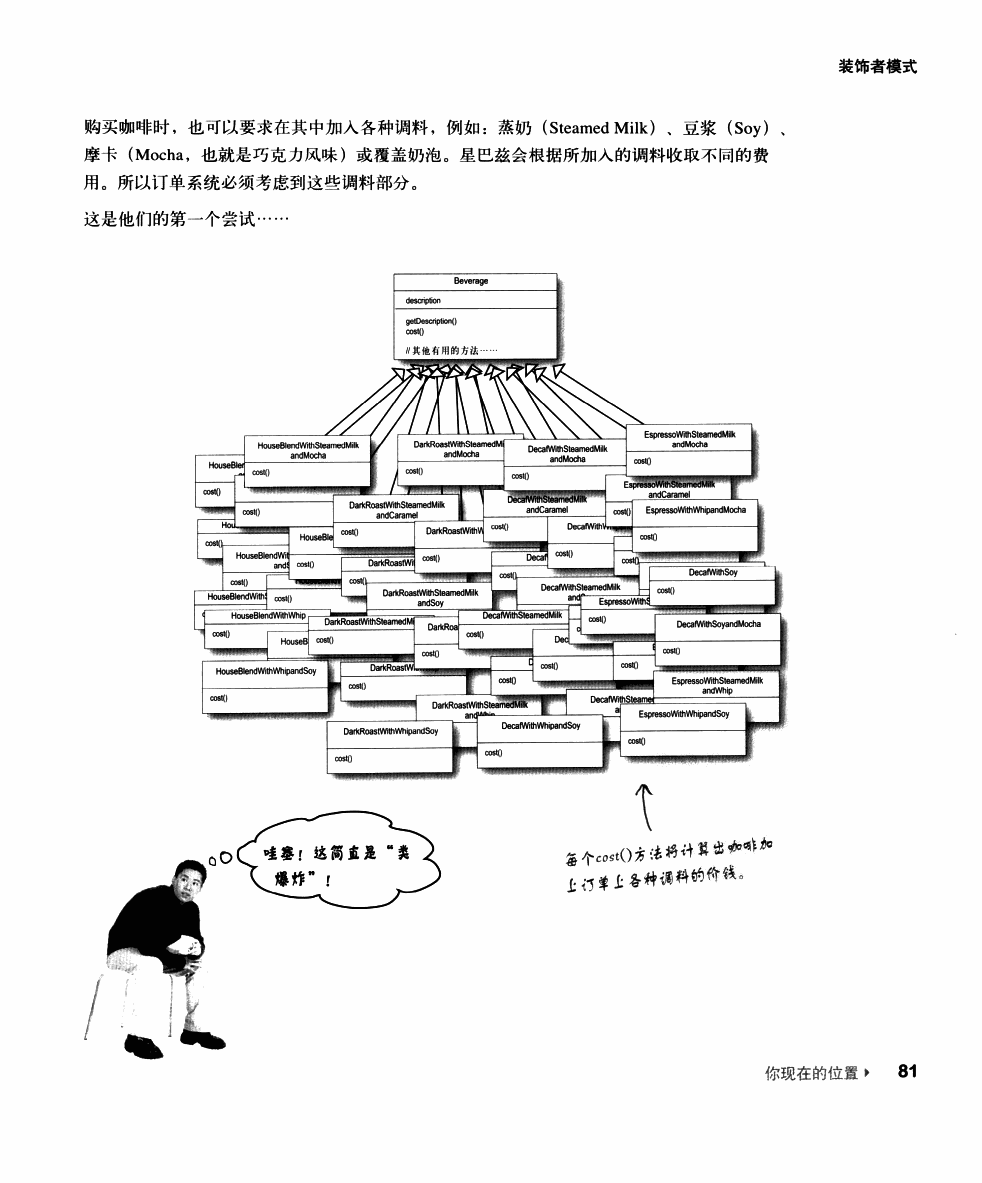
意图：动态的给一个对象添加一些额外的职责，同时又不改变其结构。

是结构型模式。

装饰者模式可以替代继承，就增加功能来说，装饰者模式相比生成子类更加灵活。

案例1：星巴兹点餐系统：刚开始的时候：



这明显设计的有问题，违反了面向接口编程而非实现、开闭原则等。

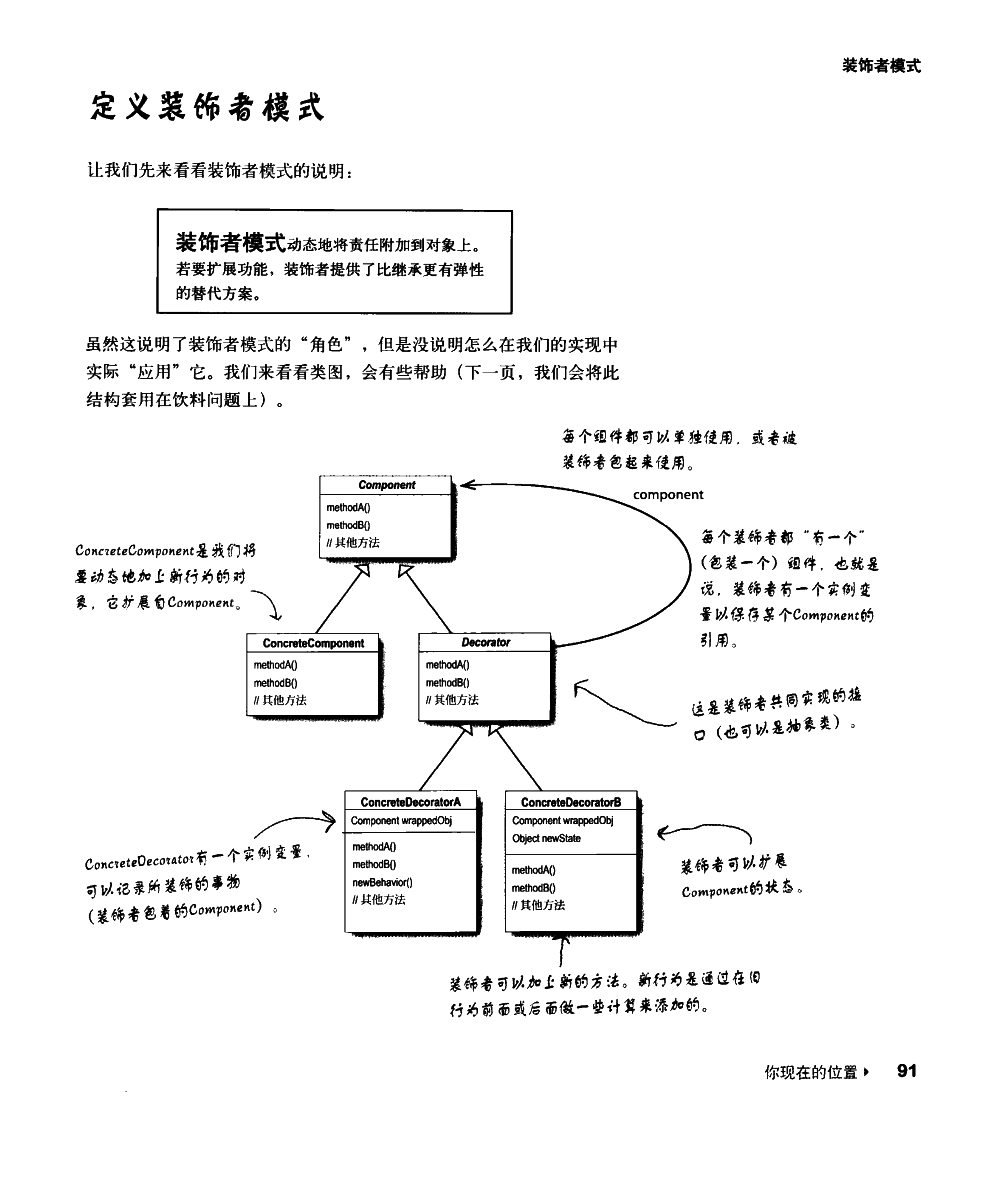
这是第二个设计：



这样的设计能够满足需求，但是缺点显而易见：

1. 当调料价格改变时，需要修改现有的代码；
2. 一旦出现新的调料，我们就要加上新的方法，并且修改超类中的cost()方法；
3. 不能满足需求“双倍摩卡”；
4. 有的饮料并不适合加某种调料，但是这种模式中所有子类都会继承这些方法，会出现“加摩卡的茶”；

所以我们需要设计一种方法，可以动态的增加功能，对修改关闭而对扩展开放，UML图如下：



实线箭头表示依赖，Decorator依赖于Component，所以Decorator中有Component实例；

装饰者模式中，装饰者Decorator仍然要继承Component，但这次继承重点在于装饰者和被装饰者需要是一样的类型，是要达到“类型匹配”，而不是利用继承获得行为。