**Assignment 5**

**Stamatia Vafeas**

Class: CMSC203 CRN 46667

Program: Assignment 5

Instructor: Dr. Grinberg

Description: The program looks at stores within Disney world and determines which store sold the most in each category and the size of the individual store’s bonuses.

Due: 08/16/2020

I pledge that I have completed the programming assignment independently.

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

I have not given my code to any student.

# Assumptions made for this project

For this project, I assumed that if the lowest value in a column was negative then no store within the list received the lowest bonus. I also assumed that none of the stores would sell the exact same amount, but if they did then the bonus would go to the first store in the column. For example, if I have the column below, then I would expect the bonus to be equivalent to the list in the third column.

|  |  |  |
| --- | --- | --- |
|  | Books | Bonuses |
| Emporium | (1000.0) | None |
| World Traveler | 2000.0 | Other Bonus |
| Discovery Trading | 4000.0 | High Bonus |
| Tatooine Traders | 4000.0 | Other Bonus |

# Pseudocode

TwoDimRaggedArrayUtility.java

Set up methods within the file

Set up constants MAX\_ROW = 10, MAX\_COL = 10

public static double[][] readFile(File file) throws FileNotFoundException

create a double 2-D array called a initialized to null

try

pass in a file

set lines = 0 //to be used to determine number of lines in array

create FileReader object then Scanner object inputFile

create null stringArray with MAX\_ROW and MAX\_COL

use a for loop(b = 0; b <stringArray[b].length; b++)

if the Scanner object inputFile hasNextLine

take that line and split the array by “ “and input into splitString[]

set stringArray[b] = splitString

increment lines

create a double array a[lines][]

for(I = 0; I less than line; increment i) //going through the rows

if the stringArray[i] does not equal null //contains values

then create a[i] equal to new double array at is the length of stringArray[i] //creating an array that is the length of each row

for loop – for every j if j < length of stringArray[i]; j++ //going through columns

set the value of a[i][j] to the parsedDouble of stringArray[i][j]

catch FileNotFoundException

print Stack Trace if file not found

return a

using two for loops, each element is parsed into Doubles and placed into an array for doubles

writeToFile(double[][]a , file)

try

Create a BufferedWriter object to point to the location of the outputFile

For i=0; i< length of data array; increment i

For j=0 ; j< length of the row[i] of data array; increment j

write the value of data[i][j] to the outputFile with a space in between each item

close the file

catch IOException

print stack trace

getTotal (double[][]a )

double total = 0

use a for loop to step through each row

use a for loop to step through each column

add the value at data[row][col] to total

return total

get Average(double [][] a)

initialize integer numElements to zero

use a for loop to step through each row

use a for loop to step through each column

increment numElements

return (divide call getTotal and divide by numElements)

getRowTotal(double [][] a, int row)

pass in double 2-D array and an index for the row

double rowTotal = 0

for loop (int i=0; i<a[row].length; i++) //for each item in the row, moving along the column

add each item to rowTotal

return rowTotal

getColumnTotal(double [][] a, int col)

pass in double 2-D array and an index for the column

double colTotal = 0

for loop(int i=0; i<numberOfColumns; i++)

try

go through loop and add [i][col] to colTotal

catch NullPointerException, allow the method to keep going since not every row will have something in every column

return colTotal

getHighestInRow(double[][] a, int index)

set the highestInRow equal to the item in the first index

use for loop to go through the selected row

if the value in the element is higher than the index

set highestInRow to the value of that element

return highestInRow

getHightestInRowIndex(double [][]a, int row)

initialize integer index to 0

use a for loop to go through the row, stopping when the index is not less than the length of the row

pass data and row into getHighestInRow and if it equals data[row][i]

set index equal to i

return index

getLowestInRow(double[][] a, int index)

set the lowestInRow equal to the item in the first index

use for loop to go through the selected row

if the value in the element is low than the first value

set the Lowest to the value of that element

return lowestInRow

getLowestInRowIndex(double [][]a, int row)

initialize integer index to 0

use a for loop to go through the row, stopping when the index is not less than the length of the row

pass data and row into getLowestInRow and if it equals data[row][i]

set index equal to i

return index

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

set highest to the value of Double.NEGATIVE\_INFINITY //no matter the value of the first

