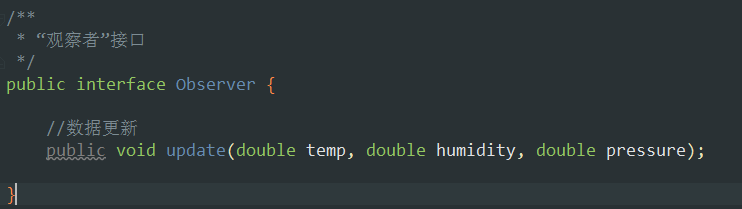
**《**Head Frist设计模式**》笔记:**

**一.观察者模式：**

**观察者模式定义了对象之间的一对多依赖，这样一来，当一个对象（该对象为主题）改变状态时，它的所有依赖者（观察者）都会收到通知并自动更新。比如报纸订阅服务，每当有新报纸（报纸为主题）时，订阅者（订阅者为观察者）会自动收到新报纸，订阅者也可以随时取消订阅，未订阅者也可以随时进行订阅。**

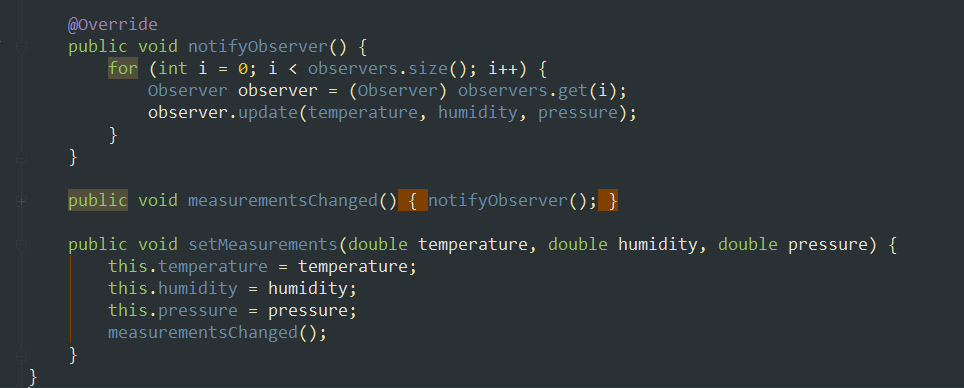
**典型：Tomcat的Listener。**

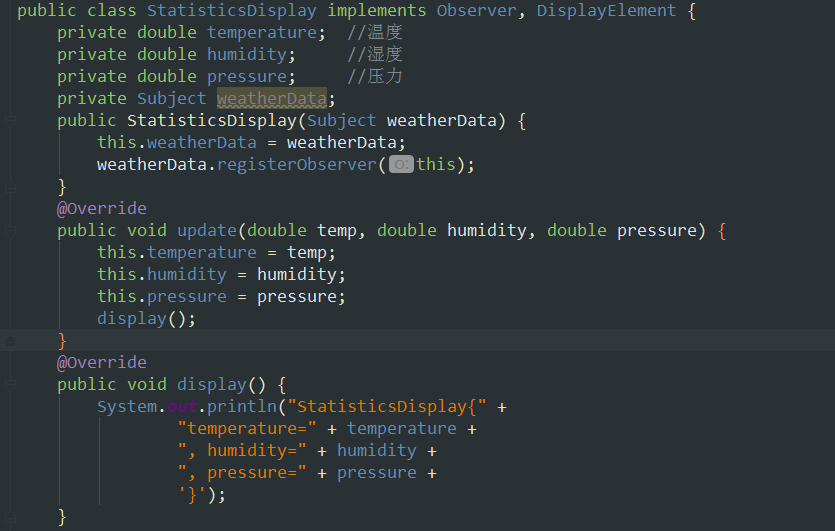


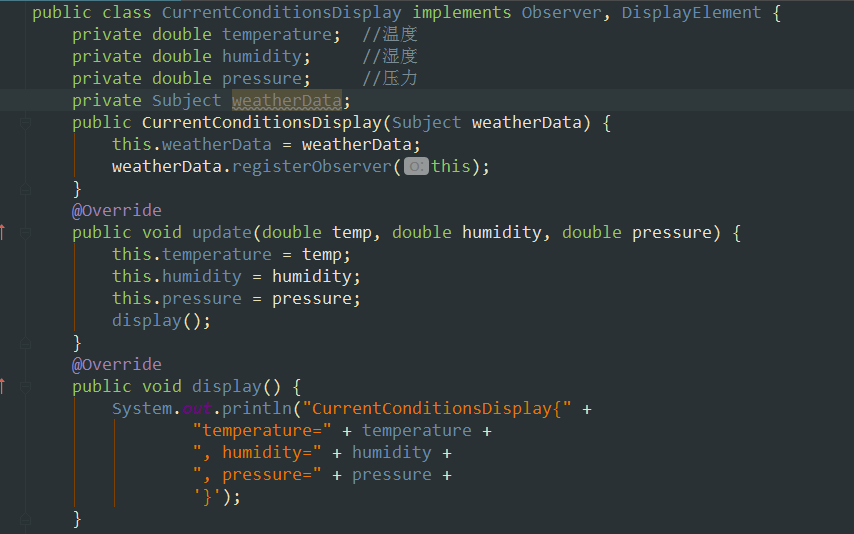


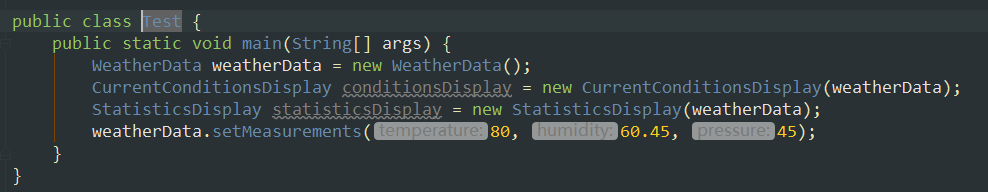




