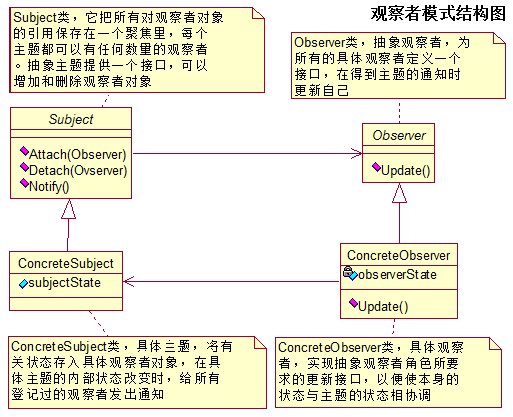
其实就是发布订阅模式，发布者发布信息，订阅者获取信息，订阅了就能收到信息，没订阅就收不到信息。

**1、这个模式的结构图**

****

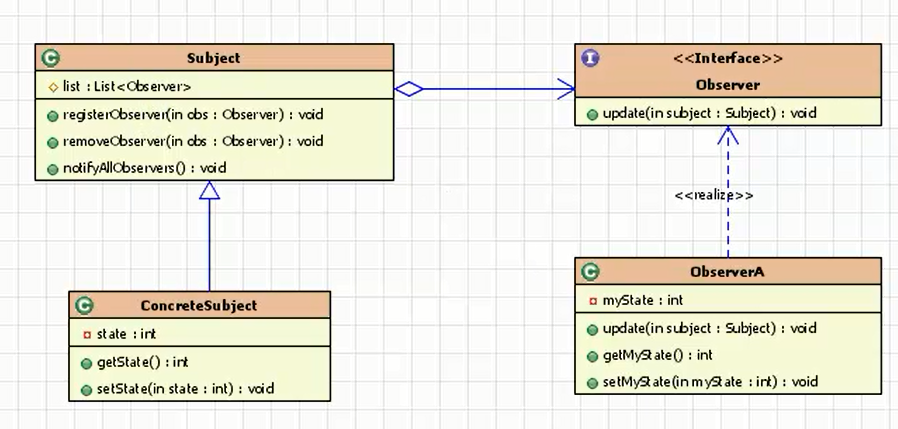
**2、可以看到，该模式包含四个角色**

**抽象被观察者角色(**Subject**)**：也就是一个抽象主题，它把所有对观察者对象的引用保存在一个集合中，每个主题都可以有任意数量的观察者。**抽象主题提供一个接口，可以增加和删除观察者角色。一般用一个抽象类和接口来实现。**

**抽象观察者角色(**Observer**)**：为所有的具体观察者定义一个接口，在**得到主题通知时更新自己**。

**具体被观察者角色(**ConcreteSubject**)**：也就是一个**具体的主题，在集体主题的内部状态改变时，所有登记过的观察者发出通知。**

**具体观察者角色(**ObserverA**)**：实现抽象观察者角色所需要的更新接口，**一边使本身的状态与制图的状态相协调**。



**package** observer;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* 抽象类  
 \*/***public class** Subject{  
  
 **protected** List<Observer> **observerList** = **new** ArrayList<>();  
  
 **public void** registerObserver(Observer observer) {  
 **observerList**.add(observer);  
 }  
  
 **public void** removeObserver(Observer observer) {  
 **observerList**.remove(observer);  
 }  
  
 **public void** nofityAllObserver() {  
 **for** (Observer observer : **observerList**) {  
 observer.update(**this**);  
 }  
 }  
  
  
}

**package** observer;  
  
**public class** ConcreteSubject **extends** Subject {  
  
 **private int state**;  
  
 **public int** getState() {  
 **return state**;  
 }  
  
 **public void** setState(**int** state) {  
 **this**.**state** = state;  
 **this**.nofityAllObserver();  
 }  
}

**package** observer;  
  
**public interface** Observer {  
 **void** update(Subject subject);  
}

**package** observer;  
  
**public class** ObserverA **implements** Observer {  
  
 **private int myState**;  
  
 @Override  
 **public void** update(Subject subject) {  
 **myState** = ((ConcreteSubject) subject).getState();  
 }  
  
 **public int** getMyState() {  
 **return myState**;  
 }  
  
 **public void** setMyState(**int** myState) {  
 **this**.**myState** = myState;  
 }  
}

**package** observer;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** Test {  
 **public static void** main(String[] args) {  
 ConcreteSubject subject = **new** ConcreteSubject();*//广播* ObserverA observer1 = **new** ObserverA();  
 ObserverA observer2 = **new** ObserverA();  
 ObserverA observer3 = **new** ObserverA();  
  
 subject.registerObserver(observer1);  
 subject.registerObserver(observer2);  
 subject.registerObserver(observer3);  
  
 subject.setState(1000);  
*// subject.nofityAllObserver();* System.***out***.println(observer1.getMyState());  
 System.***out***.println(observer2.getMyState());  
 System.***out***.println(observer3.getMyState());  
  
  
 }  
}