**Montgomery College**

**CMSC 203**

**Assignment 5 Design**

1. Write the pseudo-code for the methods of *TwoDimRaggedArrayUtility* and *HolidayBonus* class based on the Assignment 5 Description given to you. Refer to the [**Pseudocode Guideline**](#PSGdline)on how to write Pseudocode.

**class TwoDimRaggedArrayUtility {**

**// Method to read data from a file and create a ragged array**

**function readFile(File file):**

**Initialize an empty ragged array data**

**Open the file for reading**

**Loop through each line in the file:**

**Read the line**

**Split the line by spaces to get values**

**Initialize a new row in data with length equal to the number of values**

**Loop through each value:**

**Parse the value to double and store it in the row**

**Add the row to data**

**Close the file**

**Return data**

**// Method to write data from a ragged array to a file**

**function writeToFile(double[][] data, File outputFile):**

**Open the file for writing**

**Loop through each row in data:**

**Loop through each value in the row:**

**Write the value to the file**

**Write a space**

**Write a new line to the file**

**Close the file**

**// Method to calculate the total of all elements in the ragged array**

**function getTotal(double[][] data):**

**Initialize total to 0**

**Loop through each row in data:**

**Loop through each value in the row:**

**Add the value to total**

**Return total**

**// Method to calculate the average of all elements in the ragged array**

**function getAverage(double[][] data):**

**Initialize total to 0**

**Initialize count to 0**

**Loop through each row in data:**

**Loop through each value in the row:**

**Add the value to total**

**Increment count**

**Return total divided by count**

**// Method to calculate the total of a specific row in the ragged array**

**function getRowTotal(double[][] data, int row):**

**Initialize total to 0**

**Loop through each value in the specified row:**

**Add the value to total**

**Return total**

**// Method to calculate the total of a specific column in the ragged array**

**function getColumnTotal(double[][] data, int col):**

**Initialize total to 0**

**Loop through each row in data:**

**If the column index is valid for the row:**

**Add the value at the specified column index to total**

**Return total**

**// Method to find the highest value in a specific row in the ragged array**

**function getHighestInRow(double[][] data, int row):**

**Initialize highest to the first value in the row**

**Loop through each value in the specified row:**

**If the value is greater than highest:**

**Update highest to the value**

**Return highest**

**// Method to find the index