Sereen Sultana

CMSC 203 CRN 25800

Assignment 5 Sales Report: Pseudocode

1. Create two utility classes:
   1. TwoDimRaggedArrayUtility
   2. HolidayBonus
2. For the TwoDimRaggedArrayUtility class, create 16 methods.
3. For the HolidayBonus class, create 2 methods.
4. Import the File class (import java.io.File;) and Scanner class (import java.util.Scanner;).
5. Method 1: readFile()
   1. Create this method to return a two dimensional array of doubles.
   2. Create a new object for the File class.
   3. Create a new object for the Scanner class.
   4. Create an array to store the contents that will be read.
   5. Copy the contents read into the method using two loops, one nested into another loop.
   6. The outer loop will go start int i = 0, and up to the length of the array. Within this loop, each line is read via nextLine().
   7. The inner loop will start from int j = 0, and up to the length of the line. Convert each element of the string into a double and assign to the created array.
   8. Return the two dimensional array of doubles.
6. Method 2: writeToFile()
   1. Create this method to take in a two-dimensional array of doubles and also a file.
   2. Create a for loop with int i = 0, that will loop over the array and print out the elements one by one.
   3. Convert each double element to a string via Double.toString(x[index]);.
   4. Place each row on a separate line via .newLine();.
   5. Include the writing into the file within the loop by first creating an object for the file, such as
      1. FileWriter writer = new FileWriter();
   6. Include in the loop the following line to write to the file and also separate each double with a space:
      1. writer.write(array[index] + “ “);
   7. Close the file.
7. Method 3: getTotal()
   1. Create a method that takes in a two-dimensional array of doubles.
   2. The method will return a double.
   3. Declare a double sum initialized to 0.0.
   4. Calculate the sum of each row:
      1. Create two loops, one nested within the other to read the elements at each index.
      2. The variable sum will add the amounts at each index.
   5. Return the variable sum.
8. Method 4: getAverage()
   1. Create a method that takes in a two-dimensional array of doubles.
   2. The method will return a double.
   3. The sum will be taken from the method getTotal().
   4. The number of elements will be counted with two for loops, one nested within the other, to count the number of rows, and number of elements in each row.
   5. The total sum of all the elements will be divided by the number of elements.
   6. The method will return the average as a double.
9. Method 5: getRowTotal()
   1. Create a method that takes in a two-dimension array of doubles and a row index.
   2. Create one for loops to read the elements in one row.
   3. Set int i = 0, and i should loop through the length of the specific row specified.
   4. The sum will be all of the elements in the target row added together.
   5. The method will return the total as a double.
10. Method 6: getColumnTotal()
    1. Create a method that takes in a two-dimensional array of doubles and a column index.
    2. Create two for loops, one nested inside the other.
    3. The first for loop will go through the different rows in the array.
    4. The nested for loop will go through each column from each row, but will only add the element from the column at the index provided.
    5. The final sum will be returned as a double.
11. Method 7: getHighestInRow()
    1. Create a method that takes in a two-dimensional array of doubles and a row index.
    2. Create one for loops to read the elements in the target row.
    3. Set int i = 0, and i should loop through the length of the specific row specified.
    4. The largest element go into a variable; all the other elements will be checked against this variable to see if they are higher and be replaced if they are higher.
    5. The method will return the highest value as a double.
12. Method 8: getHighestRowIndex()
    1. Create a method that takes in a two-dimensional array of doubles and a row index.
    2. Create one for loops to read the elements in the target row.
    3. Set int i = 0, and i should loop through the length of the specific row specified.
    4. The largest element will go into a variable; all the other elements will be checked against this variable to see if they higher and be replaced if they are higher.
    5. For each element, the row and column will be placed in their own variables in an array format, and will updated if the highest element variable is updated.
    6. The method will return an array holding two indices for the row and column of the highest value in the row.
13. Method 9: getLowestInRow()
    1. Create a method that takes in a two-dimensional array of doubles and a row index.
    2. Create one for loops to read the elements in the target row.
    3. Set int i = 0, and i should loop through the length of the specific row specified.
    4. The smallest element will go into a variable; all the other elements will be checked against this variable to see if they lower and be replaced if they are lower.
    5. The method will return the lowest value as a double.
14. Method 10: getLowestInRowIndex()
    1. Create a method that takes in a two-dimensional array of doubles and a row index.
    2. Create one for loops to read the elements in the target row.
    3. Set int i = 0, and i should loop through the length of the specific row specified.
