Regd No: 19BQ1A05H1

Java Programming

Assignment

P. Hima Chandana

19BQ1A05H1

II-(CSE-C)

Submission:-

To: Naga Sri Harsha Sir

Date: 20-09-2020

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

J.V.M: (Java Virtual Machine)

*A JVM is a virtual machine that enables a computer to run java programs as well as programs written in other languages that are also compiled to Java bytecode.

* JVM is the main component of Java architecture & it is the part of Java Runtime Environment.

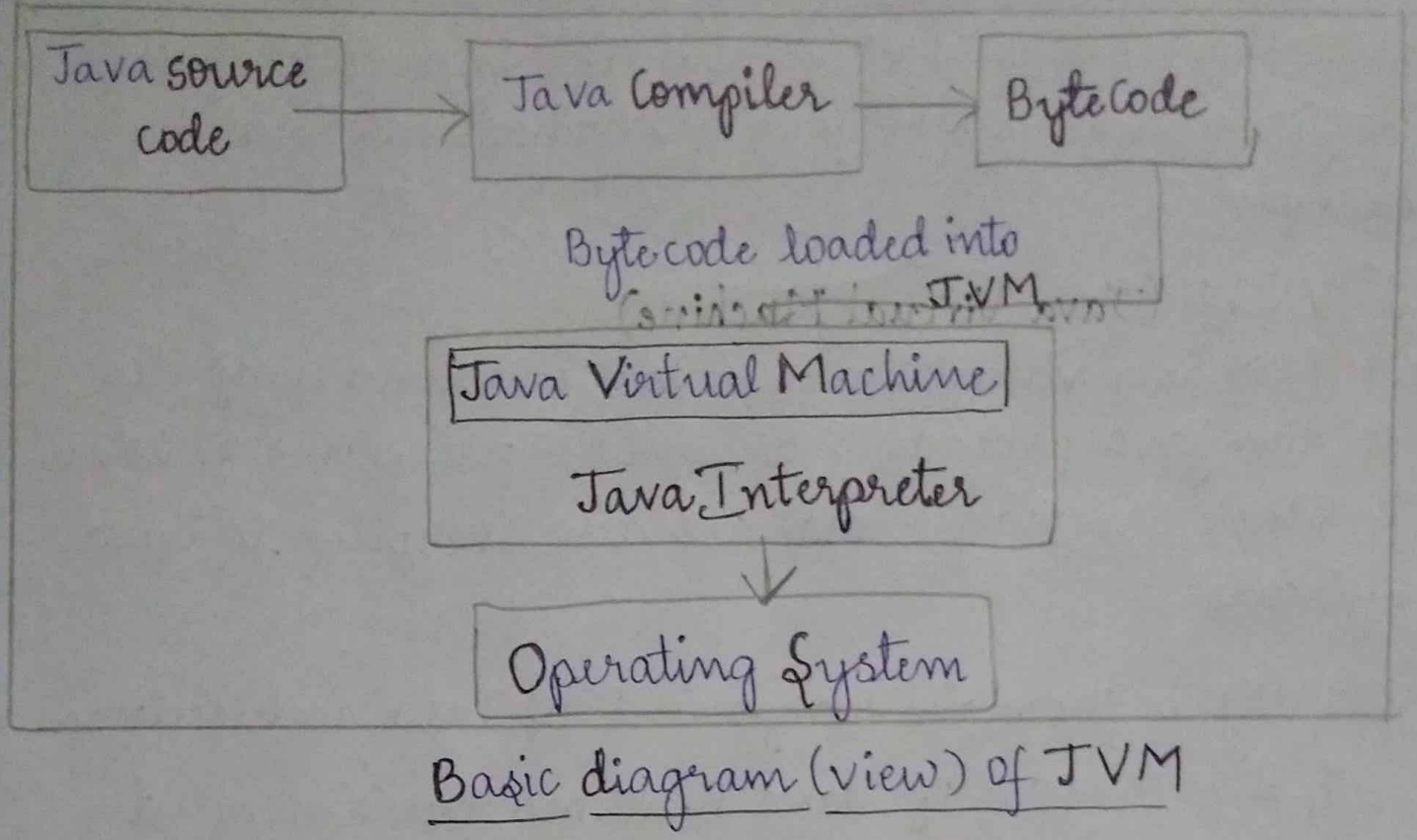
* A program of JVM is written in C Programming Language, and JVM is Operating System dependent

* JVM is responsible for allocating necessary memory needed by the Java Program.