of the highest value in a specific row in the ragged array**

**function getHighestInRowIndex(double[][] data, int row):**

**Initialize highest to the first value in the row**

**Initialize highestIndex to 0**

**Loop through each value in the specified row starting from the second value:**

**If the value is greater than highest:**

**Update highest to the value**

**Update highestIndex to the current index**

**Return highestIndex**

**// Method to find the lowest value in a specific row in the ragged array**

**function getLowestInRow(double[][] data, int row):**

**Initialize lowest to the first value in the row**

**Loop through each value in the specified row:**

**If the value is less than lowest:**

**Update lowest to the value**

**Return lowest**

**// Method to find the index of the lowest value in a specific row in the ragged array**

**function getLowestInRowIndex(double[][] data, int row):**

**Initialize lowest to the first value in the row**

**Initialize lowestIndex to 0**

**Loop through each value in the specified row starting from the second value:**

**If the value is less than lowest:**

**Update lowest to the value**

**Update lowestIndex to the current index**

**Return lowestIndex**

**// Method to find the highest value in a specific column in the ragged array**

**function getHighestInColumn(double[][] data, int col):**

**Initialize highest to negative infinity**

**Loop through each row in data:**

**If the column index is valid for the row:**

**If the value at the specified column index is greater than highest:**

**Update highest to the value**

**Return highest**

**// Method to find the index of the highest value in a specific column in the ragged array**

**function getHighestInColumnIndex(double[][] data, int col):**

**Initialize highest to negative infinity**

**Initialize highestIndex to -1**

**Loop through each row in data:**

**If the column index is valid for the row:**

**If the value at the specified column index is greater than highest:**

**Update highest to the value**

**Update highestIndex to the index of the current row**

**Return highestIndex**

**// Method to find the lowest value in a specific column in the ragged array**

**function getLowestInColumn(double[][] data, int col):**

**Initialize lowest to positive infinity**

**Loop through each row in data:**

**If the column index is valid for the row:**

**If the value at the specified column index is less than lowest:**

**Update lowest to the value**

**Return lowest**

**// Method to find the index of the lowest value in a specific column in the ragged array**

**function getLowestInColumnIndex(double[][] data, int col):**

**Initialize lowest