UNIT I INTRODUCTION TO OOP AND JAVA FUNDAMENTALS 10

Object Oriented Programming - Abstraction - objects and classes - EncapsulationInheritance - Polymorphism- OOP in Java - Characteristics of Java - The Java Environment - Java Source File - Structure - Compilation. Fundamental Programming Structures in Java - Defining classes in Java - constructors, methods -access specifiers - static members -Comments, Data Types, Variables, Operators, Control Flow, Arrays , Packages - JavaDoc comments.

1.1. Object Oriented Programming (or) OOP in Java

Object Oriented Programming is a programming approachwhich is based on the objects and similar will be used to write the implementation. It allows programmers to create the objects andfunctions to perform those objects. Manipulating these objects to get outcome is the goal of Object Oriented Programming.

Object Oriented Programming popularly known as OOP, is used in a present programming language like Java

1.1.1. Abstraction

S It refers to the act of representing necessary features without including the background details/explanation.

S It is a method of creating a new data type that is matched for a specific application. For example, while driving a car, you do not have to be worried with its internal working. Here you just need to worry about parts like steering wheel, Gears, accelerator, etc.

1.1.2.Objects and Classes

(Example program Refer any class based program in unit 1 and 2)

S Objects are real world entities in an object oriented system.
Ex: person, place, bank account.

1. Instance of a class is called an Object, and there can be several instances of an object in a program. An Object contains both the data member and the function, which operates on the data. For example - chair, bike, marker, pen, table, car, etc.

Class:

S Collection of objects are called as objects

Ex: Mango, apple & orange are member of the class fruits.

2. The class is a collection of similar entities. It is only logical component and not the physical unit. For example, if you had a class called "Expensive Cars" it could have objects similar to Mercedes, BMW, Toyota, etc. Its properties(data) can be price or speed of these cars. While the functions may be performed with these cars are driving, reverse, braking etc.

1.1.3. Encapsulation

J The wrapping up of data and functions in to a single unit is known as encapsulation.

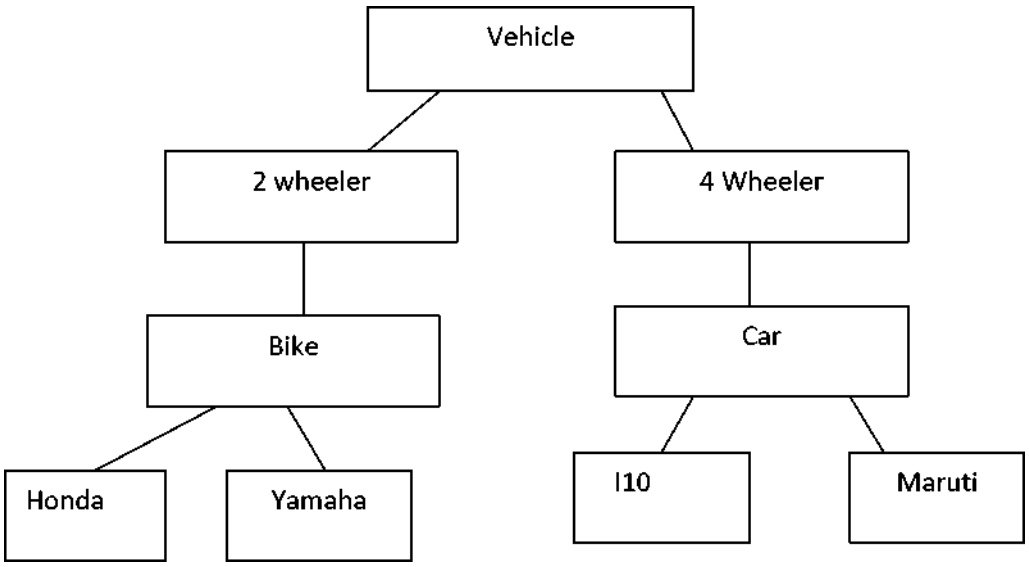
J Encapsulation is an OOP method of wrapping the data and code. In this OOPS idea, the

variables of a class are always concealed from other classes. It can only be accessed using the functions of their current class. For example - in school, a student cannot live without a class.

