CMSC 203 (CRN: 35630)

Project-5 final submission

Author: Jemil Patel

Instructor: Gary Thai

***UML class diagrams: (static methods are underlined)***

|  |
| --- |
| **TwoDimRaggedArrayUtility** |
|  |
| + TwoDimRaggedArrayUtility()  + readFile(file: File): double[][]  + writeToFile(data: double[][], outputFile: File): void  + getTotal(data: double[][]): double  + getAverage(data: double[][]): double  + getRowTotal(data: double[][], row: int): double  + getColumnTotal(data: double[][], col: int): double  + getHighestInRow(data: double[][], row: int): double  + getHighestInRowIndex(data: double[][], row: int): int  + getLowestInRow(data: double[][], row: int): double  + getLowestInRowIndex(data: double[][], row: int): int  + getHighestInColumn(data: double[][], col: int): double  + getHighestInColumnIndex(data: double[][], col: int): int  + getLowestInColumn(data: double[][], col: int): double  + getLowestInColumnIndex(data: double[][], col: int): int  + getHighestInArray(data: double[][]): double  + getLowestInArray(data: double[][]): double |

|  |
| --- |
| **HolidayBonus** |
|  |
| + HolidayBonus()  + calculateHolidayBonus(data: double[][], high: double, low: double, other: double): double[]  + calculateTotalHolidayBonus(data: double[][], high: double, low: double, other: double): double |

***Pseudocode:***

TwoDimRaggedArrayUtility class

* public TwoDimRaggedArrayUtility(): constructor
  + Empty body
* public static double[][] readFile(File file)
  + If file is empty, return null
  + Create a new 10x10 string array to store the numbers read from the file and initialize each element to null
  + Run a while loop to extract each line from the file using a Scanner object, split it into a string array of numbers, and then store the contents of that string array in the initial 10x10 array we created
  + Determine the number of rows used in the 10x10 array by checking how many times the first element of each row is not null
  + Create a new two-dimensional double array with the number of rows we determined in the previous step
  + With the help of a nested for loop, calculate the number of columns for each row, initialize each row of the two-dimensional double array with the number of columns calculated, and then copy the contents of each row of the 10x10 string array to the corresponding row of the double array using the parseDouble method
  + Return the two-dimensional double array
* public static void writeToFile(double[][] data, File outputFile)
  + Create a new PrintWriter object by passing the outputFile received in the parameter
  + Using a nested for loop that runs throughout the entire data array, print each element of the data array to the outputFile using the PrintWriter object we created
    - Print a space after each element except for the last element of each row
    - Print a new line after each row
* public static double getTotal(double[][] data)
  + Declare a double variable named total and initialize it to 0
  + Run a nested for loop throughout the entire data array and add each element of the data array to total
  + Return total
* public static double getAverage(double[][] data)
  + Declare a double variable named total and initialize it with the value returned from passing data array to getTotal
  + Declare a new int variable, numberOfElements and initialize it to 0
  + Run a nested for loop throughout the entire data array and increment numberOfElements for each iteration
  + Return total / numberOfElements
* public static double getRowTotal(double[][] data, int row)
  + Declare a double variable named rowTotal and initialize it to 0
  + Run a for loop throughout the length of the row of data array and add each element of the data array to the rowTotal
  + Return rowTotal
* public static double getColumnTotal(double[][] data, int col)
  + Declare a double variable named colTotal and initialize it to 0
  + Run a for loop for the number of rows in the data array and add the element at col index of each row to colTotal if col is less than or equal to the length of that row
  + Return colTotal
* public static double getHighestInRow(double[][] data, int row)
  + Return data[row][getHighestInRowIndex(data, row)]
* public static int getHighestInRowIndex(double[][] data, int row)
  + Declare an int variable, maxIndex and set it to 0
  + Run a for loop throughout the length of the row of data array and check if any other element is greater than the element at maxIndex, if so, then set maxIndex to that element’s index
  + Return maxIndex
* public static double getLowestInRow(double[][] data, int row)
  + Return data[row][getLowestInRowIndex(data, row)]
* public static int getLowestInRowIndex(double[][] data, int row)
  + Declare an int variable, minIndex and set it to 0
  + Run a for loop throughout the length of the row of data array and check if any other element is less than the element at minIndex, if so, then set minIndex to that element’s index
  + Return minIndex
* public static double getHighestInColumn(double[][] data, int col)
  + Return data[getHighestInColumnIndex(data, col)][col]
* public static int getHighestInColumnIndex(double[][] data, int col)
  + Declare an int variable maxIndex and set it to 0, and a boolean variable flag and set it to true
  + Run a for loop for the number of rows in the data array
    - The flag variable is used to determine the value of maxIndex for the first time when the program gets a valid element at col index of a row
      * If the above condition occurs, set maxIndex to i and flag to false
    - If any element of a row at col index is greater than the element of maxIndex row at col index then set maxIndex to the index of that row
  + Return maxIndex
* public static double getLowestInColumn(double[][] data, int col)
  + Return data[getLowestInColumnIndex(data, col)][col]
* public static int getLowestInColumnIndex(double[][] data, int col)
  + Declare an int variable minIndex and set it to 0, and a boolean variable flag and set it to true
  + Run a for loop for the number of rows in the data array
    - The flag variable is used to determine the value of minIndex for the first time when the program gets a valid element at col index of a row
      * If the above condition occurs, set minIndex to i and flag to false
    - If any element of a row at col index is less than the element of minIndex row at col index then set minIndex to the index of that row
  + Return minIndex
* public static double getHighestInArray(double[][] data)
  + Declare two int variables named maxRowIndex and maxColIndex and initialize them to 0
  + Run a nested for loop throughout the entire data array and check if any element of data array is greater than the element of data array at maxRowIndex and maxColIndex, if so, then store the index of that element in maxRowIndex and maxColIndex
  + Return data[maxRowIndex][maxColIndex]
* public static double getLowestInArray(double[][] data)
  + Declare two int variables named minRowIndex and minColIndex and initialize them to 0
  + Run a nested for loop throughout the entire data array and check if any element of data array is less than the element of data array at minRowIndex and minColIndex, if so, then store the index of that element in minRowIndex and minColIndex
  + Return data[minRowIndex][minColIndex]

