hoose.

```

1 package nimi.controller;
2
3 ...
4
5 public class NimiController implements NimiUIListener {
6     ...
7
8     @Override
9     public void enrollStudent(CourseSession session,
10         Student student) {
11         session.getAllStudents().add(student);
12     }
13     ...
14 }

```

Tusaale 16.44: Badhanka cusub ama kudar ee caleenta ardayda. Arday kudar ama ka diiwaangeli fadhiga la doortay 2-aad

Ficilka enrollStudent kaliya wuxu liiska ardayga fadhiga ku darayaa ardayga la siiyey, sidaad ka arkayso laynka 11-aad.

Intaas kadib, haddii aad kaxayso borogaraamka, kadibna aad tijaabiso badhankaasi qaabka uu u shaqaynaayo, waxad arkaysaa inuu ku xidhanyahay kolba waxa kor kaaga dooran. Sawirka **16.14** ayaad ka arkaysaa dialog-ga kuusoo boodaaya markaad gujiso badhanka Kudar.

Taasi waxay dhacaysaa kaliya markay kuu doorantahay "Kujirta Fadhiga".

16.12.2 Tuurista ardayga

Badhanka kale ee "Tuur" ama "Ka Tuur". Waxaynu cashar ahaan ugu dhaafaynaa isticmaalaha. Qaabka uu u shaqaynaayo waa sidii kii hore oo kale. Haddii ardayda oo dhan la doortay waxad nidaamka ka tuuraysaa ardayga (ama ardayda) uu isticmaaluhu doortay, haddiise la dooratay fadhiga waxa la rabaa inaad ardayga (ama ardayda) ka tuurto fadhiga, laakiinse aanad ka tuurin nidaamka. Waxaynu cashar ahaan ugu dhaafaynaa isticmaalaha. Caawimo ahaan haddii aan inyar halkan ku sharaxayo, waxad adeegsanaysaa labada ficil ee removeStudent ee Controller-ka, kadibna waxa loo baahanyahay inaad muuqaalka cusboonaysiiso.

16.12.3 Ardayda fadhiga

Intaynaan caleenta ardayda ka gudbin, waxa inoo dhiman hal arin. Caleenta ugu horaysa ee fadhiyada waxa ku yaalla badhanka "Ardayda", kaas oo ujeedadiisu tahay inuu ku tuso ardayda fadhigaas ku jirta. Badhankaasi illaa hadda ma shaqaynaayo. Waxa loo baahanyahay in badhankaas lagaga shaqaysiiyo. Marka hore caleenta ardayda waxad ku dartaa ficil lasoo wacaayo, marka badhankaas la gujiyo. Ficilkaasi wuxu noqonkaa tusaale ahaan sida tusaalaha **16.45**.

```

1 package nimi.ui;
2
3 ...
4
5 public class StudentsTab extends NimiTab {
6     ...
7
8 }

```

NIMI - Nidaamka Macluumaadka Iskuulka

Gal Warbixin

Fadhiyada Koorsooyinka **Ardayda**

☐ Dhammaan Ardayda

☒ Kujirta Fadhiga **Maths Introductory Course (MAT-01), 01.01.2011**

ID	Magac	Telefoon
1	Cabdilaahi Maxamed Muuse	

Dooro arday

Dooro ardayga aad rabto inaad ku darto fadhiga.

Khadar Cabdilaahi Muuse (2)

Khadar Cabdilaahi Muuse (2)

<balad magaca> (0)

<balad magaca> (0)

<balad magaca> (0)

<balad magaca> (0)

<balad magaca> (0)

<balad magaca> (0)

<balad magaca> (0)

Tuur Kudar

Sawirka 16.14: Kudar arday fadhiga

```

10  public void selectSession(int selectedSession) {
11      if (selectedSession != -1) {
12          updateSessions();
13          sessionStudents.setSelected(true);
14          sessionsCombo.setEnabled(true);
15          sessionsCombo.setSelectedIndex(selectedSession);
16      }
17  }
18  }

```

Tusaale 16.45: Ku kordhi labadan ficil nooca StudentsTableModel

Waxa kale oo u baahan in labada nooc ee SessionsTab iyo NimiTabbedPane in waxyar lagu daro ama laga badalo. Waxaynu ku bilaabaynaa nooca NimiTabbedPane oo aad ka arkayso tusaalaha 16.46 waxa lagu daraayo ama laga badalaayo.

```

package nimi.ui;

...

4  public class NimiTabbedPane extends JTabbedPane {
6      private NimiModel nimiModel;
7      private StudentsTab studentsTab;
8
9      ...
10
11     private void addTabs() {
12         addTab("Fadhiyada", new SessionsTab(nimiModel, this));
13         addTab("Koorsooyinka", new CoursesTab(nimiModel));
14         this.studentsTab = new StudentsTab(nimiModel);
15         addTab("Ardayda", this.studentsTab);
16     }
17
18     public void showStudentsForSession(int selectedSession) {
19         studentsTab.selectSession(selectedSession);
20         setSelectedIndex(2);
21     }
22 }

```

Tusaale 16.46: Ku kordhi labadan ficil nooca StudentsTableModel

Ugu dambayntii nooca SessionsTab waxad ku dartaa ficilka selectSession ee aad ka arkayso tusaale 16.47.

```

package nimi.ui;

...

4  public class StudentsTab extends NimiTab {
6
7      ...
8
9
10     public void selectSession(int selectedSession) {
11         if (selectedSession != -1) {
12             updateSessions();
13             sessionStudents.setSelected(true);
14             sessionsCombo.setEnabled(true);
15             sessionsCombo.setSelectedIndex(selectedSession);
16         }
17     }
18 }

```

Tusaale 16.47: Ku kordhi labadan ficil nooca StudentsTableModel

Intaas kadib badhanka "Ardayda" ee caleenta fadhiyada waa inuu shaqeeyo.

16.13 Menu - Gal

16.13.1 Kaydi

Maadaama aynu sawirguud ka haysano sida macluumaad loogu kaydinkaro fayl ama looga soo akhriy karo, waxaynu bilaabaynaa inaynu borogaraamkeenii u adeegsano. Kaydi oo aad ka helayso menu-ga sare ee Gal, waxay isticmaalaha u saamaxaysaa inuu model-ka borogaraamka ku kaydiyo fayl, si haddii tusaale ahaan kumbuyuutarku ka damo aanu macluumaadkii uga lumin. Laba siyaabood ayaa inoo

banaan si aynu model-ka fayl ugu kaydino. Sida koobaad waa inaynu model-ka u badalo qoraal sida XML, HTML ama qoraal text ah kadibna ku kaydino kumbuyuutarka. Taasi waxay innagu kalifaysaa inaynu model-ka qoraal u badalo marka la kaydinaayo, marka dambe ee la soo furana aynu qoraalkii model (walxo) u badalo, si aynu isticmaalaha ugu tusno caleemaha kala duwan ee borogaraamka. Sida ama doorashada labaad ee inoo banaan waa inaynu model-ka oo sideeda ah fayl ku kaydino, marka la soo furana sideeda u soo akhrino. Sida hore waxay ku fiicantahay hal arin; faylka qoraalka ama XML-ka ah isticmaaluhu wuu fahmayaa marka uu furo, waxaanu ku furikaraa borogaraam kale, laakiin doorashada labaad faylka borogaraamkeena mooyee mid kale ma furikaro. Haddii la furana lama fahmaayo. Laakiin doorashada labaad xagga qorista ama implement-garaynta ayay kaga fududyahay, doorashada hore. Si kastaba waxaynu halkan kusoo qaadanaynaa doorashada labaad.

Luuqadda Java haddii aad rabto inaad walax ku kaydiso fayl (ama aad u dirto kumbuyuutar kale), waxa laga rabaa hal shuruud walaxdaas nooceeda. Waa in walaxda nooceedu ama noocyada laga soo dhiraandhiriyeey midkood uu implement-gareeyo interface-ka

`java.io.Serializable`. Qasab ma aha in caloosha nooca wax laga badalo. Sidaas awgeed caloosha noocyada waxba ka badalimayno. Haddaba, intaynaan soo qaadan sida model-ka oo dhan fayl loogukaydinkaro, waxaynu soo qaadanaynaa borogaraam yar oo inoo iftiiminaaya arintan. Waxad marka hore nooca Person ku dartaa interface-ka `Serializable`, sidaad ka arkayso tusaalaha [16.48](#).

```

1 package nimi.person;
2
3 import java.io.Serializable;
4
5 public class Person implements Named, Serializable {
6     ...
7 }

```

Tusaale 16.48: Soo **import**-garee interface-ka `Serializable`, kadibna liiska interface-yada nooca Person implement-garaynaayo xaga dabme kaga dar

Intaas kadib, waxaynu soo qaadanaynaa borogaraam yar oo ku tusaaya sida aad walax Person ah fayl ugu kaydinkarto. Person waxaynu u soo qaadanay tusaale ahaan, laakiin walaxdu waxay noqon kartaa noocay rabto, kaliya waa in noocaasi implement-gareeyo interface-ka `Serializable`. Borogaraamkaas waxad ka arkaysaa [16.49](#).

```

1 package nimi;
2
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8
9 import nimi.person.Person;
10
11 public class IsbarJava {
12     public static void main (String args[]) {
13         Person cikal = new Person(123, "Cikal Shiidaad");
14         ObjectOutputStream oos = null;
15         // Kuqor Cikal fayl
16         try {
17             oos = new ObjectOutputStream(new FileOutputStream("cikal.nimi"));
18             oos.writeObject(cikal);
19         } catch (IOException e) {
20             System.out.println("Waxa timid cilad!");
21         } finally {
22             try {
23                 oos.close();
24             } catch (IOException e) {}
25         }
26
27         // Kasoo akhri Cikal fayl
28         ObjectInputStream ois = null;
29         try {
30             ois = new ObjectInputStream(new FileInputStream("cikal.nimi"));
31             Person qof = (Person)ois.readObject();
32             System.out.println("ID: " + qof.getId());
33             System.out.println("Name: " + qof.getName());
34         } catch (Exception ioe) {
35             System.out.println("Waxa timid cilad!");
36         } finally {
37             try {
38                 ois.close();
39             } catch (IOException e) {}
40         }
41     }
42 }

```

```

40     }
41   }
42 }

```

Tusaale 16.49: Borogaraam ku tusaaya sida aad walax fayl ugu kaydinkarto, kadibna aad ugasoo akhriyi karto

Borogaraamka sare ee 16.49 haddii aad kaxayso, waa inay kuusoo baxdo:

ID: 123

Name: Cigal Shiidaad

Aan ka bixiyo sharaxaad kooban borogaraamka 16.49:

1. Noocyada aynu isticmaalayno waxa ugu muhiimsan labada nooc ee `java.io.ObjectOutputStream` iyo `java.io.ObjectInputStream` oo aad ka arkayso laymanka 6-7. Nooca hore waxa loo adeegsan karaa in walaxda lagu qoro stream. Stream-kaasi wuxu noqonkaraa sida halkan `java.io.FileOutputStream`, oo aad adeegsanayso markaad rabto inaad ku qorto fayl ama nooc kale. Nooca dambe waxa loo adeegsanayaa in walax laga soo akhriyo stream. Markaad fayl kasoo akhriyayso waa in stream-ku noqdo `java.io.FileInputStream`
2. Labada ficil ee halkan ugu muhiimsan waa `writeObject` oo kamid ah nooca `ObjectOutputStream`, kaas oo aynu ku isticmaalay laynka 18-aad si aynu walaxda qofka ah ugu kaydino fayl. Markaynu kasoo akhriyayno faylka waxaynu adeegsanaynaa ficilka `readObject`, oo ka tirsan nooca `ObjectInputStream`. Ficilkaasi wuxu soo celinayaa reference ama hogaan `Object` ah, sidaas awgeed waa inaad casting ku samayso oo u badasho noocii saxda ahaa, si aad noocaas ficiladiisa u adeegsato
3. Labada nooc ee `ObjectOutputStream` iyo `ObjectInputStream` waa stream-yo, sidaas awgeed waa inaad xasuusato inaad xidho markaad ka dhamaysato
4. Ka eeg project-ka **nimi** hoostiisa fayl magaciisa la yidhaahdo `cigaal.nimi` (guji F5 si aad u refresh-garayso), si aad u aragto in runtii fayl cusubi dhashay. Faylka magaciisa waxad ka arkaysaa laynka 17-aad.

Borogaraamka sare ee 16.49, waxaynu fayl ku kaydinaa hal walax oo kaliya, laakiin laga bilaabo halkan waxaynu rabnaa inaynu model-ka borogaraamkeena oo caloosha ku haysata ardaydii, koorsooyinkii iyo fadhiyadii ku kaydino fayl, marka loo baahdana kasoo akhrino.

