# MODULE-2

Q-1 Write a program to create a “distance” class with methods where distance is computed in terms of feet and inches, create objects of a class.

CODE :

```
import java.util.*;

public class Distance {

  int feet;

  int inches;

  void getData() {

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter distance in feet: ");

    feet = sc.nextInt();

    System.out.print("Enter distance in inches: ");

    inches = sc.nextInt();


    if(inches >= 12) {

      feet += inches / 12;

      inches %= 12;

    }

  }


  void display() {

    System.out.println("Distance : " + feet + "'" + inches);

  }


  void add(Distance d) {

    feet += d.feet;

    inches += d.inches;


    if(inches >= 12) {

      inches -= 12;
```

```
            feet += 1;
        }
    }
    public static void main(String[] args) {
        Distance d1 = new Distance();
        Distance d2 = new Distance();
        System.out.println("For first object");
        d1.getData();
        d1.display();
        System.out.println("For second object");
        d2.getData();
        d2.display();
        System.out.println("Addition");
        d1.add(d2);
        d1.display();
    }
}
```

OUTPUT :

```
For first object
Enter distance in feet: 23
Enter distance in inches: 45
Distance : 26'9
For second object
Enter distance in feet: 33
Enter distance in inches: 43
Distance : 36'7
Addition
Distance : 63'4
```

Q-2 Modify the “distance” class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.

CODE :

```
import java.util.*;

 class Distance {
    public int feet, inch;

    public Distance() {
        this.feet = 4;
        this.inch = 3;
    }

    public Distance(int feet, int inch) {
        this.inch = inch % 12;
        this.feet = feet + inch / 12;
    }

    public Distance(Distance d) {
        this.feet = d.feet;
        this.inch = d.inch;
    }

    public void display() {
        System.out.println(this.feet + "'" + this.inch);
    }

    public void sum(Distance d) {
        this.inch = (this.inch + d.inch) % 12;
        this.feet = (this.feet + d.feet) + d.inch / 12;
    }


}
public class MainDistance {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.println("Default constructor:");
        Distance d1 = new Distance();
        d1.display();


        System.out.println("Parameterized constructor:");
        System.out.println("Enter distances in feet and inches respectively: ");
        Distance d2 = new Distance(sc.nextInt(), sc.nextInt());
        d2.display();


        System.out.println("Copy constructor:");
        Distance d3 = new Distance(d2);
        d3.display();


    }
}
```

OUTPUT :

```
Default constructor:
4'3
Parameterized constructor:
Enter distances in feet and inches respectively:
7 8
7'8
Copy constructor:
7'8
```

Q-3 Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.

CODE :

```
import java.util.*;


 class Employee {
    protected int ID;
    protected String name;
    protected static int count = 0;


    public Employee() {
        this.ID = 0;
        this.name = "Not assigned";
        Employee.count++;
    }


    public Employee(int ID, String name) {
        this.ID = ID;
        this.name = name;
        Employee.count++;
    }


    public Employee(Employee emp) {
        this.ID = emp.ID;
        this.name = emp.name;
        Employee.count++;
    }


    public void setID(int ID) {
        this.ID = ID;
    }
    public void setName(String name) {
        this.name = name;
    }

    public int getID() {
        return this.ID;
    }

    public String getName() {
        return this.name;
    }

    public static int getCount() {
        return Employee.count;
    }

    public String toString() {
        return "ID: " + this.ID + "\tName: " + this.name;
    }


}

class Teacher extends Employee {
    protected String qualification;

    public Teacher() {
        this.qualification = "not assigned";
    }
    public Teacher(int ID, String name, String qualification) {
        super(ID, name);
        this.qualification = qualification;
    }

    public String getQualification() {
        return this.qualification;
    }

    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

    public String toString() {
        return super.toString() + "\tQualification: " + this.qualification;
    }
}
public class Test3 {
    public static void main(String[] args) {
        Employee emp1 = new Employee(10, "Shruti");
        Employee emp2 = new Teacher(11, "Anand", "M.Tech");



        System.out.println("Employee 1: " + emp1.toString());
        System.out.println("Employee 2: " + emp2.toString());

    }
}
```

OUTPUT:

```
Employee 1: ID: 10	Name: Shruti
Employee 2: ID: 11	Name: Anand	Qualification: M.Tech
```

Q-4 Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object.

