

Chapter 1:

Overview of Object Oriented Programming & Java Platform

Programming Paradigms

In whatever form it is organized a **program** is always **data and code**.

The different **way of organization** of the code and the data, it is **called Programming Paradigms**.

A Programming Paradigms can be:

1. Unstructured
2. Structured

Unstructured Programming Paradigms

Code is written as a single unit(main function)

- Data is globally shared in the unit
- Uses goto to implement flow control
- Sometimes called monolithic or spaghetti code
- Code contains lot of repetition
- Difficult to debug, maintain and reuse
- Does not allow parallel programming

Example of unstructured programming language:

JOSS, FOCAL, TELCOMP, assembly languages, MS-DOS batch files, and early versions of BASIC, Fortran, COBOL, and MUMPS

Structured Programming Paradigms

Uses structured **flow control structures, block structure and modules** to improve clarity, quality and development time of a program.

- Each module keeps **its own data**
- Easy to **debug, maintain**
- Facilitate **reusability**
- Allows **parallel programming**

Based on the way the data and code organized, we have two structured paradigm

1. Procedural Model
2. Object Oriented Model

Procedural Model

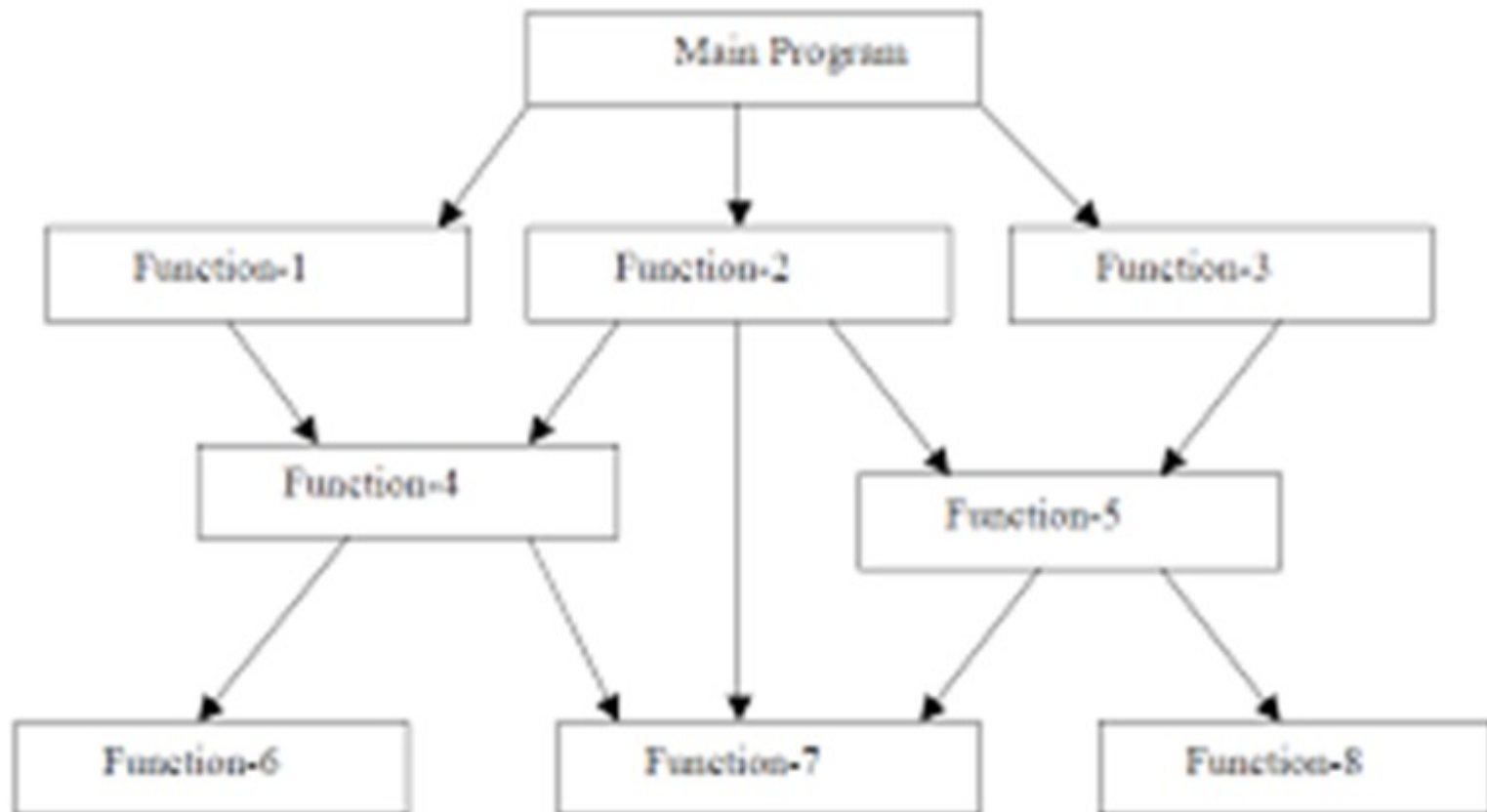
This approach **organizes a program around its code**(with a well defined interface)

- Characterized as **code controlling access to data**
- It tries to breakdown a program **as interaction of procedures**(series of computational steps) commonly known as routines, subroutines or functions
- Interaction is made with **one module calling another module**

FORTRAN ALGOL COBOL BASIC Pascal

Procedural...

A single program which is modeled as interaction of procedures, function or modules



Object Oriented Model

Object-oriented programming organizes a program **around its data(that is, objects)** and a set of **well-defined interfaces** to that data.

