3.8.1  Writing a program in Java to demonstrate the uses of classes and objects.

package classObject;

public class classObject {

int i = 5;

public static void main(String[] args) {

classObject a= new classObject();

classObject b= new classObject();

System.out.println(a.i);

System.out.println(b.i);

}

}

3.8.2 Writing a program in Java to demonstrate the uses of polymorphism.

package classObject;

class Helper {

static int Multiply(int a, int b)

{

return a \* b;

}

static int Multiply(int a, int b, int c)

{

return a \* b \* c;

}

}

public class polymorphism {

public static void main(String[] args) {

System.out.println(Helper.Multiply(2, 4));

System.out.println(Helper.Multiply(2, 7, 3));

}

}

3.8.3 Writing a program in Java to demonstrate the uses of inheritance.

package classObject;

class Bicycle {

public int gear;

public int speed;

public Bicycle(int gear, int speed)

{

this.gear = gear;

this.speed = speed;

}

public void applyBrake(int decrement)

{

speed -= decrement;

}

public void speedUp(int increment)

{

speed += increment;

}

public String toString()

{

return ("No of gears are " + gear + "\n"

+ "speed of bicycle is " + speed);

}

}

class MountainBike extends Bicycle {

public int seatHeight;

public MountainBike(int gear, int speed,

int startHeight)

{

// invoking base-class(Bicycle) constructor

super(gear, speed);

seatHeight = startHeight;

}

public void setHeight(int newValue)

{

seatHeight = newValue;

}

@Override public String toString()

{

return (super.toString() + "\nseat height is "

+ seatHeight);

}

}

public class inheritance {

public static void main(String[] args) {

MountainBike mb = new MountainBike(3, 100, 25);

System.out.println(mb.toString());

}

}

3.8.4 Writing a program in Java to demonstrate the uses of encapsulation.

package classObject;

class Encapsulate {

private String Name;

private int Roll;

private int Age;

public int getAge() { return Age; }

public String getName() { return Name; }

public int getRoll() { return Roll; }

public void setAge(int newAge) { Age = newAge; }

public void setName(String newName)

{

Name = newName;

}

public void setRoll(int newRoll)

{

Roll = newRoll;

}

}

public class Enclapsulation {

public static void main(String[] args)

{

Encapsulate obj = new Encapsulate();

obj.setName("Harsh");

obj.setAge(23);

obj.setRoll(77);

System.out.println("name: " + obj.getName());

System.out.println("age: " + obj.getAge());

System.out.println("roll: " + obj.getRoll());

}

}

3.8.5 Writing a program in Java to demonstrate the uses of abstraction.

package classObject;

abstract class Shape {

String color;

abstract double area();

public abstract String toString();

public Shape(String color)

{

System.out.println("Shape constructor called");

this.color = color;

}

public String getColor() { return color; }

}

class Circle extends Shape {

double radius;

public Circle(String color, double radius)

{

super(color);

System.out.println("Circle constructor called");

this.radius = radius;

}

@Override double area()

{

return Math.PI \* Math.pow(radius, 2);

}

@Override public String toString()

{

return "Circle color is " + super.getColor()

+ "and area is : " + area();

}

}

class Rectangle extends Shape {

double length;

double width;

public Rectangle(String color, double length,

double width)

{

// calling Shape constructor

super(color);

System.out.println("Rectangle constructor called");

this.length = length;

this.width = width;

}

@Override double area() { return length \* width; }

@Override public String toString()

{

return "Rectangle color is " + super.getColor()

+ "and area is : " + area();

}

}

public class Abstraction {

public static void main(String[] args) {

Shape s1 = new Circle("Red", 2.2);

Shape s2 = new Rectangle("Yellow", 2, 4);

System.out.println(s1.toString());

System.out.println(s2.toString());

}

}