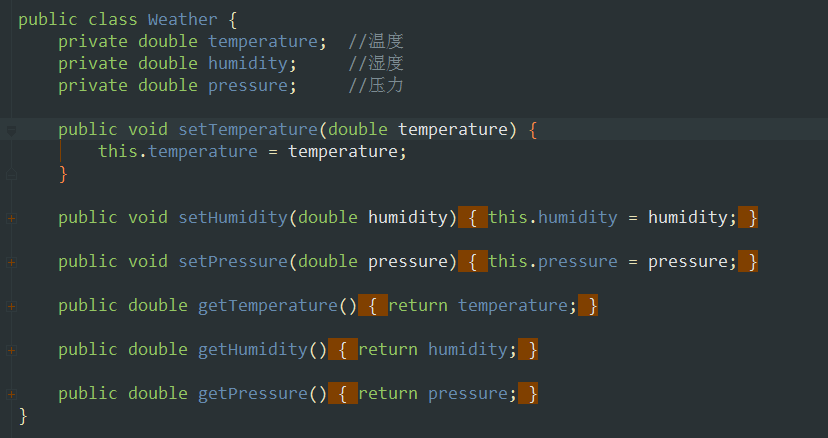


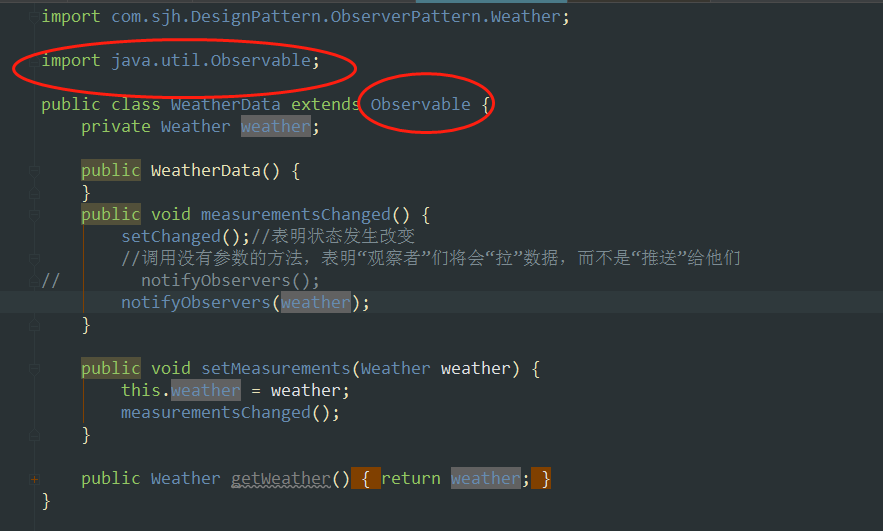




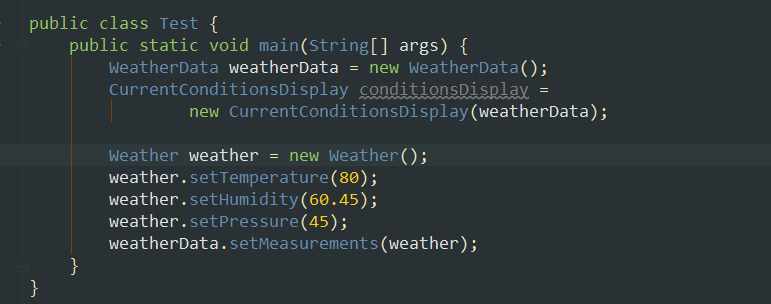


**Java内置观察者模式：**







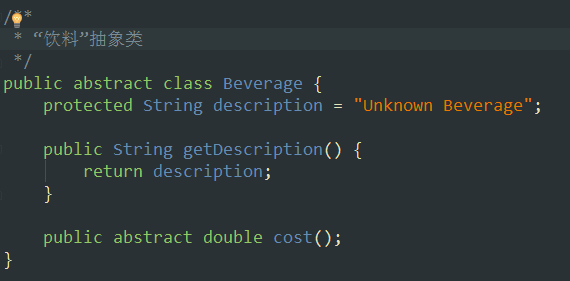


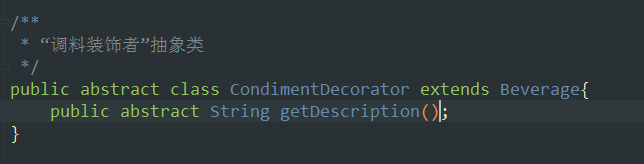
**二.装饰者模式：**

**装饰者模式动态地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。**

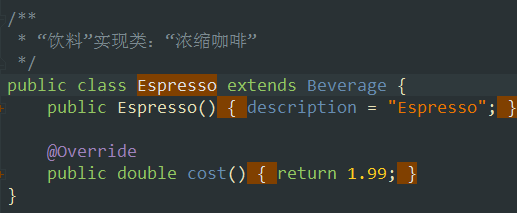
**开放-关闭原则：类应该对扩展开放，对修改关闭。**

**典型：JDK的IO流。**















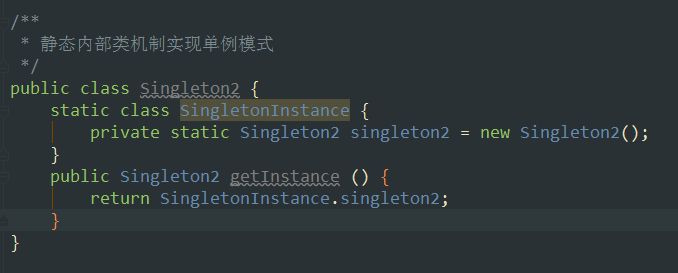