Waxaynu rabnaa marka isticmaaluhu gujiyo menuitem-ka **Gal->Kaydi** inaynu isticmaalaha waydiino faylka magaciisa, kadibna model-ka ku kaydino faylkaas. Laakiin marka hore, ku kordhi noocyada model-ka ... **implements** `Serializable` (Person mooyee inta kale, maadaama isaga aynu kor ku kordhinay), sida ka muuqata tusaalaha 16.50 ee hoose. Xasuuso inaad soo **import**-garayso `java.io.Serializable`. Noocyada `Student` iyo `Teacher` looma baahna inaad ku kordhiso, maadaama toos looga soo dhiraandhriyey nooca `Person`.

```

1 public class Course implements Serializable {
2     ...
3 }
4
5 public class CourseSession implements Serializable {
6     ...
7 }
8
9 public class NimiModel implements Serializable {
10    ...
11 }

```

Tusaale 16.50: Ku kordhi noocyada model-ka `Serializable`

Intaas kadib, waxaynu qoraynaa wixii dhici lahaa marka menuitem-ka **Gal->Kaydi** la gujiyo.

```

1 package nimi.ui;
2
3 ...
4 import java.io.IOException;
5 import java.io.ObjectOutputStream;
6 import java.io.FileOutputStream;
7 import javax.swing.JOptionPane;
8 import javax.swing.filechooser.FileFilter;
9
10 public class Nimi extends JFrame {
11

```

```

13 private void createMenu() {
14     ...
15     JMenuItem save = addMenuItem(fileMenu, "Kaydi");
16     addSaveListener(save);
17     ...
18 }
19 ...
20
21 private void addSaveListener(JMenuItem save) {
22     save.addActionListener(new ActionListener() {
23
24         public void actionPerformed(ActionEvent e) {
25             JFileChooser chooser = new JFileChooser();
26             chooser.setFileFilter(new NimiFileFilter());
27             int returnVal = chooser.showSaveDialog(Nimi.this);
28             if (returnVal == JFileChooser.APPROVE_OPTION) {
29                 ObjectOutputStream output = null;
30                 try {
31                     File saveToFile = chooser.getSelectedFile();
32                     output = new ObjectOutputStream(new FileOutputStream(saveToFile));
33                     output.writeObject(model);
34                 } catch (Exception ex) {
35                     ex.printStackTrace();
36                     JOptionPane.showMessageDialog(Nimi.this, ex.getMessage(), "Waxa kaydinta ku
37                         timid cilad",
38                         JOptionPane.ERROR_MESSAGE);
39                 } finally {
40                     try {
41                         output.close();
42                     } catch (IOException ioe) {}
43                 }
44             }
45         }
46     });
47 }
48
49 private class NimiFileFilter extends FileFilter {
50     @Override
51     public boolean accept(File f) {
52         if (f.getName().endsWith(".nimi")) {
53             return true;
54         }
55         return false;
56     }
57
58     @Override
59     public String getDescription() {
60         return "Nimi Files";
61     }
62 }

```

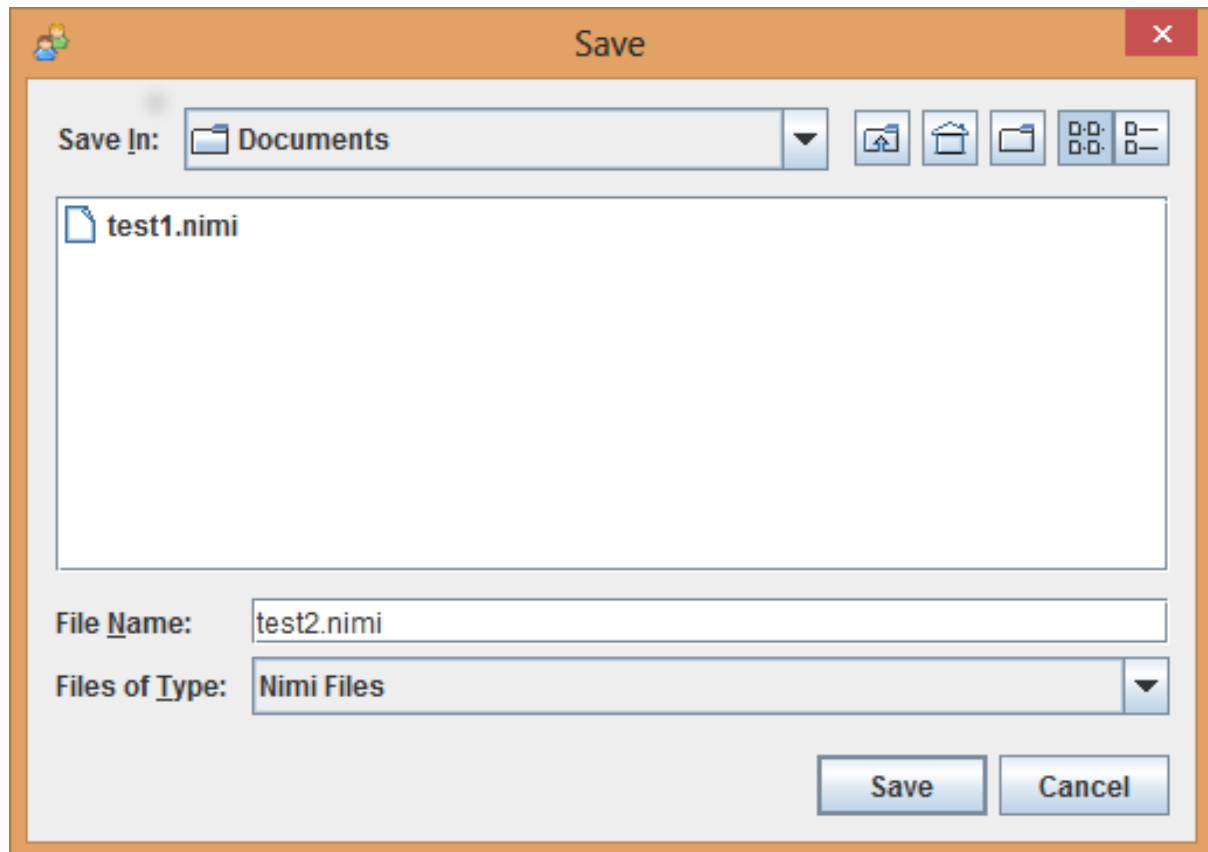
Tusaale 16.51: Ku kaydi model-ka borogaraamka fayl

Intaas kadib, haddii aad kaxayso borogaraamka, kadibna aad gujiso Gal->Kaydi, waa inay soo boodo daaqadda ama dialog-ga sawirka 16.15.

Aynu sharaxaad ka bixino tusaalaha 16.51:

1. Ficilka ugu muhiimsan waa addSaveListener, kaas oo dhagayste ku dhajinaaya menuitem-ka Kaydi, taas oo ah waxa dhacaaya marka la guujiyo
2. Waxaynu isticmaalaynaa nooca JFileChooser, si aynu isticmaalaha u waydiino faylka uu rabo inuu ku kaydiyo shaqada uu borogaraamka ku qabtay. Laynka 25 waxaynu samaynaynaa walax noocaas ah
3. Laynka 26 waxaynu walaxda JFileChooser siinaynaa walax ah NimiFileFilter, taas oo keenaysa in isticmaaluhu arko kaliya faylasha magacoodu ku dhamaado .nimi. Nooca shaandhada ah ee NimiFileFilter waxad ka arkaysaa laga bilaabo laynka 48
4. Marka uu isticmaaluhu fayl doorto, waxaynu faylkaas ku kaydinaaynaa model-kii oo dhan, waxaanad ka arkaysaa laynka 33-aad. Faylkaas magaciisa waxa fiican inuu ku dhamaado .nimi.

Sida caadiga ah marka ugu horaysa oo kaliya ayaa isticmaalaha la waydiiyaa faylka iyo meesha uu ku kaydinaayo macluumaad. Kadib, mar kasta oo aad gujiso *Kaydi* oo ka dhigan *Save*, waxay ahayd in isla



Sawirka 16.15: Daaqadda soo boodaysa markaad gujiso Kaydi

faylkii lagu kaydiyo, laakiin halkan mar kasta oo la gujiyo, waxaynu isticmaalaha waydiinaynaa meesha iyo faylka uu rabo inuu ku kaydiyo.

16.13.2 U Kaydi...

U kaydi sideeda waxay la midtahay ama ka dhigantahay "Save As.", taas oo ujeedadeedu tahay in qoraalka ama wax kasta oo aad borogaraamka lagu sameeyey ku kaydiso fayl cusub. Sidaas awgeed marka hore waa in isticmaalaha la waydiiyo magaca faylka iyo meesha uu rabo inuu ku kaydiyo. Kadibna lagu kaydiyo faylkaas. Waxaynu halkan uga boodaynaa cashar ahaan.

16.13.3 Fur...

"Fur..." oo ka dhigan "Open..." waxa loogu talagalay inaad furto macluumaad aad hore u kaydisay. Model-kii aynu kaydinay ayaynu rabnaa inaynu mar kale furno, si aynu u eegno, haddii loo baahdana wax uga badalo. Waxaynu kaga shaqaysiinaynaa menuitem-ka Fur..., taas oo ka dhigan inaynu qorno wixii dhici lahaa marka isticmaaluhu gujiyo Gal->Fur..., waxaanad ka arkaysaa tusaalaha 16.52 ee hoose.

```

1 package nimi.ui;
2
3 ...
4 import java.io.ObjectInputStream;
5 import java.io.FileInputStream;
6
7 public class Nimi extends JFrame {
8
9     private void createMenu() {
10         ...
11         JMenuItem open = addMenuItem(fileMenu, "Fur...");
12         addOpenListener(open);
13         ...
14     }
15
16     ...

```

```

18 private void addOpenListener(JMenuItem open) {
19     open.addActionListener(new ActionListener() {
20
21         public void actionPerformed(ActionEvent e) {
22             JFileChooser chooser = new JFileChooser();
23             chooser.setFileFilter(new NimiFileFilter());
24             int returnVal = chooser.showOpenDialog(Nimi.this);
25             if (returnVal == JFileChooser.APPROVE_OPTION) {
26                 ObjectInputStream input = null;
27                 try {
28                     File saveToFile = chooser.getSelectedFile();
29                     input = new ObjectInputStream(new FileInputStream(saveToFile));
30                     NimiModel newModel = (NimiModel) input.readObject();
31                     model.getSessions().clear(); // masax fadhiyada
32                     model.getCourses().clear(); // masax koorsooyinka
33                     model.getStudents().clear(); // masax ardayda
34                     // macluumaadka faylka laga soo akhriyey ku badal
35                     model.getSessions().addAll(newModel.getSessions());
36                     model.getCourses().addAll(newModel.getCourses());
37                     model.getStudents().addAll(newModel.getStudents());
38                     controller.updateAllView();
39                 } catch (Exception ex) {
40                     ex.printStackTrace();
41                     JOptionPane.showMessageDialog(Nimi.this, ex.getMessage(), "Error",
42                                             JOptionPane.ERROR_MESSAGE);
43                 } finally {
44                     try {
45                         input.close();
46                     } catch (IOException ioe) {}
47                 }
48             }
49         }
50     });
51 }

```

Tusaale 16.52: Ku kaydi model-ka borogaraamka fayl

Intaas kadib, si aad u tijaabiso borogaraamka, marka hore kaxee, kadibna fadhiyo, koorsooyin iyo arday ku dar. Intaas kadib, kaydi model-ka adiga oo adeegsanaaya **Gal->Kaydi**. Kadib, borogaraamka xidh. Mar kale borogaraamka shid, kadibna guji **Gal->Fur...**, raadi faylkii aad hore u kaydisay, kadinba fur faylkaas. Intaas kadib, waa in macluumaadkii aad hore u kaydisay kuusoo baxaan. Haddii aynu ku xigsiino sharaxaad kooban oo ku saabsan tusaalaha **16.52**:

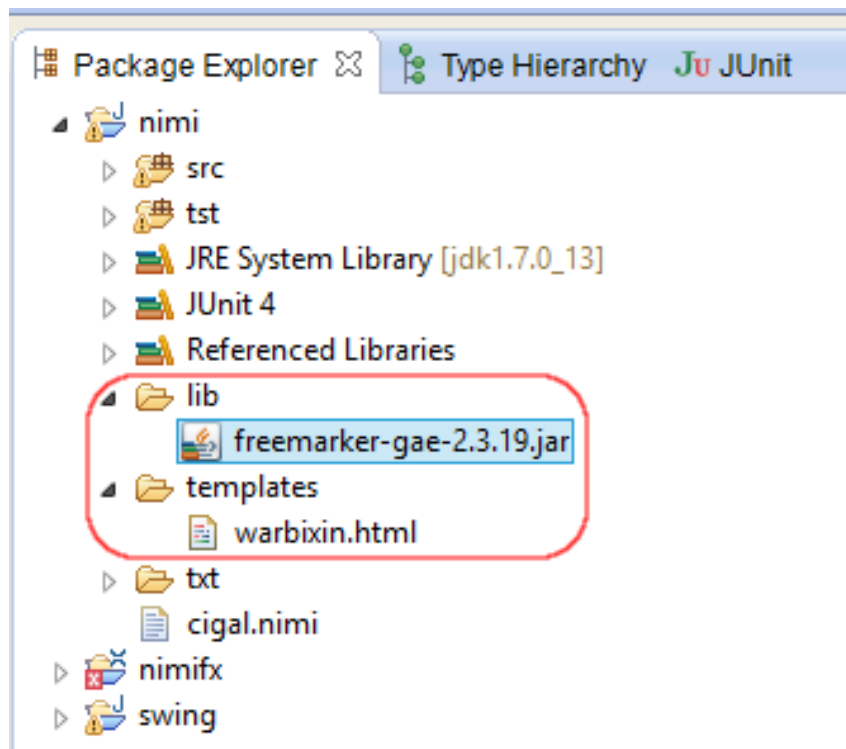
1. Waxa soo kordhay waxa ugu muhiimsan ficilka `addOpenListener`, kaas oo aad dhamaadka kaga darayso nooca `Nimi`. Ficilkaasi wuxu caloosha ku haystaa wixii dhici lahaa marka la gujiyo **Gal->Fur...**
2. Waxaynu adeegsanaynaa mar kale `JFileChooser` si aynu isticmaalaha u waydiino faylka la furaayo. Gaar ahaan ficilkiisa `showOpenDialog`, oo aad ka arkayso laynka 24
3. Mar labaad waxaynu isticmaalaynaa shaandhada `NimiFileFilter`, taas oo aynu hore u soo qornay, si aynu isticmaalaha u tusno kaliya faylasha magacoodu ku dhamaado `.nimi`
4. Waxaynu adeegsanaynaa ficilka `readObject` ee nooca `ObjectInputStream` mar labaad, waxaanad ka arkaysaa laynka 30. Waxaynu qiimaha uu ficilkaasi soo celiyo u badalaynaa `NimiModel`
5. Intaas kadib, laymanka 31-37, waxaynu daadinaynaa fadhiyadii, koorsooyinkii iyo ardaydii u hore, oo waxaynu ku badalaynaa kuwa faylka laga soo akhriyey. Sida caadiga ah waxay ahayd inaynu waydiino inuu hubo isticmaaluhu iyo in kale
6. Laynka 38, waxaynu `Controller`-ka ku amraynaa inay cusboonaysiiso muuqaalka oo dhan.

