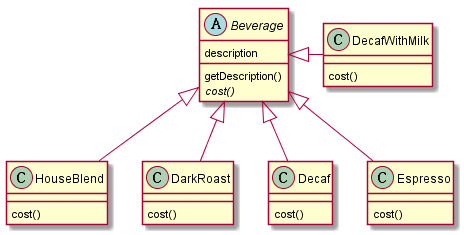
### 第三章：装饰者模式

|  |
| --- |
| ***设计原则5——类应该对扩展开放，对修改关闭*** |

我们的目标是允许类容易扩展，在不修改现有代码的情况下，就可搭配新的行为。如能实现这一的目标，就可以接受新的功能来应对改变的需求。

案例：咖啡店订单系统



针对实现编程：

使用继承，不管咖啡中加入了什么调料都是一个新的类对象，这就造成了“类爆炸”。

这里完全看不到复用， 而当我们组合调料成为新的咖啡时，应该是可以复用才好。

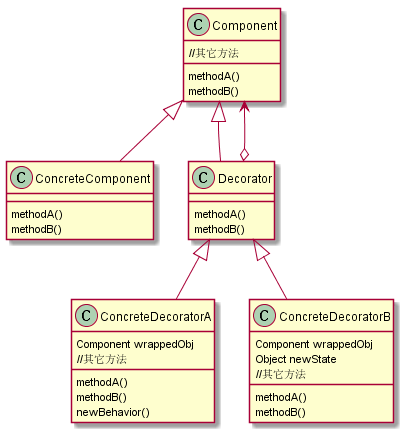
**装饰者模式：动态的将责任附加到对象上，若要扩展功能，装饰者提供了比继承更有弹性的替代方案。**

有时候我们希望给某个对象，而不是整个类添加一些功能，因此使用装饰者模式比继承更加灵活。使用继承的话，所有对象都有相同功能，事实上比如一个文本框，我们可能希望它拥有边框，而另外的文本框我们希望它拥有滚动条。

当然我们使用组合就可以实现这一职责，使用装饰者模式更多的是因为可能有大量独立的扩展，而每一种扩展将产生大量的子类，使得子类数目呈爆炸性增长。

* 装饰者和被装饰对象拥有相同的超类型。
* 可以用一个或多个装饰者包装一个对象。
* **既然装饰者和被装饰对象拥有相同的超类型，所以在任何需要原始对象（被包装的）的场合，可以用装饰过的对象代替它。**
* 装饰者可以在所委托被装饰对象的行为之前与/或之后，加上自己的行为，以达到特定的目的。
* 对象可以在任何时候被装饰，所以在运行时动态地，不限量地用你喜欢的装饰者来装饰对象。

第三点是装饰者模式的本质优点。



**要点：**

除了继承，装饰者模式也可以让我们扩展行为。

装饰者会导致设计中出现许多小对象，如果过度使用，会让程序变得很复杂。

Java I/O库中，那些复杂的小类都是用来包装InputStream的。