**Montgomery College**

**CMSC 203**

**Assignment 5 Design**

Class: CMSC203 CRN 46519

 Program: Assignment 5 Design

Instructor: Professor Eivazi

 Summary of Description: (This application will manipulate a two-dimensional ragged array of doubles.

\* This utility class will be used to create a Sales Report for Retail District #5.

\* It will accommodate positive and negative numbers. The application will read the data from the files

\* and will determine the highest, lowest and other numbers from the files. It will then calculate the average, total, highest and lowest element in array.)

 Due Date: 07/28/2022

 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: Fakhreya Mohammadi

**Part 1: Pseudo Code:**

**Pseudo Code for TwoDimRaggedArrayUtility class:**

Create a no-arg constructor called TwoDimRaggedArrayUtility

**getAverage(double[][] data) method**

Create a public getAverage(double[][] data) method of type static double

Declare variable members assign 0 to it

Declare a double variable called total assign it to 0

Loop through the data as long as k is less tha the value of the data length

Add the value of length of data at index k to the members and assign to the members

Loop through length of data at index k aslong as i is less than the value of it

Add the value of sum of all members of array to total, assign it to total

End for loop

End for loop

Return total divided by members

End method

**getColumnTotal(double[][] data, int col) method**

declare a variable called colSum of type double assign it to 0

loop through the data length as long as m is less than value of data length

if col is more than or equal to the data length at index m

continue

end for loop

add the total of column at specified m index to the colSum variable and assign it to the colSum

end if

return colSum

end method

**getHighestInArray(double[][] data) method**

declare a variable called highest of type double assign it -999999

loop through the data length as long as n is less than value of data length

loop through length of array at index n to get rows as long as h is less than value of data length

if value at specified index n and h for column and row is more than highest

assign highest to that value of that cell in array data[n][h]

end if

end for loop

end for loop

return highest

end method

**getHighestInColumn(double[][] data, int col) method**

declare a variable called highest of type double assign it -999999

loop through the data length as long as v is less than value of data length

if col is more than or equal to the value of length of data at index v

continue

end if

if value of col at index v is more than highest

set highest to the value of col at index v in array data

end if

end for loop

return highest

end method

**getHighestInColumnIndex(double[][] data, int col) method**

declare a variable called highest of type double assign it -999999

declare a variable called index of type int set it equal to -1

loop through the data length as long as b is less than value of data length

if col is more than or equal to the value of length of data at index b

continue

end if

if value of col at index b is more than highest

set highest to the value of col at index b in array data

set index equal to b

end if

end for loop

return index

end method

**getHighestInRow(double[][] data, int row) method**

declare a variable called highest of type double assign it -999999

loop through the data length as long as p is less than value of data length at index of row

if the value at specified row is more than highest

assign the specified value in row to the highest

end if

end for loop

return highest

end method

**getHighestInRowIndex(double[][] data, int row) method**

declare a variable called highest of type double assign it -999999

declare a variable called index of type int set it equal to -1

loop through the data length as long as f is less than value of data length at index row

if value in that index of row is more than highest

assign the value in that index of row to the highest

assign f to the index

end if

end for loop

return index

end method

**getLowestInArray(double[][] data) method**

declare a variable called lowest of type double assign it 999999

loop through the data length as long as g is less than value of data length

loop through length of array at index g to get rows as long as h is less than value of data length

if value at specified index g and j for column and row is less than lowest

assign lowest to that value of that cell in array data[n][h]

end if

end for loop

end for loop

return lowest

end method

**getLowestInColumn(double[][] data, int col) method**

declare a variable called lowest of type double assign it 999999

loop through the data length as long as i is less than value of data length

if col is more than or equal to the value of length of data at index i

continue

end if

if value of col at index i is less than lowest

set lowest to the value of col at index i in array data

end if

end for loop

return lowest

end method

**getLowestInColumnIndex(double[][] data, int col) method**

declare a variable called lowest of type double assign it 999999

declare a variable called index of type int set it equal to -1

loop through the data length as long as m is less than value of data length

if col is more than or equal to the value of length of data at index m

continue

end if

if value of col at index m is less than lowest

set lowest to the value of col at index m in array data

set index equal to m

end if

end for loop

return index

end method

**getLowestInRow(double[][] data, int row) method**

declare a variable called lowest of type double assign it 999999

loop through the data length as long as y is less than value of data length at index of row

if the value at specified row is less than lowest

assign the specified value in row to the lowest

end if

end for loop

return lowest

end method

**getLowestInRowIndex(double[][] data, int row) method**

declare a variable called lowest of type double assign it 999999

declare a variable called index of type int set it equal to -1

loop through the data length as long as k is less than value of data length at index row

if value in that index of row is less than lowest

assign the value in that index of row to the lowest

assign k to the index

end if

end for loop

return index

end method

**getRowTotal(double[][] data, int row) method**

declare a variable called total of type double assign it to 0

loop through the data length as long as n is less than value of data length at index row

add the total of row at specified n index to the total variable and assign it to the total

end for loop

return total

end method

**getTotal(double[][] data) method**

declare a variable called total of type double assign it to 0

loop through the data length as long as p is less than value of data length

loop through the data length as long as j is less than value of data length at index p

add the total of rows and columns at specified p and j index to the total variable and assign it to the total

end for loop

end for loop

return total

end method

**readFile(File file) method**

create an instance of scanner class called sc and pass file as argument

declare a variable called indexRow set it equal to 0

create an instance of String class called temp with maximum rows of 10

while there is next line to read

separate rows by spaces

read and store each element in row

loop through length of row as long as i is less than length of row

assign the value of row in index i to the temp at row of index indexRow and column i

end for loop

increment indexRow by 1

end while loop

create a two dimensional array of type double named dataset its row index to indexRow

loop through the indexRow as long as I is less than value of indexRow

assign the value of length of temp at index i

loop through length of temp at index I as long as j is less than the vale of length of temp at index i

print array elements separated by spaces

covert the values of array to double

end for loop

print space line

end for loop

close scanner

return data

end method

**writeToFile(double[][] data, File outputFile) method**

create an instance of PrintWriter named output and pass the outputFile to it

create an instance of StringBuilder called sb

loop through the data length as long as i is less than value of data length

loop through the data length as long as j is less than value of data length at index i  
 append data to files separated by spaces using StringBuilder instance

end for loop

append new line

end for loop

print the file using toString method

close the output file

end method

end class

**Pseudo Code for HolidayBonus class:**

Create a no-arg constructor called HolidayBonus

* **Static calcualteHolidayBonus(double[][] data, double high, double low, double other) method**

Declare an instance of one dimensional array and set its size the length of array passed to it (data.length)

* + List of variables:
    - maxCol to hold the maximum in categories which is assigned to 0
  + Loop through every row of the data passed through
    - Check if the length of data in specified index is greater than maxCol
      * If it is, assign the length of data to the maxCol

Loop through the vales of maxCol as long as the value of int j is less than the value of maxCol

Find the highest value index in column by calling getHighestInColumnIndex and passing variables data and j to it, assign the returned value to the highBonusIndex variable

Find the lowest value index in column by calling getLowestInColumnIndex and passing variables data and j to it, assign the returned value to the lowBonusIndex variable

If highest value in column is greater than 0

Add high to the bonuses[highBonusIndex] and assign it to the bonuses[highBonusIndex]

If lowest value in column is greater than 0 and highBonusIndex is not equal to the lowBonusIndex

Add low to the bonuses[lowBonusIndex] and assign it to the bonuses[lowBonusIndex]

Loop through data as long as i is less than the length of data

If j is equal or more than length of data at index i or i equals highBonusIndex or i equals lowBonusIndex or two-dimensional array at column I and row j is less than or equal to 0

Countinue

Else

Add other to the bonuses at specified I index and assign it to bonuses

End else

Print a space line

End for loop

End for loop

Return bonuses

End method

* **Static calculateTotalHolidayBonus(double [][] data, double high, double low, double other) method**

Call the calculateHolidayBonus method and pass the data, high, low, and other variables to it and assign the returned value from it to the created one-dimensional array of type double called bonusArray

