*Practical Assignment – 2*
*Object Oriented Programming With Java*

## 1. Java Program to Create a Class and Object.

***CODE :***

```
public class clASS {
    void display(){
        System.out.println("Example of class and object !");
    }

    public static void main(String[] args) {
        clASS object = new clASS();
        object.display();
    }
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java clASS.java
Example of class and object !
```

## 2. Java Program to Create a Constructor and Constructor Overloading.

***CODE :***

```
class Box{
	int w;
	int l;
	int h;
	int weight;

	Box() {
		this.w = 2;
		this.l = 3;
		this.h = 5;
	}
	Box (int width ,int length ,int heigth) {
		this.w = width;
		this.l = length;
		this.h = heigth;
	}
	Box (int width ,int length ,int heigth ,int weight) {
		this.w = width;
		this.l = length;
		this.h = heigth;
		this.weight = weight;
	}
	public void display() {
		System.out.println("width:"+w);
		System.out.println("length:"+l);
		System.out.println("heigth:"+h);
		System.out.println("weight:"+weight);
	}

	public static void main(String args[]) {
		Box box1 = new Box(1,3,6);
		box1.display();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Box.java
width:1
length:3
heigth:6
weight:0
```

**3. Program that defines a circle class with two constructors. The first from accepts a double value that represents the radius of circle. This constructor assumes that the circle is centered at the origin. The ordinate of the center and the third arguments define the radius.**

***Code :***

```
class Circlefinder{
	double radius;
	Circlefinder (double r) {
		this.radius = r;
	}
	double calculateArea() {
		double pi = 3.14;
		return pi*radius*radius;
	}
	public static void main(String args[]) {
		Circlefinder circle = new Circlefinder(4);
		System.out.println("Radius: "+circle.radius+" Area of circle is:
"+circle.calculateArea());
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Circlefinder.java
Radius: 4.0 Area of circle is: 50.24
```

**4. Write a program of method overloading to created two methods, first add() method performs addition of two numbers and second add() method performs addition of three numbers.**

***CODE :***

```
class Addition{
	public static int add(int a, int b) {
		return a+b;
	}
	public static int add(int a, int b,int c) {
		return a+b+c;
	}
	public static void main(String args[]) {
		Addition plus = new Addition();
		System.out.println("Addition of two numbers:"+plus.add(1,2));
		System.out.println("Addition of three number:"+plus.add(1,2,3));
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Addition.java
Addition of two numbers:3
Addition of three number:6
```

**5. Write a static block which will be executed before main( ) method in a class.**

***CODE :***

```
public class Static {
	static {
		System.out.println("This is static block.");
	}
public static void main(String[] args) {
	System.out.println("This is main method.");
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Static.java
This is static block.
This is main method.
```

## 6. Java Program to Show multilevel Inheritance in Class.

***CODE :***

```
class Animal {
  void eat() {
    System.out.println("eating...");
  }
}
class Dog extends Animal {
  void bark() {
    System.out.println("barking...");
  }
}
class BabyDog extends Dog {
  void weep() {
    System.out.println("weeping...");
  }
}
class Test{
  public static void main(String args[]) {
    BabyDog d = new BabyDog();
    d.weep();
    d.bark();
    d.eat();
  }
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac Test.java

D:\Temp\JAVA A-2>java Test
weeping...
barking...
eating...
```

**7. Create a class to find out whether the given year is leap year or not. (Use inheritance for this program).**

***CODE :***

```
class LeapYear {
  int year;
  void checkLeapYear() {
    if (year % 4 == 0)
      System.out.println(year + " is leap year.");
    else
      System.out.println(year + " is not leap year.");
  }
}
public class Year {
  public static void main(String[] args) {
    LeapYear y1= new LeapYear();
    y1.year = 2012;
    y1.checkLeapYear();
    y1.year = 2005;
    y1.checkLeapYear();
  }
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Year
2012 is leap year.
2005 is not leap year.
```

## 8. Java Program to Show Overriding of Methods in Classes.

***CODE :***

```
class Parent {
   void Print() {System.out.println("parent class");}
}
class subclass1 extends Parent {
   void Print() { System.out.println("subclass1"); }
}
class subclass2 extends Parent {
   void Print() {System.out.println("subclass2");}
}

class OrMain {
   public static void main(String[] args)
   {
      Parent a;
      a = new subclass1();
      a.Print();

      a = new subclass2();
      a.Print();
   }
}
```

