**装饰器模式**

1. **介绍：**

装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。

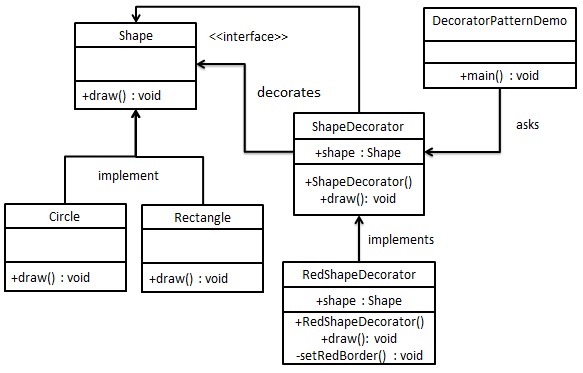
这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法完整性的前提下，提供了额外的功能。

1. **实现及类图演示：**

我们将创建一个 *Shape* 接口和实现了 *Shape* 接口的实体类。然后我们创建一个实现了 *Shape* 接口的抽象装饰类 *ShapeDecorator*，并把 *Shape* 对象作为它的实例变量。

*RedShapeDecorator* 是实现了 *ShapeDecorator* 的实体类。

*DecoratorPatternDemo*，我们的演示类使用 *RedShapeDecorator* 来装饰 *Shape* 对象。

****

1. **我的理解：**

在我看来，装饰器模式与老师上课讲解的代理（proxy）十分相似，因为使用任一方法都可以在已存在的对象方法中假如自己定义的方法，不同的是，装饰器模式中Decorator和Decorate都各自实现一个借口，而代理模式中代理类和真实处理的类都实现了同一个接口。