to positive infinity**

**Initialize lowestIndex to -1**

**Loop through each row in data:**

**If the column index is valid for the row:**

**If the value at the specified column index is less than lowest:**

**Update lowest to the value**

**Update lowestIndex to the index of the current row**

**Return lowestIndex**

**// Method to find the highest value in the entire ragged array**

**function getHighestInArray(double[][] data):**

**Initialize highest to negative infinity**

**Loop through each row in data:**

**Loop through each value in the row:**

**If the value is greater than highest:**

**Update highest to the value**

**Return highest**

**// Method to find the lowest value in the entire ragged array**

**function getLowestInArray(double[][] data):**

**Initialize lowest to positive infinity**

**Loop through each row in data:**

**Loop through each value in the row:**

**If the value is less than lowest:**

**Update lowest to the value**

**Return lowest**

**// Constants for bonus amounts**

**HIGHEST\_BONUS\_AMOUNT = 5000.0**

**LOWEST\_BONUS\_AMOUNT = 1000.0**

**OTHER\_BONUS\_AMOUNT = 2000.0**

**// Method to calculate holiday bonuses for each store**

**function calculateHolidayBonus(double[][] data):**

**Initialize an array bonuses to store holiday bonuses for each store**

**Loop through each category in the sales data:**

**Get the sales for the category**

**Find the store with the highest sales index**

**Find the store with the lowest sales index**

**Loop through each store:**

**If the store is the same as highest sales index:**

**Add HIGHEST\_BONUS\_AMOUNT to the store's bonus**

**Else if the store is the same as lowest sales index:**

**Add LOWEST\_BONUS\_AMOUNT to the store's bonus**

**Else:**

**Add OTHER\_BONUS\_AMOUNT to the store's bonus**

**Return bonuses**

**// Method to calculate total holiday bonuses for the district**

**function calculateTotalHolidayBonus(double[][] data):**

**Call calculateHolidayBonus method to get store bonuses**

**Sum all bonuses in the store bonuses array**

**Return the total bonus**

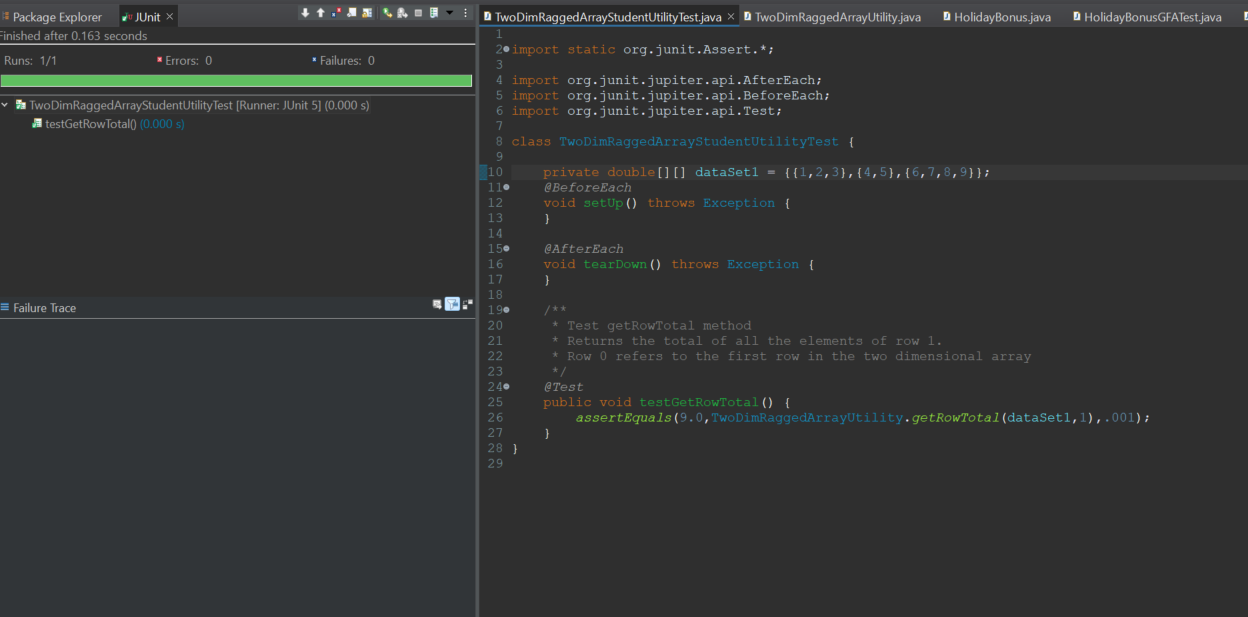
**UML:**

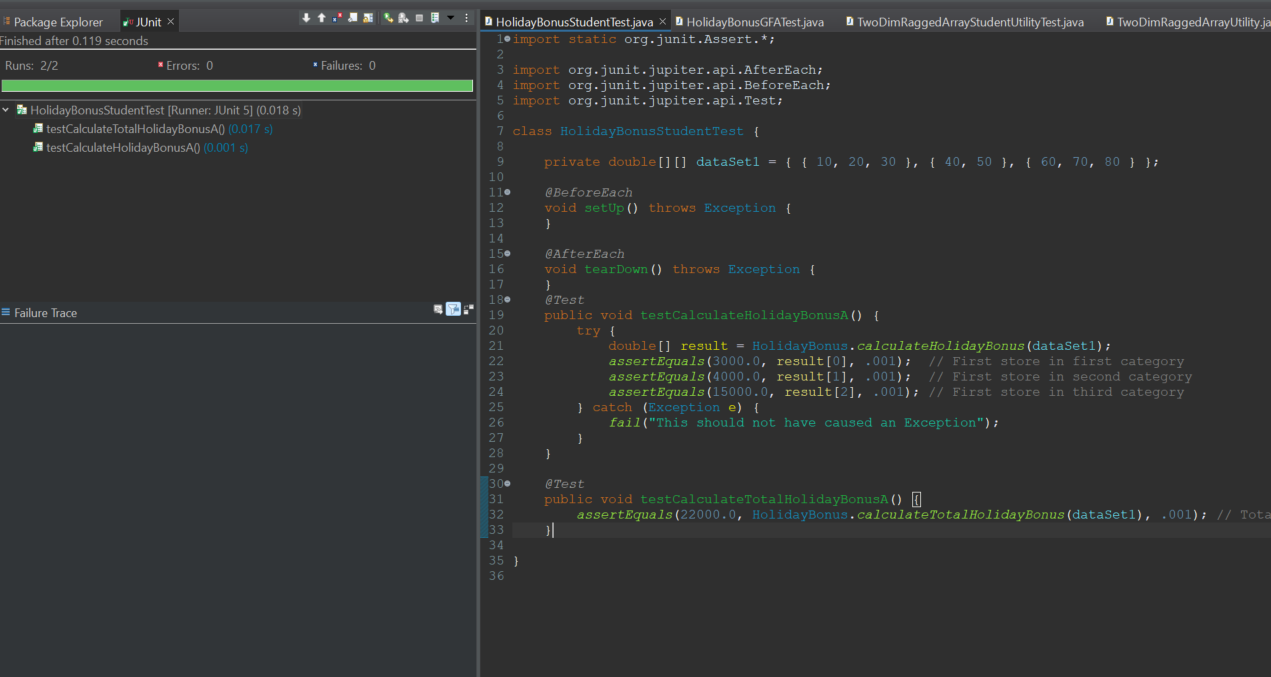
A screenshot of a computer program

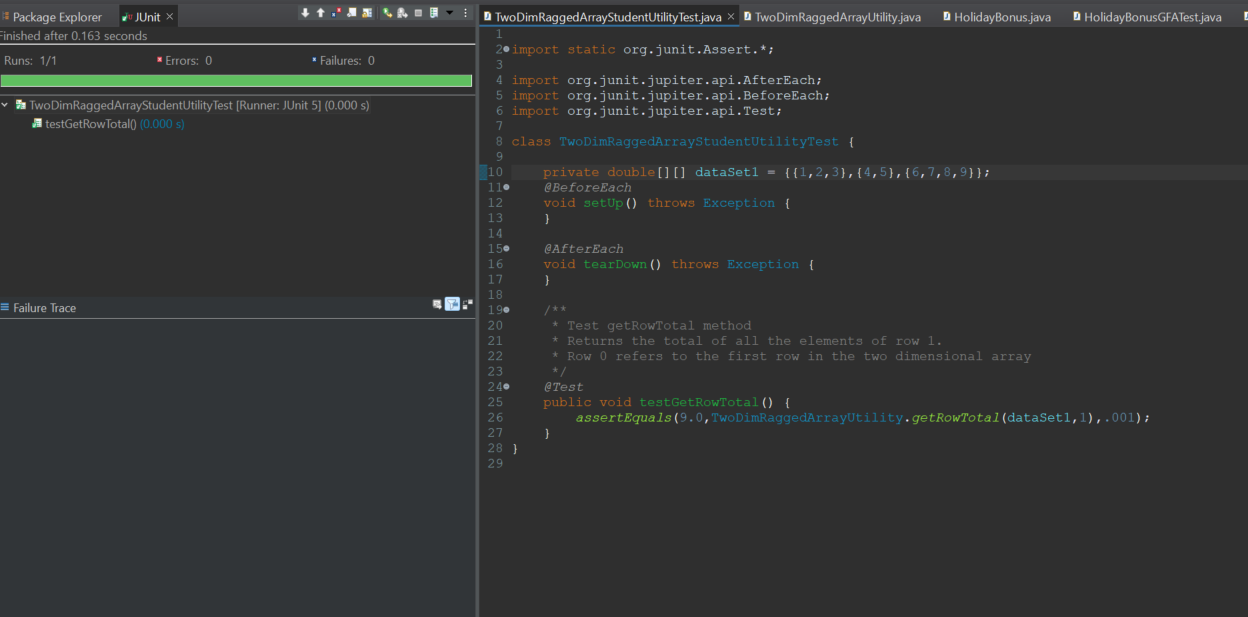
Description automatically generated

A green card with black text

Description automatically generated**Screenshots: A screenshot of a computer screen

Description automatically generated**





2)Complete the following test table. At this point you only need to complete the **Input** and **Expected** **Output** columns. Later when the implementation is complete, you will complete the **Actual Input** and **Actual Output** columns and compare them to see if the tests passed or not.

