**一、 装饰（Decorator）模式**

装饰（Decorator）模式又名包装（Wrapper）模式[GOF95]。装饰模式以对客户端透明的方式扩展对象的功能，是继承关系的一个替代方案。

**引言**

孙悟空有七十二般变化，他的每一种变化都给他带来一种附加的本领。他变成鱼儿时，就可以到水里游泳；他变成雀儿时，就可以在天上飞行。而不管悟空怎么变化，在二郎神眼里，他永远是那只猢狲。

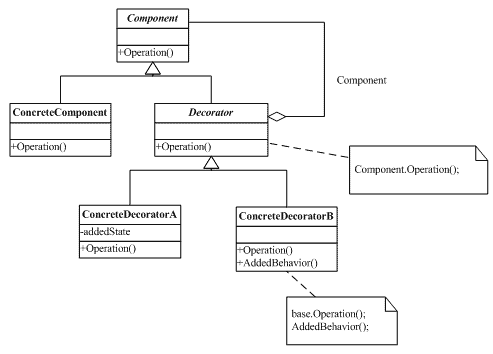
装饰模式以对客户透明的方式动态地给一个对象附加上更多的责任。换言之，客户端并不会觉得对象在装饰前和装饰后有什么不同。装饰模式可以在不使用创造更多子类的情况下，将对象的功能加以扩展。

**二、 装饰模式的结构**

装饰模式使用原来被装饰的类的一个子类的实例，把客户端的调用委派到被装饰类。装饰模式的关键在于这种扩展是完全透明的。

在孙猴子的例子里，老孙变成的鱼儿相当于老孙的子类，这条鱼儿与外界的互动要通过"委派"，交给老孙的本尊，由老孙本尊采取行动。

装饰模式的类图如下图所示：



在装饰模式中的各个角色有：

* 抽象构件（Component）角色：给出一个抽象接口，以规范准备接收附加责任的对象。
* 具体构件（Concrete Component）角色：定义一个将要接收附加责任的类。
* 装饰（Decorator）角色：持有一个构件（Component）对象的实例，并定义一个与抽象构件接口一致的接口。
* 具体装饰（Concrete Decorator）角色：负责给构件对象"贴上"附加的责任。

功能，能够实现动态的为对象添加功能，是从一个对象外部来给对象增加功能，相当于改变了对象的外观。当装饰过后，从外部使用系统的角度看，就不在是使用原始的对象，而是使用被一些列的装饰过后的对象。

能够灵活的改变一个对象的功能，只要动态组合的装饰器发生了改变，那么最终得到的对象的功能也就发生了改变。

变相的另一个好处，装饰器功能复用，可以给一个对象多次增加同一个装饰器，也可以用同一个装饰器装饰不同的对象。

对象组合：

对一个类的扩展有两种方式，继承，或功能更加强大更加灵活的对象组合的方式。

面向对象设计中，基本原则”尽量使用对象组合，而不是对象继承”用来扩展和复用功能。这就是装饰模式的思考起点规则。