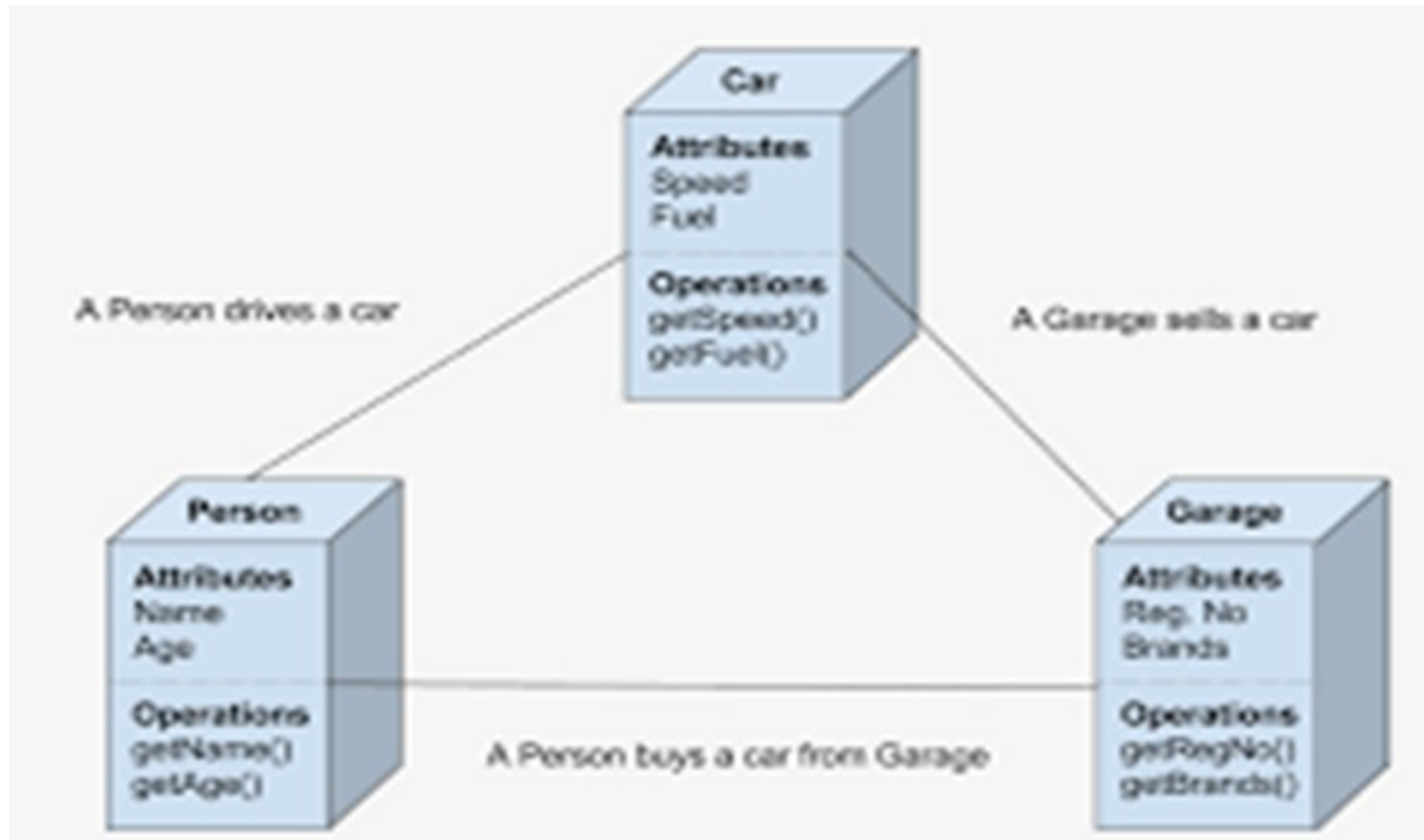
- characterized as **data controlling access to code.**
- breakdown program **as interaction of objects with a well defined interfaces.**
- Objects encapsulate data and operation
- Objects interact one object sending a message to another

a **message** is the way that one program object requests an action from another object

A message has four parts: identity of the recipient object, code to be executed by the recipient, arguments for the code and return value

Object...

A single program which is modeled **as interaction of objects**



Procedural Vs Object Oriented Model

Procedural Oriented

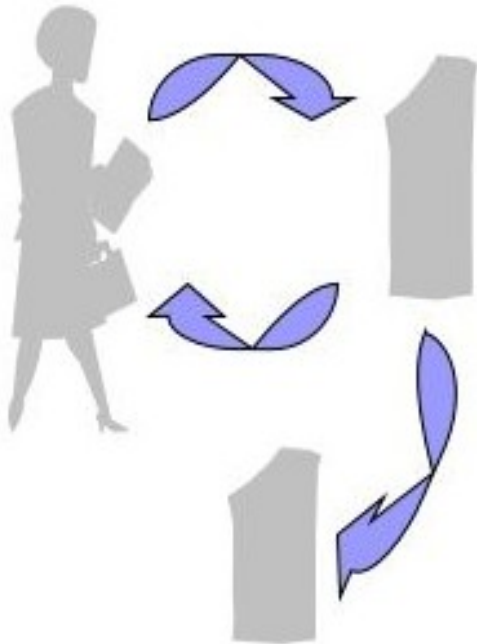
- Code driven, not focusing on data primarily
- Unit is procedure
- Uses top down approach
- No access specifier
- No data hiding-less secure
- Not easy to add new data and function
- No inheritance feature
- Not modeling real world

Object Oriented

- Data driven, not focusing on code primary
- Unit is Object
- Uses bottom up approach
- Uses access specifier such as private, public, protected etc.
- Uses data hiding-more secure
- Easy to add new data and function

Procedural Vs...

