Homework #5

CS 210, **Fundamentals of Computer Science I**
Instructor: Tatiana Harrison

BELLEVUE COLLEGE COMPUTER SCIENCE

This assignment covers methods and classes, which are discussed in lectures. This homework also assumes understanding of the concepts that are covered from the beginning of the course. The one programming exercise is worth 70 points. If you need help ask Tatiana, email, come to office hours, etc. The rubric for this assignment is available canvas.

What to hand in:

Complete "Homework 5 Q&A" quiz on Canvas – 30 points

For the programming task, upload two **.java** files to your Canvas account. The rubric has been posted to the course website.

Programming Question – StockMarket and Trader.java - 70 points

This programming section of this assignment requires you to submit **TWO** .java files. Please read all of the instructions, before you begin writing your code. Details were also provided during lecture.

For this programming question, you'll be given a skeleton of a program, `StockMarket.java`, that contains pseudocode ONLY. The java file is well documented. Follow the instructions in the java file (from top to bottom), to write the code needed to finish this programming task. You'll also need to write a second file, called `Trader.java`. It is a class representing a person at a StockMarket, a trader. The `Trader.java` class has 4 fields, 6 methods (two of which take an argument, and four of which do not), and a non-default constructor, that takes a single argument.

The setup is the following: The `StockMarket.java` program instantiates four objects of the class `Trader`, and for each of them declares a reference variable. The `Trader` class has fields for a person's name, his/her stocks gain percentage percentage (stockChangePercentage) (how "lucky" the Trader is, ranging from 0 (always loses) to 1 (always makes money)). In the `StockMarket.java` program, a while loop, with true as the condition, repeatedly prints out each Trader's name, how much money he/she has spent, how much money he/she currently has, and the Trader's net gain (or loss). At each iteration of the while loop, the program asks, "How much money should each Trader spend? Enter 0 to quit." If the user enters, via keyboard, 0, then a break statement is invoked, and the while loop (and hence program) terminates. If a non-zero value is entered, then the `tradeSomeMore` method for each of the four `Trader` objects is invoked, which either increments or decrements each trader's money made, based on whether a randomly generated decimal number is more or less than a trader's stock change percentage. See the skeleton of `StockMarket.java`, which contains pseudocode of the entire program.

The `Trader.java` class has four fields: `traderName`, `stockChangePercentage`, `dollarsSpent`, and `dollarsMade`. The class has a method, `SetStockChangePercentage`,

that takes as input the single argument `percentage` of type `double`. The class has a second method, `tradeSomeMore`, that takes as input the single argument `dollars`, of type `double`. The method `tradeSomeMore`:

- generates a random number
- checks if that number is more or less than the Trader's `stockChangePercentage`; depending on the outcome, the method prints out to the console “made” or “lost”. If a Trader makes money a round, the Trader's `dollarsMade` is incremented by 2 times the Trader's dollars. If a trader losses, they loose 3 times the amount od dollars being traded.

Five additional methods, `getTraderName`, `getDollarSpent`, `getStockChangePercentage`, `getDollarsMade`, and `getNetProfits`, take no input parameters, and return a `String`, `double`, `double`, `double`, and `double`, respectively. The `Trader` class non-default constructor takes a single argument, `name`, of type `String`. See the skeleton of `Trader.java`, which explains each field, and the methods, in general. The UML diagram for the Trader class is shown below:

Trader
- traderName : String - stockChangePercentage : double - dollarsSpent : double - dollarsMade : double
+ getStockChangePercentage() : double + setStockChangePercentage(stockChangePercentage : double) : void + tradeSomeMore(dollarsBet : double) : void + getTraderName() : String + getDollarsSpent() : double + getDollarsMade() : double + getNetProfit() : double + Trader(name : String) :

A sample invocation of the program is shown in Figure 1, in which there are three rounds of playing the stock market. In the first round, $300 is bet by each player. In the second and third rounds, $160 and $1000 is bet by each player. Each player has a different chance of making money, and by the end of the three rounds of trading, two of the Traders have a negative net gain, and the player Happy, has a net of $800.

Use Traders names and percentages used in this sample!

List inputs that break code at the top of your StockMarket file.

```
Here are the Traders, and their stock change percentages:
Lucky's stock change percentage : 0.61
Grumpy's stock change percentage : 0.19
Happy's stock change percentage : 0.81
Sleepy's stock change percentage : 0.42
```

```
Trader Lucky; spent: $0.0; current $1000.0; net $1000.0
Trader Grumpy; spent: $0.0; current $1000.0; net $1000.0
Trader Happy; spent: $0.0; current $1000.0; net $1000.0
Trader Sleepy; spent: $0.0; current $1000.0; net $1000.0
 How much money should each Trader bet? Enter 0 to quit 300
Lucky lost money
Grumpy lost money
Happy made money
Sleepy made money
Trader Lucky; spent $300.0; current $100.0; net $-200.0
Trader Grumpy; spent $300.0; current $100.0; net $-200.0
Trader Happy; spent $300.0; current $1600; net $1300.0
Trader Sleepy; spent $300.0; current $1600.0; net $1300.0
How much money should each Trader bet? Enter 0 to quit 160
Lucky lost money
Grumpy lost money
Happy lost money
Sleepy made money
Trader Lucky; spent $460.0; current $-1280.0; net $-1740.0
Trader Grumpy; spent $460.0; current $-1280.0; net $-1740.0
Trader Happy; spent $460.0; current $220.0; net $-240.0
Trader Sleepy; spent $460.0; current $2520.0; net $2060.0
How much money should each Trader bet? Enter 0 to quit 120
Lucky lost money
Grumpy lost money
Happy made money
Sleepy made money
Trader Lucky; spent $580.0; current $-3020.0; net $-3600.0
Trader Grumpy; spent $580.0; current $-3020.0; net $-3600.0
Trader Happy; spent $580.0; current $1380.0; net $800.0
Trader Sleepy; spent $580.0; current $3680.0; net $3100.0
How much money should each Trader bet? Enter 0 to quit 0
```

Figure 1: sample output of the program StockMarket.java