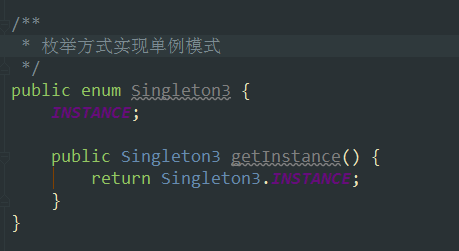
**三.单件模式：**

单件模式确保一个类只有一个实例，并提供一个全局访问点。

**典型：Spring的Bean。**



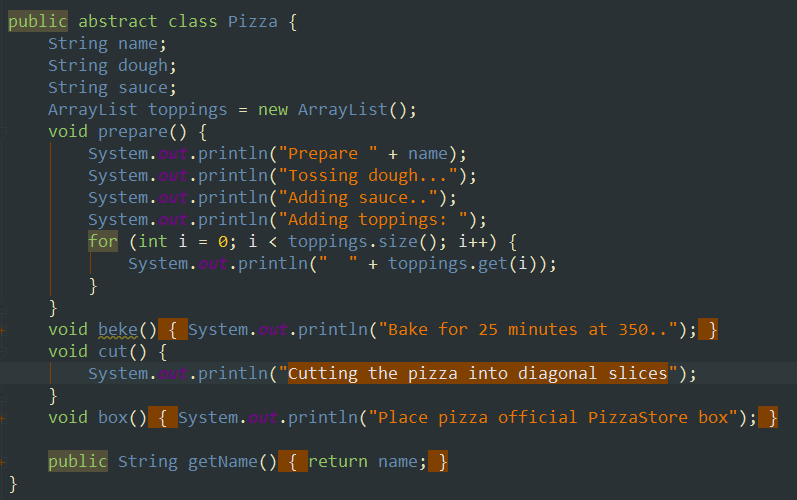


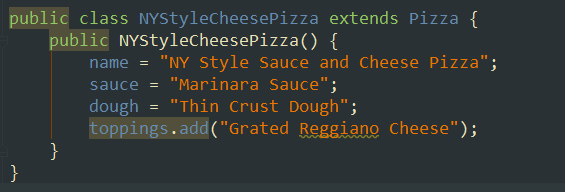


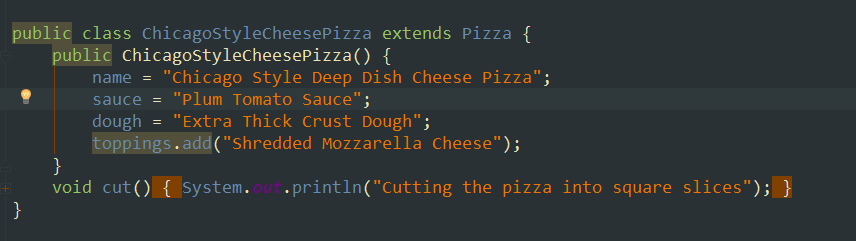
**四.工厂模式：**

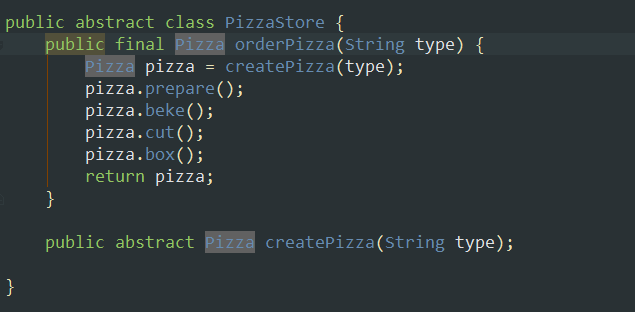
工厂方法模式定义了一个创建对象的接口，但由子类决定要实例化的类是哪一个。工厂方法让类的实例化推迟到子类。

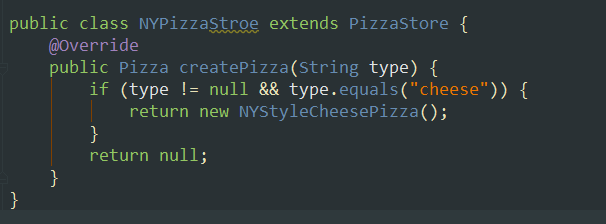
**典型：Spring的BeanFactory、MyBatis的SqlSessionFactory。**