■ Procedural



Withdraw, deposit, transfer

■ Object Oriented



Customer, money, account

Basic Object Oriented Concepts

Key Concepts of Object Oriented:

1. Abstraction
2. Model
3. Object
4. Class
5. Association
 1. Composition
 2. Aggregation

Abstraction

- Abstraction is one of the **key concepts** of object-oriented programming (OOP) languages.
- It is a fundamental technique **used to perceive the world**
- Its main goal is to **handle complexity by hiding unnecessary details** from the user.
- Objects are described by their **own property not by their make up component details**
- In order to drive a car, I don't have to know how it internally works

Model

A model is an abstract representation of a real world object

- shows only the important features or aspect of the real world object which is relevant for the consideration
- is not a real world but merely human construct
- A real object can have several model

Object oriented modeling

Object oriented modeling is a way of **thinking about problems using models organized around real world concepts.**

- OO designs starts with a visual modeling of the problem domain, involving both **the domain expert and software designers**
- Enables better communication and visualization for **all stakeholders**

Class

A class is a **blueprint or template** for creating objects

- It is a **concept**(human, dog, car), not an object(like you, your dog, your car)
- It is an **abstract specification** of data(**attributes**) and **behavior** which is common for a particular kind of objects
 - **Attribute(fields)**: data defining the state of the objects
 - **Behavior(methods)**: represents action for the class

Object

A specific instance of a class

- Concrete not concept
- Possesses attributes(data) and operations(methods)
- Consists of
 - 1. State:** represented by attributes of an object
 - 2. Behavior:** represented by method of an object
 - 3. Identity:** represented by unique name to an object
- Has well defined interface that enable it to interact with other objects
- **Example 1:** Dogs
 - **States:** name, color, breed, and “is hungry?”
 - **Behaviors:** bark, run, and wag tail
- **Example 2:** Cars
 - **States:** color, model, speed, direction
 - **Behaviors:** accelerate, turn, change gears

Characteristics Object

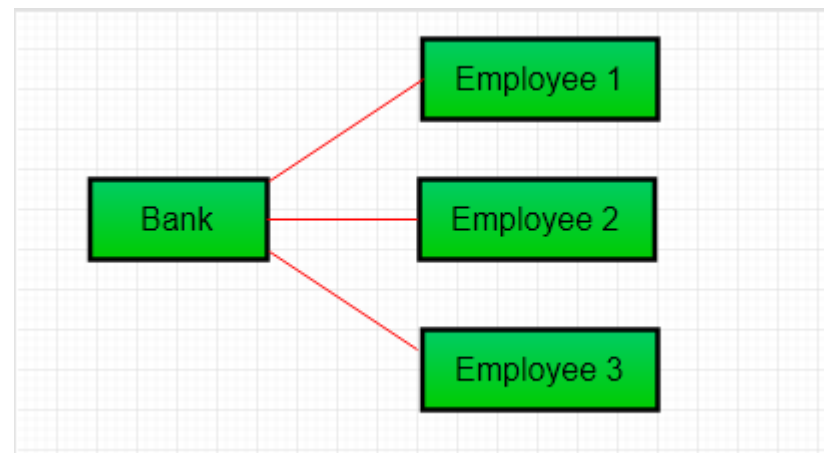
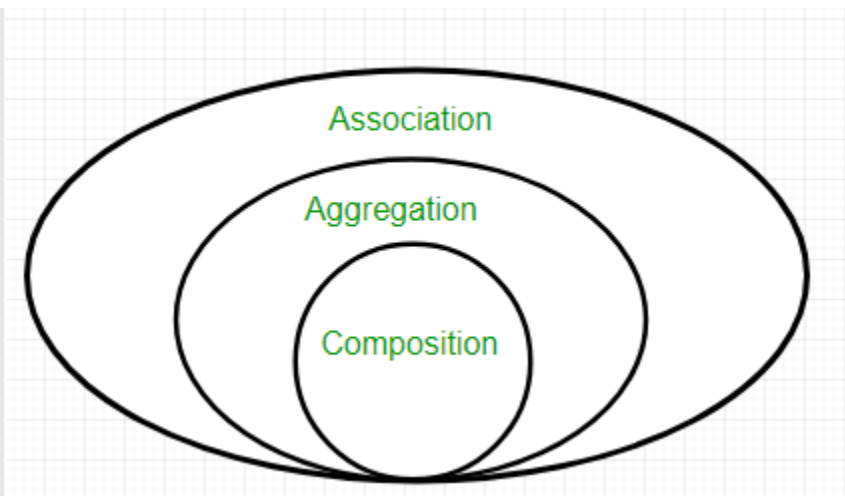
- They have unique identity.
- They fall into categories, or classes.
- They fall into hierarchies or aggregations.
- They have well defined behaviors & responsibilities.
- They separate interface from implementation.
- They hide their internal structures.
- They have states.
- They provide services.
- They send messages to other objects.
- They receive messages from other objects, and react appropriately.
- They delegate responsibilities to other objects.

Association

Association is relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many.

Forms of association:

1. **Aggregation**
2. **Composition**

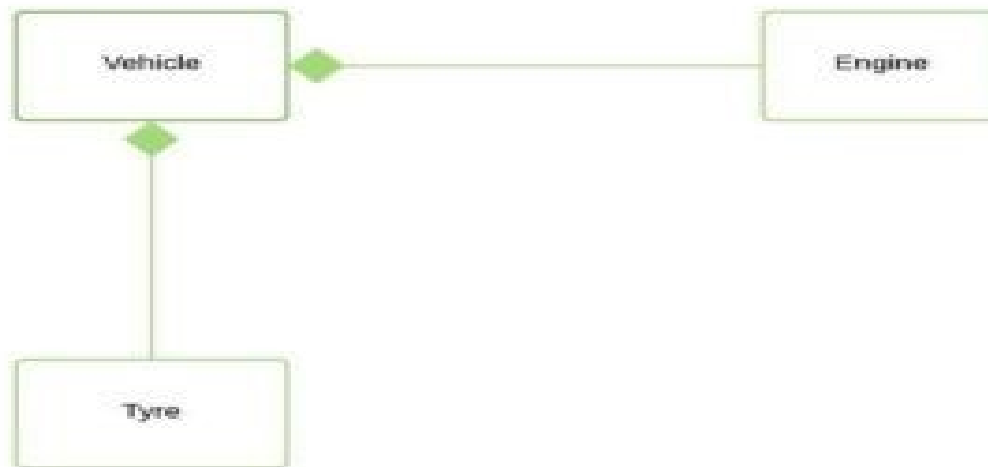


One to many
association

Composition

Composition is a relation established when one class owns other classes and other classes can not meaningfully exist, when it's owner destroyed