CODE :

```
import java.util.*;


 class Student {
    private int rollNo;
    private String name;
    private float cgpa;


    public static int count = 0;


    public Student() {
        this.rollNo = 0;
        this.name = null;
        this.cgpa = 0.0f;
        count++;
    }


    public Student(int rollNo, String name, float cgpa) {
        this.rollNo = rollNo;
        this.name = name;
        this.cgpa = cgpa;
        count++;
    }


    public Student(Student s) {
        this.rollNo = s.rollNo;
        this.name = s.name;
        this.cgpa = s.cgpa;
        count++;
    }

    public void display() {
        System.out.println("Roll number: " + this.rollNo);
        System.out.println("Name: " + this.name);
        System.out.println("CGPA: " + this.cgpa);
    }
}



public class MainStudent {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student s1, s2, s3;

        System.out.println("Default constructor for s1:");
        s1 = new Student();
        s1.display();

        System.out.println("Parameterized constructor for s2:");
        System.out.print("Enter roll number: ");
        int roll = sc.nextInt();
        System.out.print("Enter name of the student: ");
        String name = sc.nextLine();
        System.out.print("Enter CGPA: ");
        float cgpa = sc.nextFloat();
        s2 = new Student(roll, name, cgpa);
        s2.display();

        System.out.println("Copy constructor for s3:");
        s3 = new Student(s2);
        s3.display();
    }
}
```

OUTPUT :

```
Default constructor for s1:
Roll number: 0
Name: null
CGPA: 0.0
Parameterized constructor for s2:
Enter roll number: 21
Enter name of the student: Rahul
Enter CGPA: 8
Roll number: 21
Name: Rahul
CGPA: 8.0
Copy constructor for s3:
Roll number: 21
Name: Rahul
CGPA: 8.0
```

Q-5 Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword.

CODE :

```
import java.util.*;

class ABC {
    private int age;
    public int mark;
    public final int x = 5;

    public void setData() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter age: ");
        age = sc.nextInt();

        System.out.println("Enter mark: ");
        mark = sc.nextInt();
    }

    public void disp() {
        System.out.println("Age: " + age + "\nMark: " + mark + "\nX: " + x);
    }

    public void incr(int y) {
        System.out.println("Passed by value: " + ++y);
    }

    public void incr(ABC t) {
        System.out.println("Passed by reference: " + ++t.mark);
    }
}

public class MainAccess {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ABC o1 = new ABC();
        ABC o2 = new ABC();

        try {
            o1.mark = sc.nextInt();
            o1.age = sc.nextInt();
        } catch(Exception e) {System.out.println(e);}

        o2.setData();

        o1.disp();
        o2.disp();

        int y = 8;
        o1.incr(y);
        System.out.println("y: " + y);

        o1.incr(o2);
        System.out.println("Mark: " + o2.mark);

        try {
            o1.x = 69;
        } catch(Exception e) {System.out.println(e);}
    }
}
```

OUTPUT :

```
MainAccess.java:38: error: age has private access in ABC
            o1.age = sc.nextInt();
              ^
MainAccess.java:54: error: cannot assign a value to final variable x
            o1.x = 69;
              ^
2 errors
```

# if we change the private keyword to public then it will be compiled and if we remove the final keyword then the 2nd error will get solved.

Q-6 Write a program to show the use of static functions and to pass variable length arguments in a function.

CODE :

```
import java.util.*;


class Main6 {
    static void varArgs(String ...names) {
        System.out.println("Number of names passed: " + names.length);
        for(String name: names)
            System.out.println(name);
        System.out.println();
    }


    public static void main(String[] args) {
        varArgs("Shruti ", " Shivraj ", "Rahul");
        varArgs("Murli", "Dhvani");
        varArgs();
    }
}
```

OUTPUT :

```
Number of names passed: 3
Shruti
Shivraj
Rahul

Number of names passed: 2
Murli
Dhvani
Number of names passed: 0
```

Q-7 Develop minimum 4 program based on variation in methods i.e., passing by value, passing by reference, returning values and returning objects from methods.

