ITCS 209 Object Oriented Programming	Name:	Lab Score	Challenge Bonus
	ID:		
	Section:		

Lab 12: File Management and Regular Expression

Topic: **Zoom Stock Price (All Time)**

In this assignment, you will write Java code to analyze data from a CSV (Comma-Separated Values) file. The CSV file contains data on the daily stock prices of Zoom Video Communications, Inc., and your task is to read in the data, manipulate it, and generate some useful statistics. You will also use regular expressions to perform additional data validation and manipulation.

Choose one of the three tasks provided for the required assignment, and the challenge task is optional for bonus points.

Dataset:

You can download the CSV file containing the data for this assignment from the course website. The file is called "zoom_stock_price_all_time.csv".

Task 1: Data Reading and Validation

In this task, you will write Java code to read in the data from the CSV file and validate it to ensure that it is in the correct format. The data should contain seven columns: Date, Open, High, Low, Close, Adj Close, and Volume. Each row represents a daily trading day of Zoom Video Communications, Inc.

Implementation Guideline :

1. Import the necessary packages: BufferedReader, FileReader, and IOException.
2. Define the CSV file path as a constant variable.
3. Define the delimiter used in the CSV file as a constant variable.
4. In the main method, use a try-catch block to handle potential IOExceptions.
5. Instantiate a new BufferedReader and FileReader object to read the CSV file.
6. Read the header line using the readLine() method.
7. Check if the header line is null, which indicates an empty CSV file. If so, print an error message and exit the program.
8. Split the header line by the CSV delimiter and store the resulting array in a variable.
9. Check if the length of the header array is 7 and that each element corresponds to the expected column header. If not, print an error message and exit the program.
10. Define a counter variable for the row number, and use a while loop to read each subsequent line in the CSV file.
11. Split the line by the CSV delimiter and store the resulting array in a variable.
12. Check if the length of the array is 7, which indicates that it contains the correct number of fields. If not, print an error message that includes the row number, and exit the program.

13. Iterate through the fields array for columns 1 through 6 and try to parse each value as a double. If any of the values cannot be parsed, print an error message that includes the row number and column number, and exit the program.
14. Increment the row number and continue to the next line in the CSV file.
15. Close the reader object, and print a message indicating that the CSV file is valid and ready for analysis.

Rubric:

- Java code successfully reads in the data from the CSV file.
- Java code validates the data to ensure that it contains seven columns and that each row has a valid format.
- Java code handles any errors or exceptions that may occur during the data reading and validation process.
- Java code is well-written and easy to understand.

Submission: Submit your Java code in a single file named ZoomStockFileValidator.java.

Task 2: Data Manipulation and Analysis

In this task, you will write Java code to manipulate and analyze the data from the CSV file. You should calculate the following statistics:

- **Total trading volume for each year.**
- **The highest and lowest stock price values for each year.**
- **The number of days that the stock price increased, decreased, or remained unchanged for each year.**
- **You should also calculate the overall total trading volume, as well as the overall highest and lowest stock price values for the entire dataset.**

[Note : In finance, "close" refers to the closing price of a stock or security at the end of a trading day, while "adj close" refers to the adjusted closing price.

The closing price is the last traded price of a stock at the end of the trading day. It is the price that is most used in financial analysis, such as calculating the daily returns or volatility of a stock.

The adjusted closing price, on the other hand, takes into account any corporate actions that may have affected the price of the stock, such as stock splits, dividends, or stock buybacks. For example, if a stock undergoes a 2-for-1 stock split, the adjusted closing price would be half of the original closing price, since there are now twice as many shares outstanding.

