**装饰模式又名包装(Wrapper)模式。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。**

装饰模式特点：

1装饰对象和真实对象有相同的接口。

2装饰对象包含真实对象的引用。

3装饰对象接受客户端的请求，并把这些请求转发给真实对象

4装饰对象在转发请求前后，可以增加新的功能

5装饰模式是对象的继承

**package** cn.decorator;

/\*\*

\*

\* 抽象构件角色

\* **@author** Administrator

\*

\*/

**public** **interface** Component {

**void** doSomething();

}

**package** cn.decorator;

/\*\*

\* 具体构建角色

\* **@author** Administrator

\*

\*/

**public** **class** ConcreteComponent **implements** Component {

@Override

**public** **void** doSomething() {

// **TODO** Auto-generated method stub

System.*out*.println("a");

}

}

**package** cn.decorator;

/\*\*

\* 装饰角色

\* **@author** Administrator

\*

\*/

**public** **class** Decorator **implements** Component {

Component component;

**public** Decorator(Component component){

**this**.component = component;

}

@Override

**public** **void** doSomething() {

// **TODO** Auto-generated method stub

component.doSomething();

}

}

**package** cn.decorator;

/\*\*

\* 具体装饰角色

\* **@author** Administrator

\*

\*/

**public** **class** ConcreteDecorator1 **extends** Decorator{

**public** ConcreteDecorator1(Component component) {

**super**(component);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** doSomething() {

// **TODO** Auto-generated method stub

**super**.doSomething();

doAnotherThing();

}

**public** **void** doAnotherThing(){

System.*out*.println("b");

}

}

**package** cn.decorator;

**public** **class** ConcreteDecorator2 **extends** Decorator{

**public** ConcreteDecorator2(Component component) {

**super**(component);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** doSomething() {

// **TODO** Auto-generated method stub

**super**.doSomething();

doAnotherThing();

}

**public** **void** doAnotherThing(){

System.*out*.println("c");

}

}

**package** cn.decorator;

**public** **class** Test {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Component component = **new** ConcreteComponent();

Decorator decorate1 = **new** ConcreteDecorator1(component);

Decorator decorate2 = **new** ConcreteDecorator2(decorate1);

decorate2.doSomething();

}

1. 装饰模式是用来增强对象的功能，包括增强一个方法的内容，而且可以增加其他的方法。
2. 装饰模式：对象继承的一种方法，比普通继承强大，在一定程度上，解决了java单继承的的缺点。一般，我想增强某个对象的功能，可以自定义一个对象继承该对象，如果该对象是一个接口类型，可以找对象的子对象，在继承子对象。如果二个对象没有继承关系，可以让他们继承同一类或接口，在一个对象引用一个要增强的对象。这样可以增强该对象方法的，改方法不但具有本对象的功能，而且具有被增强对象的功能。而且被增强对象可以通过同样的方式，具有该对象的功能。当一个对象是接口时，而且没有子类，可以动态代理来增强该对象的功能。当增强的对象不能增加方法时，不适合动态代理模式。用装饰模式。