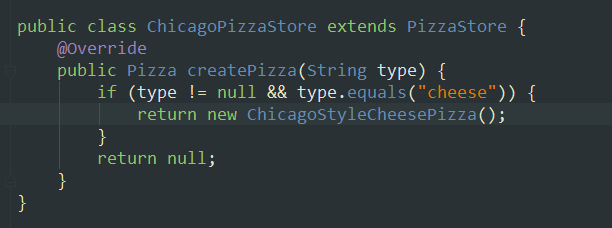


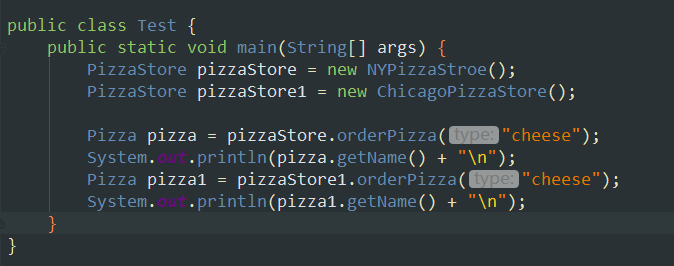




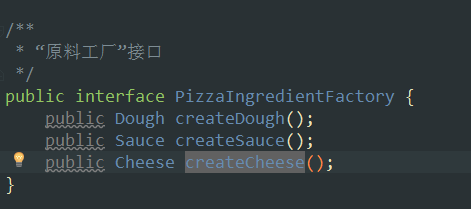


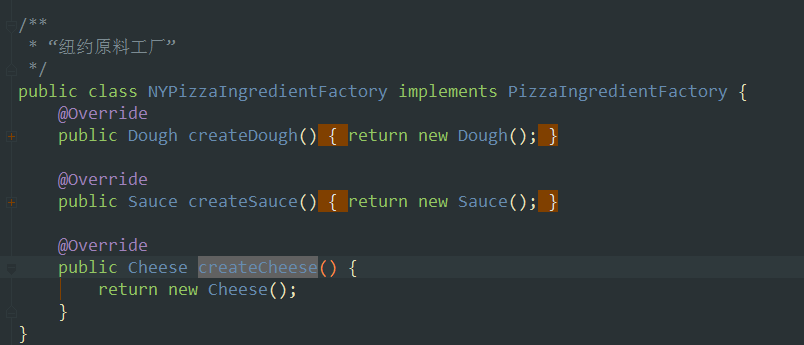


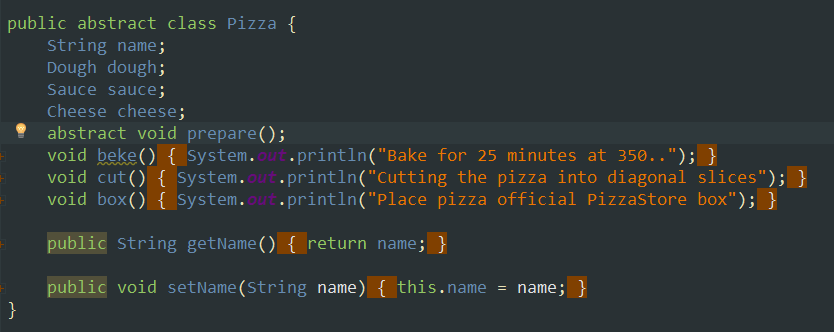


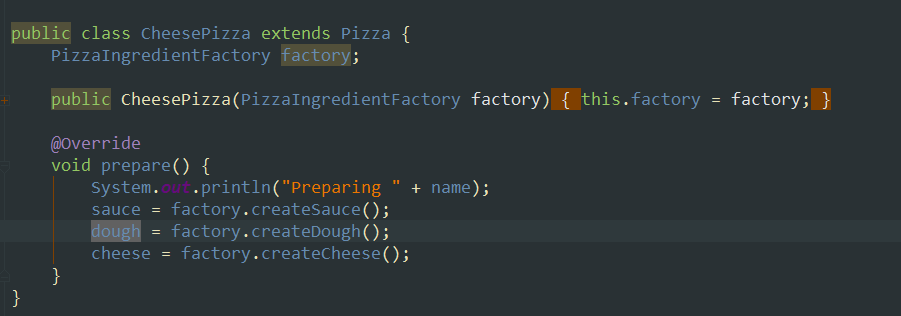


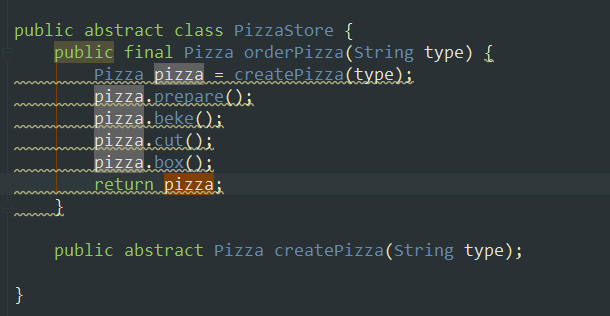
抽象工厂模式提供了一个接口，用于创建相关或依赖对象的家族，而不需要明确指定具体类。

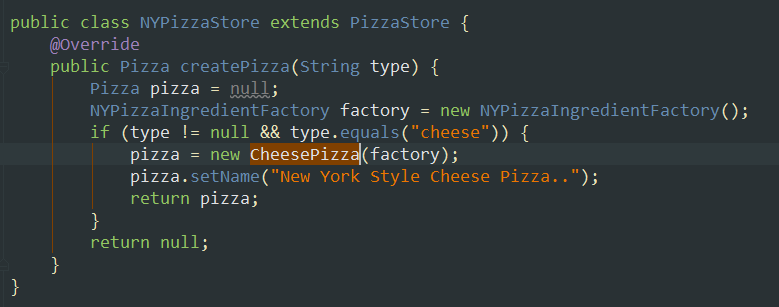


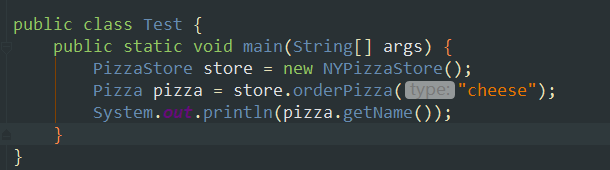






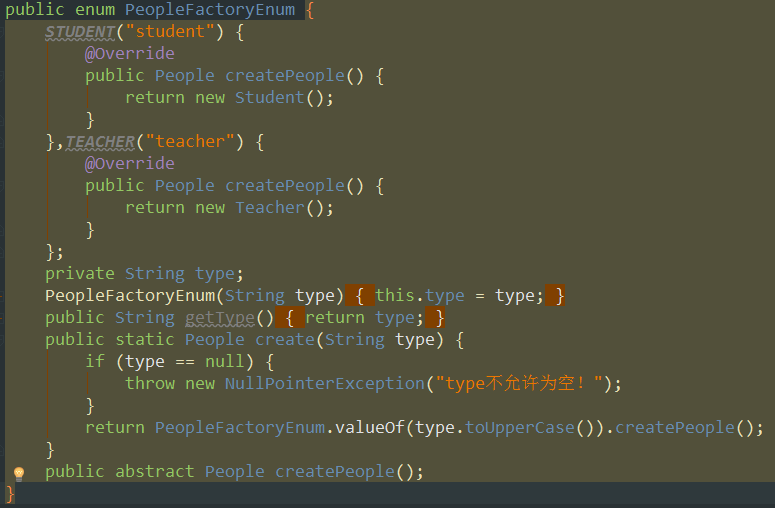






**依赖倒置原则：要依赖抽象，不要依赖具体类。**

枚举实现工厂模式：



**五.命令模式：**

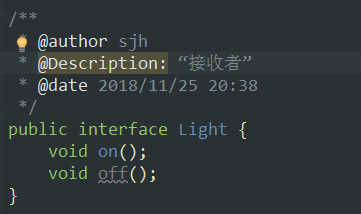
命令模式将“请求”封装成对象，以便使用不同的请求、队列或者日志来参数化其他对象。

