Java Assignment - 1

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

A) The Full form of JVM is JAVA Virtual Machine. The JVM manages system memory and provides a portable execution environment for java-based applications. The JVM has two primary functions:

1) Java programs to run on any device or operating system (known as "write once, run anywhere")

2. To manage and optimize program memory.

JVM definition:- The JVM is the specification for a software program that executes code and provide runtime environment for that code.

4) The JVM manages memory through a process called garbage collection, which continously identifies and eliminates unused memory in java programs.

Byte code is a highly optimized set of instructions designed to be execute by the java runtime system, which is called JVM.

Byte code in java is the reason java is platform-independent, as soon as a java program is compiled bytecode is generated.

The Bytecode works in this manner

Source code

Compiler

Byte code

JVM

Machine code

JVM is the only one entity that would recognize byte code in Java. The JVM is different for different platforms but implementation of JVM requires very less cost.
Finally the role of JVM is converting byte code to platform understable code.

→ JAVA API is a list of all classes that are part of the Java development kit (JDK). API full form is Application programming interface. It includes all Java packages, classes, and interfaces, along with their methods, fields and constructors. These prewritten classes provide a tremendous amount of functionality to a programmer.
A programmer should be aware of these classes and should know how to use them.

If we browse through the list of packages orin the API, we will observe that there are packages written for GUI programming, network programming managing input and output, database programming and many more.
In order to use a class from Java API one needs to include an import statement at the start of the program.
For example in order to use Scanner class, which allows a program which allows a program to accept input from the keyboard one must include the following import statement.

```
import java.util.Scanner;
```

The above import statement allows the programmer to use any method listed in the Scanner class.

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

This version of the import statement imports all the classes in the API's java.util packages and make them available to the programmer

From internet
Website - wikipedia

2. With an example program explain the concept of classes and nested class classes in Java?

Java is an object oriented programming language. Everything in java is associated with classes and objects, along with its attributes and methods. For example : in real life, a car is an object. the car has attributes, such as weight and color, and methods, such as drive and brake.

To create a class we have to use the keyword class

```
public class Myclass {
    int x = 5;
}
```

Create an object :- To create an object of Myclass, specify the class followed by the object name, and the keyword new

```
public class Myclass {
    int x = 5;

    public static void main(String[] args) {
        Myclass myobj = new Myclass();
        System.out.println(myobj.x);
    }
}
```

Declaring objects (Also called instantiating a class).

when an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behaviour of the class. An object consists of

State :- It is represented by attributes of an object. It also reflects the properties of an object

Behaviour :- It is represented by methods of an object. It also reflects the response of an object with other objects

Identity :- It gives a unique name to an object and enables one object to interact with other objects

```
public class Dog
{
  String name;
  String breed;
  int age;
  String color;
  //constructor
  public Dog ( string name , string breed, int age, string color)
  {
      this . name = name;
      this . breed = breed;
      this . age = age;
      this . Color = color;

  }
  public string getName()
  {  return name;
  }
  public string getBreed()
  {  return breed;
  }
  public string getcolor()
  {  return color;
  }
  @override
  public string tostring()
  {
    return (" Hi my name is " + this.getName()+ " In My breed, age,
      color are " + this.getBreed()+ "," + this.getAge() + "+" +
      this.getcolor());
  }
  public static void main (string[] args)
  {
     Dog tuffy = new Dog ("tuffy", "papillon", 5, "white");
     System.out.println ( tuffy. tostring());
  }
}
```

Nested classes :-

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. Thus the above example, class Nested class does not exist independently of class outerclass.

→ A nested class has access to the members, including private members, of the class in which it is nested. However, the reverse is not true i.e the enclosing class does not have access to the members of the nested class

→ A nested class is also a member of its enclosing class

→ As a member of its enclosing class, a nested class can be declared private, public, protected or package. private (default):

Nested classes are divided into two Categories

Static nested class :- Nested classes that are declared static are called static nested classes.

