

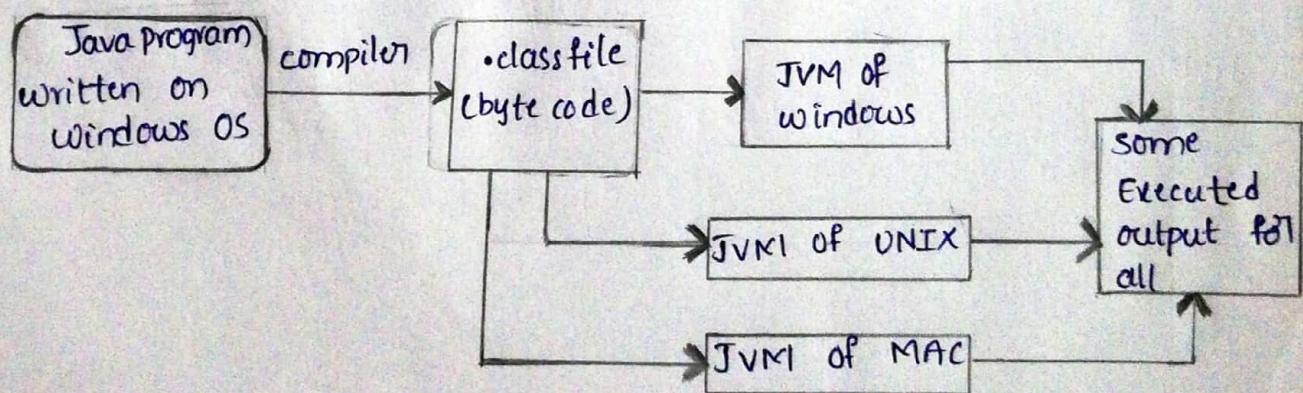
1. Write the role of JVM, JAVA API in developing the platform independent java program with suitable example.

### Role of JVM in platform independence

Java code is compiled and converted into **BYTE CODE**. The job of JVM is to take this bytecode and turn it into code that is understandable by the underlying platform. The requirements for this would be different for different platforms. So, there should be different JVMs for different platforms.

In short, you take JVM out of picture, there is no entity that would recognize the byte code. Then it won't be platform independent. JVM makes it "**Platform Independent**" by handling all those things that would have otherwise made it "**platform dependent**". The most important being, converting byte code to platform understandable code.

The JVM fetches classes from a disk or from the network, and then verifies that the byte codes are safe to be executed. For every operating system separate JVM is available which is capable of to read the .class file or a byte code.



Hence, JAVA is called platform independent language.

## Role of JAVA API

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. In java, most basic programming tasks are performed by the API's classes and packages, which are helpful in minimizing the number of lines written within the pieces of code.

The java API, included with the JDK, describes the function of each of its components. In java programming many of these components are pre-created and commonly used. Thus the programmer is able to apply prewritten code via Java API. After referring to the available API classes and packages, the programmer easily invokes the necessary code classes and packages for implementation.

For example: when you use the application on your mobile phone, the application connects to the internet and sends data to a server. That's where the waiter or API comes in. The waiter is the messenger - an API - that takes your request or order and tells the kitchen - the system - what to do. Then the waiter delivers the response back to you.

2. With an example explain the concept of classes and Nested classes in Java programming.

### Classes in Java :-

A class is a blueprint from which individual objects are created. A class contain any of the following variable types:

- 1, Local variables
- 2, Instance variables
- 3, Class variables

Ex:-

```
public class Dog {
    String breed;
    int age;
    String color;
    void barking() {
    }
    void hungry() {
    }
    void sleeping() {
    }
}
```

A class can have any no. of methods to access the value of various methods.

Here, barking(), hungry(), sleeping() are called methods.

Local Variables :- Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized with in the method and the variable will be destroyed when the method has completed.

Instance Variables :- These are the variables with in a class but outside any method. These are initialised when the class is instantiated. These can be accessed from inside any method, constructor or blocks of that particular class.

Class Variables :- These are declared within a class, outside any method, with the static keyword.

Some of the important topics that are discussed in classes are :-

1. constructors :-

Every class has a constructor. If we don't explicitly write a constructor for a class, the Java compiler builds a default constructor for that class. The main rule is that a constructor should have same name as the class. A class can have more than one constructor.

```
public class Puppy() {  
    public Puppy() {  
        // This constructor has no parameters.  
    }  
    public Puppy(String name) {  
        // This constructor has one parameter called name.  
    }  
}
```

2. creating an object :- An object is created from a class. In Java ~~new~~ the new keyword is used to create new objects.

\* Declaration - A variable declaration with a variable name with object type.

\* Instantiation - The 'new' keyword is used to create the object.

\* Initialisation - The 'new' keyword followed by a call to a constructor. This call initializes the new object.

Ex:- Puppy myPuppy = new Puppy("tommy");

It creates an object called myPuppy.

3. Accessing instance variables and methods :-

These are accessed by creating objects.

First create an object

Object Reference = new Constructor();

Now call a variable

Object Reference. variableName;

Now call a class method.

Object Reference. Methodname();

Example :-

```

Public class Demo {
    public static void main(String args[]) {
        Test t = new Test(12, 11.45f);
        t.display();
    }
}

class Test {
    int x;
    float y;
    Public Test(int a, float b) {
        x=a;
        y=b;
    }
    Public void display() {
        System.out.println("The values of x,y are "+x+", "+y);
    }
}

```

Nested classes in java :-

In java, it is possible to define a class within another class, such classes are known as nested classes. They enable you to logically group classes that are only used in one place, thus this increases the use of encapsulation, and creates more readable and maintainable code.