16.14 Menu - Warbixin

16.14.1 Warbixin Dhan...

Waxaynu ugu dambayntii qoraynaa wixii dhacilaha marka isticmaaluhu gujiyo **Warbixin->Warbixin Dhan...** Waxaynu rabnaa inaynu qoraal la fahmi karo oo noqonkara XML, HTML, Excel, ama nooc kale kasoo saarno borogaraamka si aynu tusaale ahaan qof kale ugu dirno ama meel ugu kaydino. Waxa kale oo aynu halkan ku baranaynaa sidaad u isticmaalikarto library uu qof kale ama dad kale qoreen, kaas oo aad u isticmaalayso ujeedo. **Library** macnihiisa hore ayaynu

ugasoo hadalay. Waxaynu halkan ku adeegsanaynaa library loo yaqaano *Freemarker*⁴, kaas oo aad u adeegsankaro inaad borogaraamkaaga kasoo saarto warbixino kala duwan. Waxad Freemarker kalasoo dagtaa iyada oo ah *jar* qaybta *Download* ee isla boggaas⁵. Intaas kadib, waxad faylkii *jar*-ka ah ku kaydisaa folder-ka *lib* ee project-ka nimi hoostiisa sidaad ka arkayso sawirka 16.16, laakiin marka hore samee folder-ka *lib*.



Sawirka 16.16: Library-ga Freemarker ku soo dar borogaraamka, sidaad u isticmaasho

Intaas kadib, waxad folder-ka *templates* ee isla sawirkaas ka muuqadata ku kaydisaa fayl cusub oo aad u bixiso *warbixin.html*. Marka hore samee folder-ka *templates*, kadibna guji *File->New->File*. Waxad faylkaas ku magacowdaa tusaale ahaa *warbixin.html*, waxaanad calooshiisa u eekaysiisaa sida tusaalaha 16.53.

```

1 <html>
2 <head>
3 <title>NIMI – Nidaamka Macluumaadka Iskuulka</title>
4 </head>
5 <body>
6 <h1>NIMI – Nidaamka Macluumaadka Iskuulka</h1>
7 <#escape x as x?html>
8 <h3>Fadhiyada</h3>
9 <hr/>
10 <table border="1">
11 <thead>
12 <tr><th>Koorso</th><th>Bilawga</th><th>Dhibco</th></tr>
13 </thead>
14 <tbody>
15 <#list sessions as session>
16 <tr>
17 <td>${session.course.name}</td><td>${session.startDate?datetime}</td>
18 <td>${session.numberOfCredits}</td>
19 </tr>
20 <#if session.getAllStudents()?has_content>
21 <tr>
22 <td colspan="3">
23 <ul>
24 <#list session.getAllStudents() as student>
25 <li><a href="#${student.id}>${student.name}</a></li>
26 </#list>
27 </ul>

```

⁴Freemarker: <http://freemarker.sourceforge.net/>

⁵Ama halkan toos ugalasoo dag: <http://sourceforge.net/projects/freemarker/files/freemarker/2.3.19/freemarker-gae-2.3.19.jar/download>

```

28 </td>
29 </tr>
30 </#if>
31 </#list>
32 </tbody>
33 </table>
34 <h3>Koorsooyinka</h3>
35 <hr/>
36 <table border="1">
37 <thead>
38 <tr><th>Magac</th><th>Number</th><th>Qayb</th></tr>
39 </thead>
40 <tbody>
41 <#list courses as course>
42 <tr><td>${course.name}</td><td>${course.number}</td>
43 <td>${course.department}</td>
44 </tr>
45 </#list>
46 </tbody>
47 </table>
48 <h3>Ardayda</h3>
49 <hr/>
50 <table border="1">
51 <thead>
52 <tr><th>ID</th><th>Magac</th><th>Telefoon</th></tr>
53 </thead>
54 <tbody>
55 <#list students as student>
56 <tr><td><a id="${student.id}">${student.id}</a></td>
57 <td>${student.name}</td><td>${student.phone!''}</td>
58 </tr>
59 </#list>
60 </tbody>
61 </table>
62 </#escape>
63 </body>
64 </html>

```

Tusaale 16.53: Template-ka warbixinta

Si aynu u fahamno labadaas fayl ee hore iyo sababta aynu ugu baahanayahay, waxan soo qaadanayaa sawir ka turjumaaya qaabka ay Freemarker u shaqaynayso, kaas oo aan kasoo xigtay bogga Internet-ka ee Freemarker, waxaanad ka arkaysaa sawirka 16.17. Sida sawirkaas ka muuqata Freemarker, waxa la siinayaa fayl la yidhaahdo *Template file*, kaas oo tusaale ahaan noqonkara HTML ama XML, faylkaas waxad calooshiisa ku arkaysaa waxyaalo u gaar ah Freemarker, kuwaas oo calaamado gaar ah loo isticmaalay sida {...} iyo qoraal caadi ah. Template file-kii aynu halkan ku isticmaalaynay waxad ka arkaysaa tusaalaha 16.53. Waxa kale oo Freemarker la siinayaa walax ama walxo. Intaas kadib, Freemarker waxay soo tufaysaa faylkii template-ka ahaa oo macluumaadkii walxaha lagu buuxiyey. Arintani waxay si fiican kuugu iftiimi markaynu tusaalahan dhamayno.

Waxaynu qoraynaa code-kii ka dambeeyey menuitem-ka Warbixin->Warbixin Dhan..., ee la kaxayn lahaa marka la gujiyo. Waxa borogaraamkeena kusoo kordhay si looga shaqaysiiyo menuitem-kaas, waxad ka arkaysaa tusaalaha 16.54.

```

package nimi.ui;
2
...
4
import java.io.FileWriter;
import java.io.Writer;
import java.util.HashMap;
import java.util.Map;
import freemarker.template.Configuration;
import freemarker.template.Template;
import java.awt.Desktop;
12
public class Nimi extends JFrame {
14
    ...
16
    private void createMenu() {
18
        ...
        JMenuItem fullReport = addMenuItem(reportMenu,
20
            "Warbixin dhan...");

```

```

    addReportListener(fullReport);
22  ...
    }
24  ...

26  private void addReportListener(JMenuItem menuItem) {
28      menuItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
30          File directory = null;
          JFileChooser chooser = new JFileChooser();
32          chooser.setCurrentDirectory(new java.io.File("."));
          chooser.setDialogTitle("Dooro meesha warbixinta lagu kaydinaayo");
34          chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
          chooser.setAcceptAllFileFilterUsed(false);

36          if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
38              directory = chooser.getSelectedFile();
          } else {
40              System.out.println("No Selection ");
              return;
42          }
          FileWriter fw = null;
44          try {
              Configuration cfg = new Configuration();
46              Template template = cfg.getTemplate("templates/warbixin.html");
              Map<String, Object> data = new HashMap<String, Object>();
48              data.put("sessions", model.getSessions());
              data.put("courses", model.getCourses());
50              data.put("students", model.getStudents());
              // File output
52              File ouput_file = new File(directory, "nimi_warbixin.html");
              fw = new FileWriter(ouput_file);
54              template.process(data, fw);

56              // Faylka cusub fur!
              Desktop.getDesktop().browse(ouput_file.toURI());
58          } catch (Exception e2) {
              e2.printStackTrace();
60          } finally {
              try {
62                  fw.close();
              } catch (IOException e1) {}
64          }
        }
66    });
68 }

```

Tusaale 16.54: Warbixin dhamaystiran

Intaynaan wax sharaxaad ah ka bixin haddii aad kaxayso borogaraam sare ee 16.54. Marka hore borogaraamku wuxu ku waydiinayaa meesha aad rabto inaad warbixinta ku kaydiso. Dooro meel kasta meesha aad rabto kumbuyuutarkaaga. Intaas kadib, borogaraamku wuxu samaynayaa fayl ay ku jiraan dhammaan fadhiyadii, koorsooyinkii iyo ardaydaa nidaamka, kadibna wuxu kuugu furayaa borogaraamka Browser-ka. Aniga waxa iisoo baxday sawirka 16.18.

Waxaynu ku xigsiinaynaa sharaxaad kooban oo ku saabsan tusaalaha 16.54:

1. Ficilka `addReportListener` wuxu `menuItem`-ka "Warbixin Dhan..." ku dhajinayaa wixii dhici lahaa marka la gujiyo
2. Waxa ugu muhiimsan laynka 45 halkaas oo aynu ku adeegsanayno nooca `Configuration` oo ka tirsan `Freemarker`. Waxaynu samaynaynaa walax noocaas ah, taas oo aynu siinayno `template`-kii iyo walxihii `model`keena ee aynu rabnay in warbixinta laga diyaariyo, kuwaas oo aad ka arkayso laymamka 48-50
3. Walaxda `cfg` ee aynu samaynay laynka 45-aad, waxaynu isticmaalaynaa ficilkiisa `getTemplate`, kaas oo aynu siinayno waddada laga helaayo faylka `template`-ka ah, sidaad ka arkayso laynka 46-aad
4. Ugu dambayntii waxaynu laynka 54 adeegsanaynaa ficilka `process` ee nooca `Template` uu ficilka laynka `getTemplate` inoosoo celiyey, kaas oo aynu u gudbinayno walxihii `model`ka iyo faylkii lagu kaydinalahaa natiijada.

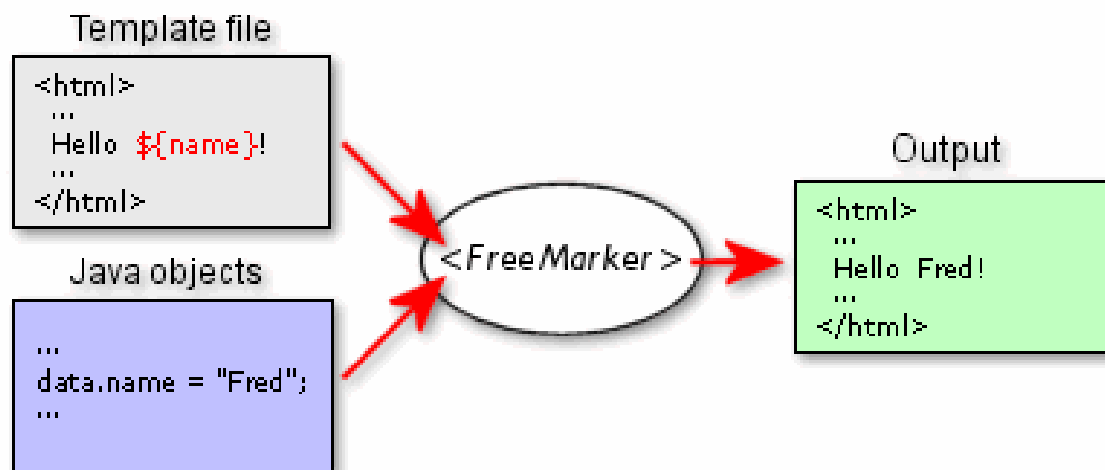
Faa'iidada ugu wayn ee qaabkani ama isticmaalka waxa lagu magacaabay "Template Engines" sida Freemarker ay leeyihiin waa inaad badali karto template-ka adiga oo aan taabanin source code-ka borogaraamka. Tusaale ahaan haddii aynu rabno inaynu warbixinta ka dhigno XML, qoraal caadi ah, ama HTML aynu wax ka badalo, waxaynu badalaynaa kaliya template-ka. Marka dambe ee aynu soo saarno warbixinta waxa borogaraamku isticmaalayaa template-ka cusub, taas oo keenaysa in wixii borogaraamku soo tufay is badasho. Si aynu arintaas tusaale ugasoo qaadano, waxad template-ka 16.53, ku badashaa qoraalka 16.55, kadibna mar kale intaad borogaraamka kaxayso guji Warbixin->Warbixin Dhan...

```

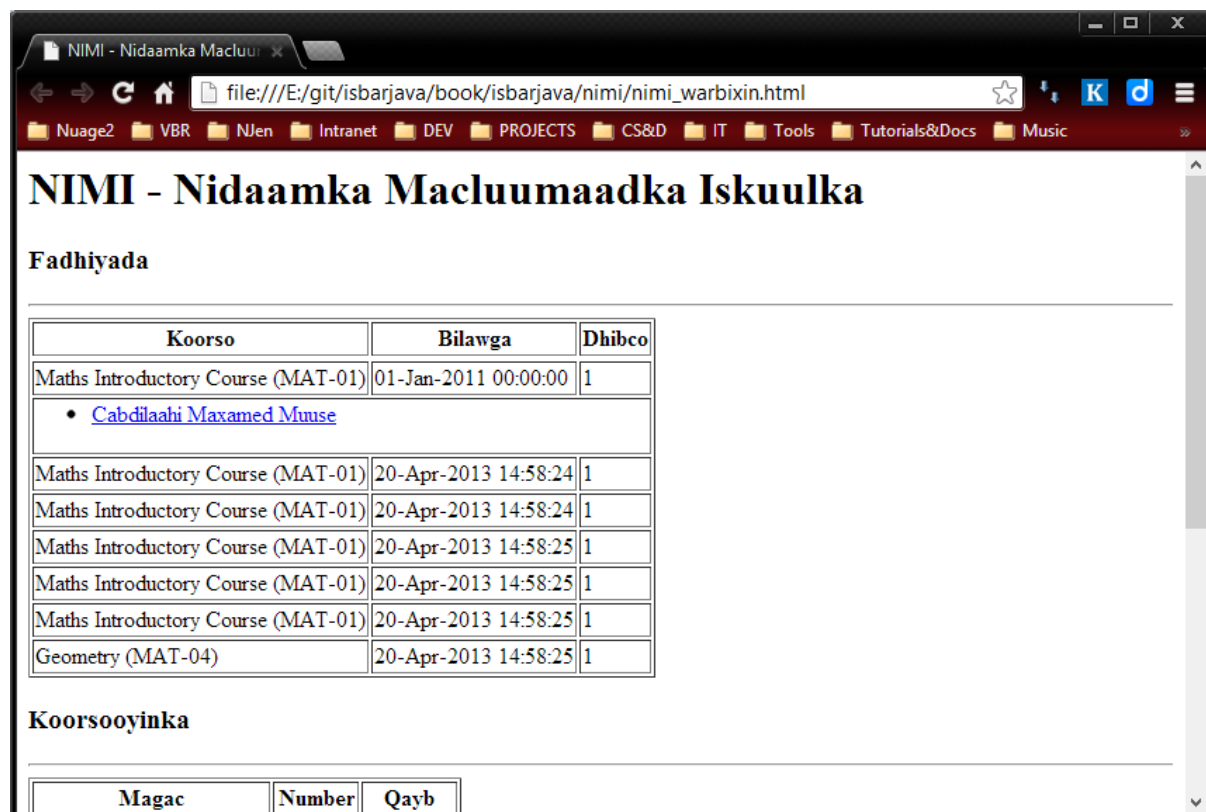
2 <pre>
3 Fadhiyada
4
5
6 <#list sessions as session>${session.course.toString()}, ${session.startDate?datetime},
7   ${session.numberOfCredits}
8 </#list>
9
10 Koorsooyinka
11
12 <#list courses as course>
13   ${course.name}, ${course.number}, ${course.department}
14 </#list>
15
16 Ardayda
17
18 <#list students as student>
19   ${student.id}, ${student.name}, ${student.phone!''}
20 </#list>
21 </pre>