***OUTPUT :***

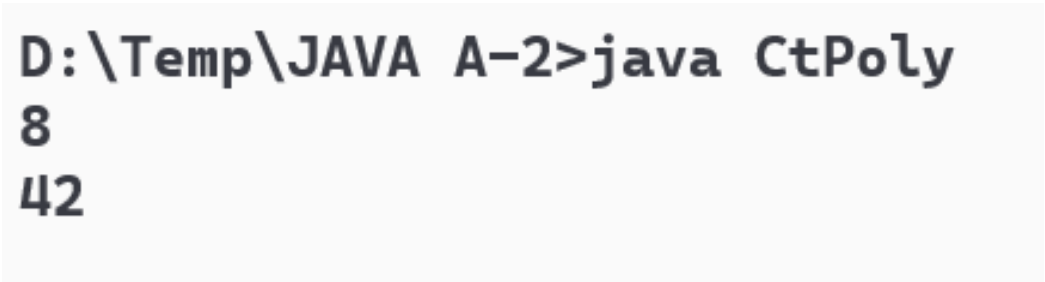
```
D:\Temp\JAVA A-2>java OrMain
subclass1
subclass2
```

**9. Java Program to Show compile time Polymorphism in Class.**

***CODE :***

```
class Multiplyer {
    static int Multiply (int a,  int b) {
        return a * b;
    }
    static int Multiply (int a , int b , int c) {
        return a * b * c;
    }
}
class CtPoly {
    public static void main(String[] args)
    {
        System.out.println(Multiplyer.Multiply(2, 4));
        System.out.println(Multiplyer.Multiply(2, 7, 3));
    }
}
```

***OUTPUT :***



## 10.Java Program to Show run time Polymorphism in Class.

***CODE :***

```
class Vehical{
	int numOfTyers() {return 0;}
}
	class Bike extends Vehical{
		int numOfTyers() {return 2;}
	}
	class Rickshaw extends Vehical{
		int numOfTyers() {return 3;}
	}
	class Car extends Vehical{
		int numOfTyers() {return 4;}
	}
	class Truck extends Vehical {
		int numOfTyers() {return 6;}
	}
	class Vcall{
		public static void main(String args[]) {
			Bike bike = new Bike();
			Rickshaw rickshaw = new Rickshaw();
			Car car = new Car();
			Truck truck = new Truck();

			System.out.println("Bike number of tyers: "+bike.numOfTyers());
			System.out.println("Rickshaw number of tyers: "+rickshaw.numOfTyers());
			System.out.println("Car number of tyers: "+car.numOfTyers());
			System.out.println("Truck number of tyers: "+truck.numOfTyers());
		}
	}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac Vcall.java

D:\Temp\JAVA A-2>java Vcall
Bike number of tyers: 2
Rickshaw number of tyers: 3
Car number of tyers: 4
Truck number of tyers: 6
```

## 11.Java Program to Show Use of Super Keyword in Class.

***CODE :***

```
class Shapes{
	String shapeRequirment = "height, width, length";
}
class Circle extends Shapes {
	String shapeRequirment ="radius";

	void display1() {
		System.out.println("Basic requirments of shapes:
"+super.shapeRequirment);
		}
	void display2() {
		System.out.println("Requirments of circle shape:
"+shapeRequirment);
	}
}
class Super{
	public static void main(String args[]) {
		Circle c = new Circle();
		c.display1();
		c.display2();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Super
Basic requirments of shapes: height, width, length
Requirments of circle shape: radius
```

**12.Java Program to Show Use of This Keyword in Class.**

***CODE :***

```
class Student{
	String name;
	int no;

	Student (String sName , int sNo) {
		this.name = sName;
		this.no = sNo;
	}

	void display () {
		System.out.println("Student name: "+this.name);
		System.out.println("Student number: "+this.no);
	}
	public static void main(String args[]) {
		Student s1= new Student("Tirth",339);
		s1.display();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Student
Student name: Tirth
Student number: 339
```

**13.Java Program to Show Usage of Static keyword in Class.**

***CODE :***

```
class Student{

	static void display()
	{
		System.out.println("Students should be inovative, not scholors. ");
	}
	public static void main(String args[]) {
		display();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Student.java
Students should be inovative, not scholors.
```

## 14.Java Program to Create Abstract Class.

***CODE :***

```
abstract class Classroom{
	abstract void printInfo();
}
class  StuInfo extends Classroom {
	void printInfo() {
		int enrollNo = 339;
		String name = "Tirth";
		int age = 18;
		String exp = "Development";

		System.out.println("Enrollment number: "+enrollNo);
		System.out.println("Student name: "+name);
		System.out.println("Student age: "+age);
		System.out.println("Student expertise: "+exp);
	}
}
class abstractC{
	public static void main(String args[]) {
		Classroom s1 = new StuInfo();
		s1.printInfo();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac abstractC.java

D:\Temp\JAVA A-2>java abstractC
Enrollment number: 339
Student name: Tirth
Student age: 18
Student expertise: Development
```