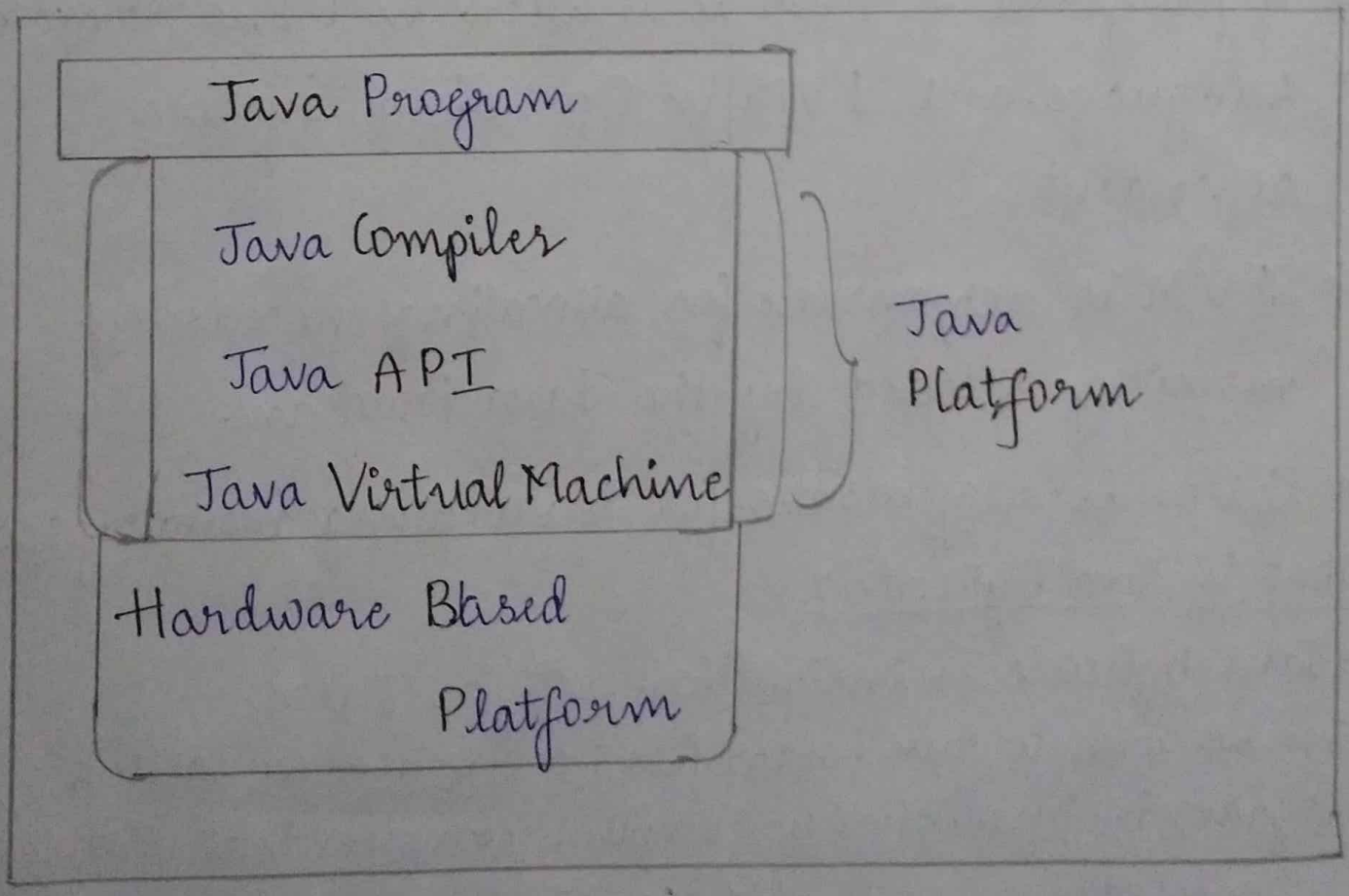
* JVM is responsible for deallocating memory space