* + List of variables:
    - total to hold the sum of all bonuses and is initialized to 0

loop through each array of the bonusArray

* + - add bonus to the total and assign it to the total for each iteration

end for loop

* + Return the total of all the holiday bonuses.
* End method

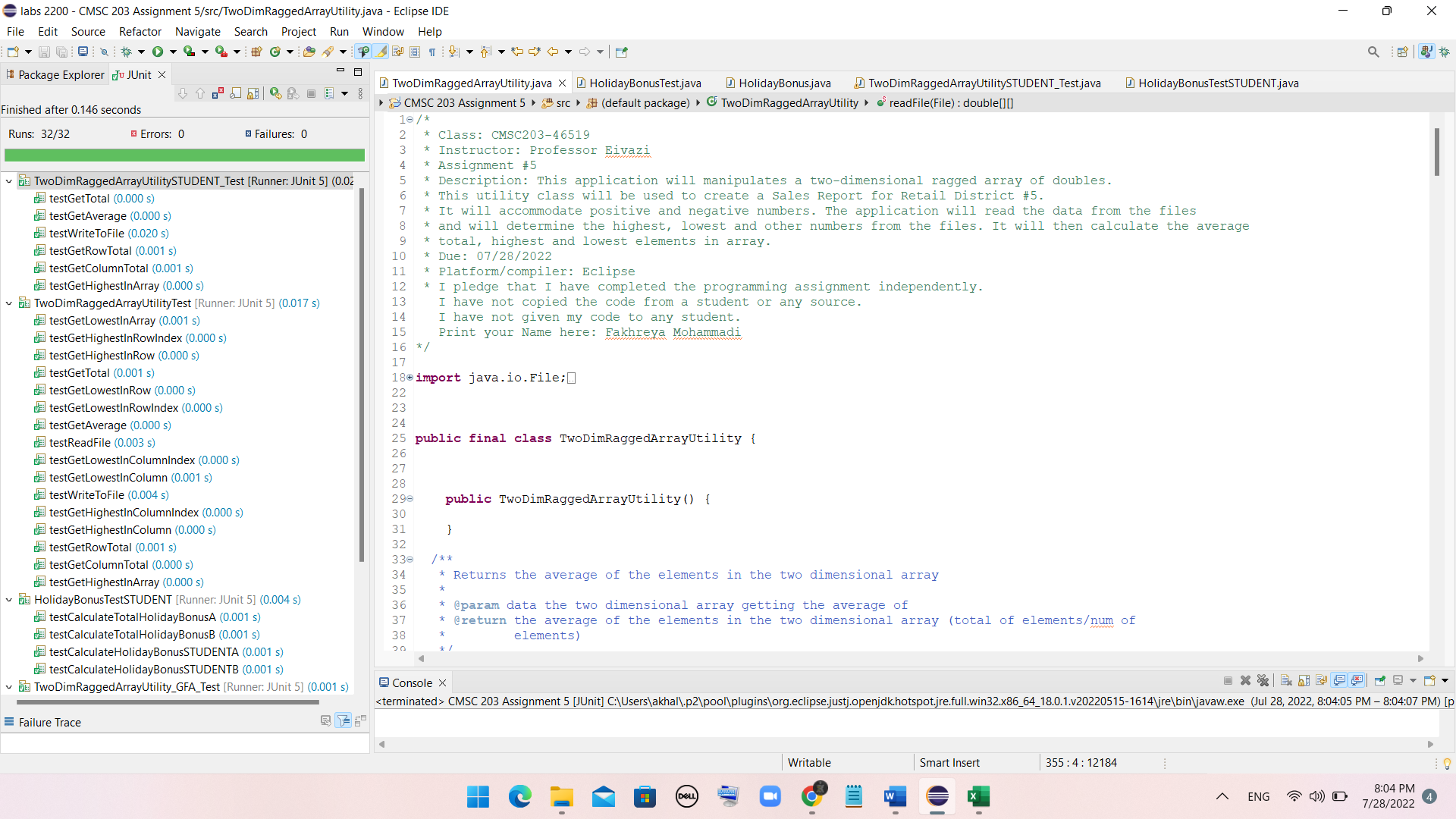
**Part 2: UML**

**UML diagrams**

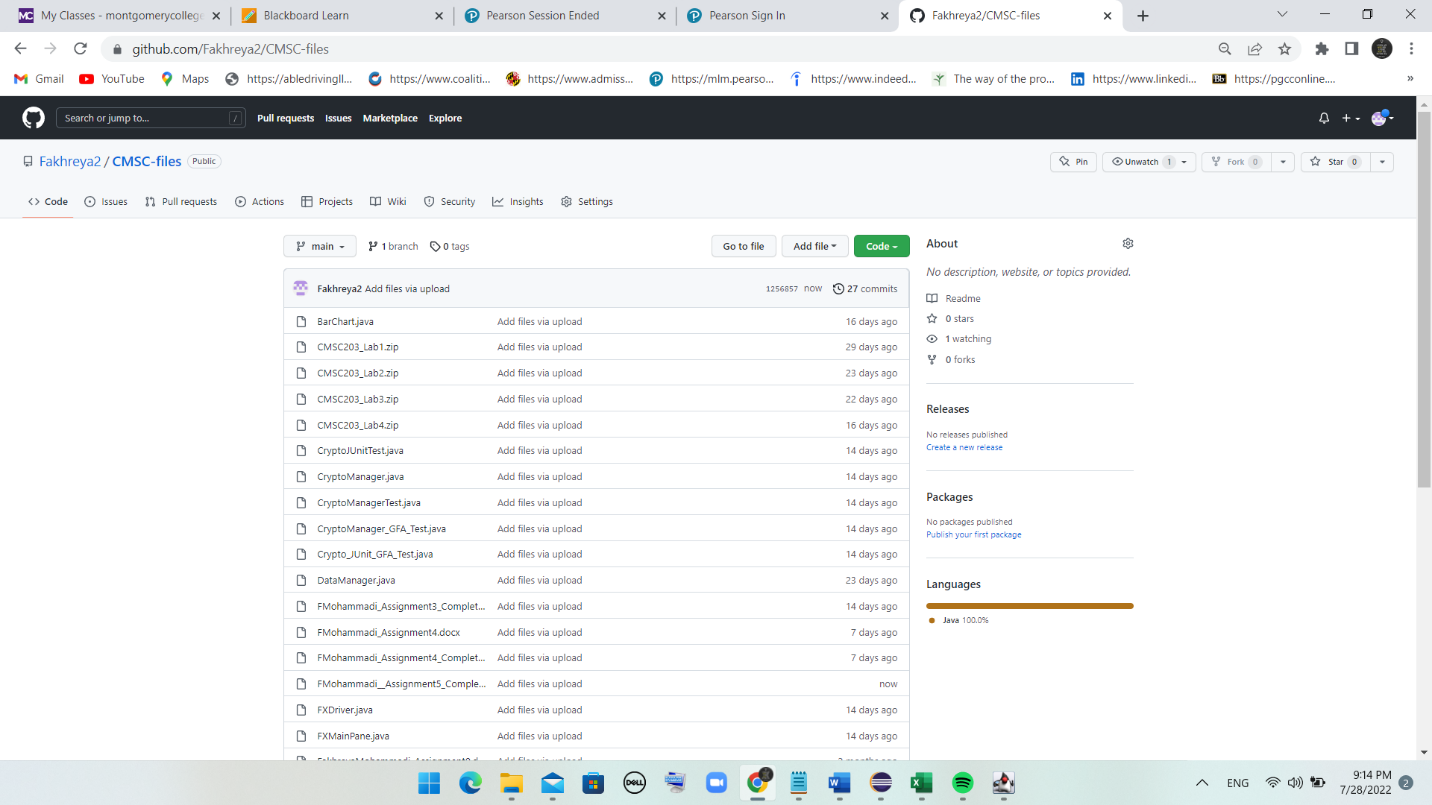
|  |
| --- |
| TwoDimRaggedArrayUtility |
|  |
| + TwoDimRaggedArrayUtility()  + getAverage(data: double[][]) : static double  + getColumnTotal(data: double[][], col:int) : static double  + getHighestInArray(data: double[][]) : static double  + getHighestInColumn(data: double[][], col:int) : static double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