//pass by value

CODE :

```
class PassByValue7{

   String name="java";
   void display(String name)
   {
      name="Java Programming";
   }
   public static void main(String arg[])
   {
      PassByValue7 p=new PassByValue7();
      System.out.println("Before changes : "+p.name);
      p.display("C++");
      System.out.println("After changes : "+p.name);
   }
}
```

OUTPUT :

```
Before changes : java
After changes : java
```

// pass by reference

CODE :

```
class PassByReference7{

   String name="Java";
   void display(PassByReference7 p)
   {
       name="Java Programming";
   }
   public static void main(String arg[])
   {
       PassByReference7 p=new PassByReference7();
       System.out.println("Before changes : "+p.name);
       p.display(p);
       System.out.println("After changes : "+p.name);
   }
}
```

OUTPUT :

```
Before changes : Java
After changes : Java Programming
```

// returning value

CODE :

```
class ReturningValue7{

    int n1,n2;
    void getValue(int a,int b)
    {
        n1=a;
        n2=b;
    }
    int returnMax()
    {
        return n1>n2?n1:n2;
    }
    public static void main(String arg[])
    {
        ReturningValue7 R=new ReturningValue7();
        R.getValue(12,50);
        System.out.println("Maximum is : "+R.returnMax());
    }
}
```

OUTPUT :

```
Maximum is : 50
```

// returning object

CODE :

```
class ReturningObject7{

    int n1,n2;
    ReturningObject7(int a,int b)
    {
        n1=a;
        n2=b;
    }
    ReturningObject7 minNumber()
    {
        ReturningObject7 min=new ReturningObject7(50,30);
        return min;
    }
    public static void main(String arg[])
    {
        ReturningObject7 R1=new ReturningObject7(60,200);
        ReturningObject7 R2=R1.minNumber();
        int minR1=R1.n1<R1.n2?R1.n1:R1.n2;
        System.out.println("Minimum of (R1) "+R1.n1+" & "+R1.n2+" is "+minR1);
        int minR2=R2    .n1<R2.n2?R2.n1:R2.n2;
        System.out.println("Minimum of (R2) "+R2.n1+" & "+R2.n2+" is "+minR2);
    }
}
```

OUTPUT :

```
Minimum of (R1) 60 & 200 is 60
Minimum of (R2) 50 & 30 is 30
```

Q-8 Write a program that implements two constructors in the class. We call the other constructor using ‘this’ pointer, from the default constructor of the class.

CODE :

```
class Main8 {
    public int x, y;

    public Main8() {
        this(23,30); // If calling constructor inside a method in class using this(), then this statement should be the first line.
        x = 4; y = 3;
        this.disp();
    }

    public Main8(int a, int b) {
        System.out.println("Parameterized constructor");
        x = a;
        y = b;
        this.disp();
    }

    public void disp() {
        System.out.println("x: " + x);
        System.out.println("y: " + y);
    }
    public static void main(String[] args) {
        Main8 o1 = new Main8();
        o1.disp();
    }
}
```