    4. The largest element will go into a variable; all the other elements will be checked against this variable to see if they lower and be replaced if they are lower.
    5. For each element, the row and column will be placed in their own variables in an array format, and will updated if the lowest element variable is updated.
    6. The method will return an array holding two indices for the row and column of the lowest value in the row.
15. Method 11: getHighestInColumn()
    1. Create a method that takes in a two-dimensional array of doubles and a column index.
    2. Create two for loops, one nested inside the other.
    3. The first for loop will go through the different rows in the array.
    4. The nested for loop will go through only the assigned column from each row.
    5. The largest element will go into a variable; all the other elements will be checked against this variable to see if they higher and be replaced if they are higher.
    6. The final value will be returned as a double.
16. Method 12: getHighestInColumnIndex()
    1. Create a method that takes in a two-dimensional array of doubles and a column index.
    2. Create two for loops, one nested inside the other.
    3. The first for loop will go through the different rows in the array.
    4. The nested for loop will go through only the assigned column from each row.
    5. The largest element will go into a variable; all the other elements will be checked against this variable to see if they higher and be replaced if they are higher.
    6. For each element, the row and column will be placed in their own variables in an array format, and will updated if the highest element variable is updated.
    7. The method will return an array holding two indices for the row and column of the largest value in the column.
17. Method 13: getLowestInColumn()
    1. Create a method that takes in a two-dimensional array of doubles and a column index.
    2. Create two for loops, one nested inside the other.
    3. The first for loop will go through the different rows in the array.
    4. The nested for loop will go through only the assigned column from each row.
    5. The lowest element will go into a variable; all the other elements will be checked against this variable to see if they lower and be replaced if they are lower.
    6. The final value will be returned as a double.
18. Method 14: getLowestInColumnIndex()
    1. Create a method that takes in a two-dimensional array of doubles and a column index.
    2. Create two for loops, one nested inside the other.
    3. The first for loop will go through the different rows in the array.
    4. The nested for loop will go through only the assigned column from each row.
    5. The lowest element will go into a variable; all the other elements will be checked against this variable to see if they lower and be replaced if they are lower.
    6. For each element, the row and column will be placed in their own variables in an array format, and will be updated if the lowest element variable is updated.
    7. The method will return an array holding two indices for the row and column of the lowest value in the column.
19. Method 15: getHighestInArray()
    1. Create a method that takes in a two-dimensional array of doubles.
    2. The method will return a double.
    3. Set int i = 0, and i should loop through the length of each row.
    4. Set int j = 0 and j will go through each column in the row i.
    5. The largest element will go into a variable; all the other elements will be checked against this variable to see if they higher and be replaced if they are higher.
    6. The method will return the highest value as a double.
20. Method 16: getLowestInArray()
    1. Create a method that takes in a two-dimensional array of doubles.
    2. The method will return a double.
    3. Set int i = 0, and i should loop through the length of each row.
    4. Set int j = 0 and j will go through each column in the row i.
    5. The smallest element will go into a variable; all the other elements will be checked against this variable to see if they lower and be replaced if they are smaller.
    6. The method will return the lowest value as a double.
21. Method 1: calculateHolidayBonus()
    1. Create a method that takes in a two-dimensional array of doubles and three doubles.
    2. Return an array of doubles.
    3. Create two for loops, one nested inside the other.
    4. The first for loop will go through the different rows in the array.
    5. The nested for loop will go through each column from each row:
       1. A conditional statement will check if the value inside the array is the value returned from the method getHighestInArray() to find the highest element in the arrays. This value will get highest bonus amount.
       2. A conditional statement will check if the value inside the array is the value returned from the method getLowestInArray() to find the lowest element in the arrays. This value will get lowest bonus amount.
       3. The else statement will have the rest of the stores getting the standard bonus.
    6. Return an array of doubles that represents the holiday bonuses for each of the stores.
22. Method 2: calculateTotalHolidayBonus()
    1. Create a method that takes in a two-dimensional array of doubles and three doubles.
    2. Return a double.
    3. Declare a double holidaySum initialized to 0.0.
    4. Calculate the sum of each row:
       1. Create two loops, one nested within the other to read the elements at each index.
       2. The variable sum will add the amounts at each index.
    5. Return the variable holidaySum.

