HW 01: CS2012 Review

Due Feb 8, 2022 by 2:55pm **Points** 1 **Submitting** a file upload **File Types** zip
Available until Feb 8, 2022 at 2:55pm

This assignment was locked Feb 8, 2022 at 2:55pm.

CS-2013 Programming with Data Structures

Homework 01: CS-2012 Review Assignment

Required Files

- **Historic Stock Data for Tesla (https://csns.cysun.org/download?fileId=7890518)**
  - Right-click, save-as to your desktop, drag to the correct location in your Eclipse project.
  - This is a csv (comma-separated values) file which lists the historic stock price data for the Tesla company.
  - csv Files can be opened in a spreadsheet software like Microsoft Excel or LibreOffice Calc. They can also be opened in a text editor like Notepad++, Sublime, or Atom.
  - I recommend that you open them in a text editor so you can see the actual format of the file.
  - Each line contains the following format:
    - date, opening_price, high_price, low_price, closing_price

General Description

For this assignment you will read and process historic stock data for the Tesla company.

Project Setup and Structure

- Create a **new java project** in Eclipse and call it **homework01.**
- Inside the **src** folder, create a **new java package** called **hw01**.
  - All of your source code files will need to be placed in the hw01 file.
- Inside the **src** folder, create a **new folder** called **files**.
  - Your .csv files should go in **files**.
- NOTE: When working with the .csv file, your program must use a **relative path** to the .csv file, NOT an **absolute path.** If you do not know the difference between the two, you will need to review on your own and do some research as to what these mean.

- An absolute path will only work on your system, and will not work on my system. This means I will not be able to correctly execute your program and you will receive a 0 for the assignment.

`TeslaStock` Class

- Create a class called `TeslaStock`. This class will be used to represent each stock's data in your project. This class should not contain any methods to read / write the data. Only implement the following in the class.
- This class should contain data fields for the date, opening price, low price, high price, and closing price of a stock. **All data fields shall have the PRIVATE access modifier.**
  - NOTE: Be sure to use the correct data types for each piece of data.
- Create getters and setters for each data field.
- Create appropriate constructors for this class.
- Implement the **toString()** method to display a `TeslaStock` object in the following format:

```
Date:            2010-06-29
Opening Price:   3.8
High Price:      5.0
Low Price:       3.508
Closing Price:   4.778
```

- No other methods are allowed.

`StockList` Class

- This class will contain the methods necessary to display various information about the stock data.
- This class shall be a **subclass** of **ArrayList<TeslaStock>.**
- This class shall **not** contain any **data fields**. (You don't need them for this class if done correctly).
- This class shall have a **default constructor** with no parameters.
  - No other constructors are allowed.
- This class shall have the following **public, non-static,** methods:
  - **printAllStocks:**
    - This method shall take no parameters and return nothing.
    - This method should print each stock in the StockList to **an output file**.
      - NOTE: This method shall **NOT** print the data to the console.
    - The output file shall be called **all_stock_data.txt.**
    - The data should be printed in a nice and easy to read format.
  - **displayFirstTenStocks:**
    - This method shall take no parameters and return nothing.

        - This method should display only the first 10 stocks in the StockList in a nice and easy to read format.
        - This method should display the data in the console.
    - **displayLastTenStocks:**
        - This method shall take no parameters and return nothing.
        - This method should display only the last 10 stocks in the StockList in a nice and easy to read format.
        - This method should display the data in the console.
    - **computeAverageOpeningPrice:**
        - This method shall take no parameters and return a **double** value.
        - This method should compute and return the average opening price of all stock data in the StockList.
    - **computeAverageClosingPrice**:
        - This method shall take no parameters and return a **double** value.
        - This method should compute and return the average opening price of all stock data in the StockList.
    - **findMaxHighPrice**:
        - This method shall take no parameters and return a single **TeslaStock** object.
        - This method should compute and return the maximum high price of all the stock data.
    - **findMinLowPrice**:
        - This method shall take no parameters and return a single **TeslaStock** object.
        - This method should compute and return the minimum low price of all the stock data.
- No other methods are allowed or necessary.

`StockReader` Class

- This class will contain the method necessary to read and parse the .csv file of stock information.
- This class shall contain a default constructor with no parameters. The constructor should be made **private**.
    - This is to prevent the class from being instantiated since the class will only contain static methods.
- This class shall contain **no data fields.**
- This class shall have a method called **`readStockData`**:
    - This method shall be a **`public`**, **`static`**, method.
    - This method shall return a **`StockList`** object once all data has been processed.
    - This method will take a **`File`** object as a parameter. This **`File`** object should link to the stock market data in your files folder created previously.
    - This method must validate that the given **`File`** object is a .csv file. If it is not, then this method shall throw an **`IllegalArgumentException`**.

- This method shall read the `File` object and process all of the stock data into `TeslaStock` objects and store each object in a `StockList`.
  - NOTE: The data must be parsed so each piece of data is correctly assigned to its respective data field in the `TeslaStock` class.
  - NOTE: You will want to ignore the first line of data as this line contains the header labels for each column of data.
- No other methods are allowed.

`Main` Class

- This class shall be called **Main** and will contain your **main()** method.
- This class shall do the following:
  - Create a File object linked to your .csv file.
    - NOTE: You do not need to ask the user for the name of the file, you can just hardcode the relative path name into your source code.
    - NOTE: Remember to use a relative path and setup your project correctly. Otherwise I will not be able to run your program.
  - Read all the stock data from the file and obtain a StockList object.
  - Execute each method from the StockList class and show the results of each method.
    - NOTE: For the printAllStocks method, you can show that the file was correctly created and open it during your video presentation.
  - Be sure that your output is neatly formatted and not a jumbled mess. Presentation of output counts towards your grade.

Video Presentation Checklist:

The following is a checklist of everything I would like you to demonstrate / explain in your video.

1. Show the code of your **TeslaStock** class.
   1. Show that each data field exists, is the correct data type, and is private.
   2. Show that you have an appropriate constructor.
   3. Show there is a getter and setter for each data field.
   4. Show that you have correctly implemented the **toString()** method.
2. Show your **StockReader** class and **explain** how the code for the **readStockData()** method works. Your explanation should be clear an concise.
3. Show that your **StockList** class is a subclass of ArrayList<TeslaStock>.
4. Run your program and demonstrate the following:

1. Your **printAllStocks()** method from the **StockList** class works by running your program to generate a new output file. Open the output file in a text editor and quickly scroll through the file.
2. Your program correctly displays the first 10 and last 10 stocks in the **StockList**.
3. Your program correctly displays the average opening and closing prices.
4. Your program correctly displays the maximum high price.
5. Your program correctly displays the minimum low price.

You should be able to do all of this in no more than 5-7 minutes. Less than 5 minutes is OK.

Deliverables

- Compress your **src** into a .zip file.
  - If done correctly, this .zip file will contain all the necessary files that I need to grade your assignment.
  - Upload your .zip folder to Canvas before the due date and time.
- Be sure to upload a .zip file of your **src** folder to Canvas by the due date and time.
- **DOCUMENTATION:** You are required to document your code for this assignment according to the canvas page and using /**Javadoc comments*/. See the example Canvas page on how to write proper javadoc comments.
- **VIDEO:** You should record a video presentation of your project according to the checklist given above.
  - Upload your video to your canvas studio space. (If using Canvas Screen and Webcam Recorder, this should be done for you automatically.)
  - Go into the video on Canvas and choose the **Share** option, then choose **"Create public link".**
  - Copy and paste the public link as a comment on the Canvas upload homework interface.