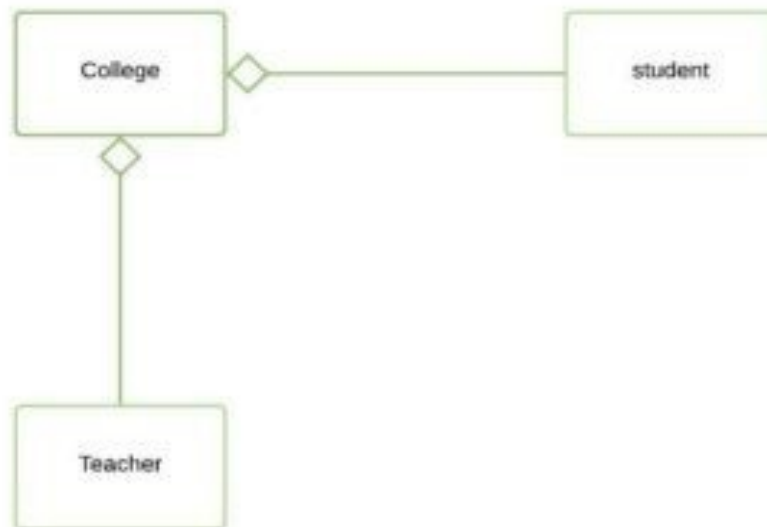
- It means that if we destroy the owner object, its members also will be destroyed with it.
- **a has** relationship
- it is often referred to as strong association



Aggregation

Aggregation is a relation established when one class owns other classes and other classes can exist, when it's owner destroyed

- Both of them can exist independently of each other.
- Has a relationship
- it is often referred to as weak association.



OOP principles

The three basic underlying principle of Object Oriented Programming are:

1. Encapsulation
2. Inheritance
3. Polymorphism

Encapsulation

Encapsulation is the mechanism that **binds together code and the data** it manipulates, and keeps both **safe from outside interference and misuse.**

- **implemented in java using class**
- Access to the code and data inside the capsule is tightly controlled through a **well-defined interface.**
- Exposing only the **client's view** of the object's responsibilities.
- The only thing that an object knows about another object is the public object's interface
- Protects other objects from relying on details that are likely to change.
- Hiding the “gears and levers”.
- It allows the developer to separate an object's

Inheritance...

Inheritance is the process by which one object **acquires the properties** of another object

We view the world as made up of objects that are related to each other in a hierarchical way.