What is Java ByteCode?

→ Java Bytecode is instruction set for JVM.
When we wish to run .class file, i.e., when we write a program in java, firstly compiler compiles that program & a bytecode is generated for that code. After first compilation, bytecode is now generated run by JVM & not processor in consideration.



Basic diagram (view) of JVM

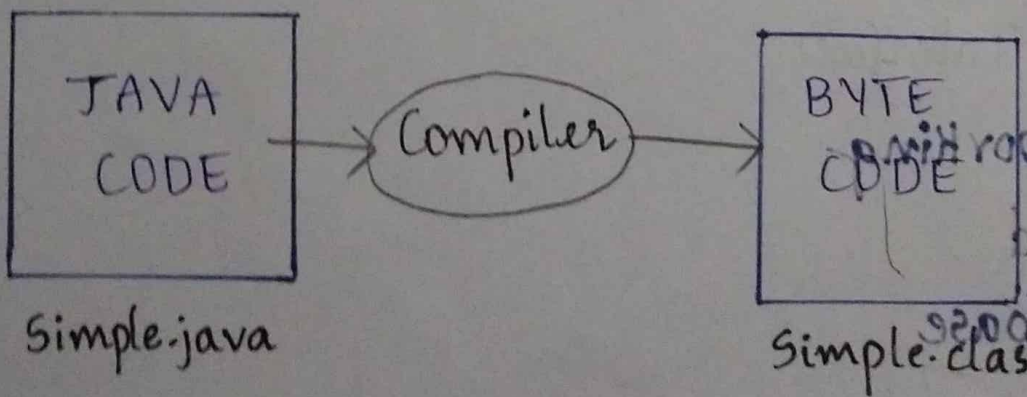


It can be better understood by seeing execution steps:

Java program execution mainly consists 5 steps:-

1. Edit :- Here, the programmer uses a editor/notepad application to write & give it ".java" extension.
2. Compile :- The programmer gives javac command & .java files are converted into bytecode that is language understood by JVM. In case of errors, compile time errors are raised here.
3. Load :- The program is loaded into memory. This is done by class loader which takes .class files containing bytecode ; stores it in the memory.
4. Verify :- The bytecode verifier checks if bytecode loaded are valid & do not neglect java security restrictions.
5. Execute :- The JVM interprets the program one bytecode at a time and runs program.

Eg: Consider a 'simple' java program

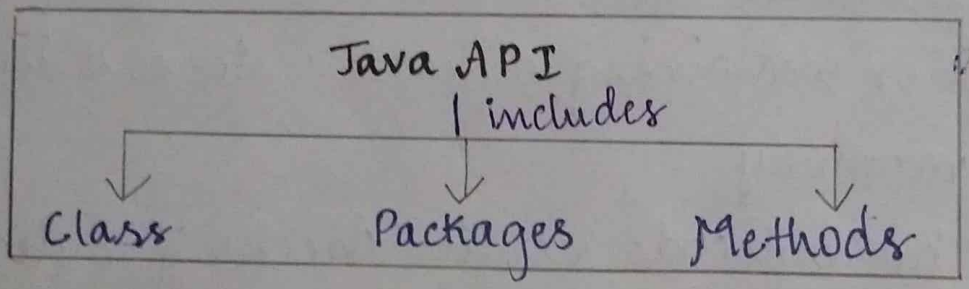


JAVA API:

→ An API Java Application Programming Interface (API) is the area of Java development Kit (JDK)

→ An API includes classes, interfaces, packages and also methods, fields, constructors...