OUTPUT:

```
Parameterized constructor
x: 23
y: 30
x: 4
y: 3
x: 4
y: 3
```

Q-9 Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance

CODE :

```
//single inheritence

class Animal {
    public Animal() {
        System.out.println("Animal is created");
    }

    void eat() {
        System.out.println("Eating");
    }
}

class Dog extends Animal {
    public Dog() {
        // super();
        System.out.println("Dog is created");
    }

    void bark() {
        System.out.println("Barking");
    }
}

public class SingleInheritance {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();
    }
}



//multilevel

class Animal {
    public Animal() {
        System.out.println("Animal is created");
    }

    void eat() {
        System.out.println("Eating");
    }
}

class Dog extends Animal {
    public Dog() {
        // super();
        System.out.println("Dog is created");
    }

    void bark() {
        System.out.println("Barking");
    }
}

class BabyDog extends Dog {
    void weep() {
        System.out.println("Weeping");
    }
}

public class MultilevelInheritance {
    public static void main(String[] args) {
        BabyDog d = new BabyDog();
        d.eat();
        d.bark();
        d.weep();
    }
}
// heirarchial

class Animal {
    public Animal() {
        System.out.println("Animal is created");
    }

    void eat() {
        System.out.println("Eating");
    }
}

class Dog extends Animal {
    public Dog() {
        // super();
        System.out.println("Dog is created");
    }

    void bark() {
        System.out.println("Barking");
    }
}

class Cat extends Animal {
    void meow() {
        System.out.println("Meow");
    }
}

public class HeirarchicalInheritance {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();
        Cat c = new Cat();
        c.eat();
        c.meow();
    }
}
```

OUTPUT :

```
//single level
Animal is created
Dog is created
Eating
Barking

// multilevel
Animal is created
Dog is created
Eating
Barking
Weeping

//hierarchical
Animal is created
Dog is created
Eating
Barking
Animal is created
Eating
Meow
```

Q-10 Java Program to demonstrate the real scenario (e.g., bank) of Java Method Overriding where three classes are overriding the method of a parent class. Creating a parent class.

CODE :

```
class Employee {
   protected int ID;
   protected String name;
   protected static int employeeCount = 0;
```

```
    public Employee() {
        System.out.println("Employee created");
        System.out.println("ID and names are not initialized.");
        employeeCount++;
    }

    public Employee(int ID, String name) {
        accept(ID, name);
        employeeCount++;
    }

    public void accept(int ID, String name) {
        this.ID = ID;
        this.name = name;
    }

    public void display() {
        System.out.println("Employee ID: " + this.ID);
        System.out.println("Employee Name: " + this.name);
    }
}

 class Engineer extends Employee{
    protected String company;
    protected int engineerCount = 0;

    public Engineer() {
        engineerCount++;
    }

    public Engineer(int ID, String name, String company) {
        super(ID, name);
        accept(company);
        engineerCount++;
    }

    public void accept(String company) {
        this.company = company;
    }

    public void display() {
        super.display();
        System.out.println("Company: " + this.company);
    }
}

 class Teacher extends Employee{
    protected String school;
    protected int teacherCount = 0;

    public Teacher() {
        teacherCount++;
    }

    public Teacher(int ID, String name, String school) {
        super(ID, name);
        accept(school);
        teacherCount++;
    }

    public void accept(String school) {
        this.school = school;
    }

    public void display() {
        super.display();
        System.out.println("School: " + this.school);
    }
}

public class Main10 {
    public static void main(String[] args) {
        Employee e1 = new Employee(1, "Ram");
        Teacher t1 = new Teacher(2, "Laxman", "KV");
        Engineer en1 = new Engineer(2, "Shyam", "Google");

        e1.display();
        t1.display();
        en1.display();
    }
}
```

OUTPUT :

```
Employee ID: 1
Employee Name: Ram
Employee ID: 2
Employee Name: Laxman
School: KV
Employee ID: 2
Employee Name: Shyam
Company: Google
```

Q-11 Write a program that implements simple example of Runtime Polymorphism with multilevel inheritance.