当需要将发出请求的对象和执行请求的对象解耦的时候，使用命令模式。

**典型：Tomcat对请求的处理。**

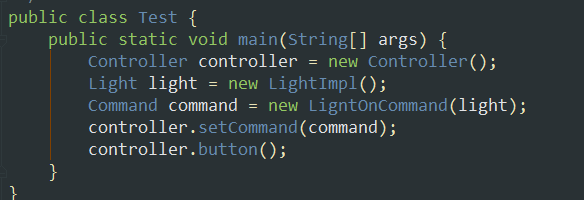






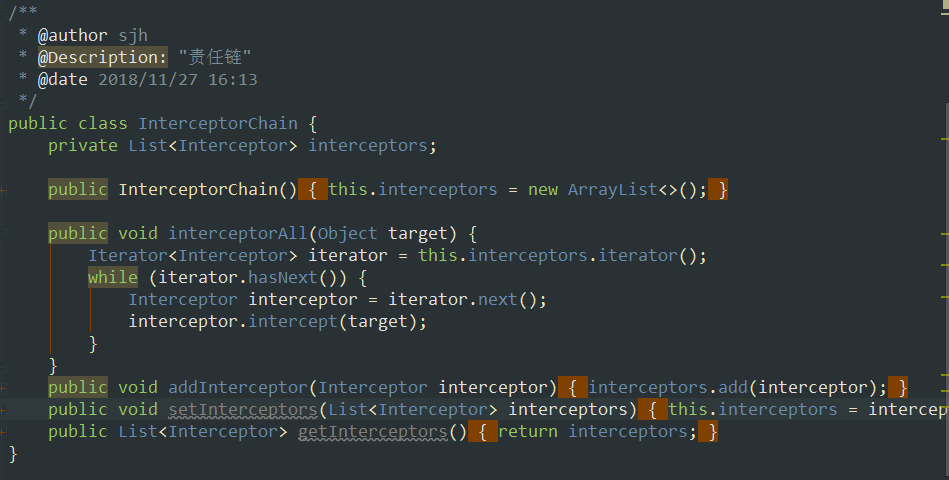


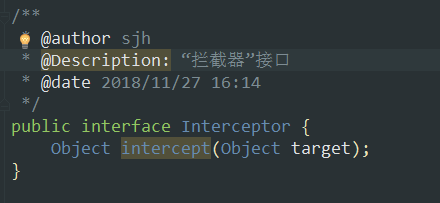


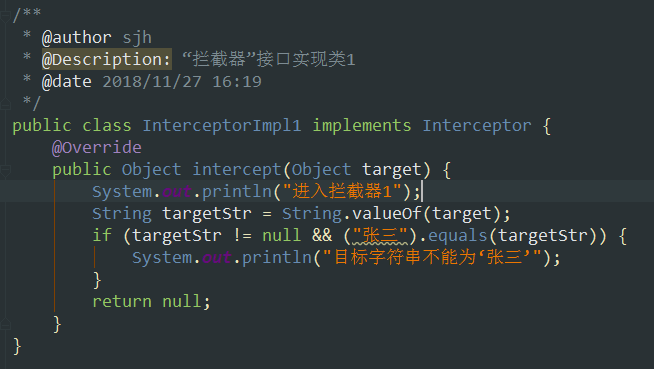


**六.责任链模式：**

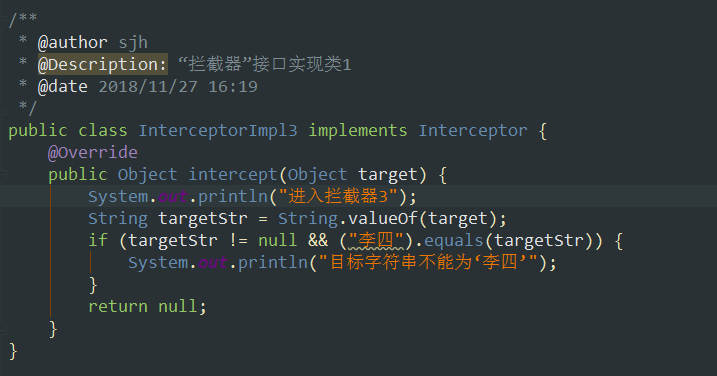
责任链模式为某个请求创建一个对象链，每个对象依次检查此请求，并对其进行处理，或者将它传到下一个对象。

