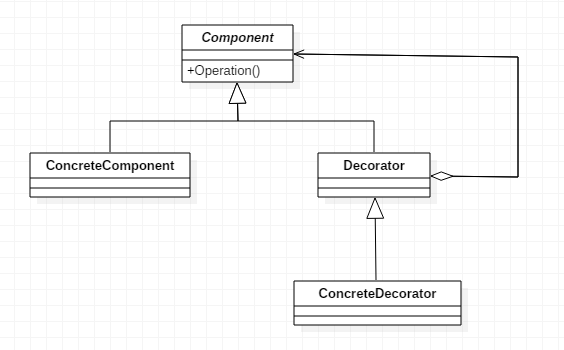
## 1.什么是装饰模式

**装饰模式（Decorator）**，动态地给一个对象添加一些额外的职责，就增加功能来说，装饰模式比生成子类更灵活。UML结构图如下：



## 2. 装饰模式的应用

### 2.1. 何时使用

* 在不想增加很多子类的情况下扩展类时

### 2.2. 方法

* 将具体功能职责划分，同时继承装饰者模式

### 2.3. 优点

* 装饰类和被装饰类可以独立发展，而不会相互耦合。它有效地把类的核心职责和装饰功能分开了
* 装饰模式是继承关系的一个替代方案
* 装饰模式可以动态地扩展一个实现类的功能

### 2.4. 缺点

* 多层装饰比较复杂。比如我们现在有很多层装饰，出了问题，一层一层检查，最后发现是最里层的装饰出问题了，想想工作量都害怕

### 2.5. 使用场景

* 需要扩展一个类的功能时
* 需要动态地给一个对象增加功能，并可以动态地撤销时
* 需要为一批的兄弟类进行改装或加装功能时

### 2.6. 应用实例

* 旧机包装成新机，手机/电脑内部配件不变，只是换个外壳
* 换衣小游戏，人还是那个人，不断给她换衣服，还可以一层套一层的
* 孙悟空有72变，变成什么后就有了它的功能，但本质还是一只猴子

## 3. 装饰模式的实现

下面我们看一个例子，我们就以上面说的换装为例。我们先分析一下，换装需要有一个人类用于指定是谁换装、一个服装类为具体服装类的父类、以及服装类下的各种具体服装的类。UML图如下：