CODE :

```
import java.util.*;

 class Student {
    protected int rollNo;
    protected String name;
    protected static int studentCount = 0;

    public Student() {
        studentCount++;
    }

    public Student(int rollNo, String name) {
        this.accept(rollNo, name);
        studentCount++;
    }

    public void accept(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
    }

    public void display() {
        System.out.println("Display method of Student class");
        System.out.println("Roll number: " + this.rollNo);
        System.out.println("Name: " + this.name);
    }
}
```

```
 class EngineeringStudent extends Student{
    protected String branch;
    protected static int engineeringStudentCount = 0;

    public EngineeringStudent() {engineeringStudentCount++;}

    public EngineeringStudent(int rollNo, String name, String branch) {
        super(rollNo, name);
        accept(branch);
        engineeringStudentCount++;
    }

    public void accept(String branch) {
        this.branch = branch;
    }

    public void display() {
        System.out.println("Display method of EngineeringStudent class");
        System.out.println("Roll number: " + this.rollNo);
        System.out.println("Name: " + this.name);
        System.out.println("Branch: " + this.branch);
    }
}

 class CSE_Student extends EngineeringStudent{
    protected static int cseStudentCount = 0;
    protected String favouriteProgrammingLanguage;

    public CSE_Student() {cseStudentCount++;}
    public CSE_Student(int rollNo, String name, String favouriteProgrammingLanguage) {
        super(rollNo, name, "Computer Science & Engineering");
        this.accept(favouriteProgrammingLanguage);
    }

    public void accept(String favouriteProgrammingLanguage) {
        this.favouriteProgrammingLanguage = favouriteProgrammingLanguage;
    }

    public void display() {
        System.out.println("Display method of CSE_Student class");
        System.out.println("Roll No: " + this.rollNo);
        System.out.println("Name: " + this.name);
        System.out.println("Branch: " + this.branch);
        System.out.println("Favourite Programming Language: " + this.favouriteProgrammingLanguage);
    }

}

public class Main11 {
    public static void main(String[] args) {
        Student s1 = new Student(100, "rahul kumar");
        EngineeringStudent s2 = new EngineeringStudent(101, "shyam sharma", "Electrical
Engineering");
        CSE_Student s3 = new CSE_Student(102, "kamlesh kumar", "JAVA");

        s1.display();
        s2.display();
        s3.display();
    }
}
```

OUTPUT :

```
Display method of Student class

Roll number: 100

Name: rahul kumar

Display method of EngineeringStudent class

Roll number: 101

Name: shyam sharma

Branch: Electrical Engineering

Display method of CSE_Student class

Roll No: 102

Name: kamlesh kumar

Branch: null

Favourite Programming Language: JAVA
```

Q-12 Write a program to compute if one string is a rotation of another. For example, pit is rotation of tip as pit has same character as tip.

CODE:

```
import java.util.*;


class Main12 {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      String s1 = sc.next();

      String s2 = sc.next();

      if(checkRotation(s1, s2))

         System.out.println(s1 + " and " + s2 + " are reverse of each other");

      else

         System.out.println(s1 + " and " + s2 + " are not reverse of each other");

   }


   static boolean checkRotation(String s1, String s2) {
        if(s1.length() == s2.length()) {
            int length = s1.length();
            for(int i=0; i<length; i++)
                if(s1.charAt(i) != s2.charAt(length-i-1))
                    return false;
            return true;
        } return false;
    }
}
```

OUTPUT :

```
sun
nus
sun and nus are reverse of each other
```

Q-13 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 i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.

CODE :

```
import java.util.*;

abstract class Shape {
    public abstract void area();
}

class Circle extends Shape {
    private float radius;

    public Circle() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter radius of circle: ");
        this.radius = sc.nextFloat();
    }

    public Circle(float radius) {
        this.radius = radius;
    }

    public void area() {
        System.out.println("Area of circle: " + Math.PI * this.radius * this.radius);
    }
}

class Rectangle extends Shape {
    private float length, breadth;

    public Rectangle() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter length and breadth of the rectangle: ");
        this.length = sc.nextFloat();
        this.breadth = sc.nextFloat();
    }

    public Rectangle(float length, float breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public void area() {
        System.out.println("Area of rectangle: " + this.length * this.breadth);
    }
```

```
}

class Triangle extends Shape {
   private float side1, side2, side3;

   public Triangle() {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the three sides of the triangle: ");
      this.side1 = sc.nextFloat();
      this.side2 = sc.nextFloat();
      this.side3 = sc.nextFloat();
   }

   public Triangle(float side1, float side2, float side3) {
      this.side1 = side1;
      this.side2 = side2;
      this.side3 = side3;
   }

   public void area() {
      float s = (side1 + side2 + side3) / 2;
      System.out.println("Area of triangle: " + Math.sqrt(s*(s-side1)*(s-side2)*(s-side3)));
   }
}

public class Main13 {
   public static void main(String[] args) {
      Circle obj1 = new Circle(1);
      Rectangle obj2 = new Rectangle(3, 4);
      Triangle obj3 = new Triangle(3, 4, 5);
```

```
        obj1.area();
        obj2.area();
        obj3.area();
    }
}
```

