Java Assignment-1 (Set-1)

Roll No:- 19BQ1A05M

① Write about the role of JVM, JAVA API in developing the platform Independent Java program with suitable example.

Ans:-

→ The meaning of platform-Independent is that the Java compiled code (byte code) can run on all operating systems.

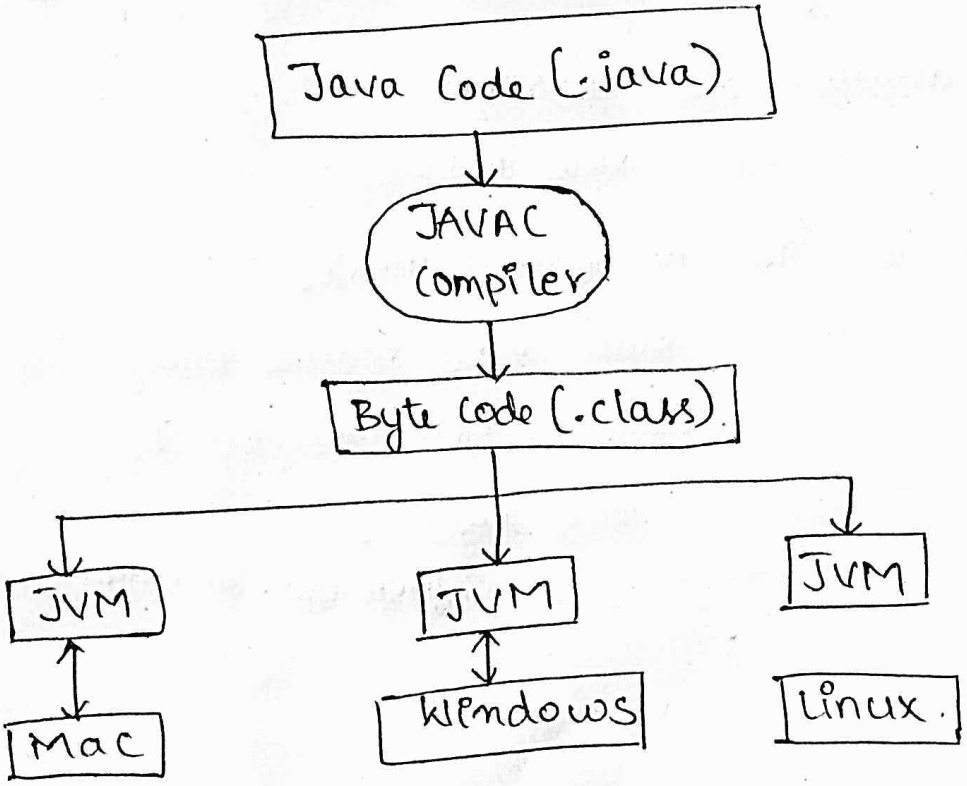
A program is written in a language that is a human-readable language. It may contain words, phrases, etc. which the machine doesnot understand. For the secure source code to be understood by the machine, it needs to be in a language understood by machines, typically a machile-level language. So here comes the role of a compiler. The compiler converst the high-level language into a format understood by the machines. Therefore, a compiler is a program that translates the source code for another program from a programming language into executable code.

This executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation in Java is the 'Java Byte Code',

Java Virtual Machine :-

→ JVM is a specification that provides runtime environment in which java bytecode can be executed. As the name implies, the JVM acts as a "Virtual" machine or process. Java's platform independence consists mostly of its 'Java Virtual Machine' (JVM). JVM makes this possible. because it is aware of the specific instruction lengths and other particularities of the platform. The JVM Performs — loads code
— Verifies code and Executes code.

*



→ This is how a java program executes on different platforms.

*Execution of Java program:-

- Whenever, a program is written in java, the javac compiles it. The result of Java compiler is the .class file or the byte code and not the machine native code (unlike C compiler). The byte code generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the "JVM" and thus Bytecode is executed by JVM. And finally program runs to give the desired output.

