**设计模式 实验2-观察者模式**

一.简答题（共1题）

设有一股票，其价格在一定范围内波动，股民需要三种报告：股票的当前价格；股票价格的统计分析报告（最低价、最高价、均价）；股票价格趋势预测。请运用观察者模式，编写上述股票行情程序（模拟股票价格的变动）。编写实验报告，要求画出相应的类图，提供java源程序。

提示：股价变动用随机数模拟：java.util.Random

//引入Random类：

import java.util.Random;

//在程序段中：

final Random random = new Random();

int x;

float mm;

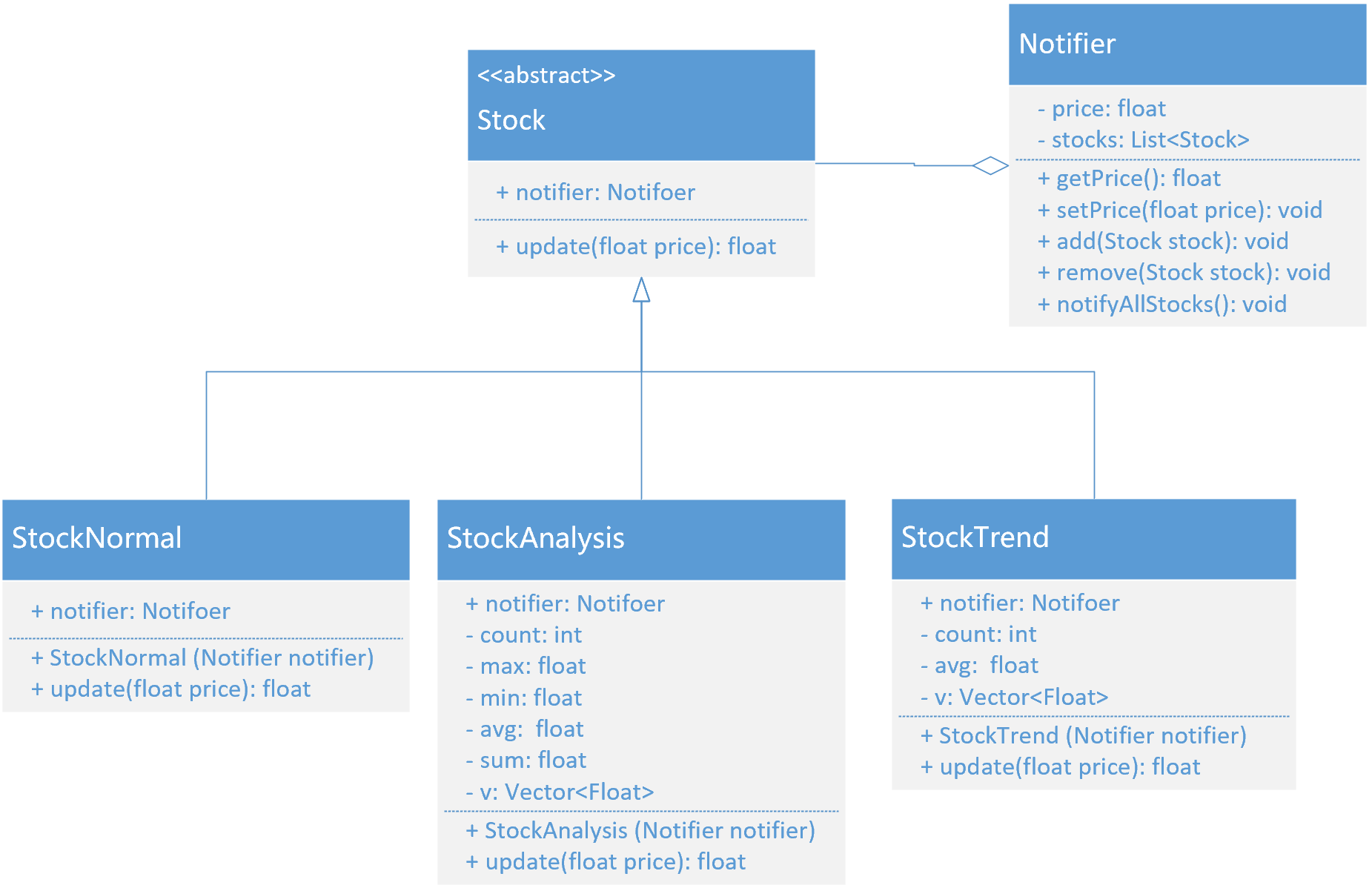
for(int i=0;i<10;i++){

mm=random.nextFloat()\*100; //浮点数

x=random.nextInt(50); //整数

}

1. 类图



1. 源程序



①创建观察者抽象类，维持一个指向通知者的引用，声明股票价格变动方法update

/\*\*  
 \* 抽象观察者 股票  
 \*/  
public abstract class Stock {  
 protected Notifier notifier;  
 public abstract void update(float price);  
}