OUTPUT :

```
Area of circle: 3.141592653589793
Area of rectangle: 12.0
Area of triangle: 6.0
```

Q-14 Write a program in Java to demonstrate multiple inheritance.

CODE :

```
interface Printable{
                void print();
}
interface Showable{
                void show();
}
class Main14 implements Printable,Showable{
public void print(){System.out.println("JAVA");}
public void show(){System.out.println("THE KING OF PROGRAMMING");}
public static void main(String args[]){
Main14 obj = new Main14();
obj.print();
obj.show();
 }
}
```

OUTPUT :

```
JAVA
THE KING OF PROGRAMMING
```

Q-15

a) Write an application that illustrates method overriding in the same package and different packages.
b) Also demonstrate accessibility rules in inside and outside packages.

// a

CODE :

```
class Restaurant

{

   void eat { System.out.println("Inside restaurant"); }

}


class Mall extends Restaurant

{

                        void eat(){ System.out.println("Inside restaurant"); }


}

public class Bill

{

   public static void main(String arg[])

   {

      Mall o1=new Mall();

      o1.eat();

   }

}
```

OUTPUT :

```
Inside restaurant    (compiled bill.java file)
```

//b

```
 package residency;
```

```
public abstract class Residency{
   String name;
   int rNumber;
   int area;
   int unit_rate;

   abstract long getPriceOfResidency();

   public Residency(){
      System.out.println("Default constructor");
   }
   public Residency(String name , int rNumber, int area){
      this.name = name;
      this.rNumber = rNumber;
      this.area = area;



   }
   public Residency(Residency r){
      r.name = name;
      r.rNumber = rNumber;
      r.area = area;
   }

   public
   String getName(){
      return name;
   }

   void setName(String name){
      this.name=name;
```

```
    }

    int getRNumber(){
        return rNumber;
    }

    void setRNumber(int rNumber){
        this.rNumber = rNumber;
    }

    int getArea(){
        return area;
    }

    void setArea(int area){
        this.area =area;
    }

    int getUnitRate(){
        return unit_rate;
    }

    void setUnitRate(int unit_rate){
        this.unit_rate = unit_rate;
    }

}

public class LuxuriousResidency extends Residency{
    int amenityCharge;
    public LuxuriousResidency(int amenityCharge){
        this.amenityCharge = amenityCharge;
    }

    long getPriceOfResidency() {
        return 500000+ amenityCharge;
    }

}

public class Semi_FurnishedResidency extends Residency{
    int furnitureCharge,parkingCharge;

    public Semi_FurnishedResidency(int furnitureCharge , int parkingCharge){
        this.furnitureCharge = furnitureCharge;
        this.parkingCharge = parkingCharge;

    }
    long getPriceOfResidency() {
        return 250000+furnitureCharge+parkingCharge;
    }

}

interface Rentable {
    abstract public int getRent();
}


public class TwoBhkResidency extends Semi_FurnishedResidency implements Rentable{
    public TwoBhkResidency(int furnitureCharge, int parkingCharge) {
        super(furnitureCharge, parkingCharge);
    }

    long getPriceOfResidency() {
        return super.getPriceOfResidency();
    }

    public int getRent() {
        return 25;
    }



}
public class ResidencyTesting {
    public static void main(String args[]) {
        TwoBhkResidency tw1 = new TwoBhkResidency(1000,1000);
        System.out.println( tw1.getPriceOfResidency());

        LuxuriousResidency l1 = new LuxuriousResidency(5000);
        System.out.println(l1.getPriceOfResidency());
    }

}
```