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

Advantages of Nested classes :-

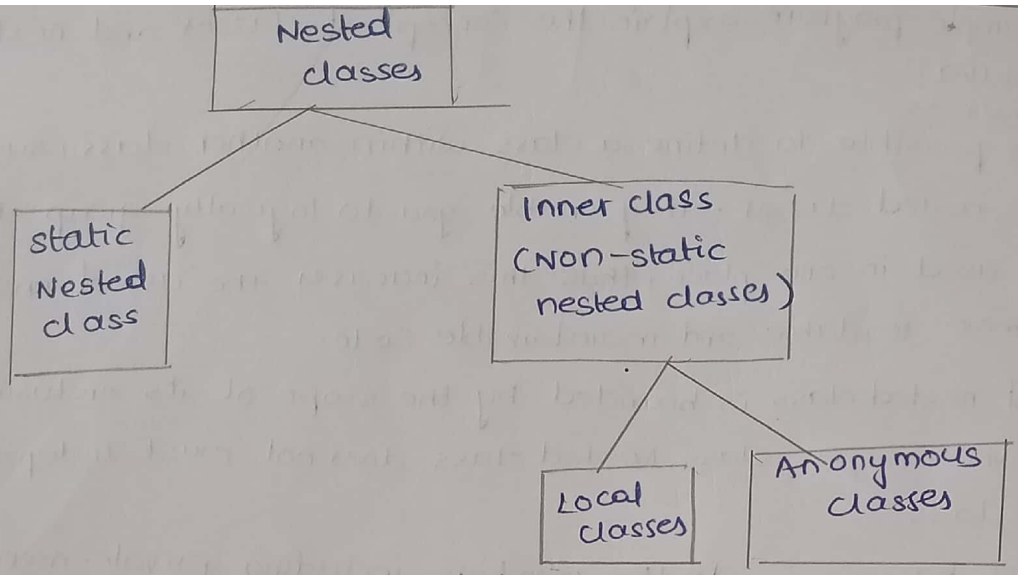
1. A Nested classes will help to create Nested classes as private

The inner class can be declared as private

The inner class can call the methods of the outer class

Syntax :-

```
class outerclass
{
  class NestedClass
  {
  }
}
```



Static and nested classes :- In the case of normal or regular inner classes, without an outer class object existing, there cannot be an inner class object i.e an object of the inner class is always strongly associated with an outer class object. But in the case of static of static nested class, without an outer class object exisiting there may be a static nested class object i-e, an object of a static nested class object is not strongly associated with the outer class 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 refer 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 class name.

For example, to create an object for the static nested class, use this syntax.

```
OuterClass.StaticNestedClass nestedObject = new OuterClass.
                                    StaticNestedClass();
```

```
// outer class
class Outerclass
{
    // static member
    static int outer_x = 10;

    // instance (non-static) member
    int outer_y = 20;

    // private member;
    private static int outer_private = 30;

    // static nested class
    static class StaticNestedclass
    {
      void display()
      {
        // can access static member of outer class
        system.out.println("outer_x " + outer_x);

        // can access display private static member of outer class
        system.out.println("outer_private = " + outer_private);

        // The following statement will give compilation error
        // as static nested class cannot directly access non-static members
        // system.out.println("outer_y = " + outer_y);

      }
    }
}

// Driver class
public class staticNestedClassDemo {
        public static void main(String[] args){
                // accessing a static nested class
                Outerclass.staticNestedclass nestedObject = new Outerclass.
                                                        staticNestedclass
        }               nestedObject.display();
}
```

o/p

Outer_x = 10

Outer-private = 30

Inner classes :- To instantiate an inner class, you must first instantiate the outer class. Then, create the inner object with in the outer object

Syntax :

```
OuterClass.InnerClass innerObject = outerObject.new InnerClass();
```

// Java program to demonstrate accessing a inner class

```
// outer class
class OuterClass
{
    static int outer_x = 10;
    int outer_y = 20;
    private int outer_private = 30;
    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();
    }
}
```

outer_x = 10

oute_y = 20

outer_private= 30

There are two special kinds of inner classes

1. Local inner classes
2. Anonymous inner classes

Local inner class :- Local Inner classes are the inner classes that are defined inside a block. Generally, this block is a method body. sometimes this block can be a for loop, or an if clause. Local Inner classes are not a member of any enclosing classes. They belong to the block they are defined within, due of which local inner classes cannot have any access modifiers associated with them. However, they can be marked as final or abstract. These class have access to the fields of the class enclosing it. Local inner class must be instantiated in the block they are defined in.

