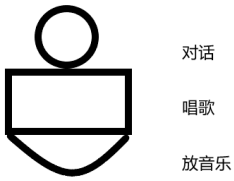
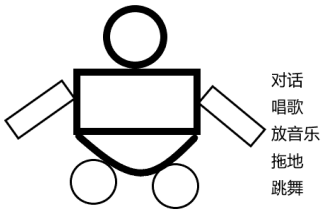
装饰器模式

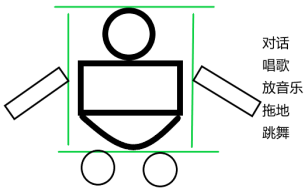
定义

指在不改变现有对象结构的情况下，动态地给该对象增加一些职责（即增加其额外功能）的模式，它属于

对象结构型模式。







特点

优点

装饰器时继承的有力补充，比继承灵活，在不改变现有对象的情况下，动态的给对象增加一些额外的功能，　　即插即用。

装饰器模式完全遵守开闭原则

缺点

装饰器模式会增加许多子类，过度使用会增加程序的复杂性

结构

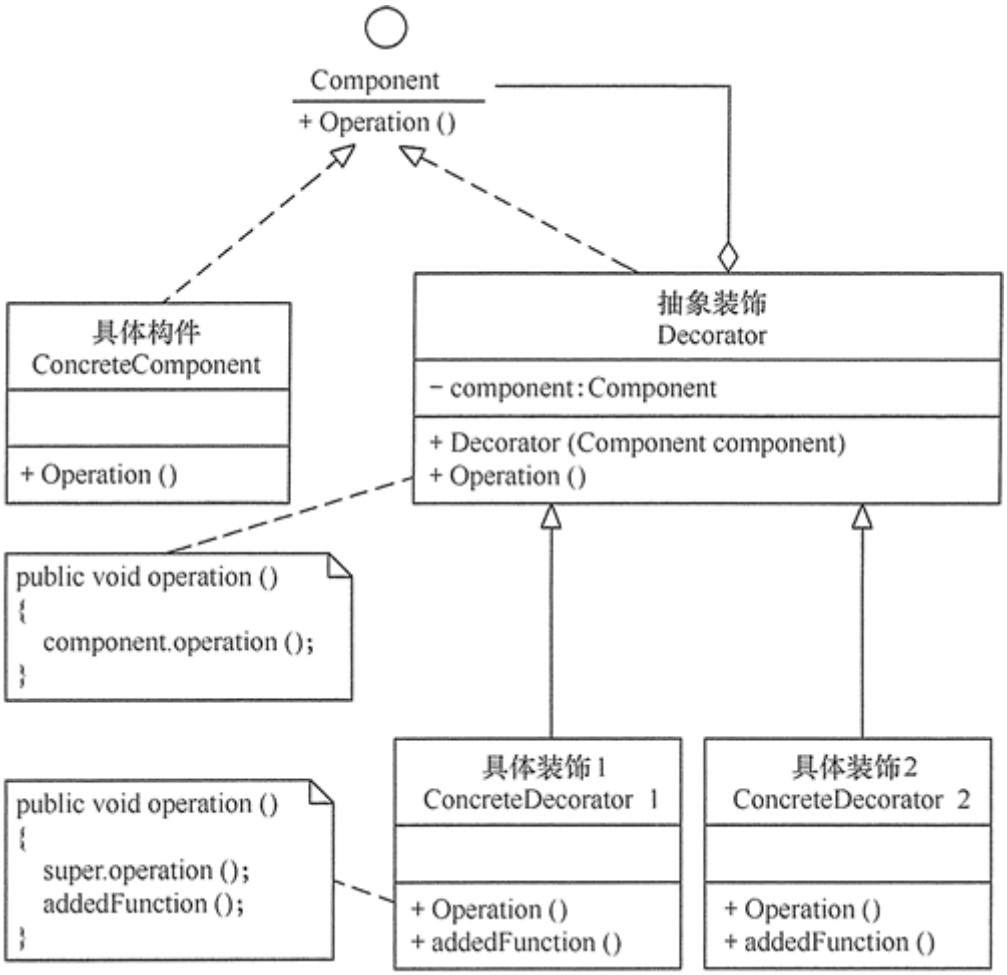
装饰器主要包含以下角色

1.抽象构建(Component)角色：定义一个抽象接口以规范准备接收附加责任的对象

2.具体构建(ConcreteComponent)角色：实现抽象构建，通过装饰角色为其添加一些职责

3.抽象装饰(Decorator)角色：继承抽象构件，并包含具体构件的实例，可以通过其子类扩展具体构件的功 能。

4.具体装饰(ConcreteDecorator)角色：实现抽象装饰的相关方法，并给具体构建对象添加附加的责任



实现

```
package decorator;

public class DecoratorPattern {
    public static void main(String[] args) {
        new RobotDecorator(new FistRobot()).doMorething;;
    }
}

//抽象构件角色(机器人接口)
interface Roboot {
    void doSomething();
}

//具体构件角色("第一代机器人")
class FistRoboot implements Roboot {
    @Override
```

```
17        public void doSomething() {
18            System.out.println("对话");
19            System.out.println("唱歌");
20            System.out.println("放音乐");
21        }
22    }
23
24    //抽象装饰角色(第二代机器人)