**Make sure your tests cover all the possible scenarios.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Actual Input** | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |

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

I learned a lot about handling two-dimensional arrays and file I/O operations in Java. Working on this project helped me understand how to read and write data to files, manipulate ragged arrays, and implement utility methods to perform various operations on arrays effectively.

b. What have you learned?

I learned about file handling, parsing data from files, and performing calculations on two-dimensional arrays in Java.

c. What did you struggle with?

I struggled a bit with managing indices and boundary conditions while iterating over arrays, especially when implementing methods like getHighestInColumnIndex and getLowestInColumnIndex.

d. What would you do differently on your next project?

Next time, I would pay more attention to error handling and input validation, ensuring robustness in handling various edge cases.

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

I was successful in implementing most of the methods in the TwoDimRaggedArrayUtility class and understanding their functionalities. However, I faced challenges in integrating the HolidayBonus class with it, as it required additional logic specific to calculating bonuses.

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

No additional resources were used for this project.

**Pseudocode Guideline**

Pseudocode is code written for human understanding­ n­ot a compiler. You can think of pseudocode as “English code,” code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.

**How to Write Pseudocode**

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

**Remember:** Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

|  |  |
| --- | --- |
| **Pseudocode** | **Real Code in Java** |
| Declare an integer variable called n  Declare an integer variable sum.  Declare an integer variable f1  Declare an integer variable f2  If n is less than 2  sum =n  else  set sum to 0  set f1 and f2 to 1  repeat n times  sum = f1 + f2  f2 = f1  f1 = sum  end loop  print sum | **int** n,k, f1, f2, sum;  **if** ( n < 2 )  sum =n;  **else**  {  sum=0;  f1 = f2 = 1;    **for**(k=2; k<n; k++)  {  sum = f1 + f2;  f2 = f1;  f1 = sum;  }  }  System.***out***.println("Fibonacci of number " + n + " is "+ sum); |

**Finding the Fibonacci numbers till n:**

**Remember that pseudocode is not language specific so we are not looking for “almost Java” code, but instead, we are looking for a strong understanding of the algorithm at hand.**