②创建具体观察者，继承抽象类，实现不同的股票报告方式  
/\*\*  
 \* 具体观察者 股票当前价格  
 \*/  
public class StockNormal extends Stock {  
 public StockNormal(Notifier notifier){  
 this.notifier=notifier; //维持一个指向通知者的引用  
 this.notifier.add(this);  
 }  
 @Override  
 public void update(float price) {  
 System.out.println("股票当前价格:");  
 System.out.println("¥"+price);  
 }  
}

/\*\*  
 \* 具体观察者 股票价格的统计分析报告(最低价、最高价、均价)  
 \*/  
public class StockAnalysis extends Stock {  
 //存储股票数量  
 private int count;  
 //存储股票最高价、最低价、均价、总价  
 private float max,min,avg,sum;  
 //存储股票价格  
 private Vector<Float> v=new Vector<Float>();  
 public StockAnalysis(Notifier notifier){  
 count=0;  
 sum=0;  
 this.notifier=notifier; //维持一个指向通知者的引用  
 this.notifier.add(this);  
 }  
 @Override  
 public void update(float price) {  
 v.add(price);  
 if(v.size()==1){  
 max=price;  
 min=price;  
 } else {  
 if(price>max) max=price;  
 if(price<min) min=price;  
 }  
 ++count;  
 sum+=price;  
 avg=sum/count;  
 System.out.println("股票分析报告:");  
 System.out.println("最高价 ¥" + max);  
 System.out.println("最低价 ¥" + min);  
 System.out.println("平均价 ¥" + avg);  
 }  
}

/\*\*  
 \* 具体观察者 股票价格趋势预测  
 \*/  
public class StockTrend extends Stock{  
 //存储股票数量  
 private int count;  
 //存储股票均价  
 private double avg;  
 //存储股票价格  
 private Vector<Float> v=new Vector<Float>();  
 public StockTrend(Notifier notifier){  
 count=0;  
 this.notifier=notifier; //维持一个指向通知者的引用  
 this.notifier.add(this);  
 }  
 //采用近三天均价预测  
 @Override  
 public void update(float price) {  
 v.add(price);  
 int n=v.size();  
 if(n==1){  
 avg=price;  
 } else if(n==2) {  
 avg=(v.get(0)+v.get(1))/2;  
 } else {  
 avg=(v.get(n-1)+v.get(n-2)+v.get(n-3))/3;  
 }  
 System.out.println("股票价格趋势预测:");  
 System.out.println("¥"+avg);  
 }  
}

③编写通知者主题类Notifier  
/\*\*  
 \* 通知者  
 \*/  
public class Notifier {  
 //ArrayList存放多种股票报告  
 private List<Stock> stocks = new ArrayList<Stock>();  
 //股票价格变动状态  
 private float price;  
 public float getPrice() {  
 return price;  
 }  
 //更新股票价格状态  
 public void setPrice(float price) {  
 this.price = price;  
 notifyAllStocks();  
 }  
 //增加股票报告  
 public void add(Stock stock) {  
 if(!stocks.contains(stock))  
 stocks.add(stock);  
 }  
 //删除股票报告  
 public void remove(Stock stock) {  
 if(stocks.contains(stock))  
 stocks.remove(stock);  
 }  
 public void notifyAllStocks(){  
 for (Stock stock : stocks) {  
 stock.update(price);  
 }  
 }  
}

④模拟10次股票价格变动。价格变动时，使用Notifier向所有股票价格报告通知价格变动信息并更新报告

/\*\*  
 \* 客户端 模拟股票价格的变动  
 \*/  
public class Client {  
 public static void main(String[] args) {  
 Notifier notifier = new Notifier();  
 new StockNormal(notifier);  
 new StockAnalysis(notifier);  
 new StockTrend(notifier);  
 final Random random = new Random();  
 float price;  
 for(int i=0;i<10;i++){  
 price=random.nextFloat()\*100;//[0.0,100.0)  
 //股票价格变动 通知所有股票报告并更新  
 System.out.println("估价第"+(i+1)+"次变动-----------------------");  
 notifier.setPrice(price);  
 }  
 }  
}

1. 运行结果截图

