ASSIGNMENT 4
2021KUCP1104

1 Write a Java program using the copy constructor to print the real and imaginary components of a complex number.

```
class Complex {
  private double real;
  private double imaginary;

  // Copy Constructor
  public Complex(Complex c) {
        this.real = c.real;
        this.imaginary = c.imaginary;
  }

  public void print() {
        System.out.println("Real: " + this.real);
        System.out.println("Imaginary: " + this.imaginary);
  }
}

public class Main {
  public static void main(String[] args) {
        Complex c1 = new Complex();
        c1.real = 3.0;
        c1.imaginary = 4.0;

        Complex c2 = new Complex(c1); // Creating a copy using the copy constructor

        c1.print();
        c2.print();
  }
}
```

2. Write a Java program to show Constructor Chaining 1. Within The Same Class. 2. From the base class.

```
// Constructor Chaining within the same class
class MyClass {
        private String name;
        private int age;
        private String address;

        // Default Constructor
```

```
    public MyClass() {
    this("John Doe", 18, "Unknown");
    }

    // Constructor with name and age parameters
    public MyClass(String name, int age) {
    this(name, age, "Unknown");
    }

    // Constructor with all parameters
    public MyClass(String name, int age, String address) {
    this.name = name;
    this.age = age;
    this.address = address;
    }

    public void display() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Address: " + address);
    }
}

// Constructor Chaining from the base class
class BaseClass {
    private int num;

    // Constructor with one parameter
    public BaseClass(int num) {
    this.num = num;
    }

    public void display() {
    System.out.println("Num: " + num);
    }
}

class DerivedClass extends BaseClass {
    private String name;

    // Constructor with two parameters
    public DerivedClass(int num, String name) {
    super(num);
    this.name = name;
    }

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

public class Main {
    public static void main(String[] args) {
    // Constructor Chaining within the same class
    MyClass obj1 = new MyClass();
    obj1.display();

    MyClass obj2 = new MyClass("Alice", 21);
    obj2.display();

    MyClass obj3 = new MyClass("Bob", 25, "123 Main St.");
    obj3.display();

    // Constructor Chaining from the base class
    DerivedClass obj4 = new DerivedClass(10, "John");
    obj4.display();
    }
}
```

3 Write java program to find area of square, area of rectangle and area of triangle using Heron’s Formula . Use Constructor Overloading .

```
class AreaCalculator {
    private double area;

    // Constructor Overloading
    public AreaCalculator(double side) {
    area = side * side; // Area of Square
    }

    public AreaCalculator(double length, double width) {
    area = length * width; // Area of Rectangle
    }

    public AreaCalculator(double a, double b, double c) {
    double s = (a + b + c) / 2;
    area = Math.sqrt(s * (s - a) * (s - b) * (s - c)); // Area of Triangle using Heron's Formula
    }
    public double getArea() {
    return area;
    }
}

public class Main {
    public static void main(String[] args) {
    // Finding the area of a square
    AreaCalculator square = new AreaCalculator(5);
    System.out.println("Area of square: " + square.getArea());

    // Finding the area of a rectangle
    AreaCalculator rectangle = new AreaCalculator(4, 6);
    System.out.println("Area of rectangle: " + rectangle.getArea());

    // Finding the area of a triangle
    AreaCalculator triangle = new AreaCalculator(3, 4, 5);
    System.out.println("Area of triangle: " + triangle.getArea());
    }
}
```

4. Write java program to calculate area of rectangle using interface.

```
interface Shape {
    public double getArea();
}

class Rectangle implements Shape {
    private double length, width;

    public Rectangle(double length, double width) {
    this.length = length;
    this.width = width;
    }

    @Override
    public double getArea() {
    return length * width;
    }
}

public class Main {
    public static void main(String[] args) {
    // Creating a rectangle object
    Shape rectangle = new Rectangle(4.0, 5.0);

    // Calculating the area of the rectangle using the interface
    double area = rectangle.getArea();

    // Displaying the area of the rectangle
    System.out.println("Area of Rectangle: " + area);
    }
}
```

5 WAP to show how the class Demo implements interface Inf2, Also show how the class will provide the implementation of all the methods of interface Inf1 as well, because interface Inf2 extends Inf1.

```
public interface Inf1 {
    void method1();
    void method2();
}

public interface Inf2 extends Inf1 {
    void method3();
}

public class Demo implements Inf2 {
    public void method1() {
    // Implementation of method1
    }

    public void method2() {
    // Implementation of method2
    }

    public void method3() {
    // Implementation of method3
    }

    // Other methods of the Demo class
}
```