```

Tusaale 16.55: Template-ka warbixinta oo la badalay



Sawirka 16.17: Sawir ka turjumaaya qaabka ay Freemarker u shaqaynayso



Sawirka 16.18: Warbixin dhamaystiran oo borogaraamka nimi soo tufay marka la gujiyo Warbixin->Warbixin Dhan...

Cutub 17

Sheeko gaaban: JavaFX

Wakhtigii aan buuggan qoristiisa bilaabay **JavaFX** waxay ahayd mashruuc bilaw ah, oo muddo gaaban socday. Sidaas awgeed arin laysku huro oo wax weyn laga soo qaado muu ahayn. Haddaba intii aan buuggan qoristiisa ku jiray wax badan ayaa iska badalay mashruucaas, oo wuxu sameeyey hormar la taaban karo. Sidaas awgeed waxan rabaa inaan cutubkan si dulmar ah ugaga hadlo **JavaFX** iyo waxay tahay.

17.1 Waa maxay JavaFX

Intaynaan isku dayin inaynu meel ku tiirino **JavaFX** waxay tahay, waxad markiiba magaca qaybtiisa hore ka arkaysaa inay tahay shay ku tiirsan luuqadda Java. Taasi waxay keenaysaa inaanay **JavaFX** jirikarin Java la'aanteed, sababtoo ah waxa lagu dhisay luuqadda Java, waxaanay adeegsanaysaa **JRE**-da Java. **JavaFX** waxa la odhan karaa waa **Java Applets**-kii, ay Java ku bilaabantay oo la casriyeeyey. Halkii **Java Applets**-ku ku shaqaynayeen kaliya caloosha borogaraamyada **Browsers**-ka, borogaraamyada **JavaFX** waxay dheeryihiin in lagu isticmaalikaro kumbuyuutarada caadiga ah iyo moobilada.

JavaFX waxa loogu talagalay in lagu dhiso waxa lagu magacaabay **Rich Internet Application (RIA)** [3]. **RIA** waxay ka midtahay evolusion-ka ama marxaladaha kala duwan ee ay soo martay teknolojiyadda ama farsamada lagu dhiso User Interface-ku. Sideedaba haddii aynu inyar dib u milicsanno marxaladaha kala duwan ee ay soo martay dhisidda User Interface-ku, waxa la odhan karaa waxay ka bilaabantahay kaadhadhkii daldaloolay (punch cards) oo aad loo adeegsanjiray qarniyadii 18-19-aad. Waxa ka dambeeyey console ama terminal applications. Dad badan ayaa xasuusta wakhtigii DOS-ka. Kadib, waxa yimid **OS**-yadii casriga ahaa oo keenay dhalashadii **Desktop Applications**-ka, oo aynu hore u soo sheegnay. Waxa la odhankaraa illaa hadda waxaynu dhisaynay **Desktop Application**. Waxay **Desktop Applications**-kii marba markii ka dambaysay bilaabeen inay u guuraan ama loo raro dhanka Web-ka, taasina waxay keentay dhalashiidii **Web Applications**-ka. **Web Applications**-kii waxay ugu dambayntii marayaan oo ay isu badalayaan **RIA**.

Sideedaba **RIA** waa borogaraamyadii caadiga ahaa ee **Desktop Applications** oo loo raray Internet-ka, kuna shaqaynaaya caloosha borogaraamyada **Browser**-ka. Faa'iidooyinka ugu muhiimsan ee ay soo kordhiyeen **RIA** waa in borogaraamkii laga isticmaali karo meel kasta oo aad joogto iyo inaan loo baahnayn in lagu kaydiyo kumbuyuutarada isticmaalaha, maadaama ay toos ugu shaqaynayaan caloosha borogaraamka **Browser**-ka. Meesha ugu fiican ee akhristuhu ka bilaabi karo barashada **JavaFX** waa [9].

Teknolojiyadaha la tartamaaya **JavaFX** waxa ka mid ah *Adobe Flex*¹ iyo *Silverlight*². Akhristuhu haddii uu rabo wuxu booqan karaa website-yadaas si uu macluumaad dheeraad ah uga helo ama isu barbardhigo teknolojiyadahaas kale iyo **JavaFX**. Qodobkan waxan u soo qaatay si uu akhristuhu u ogaado inaanay **JavaFX** kaligeed ahayn ee ay jiraan teknolojiyaddo kale oo u dhigma.

Sawirka 17.1 waxad ka arkaysaa dhisme ahaan unugyada ay ka koobantahay JavaFX. Marka ugu horaysa waxad arkaysaa inay ku dulshaqaynayso **JVM**-ka Java. Taas kadib, **JavaFX** waxay adeegsanaysaa **library**-yada iyo agabka **JDK**-ka Java. **JavaFX** waxay kuu saamaxaysaa inaad samayso graphics ama sawirro 2D iyo 3D labada ah. Waxa kale oo ay kuu saamaxaysaa inaad borogaraamyadaada si fudud ugusoo darto maqal iyo muuqaalba.

Waxyaalaha ay ku fiicantahay **JavaFX**, ka sakow kuwa aan kor kusoo sheegay, waxa kamid ah in borogaraamyada lagu qoray Swing lasoo dhexgalinkaro waxyaalo laga soo qaatay **JavaFX**. Taas oo keenaysaa in borogaraamyada hadda jira ee aan lagu qorin **JavaFX** ay ka faa'iidaystaan **JavaFX**.

¹<http://www.adobe.com/devnet/flex.html>

²<http://www.microsoft.com/silverlight/>

17.2 Dhisidda borogaraamyada JavaFX

Waxad mooddaa in IDE-da ugu haboon in lagu dhiso borogaraamyada **JavaFX** ay tahay **NetBeans**. Inkastoo ay fududahay sida **Eclipse** loogudaro **plugins**-ka ka dhiman si loogu adeegsado dhisidda borogaraamyada **JavaFX**. Waxay halkan kusoo qaadanaaynaa **NetBeans**. Borogaraamka **Eclipse** oo toos loogu adeegsankaro dhisidda borogaraamyada **JavaFX** waxad ka helaya halkan <http://efclipse.org/install.html>.

17.2.1 Lasoo dagista NetBeans

NetBeans waxad kalasoo dagikartaa adarayska <https://netbeans.org/downloads/index.html>. Qaabka aad ulasoo dagayso waxad ka arkaysaa sawirka 17.2. Marka faylkaas oo dhamaystiran kuusoo dago laba jeer guji, kadibna dhamaystir installations-ka, si borogaraamku ugu kaydsamo kumbyuutarkaaga. Intaas kadib ka raadi borogaraamka cusub kumbyuutarkaaga, kadib laba jeer guji si aad u istaado.

NetBeans qaabka ay u eegtahay marka ugu horaysa, gaar ahaan **OS**-ka Windows 8, waxad ka arkaysaa sawirka 17.3 ee hoose.

Intaas kadib toos ayaad u adeegsan kartaa borogaraamka **NetBeans** si aad ugu dhisto borogaraamyada **JavaFX** iyo kuwa Java ee guudba. Borogaraamyada **Eclipse** iyo **NetBeans** looguma talagalin kaliya dhisidda borogaraamyada Java, ee waxad u isticmaali kartoo luuqado kala duwan sida: *C/C++*, *PHP*, *HTML5*, iwm. sidaad ka arkayso sawirka 17.2, dhankiisa bidixda. Waxaynu samaynaynaa borogaraamyada yar oo tusaale ah innaga oo adeegsanayna **NetBeans**. Waxad ku bilawdaa talaabooyinkan:

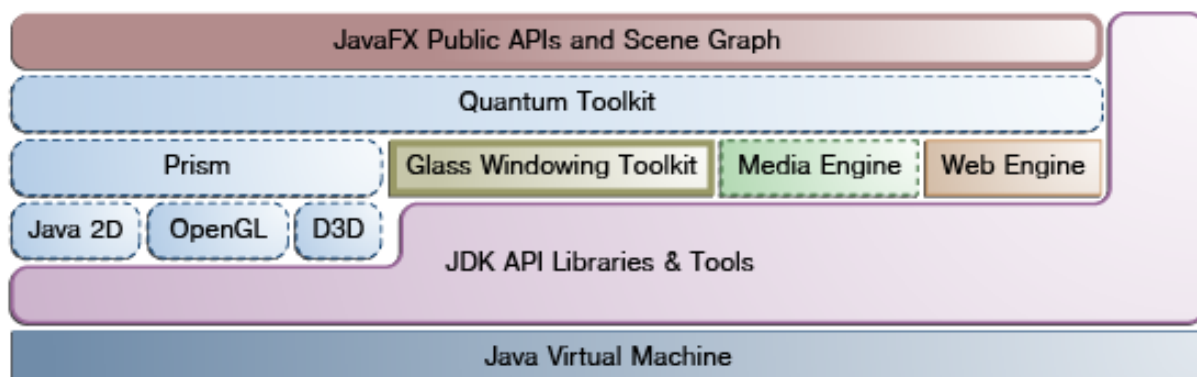
- Waxad gujisaa **File->New Project...**, kadib waxa kuusoo baxaysa daaqadda 17.4. Waxad daaqadaas ka doorataa *JavaFX* (Categories) iyo *JavaFX application* sidaad isla sawirkaas ka arkayso. Sideedaba borogaraamyada **Eclipse** iyo **NetBeans** midna ma adeegsan kartid haddii aanad project cusub ku bilaabin.
- Intaas kadib waxad gujisaa *Next*
- Waxad magaca Project-ka ku bixisaa *JavaFXHelloWorld* sida ka muuqata sawirka 17.5. Xasuuso meesha **NetBeans** uu borogaraamka kaga kaydinaayo kumbyuutarkaaga, oo ah meesha ka hoosaysa magaca. Intaas kadib waxad gujisaa **Finish**

Waxaynu samaynaynaa borogaraamyada yar oo ka kooban hal badhan oo marka isticmaaluhu gujiyo aynu tusayno fariin yar oo kooban oo ku saabsan taariikhda soosaarka Java ee isticmaaluhu adeegsanaayo. Waxad nooca *JavaFXHelloWorld* ee faylka *JavaFXHelloWorld.java* u eekaysiisaa sida borogaraamka 17.2.1 ee hoose.

