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 byte code and turn it into code that is understandable by the underlying platform. The requisities 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 operalines system seperate JVM is available which is capable of to read the class file or a byte code

19BQ1A05H5

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 vice Java API. After refering 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. thats where the waiter or API comes in. The waiter is the messenger - or API - that takes your request or order and tells the kitchen - the system - what to do - then the waiter deliver the response back to you.

19BQ1A05H5

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

Classes in Java:-

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

1, Local variable
2, Instance variable
3, class variable

Ex:-
```
Public class Dog {
    String breed;
    int age;
    String color;
    void barking() {
    }
    void hungry() {
    }
    void sleeping() {
    }
}
```

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

Instance variables: these are 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 explictly 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 some name as the class. A class can have more than one constructor.

```
Public class puppy () {
        Public puppy () {
        }
        Public puppy (string name) {
           // this constructor has one parameter called name
        }
}
```

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

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 is called initialize the new object.

Ex: Puppy my puppy = new puppy ("tommy");

It creates an object called my puppy

3. Accessing instance variables and methods:-

These are accessed by creating objects

- first creat an object

object Reference = new constructor();

Now call a variable

object Reference. variable Name;

Now call a class method.

Object Reference. Method name();

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){
     system out. println (" The values of x, y are" + x + ", "+ Y);
      }
}
```

Nested classes in java :-

In java, it is possible to define a class with in 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 data.

19BQ1A05H5

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 outer class
{
......
    class Nested class
    {
        ......
        ......
    }
}
```

Static nested classes

In case of static nested class, with out 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 refered 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:

```
Outer class. static Nested class      nested Object =
    new Outer class. static nested class();
```

19B81A05H5

Page no: 7

Java program to demonstrate accessing a static nested class

```
class outer class {
        static int outer_x = 10;
        int outer y = 20;
        Private static int outer_private = 30;
        static class Static Nested class {
                void display( ) {
                system. out. Print ln ("outer_x = " + order_x);
                system. out. Print ln ("outer_private = "+outer_private);
                }
        }
}
Public class static Nested Demo{
        Public static void main( string args[]){
                outer class. Static Nested class nested object = new
                                              outer class.
                                              Static Nested class();
                nested object. display();
                                              accessing a static
            }                                 nested class
    }
```

In the above program if you mention the statement like

system. out. Print ln("outer_y = " + outer_y);

then it will show compilation error because static nested class can not directly access non-static members

<u>out Put:-</u>

```
outer_x = 10
outer_private = 30
```

Inner classes

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

Syntax: -

```
outerclass. Innerclass innerobject = outer object. new Innerclass();
```

Example:

```
class Outer class {
        static int outer_x=10;
        int outer_y=20;
        Private int outer_Private =3;
        class Inner class{
                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  Inner class Demo{
        Public static void main (string args[]){
            outer class. Outer object = new object class();
            outer class. Inner object= Outer object. new Inner class();
            inner object . display();
        }
}
```

Out put:

```
outer_x =10
outer_y =20
outer_Private =30
```

③ Code:-

```
import java.io.*;
import java.util.*;
class Railway Ticket {
      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. next long();
           system. out. println("Enter coach type");
           coach = sc.next();
           system.out. println ("Enter basic amount of ticket");
           amt = sc. next Int ();
   }
   Public  void  update() {
        if (coach. equals ("first_AC"))
                 totalant = amt + 700;
        else if (coach. equals ("second_AC"))
                 totalant = amt+ 500;
        else if (coach. equals ("Third-AC"))
                 totalant = amt +250;
        else
           totalant= 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:" + totalent);
  }
```

19BQ1A05H5
Page no: 10

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

Output:

```
Enter name
Pavan
Enter mobile number
9913579621
Enter coach type
Third-AC
Enter basic Amount of ticket
62
name: raju
coach: Third-AC
Mobile number: 9913579621
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: $V = 4/3 \times 22/7 \times r^3$

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

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

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:

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

19BQ1A05H5
Page no: (11)

Code:-

```
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;
        }
        Public static void main(string args[]){
                Overloading  obj = new Overloading();
                double r = obj. volume (6.1);
                double y = obj. volume (10.05, 22.2);
                double z = obj. volume( 8.1, 2.5, 9.5);
                system. out. Println("volume of sphere: "+r);
                system. out. Println("volume of cylinder:"+y);
                system. out. Println("volume of cuboid: "+z);

        }
}
```

Output:-

```
Volume of sphere: 951.1584761904762
volume of cylinder: 15566.7034285 7143
volume of cuboid: 144.875
```