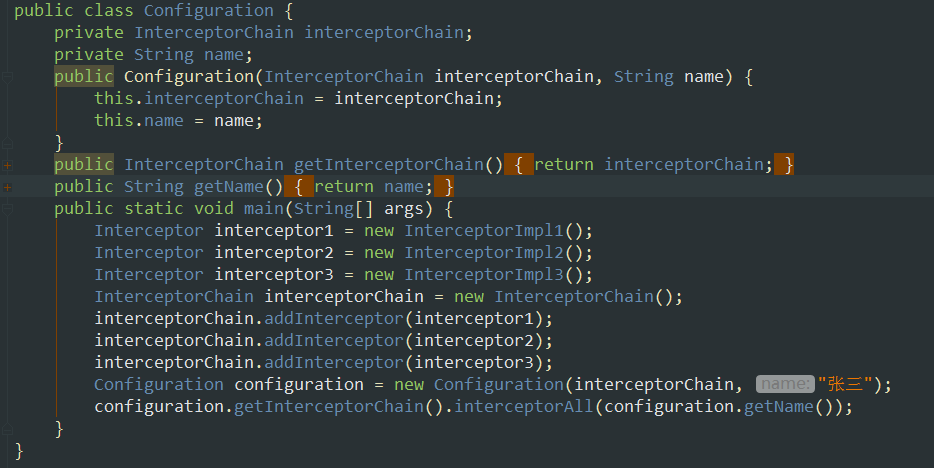




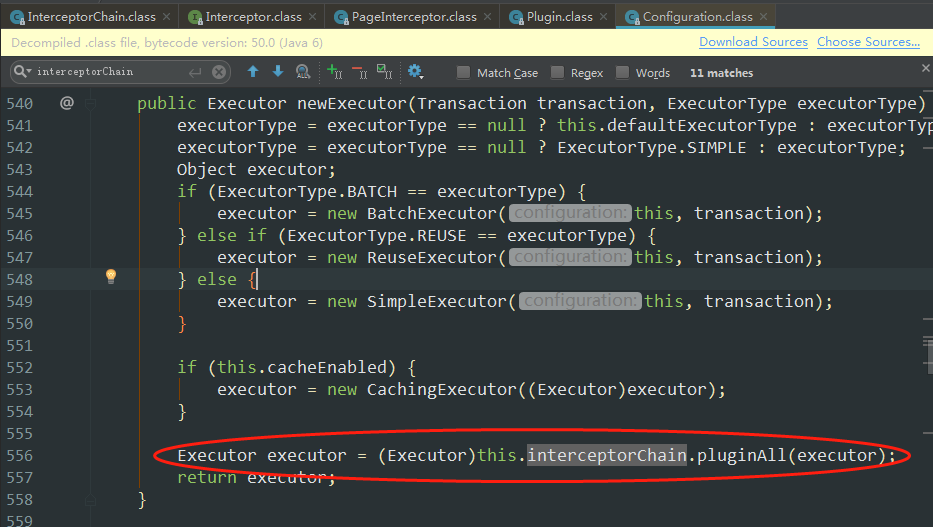


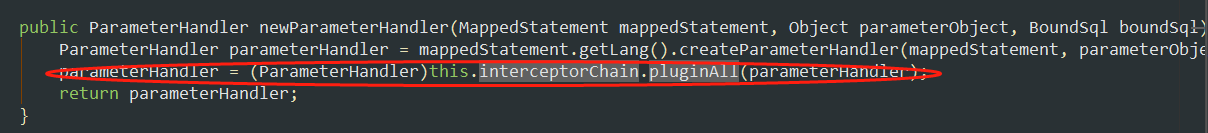


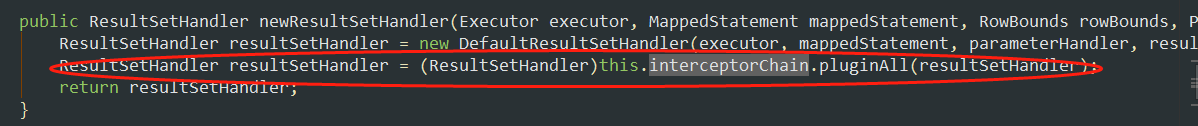


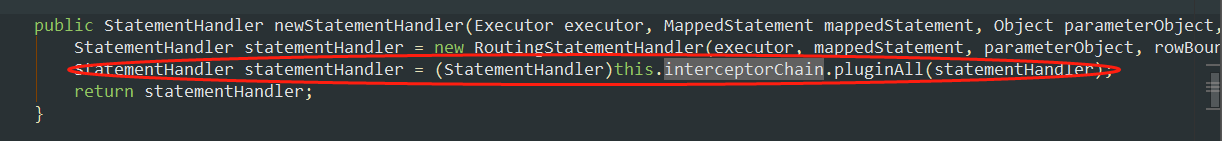


**典型示例（MyBatis插件机制）：**





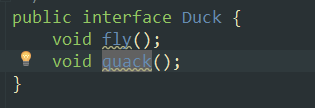


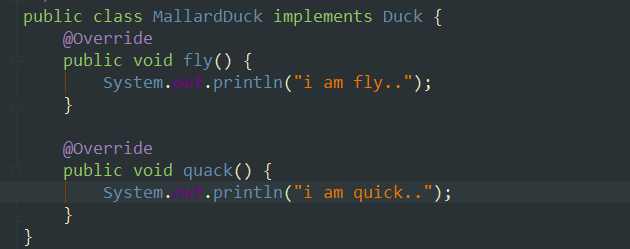


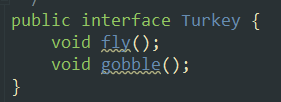
**七.适配器模式和外观模式：**

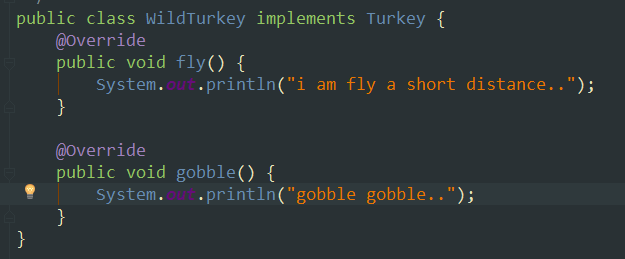
一、适配器模式：将一个类的接口转换成客户期望的另一个接口，适配器让原本不兼容的类可以合作无间。可以让客户与被适配者解耦。

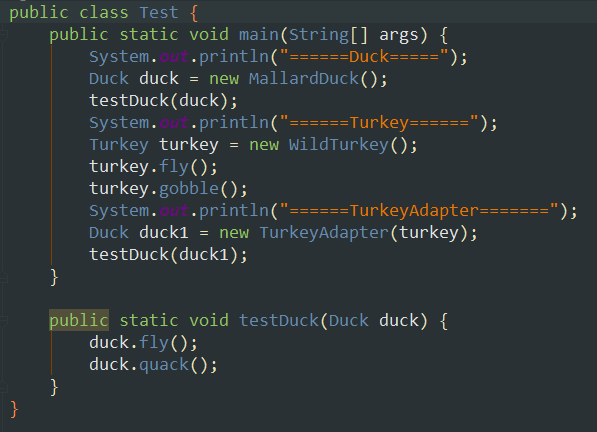








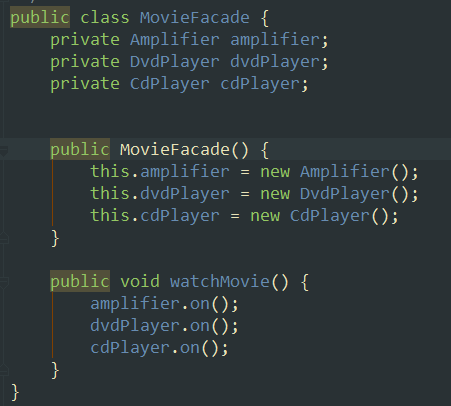


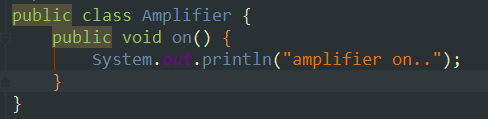


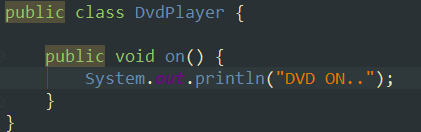
二、外观模式：

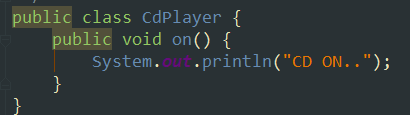
外观模式提供了统一的接口，用来访问子系统中的一群接口。外观定义了一个高层接口，让子系统更容易使用，但是也不影响单独使用子系统。

