Mubeen Khalid (SP21-BSE-015)

design patterns

Lab Assignment

# Source Code:

## Shape:

public interface Shape {

    void draw();

    double computeCost();

}

## Circle:

public class Circle implements Shape {

    private double price;

    public Circle(double price) {

        this.price = price;

    }

    @Override

    public void draw() {

        System.out.println("Shape: Circle | Cost without border: " + computeCost());

    }

    @Override

    public double computeCost() {

        return price;

    }

}

## Rectangle:

public class Rectangle implements Shape {

    private double price;

    public Rectangle(double price) {

        this.price = price;

    }

    @Override

    public void draw() {

        System.out.println("Shape: Rectangle | Cost without border: " + computeCost());

    }

    @Override

    public double computeCost() {

        return price;

    }

}

## ShapeDecorator:

public abstract class ShapeDecorator implements Shape {

    protected Shape decoratedShape;

    protected double price;

    public ShapeDecorator(Shape decoratedShape, double price) {

        this.decoratedShape = decoratedShape;

        this.price = price;

    }

    @Override

    public void draw() {

        decoratedShape.draw();

    }

    @Override

    public double computeCost() {

        return decoratedShape.computeCost() + price;

    }

}

## RedShapeDecorator:

public class RedShapeDecorator extends ShapeDecorator {

    public RedShapeDecorator(Shape decoratedShape, double price) {

        super(decoratedShape, price);

    }

    @Override

    public void draw() {

        decoratedShape.draw();

        setRedBorder(decoratedShape);

        System.out.println("Total Cost with Red Border: " + computeCost());

    }

    private void setRedBorder(Shape decoratedShape) {

        System.out.println("Border Color: Red");

    }

}

## Demo:

public class DecoratorPatternDemo {

    public static void main(String[] args) {

        Shape circle = new Circle(20.0);

        Shape rectangle = new Circle(25.0);

        Shape redCircle = new RedShapeDecorator(circle, 5.0);

        Shape redRectangle = new RedShapeDecorator(rectangle, 7.0);

        System.out.println("Circle with normal border");

        circle.draw();

        System.out.println("Rectangle with normal border");

        rectangle.draw();

        System.out.println("\nCircle with border");

        redCircle.draw();

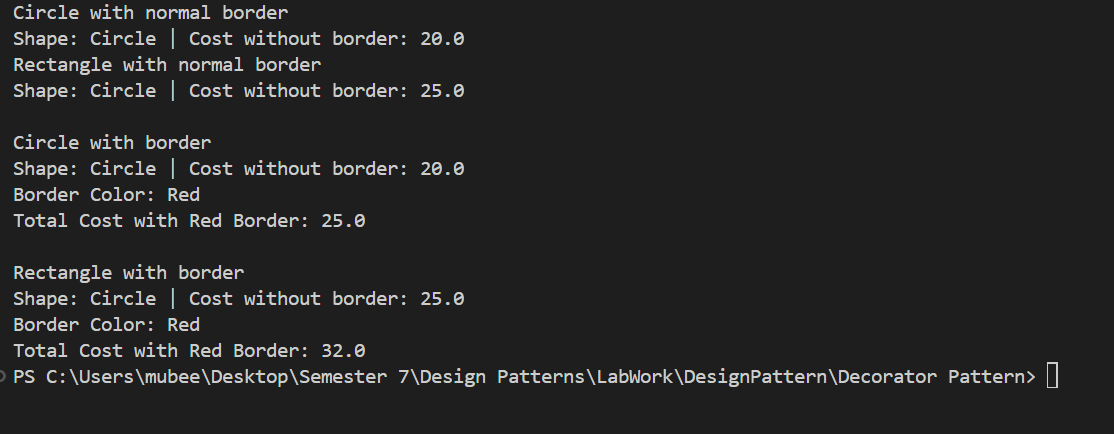
        System.out.println("\nRectangle with border");

        redRectangle.draw();

    }

}

## Output:



## Git hub Link:

[DesignPattern/Decorator Pattern at main · mubeen070/DesignPattern (github.com)](https://github.com/mubeen070/DesignPattern/tree/main/Decorator%20Pattern)