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package javafxhelloworld;
6
7  import java.text.DateFormat;
8  import java.text.SimpleDateFormat;
9  import java.util.Date;
10 import javafx.application.Application;
11 import javafx.event.ActionEvent;
12 import javafx.event.EventHandler;
13 import javafx.scene.Scene;

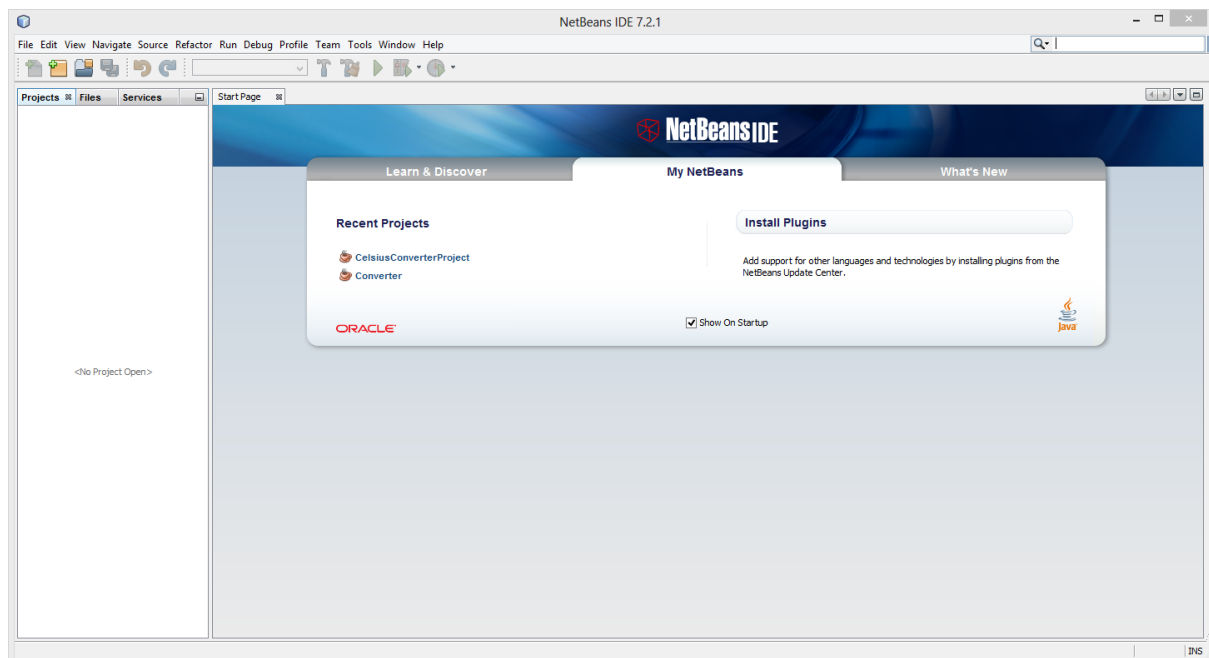
```



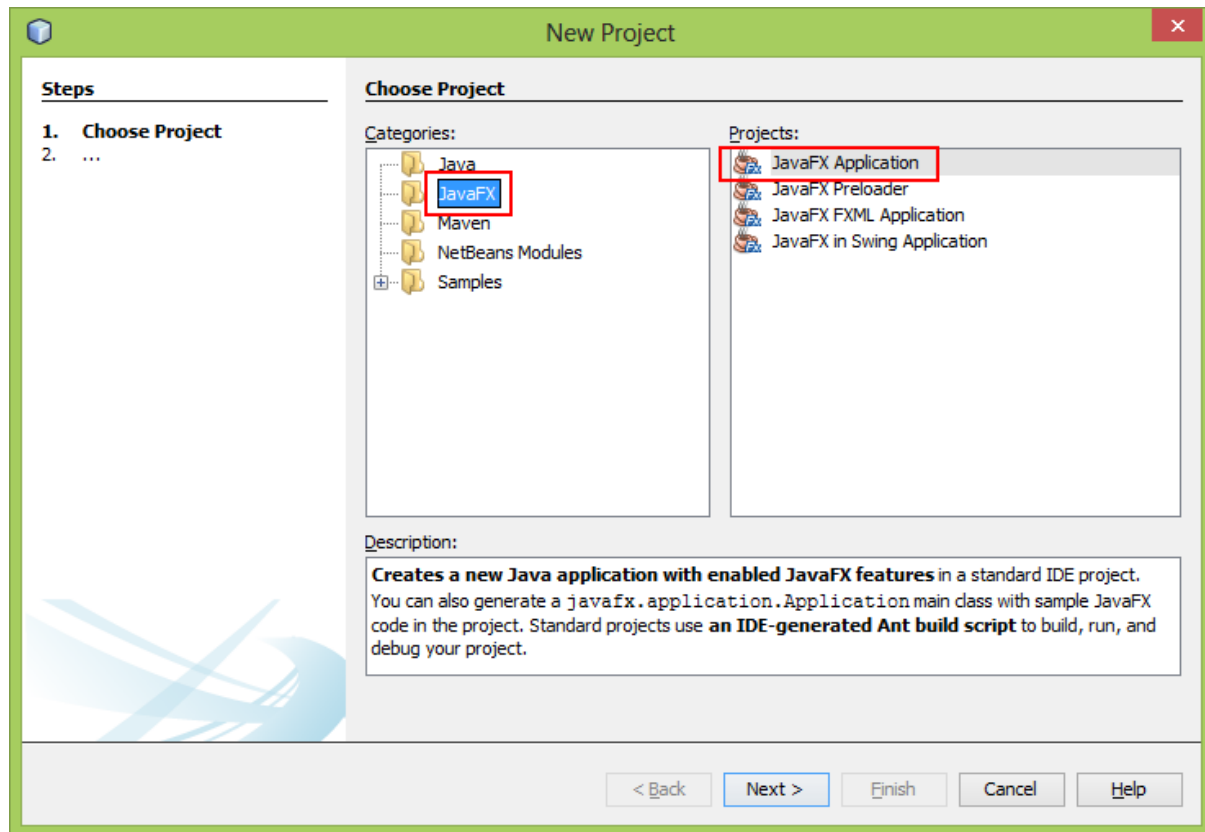
Sawirka 17.1: Qaabka ay u dhisantahay iyo waxyaalaha ay ka koobantahay ama isticmaalayso JavaFX

NetBeans IDE Download Bundles					
Supported technologies *	Java SE	Java EE	C/C++	PHP	All
• NetBeans Platform SDK	•	•			•
• Java SE	•	•			•
• Java FX	•	•			•
• Java EE		•			•
• Java ME		•			•
• HTML5		•		•	•
• Java Card™ 3 Connected					•
• C/C++			•		•
• Groovy				•	•
• PHP				•	•
Bundled servers					
• GlassFish Server Open Source Edition 3.1.2.2		•			•
• Apache Tomcat 7.0.34		•			•
	Download	Download	Download	Download	Download
	Free, 80 MB	Free, 178 MB	Free, 53 MB	Free, 53 MB	Free, 198 MB

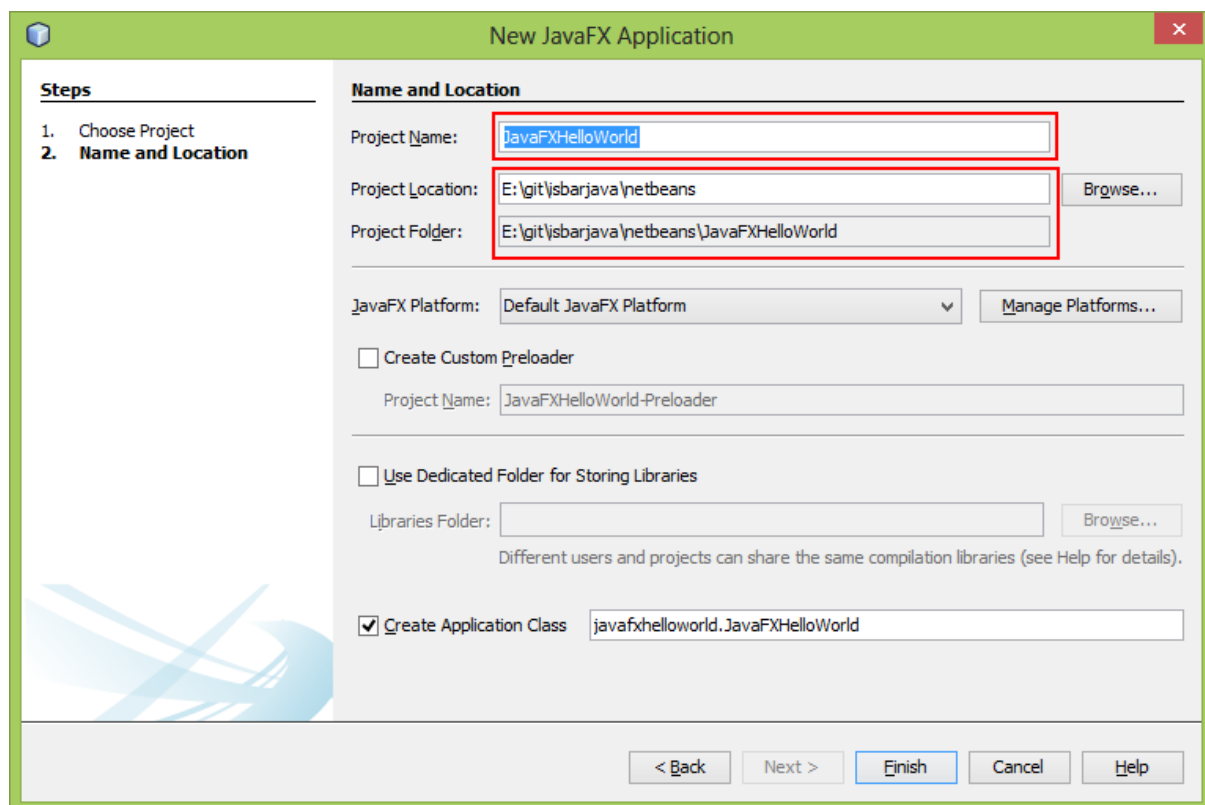
Sawirka 17.2: Lasoo dagista borogaraamka Netbeans



Sawirka 17.3: Borogaraamka Netbeans oo lagu isticmaalaayo Windows 8, gaar ahaan markii ugu horaysay ee la shiday



Sawirka 17.4: Daaqadda samaynta project-ka cusub ee NetBeans



Sawirka 17.5: Macluumaadka project-ka cusub

```

14 import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
16 import javafx.scene.paint.Color;
import javafx.scene.text.Text;
18 import javafx.stage.Stage;

20 /**
 *
22  * @author Hussein
 */
24 public class JavaFXHelloWorld extends Application {
    private int tiradaGujooyinka = 1;
26    @Override
    public void start(Stage primaryStage) {
28        Button btn = new Button();
        final Text fariinTxt = new Text();
30        btn.setText("Tus macluumaadka Java");
        btn.setOnAction(new EventHandler<ActionEvent>() {
32            @Override
            public void handle(ActionEvent event) {
34                fariinTxt.setFill(Color.FIREBRICK);
                DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
36                Date date = new Date();
                fariinTxt.setText(String.format(
38                    "Taariikhda: %s\n"
                    + "Version-ka Java: %s\n"
40                    + "Inta jeer ee aad i waydiisay: %d",
                    dateFormat.format(date),
42                    System.getProperty("java.version"),
                    tiradaGujooyinka));
44                tiradaGujooyinka++;
            }
46        });

48        VBox root = new VBox();
        root.getChildren().addAll(btn, fariinTxt);
50
        Scene scene = new Scene(root, 200, 100);
52
        primaryStage.setTitle("Macluumaadka nidaamka!");
        primaryStage.setScene(scene);
54        primaryStage.show();
56    }

58    /**
 * The main() method is ignored in correctly deployed JavaFX application.
60    * main() serves only as fallback in case the application can not be
 * launched through deployment artifacts, e.g., in IDEs with limited FX
62    * support. NetBeans ignores main().
 *
64    * @param args the command line arguments
 */
66    public static void main(String[] args) {
        launch(args);
68    }
}

```

Haddii aan sharaxaad kooban ka bixiyo **17.2.1**. Marka badhanka la gujiyo ayaa waxaynu badhanka hoostiisa kusoo bandhigaynaa fariin yar oo u eeg sidan:

```

Taariikhda: xx
Version-ka Java: yy
Inta jeer ee aad i waydiisay: zz

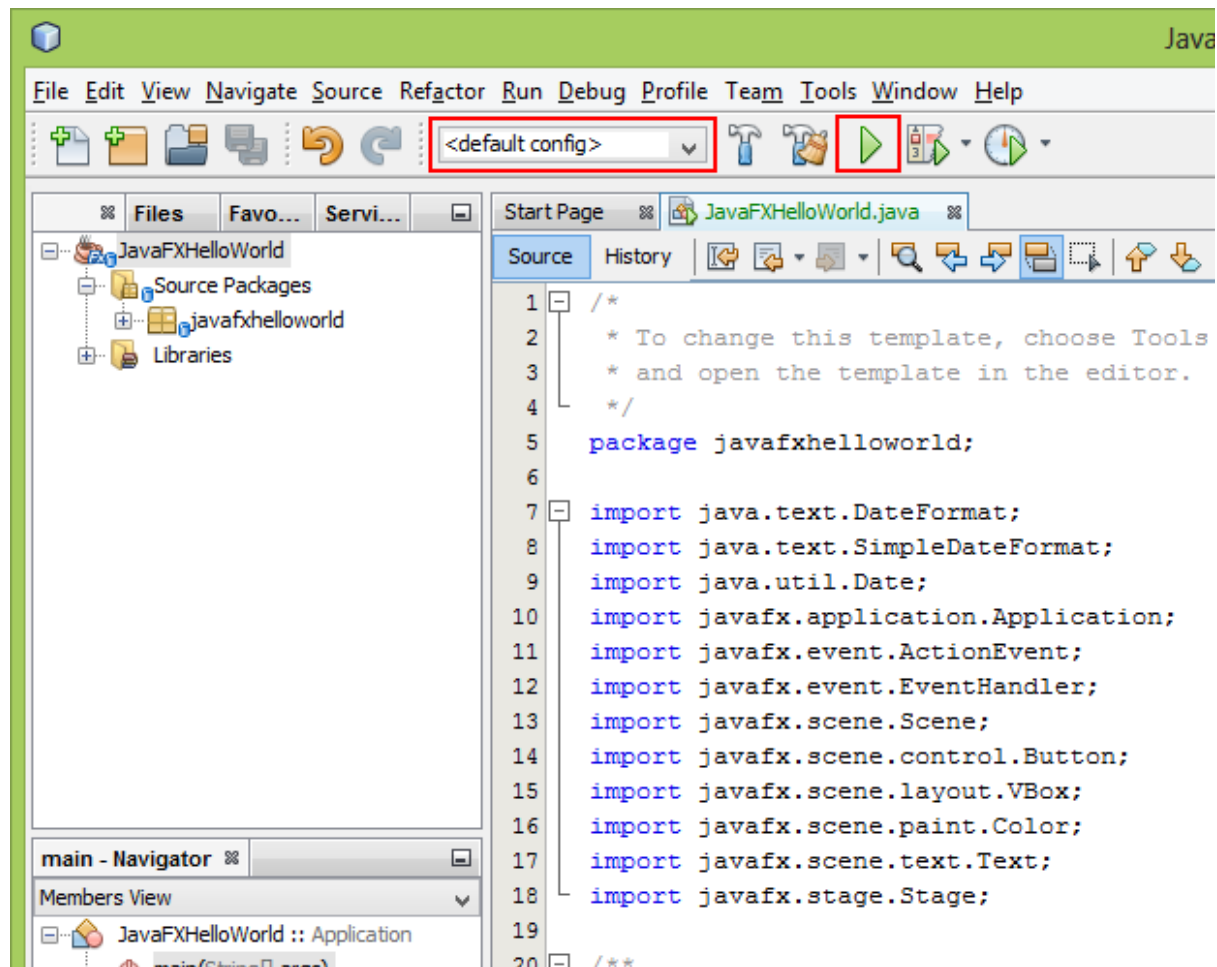
```

Sidaan soo sheegayba borogaraamyada lagu sameeyey **JavaFX** waxa loo isticmaali karaa ugu yaraan saddex siyaabood oo kala ah:

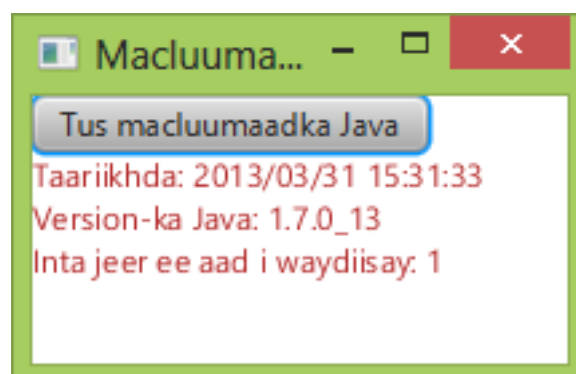
1. Sida caadiga ah ee borogaraamyada kale ee Java loo isticmaali karo oo ah in lagu kaxeeyo kumbuyuutar qaab **Desktop Application** ahaan
2. In lagu kaxeeyo borogaraamyada **Browser** calooshooda qaab **Java Applet** ahaan
3. Qaabka saddexaad waxa loo yaqaan **Java Web Start**, waxaanu saamaxayaa in borogaraamyada Java laga istaado Internet-ka, kadibna isticmaaluhu toos ugu isticmaalo kumbuyuutarkiisa sidii qaab **Desktop Application** ahaan

- Labada qaab ee dambe waxay saamaxayaan in borogaraamyada Java laga isticmaalo Internet-ka ama kumbuyuutar kale. Labadaba waxa la adeegsanayaa borogaraamka **Browser**-ka. Qaabka hore borogaraamka Java waxad ku isticmaalaysaa toos caloosha **Browser**-ka, halka qaabka dambe lagu siinaayo link aad ka istaado borogaraamka Java. Saddexdan qaabba waxan idinku tusayaa halkan.

Waxaan rabaa inaynu borogaraamka sare ee tusaalaha ah ee 17.2.1 aynu u isticmaalo saddex qaab ee kala duwan ee aan kor kusoo sheegay. Waxaynu ku bilaabaynaa qaabka ugu horeeya oo an inaynu u isticmaalo qaab **Desktop Application** ahaan. Sida ka muuqata sawirka 17.6 waxad combobox-ka aan calaamadiyey ka doorataa *default config*, kadibna waxad gujisa falaadha cagaarka ah ee midigta u jeedda ee iyana aan calaamadiyey. Intaas kadib waa in borogaraamkii uu istaadhmo kadibna uu kuusoo baxo sida tusaalaha 17.7 oo kale. Xagga muuqaalka waxa suurtoagal inaad iga duwanaato, maadaama aan adeegsanaayo Windows 8, kumbuyuutar uu ku jiro.

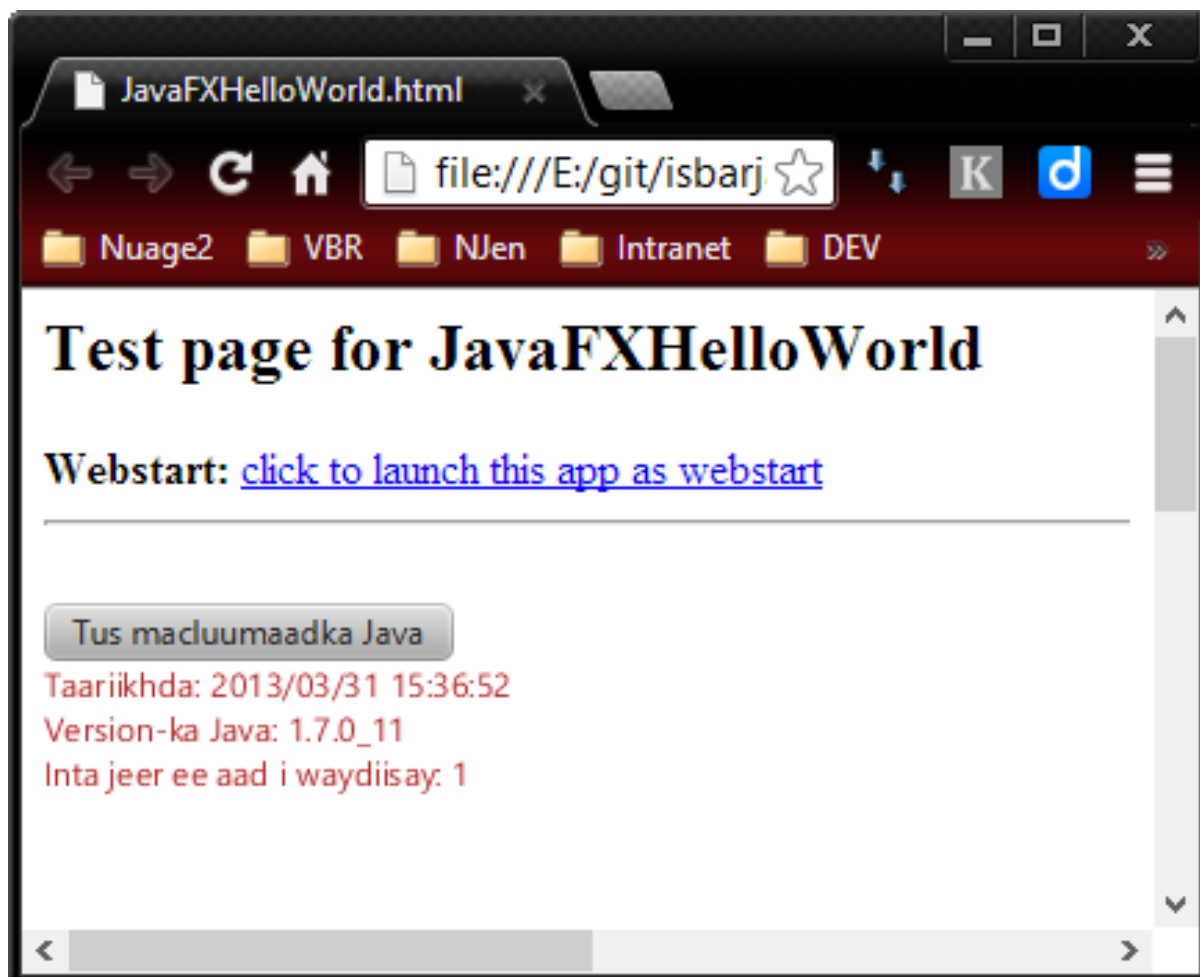


Sawirka 17.6: Qaababka kala duwan ee aad isticmaali karto borogaraamka JavaFX



Sawirka 17.7: Borogaraamka marka loo isticmaalo qaabka Desktop Application

Intaas kadib, si aad u aragto qaabka labaad waxad mar labaad combobox-ka sawirka 17.6 ka doorataa *Run in Browser*, intaas kadib guji falaadda cagaarka ee kaxee mar labaad. Intaas kadib waa in borogaraamkii aynu samaynay ee Java uu ku furmo Browser-ka kumbuyuutarkaaga sida ka muuqata sawirka 17.8. Anigu maadaama aan adeegsanaayo Browser-ka Google Chrome waxa iisoo baxday daaqadda noocaas ah. Adiga waxa kuusoo baxaysa kolba Browser-ka aad adeegsanayso, taas oo noqon karta tan aan anigu isticmaalaayo, Internet Explorer, Firefox ama nooc kale. Muhiimaddu waa inaynu aragno in borogaraamkeena lagu isticmaali karo caloosha borogaraamka Browser-ka.



Sawirka 17.8: Borogaraamka JavaFX marka lagu kaxeeyo Browser-ka caloosheeda

Qaabka ugu dambeeya oo loo yaqaano **Java Web Start** waxad ku istaadhi kartaa adiga oo gujisa link-kiga qaabka hore ee ay ku qorantahay *click to launch this app as webstart* ee badhanka ka sareeya. Natiijada kuusoo baxda waa inay noqoto sida qaabkii ugu horeeyey oo kale. Qaabkani waxay saamaxaysaa in borogaraamyada Java laga istaado meel kasta aduunka aad joogto. Kaliya waxa loo baahanyahay link-kigii aad ka istaadi lahayd sida tusaalahan oo kale. Intaas ayaan kusoo gabogabaynayaa cutubkan ka hadlaayo teknoolojiyadda **JavaFX**.

Jahaynta mustaqbalka

Waxan qabaa in buuggan kadib uu akhristuhu diyaar u yahay inuu si fiican u dhexgalo aduunka Java. Luuqadda Java sidaynu isla aragnayba waxay ka koobantahay qaybo badan, waxaanay u kala baxdaa dhawr nooc. Buuggan kadib, meesha ugu muhiimsan ee aan akhristaha kula talinaayo inuu sifiican u akhristo waa *The Java Tutorials* oo aad ka helayso adarayska

<http://docs.oracle.com/javase/tutorial/>. Waxa fiican inuu akhristuhu qaybaha kala duwan ee tutorial-kaas wada maro. Dhanka buugaagta, waxa jira qaar badan oo bilaash oo Net-ka laga helaayo sida kan

<http://newdata.box.sk/bx/java/httoc.html>. Waxa kale oo jira buugaag badan oo iib ah. Haddii aad helikarto waxad soo iibsataa buugga [7].

Borogaraamyada **Eclipse** iyo **NetBeans** lasoo dag oo kumbuutarkaaga ku kaydso, kadibna isu eeg oo kii aad jeclaysato dooro. Anigu waxaan mudo dheer isticmaalaayey **Eclipse**, walina wax aan kaga guuro maan arag. Ugu dambayntii iskuday inaad wax ka ogaato **Java EE**.

Wixii mawduuc ee faahfaahin u baahan waxan ku dadaalayaa inaan bogga buugga ee www.isbarjava.com kusoo bandhigo. Maadaama uu wakhtigu igu yaraa dhammaadkii buugga, waxan kusoo gabagabeeyey si degdeg ah, sidaas awgeed waxa jira qaladaad badan oo dhanka qoraalka ah, sidaas awgeed waxan rajaynayaa inaad noosoo gudbisaan wixii aad aragtaan. Borogaraamka **nimi** iyo wixii casharo ah ee aan kusoo bandhigay buuggan waxad iyana ka haysaan isla boggaas. Haddii ay jiraan su'aalo aad qabto ama mawduuc aad isleedahay wuxu u baahanyahay faahfaahin dheeraad ah ama waa qabyo, fadlan boggaas noogu soo dir. Waxa kale oo aad isku daydaa inaad dadka kale la qaybsato oo wixii aad barato kusoo bandhigto isla boggaas.

Liiska Ereyada

Access Modifier

Ereyada public, private ama protected oo kantoroolaaya aragga nooca, sifada ama ficilka. [58](#)

Anonymous Inner Class

Nooc nooc kale calooshiisa ku jira oo aan magac lahayn. [164](#)

Apache Ant

Eeg <http://ant.apache.org/>. [23](#)

Argument

Qiimaha ama qiimeyaasha loo gudbinaayo ficilka. [68](#)

Attribute

Sifo.. [58](#)

Bit

Waa shayga aasaaska u ah kumbuyuutaradda, waxaanay yeelan kartaa kaliya laba qiime oo kala ah 0 ama 1. [59](#)

Browser

Waxaynu odhan karnaa waa borogaraamka ugu muhiimsan ee Internet-ka. Wuxu kuu saamaxayaa inaad booqato bogagga Internet-ka ee kala duwan. Kuwa loogu jecelyahay waxa kamid ah Firefox, Chrome, Internet Explorer, Safari, iyo Opera. [5](#), [221](#), [225–227](#)

Bug

Cilad ku jirta borogaraamka. [13](#)

Bytecode

Waa qaabka loo badalaayo borogaraamyada Java kaxaynta ka hor. Faa'iidooyinka ugu wayn ee bytecode waa in lagu kaxaynkaro kumbuyuutarro kala duwan oo OS-yo kala duwan ku jiraan. [16](#)

Class

Af-soomaali waxan ku macneeyay nooc, waa mid ka mida sifooyinka ugu muhiimsan ee fikira OOP. [9](#)

Class Diagram

Mid ka mid ah tusmooyinka UML. [11](#), eeg [UML](#)

Code

Halkan wuxu la macno yahay Source Code. [8](#), eeg [source code](#)

Code re-use

Mid ka mida sifooyinka ugu muhiimsan ee fikira OOP. Sharaxaad dheeraad ah waxad ka helaya cutubka koobaad ee buugga. [9](#)

Command line

Waa borogaraam aan lahayn UI oo inta badan laga hago ama amarro laga siiyo OS-ka. [16](#), [23](#)

Comment

Sharaxis ama sharaxaad. [64](#)

Compiled Language

Luuqadaha kumbuyuutarka qayb kamid ah. Luuqadaha noocan ah borogaraamyada lagu qoro uma baahna borogaraam kale si loo kaxeeyo, ee toos ayaa loo kaxaynkaraa. Luuqadaha noocan oo kale ah waxa kamid ah C++ see. [8](#)

Compiler

Uruurshe. Waa borogaraam lagu badalo qoraalka borogaraamka oo loogu badalo qaab uu kumbuyuutarku fahmaayo. Inta badan qoraal farabadan oo tusaale ahaan lagu qoray C++ ayuu ka dhigayaa hal borogaraam, sidaas awgeed waa uruurshe. [15](#)

Concatenation

Isku dabaqabasho ama isku dhajin laba qoraal qoraalka. [81](#)

Constructor

Ficilka dhisaha.. [41](#), [61](#)

Customer

Rukun. [33](#)

Data

Macluumaad. Macluumaadku wuxu yeelan karaa noocyo badan. [8](#)

Database

Waxa loo soo gaabiyaa marmar DB, waana macluumaad wayn oo si fudud looga raadinkaro waxa loo baahanyahay. Waxa jira borogaraamka loogu talagalay ina maamulaan database-yada, kuwaas oo loo yaqaano Database Management Systems, waxaana kamid ah tusaale ahaan Mysql, Solid, Postgres, iwm. [5](#), [169](#)

Desktop Application

Borogaraamyada miiska sida Word, Excel, Notepad, iwm. [221](#), [225](#), [226](#)

Developer

Magacyada loo isticmaalo dadka borogaraamyada kumbuyuutarka sameeya mid ka mid ah. [xv](#), [15](#)

Development Tools

Borogaraamyada ama agabka lagu dhiso borogaraamyada kumbuyuutarka. [16](#)

Editor

Borogaraamyada qoraalka lagu qoro sida qoraalka borogaraamka. Inta badan waa borogaraamyo ka fudud tusaale ahaan Word oo bilaash lagu helo. [15](#)

Encapsulation

Mid ka mida sifooyinka ugu muhiimsan ee fikira OOP. Sharaxaad dheeraad ah waxad ka helaya cutubka koobaad ee buugga. [9](#)

Environment Variable

Waa qaab ay borogaraamyada kumbuyuutarku isugu gudbiyaan qiimeyaasha kala duwan, tusaale ahaan meesha ama wadada laga helaayo borogaraam kale. [24](#)

Event Listener

Dhacdo dhagayste. [161](#)

Exception

Waxa laga soo gaabiyey exceptional event, waxaanu ka dhiganyahay xaalad aan caadi ahayn ama dhacdo gaar ah. [60](#)

Field

Sifo. [58](#), eeg [Attribute](#)

File

Wuxu ku macneeyey Dr. Barwaaqo [1] ‘Gal’. Anigu waxan inta badan u adeegsaday kaliya fayl. File sideedaba waa shayga ugu muhiimsan ee kumbuyuutarka lagu kaydsado. File wuxu yeelan karaa noocyo badan sida sawir, qoraal, fiidiyaw, hees, iwm. 15