→ A user interface offers basic user interaction among user & computer, in same manner, API works as an application program interface which gives connection among software as well as the customer.



* Java API is a vital element of JDK & identifies feauters of every element. Although Programming in Java, component is already produced and done it. Programmers can use API for:

* Graphics
* User Interface
* Networking
* Sound
* Database

Program that demonstrates encapsulations:

```
//A Account class which is a fully encapsulated class.
//It has a private datamember and getter, setter
  methods.
class Account {
    private long acc_no;
    private String name,
    private String email;
    private float amount;
    //setter and getter methods
    public long getAcc_no{
        return acc_no;
    }
    public void setACC_no (long acc_no)
    {
        this.acc_no = acc_no;
    }
    public String getName(){
        return name;
    }
    public void setName(String name)
    {
```

```
        this.name = name;
        }
        public String getEmail()
        {
            return email;
        }
        public void setEmail(String email)
        {
            this.email = email;
        }
        public float getAmount()
        {
            return amount;
        }
        public void setAMount(float amount)
        {
            this.amount = amount;
        }
}
```

2) Write an example program explain the concept of classes & nested classes in java.

A class is a template, blue-print, or contract that defines what an objects data fields and methods will be.

The General Form of a class:-

A class is declared by use of class keyword. Classes -General form:

```
class classname {
    type instance-variable 1;
    type instance-variable 2;
    //...,
    type instance-variable N;
    type methodname 1 (parameters) {
    //body
    }
    type methodname 2 (parameter list) {
    // body of method
    }
}
```

The data/ variables defined within a class are 'instance variables'. The code is contained within methods. Collectively, the methods and variables defined in a class are members of the class.

Variables defined within a class are instance variables because each instance of the class contains

its own copy of these variables.

→ All methods have same general form - main(), However most methods not specified as static/public.

→ Java do not need to have a main() method. You only specify one if that class is starting point for your program

A Sample Class (Example):

Here is a class Box that defines three instance variables width, height & depth.

```
class Box {
    double width;
    double height;
    double depth;
}
```

It is important to remember i.e., a class declaration only creates a template; it doesn't create actual object. Thus, the preceeding code does not cause any objects of type Box to come into existence.

```
Box mybox = new Box();
```

After this statement executes, mybox will be an instance of Box. To access instance variables we use dot(.) operator. The dot operator links name of object with name of instance variable.

Eg: mybox.width = 100;

This statement tells compiler to assign copy of width that is contained within mybox object the value of 100.
In general, the dot operator to access both instance variables & methods within an object.
→ Although commonly referred to as dot operator, formal specification for java categorizes the . as a seperator.

Sample Program

```
/*
    Program of Box Class
*/
class Box{
    double width;
    double height;
    double depth;
}
// This class declares an object of type Box
class BoxDemo{
    public static void main (String args[]){
    double vol;
    // assign values to instance variables
    Box mybox = new Box();
    mybox.width = 100;
    mybox.heigth = 125;
    mybox.depth = 157;
```

```
    vol = mybox.width * mybox.height * mybox.depth;
    //printing volume
    System.out.println("The volume is " + vol);
    }
}
```

NESTED CLASSES:-

⇒ In java, it is possible to define a class within another class, these classes are called nested classes.

→ They enable to logically group classes i.e., are only used in 1 place, thus increases use of encapsulation, creates more readable & maintainable code.

→ The scope of a nested class is bounded by scope of its enclasing class.

