13.Crear la subclase Circulo que hereda de FiguraGeometrica y debes implementarla considerando a FiguraGeometrica una superclase. Realiza las siguientes instrucciones sobre
la clase Circulo:

```
public class Circulo extends FiguraGeometrica {
    private Punto centro;
    private int radio;

    public Circulo(String nombre, Punto centro, int radio) {
        super(nombre);
        this.centro = centro;
        this.radio = radio;
    }

    public double calcularArea() {
        return Math.PI * Math.pow(radio, 2);
    }

    public boolean esRegular() {
        return true; // Un círculo siempre es una figura regular.
    }

    @Override
    public String toString() {
        return "Nombre: " + getNombre() + ", Centro: " + centro.toString() + ", Radio: " + radio;
    }

    // Métodos de acceso y manipulación para centro y radio
    public Punto getCentro() {
        return centro;
    }

    public void setCentro(Punto centro) {
        this.centro = centro;
    }

    public int getRadio() {
        return radio;
    }

    public void setRadio(int radio) {
        this.radio = radio;
    }
}
```

14.Crear la subclase Cuadrilatero que hereda de FiguraGeometrica. Sin embargo, la clase Cuadrilatero es una clase abstracta ya que no es posible implementar el método que calcula el área sin conocer el cuadrilátero concreto. Realiza las siguientes instrucciones sobre la clase Cuadrilatero:

```
public abstract class Cuadrilatero extends FiguraGeometrica {
    private Punto vertice1;
    private Punto vertice2;
    private Punto vertice3;
    private Punto vertice4;

    public Cuadrilatero(String nombre, Punto vertice1, Punto vertice2, Punto vertice3, Punto
vertice4) {
        super(nombre);
        this.vertice1 = vertice1;
        this.vertice2 = vertice2;
        this.vertice3 = vertice3;
        this.vertice4 = vertice4;
    }

    public boolean esRegular() {
        return checkRectangulo();
    }

    @Override
    public String toString() {
        return "Nombre: " + getNombre() + ", Vértices: [" + vertice1.toString() + ", " +
vertice2.toString() + ", " +
                vertice3.toString() + ", " + vertice4.toString() + "]";
    }

    // Métodos de acceso y manipulación para los vértices
    public Punto getVertice1() {
        return vertice1;
    }

    public void setVertice1(Punto vertice1) {
        this.vertice1 = vertice1;
    }

    public Punto getVertice2() {
        return vertice2;
    }

    public void setVertice2(Punto vertice2) {
        this.vertice2 = vertice2;
    }
```

```
    public Punto getVertice3() {
        return vertice3;
    }

    public void setVertice3(Punto vertice3) {
        this.vertice3 = vertice3;
    }

    public Punto getVertice4() {
        return vertice4;
    }

    public void setVertice4(Punto vertice4) {
        this.vertice4 = vertice4;
    }

    // Métodos privados para verificar si el cuadrilátero es un rectángulo o cuadrado
    private boolean checkRectangulo() {
        return escalarProducto(vertice2, vertice1, vertice3) == 0 &&
                escalarProducto(vertice3, vertice2, vertice4) == 0 &&
                escalarProducto(vertice4, vertice3, vertice1) == 0;
    }

    private int escalarProducto(Punto a, Punto b, Punto c) {
        int x1 = b.getX() - a.getX();
        int y1 = b.getY() - a.getY();
        int x2 = c.getX() - a.getX();
        int y2 = c.getY() - a.getY();
        return x1 * x2 + y1 * y2;
    }
}
```

15.La clase Rectangulo hereda de Cuadrilatero. Realiza las siguientes acciones sobre la clase
Rectangulo:

Primero, agregamos el método calcularArea() en la clase Rectangulo:

```
public class Rectangulo extends Cuadrilatero {
    public Rectangulo(String nombre, Punto vert1, Punto vert2, Punto vert3, Punto vert4) {
        super(nombre, vert1, vert2, vert3, vert4);

        if (!esRegular()) {
            System.out.println("El rectángulo no es un cuadrilátero regular.");
        }
```

```
    }

    public double calcularArea() {
        double base = Math.sqrt(Math.pow(getVertice1().getX() - getVertice2().getX(), 2) +
                                Math.pow(getVertice1().getY() - getVertice2().getY(), 2));
        double altura = Math.sqrt(Math.pow(getVertice1().getX() - getVertice4().getX(), 2) +
                                  Math.pow(getVertice1().getY() - getVertice4().getY(), 2));
        return base * altura;
    }

    @Override
    public String toString() {
        return "Nombre: " + getNombre() + ", Vértices: [" + getVertice1().toString() + ", " +
                getVertice2().toString() + ", " + getVertice3().toString() + ", " +
getVertice4().toString() + "]";
    }
}
```