1.1.4. Inheritance

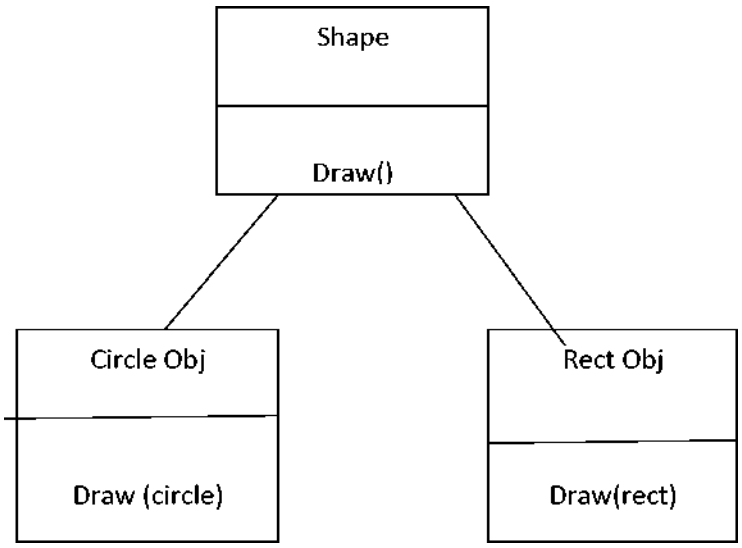
(Example program Refer unit-2)

- Objects of one class take the properties of objects of another class.
- Inheritance is an OOPS concept in which one object holds the properties and behaviors of the parent object. It's creating a parent-child relationship among two classes. It offers robust and natural method for organizing and structure of any software.
-



1.1.5.Polymorphism

- The ability to take more than one form.
- Polymorphism refers to the capability of a variable, object or method to take on multiple forms. For example, in English, the verb "run" has a various meaning if you use it with "a laptop," "a foot race, and ''business. Here, we recognize the meaning of "run" based on the other words used beside with it.



//compile time polymorphism
// Method overloading (add is overloading method -type, arguments and operations are different)

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

//Run time polymorphism example
// Method overriding (run is overriding method -type, arguments are same and operations is //different)

```
class Bike
{
void run() //Method overriding
{
System.out.println("running");
}
}
class Splender extends Bike
{
void run() //Method overriding
{
System.out.println("running safely with 60km");
}
public static void main(String args[]){
   Bike b = new Splender();//upcasting
```

```
   b.run();
}
}
```

Output: running safely with 60km.

1.1.6. Dynamic Binding

J The code related with a given method call is not known until the moment of call at runtime.

1.1.7. Message Communication

The process of programming in an OO language represented below

1 .Creating classess that describe objects and their behavior.

2 .Creating objects from class definitions.

Establishing communication among objects through message passing..

1.2. CHARACTERISTICS OF JAVA:

The various features of java programming are listed down below

Simple :

- Java is very comfortable to write and read.
- Java has a brief, cohesive set of concepts that makes it easy to learn and use.
- Most of the concepts are derived from C++ thus making Java learning process simple.

Secure :

- Java program cannot damage other system thus making it secure.
- Java presents a secure means of creating Internet applications.
- Java presents secure way to access web applications.

Portable :

- Java programs can execute in any system environment with the help of Java runtime system.(JVM-Java Virtual Machine)
- Java programs can be run on any platform (Linux,Window,Mac)
- Java programs can be used over world wide web (e.g applets)

Object-oriented :

- Java programming is object-oriented programming language.
- Like C++ java provides most of the object oriented concepts.
- Java is a complete OOP. Language. (while C++ is semi object oriented)

Robust :

- Java provides error-free programming by being severely typed and performing runtime checks.

Multithreaded :

- Java provides integrated support for multithreaded concepts.

Architecture-neutral :
- Java is not joined to a specific machine or operating system architecture.
- Machine Independent

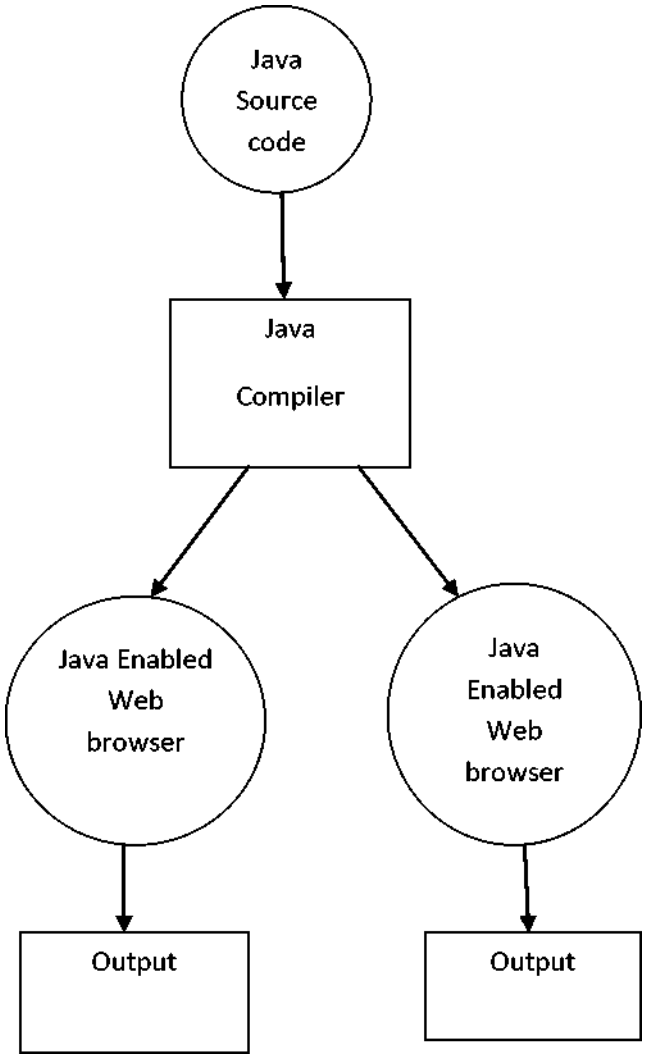
Interpreted:
- Java provides cross-platform code through the use of Java bytecode.
- With the help of JVM,Bytecode can be interpreted on any platform