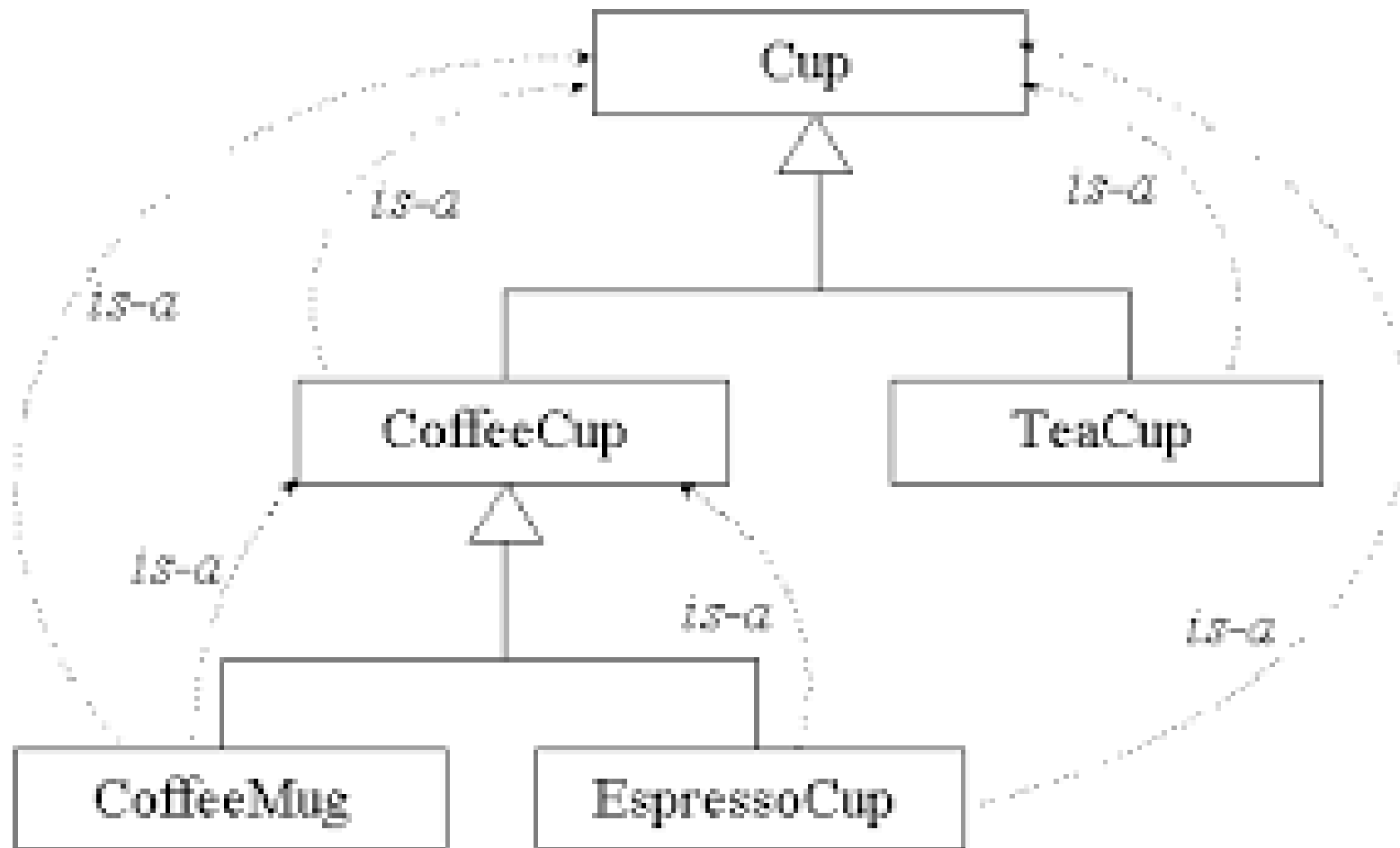
Living thing->animals->mammal->dog

A dog has all attributes of mammal plus its own special attributes that specialize it.

A mammal has also all the property of animal plus its own special attributes.

An animal has also all the property of

Inheritance...



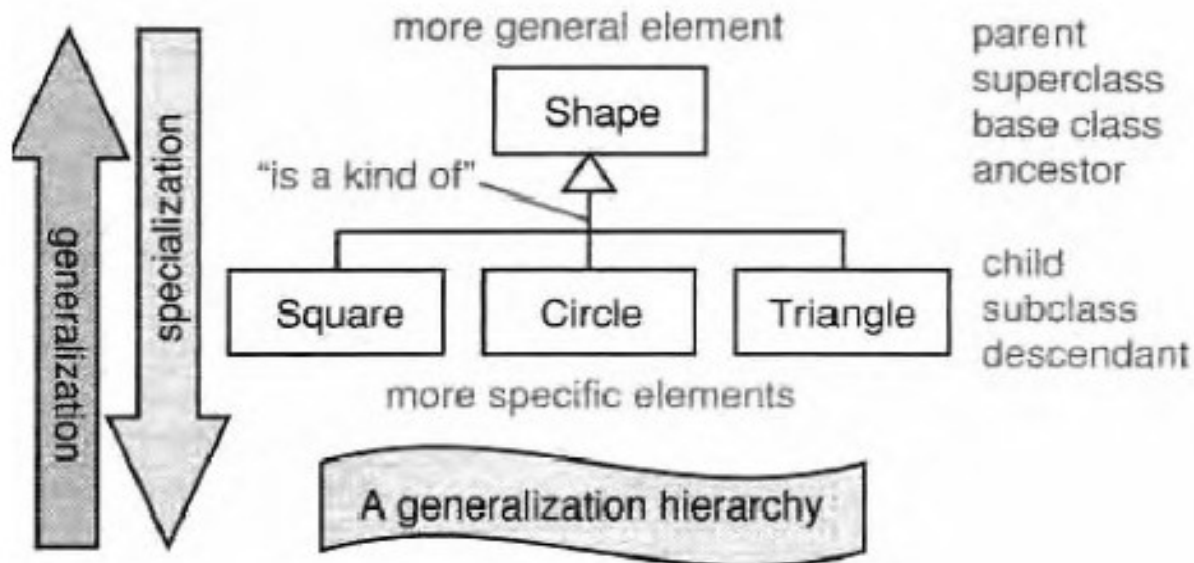
Inheritance

Inheritance allows programmers to create a new class that is built upon existing classes.

- The new class inherits the attribute and methods of the existing class
- The newly created class that inherits from a class is called **a child class, derived class or a subclass**
- The existing class which is inherited is called **the parent class, base class or superclass**
- **Is relationship**(the derived class **is** a type of the base class)

Why Inheritance?

- allows reuse of code.
- specialization(focus on difference)
 - *specialization* means creating new subclasses from an existing class.
- generalization(focus on commonality)
 - *Generalization* is the process of extracting shared characteristics from two or more classes, and combining them into a generalized



Polymorphism is the ability of an object to take on many forms.

There are 2 types of polymorphism:

1. Static Polymorphism

This is also mentioned as **Compile-Time polymorphism, Static binding, Compile-Time binding, Early binding and Method overloading.**

- Same method name is overloaded with different type or number of parameters in same class (different signature).
- Targeted method call is resolved at compile time.

2. Dynamic Polymorphism

This is also mentioned as **Run-Time polymorphism, Dynamic binding, Run-Time binding, Late binding and Method overriding.**

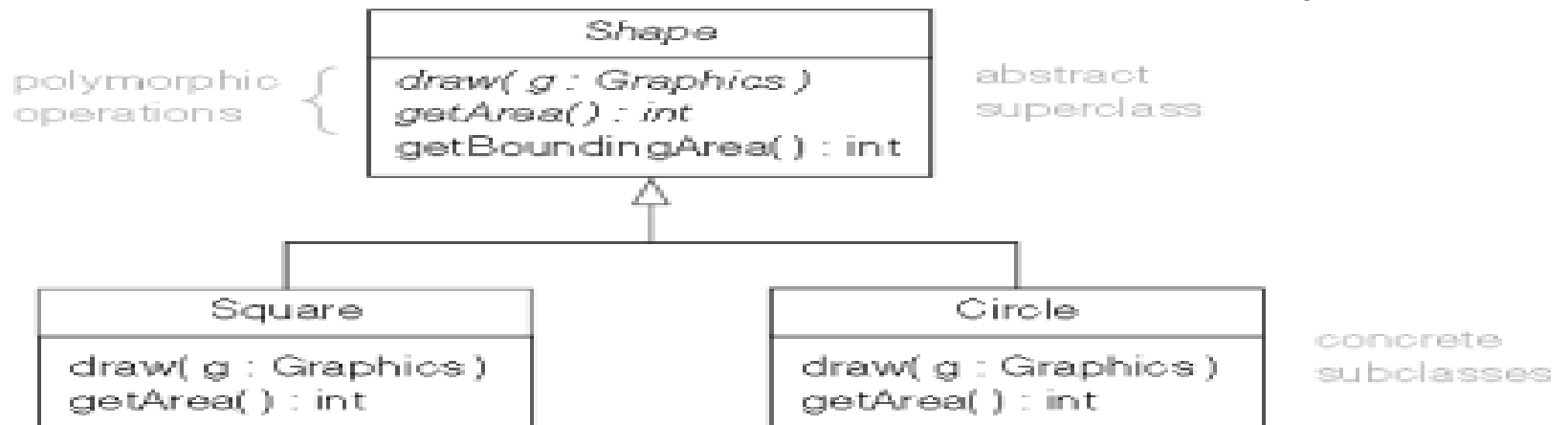
- Here having many forms is happening in different classes.
- Same method is overridden with same signature in different classes.

Targeted method call is resolved at run time

Dynamic Polymorphism

Dynamic Polymorphism is a feature that allows one interface to be used for a general class of actions.

- any child class object can take any form of a class in its parent hierarchy and of course itself as well
- this actually means is that the child class object can be assigned to any class reference in its parent hierarchy and of course itself as well.
- “one interface, multiple methods.”. A single interface will be implemented differently by different objects.
- Each objects respond in its own way
- A draw method can be used to draw different shape



OOP Principles...

Polymorphism, Encapsulation, and Inheritance Work Together.

When properly applied, polymorphism, encapsulation, and inheritance combine to produce a programming environment that supports the development of far **more robust and scalable** programs than does the process-oriented model.

Question

Suppose we want to develop a dama game program

1. Identify the different objects in the problem domain
2. Model each objects
3. Show the relationship in your models

What is Java?

Java is a **general-purpose programming** language that is **class based, object-oriented** and designed to have as **few implementation dependencies** as possible.

It is intended to let application developers **write once, run anywhere any time(Portable)**

History of Java

Java is the successor of **C and C++**. Java was originally conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991.

Detailed history of java is left reading exercise

Pillars of Modern Programming Lang

- Speed
- Portability
- *Security*

Features of Java?

1. Simple

- java's design goal: **simplify, simplify, simplify**
- *simple to use and learn*
- *Syntax borrowed from C/C++ (so easier for programmers to learn it after C++)*
- *No preprocessor directive commands*
- *Java provides package for organizing classes*
- *Java has removed many complicated and rarely-used features, for example, **explicit pointers, operator overloading**, etc.*
- *There is no need to remove unreferenced objects because there is an **Automatic Garbage Collection in Java**.*
- *Does not support **multiple inheritance***

Features of Java?

2. Object-Oriented

- Everything in java is object.
- Structured in terms of classes
- Java can be easily extended since it is based on the Object model.

3. Portable

- Java is portable because it facilitates you to carry the Java bytecode to any platform.

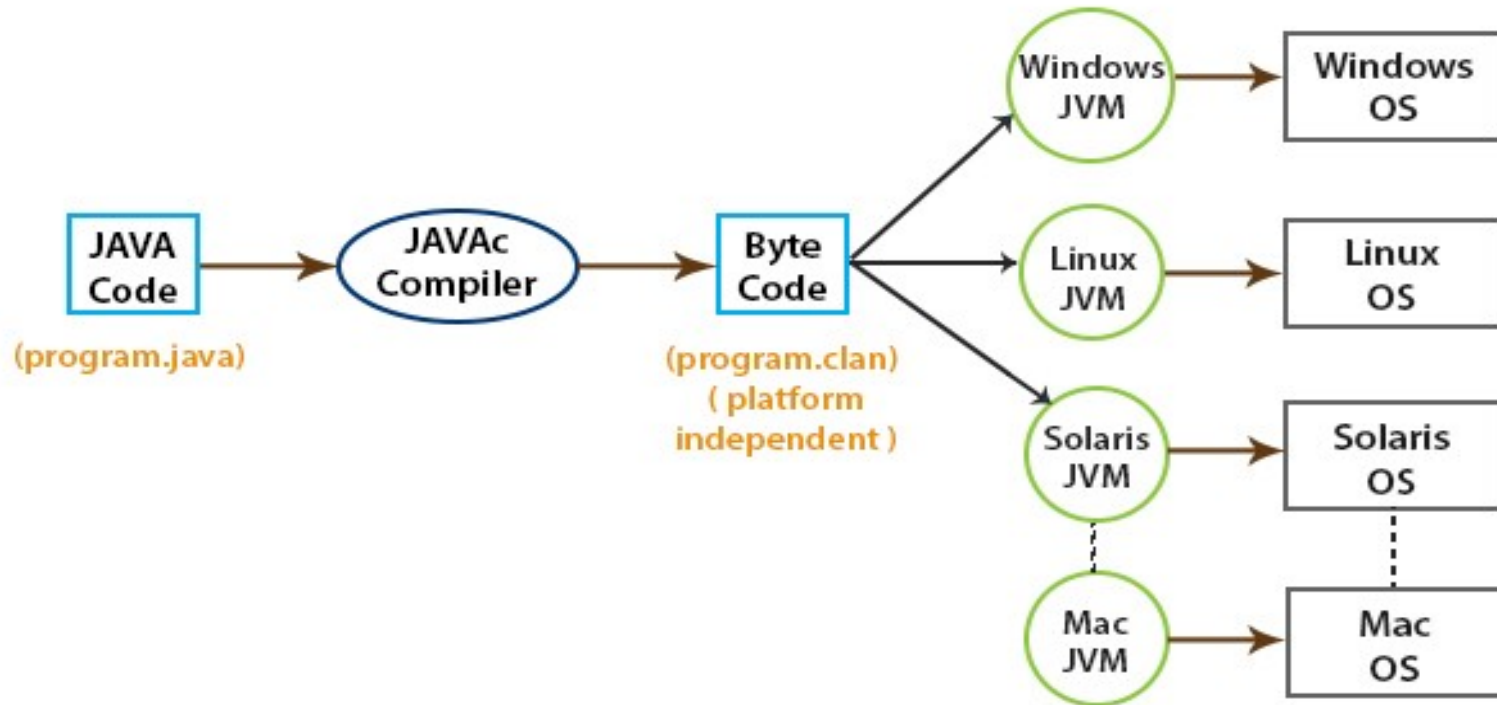
Write Once, run anywhere.

- Java also tries to balance functionality, speed, and portability

Portability....



Platform Independence in Java



Features ...

4. Architectural neutral

- Java is architecture neutral because there are no implementation dependent features, for example, **the size of primitive types is fixed.**
- Changes and upgrades in operating system, processors and system resources will not force any changes in the java program.
- **Run anywhere + Run anytime**

5. Secured

- Three mechanisms for security
 - The bytecode is verified by the VM ensuring that no illegal code is included.
 - The code could be signed, i.e. cryptographic algorithms are applied to ensure that the code originates from someone the user trusts.
 - Uses the so called "sandbox"-model:
 - The program runs in a memory area of the system, which is controlled by the virtual machine.
 - The program which is allowed to perform any operation within the sandbox, has limited access to

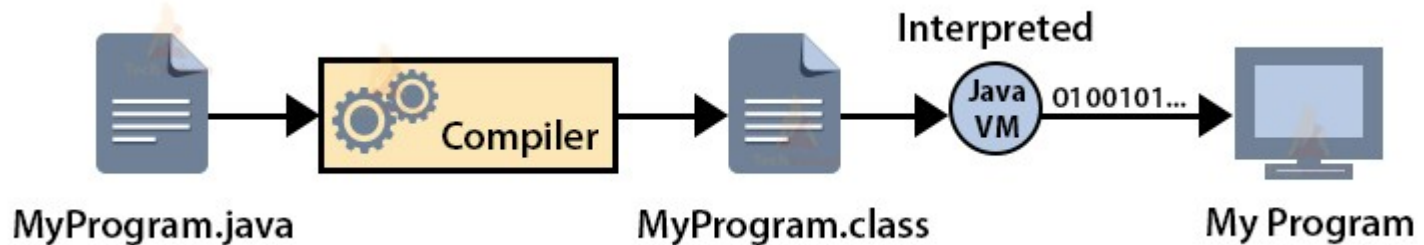
4. Robust

- Robust software doesn't "break" easily because of programming bugs or logic errors in it.
- Strongly typed language
 - every variable must have a declared type.
 - The Java language is designed to enforce type safety.
 - Type safety means that a program cannot perform an operation on an object unless that operation is valid for that object.
 - Data type issues and problems are resolved at compile-time
 - all assignments, whether explicit or via parameter passing in method calls, are checked for type compatibility
 - Any type mismatches are errors that must be corrected before the compiler will finish compiling the class
- does not support direct pointer manipulation or arithmetic
 - makes it impossible for a Java program to overwrite

Features ...

4. Compiled and interpreted

- Java integrates the power of Compiled Languages with the flexibility of Interpreted Languages.
- The Java compiler (javac) compiles the java source code into the bytecode.
- Java Virtual Machine (JVM) then executes this bytecode which is executable on many operating systems.



5. High Performance

Java is faster than other **traditional interpreted**

Features ...

4. Multithreaded

Java supports Multithreading or concurrency

5. Distributed

- Java is specifically designed to work within **a networked environment**
- Users can **create distributed applications** in Java
- Java has a feature called **Remote Method Invocation (RMI)** using which a program can invoke method of another program across a network and get the output.
- Classes also are provided for establishing **socket-level connections**.

6. Dynamic

- Java gives the facility of **dynamically linking** new class libraries, methods, and objects
- Java even supports functions written in other languages such as C and C++ to be written in Java programs.. These methods are **dynamically linked at runtime**.

Java platform consists of the following components.

- **The JVM(Java Virtual Machine)** is an implementation of a virtual machine which executes a Java program.
 - responsible for functions like Load and Store, Arithmetic calculation, Type conversion, Object Creation, Object Manipulation, Control Transfer, Throwing exception, etc.

Components of the JVM are:

- **Class Loaders**: used to load class files. Whenever we run the java program, class loader loads it first.
- **Run-Time Data Areas**: It is one of the Data Area in JVM
 - **Class area**: storage for class elements
 - **Heap**: runtime storage allocation
 - **Stack**: storage for local variables and partial results

- **Execution Engine**

- **Interpreter:** executes the bytecode line by line.
- **Just-In-Time (JIT):** compiler optimizes the process of converting byte code to machine-specific language.
- **Java Garbage collector:** is responsible to get rid of objects which are not being used by a Java application anymore. It is a form of automatic memory management.

- **The JRE(Java Runtime Library)** is a bundle of software components used to run Java applications.

Core components of the JRE include:

- **An implementation of a Java Virtual Machine (JVM)**
- **Classes required to run the Java programs**
- **Property Files, java configuration files**

Components of Java Platform

- The JDK(Java Development Kit) provides environment and tools **for developing, compiling, debugging, and executing** a Java program.

Core components of JDK include:

- JRE
- Development Tools
 - **javac** – reads class and interface definitions and compiles them into class files(bytecode)
 - **java** – launches the Java application
 - **javadoc** – generates HTML pages of API documentation from Java source files
 - **appletviewer** – enables us to run Java applets without a web browser
 - **jar** – packages Java applets or applications into a single archive

1. Standalone Application or desktop application

- These are traditional software that we need to install on every machine.
- AWT and Swing are used in Java for creating standalone applications.

2. Web Application

- An application that runs on the server side and creates a dynamic page is called a web application.
- Currently, Servlet, JSP, [Struts](#), [Spring](#), [Hibernate](#), [JSF](#), etc. technologies are used for creating web applications in Java.

