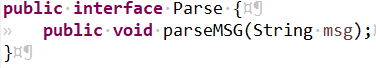
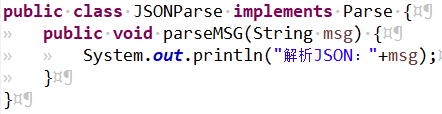
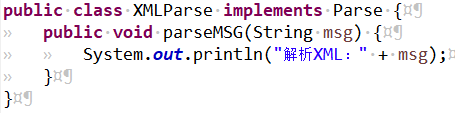
# Java设计模式之策略模式与状态模式

## 策略模式

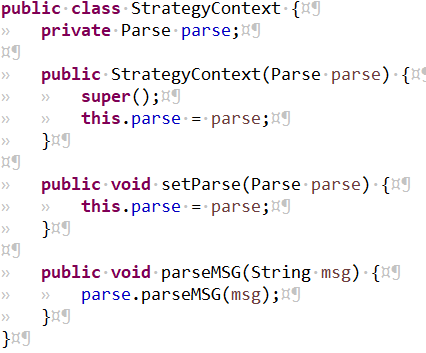
假设你的系统要与第三方接口进行数据的交互，内容都是一样的，但是A系统用XML，B系统用JSON。你这边要是粗暴地用if/else就太low了。我们用策略模式来做。



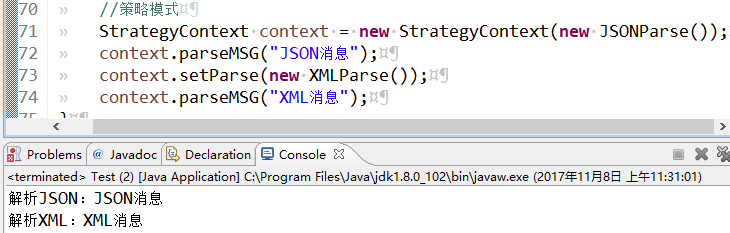




环境类：



测试：

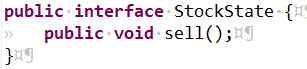


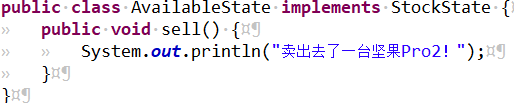
这么做，以后再多个消息格式，也能很好的扩展，多加个解析类即可。当然这也有一个很明显的问题，客户端必须知道所有策略，并且明确知道要用哪个策略。

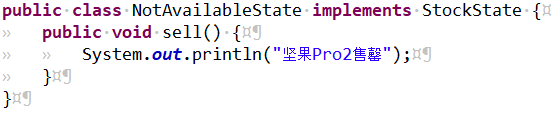
## 状态模式

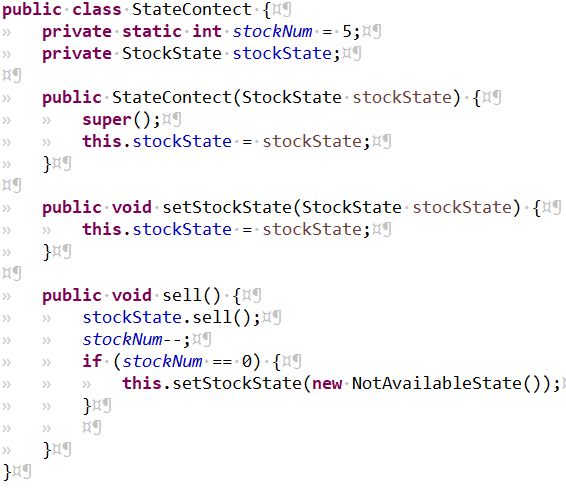
假设一个库存表，有一条记录：锤子手机坚果Pro2，库存5台。假设我们只有“卖”这一个动作，如果库存大于0，库存就-1；如果库存等于0，就告知“售罄”。

简单粗暴一点，那就还是if/else判断，然后进行处理呗。但是如果过两天又多了个要求：如果库存等于2，就发通知补货。如果还是加判断分支，加处理代码的话，可想而知，随着需求的增加，状态的细分，这个处理类，这个分支得是多么大。我们用状态模式实现一下。

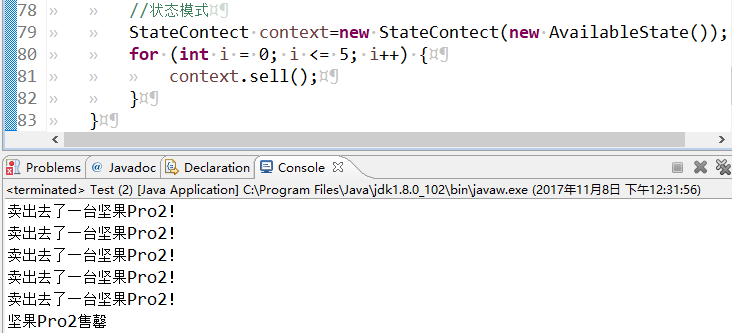








测试：



以后如果有了新需求，比如库存告警，就只要增加一个告警处理类，并在StateContext上加分支即可。

有人会问这和用一个处理类，在里面增加判断分支有什么好处呢？反正都是加分支，开闭原则已经违反了，用状态模式，我还要多写一个处理类。

在文章这个例子里，逻辑清晰，处理代码少，当然不如直接上一个类方便，但是大家想想，如果在实际工作中，处理代码很多，分支之内又有分支，这样放在一个类里还合适么？我在工作中遇到过这样的，一个类小几千行代码，分支套子分支，子分支套孙分支。当时改的我真是要吐血啊！

## 区别

策略与状态两个模式，一看就知道这俩太像了，画UML类图的话，这俩简直一模一样，那么有什么区别呢？

简单地说，策略模式，要求客户端必须指定用哪个策略，什么时候换策略由客户端决定，客户端通过Context类进行切换。

而状态模式，客户端顶多指定一个初始状态，剩下的事儿，由Context根据内部的状态变化进行处理、切换，没客户端的事了。

如果有喜欢在家泡咖啡喝的朋友应该能明白，策略模式就是半自动，什么时候烧水，什么时候研磨咖啡豆，什么时候冲泡，由你自己决定并切换。



而状态模式就是全自动，你按一下按钮，什么时候该干什么事，咖啡机自己决定。