General-purpose Programming Language

Luuqad kumbuyuutar oo guud oo loo isticmaalikaro xaalado kala duwan. Haddii luuqaddu aaney ahayn mid guud, waxay noqonaysaa domain-specific language ama luuqad meel gaar ah loo adeegsado. 5

Generalization

Buuggan waxaynu ula jeednaa, oo la macno yahay Inheritance. 11, *eeg inheritance*

Generic

Waa qaab lagusoo daray luuqadda Java sanadkii 2004. 61

GPA

Heerka celceliska darajada ama dhibcaha waxbarashada uu ka helay ardaygu. 49, 51

Hard Disk

Xasuusta wayn ee kumbuyuutarka. Inta badan qaadkeeda waxa lagu cabiraa GB, tusaale ahaan 4GB, 20GB, 250GB, iwm. Qolofka maskaxda ama CPU-ga kumbyuutarka ayay ku dhexjirtaa. 133

Hardware

Macno badan ayay yeelan kartaa, laakiin marka laga hadlaayo kumbuyuutarka dadku isticmaalaan, waxa loola jeedaa qalabka kala duwan ee uu kumbuyuutarku ka koobanyahay sida: shaashadda, motherboard-ka, jiirka, badhanada, iwm. xvi, 8

HTTP

HTTP waxa laga soo gaabiyey Hypertext Transfer Protocol. Waana protocol-ka ugu muhiimsan ee nidaamka WWW, kaas oo saamaxaaya in miishiinada server-ka ah iyo kuwa kale ee caadiga ah ay isu gudbiyaan qoraalada iyo faylalka kala duwan. Protocol-ku sideedaba waa luuqadda ay isku fahmaan kumbuyuutarrada kala duwan markay rabaan inay wax iswaydaarsadaan. xv

HTTPS

Wuxu lamid yahay HTTP, laakiin wuxu kaga fiican yahay dhanka amaanka iyo in aan macluumaad ay kumbuyuutaradu iswaydaarsanayaan dhexda laga qabsan. xv

IDE

Borogaraam isu uruurinaaya dhammaan waxyaalaha loo baahanyahay si loo sameeyo borogaraamyada kumbuyuutarada ee kala duwan. Waxa jirikara IDE-yo loogu talagalay luuqado gaar iyo gaar guud oo luuqado kala duwan loo adeegsano karo. Borogaraamyada IDE loo yaqaano waxa kamid ah tusaale ahaan Eclipse, Visual Studio, NetBeans, iwm. 16, 21, 222

Identifier

Magacyada kala duwan sida, magacyada noocyada, magacyada sifooyinka, magacyada cilada iyo magacyada doorsoomeyaasha. 57

Inheritance

Inheritance waxay ka midtahay sifooyinka ugu muhiimsan ee fikirka OOP. Af-soomaali waxan ku macneeyey takhasus, sababtoo ah waxay la xidhiidhaa takhasuska caadiga ah ee dadku wada yaqaano. Inheritance waxay saamaxaysaa in noocu, nooc kale ka dhaxlo sifooyin iyo ficillo, kadibna uu ku daro waxyaalo cusub oo isaga u gaar ah ama uu wax ka badalo waxyaalaha uu ka dhaxlay nooca kale. Noocaas kale waxa loo yaqaan base class. 6, 11

Inner Class

Nooc nooc kale calooshiisa ku jira. 163

Input-Output

Input-Output waa qayb muhiim ah oo ka mid ah kumbuyuutarada. IO waxa saamaxaysaa in tusaale ahaan borogaraamku uu macluumaad kasoo akhriyo ama ku qoro qalabka kala duwan ee kumbuyuutarka. [133](#)

Instantiation

Instantiation waa samaynta walax cusub. [10](#)

Interactive

Macno badan ayay yeelan kartaa, laakiin marka aynu ka hadlayno borogaraamyada kumbuyuutarka waxaynu u jeednaa borogaraam ka jawaabaaya dhaqdhaqaaqa isticmaalaha. [5](#)

Interface

Mid ka mida sifooyinka ugu muhiimsan ee fikira OOP. Sharaxaad dheeraad ah waxad ka helaysoo cutubka koobaad ee buugga. [9](#), [170–173](#), [175](#)

Interpreted

Luuqadaha kumbuyuutarka qayb kamid ah. Luuqada noocan oo kale waxay u baahanyihiin borogaraam kale, si loo kaxeeyo. Luuqadda Java waa interpreted, sababto ah borogaraamka JVM la'aantiis ma kaxaynkartid borogaraamyada luuqadda Java lagu qoray. [6](#), [8](#)

Java

Java waa luuqad kumbuyuutar oo guud (general-purpose). Borogaraamyada lagu qoray Java waxad ku isticmaali kartaa kumbuyuutarro kala duwan oo ay ku jiraan Operating System-yo kala duwan. [xv](#)

Java Applet

Borogaraamyoy yaryar oo Java lagu qoray oo la dhexgaliyo borogaraamyada Browser-ka, si ay ugu soo kordhiyaan waxqabad dheeraad ah. [5](#), [221](#), [225](#)

Java Keywords

Ereyada u gaarka ah luuqadda Java. [57](#)

Java Web Start

Waa teknoolojiyad saamaxaysa inaad borogaraamyada lagu qoray luuqadda Java toos uga istaado Internet-ka, adiga oo adeegsanaaya borogaraamka Browser-ka. Arintani waxay saamaxaysaa in borogaraamkii Java laga isticmaalo. [225](#), [227](#)

JavaFX

JavaFX waxa loogu talagalay in lagu dhiso waxa lagu magacaabay Rich Internet Application (RIA) see. [221](#), [222](#), [225](#), [227](#)

JDK

Agabka loo adeegsado dhisidda ama samaynta borogaraamyada Java. Waxay ka koobantahay JRE iyo agabkii loo baahnaa si loo sameeyo borogaraamyada Java. [23](#)

JRE

Agabka loo adeegsado kaxaynta borogaraamyada Java. Waxa ugu muhiimsan JVM. [21](#), [23](#), [221](#)

JUnit

JUnit waa library loo adeegsado qorista tijaabooyinka. [13](#)

JVM

Waa borogaraam lagama maarmaan ah si loo kaxeeyo ama loo isticmaalo borogaraamyada lagu qoray luuqadda Java. Ujeedadiisuna waa inuu kala dhexgalo borogaraamka lagu qoray Java iyo Operating System (OS)-ka, Si borogaraamyada Java aanay toos ugu xidhmin Operating System-ka. [6](#), [8](#), [16](#), [221](#)

Library

Library waxa loo yaqaan borogaraamyoy yaryar oo loogu talagalay in lagu adeegsado borogaraamyada kale, kuwaas oo qabanaaya hawl khaas ah. Inta badan Library-ga kalidii lama isticmaalikaro. [215](#), [221](#)

Linux

Waa Operating System ka duwan Windows-ka. Linux waxyaalaha uu kaga duwanyahay Windows waxa kamid ah inuu yahay bilaash oo cidkastaaba isticmaali karto, waxna ka badalikarto. Linux wuxu kusoo baxaa noocyo badan sida Ubuntu, Red Hat, openSUSE, iwm. Noocyadan waxa loo yaqaan magaca *Linux distributions*. 6

Main memory

Waxa kale oo la odhan kaliya Memory. Waa xasuusta dhexe ee kumbuyuutarka. Marka kumbuyuutarka dabku ka go'o wixii macluumaad ku kaydsanaa xasuustaas way lumayaan. Wuxu [1] ku magacaabay *Is*. xiii

Model

Qayb kamid borogaraamka, gaar ahaan qaybta kaydinaysa macluumaadka muhiim ka ah iyo ficiladda la xidhiidha. 125, 169

Modularity

Mid ka mid ah fikirada ugu mihiimsan ee dhisidda borogaraamyada kumbuyuutarada. Waxaanay ka dhigantahay guud ahaan in arinta la xalinaayo loo jajabiyo ama yaryareeyo qaybo yaryar oo la maamulikaro. 9

Module

Marka laga hadlaayo luuqadaha kumbuyuutarka waa qayb gooni ah, oo qabanaysa hal arin. Luuqadaha C++ iyo Java lagama isticmaalo, laakiin haddii la isticmaalo waxa inta badan loola jeedaa noocyada. 8

MVC

Qaab caan ah oo inta badan la adeegsado marka la dhisaayo borogaraamyada wajiga leh, kaas oo soo ifbaxday 1980-kii. 168–170, 190

NIMI

NIdaamka **Ma**cluumaadka **I**skuulka. 33

Object

Buuggan ereyga aan af-soomaali ku macneeyey waa walax, laakiin wuxu noqon karaa oo kale shay. Waa shayga ugu muhiimsan fikirka OOP, waana dhisme ama ka mid ah qoraalka borogaraamka, kaas oo isu uruurinaaya sifooyinka iyo ficilada. 6, 8

OOP

Object-Oriented Programming (OOP) waa fikir soo caanbaxay 1970-dii oo lagu bartay luuqadda kumbuyuutarada ee Smalltalk. OOP waa hab borogaraamyada kumbuyuutarada loo dhiso, kaas oo ay aasaas u yihiin waxa lagu magacaabay walxaha. 6, 8, 9, 57

OS

Operating System waa borogaraamka ugu muhiimsan kumbuyuutarka. Wuxu maamulayaa qaybaha kala duwan ee kumbuyuutarka, waxaanu fududaynayaa isticmaalkiisa. Waxa kale oo uu saamaxayaa samaynta borogaraamyada kumbuyuutarka, kuwaas oo adeegsanaaya OS-ka qaybihiisa kala duwan. 8, 16, 221, 222

OS X

OS X oo markii hore loo yaqaaney Mac OS X waa Operating System, waxaana soosaarta shirkadda Apple Inc. Mishiinada uu ku shaqeeyo waxa loo yaqaan Mac, kalarkoodu waa cadaan, halka kuwa kale loo yaqaano PC, una badanyihiin kalarka madawga. 6

Package

Qaab kamid ah luuqadda Java oo saamaxaaya in noocyada isku hawl la xidhiidhi meel laysugu uruuriyo. 82

Path Separator

Marka Windows laga hadlaayo waa calaamadda ;, aduunka Linux waxa u dhiganta calaamadda :. Waxa kale oo iyana jirta *Separator* oo Windows marka la joogo ah \, Linux-na /. 24

Performance

Waa fulinta ama xawaaraha kumbuyuutarka guud ahaan. Waxa kale oo ay noqon kartaa xawaaraha borogaraamka. [6](#)

Platform neutrality

Platform neutral ama cross-platform waxay ka mid tahay sifooyinka borogaraamyada kumbuyuutarka. Gaar ahaan waxa lagu magacaabaa kuwa aan ku xidhayn Operating System ama nidaam gaar ah, oo lagu isticmaalikaro OS-yo kala duwan. Tusaale ahaan byte code-ka Java waa platform neutral, sababtoo ah waxad ku kaxaynkartaa platform-yo kala duwan. [6](#)

Plugin

Borogaraam yar oo diyaar ah oo inta badan lagu kordhiyo borogaraam kale si uu ugu qabto arin khaas ah. Borogaraamyada Browser-ka ayaa aad u adeegsada plugin-yada. Wuu ka duwanyahay Library-ga. Library-ga waxa la adeegsadaa marka borogaraamka la samaynaayo, laakiin plugin-ka borogaraamka oo socdo ayaa lagu kordhinkaraa. [222](#)

Pointer

Buuggan waxan loola jeedaa arin ka mid ah luuqadda kumbuyuutarada ee C++. [6](#)

Polymorphism

Mid ka mida sifooyinka ugu muhiimsan ee fikira OOP. Sharaxaad dheeraad ah waxad ka helaysaa cutubka koobaad ee buugga. [9](#)

Primitive Data Types

Noocyada asalka ah ee Java. [59](#)