HolidayBonus class

* public static double[] calculateHolidayBonus(double[][] data, double high, double low, double other)
  + Create a new double[] array named bonus and initialize it to the length of data array
  + Declare an int variable count and set it to 0
  + Run a nested for loop throughout the entire data array
    - If the element of the data array is highest in that column and greater than 0, then add high to bonus[count]
    - Else if the element of the data array is lowest in that column and greater than 0, then add low to bonus[count]
    - Else if the element of the data array is greater than 0, then add other to bonus[count]
    - Increment count after iteration of each row
  + Return the bonus array
* public static double calculateTotalHolidayBonus(double[][] data, double high, double low, double other)
  + Create a new double[] array named bonus and initialize it with the array returned from calling calculateHolidayBonus on passing data, high, low, other
  + Declare a new double variable named totalBonus and set it to 0
  + Run a for loop throughout the length of bonus array and add each element of the bonus array to totalBonus
  + Return totalBonus

***Test Cases:***

TwoDimRaggedArrayUtilitySTUDENT\_Test.java (JUnit student test file)

|  |  |  |
| --- | --- | --- |
| **Method name** | **Expected output** | **Did the test pass?** |
| testGetTotal | 67.52 | Yes |
| testGetAverage | 3.751 | Yes |
| testGetRowTotal | 62.11 => row 0  33.03 => row 1  -3.82 => row 2  -4.85 => row 3  -18.95 => row 4 | Yes |
| testGetColumnTotal | -28.86 => column 0  35.58 => column 1  -52.31 => column 2  86.8 => column 3  26.31 => column 4 | Yes |
| testGetHighestInArray | 42.10 | Yes |
| testWriteToFile | 4.97 => array[0][0]  25.71 => array[0][1]  -4.22 => array[0][2]  35.65 => array[0][3]  27.91 => array[1][0]  -30.92 => array[1][1]  0.68 => array[1][2]  9.05 => array[1][3]  26.31 => array[1][4]  -32.18 => array[2][0]  27.87 => array[2][1]  0.49 => array[2][2]  5.37 => array[3][0]  -3.06 => array[3][1]  -49.26 => array[3][2]  42.10 => array[3][3]  -34.93 => array[4][0]  15.98 => array[4][1] | Yes |

HolidayBonusTestSTUDENT.java (JUnit student test file)

|  |  |  |
| --- | --- | --- |
| **Method name** | **Expected output** | **Did the test pass?** |
| testCalculateHolidayBonusA | dataSetSTUDENT1  2000.0 => result[0]  12000.0 => result[1]  7000.0 => result[2]  6000.0 => result[3]  dataSetSTUDENT2  2000.0 => result[0]  17000.0 => result[1]  3000.0 => result[2]  7000.0 => result[3] | Yes |
| testCalculateHolidayBonusB | dataSetSTUDENT1  500.0 => result[0]  2500.0 => result[1]  1500.0 => result[2]  1250.0 => result[3]  dataSetSTUDENT2  500.0 => result[0]  3500.0 => result[1]  750.0 => result[2]  1500.0 => result[3] | Yes |
| testCalculateTotalHolidayBonusA | dataSetSTUDENT1  27000.0  dataSetSTUDENT2  29000.0 | Yes |
| testCalculateTotalHolidayBonusB | dataSetSTUDENT1  5750.0  dataSetSTUDENT2  6250.0 | Yes |

