设计模式:JDKObserver设计模式之研究 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E8_AE_BE_E8_AE_A1_E6_A8_A1_E5_c104_144667.htm 目前设计模式的介绍性文章越来越多，但设计模式的研究性文章仍然比较欠缺，这着实让人觉得有点遗憾。本文旨在抛砖引玉，具体分析一下java中jdk自带的observer设计模式（下文如没特别指出，observer设计模式就意指java中jdk自带的observer设计模式）的实现。1. Observer 设计模式概要Observer设计模式在GOF里属于行为设计模式。JDK里提供的observer设计模式的实现由java.util.Observable类和java.util.Observer接口组成。从名字上可以清楚的看出两者在Observer 设计模式中分别扮演的角色：Observer是观察者角色，Observable是被观察目标(subject)角色。Observable是一个封装subject基本功能的类，比如注册observer（attach功能），注销observer（detatch功能）等。这些功能是任何一个扮演observerable角色的类都需要实现的，从这一点上来讲，JDK里将这些通用功能专门封装在一个类里，显得合情合理。通常情况下，我们的类只要从Observerable类派生就可以称为observerable角色类，使用非常简单。2. 使用observer设计模式存在的困难但我们不得不注意到，在项目实际开发当中，情况往往要复杂得多。java不支持多继承特性在很多时候是阻碍我们使用observer设计模式的绊脚石。比如说，我们设计的一个类已经是某个类的派生类，在这种情况下同时想让它扮演observerable角色将变得麻烦。如何实现“ 多继承 ”的效果是摆在我们面前的一大难题。下面我们首先分析一下Observable类。3. Observable类“ 触发

通知 ” 的原理Observable必须 “ 有变化 ” 才能触发通知observer这一任务，这是它的本质体现。查看源码便可知一二。Observerable部分源码如下：//……省略……private boolean changed = false.//……省略……public void notifyObservers(Object arg) {//……省略……Object[] arrLocal.synchronized (this) {//……省略……if (!changed)return.arrLocal = obs.toArray().clearChanged().}//……省略……protected synchronized void setChanged() {changed = true.}protected synchronized void clearChanged() {changed = false.}正如粗的斜体标注部分所示，在notifyObservers(Object arg) 方法里if (!changed)return.语句告诉我们，若changed属性值为false，将直接返回，根本不会触发通知操作。并且我们注意到changed 属性被初始化为false，这将意味着如果我们不主动设置changed属性为true，将不会有任何变化，也就是说根本起不到 “ 通知 ” 作用。因此，设置changed属性的值是我们应用jdk observer 设计模式的关键所在。那么如何才能设置changed属性呢？从源码可以看出，唯一的入口是通过setChanged()。下面我们分析一下changed属性及相关的方法setChanged()和clearChanged()。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com