- In case of c or c++ (language that are not platform independent), the compiler generates an .exe file which is OS dependent. When we try to run this .exe file on another OS it doesn't run, since it is OS dependent and hence is not compatible with the other OS.

* Thus, JAVA is platform independent but Java Virtual Machine (JVM) is platform dependent.

→ In java, the main point here is that the JVM depends on the operating system - so if you are running Mac OS X you will have a different JVM than if you are running Windows or some other operating system. This fact can be verified by trying to download the JVM for your particular machine - when trying to download

it, you will be given a list of JVM's corresponding to different OS's, and you will obviously pick whichever JVM is targeted for the operating system that you are running. So we can conclude that JVM is platform-dependent and it is the reason why Java is able to become "Platform Independent".

Some important points:-

→ In the case of Java, it is the magic of Bytecode that makes it platform independent.

→ This adds to an important feature in the JAVA language termed as portability. Every system has its own JVM which gets installed automatically when the jdk software is installed. For every os, seperate JVM is available which is capable to read the .class file or byte code.

→ An important point to be noted is that while Java is platform-Independent language, the JVM is platform dependent. Different JVM is designed for different OS and byte code is able to run on different OS.

(b) → Java Application Programming Interface (JAVA API) is a list of classes that are part of the Java Development Kit (JDK). An application programming interface (API), in the context of Java, is a collection of prewritten packages, classes, and interfaces with their respective methods, fields and constructors. Similar to user interface, which facilitates interaction between human and computers, an API serves as a software program interface facilitating interaction.

② With an example program explain the concept of classes and nested classes in java.

Ans:

Java class:-

→ A class is a blueprint for the object. Before you create objects in Java, you need to define a class.

→ class is a blueprint or a set of instructions to build a specific type of object. It is a basic concept of Object Oriented Programming which revolve around the real-life entities. Class in Java determines how an object will behave and what the object will contain.

Syntax:-

```
class <class_name>{
    field;
    method;
}.
```

→ A class in Java contains:

- Fields
- Methods
- Constructors
- Blocks.
- Nested class and interface

The general form of a class:-

```
class  Class Name
{
        Instance variables ;
        method Name (parameter list)
        {
            // body.
        }
}.
```

The data or variables are defined within a class are called instance variables. The code is contained within methods. Collectively, the methods & variables are defined within a class are called members of the class. Variables defined within a class are called instance variables because such each instance of the class (that is each object of the class) contains its own copy of these variables.

Ex:-

```
Public class Dog {
        String breed, colour;
        int age;

        void brooking() {
        }
        void hungry() {
        }
        void sleeping {
        }
}
```

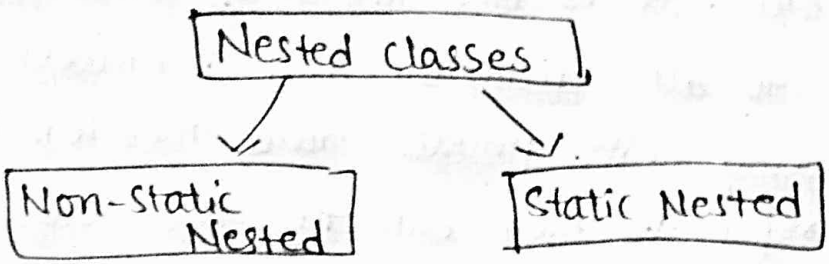
Java Nested class:-

* A class defined within another class is known as Nested class. The scope of the nested class is bounded by the scope of its enclosing class.

Syntax:-

```
class Outer{
    // class Outer members

    class Inner{
        // class Inner members.
    }
}
```



Advantages of Nested class:-

1. It is a way of logically grouping classes that are only used in one place.
2. It increases encapsulation.
3. It can lead to more readable and maintainable code.