***Screenshots:***

TwoDimRaggedArrayUtility\_GFA\_Test.java

Graphical user interface, text, application

Description automatically generated

TwoDimRaggedArrayUtilityTest.java

Graphical user interface, text, application

Description automatically generated

TestOut.txt generated from running TwoDimRaggedArrayUtilityTest.java

Graphical user interface, text, application

Description automatically generated

TwoDimRaggedArrayUtilitySTUDENT\_Test.java

Graphical user interface, text, application

Description automatically generated

TestOutStudent.txt generated from running TwoDimRaggedArrayUtilitySTUDENT\_Test.java

Text

Description automatically generated

HolidayBonus\_GFA\_Test.java

Graphical user interface, text, application

Description automatically generated

HolidayBonusTest.java

Graphical user interface, text, application

Description automatically generated

HolidayBonusTestSTUDENT.java

Graphical user interface, text, application

Description automatically generated

File containing sales data (dataSet1.txt)

Graphical user interface, text, application

Description automatically generated

Result after selecting Load Sales Data (Note: ₹ is showing instead of $ since I live in India)

A picture containing chart

Description automatically generated

File containing sales data (dataSet2.txt)

Graphical user interface, text, application

Description automatically generated

Result after selecting Load Sales Data

A picture containing graphical user interface

Description automatically generated

File containing sales data (dataSet3.txt)

Graphical user interface, text, application

Description automatically generated

Result after selecting Load Sales Data

A picture containing graphical user interface

Description automatically generated

File containing sales data (dataSet4.txt)

Graphical user interface, text, application

Description automatically generated

Result after selecting Load Sales Data

A picture containing graphical user interface

Description automatically generated

File containing sales data (district3.txt)

Text

Description automatically generated

Result after selecting Load Sales Data

A picture containing graphical user interface

Description automatically generated

File containing sales data (district4.txt)

Text

Description automatically generated

Result after selecting Load Sales Data

A picture containing chart

Description automatically generated

File containing sales data (district5.txt)

Text

Description automatically generated

Result after selecting Load Sales Data

A picture containing timeline

Description automatically generated

File containing sales data (studentFile1.txt): created by student

Text

Description automatically generated

Result after selecting Load Sales Data

A picture containing graphical user interface

Description automatically generated

File containing sales data (studentFile2.txt): created by student

Graphical user interface, text, application

Description automatically generated

Result after selecting Load Sales Data

A picture containing chart

Description automatically generated

***Learning Experience***

This project was a thorough test of my knowledge of two-dimensional arrays. Before working on this project, I never used ragged arrays. It was quite interesting to note that ragged arrays actually resemble more real-life situations instead of a perfectly filled array. I also learned how to read numbers from a file separated by spaces and then store them in a two-dimensional array. Overall, the project helped me learn how to deal with ragged arrays and extract useful information from it like sum of a row or sum of a column.

Firstly, reading numbers from a file and storing them into a double array was a bit tough but the split method made my work easier. Secondly, I needed to keep this thing in my mind that each row has a different number of elements and so I needed to be careful and not cause an ArrayIndexOutOfBounds exception. Thirdly, while creating the getHighestInColumn method I had to add a condition inside the for loop in order to determine the maxIndex for the first iteration since I cannot assign it zero initially as I did in getHighestInRow or else if the first row did not have any number at that column index then again it would have caused an ArrayOutOfBoundsException. Lastly, it took me a while to figure out the bonus calculation procedure and also I had to check that I do not assign any bonus for a negative sales amount.

I will follow the same steps to work on my next project that I used in my previous projects, that is, reading and understanding all the expectations and documentation, forming a rough base program, and then slowly and gradually adding all the conditions and checks for the final testing. In my opinion I successfully met all the requirements for this project.

***GitHub screenshot:***

A screenshot of a computer

Description automatically generated

Assignment 5 Check List (include Yes/No or N/A for each item)

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N or N/A** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment5\_Moss.zip | **Y** |  |
|  | * FirstInitialLastName\_Assignment5\_Complete.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs with desired outputs related to a Test Plan** | **Y** |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Y** |  |
|  | * Screenshots for each Junit Test | **Y** |  |
|  | * Screenshots for each Test case listed in the Test Plan | **Y** |  |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Y** |  |
|  | * UML Diagram | **Y** |  |
|  | * Algorithms/Pseudocode | **Y** |  |
|  | * Flowchart (if required) | **N/A** |  |
|  | * Lessons Learned | **Y** |  |
|  | * Checklist is completed and included in the Documentation | **Y** |  |