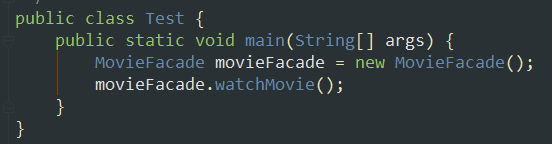
外观模式让客户端与子系统之间避免紧耦合，让接口更简单。











**“最少知识”（迪米特、陌生人）原则：只和你的密友说话。在系统设计中，不要让太多的类耦合在一起，避免修改系统中的一部分会影响到其他部分。**

**方针：**

**就任何对象而言，在该对象的方法内，我们只应该调用属于以下范围的方法：**

**1）、该对象本身；**

**2）、被当做方法的参数而传递进来的对象；**

**3）、此方法所创建或实例化的任何对象；**

**4）、对象的任何组件。**

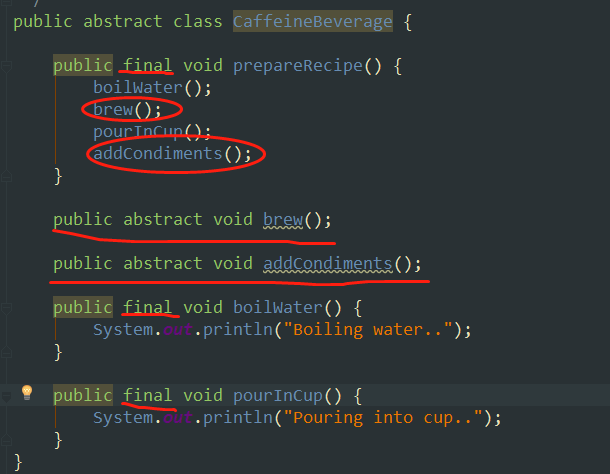
**如果某对象是调用其他的方法的返回结果，不要调用该对象的方法！**

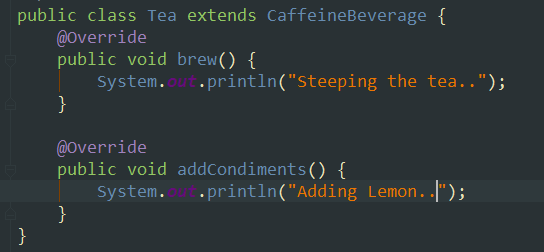
**适配器将一个对象包装起来以改变其接口；装饰者将一个对象包装起来以增加新的行为和责任；外观将一群对象包装起来以简化其接口。**

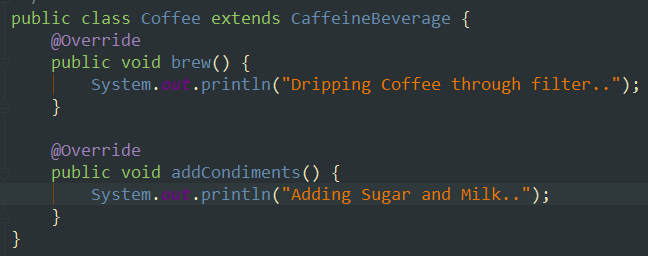
**八.模板方法模式：**

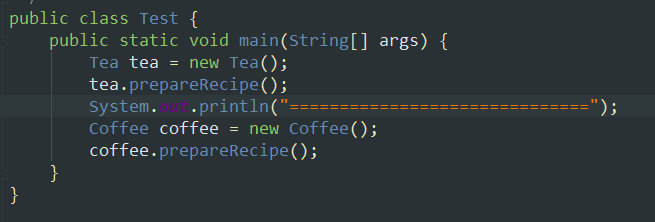
模板方法模式在一个方法中定义一个算法的骨架，而将一些步骤延迟到子类中。模板方法使得子类可以在不改变算法结构的情况下，重新定义算法中的某些步骤。

这个模式是用来创建一个算法的模板。模板就是一个方法，更具体地说，这个方法将算法定义成一组步骤，其中的任何步骤都可以是抽象的，由子类负责实现。这可以确保算法的结构保持不变，同时由子类提供部分实现。







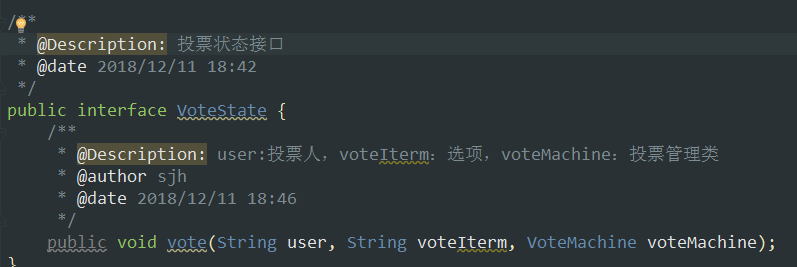


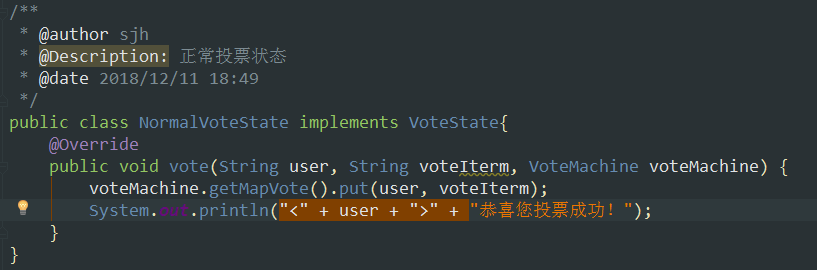
钩子是一种被声明在抽象类中的方法，但只有空的或者默认的实现。钩子的存在，可以让子类有能力对算法的不同点进行挂钩。要不要挂钩，完全由子类自行决定。

