一、观察者模式的原理

1. internet气象站小demo，普通OO设计方案，有些问题

提供温度、气压和温度的接口

测量数据更新时需要通知给第三方

需要设计开放型API，便于其他第三方公司也能接入气象站获取数据。

传统方案：在气温变化的类中第三方对象放入其中，在里面通知第三方公司。但是这样扩展性不好，无法动态添加第三方公司。

2. 引入了观察者模式概念

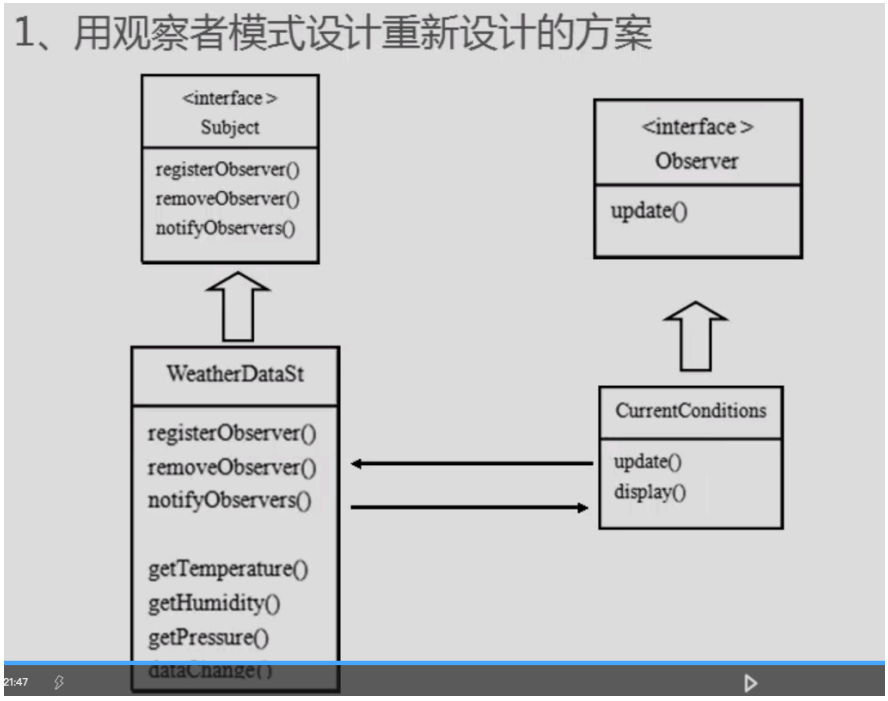
就像订牛奶业务。（1）奶站，Subject （2）用户，Observer

Subject 功能：登记、移除和通知

Observer功能：接受输入

观察者模式定义：对象之间多对一依赖的一种设计方案，被依赖的对象为Subject，依赖的对象为Observer，Subject通知Observer变化。

3. 新的设计方案



二、demo中的问题

- [ ] 新增一个观察者和移除一个不够灵活

三、java内置的观察者模式

（1）java内置的观察者

Observable类 注意这个其实是对应着Subject接口 相当于观察者订阅的一个主题 这个也去定义了

register、remove、notify等这些功能

注册观察者的顺序和通知的顺序是相反的。

Observable中提供了灵活的通知观察者的控制条件，setChanged方法，这个方法调用才会通知观察者。这样增加了灵活性。

Observer接口 也是定义了update接口。这里的参数可以不传，选择pull还是push的方式，结合Observable的两个通知方法

notifyObservers() notifyObservers(Object args) 一个拉一个推的方法

```
/**
 * A class can implement the <code>Observer</code> interface when it
 * wants to be informed of changes in observable objects.
 *
 * @author  Chris Warth
 * @see     java.util.Observable
 * @since   JDK1.0
 */
public interface Observer {
    /**
     * This method is called whenever the observed object is changed. An
     * application calls an <tt>Observable</tt> object's
     * <code>notifyObservers</code> method to have all the object's
     * observers notified of the change.
     *
     * @param   o     the observable object.
     * @param   arg   an argument passed to the <code>notifyObservers</code>
     *                 method.
     */
    void update(Observable o, Object arg);
}
```

（2）使用java内置观察者模式重写项目

Subject----Observable

Observer---Observer

（3）java内置观察者的注意点

Observable是类不是接口，通知观察者，一定要setChanged

通知观察者一个是pull方式、一个push方式。

四、观察者模式的总结

1. 观察者模式解决了什么问题：

解决了对象之间多对一的一种对应关系。一指的就是Subject，多的对象指的就是观察者Observer

2. 有什么意义

松耦合，把多对一的关系松耦合。没有依赖关系。隔离了相互的影响。

3. java内置观察值注意点

见上