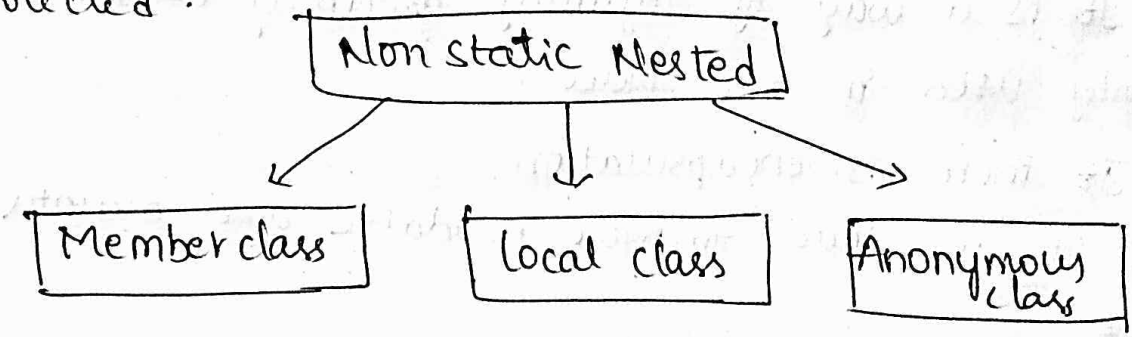
* If we want to create a class which is to be used only by enclosing class, then it is not necessary to create a seperate file for that. Instead, you can add it as "Inner class"

Static Nested Class:-

→ If the nested class i.e., the class defined within another class, has static modifier applied int it, then it is called as static nested class. Since it is, static nested classes can acess only static members of its outer class i.e., it cannot refer to non-static members of its enclosing class directly. Because of this restriction, static nested class is rarely used.

Non static Nested class:-

→ Non static nested class is the most important type of nested class. It is also known as Inner class. It has access to all variables and methods of outer class including its private data members & methods and may refer to them directly. But Outer class can't directly access the members of inner class. Outer class can have public acces & inner class has public, private & protected.



* Java Member Inner class:-

→ A non static class that created inside a class but outside a method is called member inner class.

Syntax:-

```
Class outer {
    //code
    Class inner {
    }
    //code
}
```

Ex:-
```
Class Test Member Outer1{
    private int data=30;
    class Inner {
        void msg(){ sg
            system.out.println("data is" + data);
        }
    }
    Public static void main (string args[]){
        Test Members outer1 obj = new Test MembersOuter1();
        TestMembersouter1.Inner in=obj.new Inner();
        in.msg();
    }
}
```

output :- data is 30.

* Java Local Inner class :-

→ A class i.e., created inside a method is called local inner class in java. If you want to invoke the methods of local inner class, you must instantiate this class inside the method.

Ex:-
```
Public class localInner {
    private int data =30;   //Instance.
    void display(){
        class Local {
            void msg(){
                system.out.println(data);
            }
        }
        Local l = new Local();
        l.msg();
    }
```

```
    Public static void main (String args[]){
        local Inner1 obj = new local Inner1();
        obj.display();
    }
}
```

output :- 30

* Java Anonymous inner class.

→ A class that have no name is known as anonymous inner class in java. It should be used if you have to override method of class or interface. Java Anonymous inner class can be created by two ways:

1. class (may be abstract) 2. Interface

* Java anonymous inner class example using class:

```
abstract class person{
    abstract void eat();
}
Class Test-AnonymousInner{
    public static void main(String args[]){
        person p= new Person(){
            Void eat(){
                System.out.println("nice fruit");
            }
        };
        P.eat();
    }
}
```

output:- nice fruit.

** Java anonymous Inner class example using Interface :-

```
interface Eatable(){
    void eat();
}
class TestAnonymousInner1{
    public static void main(String args[]){
        Eatable e = new Eatable(){
        public void eat(){
            System.out.println("nice fruits");
        }
        };
        e.eat();
    }
}
```

output :- nice fruits.