25    class RobootDecorator implements Robot {
26        private Robot robot;
27
28        public RobootDecorator(Roboot robot) {
29            this.robot = robot;
30        }
31        @Override
32        public void doSomething() {
33            robot.doSomething();
34        }
35        public void doMorething(){
36            robot.doSomething();
37            System.out.println("跳舞");
38            System.out.println("拖地");
39
40        }
41    }
```

```
1    运行结果：
2        对话
3        唱歌
4        放音乐
5        跳舞
6        拖地
```

装饰器模式的应用场景

装饰器通常在以下情况下使用:

当需要给一个现有类添加附加职责，又不能采用生成子类的方法进行扩充时.例如：该类被隐藏或者该类是终极类或者采用继承方式会产生大量的子类.

当需要通过对现有的一组基本功能进行排列组合而产生非常多的功能时，采用继承关系很难实现，而采用装饰器模式却很好实现。

当对象的功能要求可以动态地添加，也可以再动态地撤销时。

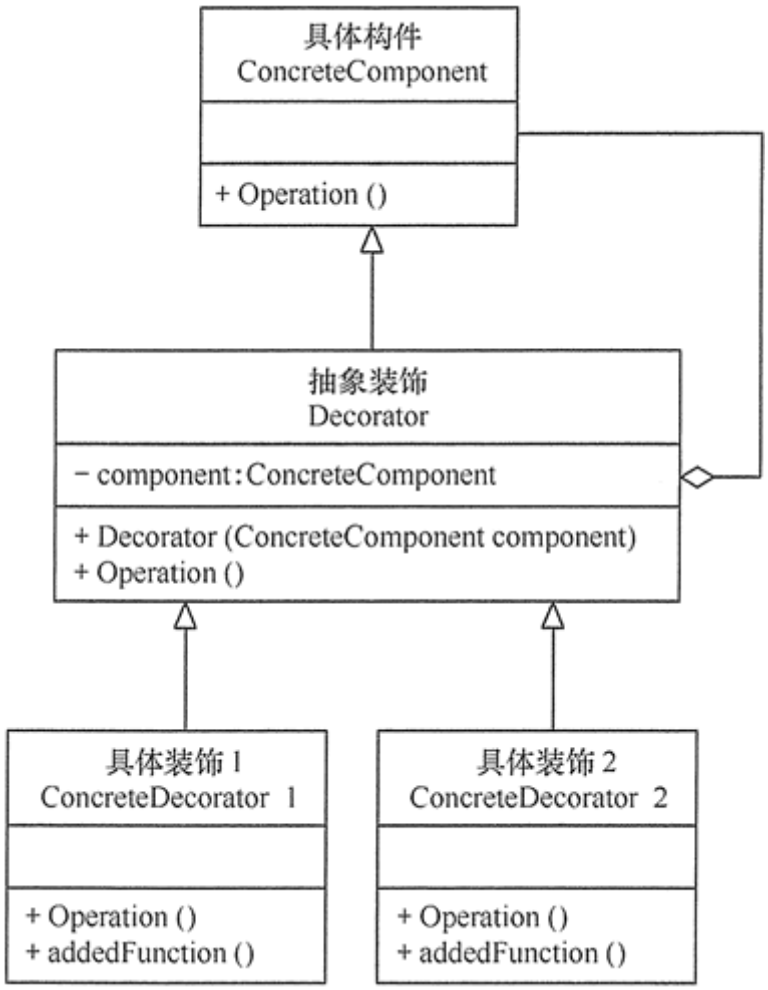
装饰器模式在Java语言中的最著名的应用莫过于Java I/O标准库的设计了，例如：InputStream 的子类 FilterInputStream，OutputStream 的子类 FilterOutputStream，Reader 的子类 BufferedReader 以及 FilterReader，还有 Writer 的子类 BufferedWriter、FilterWriter 以及 PrintWriter 等，它们都是抽象装饰类。

装饰器模式的扩展

装饰器模式所包含的4个角色不是任何时候都要存在的，在有些应用环境下是可以简化的

情况一

如果有一个具体构件而没有抽象构件时，可以让抽象装饰具体构建



情况二

如果只有一个具体装饰时，可以将抽象装饰和具体装饰合并