Assignment 5 Sales Report: UML Diagram

|  |
| --- |
| HolidayBonus |
| - salesData[] [] : double |
| + calculateHolidayBonus(salesData[] [] : double, highestBonus : double, lowestBonus : double, regularBonus : double) : double [] []  + calculateTotalHolidayBonus(salesData[] []: double, highestBonus : double, lowestBonus : double, regularBonus : double) : double |

|  |
| --- |
| TwoDimRaggedArrayUtility |
| - salesData[] [] : double  - tempData[] [] : double |
| + readFile() : double [] []  + writeToFile(salesData[] [] : double, file)  + getTotal((salesData[] [] : double) : double  + getAverage(salesData[] [] : double) : double  + getRawTotal(salesData[] [] : double, int[]) : double  + getColumnTotal(salesData[] [] : double, int[]) : double  + getHighestInRow(salesData[] [] : double, int[]) : double  + getHighestRowIndex(salesData[] [] : double, int[]) : int[] []  + getLowestInRow(salesData[] [] : double, int[]) : double  + getLowestIntRowIndex(salesData[] [] : double, int[]) : int[] []  + getHighestInColumn(salesData[] [] : double, int[]) : double  + getHigestInColumnIndex(salesData[] [] : double, int[]) : int[] []  + getLowestInColumn(salesData[] [] : double, int[]) : double  + getLowestInColumnIndex(salesData[] [] : double, int[]) : int[] []  + getHighestInArray(salesData[] [] : double) : double  + getLowestInArray(salesData[] [] : double) : double |

Assignment 5 Sales Report: Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Input | Expected Output | Actual Output | Did the test pass? |
| 1 | [8372.38 3377.33 -277.32 3837.37 373.99 9373.37] [3623.63 3732.36 3636.99 464.37 32.33] [32.33 28273.33 -3.98 938.23] [737.37 99.33 213.37 558.48] [442.38 48.37 848.23 484.99 975.44] [757.48 59.87 858.26 453.44 3834.48 8.44] | Total [25057.12] [11489.68] [29239.91] [1608.55] [2799.41] [5971.97]  Holiday Bonus [19000.00] [12000.00] [8000.00] [8000.00] [9000.00] [13000.00] | Total [25057.12] [11489.68] [29239.91] [1608.55] [2799.41] [5971.97]  Holiday Bonus [19000.00] [12000.00] [8000.00] [8000.00] [9000.00] [13000.00] | Yes |
| 2 | [80.00 70.00 60.00 40.00 33.88 26.37] [30.00 70.00 60.00 32.00 44.00 37.73] [30.00 70.00 60.00 31.00 22.11 21.90] [40.00 23.00 23.00 31.00 35.00 15.27] [23.00 20.00 30.00 12.00 35.00 35.23] | Total [310.25] [273.73] [235.01] [167.27] [155.23]  Holiday Bonus: [24000.00] [18000.00] [11000.00] [10000.00] [9000.00] | Total [310.25] [273.73] [235.01] [167.27] [155.23]  Holiday Bonus: [24000.00] [18000.00] [11000.00] [10000.00] [9000.00] | No: Due to some values being the same, the program cannot declare an absolute minimum nor maximum for some of the rows and columns |
| 3 | [232.33 291.08 682.38 abc] [28.38 102.48 182.12] [1821.81 -281.28 191.28] [18.21 281.93 93.28] | Error message | No change to GUI | No: the abc string as an element in the array causes to not compile |
| 4 | [33.77 56.78 35.94] [A 68.34 70.85 365.90] [577.67 678.87] [-758.67 636.87 658.89 78.88] | Conversion of A to the respective ASCII number as a Char value | No change to GUI | No: the A string element in the array did not change to a char character and therefore the strong causes the prgoram to not compile |