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

**Part 3: Junit Test Screenshot**

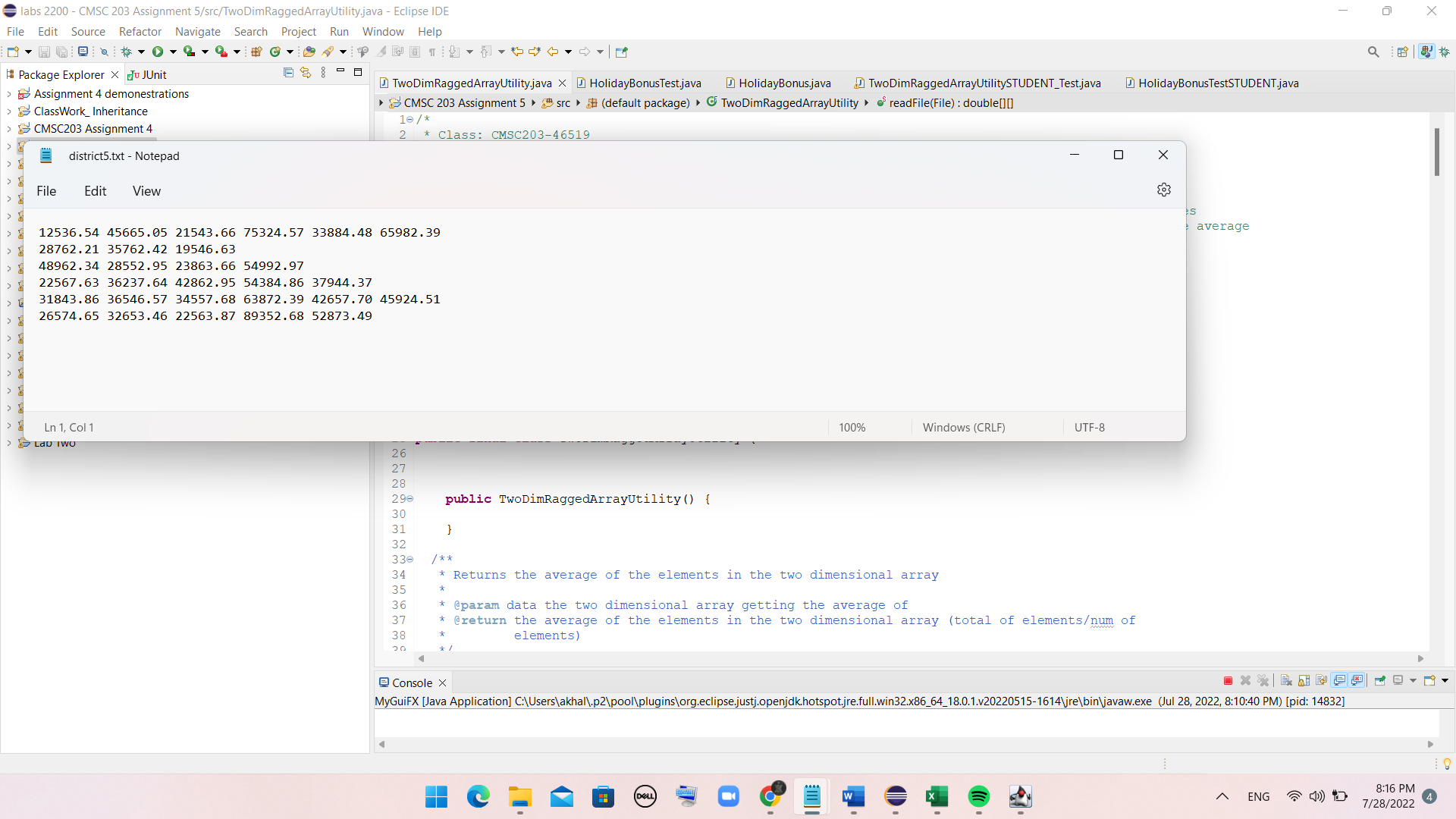


**Part 4: GetHub Submissions**

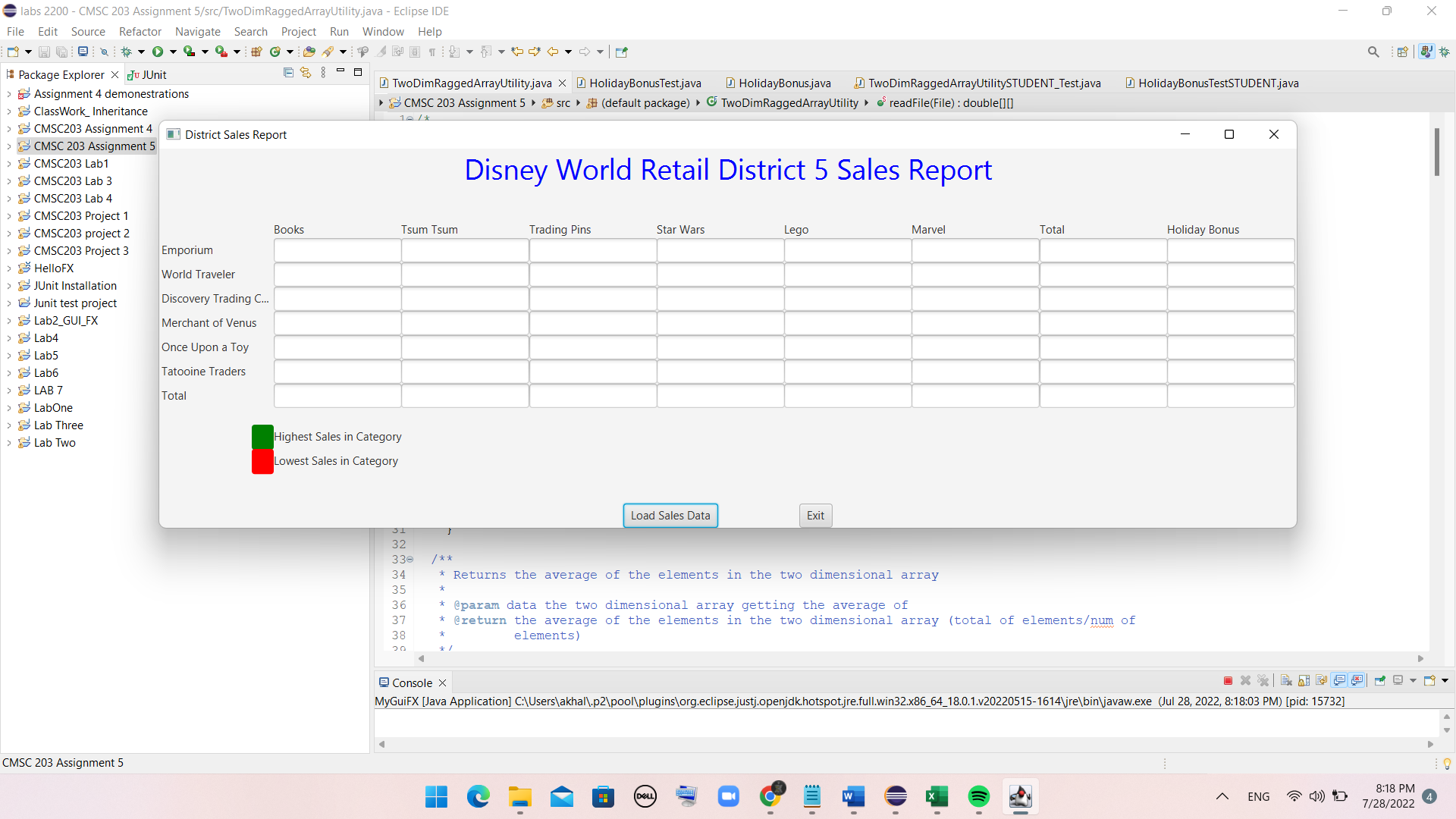


**Part 5: Screen snapshots of the GUI with several properties and tests**

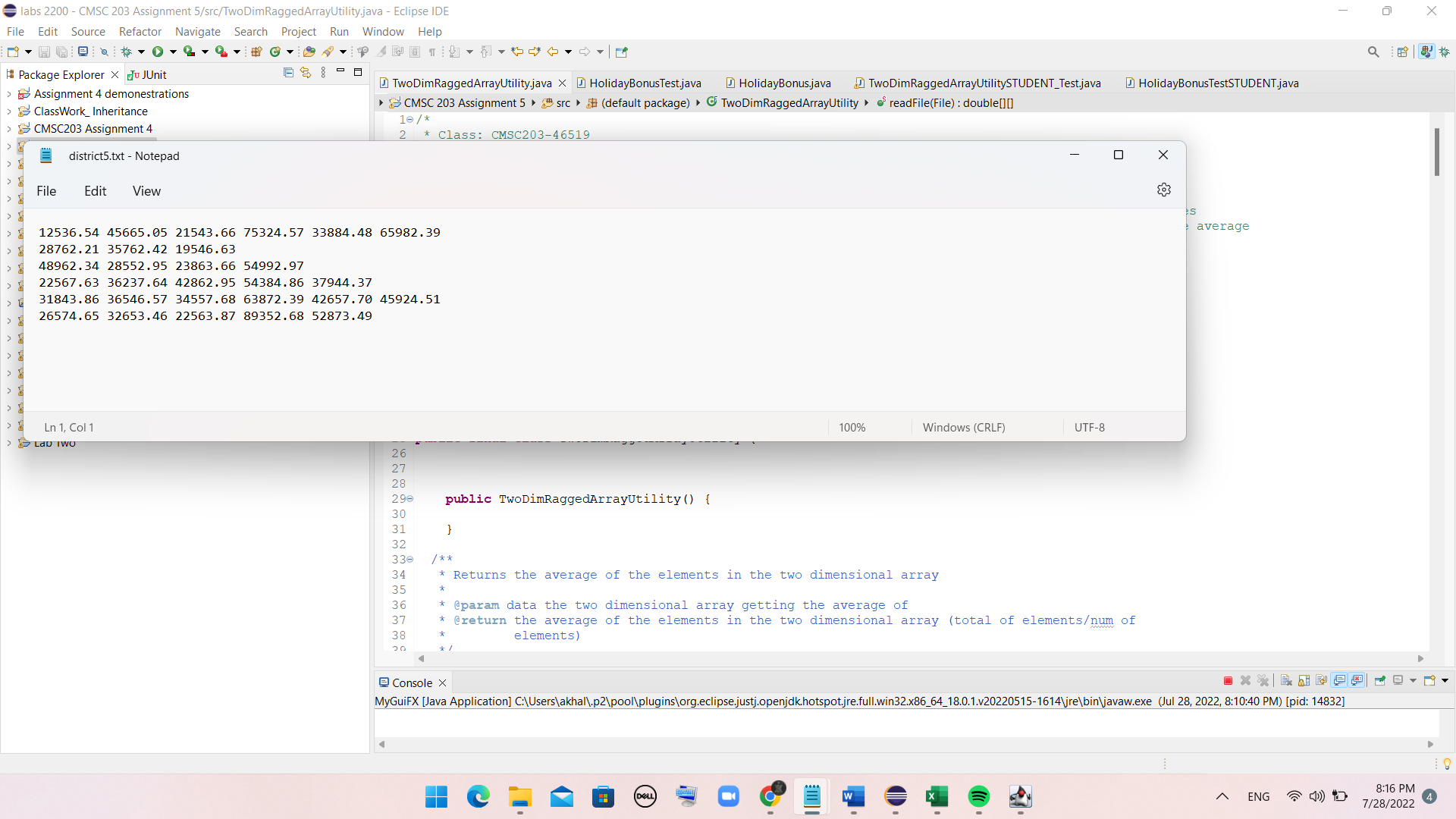
File Format:



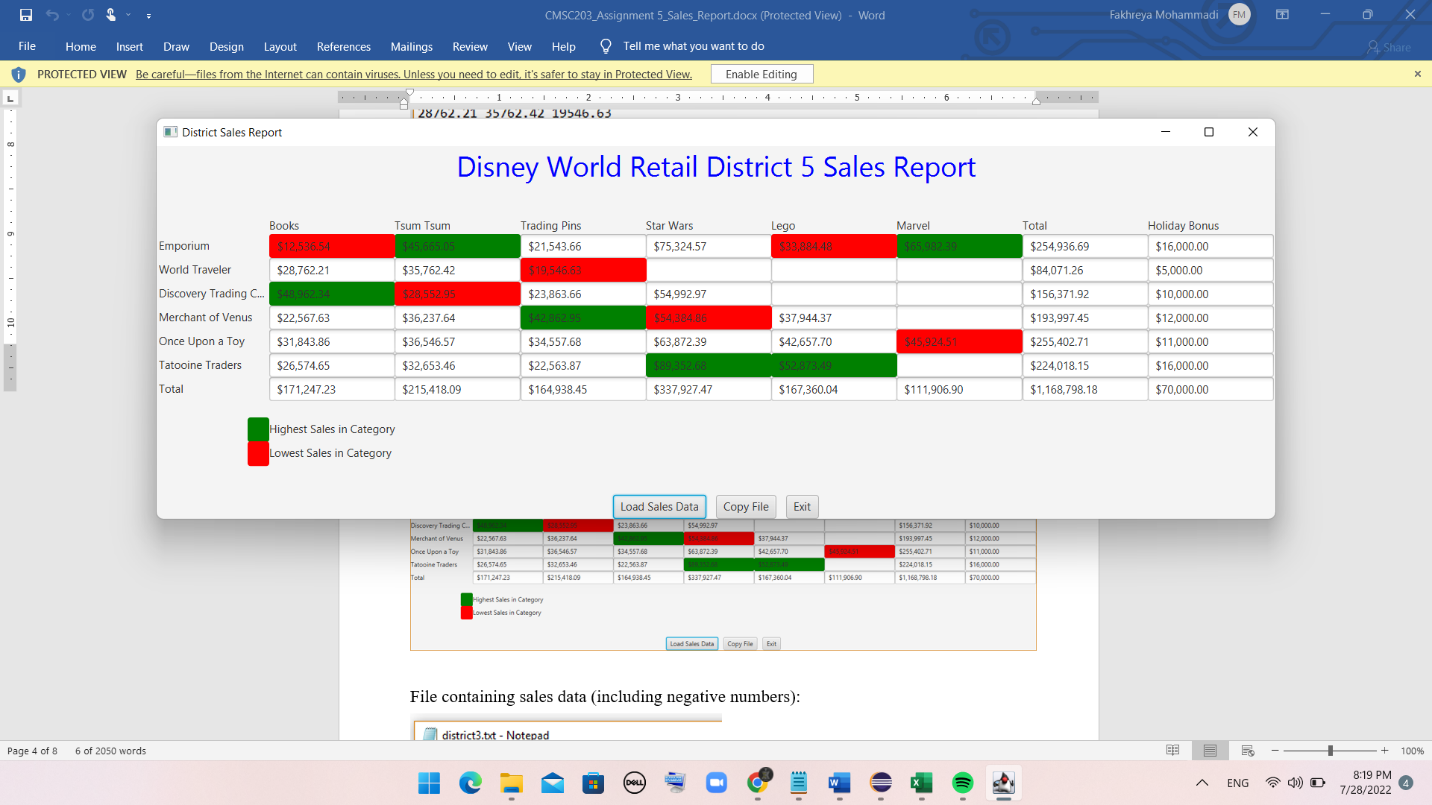
When application starts:



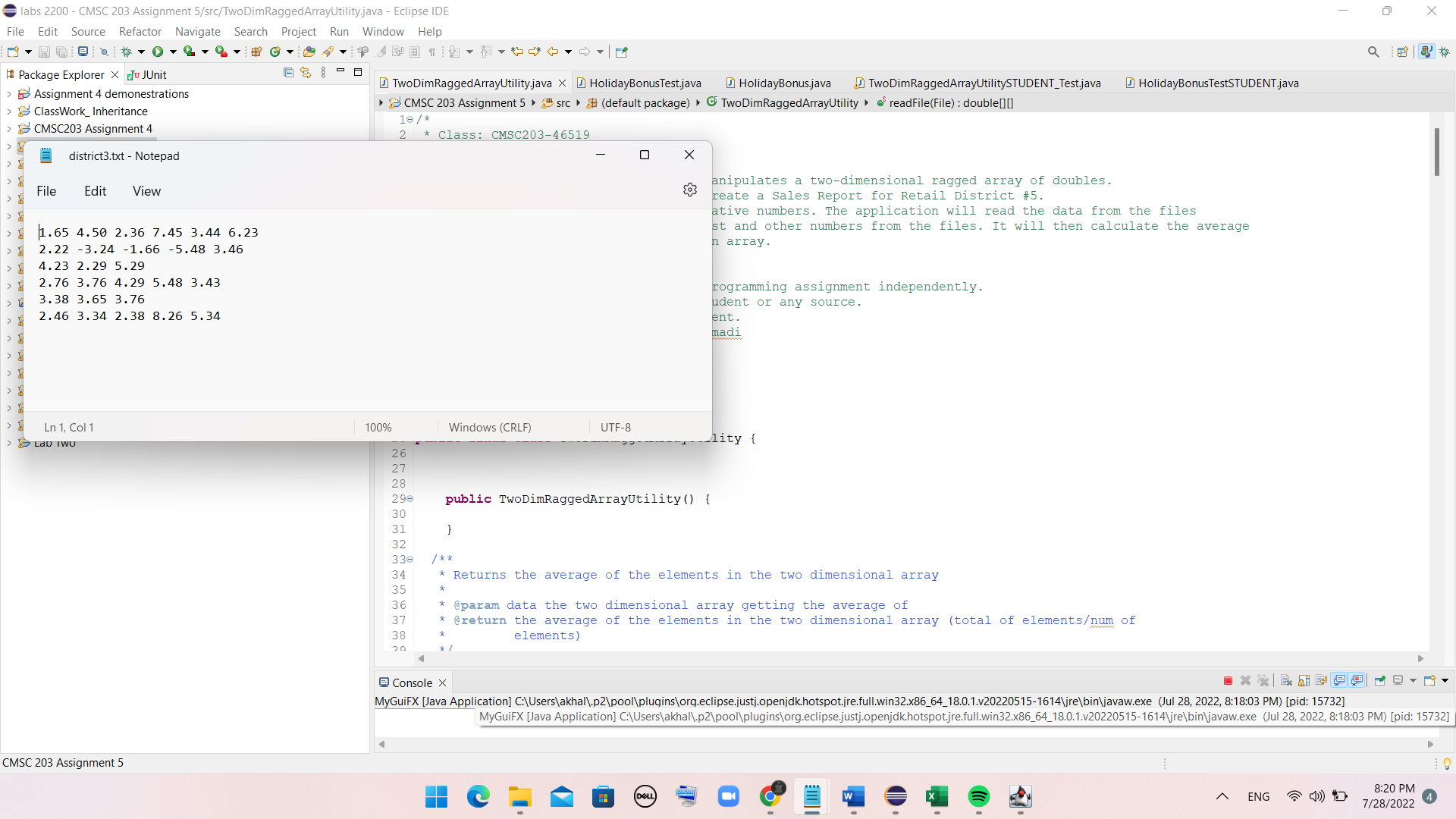
File containing sales data:



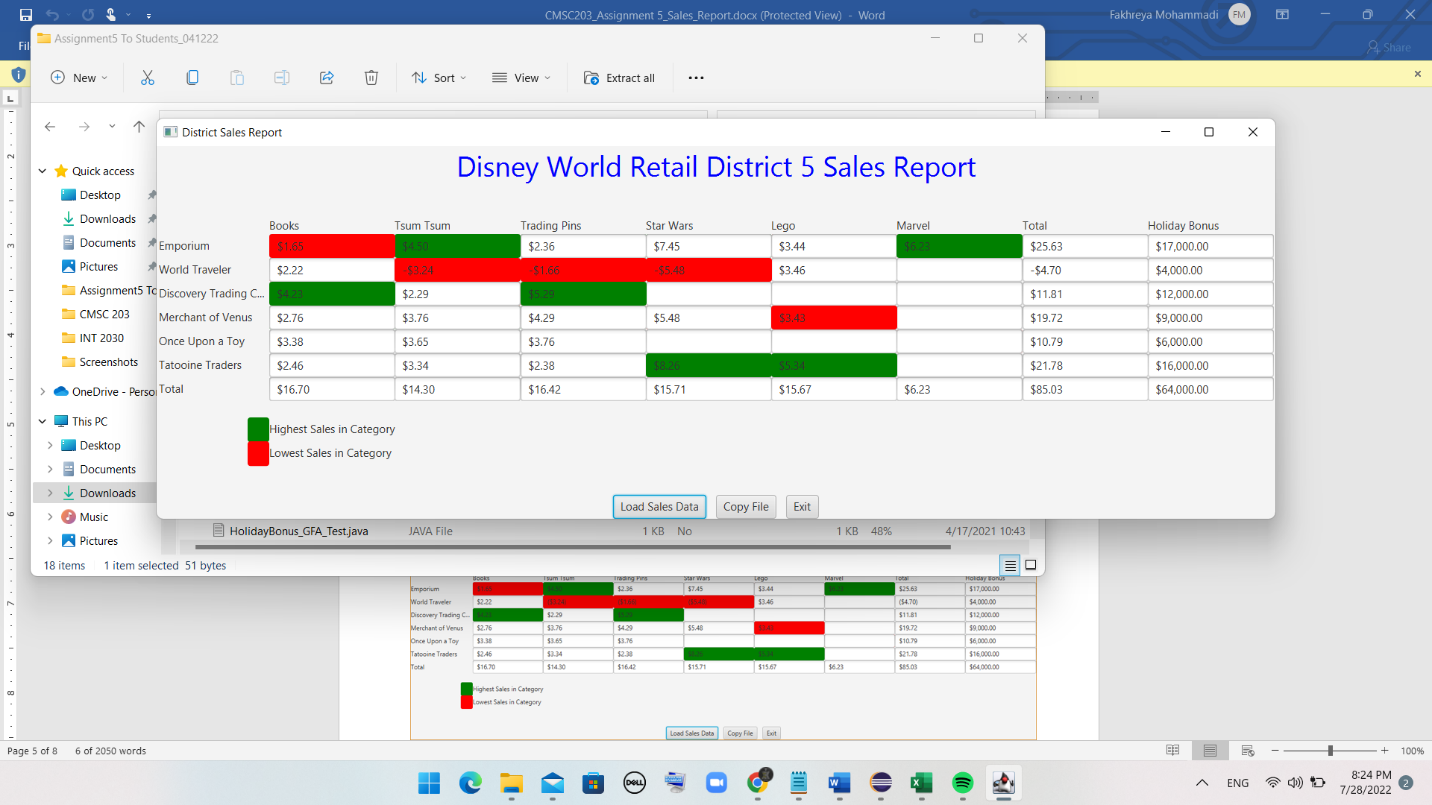
Result after selecting Load Sales Data:



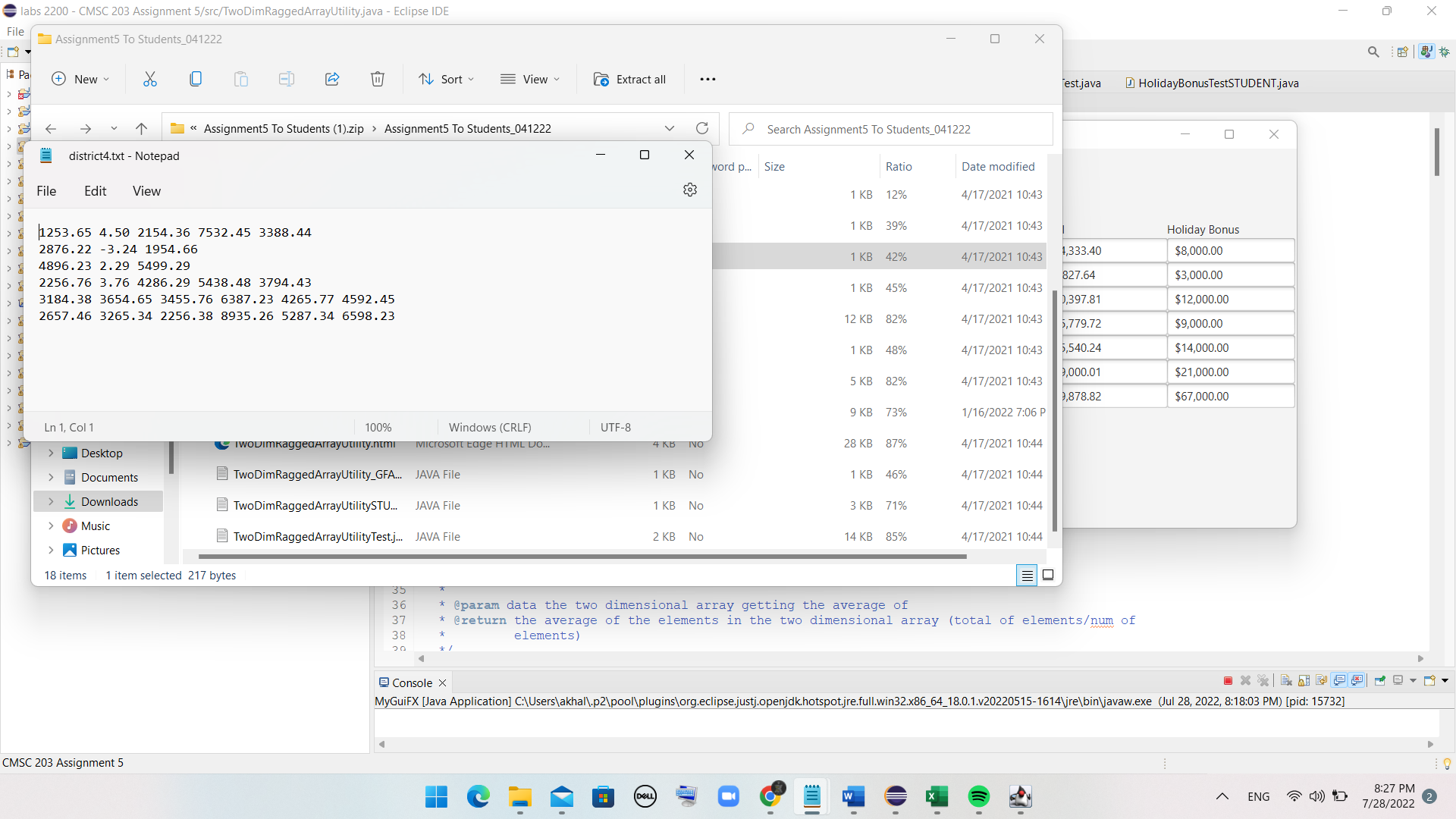
File containing sales data (including negative numbers):



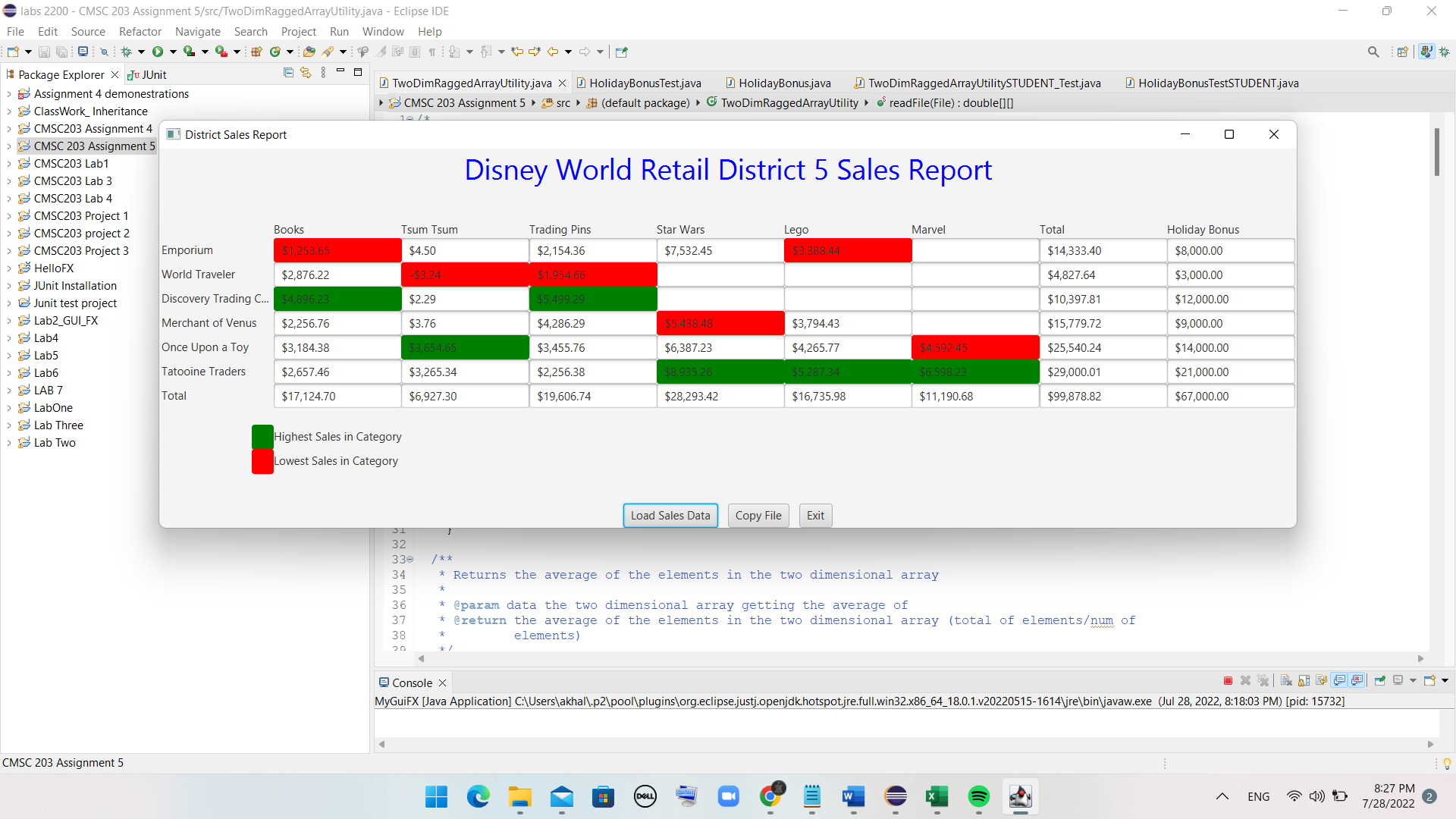
Result after selecting Load Sales Data:



File containing sales data (including negative numbers):



Result after selecting Load Sales Data:



**Part 6: Test Cases**

**Test case for TwoDimRaggedArrayUtilitySTUDENT\_Test**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | Input | Actual output | Expected output | Did test pass? |
| 1 | {10.0, 20.0, 30.0, 40.0}, {50.0, 60.0, 40.0, 70.0}, {100.0, 90.0} | Total: 510.0  Average: 51.0  Total of row 1: 220.0  Total of column 0:  160.0  Highest in array:  100.0  Write to file: 10.0 20.0 30.0 40.0, 50.0 60.0 40.0 70.0, 100.0 90.0 | Total: 510.0  Average: 51.0  Total of row 1: 220.0  Total of column 0:  160.0  Highest in array:  100.0  Write to file: 10.0 20.0 30.0 40.0, 50.0 60.0 40.0 70.0, 100.0 90.0 | Yes |
| 2 | {-10.0, 30.0, 40.0, -50.0}, {90}, {50.0}, {20.5, 70.5, 30.0, 40.5} | Total: 311.5  Average: 31.15  Total of row 0:  10.0  Total of column 0:  150.5  Highest in array:  90.0  Write to file: -10.0 30.0 40.0 -50.0, 90, 50.0, 20.5 70.5 30.0 40.5 | Total: 311.5  Average: 31.15  Total of row 0:  10.0  Total of column 0:  150.5  Highest in array:  90.0  Write to file: -10.0 30.0 40.0 -50.0, 90, 50.0, 20.5 70.5 30.0 40.5 | Yes |
| 3 | {100.0, 20.0, 45.5, 30.0, 90.0}, {-9, -5}, {-4, 7.0, 3.0, 2.5, 4.0}, {1.0, 50.0, 40.0} | Total: 375.0  Average: 25.0  Total of row 2:  12.5  Total of column 1:  72.0  Highest in array:  100.0  Write to file:  100.0 20.0 45.5 30.0 90.0, -9 -5, -4 7.0 3.0 2.0 4.0, 1.0 50.0 40.0 | Total: 375.0  Average: 25.0  Total of row 2:  12.5  Total of column 1:  72.0  Highest in array:  100.0  Write to file:  100.0 20.0 45.5 30.0 90.0, -9 -5, -4 7.0 3.0 2.0 4.0, 1.0 50.0 40.0 | Yes |

**Test case for HolidayBonusTestSTUDENT**

**testCalculateHolidayBonusSTUDENTA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | input | Actual output | Expected output | Did test pass? |
| 1 | {1.0, 2.0, 3.0, 4.0}, {10.0, 65.0, 32.9, 43.0}, {43.98, 435.08} | Result 0: 4000.0  Result 1: 14000.0  Result 2:  10000.0 | Result 0: 4000.0  Result 1: 14000.0  Result 2:  10000.0 | Yes |
| 2 | {12.0, 23.0, 23.9, 44.34, 34.4}, {34.0}, {-1.0, 33.0, 34.34, -42.0}, {132.0, 343.0} | Result 0: 14000.0  Result 1: 2000.0  Result 2:  7000.0 | Result 0: 14000.0  Result 1: 2000.0  Result 2:  7000.0 | Yes |
| 3 | {-3.34, 34.34, 123.04, 43.34, 434.0}, {123.4, -3,0}, {34.45, 54.6, -43.0, 45.56} | Result 0: 14000.0  Result 1: 2000.0  Result 2:  9000.0 | Result 0: 14000.0  Result 1: 2000.0  Result 2:  9000.0 | Yes |

**testCalculateHolidayBonusSTUDENTB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | input | Actual output | Expected output | Did test pass? |
| 1 | {1.0, 2.0, 3.0, 4.0}, {10.0, 65.0, 32.9, 43.0}, {43.98, 435.08} | Result 0: 1000.0  Result 1: 3000.0  Result 2:  2000.0 | Result 0: 1000.0  Result 1: 3000.0  Result 2:  2000.0 | Yes |
| 2 | {12.0, 23.0, 23.9, 44.34, 34.4}, {34.0}, {-1.0, 33.0, 34.34, -42.0}, {132.0, 343.0} | Result 0: 3000.0  Result 1: 500.0  Result 2:  1500.0 | Result 0: 3000.0  Result 1: 500.0  Result 2:  1500.0 | Yes |
| 3 | {-3.34, 34.34, 123.04, 43.34, 434.0}, {123.4, -3,0}, {34.45, 54.6, -43.0, 45.56} | Result 0: 3000.0  Result 1: 500.0  Result 2:  2000.0 | Result 0: 3000.0  Result 1: 500.0  Result 2:  2000.0 | Yes |

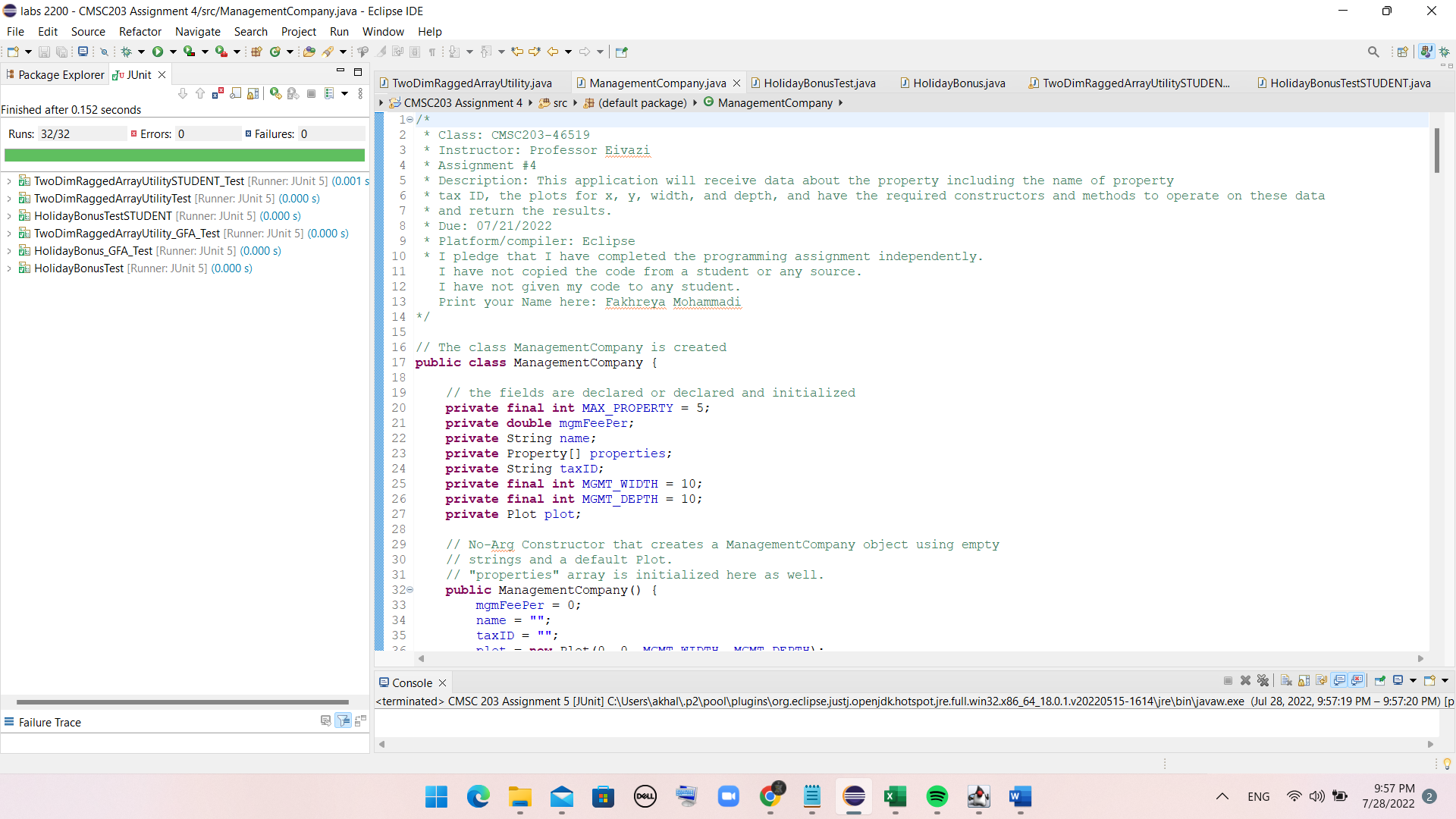
**testCalculateTotalHolidayBonusA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | input | Actual output | Expected output | Did test pass? |
| 1 | {1.0, 2.0, 3.0, 4.0}, {10.0, 65.0, 32.9, 43.0}, {43.98, 435.08} | 28000.0 | 28000.0 | Yes |
| 2 | {12.0, 23.0, 23.9, 44.34, 34.4}, {34.0}, {-1.0, 33.0, 34.34, -42.0}, {132.0, 343.0} | 33000.0 | 33000.0 | Yes |
| 3 | {-3.34, 34.34, 123.04, 43.34, 434.0}, {123.4, -3,0}, {34.45, 54.6, -43.0, 45.56} | 37000.0 | 8000.0 | Yes |