The adjusted closing price is often used in long-term analysis, as it provides a more accurate picture of the historical performance of a stock, taking into account any corporate actions that may have affected its price over time. *** Taking the maximum value between the closing price and the adjusted closing price. *****]**

Implementation Guideline :

1. Initialize variables for statistics, including total volume, overall high, and overall low.

2. Loop through each year from 2019 to 2021.
3. Initialize variables for year-specific statistics, including year volume, year high, year low, and counts of days with increased, decreased, or unchanged stock prices.
4. Open the CSV file using a BufferedReader and read each line.
5. For each line, split the fields using the CSV delimiter and extract the year, volume, open price, close price, and adjusted close price.
6. If the year matches the current loop iteration, update the year-specific statistics and the overall statistics.
7. When finished looping through all the lines, print out the year-specific statistics for the current year.
8. Continue looping through each year until all years have been processed.
9. After processing all years, print out the overall statistics.

Rubric:

- Java code correctly manipulates and analyzes the data from the CSV file to calculate the required statistics.
- Java code handles any errors or exceptions that may occur during the data manipulation and analysis process.
- Java code outputs the calculated statistics in a clear and organized manner.
- Java code is well-written and easy to understand.

Submission: Submit your Java code in a single file named ZoomStockPriceAnalyzer.java.

Task 3: Regular Expressions

In this task, you will use regular expressions to perform additional data validation and manipulation. You should use regular expressions to:

- **Validate that the date column is in the correct format (e.g. MM/DD/YYYY).**
- **Extract the year, month and day from the date column.**
- **Extract the change in stock price from the Close and Adj Close columns.**

Implementation Guideline :

1. The Java code reads in the Zoom stock price data from a CSV file using a BufferedReader.
2. It initializes variables for tracking the previous Close and Adj Close prices and the maximum change in stock price.
3. For each line of the CSV file, it uses regular expressions to validate the format of the date, Close price, and Adj Close price.
4. If the data is valid, it extracts the year, month, and day from the date column.
5. It then calculates the change in stock price between the current Close and Adj Close prices and the previous Close and Adj Close prices.
6. It determines the maximum change in stock price between the two.
7. It updates the variables for tracking the previous Close and Adj Close prices and the maximum change in stock price.
8. It outputs the date and maximum change in stock price for each line of valid data.
9. Finally, it outputs the overall maximum change in stock price for the entire dataset.

Rubric:

- Java code successfully uses regular expressions to perform data validation and manipulation.
- Java code handles any errors or exceptions that may occur during the regular expression processing.
- Regular expressions are used in a clear and effective manner to achieve the desired results.
- Java code is well-written and easy to understand.

Submission: Submit your Java code in a single file named RegExDataValidator.java.

Challenge Task:

In this optional challenge task, you can use your knowledge of file I/O and regular expressions to implement additional functionality for the program.

- **Use the BufferedReader class to read the CSV file line by line.**
- **Use the java.time library, specifically the LocalDate class and DateTimeFormatter, to handle date parsing and comparison.**
- **Prompt the user for input using the Scanner class, and validate the input.**
- **Perform the analysis within the specified date range and calculate the total trading volume, highest and lowest stock prices.**
- **Display the results on the console.**

Implementation Guideline :

1. Declare and initialize the file path for the CSV file containing the Zoom stock price data.
2. Create a new Scanner object to read input from the user.
3. Prompt the user to enter the start date in the format "MM/dd/yyyy" and read the input as a String.
4. Prompt the user to enter the end date in the same format and read the input as a String.
5. Parse the start and end dates using a DateTimeFormatter object with the pattern "M/d/yyyy".
6. If the end date is before the start date, print an error message and exit the program.
7. Use a try-with-resources block to open the CSV file for reading and create a BufferedReader object.
8. Read the first line of the file (the header) and discard it.
9. Loop through each subsequent line in the file:
    a. Split the line into an array of String fields using a comma delimiter.
    b. Parse the date field as a LocalDate object using the same DateTimeFormatter.
    c. If the date is between the start and end dates (inclusive):
    d. Parse the high and low price fields as double values.
    e. Parse the volume field as a long value.
    f. Update the highestPrice and lowestPrice variables if necessary.
    g. Add the volume to the totalVolume variable.
10. Print the total trading volume, highest stock price, and lowest stock price using formatted printf statements.
11. Close the Scanner and BufferedReader objects.

Submission: Submit your Java code in a single file named ZoomStockQuery.java