Aayush Raj Regmi

Roll no:181509

B.E. IT (DAY)

Section 3.

Solution:

package com.company;

class Shape {  
 private int xaxis;  
 private int yaxis;  
 private int length;  
 private int width;  
  
 public Shape(int x,int y,int l,int w){  
 this.xaxis=x;  
 this.yaxis=y;  
 this.length=l;  
 this.width=w;  
 System.*out*.println(xaxis+" "+yaxis+" "+length+" "+width);  
 }  
 public Shape(int l,int w){  
 this(0,0,l,w);  
 }  
 public Shape(){  
 this(0,0,1,1);  
 }  
}  
  
public class Main{  
  
 public static void main(String[] args){  
 Shape obj1=new Shape(9,8,7,6);  
 Shape obj2=new Shape(7,5);  
 Shape obj3 =new Shape();

}  
  
  
}

SECTION 2

Q.no.1.2.3.

package Aayush181509;

 class Student{

    private String name;

    private int roll;

    private String college;

    public void setData(String n,int r,String c){

        this.name=n;

        this.roll=r;

        this.college=c;

    }

    public void getData(){

        System.out.println("Name:"+name+"\n"+"Roll no "+roll+"\n"+"College "+college);

    }

}

public class Main{

    public static void main(String[] args) {

        Student obj1=new Student();

        obj1.setData("Aayush", 181509, "NCIT");

        obj1.getData();

    }

}

Q.no.4.

package Aayush181509;

 class Student{

    private String name;

    private int roll;

    private String college;

    public Student(String n,int r,String c){

        this.name=n;

        this.roll=r;

        this.college=c;

    }

    public void getData(){

        System.out.println("Name:"+name+"\n"+"Roll no "+roll+"\n"+"College "+college);

    }

}

public class Main{

    public static void main(String[] args) {

        Student obj1=new Student("Aayush", 181509, "NCIT");

        obj1.getData();

    }

}

Q.no. 5.

package Aayush181509;

 class Student{

    private String name;

    private int roll;

    private String college;

    static private int counter=0;

    public Student(String n,int r,String c){

        this.name=n;

        this.roll=r;

        this.college=c;

        counter+=1;

    }

    public void getData(){

        System.out.println("Name:"+name+"\n"+"Roll no "+roll+"\n"+"College "+college);

    }

    static void total(){

        System.out.println("No of Object Created= "+counter);

    }

}

public class Main{

    public static void main(String[] args) {

        Student obj1=new Student("Aayush", 181509, "NCIT");

        Student obj2=new Student("Ujjwal", 181547, "NCIT");

        Student obj3=new Student("Umesh", 181548, "NCIT");

        obj1.getData();

        obj2.getData();

        obj3.getData();

        Student.total();

    }

}

Section 1.

Q.no. 1. What are the features of Java ?

Following are the notable features of Java:

**Object Oriented**

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

**Platform Independent**

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

**Simple**

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

**Secure**

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

**Architecture-neutral**

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

**Portable**

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

**Robust**

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

**Multithreaded**

|  |  |  |
| --- | --- | --- |
| **Comparison Index** | **C++** | Java |
| **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. |
| **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| **Design Goal** | C++ was designed for systems and applications programming. It was an extension of [C programming language](https://www.javatpoint.com/c-programming-language-tutorial). | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience. |
| **Goto** | C++ supports the [goto](https://www.javatpoint.com/cpp-goto-statement) statement. | Java doesn't support the goto statement. |
| **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java). |
| **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. |
| **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java. |
| **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent. |
| **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| **Structure and Union** | C++ supports structures and unions. | Java doesn't support structures and unions. |
| **Thread Support** | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in [thread](https://www.javatpoint.com/multithreading-in-java) support. |
| **Documentation comment** | C++ doesn't support documentation comment. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. |
| **Virtual Keyword** | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| **unsigned right shift >>>** | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| **Inheritance Tree** | C++ creates a new inheritance tree always. | Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. |
| **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. |
| **Object-oriented** | C++ is an object-oriented language. However, in C language, single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

**Interpreted**

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

**High Performance**

With the use of Just-In-Time compilers, Java enables high performance.

**Distributed**

Java is designed for the distributed environment of the internet.

**Dynamic**

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

Q.no.2. Differences between Java and C++ .