Term Assignments, CIS 234: Introduction to Java Programming

Per our syllabus, the fundamentals of object oriented java programming will be taught over the course of 4 assignments. All will require individual programming and concrete deliverables, with time-sensitive deadlines.

Assignment One

The premise of the term assignment is that we operate an electronic stock market, where shares in companies are bought, sold and priced. The market is a digital service which accepts orders, human and machine-originated.

The sales price of a stock is set by a seller, in relation to demand. Hence, as few orders come in, prices may discount to stimulate buying.

Your initial market must do the following.

1. **Market setup**. Six thousand unique stocks must open, with randomly determined ticker symbols and start pricing. Each symbol, price, buy volume and seller is to be entered into an ArrayList, called the *market*. Symbols may have up to four letters.

2. **Symbol lookup**. A buyer enters a symbol by its four letter name, then receives a price, *last sale* information, and *shares outstanding*.
   - This requires for a volume of shares to be specified, when the stock is created, stating how many shares are available.
   - When a buy takes place, the number of available shares should decrease accordingly.

3. **Purchase order**. After lookup takes place, a buy is placed, updating its price within the ArrayList item associated with it. The updated price also becomes the *last sale* information, as needed in #2.
   - The buyer also demands a certain number of shares. When a buy takes place, the number of available shares should decrease accordingly.

Strategies

- ArrayList and the Stock class are helpful tools to use together. Rely upon the get, set methods to retrieve data you need from each stock. The way to iterate, or traverse the array list is to use a for-loop, wherein you can access the value of any given stock.

- During the market's creation, several thousand stocks are intialized, each with unique pricing and symbols, determined randomly. Use a similar, but simpler loop to search the index, by symbol name. This would require a Scanner object to take input from the user, which you can compare to any given stock's symbol, using getSymbol, or similar method within Stock

- Public versus Private variables inside of Stock. **`Public`** variables do not need get/set accessor methods within your stock class. They can be accessed using dot notation in your main class:

```
Stock n = new Stock();
System.out.println("stock's symbol is " + s.symbol);
```

Deliverable:

Test each feature completely. Copy a screen shot of the console output to a Word document. Print the source code with the screen shot, name and label "Assignment One." Submit, printed by the due date.

Assignment Two

Add a large group of **Agents** into the simulation. An **Agent** can buy and sell stocks. For assignment two, we'll accomplish the buy portion of their identity. *We can casually refer to each Agent as a buyer, or investor.*

Each Agent:
Begins with a portfolio of stocks, which they purchase automatically upon initialization. Each Agent has a budget, of random size, from which they execute stock transactions.

- Each Agent must decide how many transactions to make, then divide their budget appropriately, executing a buy from the market's index for each transaction.
- In order to buy a stock, an Agent must generate a random number, then choose a stock at that index value, from the market index. Each stock is bought at its current market price until all the Agent's initial money is spent. If there is some money left over, it remains in the budget.
- After a transaction takes place, have the market increase the price of the stock by some small but random amount.
- Each Agent will need a buyer id, which is a randomly generated string of numbers which help identify each Agent.

New features for the stock class

- Each stock will contain an array which retains the history of its price. This is know as a price history. The price history also tracks the volume (or amount of stock) sold for each transaction.
- Supply a method to print the price history
- Supply a method to calculate the difference between the last two prices (how much it is up or down)

Other requirements
Create as many buyers as your laptop or personal computer will tolerate. Having thousands of buyers is a good thing.

Given the fact that the buyers will be purchasing the stocks (and not the user of the application), you may stop printing the market index. Instead, you may wish to print each buyer as they come online, then populate their stock portfolio with transactions.