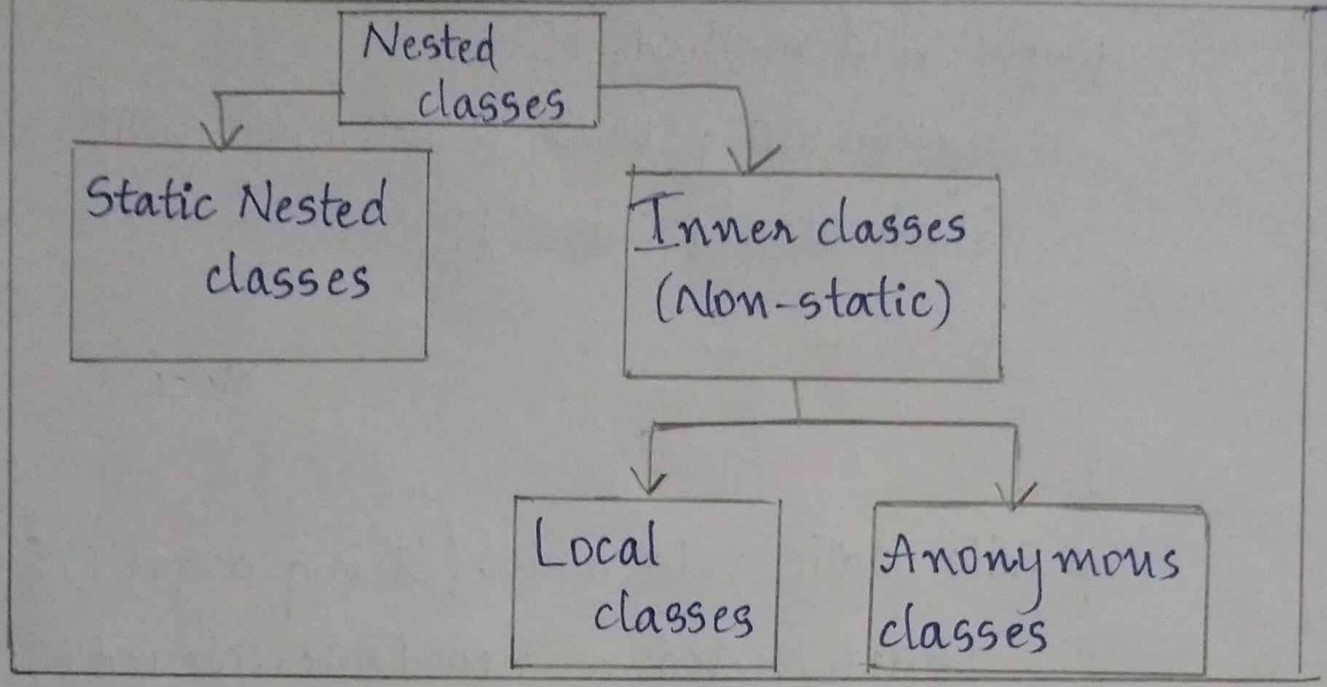
→ A nested class has access to members, including private members, of the class i.e., mentioned.

→ As a member of its enclosing class, nested class can be declared as private, public, protected

→ Nested class are divided into 2 categories:

1. static nested class: Nested classes that are declared as 'static'.

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



→ In case of normal / regular inner classes, without an outer class object existing, there cannot be an inner class object, i.e., an object of inner class is always strongly associated with an outer class object

→ But in case of static nested class, without an outer class object existing, there may be a static nested class object i.e., an object of a static nested class is not strongly associated with outer class object.

Example Program (Static Nested class)

```
public class Outer{
    public void method(){
        System.out.println("This is Outer Class");
    }
    static class Nested{
```

```
    public void method(){
        System.out.println("This belongs to
        Static Nested Class");
        }
    }
    public static void main(String args[]){
        Outer.Nested n = new Outer.Nested();
        n.method();
    }
}
```

Anonymous Inner class :-

→ It is an inner class without a name & for which only a single object is created. An anonymous inner class can be useful when making an instance of an object, with certain over loading methods of a class / interface, without having to actually sub-class a class.

