Ejercicio Herencia GeometricObject, Circle y Rectangle

Ingeniería Informática

Joselyn Gallegos Abreo

Expediente 285802

2025-1

- **Código**

```java
package prueba;


import java.util.Date;

public class GeometricObject {

 private String color;
 private boolean filled;
 private Date dateCreated;

 public GeometricObject() {
     this.color = "white";
     this.filled = false;
     this.dateCreated = new Date();
 }

 public GeometricObject(String color, boolean filled) {
     this.color = color;
     this.filled = filled;
     this.dateCreated = new Date();
 }

 public String getColor() {
     return color;
 }

 public void setColor(String color) {
     this.color = color;
 }

 public boolean isFilled() {
     return filled;
 }

 public void setFilled(boolean filled) {
     this.filled = filled;
 }

 public Date getDateCreated() {
return dateCreated;
}
```

```
public String toString() {
return "Created on " + dateCreated + "\ncolor: " + color + " and filled: " + filled;
}

public static void main(String[] args)
{
Circle circle1 =new Circle();
System.out.println("Circle:");
System.out.println("Color: " + circle1.getColor());
System.out.println("Filled: " + circle1.isFilled());
System.out.println("Radius: " + circle1.getRadius());
System.out.println("Area: " + circle1.getArea());
System.out.println("Perimeter: " + circle1.getPerimeter());
System.out.println();
Circle circle = new Circle(5, "blue", true);
System.out.println("Circle:");
System.out.println("Color: " + circle.getColor());
System.out.println("Filled: " + circle.isFilled());
System.out.println("Radius: " + circle.getRadius());
System.out.println("Area: " + circle.getArea());
System.out.println("Perimeter: " + circle.getPerimeter());
System.out.println();
Rectangle rectangle1 = new Rectangle();
System.out.println("Rectangle:");
System.out.println("Color: " + rectangle1.getColor());
System.out.println("Filled: " + rectangle1.isFilled());
System.out.println("Width: " + rectangle1.getWidth());
System.out.println("Height: " + rectangle1.getHeight());
System.out.println("Area: " + rectangle1.getArea());
System.out.println("Perimeter: " + rectangle1.getPerimeter());
Rectangle rectangle = new Rectangle(4, 7, "red", false);
System.out.println("Rectangle:");
System.out.println("Color: " + rectangle.getColor());
System.out.println("Filled: " + rectangle.isFilled());
System.out.println("Width: " + rectangle.getWidth());
System.out.println("Height: " + rectangle.getHeight());
System.out.println("Area: " + rectangle.getArea());
System.out.println("Perimeter: " + rectangle.getPerimeter());
}
}
```

```
package prueba;

public class Circle extends GeometricObject {
private double radius;

public Circle() {
this.radius = 1.0;
}

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

public Circle(double radius, String color, boolean filled) {
```

```
super(color, filled);
this.radius = radius;
}

public double getRadius() {
return radius;
}

public void setRadius(double radius) {
this.radius = radius;
}

public double getArea() {
return Math.PI * radius * radius;
}

public double getPerimeter() {
return 2 * Math.PI * radius;
}

public double getDiameter() {
return 2 * radius;
}

public void printCircle() {
System.out.println("The circle is created " + getDateCreated() + " and the radius is " +
radius);
}
}
```

```
package prueba;
public class Rectangle extends GeometricObject {
    private double width;
    private double height;

    public Rectangle() {
        this.width = 1.0;
        this.height = 1.0;
    }

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

    public Rectangle(double width, double height, String color, boolean filled) {
        super(color, filled);
        this.width = width;
        this.height = height;
    }

    public double getWidth() {
        return width;
    }
```

```
    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public double getArea() {
        return width * height;
    }

    public double getPerimeter() {
        return 2 * (width + height);
    }
}
```

- **Conceptos**

1. **Clase:** una clase es como un molde o plantilla que define qué características y funciones va a tener un tipo de objeto. En este caso, GeometricObject, Circle y Rectangle son clases que establecen qué propiedades (como color, radio o ancho) y qué métodos tendrán.
2. **Objeto:** un objeto es algo que creamos a partir de una clase. Es una entidad real que tiene valores específicos. Por ejemplo, Circle circle1 = new Circle(5, "blue", true); crea un objeto circle1 con un radio de 5, color azul y que está lleno.
3. **Instancia:** una instancia es simplemente el proceso de crear un objeto a partir de una clase usando new.
4. **Diagrama UML de Clase**: es como un esquema que ayuda a visualizar cómo se relacionan las clases en un programa. En la imagen que compartí, se ve que GeometricObject es la clase principal y Circle y Rectangle heredan de ella, lo que ayuda a organizar el código de forma más estructurada.
5. **Herencia:** la herencia es cuando una clase obtiene las características de otra. En este código, Circle y Rectangle heredan de GeometricObject, lo que significa que no necesitan definir de nuevo los atributos color y filled, porque ya los heredan de su clase padre. Así se evita repetir código innecesario.
6. **Polimorfismo: e**s cuando un mismo método se comporta de manera diferente dependiendo de la clase que lo use. En este caso, aunque Circle y Rectangle tienen el método getArea(), su implementación es distinta en cada clase porque el área se calcula de manera diferente para un círculo que para un rectángulo.
7. **Modificadores de Visibilidad: s**on reglas que indican qué partes del código pueden acceder a ciertos atributos o métodos. Por ejemplo, los atributos private en GeometricObject (color y filled) solo pueden ser modificados a través de sus métodos setColor() y setFilled(), asegurando que no se cambien de manera incorrecta.
8. **Encapsulamiento:** es la práctica de proteger los datos de una clase y permitir que solo sean modificados de forma controlada. En el código, esto se hace con los atributos private, los cuales solo pueden cambiarse usando métodos public, como setRadius() en Circle. Así se evita que el usuario del código pueda alterar valores directamente y causar errores no esperados.