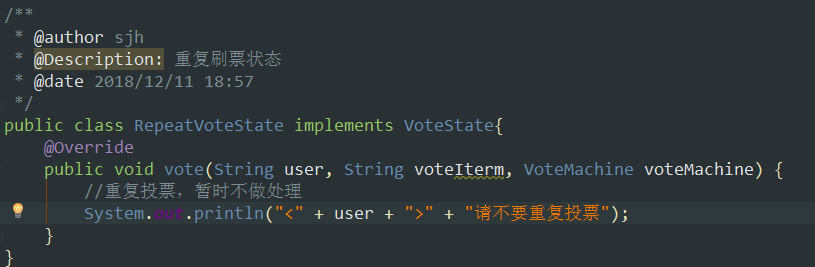
“好莱坞”原则：允许低层组件将自己挂钩到系统上，但是高层组件会觉得什么时候和怎样使用这些低层组件。换句话说，高层组件对待低层组件的方式是“别调用我们，我们会调用你”。

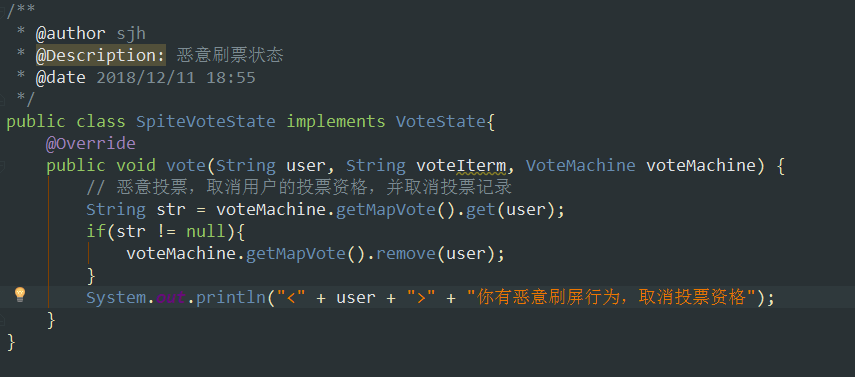
**九.状态模式：**

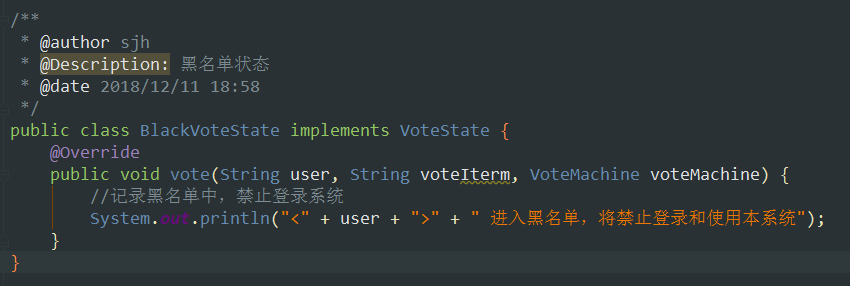
状态模式允许对象在内部状态改变时改变它的行为，对象看起来好像修改了它的类。



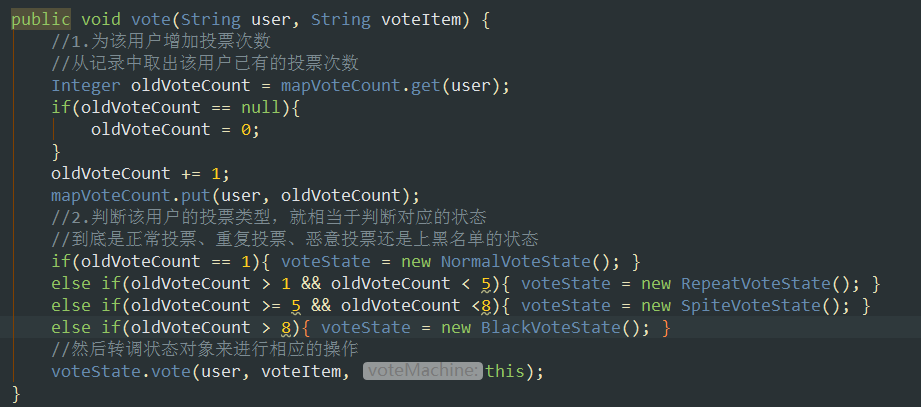


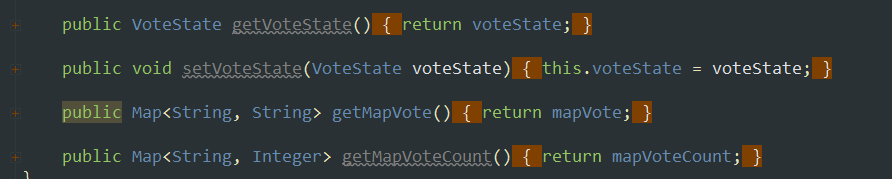


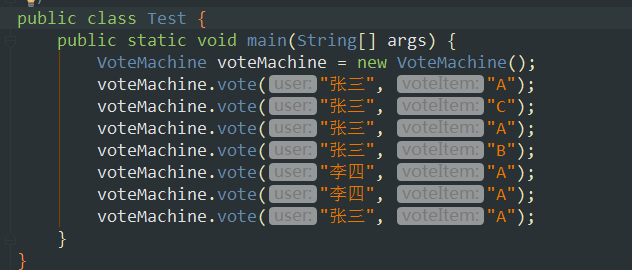


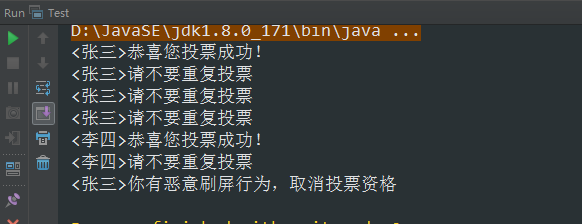












**十.代理模式：**

代理模式为另一个对象提供了一个替身或占位符以控制对这个对象的访问。

