**Montgomery College**

**CMSC 203**

**Assignment 5 Implementation**

Class: CMSC203 CRN 31569

 Program: Assignment 5 Design

Instructor: Ahmed Tarek

 Summary of Description: Sending holiday bonuses to hard-working employees in Retail District #5. The bonuses are calculated based on how much each retail store sold in each category. The retail store with the highest amount sold in a category will receive $5,000. The retail store with the lowest amount sold in a category will receive $1,000. All other retail stores in district #5 will receive $2,000. If a retail store didn’t sale anything in a category, or they have a negative sales amount, they are not eligible for a bonus in that category. If only one retail store sold items in a category, they are eligible to receive only the bonus of $5000 for that category.

 Due Date: 04/11/2023

 Integrity Pledge: I pledge that I have completed the programming assignment independently.

 I have not copied the code from a student or any source.

Student: Daniel Kim

**Part 1: Pseudo Code and UML diagram:**

**HolidayBonus:**

a. calculatedHolidayBonus

i. Initialize a bonus array with the length equal to the number of rows in the data array

ii. Loop through each row in the data array

1. Loop through each column in the row

a. Check if the value in the current cell is greater than 0

i. If it’s the highest value in the column, add the high bonus to the corresponding bonus array element

ii. Else if it’s the lowest value in the column, add the low bonus to the corresponding bonus array element

iii. Else, add the other bonus to the corresponding bonus array element

iii. Return the bonus array

b. calculateTotalHolidayBonus

i. Initialize a bonus array with the length equal to the number of rows in the data array

ii. Initialize a sum variable with the value of 0

iii. Loop through each row in the data array

1. Loop through each column in the row

a. Check if the value in the current cell is greater than 0

i. If it’s the highest value in the column, add the high bonus to the corresponding bonus array element

ii. Else if it’s the lowest value in the column, add the low bonus to the corresponding bonus array element

iii. Else, add the other bonus to the corresponding bonus array element

iv. Loop through each element in the bonus array

2. Add the value of the bonus array element to the sum variable

v. Return the sum variable

**TwoDimRaggedArrayUtility:**

Define the following static methods:

a. getAverage:

i. Initialize count and total to 0

ii. Loop through each row and column in the array, increment count and add the value to the total iii. Return the average (total divided by count)

iii. Return the average (total divided by count

b. getColumnTotal:

i. Initialize total to 0

ii. Loop through each row

1. If the row length is less than or equal to the column or column is less than 0, continue

2. Add the value in the column to the total

iii. Return the column total

i. Initialize highest to 0

ii. Loop through each row and column in the array

1. If the value is greater than the current highest, update the highest

iii. Return the highest value

d. getHighestInColumn:

i.Initialize highest to 0

ii. Loop through each row

1. If the row length is less than or equal to the column or column is less than 0, continue

2. If the value in the column is greater than the current highest, update the highest

iii. Return the highest in the column

e. getHighestInColumnIndex:

i. Initialize highest to 0 and index to 0

ii. Loop through each row

1. If the row length is less than or equal to the column or column is less than 0, continue

2. If the value in the column is greater than the current highest, update the highest and index

iii. Return the index of the highest value in the column

f. getHighestInRow:

i. Initialize highest to 0

ii. Loop through each column in the specified row

1. If the value is greater than the current highest, update the highest

iii. Return the highest value in the row

g. getHIghestInRowIndex:

i. Initialize highest to 0 and index to 0

ii. Loop through each column in the specified row

1. If the value is greater than the current highest, update the highest and index

iii. Return the index of the highest value in the row

h. getLowestInArray, getLowestInColumn, getLowestInColumnIndex, getLowestInRow, and getLowestInRowIndex:

(Similar to their respective “getHighest…” methods, but instead of finding the highest value, find the lowest value)

I. getRowTotal:

i. Initialize total to 0

ii. Loop through each column in the specified row

1. Add the value to the total

iii. Return the row total

j. getTotal:

i. Initialize total to 0

ii. Loop through each row and column in the array

1. Add the value to the total

iii. Return the total

k. readFile:

i. Initialize variables for the array, rows, and currentRow

ii. Open a Scanner to read the file

iii. Count the number of rows in the file and initialize the array with that number of rows

iv. Open another Scanner to read the file again

v. Loop through each row in the file

1. Split the line into columns and initialize the current row with the number of columns

2. Loop through each column

a. Convert the value to a double and store it in the current row and column of the array

3. Increment the currentRow

vi. Close the Scanner and return the array

L. writeToFile:

i. Open a FileWriter with the specified outputFile

ii. Loop through each row in the data array

1. Loop through each column in the row

a. Write the value to the FileWriter, followed by a space if it’s not the last column int the row

2. Write a newline character to the FileWriter after writing all the columns in the row

iii. Close the FileWriter and handle any IOExceptions if they occur

**UML diagram:**

HolidayBonus:

--------------------------------

| HolidayBonus |

--------------------------------

| |

--------------------------------

+ calculateHolidayBonus(data: double[][], high: double, low: double, other: double): double[]

+ calculateTotalHolidayBonus(data: double[][], high: double, low: double, other: double): double

--------------------------------

TwoDimRaggedArrayUtility:

--------------------------------

| TwoDimRaggedArrayUtility |

--------------------------------

| |

--------------------------------

+ getAverage(a: double[][]): double

+ getColumnTotal(a: double[][], c: int): double

+ getHighestInArray(a: double[][]): double

+ getHighestInColumn(a: double[][], c: int): double

+ getHighestInColumnIndex(a: double[][], c: int): int

+ getHighestInRow(a: double[][], r: int): double

+ getHighestInRowIndex(a: double[][], r: int): int

+ getLowestInArray(a: double[][]): double

+ getLowestInColumn(a: double[][], c: int): double

+ getLowestInColumnIndex(a: double[][], c: int): int

+ getLowestInRow(a: double[][], r: int): double

+ getLowestInRowIndex(a: double[][], r: int): int

+ getRowTotal(a: double[][], r: int): double

+ getTotal(a: double[][]): double

+ readFile(f: File): double[][]

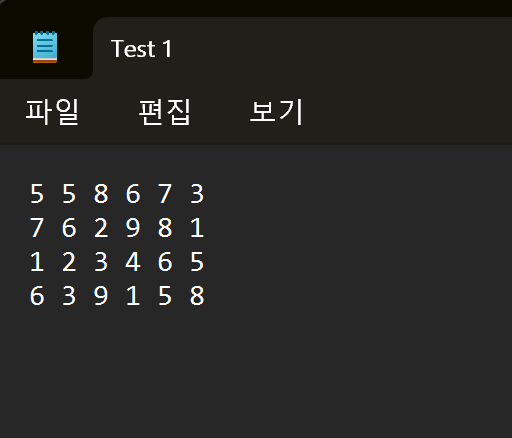
+ writeToFile(data: double[][], outputFile: File)

--------------------------------

**Part 2: Comprehensive Test Plan**

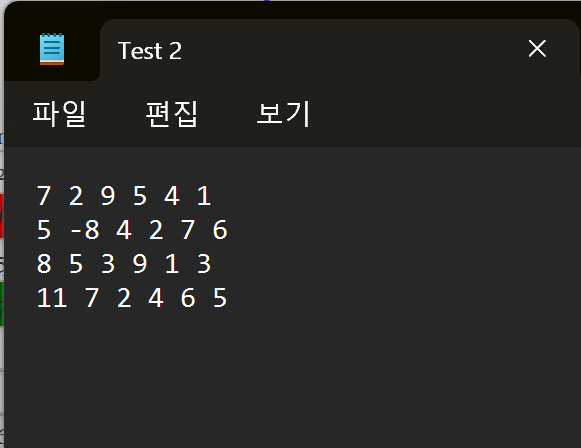
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 | 5 5 8 6 7 3  7 6 2 9 8 1  1 2 3 4 6 5  6 3 9 1 5 8 | Total: 120  Average: 5  Total of row 1:  33  Total of column 0:  19  Highest in row 0:  8  Lowest in column 0:  1  Lowest in Array:  1 | Total: 120  Average: 5  Total of row 1:  33  Total of column 0:  19  Highest in row 0:  8  Lowest in column 0:  1  Lowest in Array:  1 | yes |
| 2 | 7 2 9 5 4 1  5 -8 4 2 7 6  8 5 3 9 1 3  11 7 2 4 6 5 | Total: 108  Average: 4.5  Total of row 1: 16  Total of column 0:  18  Highest in row 2:  9  Lowest in column 2:  2  Lowest in Array:  -8 | Total: 108  Average: 4.5  Total of row 1: 16  Total of column 0:  18  Highest in row 2:  9  Lowest in column 2:  2  Lowest in Array:  -8 | yes |
| 3 |  |  |  |  |

**Screenshots:**

****

차트이(가) 표시된 사진

자동 생성된 설명



차트이(가) 표시된 사진

자동 생성된 설명

**Lessons Learned:**

Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.

What have you learned?

I learned how to work with ragged arrays and gained experience in implementing utility methods for specific data structures

What did you struggle with?

I initially struggled with understanding the requirements for handling ragged arrays

What would you do differently on your next project?

I will spend more time upfront understanding the requirements and clarifying any ambiquities

What parts of this assignment were you successful with, and what parts (if any) were you not successful with?

I was successful with writing clean and organized code

Provide any additional resources/links/videos you used to while working on this assignment/project.

<Provide answers to the questions listed above>

**Check List:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment#\_Moss.zip | **Yes** | **Daniel Kim(DK)** |
|  | * FirstInitialLastName\_Assignment#.docx/.pdf | **Yes** | **Daniel Kim(DK)** |
|  | * Source java files | **Yes** | **Daniel Kim(DK)** |
|  | **Program compiles** | **Yes** | **Daniel Kim(DK)** |
|  | **Program runs with desired outputs related to a Test Plan** | **Yes** | **Daniel Kim(DK)** |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Yes** | **Daniel Kim(DK)** |
|  | * Screenshots for each Test case listed in the Test Plan | **Yes** | **Daniel Kim(DK)** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Yes** | **Daniel Kim(DK)** |
|  | * UML Diagram (if required) | **Yes** | **Daniel Kim(DK)** |
|  | * Algorithms/Pseudocode (if required) | **Yes** | **Daniel Kim(DK)** |
|  | * Flowchart (if required) | **Yes or No or N/A** |  |
|  | * Lessons Learned | **Yes** | **Daniel Kim(DK)** |
|  | * Checklist is completed and included in the Documentation | **Yes or No** | **Daniel Kim(DK)** |