



Object Oriented Programming

(CSE 2202)

Lecture One and Two

Part-I

Object Oriented Concepts & Java Programming basic

Objectives

After studying this chapter, students should be able to:

- Learn about Programming Paradigms
 - Structured Vs Object-Oriented paradigm
- Understand OOP Concepts and Features
- Fundamental Java Programming Structures:
 - Tokens: Identifiers, Separators, Operators, keywords & Literals
 - Constants, variables and primitive data types
 - The main() method
 - Control statements: if, switch, while, do...while, for and Jumps in loops
 - Array & Strings in java

Contd...

OOP Approach

- A modern programming paradigm that allows the Programmer to model a problem in a real-world fashion as an object.
- Major objective is to eliminate some of the flaws/faults/errors encountered in the procedural approach.
- OOP allows us to decompose a problem into a number of entities called objects and then build data and methods (functions) around these entities.
- The data of an object can be accessed only by the methods associated with the object
- Follows bottom-up approach in program design

Contd...

Features of OOP

- Emphasis is given to both data and procedure.
- Programs are divided into objects/ classes.
- Data Structures/attributes/fields are designed such that they characterize the objects.
- Methods that operate on the data of an object are tied together in the data structure.
- Data is hidden from external world and can not be accessed by external functions.
- Objects may communicate with each other

Basic OOP Concepts

The following are the basic OOP concepts:

1. Objects
2. Classes
3. Data Abstraction
4. Data Encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing

Contd...

1.Object

- An object is an any real world entity which may represent place, person, data item related to program.
- An object has **state, behavior and identity**.
- Ex. A 'Mouse' object has,
 - State: moving or working
 - Behavior: picking or pointing an object on the screen
 - Identity: color, company, Identification No. etc.
- An object is a variable/instance of class.
- An object is a run-time entity.

2.Class

- Is the template or blueprint that defines the states and the behaviors common to all objects of a certain kind.
- It is a collection of objects of similar type.
- Classes are user-defined data types & behave like the built-in types of programming language.

Contd...

3. Data Abstraction

- **Abstraction** means representing essential features without including the background details or explanations
- A classes can use the concept of *Abstraction* and are defined as list of abstract data and functions to operate on these data.
- Since the classes use the concept of Data Abstraction, they are known as *Abstract Data Type(ADT)*.

4. Data Encapsulation

- The wrapping up of data and methods into a single unit (called class) is known as **encapsulation**.
- This insulation of the data from direct access by

Contd...

5. Inheritance

- Is the process by which objects of one class acquire the properties of objects of another class.
- It provides the idea of reusability(reusing the code)

6. Polymorphism

- In polymorphism, '*Poly*' means **many** and '*morph*' means **forms**, i.e. *many forms*.
- Is the ability to take more than one form. It allows a function responding in different ways.

7. Dynamic Binding

- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at runtime.

Contd...

8. Message Passing

- The process of invoking an operation of an object is called *Message Passing*.
- In response to the given message, the respective method or operation is called.
- OOPs includes objects which communicates by sending/ receiving information with each other.
- Message Passing involves *specifying the name* of the object, *the name of the function* (message) and *the information* to be sent.

employee.salary (name);

- A message for an object is a request for the execution of a procedure.

Features of Java (Java Buzzwords)

- **Simple:** Learning and practicing Java is easy because of resemblance with C and C++.
- **Object Oriented:** Unlike C++, Java is purely OOP.
- **Distributed:** Java is designed for use on network; it has an extensive library which works in agreement with TCP/IP.
- **Secure:** Java is designed for use on Internet. Java enables the construction of virus-free, tamper free systems.
- **Robust (Strong/ Powerful):** Java programs will not crash because of its exception handling and its memory management features.
- **Portable:** Java does not have implementation dependent aspects and it yields or gives same

Contd...

- **Interpreted:** Java programs are compiled to generate the byte code(.class file). This byte code can be interpreted by the interpreter contained in JVM.
- **Architectural Neutral Language:** Java byte code is not machine dependent, it can run on any machine with any processor and with any OS.
- **High Performance:** Along with interpreter there will be JIT (Just In Time) compiler which enhances the speed of execution.
- **Multithreaded:** Executing different parts of program simultaneously is called multithreading. This is an essential feature to design server side programs.
- **Dynamic:** We can develop programs in Java

C++ Vs Java

- Java does not support operator overloading.
- Java does not have template classes as in C++.
- Java does not support multiple inheritance.
 - This is accomplished using a new feature called “interface”
- Java does not support global variables.
 - Every variable and method is declared within a class and forms part of that class.
- Java does not use pointers
- Java has replaced the destructor function with a finalize() function
- There are not header files in Java.

Java API

- It includes hundreds of classes and methods grouped into several functional packages.
- Most commonly used packages are:
 - **Language Support Package:** a collection of classes and methods required for implementing basic features of java.
 - **Utilities Package:** a collection of classes to provide utility functions such as date and time functions.
 - **Input/output Package:** a collection of classes required for input/output manipulation.
 - **Networking Package:** a collection of classes for communicating with other computers via Internet.
 - **AWT Package (Abstract Window Tool kit package):** contains classes that implements platform-independent graphical user interface.
 - **Applet Package:** includes set of classes that allows us to create java applets.

Programming Structure

- In the Java programming language:
 - A program is made up of one or more *classes*
 - A class contains one or more *methods*
 - A method contains program *statements*
- In Java, first we need to import the required packages. By default, `java.lang.*` is imported. Java has several such packages in its library.
- A package is a kind of directory that contains a group of related classes and interfaces. A class or interface contains methods.
- Since Java is purely an Object Oriented Programming language, we cannot write a Java program without having at least one class or object. So, it is mandatory to write a class in Java program. We should use class keyword for this purpose and then write class name.

Contd...

```
// comments about the class
public class MyProgram
{
    // comments about the method
    public static void main (String[] args)
    {
    }
}
```

class header (points to `public class MyProgram`)

method body (points to the curly braces of the `main` method)

method header (points to `public static void main (String[] args)`)

class body (points to the entire class structure from `{` to `}`)

Comments can be placed almost anywhere

5. Operators

- Are symbols that take one or more arguments (operands) and operates on them to produce a result.
- Are used in programs to manipulate data and variables.
- They usually form a part of mathematical or logical expressions.
- Expressions can be combinations of variables, primitives and operators that result in a value.

Java Operators

- There are 8 different groups of operators in Java:
 - Arithmetic operators
 - Relational operators
 - Logical operators
 - Assignment operator
 - Increment/Decrement operators
 - Conditional operators
 - Bitwise operators
 - Special operators

Variables

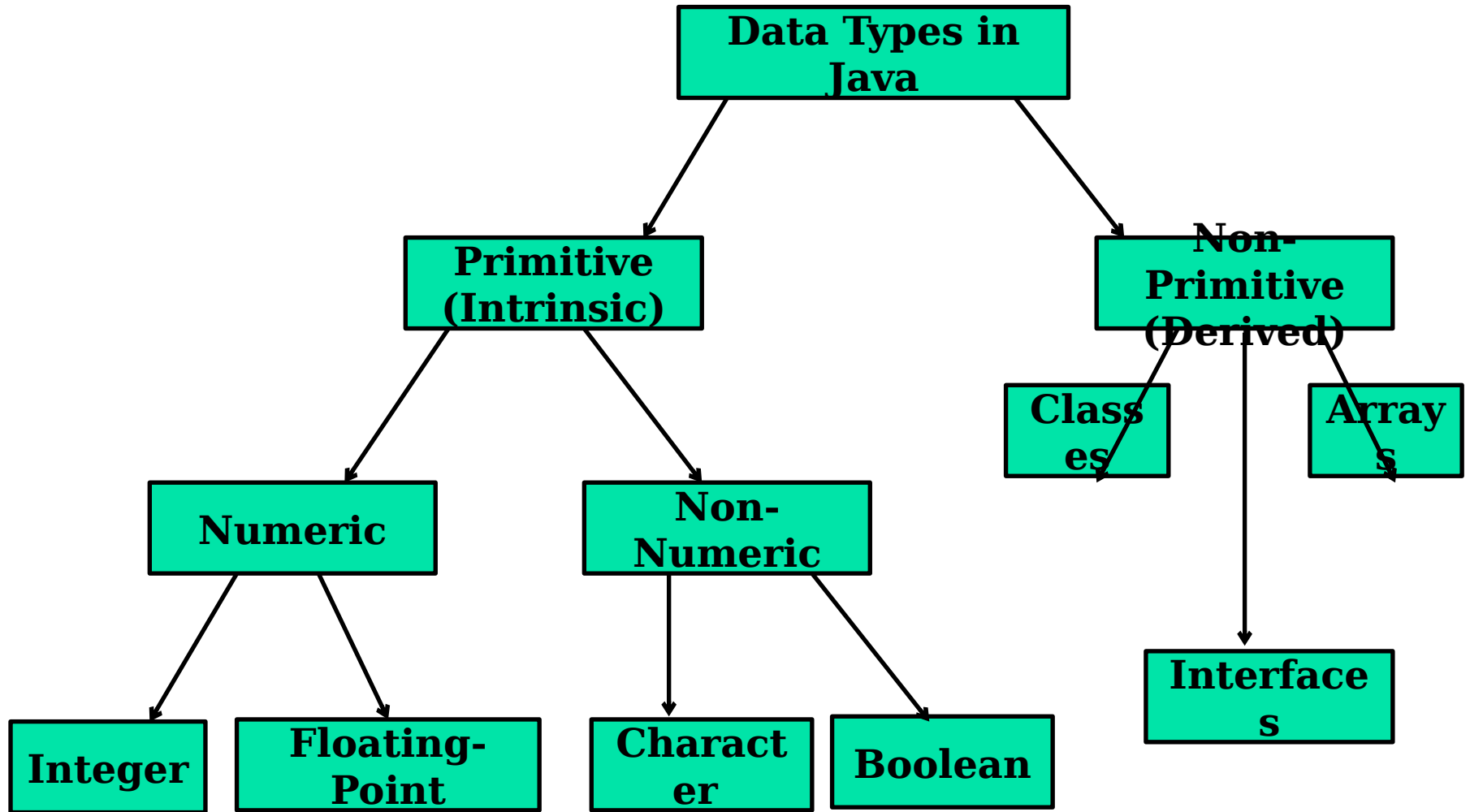
- A variable is an identifier that denotes a storage location used to store a data value.
- Unlike constants, that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program.
- It is good practice to select variable names that give a good indication of the sort of data they hold:
 - For example, if you want to record the size of a hat, `hatSize` is a good choice for a name whereas `qqq` would be a bad choice.

Contd...

- Variable names may consist of alphabets, digits, the underscore (_) and dollar (\$) characters, subject to the following conditions:
 1. They should not begin with a digit.
 2. Keywords should not be used as a variable name.
 3. White spaces are not allowed.
 4. Uppercase and lowercase are distinct. i.e. A `rose` is not a `Rose` is not a `ROSE`.
 5. Variable names can be of any length.

Data Types

- Every variable in Java has a data type.
- Data types specify the size and type of values that can be stored.
- Java language is rich in the data types.
- Java data types are of two type:
 - Primitive Data Types (also called intrinsic or built-in data types)
 - Non-Primitive data Types (also known as Derived or reference types)



Data Types in Java

Primitive Data Types

- There are eight built-in data types in Java:
 - 4 integer types (`byte`, `short`, `int`, `long`)
 - 2 floating point types (`float`, `double`)
 - Boolean (`boolean`)
 - Character (`char`)
- All variables must be declared with a data type before they are used.
- Each variable's declared type does not change over the course of the program.

Read Statement

- It is also possible to assign value for variables interactively through the keyboard using the *readLine()* method which belongs to the *DataInputStream* class.
- The *readLine()* method reads the input from the keyboard as a string which is then converted into the corresponding data type using the data type wrapper classes.
- The wrapper classes are contained in the *java.lang* package.
- Wrapper classes wrap a value of the primitive types into an object.
- The keywords *try* and *catch* are used to handle any errors that might occur during the reading process.

// A program to read data from the Keyboard

```
import java.io.DataInputStream;
```

```
public class Reading{
```

```
    public static void main(String[] args)
```

```
{
```

```
    DataInputStream in = new DataInputStream(System.in);
```

```
    int intNumber=0;
```

```
    float floatNumber=0.0f;
```

```
    double doubleNumber=0.0;
```

```
    String Name=null;
```

```
    try{
```

```
        System.out.println("Enter your Name: ");
```

```
        Name=in.readLine();
```

```
        System.out.println("Enter an Integer Number: ");
```

```
        intNumber=Integer.parseInt(in.readLine());
```

```
        System.out.println("Enter a float Number: ");
```

```
        floatNumber=Float.parseFloat(in.readLine());
```

```
        System.out.println("Enter a Double Number Number: ");
```

```
        doubleNumber=Double.parseDouble(in.readLine());
```

```
    }
```

```
    catch(Exception e){ }
```

```
    System.out.println("Hello : "+Name);
```

```
    System.out.println("The Integer Number is : "+intNumber);
```

```
    System.out.println("The Float Number is : "+floatNumber);
```

```
    System.out.println("The Double Number is :
```

```
    "+doubleNumber);
```

```
    }
```

```
}
```

Example-2

```
import java.util.*;

public class Demoif
{
    public static void main(String k[ ]){
        Scanner t=new Scanner (System.in);
        System.out.println("pls enter the integer
value");

        int x = t.nextInt();
        char grade ;
        if(x>= 85){
            grade = 'A';
        }
        if(x >= 70 && x < 85){
            grade = 'B';
        }
        System.out.println("x = " + x + " and grade =
"+ grade);
    }
}
```

Scope of Variables

- 1. Instance Variables:** are declared in a class, but outside a method, constructor or any block.
 - are created when an object is created with the use of the key word 'new' and destroyed when the object is destroyed.
 - They take different values for each object
- 2. Class Variables:** are also known as static variables, are declared with the *static* keyword in a class, but outside a method, constructor or a block.
 - Are global to a class and belong to the entire set of objects that class creates.
 - Only one memory location is created for each class variable.
- 3. Local Variables:** are variables declared and used inside methods.
 - Can also be declared inside program blocks that are define between { and }.

Scope of Variables

- 1. Instance Variables:** are declared in a class, but outside a method, constructor or any block.
 - are created when an object is created with the use of the key word 'new' and destroyed when the object is destroyed.
 - They take different values for each object
- 2. Class Variables:** are also known as static variables, are declared with the *static* keyword in a class, but outside a method, constructor or a block.
 - Are global to a class and belong to the entire set of objects that class creates.
 - Only one memory location is created for each class variable.
- 3. Local Variables:** are variables declared and used inside methods.
 - Can also be declared inside program blocks that are define between { and }.



THANK YOU!
Q?