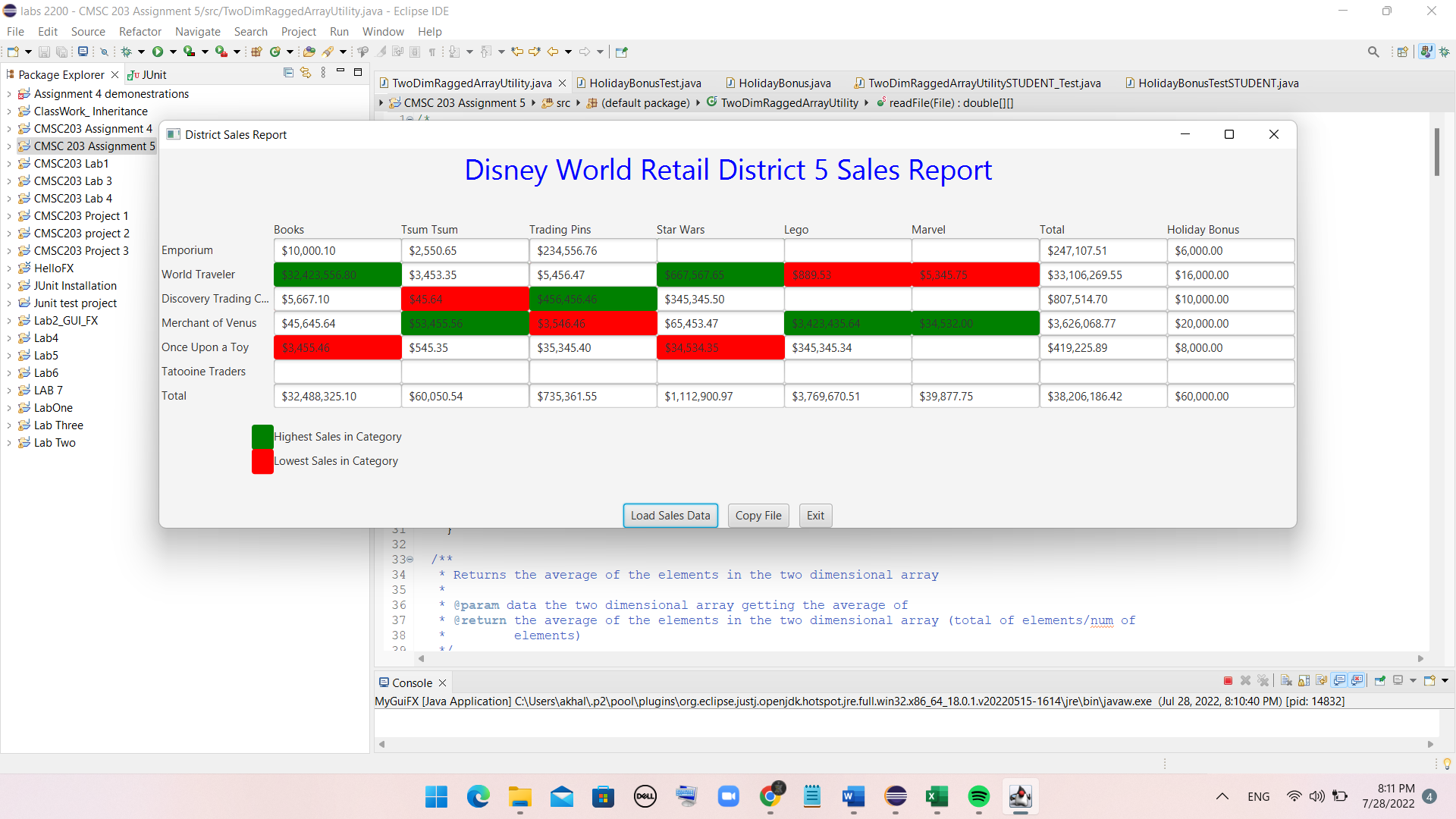
**testCalculateTotalHolidayBonusB**

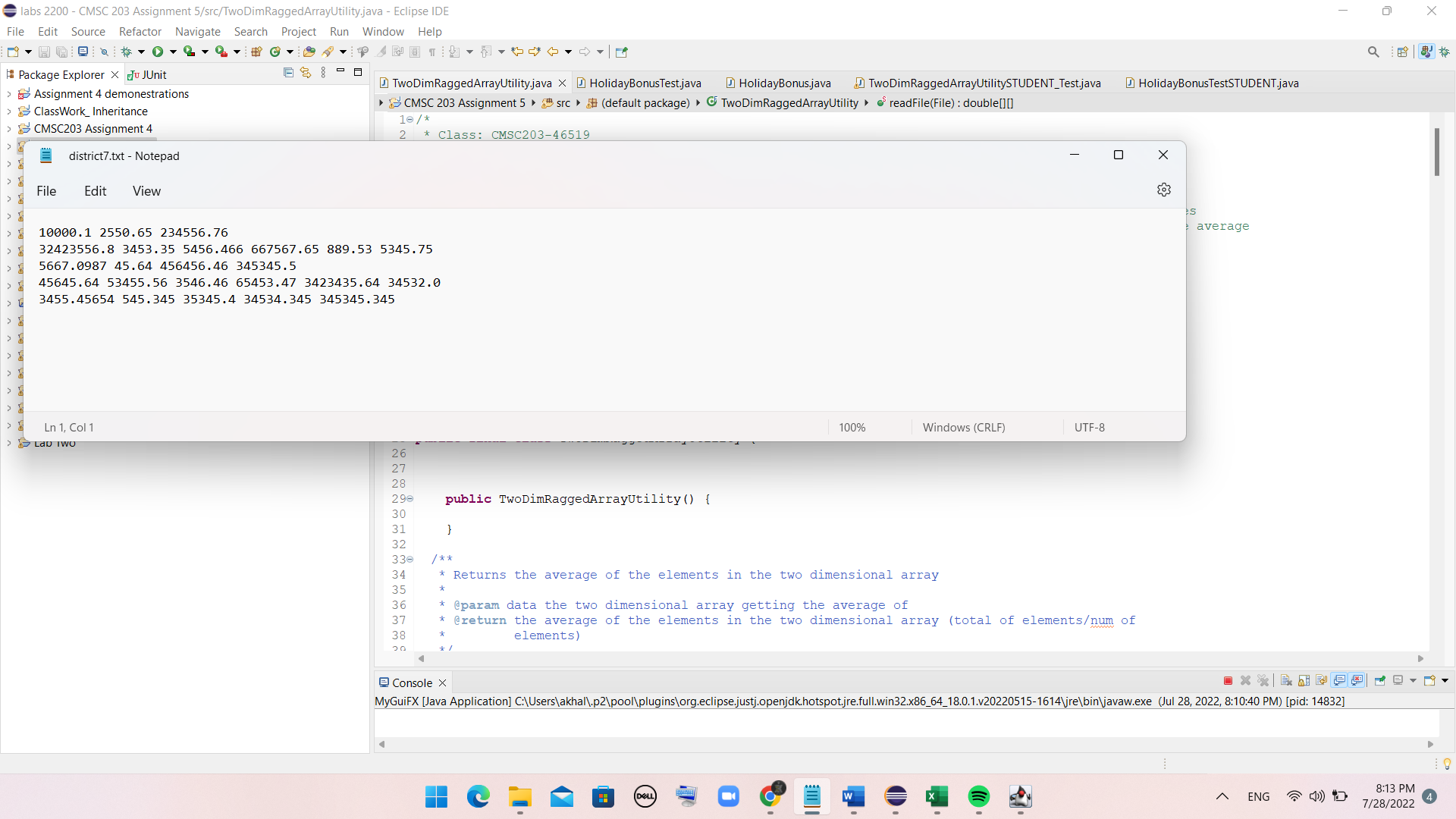
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | input | Actual output | Expected output | Did test pass? |
| 1 | {1.0, 2.0, 3.0, 4.0}, {10.0, 65.0, 32.9, 43.0}, {43.98, 435.08} | 6000.0 | 6000.0 | Yes |
| 2 | {12.0, 23.0, 23.9, 44.34, 34.4}, {34.0}, {-1.0, 33.0, 34.34, -42.0}, {132.0, 343.0} | 7000.0 | 7000.0 | Yes |
| 3 | {-3.34, 34.34, 123.04, 43.34, 434.0}, {123.4, -3,0}, {34.45, 54.6, -43.0, 45.56} | 8000.0 | 8000.0 | Yes |