Test Cases

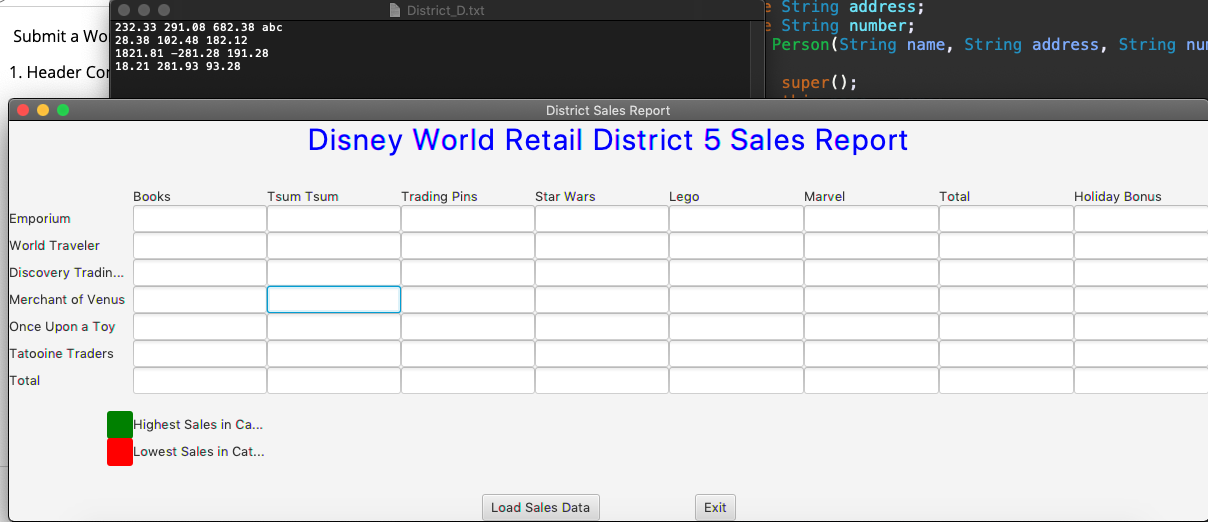
Case 1:



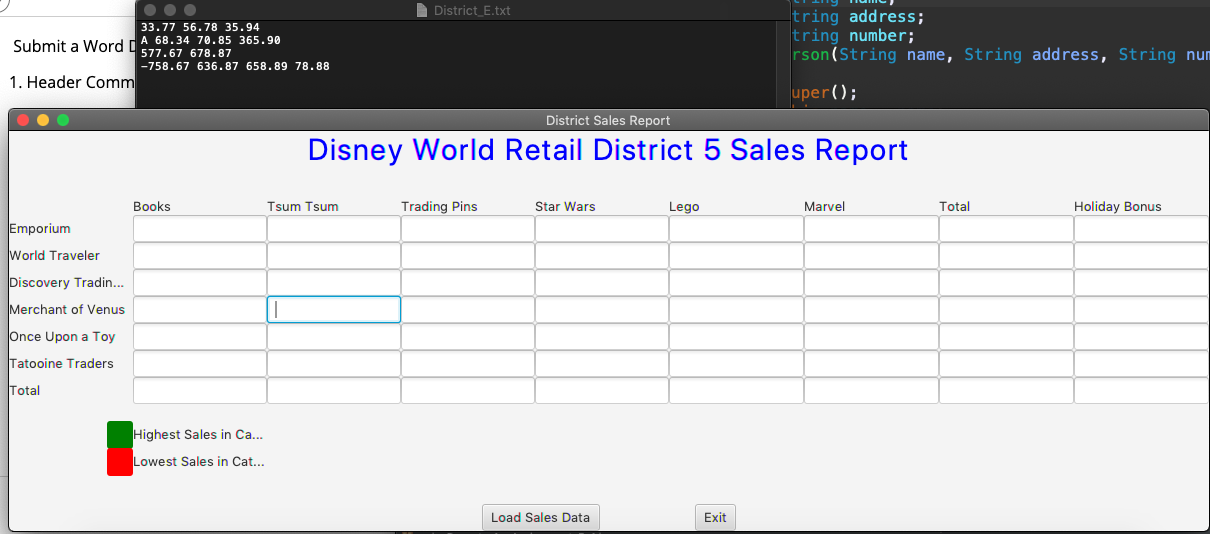
Case 2:



Case 3:

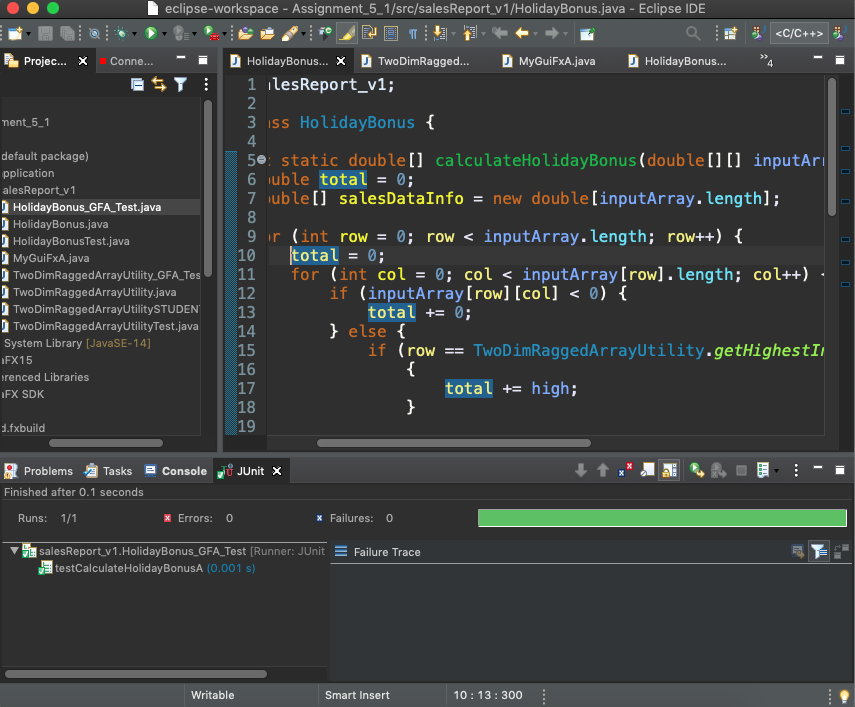


Case 4:

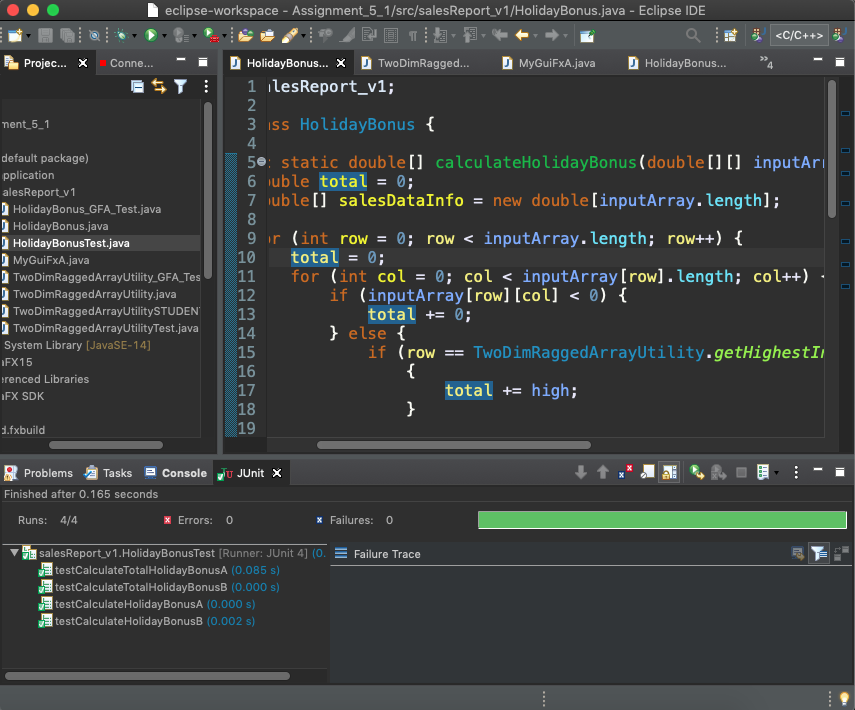


JUnit Tests

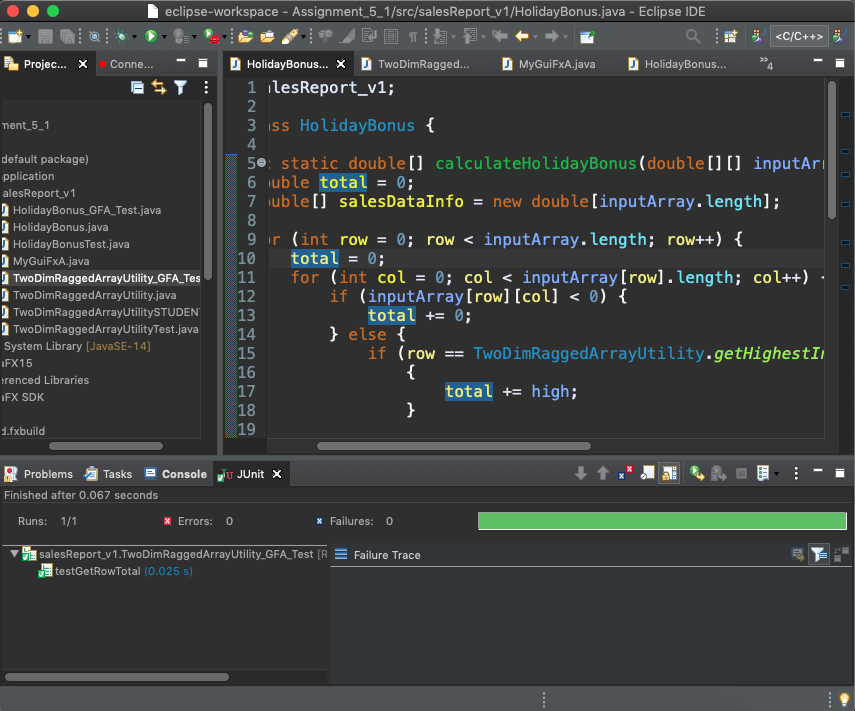
HolidayBonus\_GFA\_Test



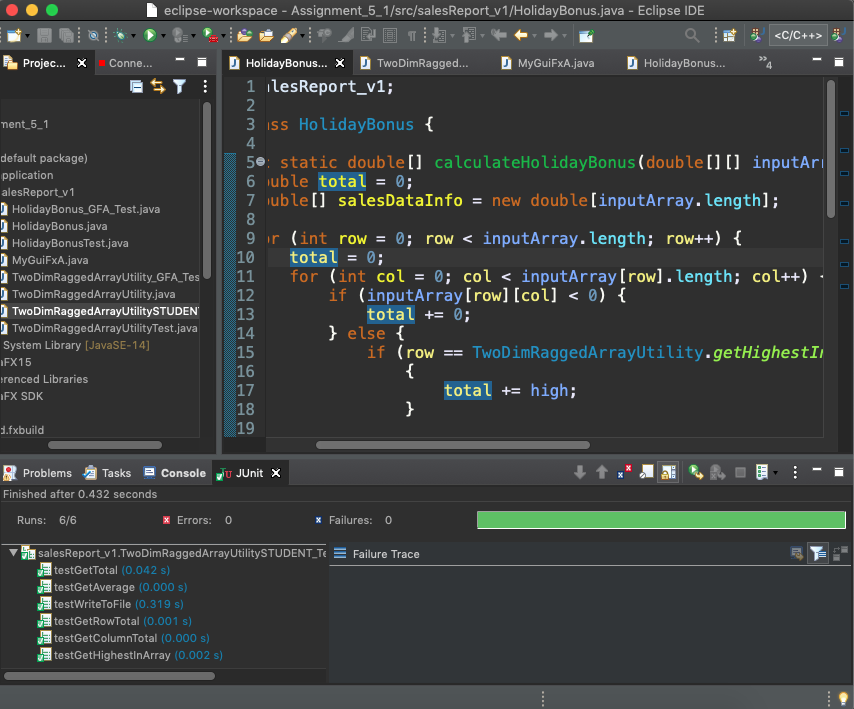
HolidayBonusTest



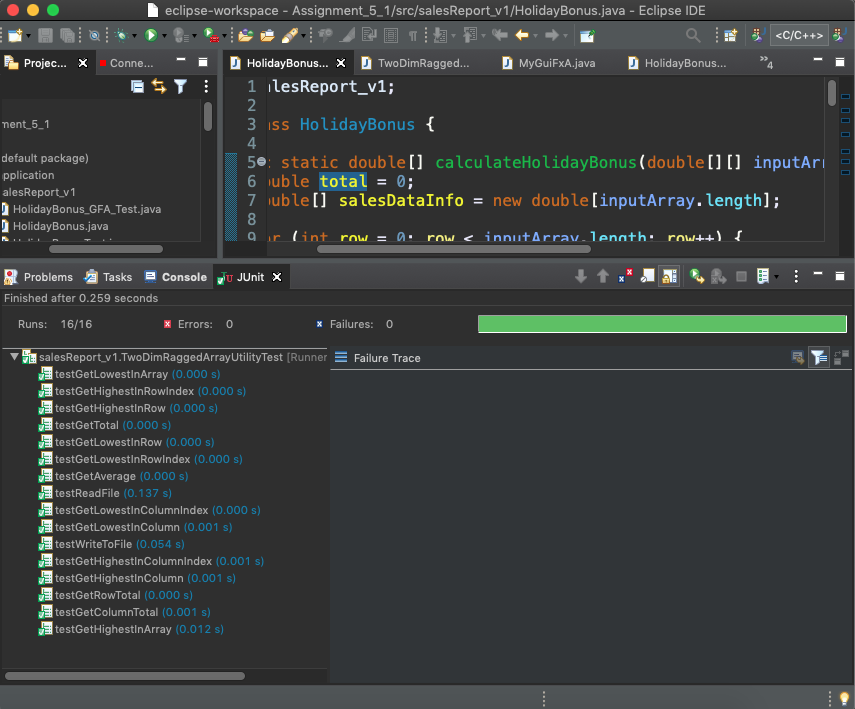
TwoDimRaggedArrayUtility\_GFA\_Test



TwoDimRaggedArrayUtilitySTUDENT\_Test



TwoDimRaggedArrayUtilityTest



Assignment 5: GitHub

Lessons Learned

Completing the arrays project taught me a lot of new things and I am happy I was able to implement them. In the beginning, it was hard to imagine a 2-dimensional array in my mind so I drew a small diagram on paper and it helped me focus and organizing my thoughts. With the diagram, I was able to understand how the different methods of my code will work and wrote my pseudocode with the paper diagram to guide me. This was something I struggled with but overcame this conflict by finding a method that works for me.

Something I was successful with was learning how to use loops to read arrays properly; this project was the second one I did using this concept. Before beginning this project, I completed my CMSC 140 project using arrays to practice how they would be used and it gave me more understanding to write the complex methods for this project. It was hard to work with them originally, but I was determined to complete both projects using arrays. After working through both projects, I feel that I have learned a new way to organize data and using loops allows me to access all the information within each index.

