

# SA第十五次作业

## 1、什么是透明装饰模式，什么是半透明装饰模式？请举例说明。

答：

(1)**透明装饰模式**：在透明装饰模式中，装饰器和被装饰对象的接口是相同的，这意味着装饰器和被装饰对象可以互换使用，客户端无需知道装饰器的存在。透明装饰模式的关键在于装饰器类要继承被装饰对象的接口，并且在其中保存一个被装饰对象的引用，以便将请求转发给被装饰对象，同时可以在请求前后添加额外的行为。

举例来说，假设有一个简单的接口Shape，包含一个方法draw()。

```
public interface Shape {  
    void draw();  
}
```

有一个具体的类Circle实现了这个接口。

```
public class Circle implements Shape{  
    public void draw() {  
        System.out.println("Shape: Circle");  
    }  
}
```

现在想要给Circle添加一些额外的功能，比如颜色，我们可以创建一个ColorDecorator类，实现Shape接口，并在其中保存一个Shape对象的引用，然后在draw()方法中先设置颜色，再调用被装饰对象的draw()方法。

```
public class ColorDecorator implements Shape{  
  
    protected Shape decoratedShape;  
  
    public ColorDecorator(Shape decoratedShape) {  
        this.decoratedShape = decoratedShape;  
    }  
  
    public void draw() {  
        decoratedShape.draw();  
        setColor();  
    }  
  
    private void setColor() {  
        System.out.println("Color: Red");  
    }  
}
```

对于透明装饰模式，装饰器对于客户端来说透明，客户端不应该声明装饰器变量(ColorDecorator)，而应该声明的是被装饰的对象(Shape)。

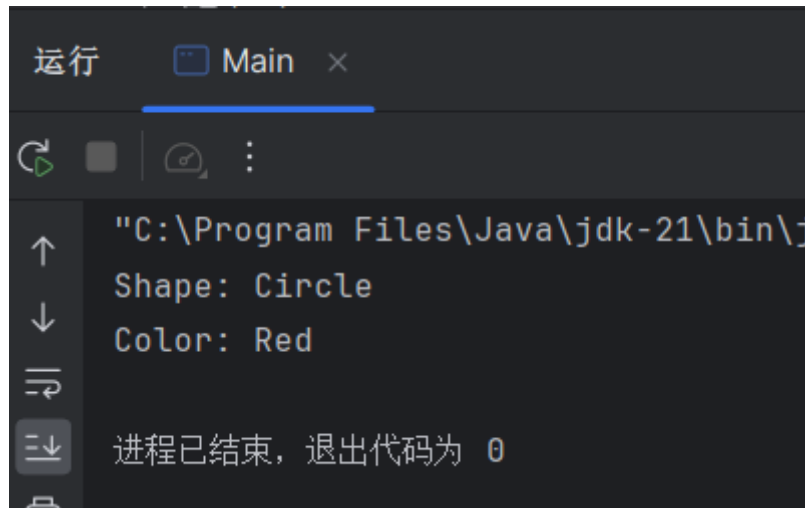
```

public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle();
        // 使用装饰器给 Circle 添加颜色
        Shape redCircle = new ColorDecorator(circle);

        // 绘制红色的圆形
        redCircle.draw();
    }
}

```

运行结果：



```

运行 Main
"C:\Program Files\Java\jdk-21\bin\java.exe"
Shape: Circle
Color: Red
进程已结束，退出代码为 0

```

(2)**半透明装饰模式**：半透明装饰模式中，装饰器和被装饰对象的接口不一定相同，装饰器可能会添加自己的方法，而客户端可能需要使用这些方法。但是，客户端需要知道装饰器的存在，因为它们不能直接替代被装饰对象。

继续上面的例子，假设现在需要为Circle添加一个resize()方法来调整其大小，我们可以创建一个ResizableDecorator类来实现这个功能。

```

public class ResizableDecorator implements Shape{
    protected Shape decoratedShape;

    public ResizableDecorator(Shape decoratedShape) {
        this.decoratedShape = decoratedShape;
    }

    public void draw() {
        decoratedShape.draw();
        resize();
    }

    public void resize() {
        System.out.println("Resize Circle");
    }
}

```

但是客户端需要知道，如果要调整大小，必须使用ResizableDecorator，而不能直接使用Circle。这就是半透明装饰模式，因为客户端需要知道装饰器的存在和如何使用它们。

```
public class Main {  
    public static void main(String[] args) {  
        Shape circle = new Circle();  
        // 使用装饰器给 Circle 添加可调整大小的功能  
        ResizableDecorator resizableCircle = new ResizableDecorator(circle);  
  
        // 绘制并调整大小  
        resizableCircle.draw();  
    }  
}
```

运行结果：



```
运行  Main ×  
↑ "C:\Program Files\Java\jdk-21\bin\java.exe"  
↓ Shape: Circle  
⇌ Resize Circle  
⇓ 进程已结束，退出代码为 0  
🖨
```