一、引入实践项目

Internet气象站项目：

提供温度、气压和湿度的接口

测量数据更新时通知给第三方

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

通常的设计方案：

|  |
| --- |
| **CurrentConditions** |
| Update()  Display() |

|  |
| --- |
| **WeatherData** |
| getTemperature()  getHumidity()  getPressure()  dataChange() |

1. 引入问题
2. 其他第三方公司接入气象站获取数据的问题
3. 无法动态的添加第三方

获取气象站信息的公司有很多，即有很多类似CurrentConditions的类，每增加一个类，WeatherData类就需增加一个接口，这样很大降低了代码的可维护性和可扩展性。

1. 解决方式

采用接口和实现的方式，将变化的部分抽取成接口，即第三方公司获取气象站信息的类是变化的，写一个接口，继而写可以注册和移除第三方公司的方法，气象站有新的数据的时候，会通知那些之前注册的第三方公司，没有注册和移除的公司将不会收到气象站的数据。

即使用观察者模式。

|  |
| --- |
| <interface>  subject |
| registerObserver() //注册  removeObserver() //移除  notifyObservers()//通知 |

|  |
| --- |
| <interface>  Observer |
| update() //更新 |

观察者模式：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。

用观察者模式设计上面的方案：（代码见附件）

|  |
| --- |
| <interface>  subject |
| registerObserver() //注册  removeObserver() //移除  notifyObservers()//通知 |

|  |
| --- |
| <interface>  Observer |
| update() //更新 |

|  |
| --- |
| WeatherDataSt |
| registerObserver() //注册  removeObserver() //移除  notifyObservers()//通知  GetTemperature()  getHumidity()  getPressure()  dataChange() |

|  |
| --- |
| CurrentConditions |
| update() //更新  display() |

附件：

1. Subject类

public interface Subject {

public void registerObserver(Observer o);

public void removeObserver(Observer o);

public void notifyObservers();

}

2、Observer类

public interface Observer {

public void update(float mTemperatrue,float mPressure,float mHumidity);

}

3、WeatherDataSt类

public void registerObserver(Observer o) {

mObservers.add(o);

}

@Override

public void removeObserver(Observer o) {

if(mObservers.contains(o)){

mObservers.remove(o);}

}

@Override

public void notifyObservers() {

for(int i=0,len=mObservers.size();i<len;i++){

mObservers.get(i).update(getTemperature(),getPressure(),getHumidity());

}

}

GetTemperature(){}

getHumidity(){}

getPressure(){}

setData{}

1. CurrentConditions类

public void display()

{

System.out.println("\*\*明天温度:"+(mTemperatrue+Math.random())+"\*\*");

System.out.println("\*\*明天气压:"+(mPressure+10\*Math.random())+"\*\*");

System.out.println("\*\*明天湿度:"+(mHumidity+Math.random())+"\*\*");

}

public void update(float mTemperatrue, float mPressure, float mHumidity) {}