The scope of nested class is bounded by the scope of its enclosing class. A nested class has access to the members, including private members of the class in which it is nested. But the enclosing class does not have access to members of nested class. It is also a member of its enclosing class.

Nested classes are divided into two categories:

1. Static nested class:- Nested classes that are declared static are known as static nested classes.

2. Inner class:- An inner class is a non-static nested class.

Syntax:-

```
class Outerclass
```

```
{
```

```
    ....
```

```
    class Nestedclass
```

```
{
```

```
    ....
```

```
    ....
```

```
3
```

```
4
```

### Static nested classes

In case of static nested class, without an outerclass object existing, there may be a static nested class object. i.e., an object of a static nested class is not strongly associated with the outerclass object. As with class methods and variables, a static nested class is associated with its outer class. And like static class methods, a static nested class cannot be referred directly to instance variables or methods defined in its enclosing class. It can use them only through an object reference. They are accessed using the enclosing name.

To create an object for static nested class use the syntax:

```
Outerclass.StaticNestedClass nestedObject
```

```
new Outerclass.StaticNestedClass();
```

Java program to demonstrate accessing a static nested class

```

class Outerclass{
    static int outer_x = 10;
    int outer_y = 20;
    private static int outer_private = 30;
    static class StaticNestedClass {
        void display() {
            System.out.println("outer_x = " + outer_x);
            System.out.println("outer_private = " + outer_private);
        }
    }
}

public class StaticNestedClassDemo {
    public static void main(String args[]) {
        Outerclass.StaticNestedClass nestedObject = new Outerclass.
            StaticNestedClass();
        nestedObject.display();
    }
}

```

↓  
accessing a static  
nested class.

In the above program if you mention the statement like

System.out.println("outer\_y = " + outer\_y);

then it will show compilation error because static nested class cannot directly access non-static members.

Output:-

outer\_x=10  
outer\_private=30

Inner classes

In case of innerclass, without an outerclass object existing, there cannot be an innerclass object i.e., an object of the innerclass is always strongly associated with an outer class object. To instantiate an innerclass, you must first instantiate outer class and then, create the inner object within the outer object.

Syntax:-

```
Outerclass.Innerclass innerObject = outerObject.new Innerclass();
```

Example:-

```
class OuterClass {
    static int outer-X=10;
    int outer-Y=20;
    private int outer-private=3;
    class InnerClass {
        void display() {
            System.out.println("outer-X = " + outer-X);
            System.out.println("outer-Y = " + outer-Y);
            System.out.println("outer-private = " + outer-private);
        }
    }
}

public class InnerclassDemo {
    public static void main(String args[]) {
        OuterClass.outerObject = new OuterClass();
        OuterClass.InnerClass innerObject = outerObject.new InnerClass();
        innerObject.display();
    }
}
```

Output:-

outer-X = 10

outer-Y = 20

outer-private = 30

3. Design a class Railway Ticket with the following description.

Instance variables / data members:

String name: to store name of customer

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

long mobno: to store customer's mobile number

int amt: to store basic amount of ticket

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

Methods:

void accept(): to take input for name, coach, mobilenumber, 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 details of a customer such as name, coach, mobilenumber and total amount.

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

code:-

```
import java.io.*;
import java.util.*;

class RailwayTicket {
    String name, coach;
    long mobno;
    int amt, totalamt;

    public void accept() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name");
        name = sc.next();
        System.out.println("Enter mobile number");
        mobno = sc.nextLong();
        System.out.println("Enter coach type");
        coach = sc.next();
        System.out.println("Enter basic basic amount of ticket");
        amt = sc.nextInt();
    }

    public void update() {
        if (coach.equals("First-Ac"))
            totalamt = amt + 700;
        else if (coach.equals("Second-Ac"))
            totalamt = amt + 500;
        else if (coach.equals("Third-Ac"))
            totalamt = amt + 250;
        else
            totalamt = amt;
    }

    public void display() {
        System.out.println("name: " + name);
        System.out.println("coach: " + coach);
        System.out.println("Mobile number: " + mobno);
        System.out.println("total amount: " + totalamt);
    }
}
```

```

public static void main (String args[])
{
    Railway Ticket r = new RailwayTicket()
    r.accept();
    r.update();
    r.display();
}

```

3

Output:-

Enter name

raju

Enter mobile number

9652321864

Enter coach type

Third - AC

Enter basic amount of ticket

62

name : raju

coach : Third - AC

Mobile number : 9652321864

total amount : 312

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

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

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

$$V = \frac{22}{7} \times r^2 \times h$$

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 :

$$V = l \times b \times h$$

Code:-~~class Overloading~~)

class Overloading {

double volume (double r) {

double v = (4.0/3)\*(22.0/7)\*r\*r\*r;

return v;

}

double volume (double h, double r) {

double v = (22.0/7)\*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[]) {

Overloading obj = new Overloading();

double x = obj.volume(6.1);

double y = obj.volume(10.05, 22.2);

double z = obj.volume(6.1, 2.5, 4.5);

System.out.println("volume of sphere: " + x);

System.out.println("volume of cylinder: " + y);

System.out.println("volume of cuboid: " + z);

}

y

Output:-

Volume of sphere: 951.1584761904762

Volume of cylinder: 15566.70342857143

Volume of cuboid: 144.875

For reference I have used some online resources like  
Geeks for Geeks , techopedia