High performance:
- Bytecodes are extremely optimized.
- JVM can execute them much quicker.

Distributed :
- Java was designed with the distributed background.
- Java can be transmitting,run over internet.

Dynamic :
Java was designed to adjust to an evolving environment:
- Even after binaries have been released, they can adjust to a changing environment
- Java loads in classes as they are wanted, even from across the network
- It defers lots of decisions (like object layout) to runtime, which solves several version problems like C++.

1.3. The Java Environment(java source file,structure,compilation)



JVM -Java Virtual Machine
A Java virtual machine (*JVM*) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages and compiled to Java bytecode

JDK- Java Development Kit
he Java Development Kit (*JDK*) is a software development environment used for developing Java applications and applets

JRE-Java Runtime Environment
JDK includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

Overview:
Two types of Java Programs: 1.Stand alone applications 2. Web applets

1.Stand alone: Can read and write files and perform certain operations. Any **java** class with a main function can be considered a **standalone java application**.

2.Applet: Small java programs that will be used for internet application.An applet located on a remote computer (server) can be downloaded via internet & executed on a home computer (client) using Java capable browser.

Difference between Java & C

- Java has method to define classes & objects.
- Java does not consist of the C unique statements keywords goto, sizeof & typedef.
- Java does not have the data types struct, union and enum.
- Java does not describe the type modifiers keywords auto, extern, register, signed and unsigned.
- Does not contain a pre processor and we cannot use #define, #include & # ifdef statements.
- Java requires that the methods with no arguments must be declared with none parenthesis & not with the void keyword as done in c.

Difference between Java & C++

- Operator overloading is not possible in java programming
- Java does not have template classes as in c++.
- Java does not support multiple inheritance of classes.(indirectly used with new feature called interface).
- It does not support global variables. Every variable &function is declared within a class & forms part of that class.
- Pointer is not possible in java
- It replaces destructor function with a finalize() function
- No header files in Java like c++

1.3.1. JAVA SOURCE FILE

Simple Java Program

```
class Example
{
public static void main(String args[])
{
System.out.prmtln("HeHo”);
}
}
```

Class : Example Declares a class[Every must be placed inside a class] Class:Keyword, Example:identifier (specifies the name).

Main line:

main()-Starting point for the interpreter to start the execution of the program.

public: access specifier (making it reachable to all the other classes)

static: main must be declared as static; because the main interpreter uses this function before any objects are created.
void-main :function does not return any value(but simply prints a few text to the screen). All the parameters to a function are declared inside a pair of parenthesis.
String args[] declares a argument named args, which hold Array of objects of the type string.

Output Line: println-function is a member of the out object,which is a static data field of system class.

- Println all the time appends a newline character to the close of the string.
- This means that any following output will start on a new line.
- Every java must close with a semi colon.

1.3.2. JAVA PROGRAM STRUCTURE

(Example program refer any program in unit 1 any 2)
The following table shows the structure of java program

Documentation Section
Package Statement
Import Statements
Interface Statement
Class Definition
Main Method Class
Main Method Definition

1.3.3. JAVA COMPILATION

The steps to execute java program

- preparation of the program text
- compilation of the program
- execution of the compiled program

1. *Preparation of the program text*
To write the program text we need to write a file containing code. For a Java program, The file name must be represented

***ClassName*.java**
where *ClassName*is the name of the class defined in the program. E.g.,Example.java
The program can be written by any program that allows one text file (editor). E.g., NotePad, Emacs, ...

2. Compilation of the program

The compilation of the program is essential to convert the program into a sequence of commands that can be openly executed by the computer. The standard Java compiler, which is part of the

Java Standard Development Kit (Java SDK), is javac. To make use of it, you have to execute the following command:

javac *ClassName*.java

The compilation yields a file called *ClassName*.class, which contains the command that can be straight executed by the system. For example:

javac Example.java

- creates the file Example.class.

3. Execution of the compiled program

After compilation step, we have to execute the java program.In Java the execution of a program is done in following manner

java *ClassName*

(without .class). For example, the command

Java Example

causes the execution of the code or programExample (or, more exactly, of the main method of the class First), and hence displays on the screen.