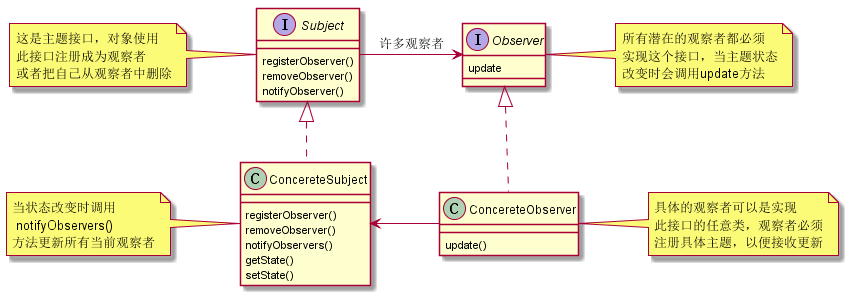
## 第二章：观察者模式

*观察者模式定义了对象之间的一对多依赖，这样一来，当一个对象改变状态时，它的所有依赖者都会收到通知并自动更新。*



主题是真正拥有数据的人，观察者是主题的依赖者，在数据变化时更新，这样比起让许多对象控制同一份数据来，可以得到更干净的OO设计。

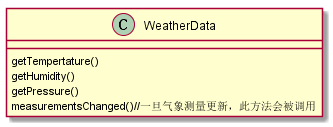
观察者模式提供了一种对象设计，让主题和观察者之间松耦合。

我们可以独立的复用主题或观察者，如果我们在其它地方需要使用主题或观察者，可以轻易地复用，因为二者并非紧耦合。

改变主题或观察者其中一方，并不会影响到另一方。因为二者是松耦合的，所以只要他们之间的接口仍被遵守，我们就可以自由地改变他们

|  |
| --- |
| *设计原则4：为了交互对象之间的松耦合设计而努力* |

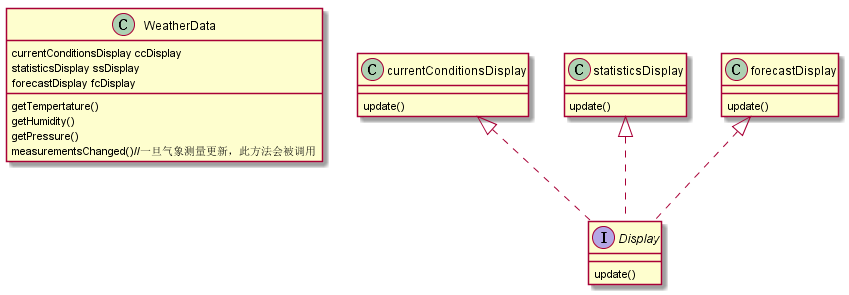
案例：WeatherData对象负责追踪目前的天气状况（温度，湿度，气压），有三种布告板，分别显示当前的状况，气象统计以及简单的预报，当WeatherData数据更新时，三种布告板必须实时更新。而且必须是可扩展的，让其它人可以自定义自己的布告板，插入到我们的应用中来。



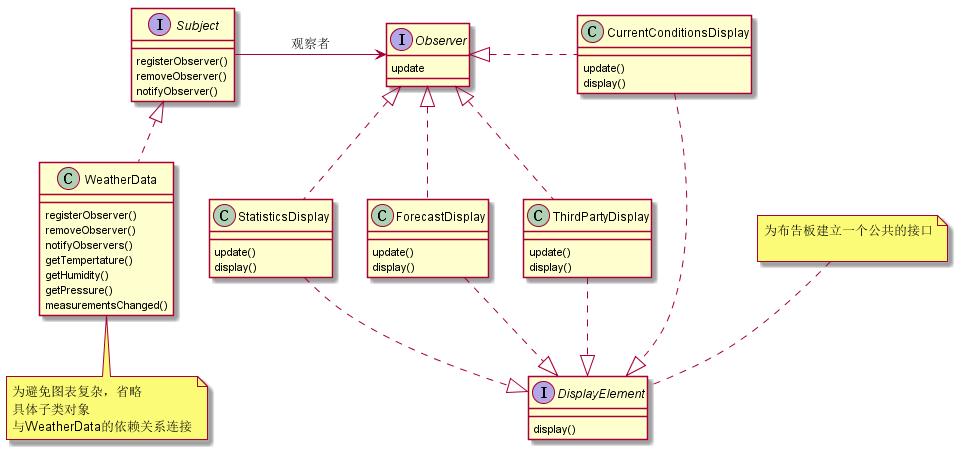
我们的工作是实现measurementsChanged()方法实时更新布告板。

针对实现编程：

|  |
| --- |
| public void measurementsChanged(){  float temp = getTemperature();  float humidity = getHumidity();  float pressure = getPressure();  currentConditionDisplay.update(temp, humidity, pressure);  statisticsDisplay.update(temp, humidity, pressure);  forecastDisplay.update(temp, humidity, pressure);  } |



针对接口编程：



针对接口编程的好处是我们把可变的Subject与Observer从WeatherData中分离了出来进行复用，我们只需要处理这个分离出的可变接口需要完成什么样的功能，提供接口即可，而不用去管具体实现，具体实现留给子类去实现接口。