(3) Design a class Railway Ticket with the following description:

Instance variables / data members:

String name: to store the name of customer.

String Coach: to store the type of coach customer wants to travel.

long mobile: to store customer's mobile number.

Int amt: to store basic amount of ticket.

Int totalamt: to store the basic amount of ticket to be paid after updating the original amount.

Methods:

Void accept(): to take input for name, coach, mobile number and amount.

Void update() to update the amount as per the coach selected. Extra amount to be added in the amount as follows:

Type of coaches - Amount

First_AC 700, Second_AC 500, Third_AC - 250.

Sleeper None.

Void display(): To display all the details of a customer such as name, coach, total amt & mobile no.

Write a main() method to creat an object of the class and call the above methods.

Ans:-

```
import java.io.*;
import java.util.scanner;
class RailwayTicket {
        String name, coach;
        int amnt, totalamt;
        long mobile;

        Void accept() {
            Scanner sc = new Scanner(system.in);
            System.out.println("Enter passenger's name: ");
            name = sc.next();
            System.out.println("Enter mobile number: ");
            mobile = sc.nextInt(); (or) sc.nextLong();
            System.out.println("enter coach(FirstAC/secondAC/ThirdAC/
                                                    Sleeper "));
            coach = sc.next();
            System.out.println("Enter basic amount of ticket:");
            amt = sc.nextInt();
        }
        Void update() {
            if(coach.equals("First AC"))
                    totalamt = amt + 700;
            else if(coach.equals("Second AC"))
                    totalamt = amt + 300;
            else if(coach.equals("Third AC"))
                    totalamt = amt + 250;
            else
                totalamt = amt;
        }
```

```
void display(){
    System.out.println("Name :"+ name);
    System.out.println("Mobilenumber:"+mobile);
    System.out.println("Coach :"+ coach);
    System.out.println("Totalamount :"+totalamt);
}
Public static void main(String args[]){
    RailwayTicket rt= new Railway Ticket();
    rt.update()
    rt.accept();
    rt.update();
    rt.display();
}
}
```

Input :- Enter passenger's name :
Chandrika
Enter mobile numer :
1234567890
Enter coach [First AC / Second AC / Third AC / Sleeper].
Second AC.
Enter basic amount of ticket
200.

Output :-
Name : Chandrika
Mobilenumber : 1234567890
Coach : Second AC.
Total amount : 700.

(4) Design a class to overload a function volume() as follows:

(i) double volume(double r) - with radius 'r' as an argument, returns the volume of sphere using the formula: $V = 4/3 \times 22/7 \times r^3$

(ii) double volume(double h, double r) - with height 'h' and radius 'r' as the arguments, return the volume of cylinder, using the formula:

$V = 22/7 \times r^2 \times h$.

(iii) double volume(double l, double b, double h) - with length 'l', breadth 'b' and height 'h' as the arguments, returns the volume of a cuboid using the formula: $V = l \times b \times h$.

Ans:

```
class Dimension
{
    double volume(double r)
    {
        double V = (4/3)*(22/7)*(r*r*r);
        return V;
    }
    double volume(double h, double r)
    {
        double V = (22/7)*(r*r)*h;
        return V;
    }
```

```
    double volume(double l, double b, double h)
    {
        double V= l* b*h;
        return V;
    }
Public static void main (String args[])
{
    Dimension obj = new Dimension();
    double x=obj.volume(6.1);
    double y = obj.volume(10.05, 22.2);
    double z = obj.volume(7.3, 6.2, 12.3);
    System.out.println("Volume of sphere is"+x);
    System.out.println("Volume of cylinder is"+y);
    System.out.println("Volume of cuboid is"+z);
}
}
```

Output:-

Volume of sphere is 680.9429999999.

Volume of cylinder is 1489.1259999998.

Vdume of cuboid is 556.698.