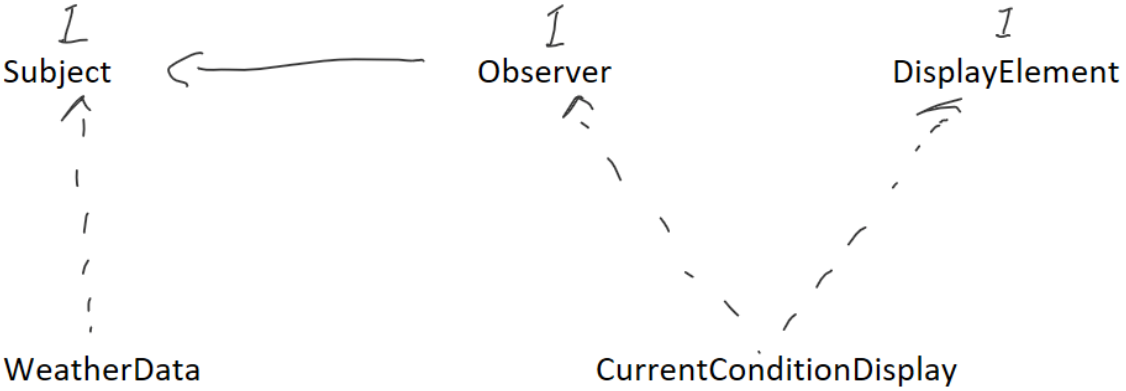
我们这里就直接照搬head first上的案例进行讲解吧。这个案例是啥呢，稍微做下介绍，气象站会实时提供温度，湿度，气压，要求，要实现几个布告板，需要在温度，湿度，气压更新时也实时更新。气象站会实时更新时调用一个方法，具体怎么调用不管，我们只需要知道它会实时调用，我们的模拟实现里面也一样模拟这个调用过程。

所以说有几个布告板想要实时更新同样的数据，而有个地方要实时发布数据，这不就是了然的就是观察者模式嘛，篇幅原因，这里我只实现一个观察者，下面上关系图。



很清晰的首先有一个主题接口和一个观察者接口，至于第三个接口是用于显示的，WeatherData则是具体的主题，CurrentConditionDisplay则是具体的一个观察者。

我们先依次观察下三个接口的代码。

Subject

```
public interface Subject {

    void registerObserver(Observer o);// 注册观察者

    void removeObserver(Observer o);// 取消观察者

    void notifyObservers();// 通知观察者

}
```

Observer

```
public interface Observer {

    void update(float temp, float humidity, float pressure);// 更新数据，温度，湿度，气压
}
```

DisplayElement

```
public interface DisplayElement {

	void display();// 显示
}
```

然后我们看看具体的主题WeatherData

```
public class WeatherData implements Subject {

	private List<Observer> observers;// 观察者集合
	private float temp;// 温度
	private float humidity;// 湿度
	private float pressure;// 气压

	public WeatherData() {
		observers = new ArrayList<Observer>();
	}

	@Override
	public void registerObserver(Observer o) {
		observers.add(o);
	}

	@Override
	public void removeObserver(Observer o) {
		int i = observers.indexOf(o);
		if (i >= 0)
			observers.remove(i);
	}

	@Override
	public void notifyObservers() {
		for (int i = 0; i < observers.size(); i++) {
			Observer observer = observers.get(i);
			observer.update(temp, humidity, pressure);
		}
	}

	public void measurementsChanged() {
		notifyObservers();
	}

	public void setMeasurements(float temp, float humidity, float pressure) {
		this.temp = temp;
		this.humidity = humidity;
		this.pressure = pressure;
		measurementsChanged();
	}

}
```

可以看到我们定义了三个通知属性（温度，湿度，气压），还有一个观察者的集合（这个就是存档订阅的观察者的）。嗯，操作上代码里都比较清晰，我就不详细说了，提一下setMwasurements这个方法，上游会在数据更新时调用measurementsChanged，但我们这里肯定没有，所以写这个方法是模拟这个过程。

最后展示下CurrentConditionDisplay

```
public class CurrentConditionDisplay implements Observer, DisplayElement {

    private float temp;
    private float humidity;
    private Subject weatherData;

    public CurrentConditionDisplay(Subject weatherData) {
        this.weatherData = weatherData;
        weatherData.registerObserver(this);
    }

    @Override
    public void display() {
        System.out.println("当前温度" + temp + "°，当前湿度" + humidity);
    }

    @Override
    public void update(float temp, float humidity, float pressure) {
        this.temp = temp;
        this.humidity = humidity;
        display();
    }

}
```

可以看到这里的构造器会注入主题接口的对象并完成注册，这样一个观察者就注册进去了，因为只对接接口，所以和具体主题不存在耦合。