# **Programming Assignment Unit 4**

Computer Science, University of the People

CS 1101-01 Programming Fundamentals - CS 1102-01 - AY2024-T2

Instructor, Noman Shihadeh

December 12, 2023

## Stock Data Analysis Project

For this assignment, we were asked to write a program that analyses stock data provided to us in an array of data containing 10 days of stock prices as float values. The analysis is done using four functions as described here:

1. Calculate the average stock price.
2. Find the maximum stock price.
3. Find the occurrence count of a given stock price.
4. Calculate the cumulative sum of all the stock prices.

Source Code:

// StockAnalysis.java

import java.util.ArrayList;

/\*\*

 \* This class provides methods to analyze stock prices,

 \* including:

 \* 1. Calculating average price

 \* 2. Finding maximum price

 \* 3. Counting occurrences of a specific price

 \* 4. Computing cumulative sums

 \*/

public class StockAnalysis {

    /\*\*

     \* Main method to demonstrate the functionality of the methods

     \* and display them to the user in the console.

     \*

     \* @param args Command line arguments.

     \*/

    public static void main(String[] args) {

        // Demo data for array

        float[] stockPricesArray = { 100f, 150f, 200f, 250f, 300f, 350f, 400f, 450f, 500f, 550f };

        // Convert array to ArrayList by copying each element

        ArrayList<Float> stockPricesArrayList = new ArrayList<>();

        for (float price : stockPricesArray) {

            stockPricesArrayList.add(price);

        }

        // Demonstrate calculateAveragePrice method

        float averagePrice = calculateAveragePrice(stockPricesArray);

        System.out.println("Average Stock Price: " + averagePrice);

        // Demonstrate findMaximumPrice method

        float maxPrice = findMaximumPrice(stockPricesArray);

        System.out.println("Maximum Stock Price: " + maxPrice);

        // Demonstrate countOccurrences method with a target price of 300

        float targetPrice = 300f;

        int occurrenceCount = countOccurrences(stockPricesArray, targetPrice);

        System.out.println("Occurrences of price " + targetPrice + ": " + occurrenceCount);

        // Demonstrate computeCumulativeSum method

        ArrayList<Float> cumulativeSums = computeCumulativeSum(stockPricesArrayList);

        System.out.println("Cumulative Sums: " + cumulativeSums);

    }

    /\*\*

     \* Calculate the average price of stocks.

     \*

     \* @param prices Array of stock prices.

     \* @return Average stock price.

     \*/

    public static float calculateAveragePrice(float[] prices) {

        float sum = 0;

        for (float price : prices) {

            sum += price;

        }

        return sum / prices.length;

    }

    /\*\*

     \* Find the maximum price among the stocks.

     \*

     \* @param prices Array of stock prices.

     \* @return Maximum stock price.

     \*/

    public static float findMaximumPrice(float[] prices) {

        float maxPrice = prices[0];

        for (float price : prices) {

            if (price > maxPrice) {

                maxPrice = price;

            }

        }

        return maxPrice;

    }

    /\*\*

     \* Count the occurrence of a specific price.

     \*

     \* @param prices      Array of stock prices.

     \* @param targetPrice Target price to count occurrences of.

     \* @return Count of occurrences of the target price.

     \*/

    public static int countOccurrences(float[] prices, float targetPrice) {

        int count = 0;

        for (float price : prices) {

            if (price == targetPrice) {

                count++;

            }

        }

        return count;

    }

    /\*\*

     \* Compute the cumulative sum of stock prices.

     \*

     \* @param prices ArrayList of stock prices.

     \* @return ArrayList containing cumulative sums.

     \*/

    public static ArrayList<Float> computeCumulativeSum(ArrayList<Float> prices) {

        ArrayList<Float> cumulativeSums = new ArrayList<>();

        float sum = 0;

        for (float price : prices) {

            sum += price;

            cumulativeSums.add(sum);

        }

        return cumulativeSums;

    }

}

Output:

Average Stock Price: 325.0

Maximum Stock Price: 550.0

Occurrences of price 300.0: 1

Cumulative Sums: [100.0, 250.0, 450.0, 700.0, 1000.0, 1350.0, 1750.0, 2200.0, 2700.0, 3250.0]

The main parts of the code:

1. The “main” method is our program entry point, it is also where we build our demo stock data to be passed into the different methods and then we print the results to the user interface.
2. The “calculateAveragePrice” method takes in a single parameter called “prices” and contains a float array. The method then iterates over the array while for each iteration it accumulates the value in the array’s current location into a helper variable. After the loop exits, we return the helper variable as a return value.
3. The “findMaximumPrice” method takes in a single parameter called “prices” and contains a float array. After initializing and creating a local helper variable to the value located in the first position of the array. Next, we iterate over the array while for each item in the array, we check to see it is a value that is greater than the value currently found in our helper variable. If there is, we replace the value in the helper variable with the current array value. Finally, when we exit the loop, we return the value found in the helper variable since it now should hold the largest value found in the array.
4. The “countOccurrences” method takes in two parameters the first, is called “prices” and contains a float array. The second is called “targetPrice” and contains a float. In this method, we first create and initialize a local helper variable which will contain the incrementor that will represent and count the number of times the value “targetPrice” was found in the array. We next iterate through the array each time checking if the current value in the array is equal to the value found in “targetPrice” If yes, we increment the helper variable if not, we continue with no action. At the end of the loop, we return the value in the helper variable that represents the number of times we encountered the value in the array.
5. The “computeCumulativeSum” method takes in a single parameter called “prices” and contains an ArrayList of floats. The first thing we do is to create and initialize a new local variable of ArrayList that we will use and populate with the new values from the calculations we will perform in this method. We will also need another helper variable to represent a running summary of the values as we iterate the list. We now iterate over the list each time adding the current list value to the helper variable and then adding the calculated value into the new local list variable. At the end of the loop, we return the local list value which will contain a running cumulative sum of the list that was passed in the parameters.

## References

Java Language and Virtual Machine Specifications

<https://docs.oracle.com/javase/specs/index.html>

Introduction to Programming Using Java - Version 9.0, JavaFX Edition

<https://math.hws.edu/javanotes/>

Source Code:

