LAB 11

Introduction to Computer Science
CSI201
Fall 2012

Financial Markets Version Do this or the other version.

SUMMARY

In this lab you will create new class, and its associated objects and methods. The class will represent some aspects of trading in a stock of a company. In the last generation, much of the stock markets where these are traded has moved to computers. And recently, computers have been programmed to do the buying and selling automatically, cutting out the human entirely. As you might imagine, this occasionally does not work out well. For instance, in May 2010, the stock market nosedived briefly in reaction to a computer's trading activities (the infamous "Flash Crash"). In reaction to this, the stock markets have instituted safeguards. One of these is called a "circuit breaker." In a circuit breaker, if a stock's price moved up or down more than a certain percentage (e.g. 10%), the market will stop trading in that stock, under the assumption that something bad is happening.

Each instance of your new class, Stock, will represent the information that might have be used to make circuit breaker decisions.

PRE-LAB

You should have reviewed your notes and/or the web material from Lecture 22, Monday, Nov. 19, and begun to read chapter 11 in G&E's book.

IN-LAB

STOCK

In this lab you will **create a class** to be the blueprint for an object representing a trading history in a stock, which will contain an array of type **`double`**. This array will represent the prices of the various trades in this stock. The array will be big enough to hold at least one thousand prices. In addition, there will be an integer in the object to keep track of the number of trades recorded in the array. You will also write a main method that uses an instance of your **`Stock`** class.

You will do this by ***refactoring*** the **`StockPricePlotter`** class that was live coded on Monday, Nov. 19. Begin by copying the original and the data files into a fresh directory for Lab11. Get them from the Web outside of lab; inside of lab get them by:

```
cd CSI201 #or other name for your CSI201 folder/dir.
# begins a comment in Unix shell languages.
mkdir Lab11
cd Lab11

cp /usr/local/depts/cs/geintro/Fal12Lab11/*    •
/usr/local/depts/cs/geintro/drjava
```

VERIFY IT COMPILES AND RUNS BEFORE YOU GO ON!!!

FIELDS

Your new class should be named **`Stock`**. This class should specify THREE (3) fields:

- An array of **`doubles`** referred to by **`dailyPrices`**
- An **`int`** named **`numberOfDaysWithData`**
- A double named **`lastPChange`**

CONSTRUCTORS

(We will cover constructors officially in the lectures, and after that, you will make constructors in future labs.)

METHODS

Add the following methods in your **`Stock`** class:

`void addTrade(double price)`

Refactor: Move the code

```
dailyPrices[arrayIndex]
    = currPrice;
arrayIndex=arrayIndex+1;
this.numberOfDaysWithData++;
```

from the **`readAndPlot`** to your **`addTrade`** method. Of course, you must REPLACE the now out-of-scope variable **`currPrice`** with the parameter variable so your code records (in an element of the array) the price stored in the parameter **`price`**.
And, you must use the appropriate field of the class (**`numberOfDaysWithData`**) instead of **`arrayIndex`** to locate **WHERE** to store the latest added price.

Finally, add functionality:
UPDATE the field `lastPChange` according to the specification of what **`lastPChange`** is supposed to mean**.**

`int numTrades()`

This method should return the current number of trades in the array for this stock.

`double lastPriceChange()`

If there are two or more prices in the array, it will return the difference between the last two: (previous – last). If there are fewer than 2 items in the array, the method should return zero. It should use the value stored in **`lastPChange`**.

`void checkBreakerAndPrint()`

Check the breaker condition (see the summary) for the last added trade, and, if the breaker trips with a 10% or more change, up or down, PRINT a scary message of your choice!

Followup: Also, make it plot in RED the bar representing the price that it changed TO, for each time it changed by 10% or more, up or down.