Rules of Local Inner class :-

1. The scope of local inner class is restricted to the block they are defined in.
2. Local inner class cannot be instantiated from outside the block where it is created in
3. A local class has access to the members of its enclosing class
4. Local inner class can extend an abstract class or can also implement an interface

```
Public class Outer
{
  private void getvalue()
  {
    int sum = 20;
    class Inner
    {
      public int divisor;
      public int remainder;
      public Inner()
      {
        divisor = 4;
        remainder = sum % divisor;
      }
      private int getDivisor()
      {
        return divisor;
      }
      private int getRemainder()
      {
        return sum % divisor;
      }
      private int getQuotient()
      {
        system.out.println(" Inside inner class");
        return sum/ divisor;
      }
    }
    Inner inner = new Inner();
    System.out.println("Divisor = " + inner.getDivisor());
    System.out.println("Remainder =" + inner.getRemainder());
    system.out.println(" Quotient = " + inner.getQuotient());
  }
  public static void main (string [] args)
  {
    Outer outer = new outer();
    outer.getValue();
  }
```

op Divisor = 4 Quotient = 5
Remainder = 0
Inside inner class

Anonymous inner class are mainly useful in writing implentation classes for listener interfaces in graphics programming

Anonymous inner class are mainly created in two ways :

→ class
→ Interface

Syntax :-

```
Test t = new Test()
{
  // data members and methods ()
  public void test_method()
  {
    ___
    ___
    ___
  }
}
```

```
interface Age
{
  int x = 21;
  void getAge();
}
class AnonymousDemo
{
  public static void main(String[] args)
  {
    Myclass obj = new Myclass();
    obj.getAge();
  }
}
class Myclass implements Age
{
  @override
  public void getAge
  {
    System.out.print("Age is "+x);
  }
}
```

From internet
website - geeks for geeks

3. Design a class RailwayTicket 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 mobno : to store coustomer's mobile no.

int amt : to store the amount of ticket

int totalamt : to store the amount to be paid after updating original

Methods

Void accept() - to take input for name, coach, mobile no and amount

Void update() - to update the amount as per the coach selected.

Type of Coaches Amount

First_AC 700 , Second_AC 500 , Third_AC 250 , Sleeper None.

Void display() - TO display all methods such as name, coach, total amount and mobile number

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

```
import java.io.*;
import java.util.Scanner;
class RailwayTicket {
    private String name;
    private String coach;
    private long mobno;
    private int amt;
    private int totalamt;
    public accept RailwayTicket() {
            name = "Vanaja";
            Coach = "First_AC";
            mobno = 9866091555;
            amt = 800;
    }
    public RailwayTicket(string name, string coach, long mobno, int amt, int totalamt)
    {
            this.name = name;
            this.coach = coach;
            this.mobno = mobno;
            this.amt = amt;
            this.totalamt = totalamt;
    }
    // setters
    public void setName(string name){
            this.name = name;
    }
    public void setcoach(string coach) {
            this.coach = coach;
    }
    public void setMobno(long mobno) {
            this.mobno = mobno;
    }
    public void setAmt(int amt) {
            this.amt = amt;
```

```
public void setTotAmt(int totalamt) {
        this.totalamt = totalamt;
}
// getters
public String getName() {
        return name;
}
public String getCoach() {
        return coach;
}
public long getMobno() {
        return mobno;
}
public int getAmt() {
        return amt;
}
public int getTotAmt() {
        return totalamt;
}
public void dispTotal() {
    if (coach.equals("First_AC"))
        totalamt = amt + 700;
    else if (coach.equals("Second_AC"))
        totalamt = amt + 500;
    totalamt = amt + 250;
    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(" Total Amount : " + totalamt);
        System.out.println(" Mobile No : " + name);
    }
    public static void main (string args[])
    {
        RailwayTicket t = new RailwayTicket();
        t.accept();
        t.update();
        t.display();
    }
}
```

I wrote this program on my own.

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 formulae.

$V = 4/3 \times 22 \times 7/3$

ii) double volume(double h, double r) - with height 'h' and radius 'r' as the arguments, returns the $V = 22/7 \times r^2 \times h$

iii) double volume(double l, double b, double h) - with length 'l', breadth 'b' and height 'h' as the argument returns the cuboid formula $V = l \times b \times h$

```
import java.io.*;
import java.util.*;
class overloaded
{
public double volume (double R)
  {
    double vol = 4/3 * 22/7 * R * R * R;
    return vol;
  }
  double volume (double H, double R)
  {
    double vol = 22/7 * R * R * H;
  }
  double volume ( double L, double B, double H)
  {
    double vol = L * B * H;
  }
  public static void main (string args[])
  {
    Scanner sc = new Scanner (System.in);
    Overlooded ob = new Overlooded();
    System.out.println(" Volume of sphere = " + ob.volume(6.4));
    system.out.println(" Volume of cylinder = " + ob.volume(8.2, 3.9));
    System.out.println(" Volume of cuboid = " + ob.volume(5.3, 4.5, 8.6));
  }
}
```

From internet
sarthaks.com