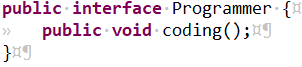
# Spring设计模式之装饰器模式

## 前言

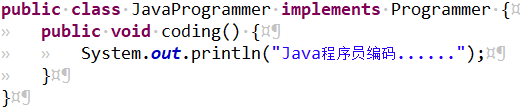
假设有这么一种情景，有两个程序员，一个Java、一个Python，都只负责一件事：编程。突然有一天，领导要求，在编程之前要做设计，在编程之后要进行测试。大家最先想到的，肯定就是要求这两个程序员各自在“编程”这个方法前后加上“设计”、“测试”的功能。但是在编程中违反了“开闭原则”，程序员原来的“编程”这个方法运行的很好，不要动他（修改关闭），但是你可以另找人来实现新的功能（扩展开放）。我们用代码实现一下。

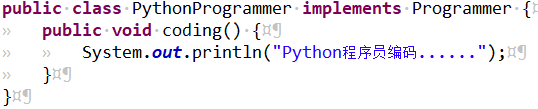
## 实现

程序员接口：



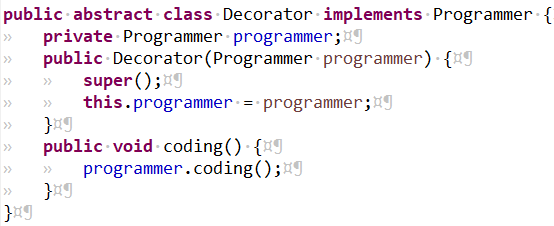
实现类：



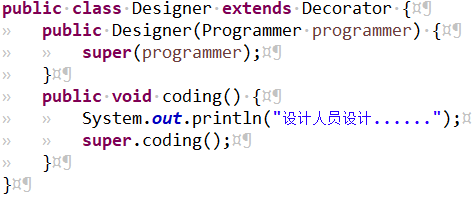


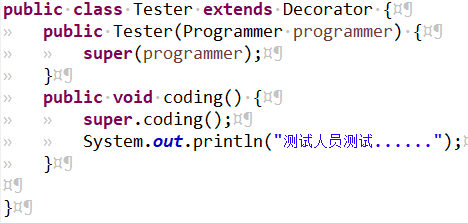
如果我们需要设计和测试功能，首先想到的就是在这两个类里面加代码，这样就违反了开闭原则。我们用装饰器模式来实现。

装饰器静态类：

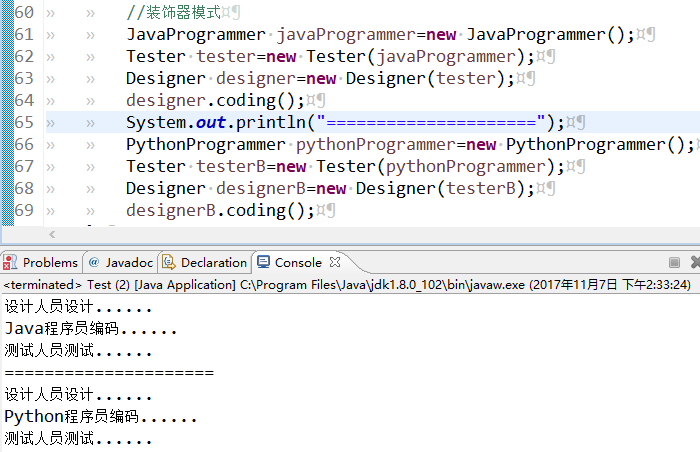


子类：





测试结果：



## 扩展

看到这，一定会有人说，装饰器模式和代理模式、适配器模式、模板方法模式看起来好像啊，有什么区别么？

**装饰器：**原来的方法（编码）已经不能满足新需求了，需要对其进行**扩展**（设计、测试），理论上我们可以对原方法无限地装饰下去，比如我们可以在“设计”之前再加个“需求分析”，在“测试”之后再加个“部署实施”等等。

**适配器：**原来的接口已经**不兼容**了，适配器在原对象和目标对象中间，通过对原对象兼容的那个接口，通过转换，调用目标对象那个不兼容的接口。举个不恰当的例子，一个英国人去买饭，听不懂中国服务员说什么（接口不兼容），这时候来了一个翻译（适配器），他能与英国人交流（接口适配成功），然后翻译不干活，而是通过中国服务员的活动，将结果再返回给英国人。

**代理：**你们要实现什么功能我不管，我只负责**调用**该调用的方法。有点类似前台MM，你们找谁，干什么我不管，你就告诉我你找谁，我给你找去。

**模板方法：**如果说装饰器是在原有的方法上扩展很多方法，那么模板方法模式就是将原来很多固定的方法抽出到父类里。一个是加法，一个是减法。