Anonymous classes are mainly created in 2 ways:

1. Class (may be abstract)
2. Interface

Anonymous Class Example:

```
abstract class AnonClass{
        public abstract void method1();
}
public class Outer {
        public static void main (String args[]){
          AnonClass ac = new AnonClass()
          {
            public void method1(){
                System.out.println("This is an
                anonymous class);
            }
          };
          ac.method1;
        }
}
```

Local Inner Classes:

These are inner classes; that are defined inside a block. Local inner classes are not a member of any enclosing classes. They belong to the block they are defined, they have access to the fields of class enclosing it. They must be instantiated in the block defined in.

→ Local inner classes can extend an abstract class / can also implement an interface

→ A local inner class defined inside a method body, have access to its parameters.

Local inner class Example Program:

```
class Outer {
    class Inner {
        public void meth() {
            System.out.println("Inner Class");
        }
    }
    void print() {
        Inner inner = new Inner();
        inner.meth();
        }
    }
public class Main{
    public static void main (String args[]) {
        {
        Outer outer = new Outer();
        outer.print();
        }
    }
}
```

3) Describe a class RailwayTicket with following description:

Instance variables / data members:

String name: to store name of the 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 totamt: to store amount to be paid after updating the original amount.

Methods:

void accept(): to take input for name, coach, mobile number & amount.

void update(): to update amount as per coach selected.

Extra amount to be added 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, total amount and mobile number.

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

```
import java.io.*;
import java.util.Scanner;
class RailwayTicket {  //specifier-public/private
        String name;
        String coach;
        long mobno;
        int amt;
        int totalamt;
        }
        void accept()  //Throws Exception
```

Based on our choice, we can declare method as public, public (or) protected.

So, let the program be written by using public specifier.

Program:

```
import java.io.*;
import java.util.Scanner;
class RailwayTicket{
 public String name;
 public String coach;
 public long mobno;
 public int amt;
 public int totalamt;
    public void accept()
```

```
{
Scanner scan = new Scanner (System.in);
System.out.println ("Enter passenger name");
name = scan.nextline();
System.out.println ("Enter coach type");
coach = scan.nextline();
System.out.println ("Enter mobile number");
mobno = scan.nextlong();
System.out.println ("Enter total amount");
totalamt = scan.nextInt();
}
public void update()
{
  if (toach.equals(First_AC)){
      totalamt = amt + 700;
  }
  else if (coach.equals (Second_AC)){
      totalamt = amt+500;
  }
  elseif(coach.equals(Third_AC)){
      totalamt = amt+250;
  }
  else{
      totalamt = amt
} }
```

```
        public void display(){
            System.out.println("The passenger name
            is " + name);
            System.out.println("The coach type
            is " + coach);
            System.out.println("Mobile number
            is " + mobno);
            System.out.println("Total Amount is "
            + totalamt);
        }
    public static void main(String args[])
    {
        RailwayTicket ticket = new RailwayTicket();
        ticket.accept();
        ticket.update();
        ticket.display();
    }
}
```

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: $V = 4/3 \times 22/7 \times r3$

(ii) double volume(double h, double r) - with height 'h' and radius 'r' as the arguments, returns volume of a cylinder using formula: V = 22/7 X r2 x h

(iii) double volume(double l, double b, double h) - with length 'l', breadth 'b', & height 'h' as arguments, returns volume of a cuboid using formula: V = l x b x h

CODE:

```
class volume(){
    public static double volume(double R)
    {
        double V = 4.0/3 x 22.0/7 * Math.pow(R,3);
        return V;
    }
    public static double volume(double H, double R)
    {
        double V = 22.0/7 * R * R * H;
        return V;
    }
    public static double volume(double L, double B,
    double H) {
        double V = L * B * H;
        return V;
    }
}
```

Resources:-

1. (1st question) :- wikipedia, javaTpoint, educba.com, geeks for geeks.

2. (2nd question) :- Textbooks:

→ Java-The Complete Reference
-8th Edition

→ Introduction to Java Programming
-10th Edition.

→ geeks for geeks

3. (3rd question) :- Shaalaa.com

4. geeksfor geeks, Shaalaa.com