**15.Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object**

***CODE :***

```
abstract class Shapes {  abstract double area();  }
class Circle extends Shapes {
   double radius;
        double pi = 3.14;
        Circle (double radius) { this.radius = radius; }
        double area() {return pi*radius*radius;}
}
class Triangle extends Shapes {
        double base;
        double height;
        Triangle (double base, double height) {
                this.base = base;
                this.height = height;
        }
        double area () { return 0.5*base*height; }
}
class Rectangle extends Shapes {
        double length;
        double width;
        Rectangle (double l ,double w) {
                this. length = l;
                this.width = w;
        }
        double area () { return length*width; }
}
class Calc {
   public static void main(String[] args) {
      Shapes circle = new Circle(4);
      Shapes triangle = new Triangle(5,6);
                Shapes rectangle = new Rectangle(6,7);

                System.out.println("Area of circle: "+ circle.area());
                System.out.println("Area of triangle: "+ triangle.area());
                System.out.println("Area of rectangle: "+ rectangle.area());
   }
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Calc
Area of circle: 50.24
Area of triangle: 15.0
Area of rectangle: 42.0
```

**16.Java Program to Create an Interface.**

***CODE :***

```
interface Student{
	public void info();
}
class StuInfo implements Student {
	public void info () {
		int enrollNo = 339;
		String name = "Tirth";
		int age = 18;
		String exp = "Development";

		System.out.println("Enrollment number: "+enrollNo);
		System.out.println("Student name: "+name);
		System.out.println("Student age: "+age);
		System.out.println("Student expertise: "+exp);
	}
}
public class inter {
	public static void main(String argsp[]) {
		StuInfo s1 = new StuInfo();;
		s1.info();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac inter.java

D:\Temp\JAVA A-2>java inter
Enrollment number: 339
Student name: Tirth
Student age: 18
Student expertise: Development
```

**17.Write a program in Java to demonstrate implementation of multiple inheritance using interfaces.**

***CODE :***

```
interface A {
	default void display () {
		System.out.println("This is class 'A' ");
	}
}
interface B {
	default void display () {
		System.out.println("This is class 'B' ");
	}
}
class mInheritance implements A,B {
	public void display() {
		A.super.display();
		B.super.display();
	}
	public static void main (String args[]) {
		mInheritance test = new mInheritance();
		test.display();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac mInheritance.java

D:\Temp\JAVA A-2>java mInheritance
This is class 'A'
This is class 'B'
```

**18.Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.**

***CODE :***

```
class Bank {
   double getRateOfInterest() { return 0.0; }
}

class SBI extends Bank {
   double getRateOfInterest() { return 8.0; }
}

class ICICI extends Bank {
   double getRateOfInterest() { return 7.0; }
}

class AXIS extends Bank {
   double getRateOfInterest() { return 9.0; }
}

public class Main {
   public static void main(String[] args) {
      Bank sbiBank = new SBI();
      Bank iciciBank = new ICICI();
      Bank axisBank = new AXIS();

      System.out.println("SBI Bank Rate of Interest: " +
sbiBank.getRateOfInterest() + "%");
      System.out.println("ICICI Bank Rate of Interest: " +
iciciBank.getRateOfInterest() + "%");
      System.out.println("AXIS Bank Rate of Interest: " +
axisBank.getRateOfInterest() + "%");
   }
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>javac Main.java

D:\Temp\JAVA A-2>java Main
SBI Bank Rate of Interest: 8.0%
ICICI Bank Rate of Interest: 7.0%
AXIS Bank Rate of Interest: 9.0%
```

**19.Write a program that illustrates interface inheritance. Interface P12 inherits from both P1 and P2. Each interface declares one constant and one method. The class Qimplements P12. Instantiate Q and invoke each of its methods. Each method displays one of the constants**

***CODE :***

```
interface P1{
	int a=101;
	void displayP1();
}
interface P2{
	int b = 201;
	void displayP2();
}
interface P12 extends P1,P2{
	int c  = 301;
	void displayP12();
}
class Q implements P12 {
	public void displayP1() {
		System.out.println("A: "+a);
	}
	public void displayP2() {
		System.out.println("B: "+b);
	}
	public void displayP12() {
		System.out.println("C: "+c);
	}
}
public class Qtest {
	public static void main(String args[]) {
		Q q = new Q();
		q.displayP1();
		q.displayP2();
		q.displayP12();
	}
}
```

***OUTPUT :***

```
D:\Temp\JAVA A-2>java Qtest
A: 101
B: 201
```