Working through this project, I found it difficult to separate the idea of the index of an element versus the value inside of the index and how I can illustrate and retrive those two pieces of information. I struggled with this but after re-reading the book, I was able to write those two methods. One thing I would do differently next time is learn more about the different types of test cases that are appropriate for arrays. I attempted to test my program with the same values for the highest and lowest elements, and also using strings instead of doubles as the elements inside the array, but perhaps there are different types of test cases I can also use for 2-dimensional arrays. This was a good project to learn a new concept and it also showed the power of using loops in coding.

Assignment 5: Grading Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
| **1.** | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment 5\_Moss.zip | **<Yes or No>** | Yes |
|  | * FirstInitialLastName\_Assignment5.zip | **<Yes or No>** | Yes |
| **2.** | **Program compiles** | **<Yes or No>** | Yes |
| **3.** | **Program runs with desired outputs related to a**  **Test Plan** | **<Yes or No>** | Yes |
| **4.** | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **<Yes or No>** | Yes |
|  | * Screenshots for each Test case listed in the Test Plan | **<Yes or No>** | Yes |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **<Yes or No>** | Yes |
|  | * UML Diagram | **<Yes or No>** | Yes |
|  | * Algorithms/Pseudocode | **<Yes or No>** | Yes |
|  | * Flowchart (if required) | **<Yes or No or N/A>** | N/A |
|  | * Lessons Learned | **<Yes or No>** | Yes |
|  | * Checklist is completed and included in the Documentation | **<Yes or No>** | Yes |