Luego, agregamos el método distancia en la clase Punto para calcular la distancia entre dos puntos:

```
public class Punto {
    private int x;
    private int y;

    public Punto(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public double distancia(Punto otroPunto) {
        int deltaX = this.x - otroPunto.x;
        int deltaY = this.y - otroPunto.y;
        return Math.sqrt(deltaX * deltaX + deltaY * deltaY);
    }

    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
```

```
}
```

16.Crea una interface Shape que contenga el siguiente método: obtenerPerimetro() y que cada
subclase Triágunlo, Círculo y Rectángulo.

```
// Definición de la interfaz Shape
public interface Shape {
    double obtenerPerimetro();
}

// Implementación de la interfaz para la subclase Triangulo
public class Triangulo implements Shape {
    private double lado1;
    private double lado2;
    private double lado3;

    public Triangulo(double lado1, double lado2, double lado3) {
        this.lado1 = lado1;
        this.lado2 = lado2;
        this.lado3 = lado3;
    }

    @Override
    public double obtenerPerimetro() {
        return lado1 + lado2 + lado3;
    }
}

// Implementación de la interfaz para la subclase Circulo
public class Circulo implements Shape {
    private double radio;

    public Circulo(double radio) {
        this.radio = radio;
    }

    @Override
    public double obtenerPerimetro() {
        return 2 * Math.PI * radio;
    }
```

```
}

// Implementación de la interfaz para la subclase Rectangulo
public class Rectangulo implements Shape {
    private double base;
    private double altura;

    public Rectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

    @Override
    public double obtenerPerimetro() {
        return 2 * (base + altura);
    }
}
```

17. Al objeto instanciado de la clase SuperficiePlana agregar las siguientes figuras geométricas: Círculo y Rectángulo. Previamente, se debió agregar Triángulo.

```
import java.util.ArrayList;
import java.util.List;

public class SuperficiePlana {
    private List<Shape> figuras;

    public SuperficiePlana() {
        figuras = new ArrayList<>();
    }

    public void agregarFigura(Shape figura) {
        figuras.add(figura);
    }

    public double calcularPerimetroTotal() {
        double perimetroTotal = 0;
        for (Shape figura : figuras) {
            perimetroTotal += figura.obtenerPerimetro();
        }
        return perimetroTotal;
    }
}
```

—----------

```
public class Main {
    public static void main(String[] args) {
        // Crear una instancia de SuperficiePlana
        SuperficiePlana superficie = new SuperficiePlana();

        // Agregar un Triangulo
        Triangulo triangulo = new Triangulo(3, 4, 5);
        superficie.agregarFigura(triangulo);

        // Agregar un Circulo
        Circulo circulo = new Circulo(5);
        superficie.agregarFigura(circulo);

        // Agregar un Rectangulo
        Rectangulo rectangulo = new Rectangulo(4, 6);
        superficie.agregarFigura(rectangulo);

        // Calcular el perímetro total
        double perimetroTotal = superficie.calcularPerimetroTotal();
        System.out.println("El perímetro total de la superficie es: " + perimetroTotal);
    }
}
```

18.Imprime el resultado del calculo del area total de las figuras y también la información sobre las
figuras (área, toString, perímetro y sus vértices).

```
public class Main {
    public static void main(String[] args) {
        // Crear una instancia de SuperficiePlana
        SuperficiePlana superficie = new SuperficiePlana();

        // Agregar un Triangulo
        Triangulo triangulo = new Triangulo(3, 4, 5);
        superficie.agregarFigura(triangulo);

        // Agregar un Circulo
        Circulo circulo = new Circulo(5);
        superficie.agregarFigura(circulo);

        // Agregar un Rectangulo
        Rectangulo rectangulo = new Rectangulo(4, 6);
        superficie.agregarFigura(rectangulo);

        // Calcular el área total y mostrar información de las figuras
```

```
        double areaTotal = calcularAreaTotal(superficie);
        System.out.println("El área total de la superficie es: " + areaTotal);

        mostrarInformacionFiguras(superficie);
    }

    // Método para calcular el área total de las figuras
    public static double calcularAreaTotal(SuperficiePlana superficie) {
        double areaTotal = 0;
        for (Shape figura : superficie.getFiguras()) {
            if (figura instanceof FiguraConArea) {
                areaTotal += ((FiguraConArea) figura).calcularArea();
            }
        }
        return areaTotal;
    }

    // Método para mostrar información de las figuras
    public static void mostrarInformacionFiguras(SuperficiePlana superficie) {
        System.out.println("Información de las figuras:");
        for (Shape figura : superficie.getFiguras()) {
            System.out.println("====================================");
            System.out.println("Tipo de figura: " + figura.getClass().getSimpleName());
            System.out.println("Área: " + ((FiguraConArea) figura).calcularArea());
            System.out.println("Perímetro: " + figura.obtenerPerimetro());
            System.out.println("Vértices: " + figura.toString());
        }
    }
}
```