Screenshot of test cases plan:

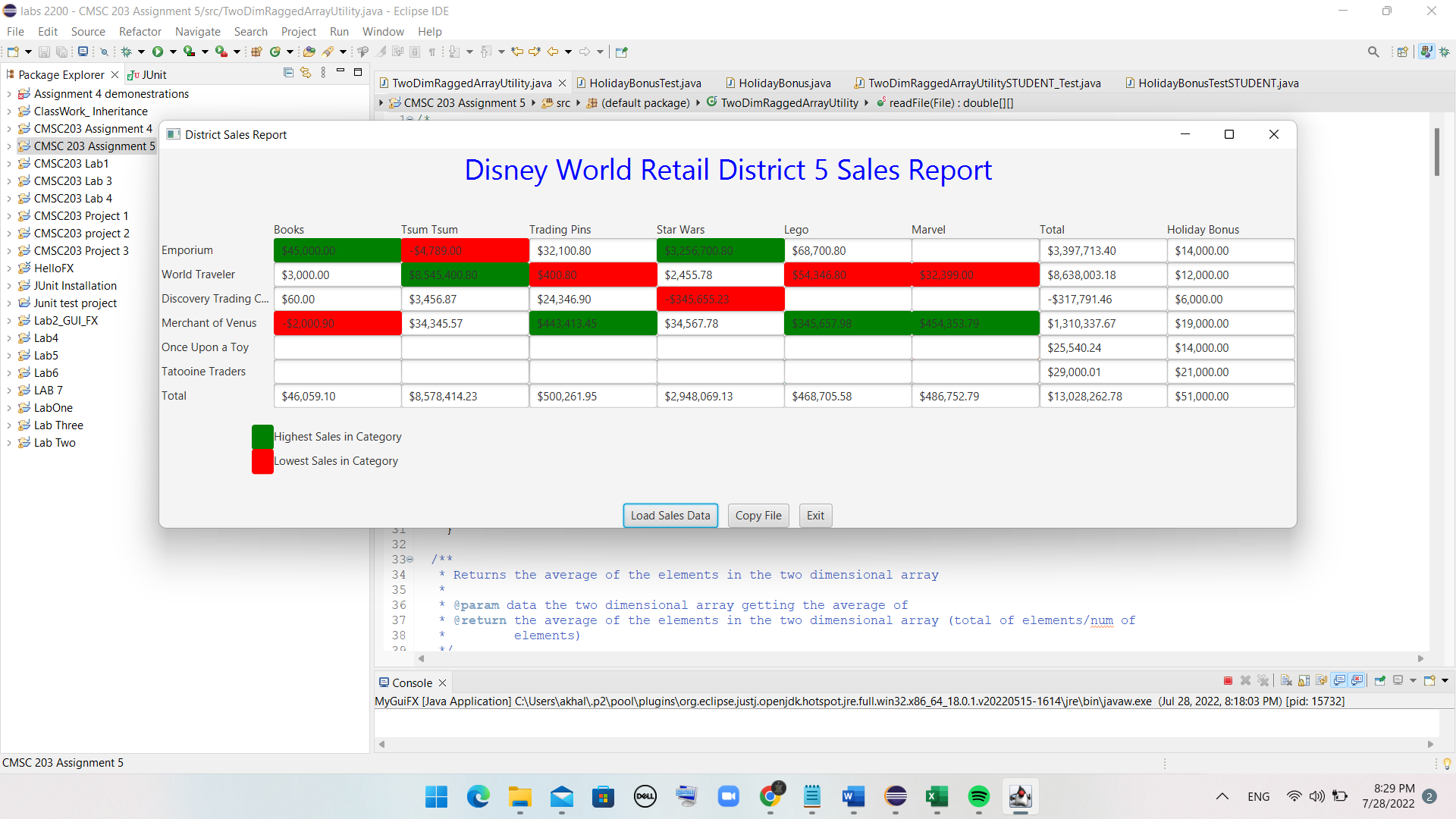
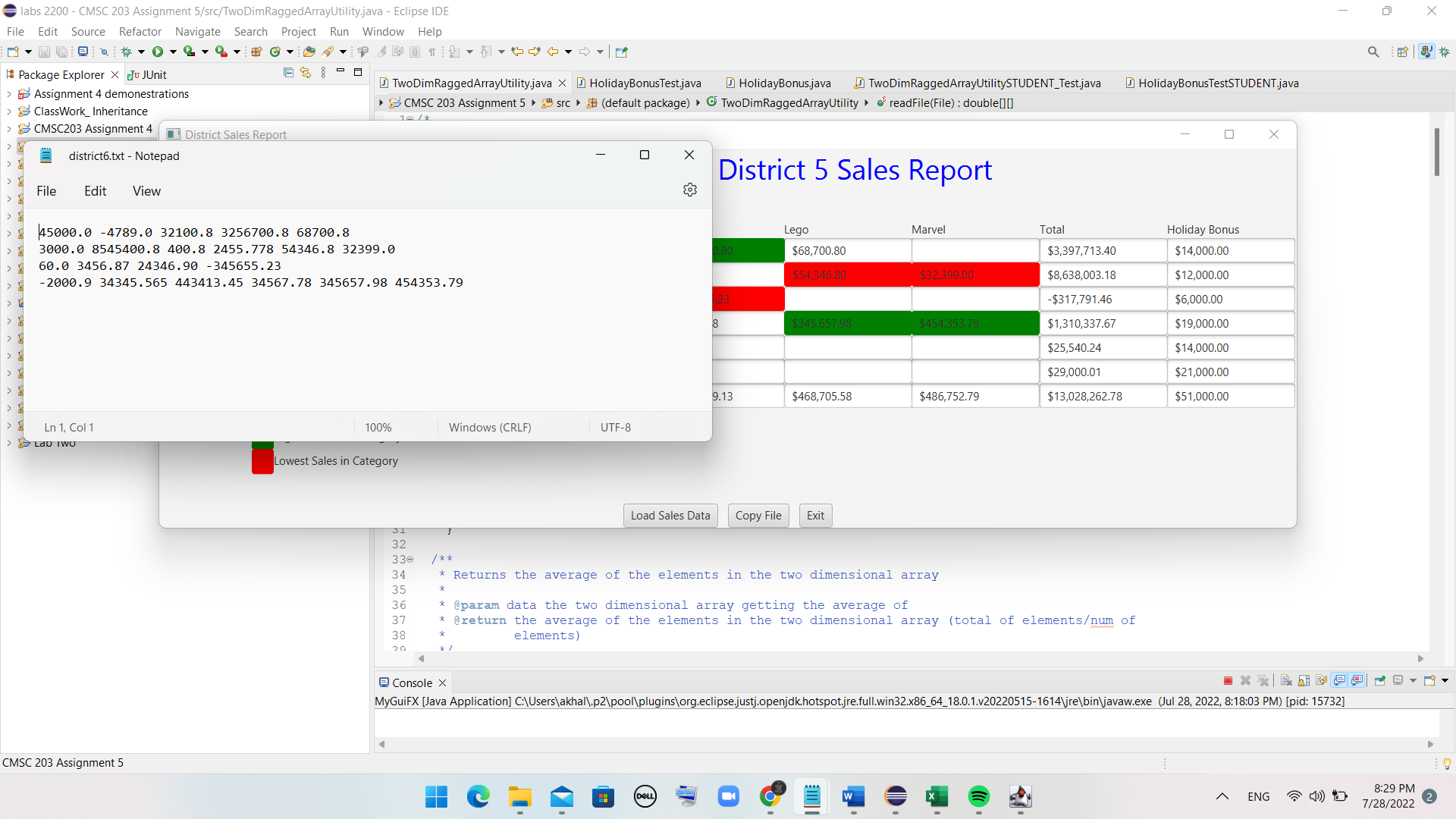


**Part 7: Two additional input files and screenshot of the results:**





File containing sales data (including negative numbers):



* **Lessons Learned: highlight your lessons learned and learning experience from working on this project.**

My experience with this assignment was very good. I was doing well in almost all parts of the assignment, for example, creating methods, passing arrays as arguments, and Junit testing. However, there were some parts I was struggling with to solve them; For instance, writing the writeToFile and readFile methods was challenging. Nevertheless, by studying the provided videos and instructions, I was able to write the codes for the mention challenging methods. Overall, working on this assignment was enjoyable.

* What have you learned?
* I learned how to create classes based on Javadoc. I understand how to implement Two Dimensional Ragged Arrays, how to pass arrays to and from methods, how to Create a Utility class (static methods), how to create and run JUnit testing, and how to read from a file and writing to a file
* What did you struggle with?

I struggled with the creation of some methods for reading and writing data in the files. Additionally, the method that we had to write in order to calculate bonus was a little challenging.

* What will you do differently on your next project?

I will use the office hours of instructor and the coach to discuss assignment as early as possible.

* Include what parts of the project you were successful at, and what parts (if any) you were not successful at.

I was successful with all parts of assignment.

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 |  |