3. Enterprise Application

- An application that is distributed in nature, such as banking applications, etc. is called enterprise application
- In Java, [EJB](#) is used for creating enterprise applications.

4. Mobile Application

- An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

1. Java SE (Java Platform Standard Edition):

It has concepts for developing software for **Desktop based (standalone) CUI (command user interface) and GUI (graphical user interface) applications, applets, database Interaction application, distributed application, and XML parsing applications.**

2. Java EE (Java Platform Enterprise Edition):-

It has concepts to develop software for **Web applications, Enterprise applications, and Interoperable applications.** These applications are called high-scale applications. Some examples are:- banking and insurance-based applications.

3. Java ME (Java Platform Micro Edition):-

It has concepts to develop software for **consumer electronic devices means embedded systems**, like mobile and electronic level applications. Java ME was popular for developing mobile gaming applications. This edition was called micro because these edition programs are embedded in small chips. The program embedded in the chip is called micro (small).

1. **Eclipse:** is an **integrated development environment (IDE)** which is used for almost every language using Plugins.(Open Source)
2. **Netbeans:** is another **IDE** which is used for developing java codes. .(Open Source)
3. **JUnit:** is a **Testing Framework** for a Java programming language. .(Open Source)
4. **Apache Spark:** It is an **open source framework** that is used for large scale applications across clustered computers.
5. **Jenkins:** It is used for **Testing** and is an open source framework which is written in Java programming language.
6. **IntelliJ IDEA:** It is an **IDE** developed by JetBrains. It is available as Community edition and Proprietary edition
7. **Android Studio:** It is used for developing apps on every type of Android device.
8. **Apache Maven:** It is a **project management tool** based on the concept of the Project Object Model (POM).
9. **Gradle:** It is an open source **automation tool**. It is based on the concept of Apache Ant and Apache Maven.
10. **Jcreator:** It is an open source **IDE** created by Xinox Software
11. **Source:** It is a light java **IDE** which uses Swing components.

Installation and setup of java

1. Installing java(JRE)

If you are only interested in running java application,

- Download Java/JRE from the following link
 - https://download.cnet.com/Java-Runtime-Environment-JRE/3000-2213_4-10009607.html
 - Run the downloaded installer and follow the instruction

2. Install JDK

if you are interested in developing and running java application

- Download JDK from the following link
- Run the download
<https://www.oracle.com/java/technologies/javase-jdk15-downloads.html>
- ed installer and follow the instruction

The JRE is installed also when you install JDK

3. Install Eclipse or Netbeans

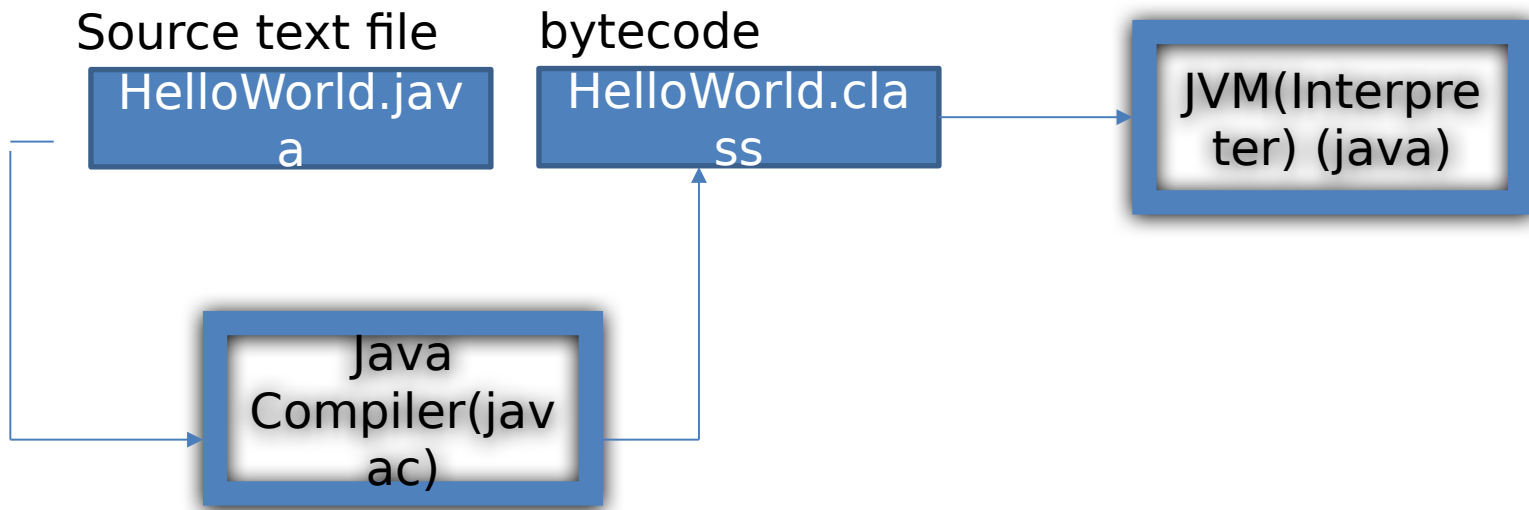
- Download Eclipse or Netbeans from the following link
 - <https://www.eclipse.org/downloads/>
 - <https://downloads.apache.org/netbeans/netbeans/12.2/Apache-NetBeans-12.2-bin-windows-x64.exe>
- Run the downloaded installer and follow the instruction

Installation...

4. Setting Environment variable

- Type control panel in windows search box
- Click control panel
- Click System
- Click Advanced System Setting
- On Advanced tab click Environment Variables
- Go to system variable box and select the Path option and click edit
- Click New
- and then click Browse
- Specify where the jdk binary files are located
If you use the default installation it will be located
C:\Program Files\Java\jdk-14.0.2\bin

Compilation of Java source



The compiled bytecode is independent of platform