Processor

Wuxu ku macneeyey Dr. Barwaaqo [\[1\]](#) ‘Alooseen’. Si kastaba waa shayga ugu muhiimsan kumbuyuutarka, oo tusaale ahaan ku shaqa leh kaxaynta borogaraamyada. [xiii](#)

Program

Software application ama software program waa borogaraamyada kumbuyuutarka. Waxa tusaale ahaan kamid ah Word, Excel, iwm. [xvii](#)

Programmer

Wuxu la macno yahay halkan Developer. [15](#), *eeg developer*

Refactor

Wax ka badalka ama wanaajinta qaabka borogaraam loo sameeyey ama qayb kamid ah loo sameeyey. [12](#)

Release

Marka laga hadlaayo borogaraamyada kumbuyuutarka ereygan waxa loola jeedaa soosaarka borogaraamkaas. Markasta oo wax lagu daro ama laga badalo wuxu yeelanayaa soosaar cusub oo kii hore ka dambeeyey. [5](#)

RIA

Haddii aan soo gaabiyo waa borogaraamyadii miiska ee Desktop-ka oo loo raray Internet-ka, laguna isticmaalikaro borogaraamka Browser-ka. [221](#)

Screen

Screen waa shaashadda ma muraayadda kumbuyuutarka. Waxa kale oo loo isticmaalaa magaca *monitor*. [xiii](#)

SDK

Software Development Kit waa agab aad u baahantahay si aad u samayso borogaraamyo kala duwan. Inta badan nidaam kastaaba wuxu leeyahay SDK u gaar ah. SDK-da waxa ka mid ah tusaale ahaan compiler-ka, library-yada, iwm. laakiin inta badan kuma jirto Editor-kii lagu qorilaha qoraalka borogaraamka. IDE-da ayaa midaynaysa SDK-da iyo waxyaalaha ka dhiman ee aad u baahantahay sidaad u samayso borogaraamyo kala duwan. [16](#)

Senior Software Designer

Shakhsi ama qof ku shaqaysta qorshaynta, dhisidda, dajactirista, tijaabinta, iwm. borogaraamyada kumbuyuutarka, gaar ahaan qof mudo shaqadaas hayey. [xiii](#)

Server

Waa kumbuyuutarro inta badan ka quwad wayn kuwa caadiga ah oo sababo kala duwan loo adeegsado. Tusaale ahaan qaybinta bogagga Internet-ka kuwa loo isticmaalo waxa la yidhaahdaa Web Server. Waxyaaba kale ee loo isticmaalo waxa ka mid tusaale ahaan Email-ka. Maadaama ay yihiin mishiino loo baahanyahay marwalba inta badan lama damiyo oo 24/7 way shidanyihiin. [5](#), [6](#)

Software

Wuxu halkan la macno yahay ereyga program. [xvi](#), [eeg program](#)

Software Build Tool

Agab ama borogaraamyo loo adeegsado fududaynta dhisidda borogaraamyada kumbuyuutarada. [23](#)

Software Version

Waa nambar ama magac loo bixiyo soosaarka kala duwan ee isla hal borogaraam. Tusaale ahaan borogaraamka **Eclipse** waxaynu buuggan ku adeegsanay version-kiisa 3.6 oo loo yaqaano Galileo iyo version-ka 3.7 oo isna loo yaqaano Helios. [24](#)

Source Code

Source code ama code waa qoraalka borogaraamka, ka hor intaan la compile-garayn. Source code-ka waxa lagu qoraa luuqadda kumbuyuutarka oo ay dadku fahmayaan, sida Java, C++, C#, iwm. Source code-ku inta badan wuxu u badanyahay qoraal text ah, laakiin waxa kale oo ku jiri kara sawirro, fiidiyaw, iwm. [9](#), [15](#)

Spagetti Code

Magac loo isticmaalo qaabkii ay u eekaayeen borogaraamyadii hore, kuwaas oo sida baastada isugu qasnaa, oo aan lahayn wax dhisme ah. [6](#)

Subclass

Nooc-hoosaad. Nooc laga soo dhiraandhiriyeey nooc kale. [11](#)

Superclass

Nooca sare. Mid ka mid ah noocan noocyadiisa sare. [11](#)

Syntax

Israaca erayga ama erayo ee luuqad. [57](#)

TDD

Mid ka mid qaababka loo sameeyo borogaraamyada kumbuyuutarka, kaas oo ay xudun u tahay tijaabinta ama qorista tijaabooyinku. [12](#), [13](#), [33](#), [37](#), [41–43](#), [46](#), [48](#), [52](#), [87](#), [95](#), [117](#), [143](#)

Test

Waxay halkan la macno tahay Test Case see. [55](#)

Test Case

Hal tijaabo.. [55](#)

Test Suite

Guruub tijaabooyin ah oo laysu uruuriyey. [55](#)

Tester

Qof tijaabiye ah, gaar ahaan borogaraamyada kumbuyuutarada tijaabiya. [12](#)

Top-Level Component

JFrame iyo JDialog. [159](#)

Ubuntu

Mid ka mid ah OS-yada bilaashka ah ee Linux. [21](#)

UML

Waa luuqad tusmo ama sawir-ahaaneed oo ay soosaartay Object Management Groupin (OMG) sanadkii 1997. Waxa ugu muhiimsan ee inta badan laga adeegsado luuqadda UML waa class diagram-yada, laakiin in ka badan 10 diagram oo kala duwan ayaa lagu sawirikaraa. Muhiimadda ugu wayn uu UML waa fududaynta isfahamka dadka kalawadan ee borogaraamka wada dhisaaya iyo sawir-kabixinta qaabka borogaraamka loo sameeyey iyo qaybaha kala duwan ee uu ka koobanyahay. [11](#)

UML Diagram

Sawirrada lagu sameeyey luuqadda UML ayaa inta badan loo yaqaan UML diagram. [11](#)

Unit Testing

Nooc kamid ah noocyada kala duwan ee borogaraamyada kumbuyuutarada loo tijaabo, si looga saaro wixii cilado ku jira. [12](#)

User

Isticmaale. [16](#)

Web Application

Web.... [221](#)

Windows

Operating System-ka Windows, waxa soosaarta oo iska leh shirkadda weyn ee Microsoft. Wakhtigan qoraalka buugga soosaarkii ugu dambeeyey waa Windows 8. Waa OS-ka ku jira kumbuyuutarada ugu badan aduunka ee maanta la isticmaalo. [6](#)

Word Processing

Isticmaalka borogaraamyada loogu talagalay in qoraalka lagu qorto sida Word, si aad qoraallo kala duwan ugu qorto. [xv](#)

WWW

Waxa laga soo gaabiyey World Wide Web (oo marmar loo soo gaabiyo W3 ama Web), waxaanu ka midyahay adeegyada Internet-ka. Waxa la odhan karaa waa adeegga ugu muhiimsan. Wuxu saamaxayaa in faylal kala duwan oo kumbuyuutarro kala duwan oo meelo kala duwan yaalla meel kaliya laysugu keeno ama laysku xidho. Tusaale ahaan bogagga Internet-ka macluumaadka aysoo bandhigayaan waxa suurtoagal inay ka yimaadoon kumbuyuutarro kala duwan oo meelo kala duwan yaalla. Waxa la hindisay sanadkii 1991. [xv](#)

XP

Waa qaab ama nidaam la raaco marka la dhisaayo borogaraamyada kumbuyuutarka, kaas oo soo baxay dhamaadkii 1990-kii. TDD waxay ka midtahay nidaamka wayn ee XP.. [12](#)

Tixraac

- [1] Cabdirahmaan C. Faarax “Barwaaqo”. *Hal-bixinta Ereyada Kumbuyuutarka (Computer terms in Somali)*. Hal-aqoon Publishers, 2007.
- [2] Thomas L. Case David A. Stamper. *Business Data Communications*. Prentice Hall, sixth edition edition, 2003.
- [3] J.F. DiMarzio. *JavaFX A Beginners Guide Build Interactive Desktop, Browser, and Mobile Applications*. The McGraw-Hill Companies, first edition edition, 2011.
- [4] Boobe Yuusuf Ducaale. *Diiwaanka maansada Cabdillahi Suldaan Maxamed (Timacadde)*. Flamingo Printing Press, daabacaaddii 2aad edition, 2005.
- [5] Tommi Mikkonen Kai Koskimies. *Ohjelmistoarkkitehtuurit*. Talentum Media Oy, 2005.
- [6] Lasse Koskela. *TEST DRIVEN Practical TDD and Acceptance TDD for Java Developers*. Manning Publications Co., 2008.
- [7] Jeff Langr. *Agile JavaTM Crafting Code with Test-Driven Development*. Prentice Hall, 2006.
- [8] Ahmed Hussein Mire. *Somali-Abwaanary. Advanced English-Somali dictionary*. -, second edition edition, 2009.
- [9] Oracle. Javafx documentation home — javafx 2 tutorials and documentation, January 2013.

Tilmaame

7zip, 23

Access Modifier, 58, 59

Anonymous Inner Class, 164

Apache Ant, 21, 23

Applet, 5

Argument, 68

Arithmetic, 70

Assignment, 70

Attribute, 58, 65

Bit, 59

Block, 74

Browser, 5

Bug, 13

Bytecode, 16

C++, 5, 8

Class, 9, 57

class diagrams, 11

Code, 8

Code re-use, 9

Command line, 24

Commandline, 16

Comment, 64

Compiler, 16

Component, 157

Concatenation, 81

Conditional, 73

Constructor, 41

Control, 157

Control Flow Statement, 75

Control Panel, 24

Customer, 33

Data, 8

Developer, xv, 15

Eclipse, 16, 21

Editor, 15, 16, 21

Encapsulation, 9

Enum, 51

Environment Variable, 24

Equality, 72

Event Listener, 161

Exception, 60

Expression, 74

Extends, 58

Field, 58, 65

File, 15, 17

GEdit, 22

General-purpose programming language, 5

Generic, 61

GPA, 49

Hardware, 8

Hello World, 24

HTTP, xv

HTTPS, xv

IDE, 16

Identifier, 57

Inheritance, 6, 11, 49

Inner Class, 163

Input-Output, 133

Instantiation, 10

Interactive, 5

Interface, 9, 171

Java, xv

Java IO, 133

Java Keywords, 57

JAVA_HOME, 24

JDK, 23

JRE, 21, 23

JVM, 6, 8, 16, 17, 23

Linux, 6, 16, 21

Local Variable, 65

Main Memory, xiii

Menu, 17

Method, 60

Method overloading, 152

Method overriding, 152

Modularity, 9

Netbeans, 16

NIMI, 33

Notepad, 22

Notepad++, 22

Null, 69

Object, 6

Object-Oriented Programming, 6

OOP, 8, 57

Operand, 70

Operating System, 8

Operator, 69

OS, 16, 21, 22

OS X, 6

Pass-by-reference, 68
Pass-by-value, 68
PATH, 24
Path, 24
Path Separator, 24
Performance, 6
Platform, 6
Pointer, 6
Polymorphism, 9
Primitive Data Types, 59
Private, 58
Processor, xiii
Program, xvii
Programmer, 15, 23
Project, 25
Protected, 58
Public, 58

Relational Operator, 72
Release, 5

Screen, xiii
SDK, 21
Server, 5
Shell, 24
Software Version, 24
Source Code, 9, 15, 57
Statement, 74
Stream, 133
Swing, 157
Syntax, 57

TDD, 12, 33
Test, 55
Test Case, 55
Test Suite, 55
Tester, 12
Type Casting, 204

Ubuntu, 21, 22
UML diagrams, 11
Unary, 71
Unit Testing, 12
User, 16
Utility, 60

Windex, 16
Windows, 6, 22
WWW, xv

ZIP, 23