//readable line in the array, it will be greater than negative infinity

step through each row in a column using a for loop

try

if col is greater than length of the row being evaluated

continue

else if highest is less than data[i][col]

set highest to data[i][col]

catch (NullPointerException)

continue

return highest

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

declare integers lines and numCol, set both to zero

declare integers i and j

intitalize double highVal t= getHighestInColumn(data,col)

set lines equal to data.length //number of rows

use a for loop to step through each row

if numCol is less than the length of the row being evaluated

set numCol to the data length

use a for loop to step each column, continue until j is not less than numCol

use a for loop to step through each row, continue until i is not less than lines

try

if j is greater than or equal to the length of the row currently evaluated

continue //accounts for empty spaces in the column

else if data[i][j] is equal to highVal

set index equal to i

catch NullPointerException

continue

return index

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

set lowest to the value of Double.Positive\_INFINITY //no matter the value of the first

//readable line in the array, it will be less than positive infinity

step through each row in a column using a for loop

try

if col is greater than length of the row being evaluated

continue

else if lowest is greater than data[i][col]

set lowest to data[i][col]

catch (NullPointerException)

continue

return lowest

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

declare integers lines and numCol, set both to zero

declare integers i and j

intitalize double lowVal t= getLowestInColumn(data,col)

set lines equal to data.length //number of rows

use a for loop to step through each row

if numCol is less than the length of the row being evaluated

set numCol to the data length

use a for loop to step each column, continue until j is not less than numCol

use a for loop to step through each row, continue until i is not less than lines

try

if j is greater than or equal to the length of the row currently evaluated

continue //accounts for empty spaces in the column

else if data[i][j] is equal to highVal

set index equal to i

catch NullPointerException

continue

return index

getLowestInColumnIndex(double [][]a, int column)

using the same methods are in getLowestInColumn, when the Lowest value is set, the index is set to the index of the Lowest value. This number will continue updating until the loop is complete

use try-catch for a nullPointerException for the elements in the array that might not have anything in them

getHighestInArray(double[][] data)

set double highest equal to negative infinity

using a for loop step through each row

using a for loop step through each column (j<data[i].length)

if highest is less than data[i][j]

set highest to data[i][j]

return highest

getLowestInArray(double[][] data)

set double lowest equal to positive infinity

using a for loop step through each row

using a for loop step through each column (j<data[i].length)

if lowest is greater than data[i][j]

set lowest to data[i][j]

return lowest

HolidayBonus.java

Create fields private double [] bonus

Set up methods within the class

public static double[] calculateHolidayBonus(double[][] data, double high, double low, double other)

set length of bonus array to length of data

declare integer lengthRow]

use a for loop to determine the length of the longest row

if data[i].length is greater than lengthRow

set lengthRow equal to data[i].length

use a for loop to step through each column, the loops ends when the index j is not less than lengthRow

set indexHigh equal to TwoDimRaggedArrayUtility.getHighestInColumnIndex(data, j)

set indexLow equal to TwoDimRaggedArrayUtility.getLowestInColumnIndex(data, j)

try

use a for loop to step through each row, loop ends when there are no more rows (i is the row index)

if columnIndex j is greater than the number of elements in the row

continue

else

if (i is equal to indexHigh AND data[i][j] >0)

high is added to bonus[i]

else if (i is equal to indexLow AND data[i][j] > 0)

low is added to bonus [i]

else if(data[i][j] is greater than 0)

other is added to bonus[i]

catch NullPointerException

continue

return bonus

public static double calculcateTotalHolidayBonus(double[][] data, double high, double low, double other)

set double[] holidayBonus to null

initialize double totalBonus to 0

holidayBonus is initialized with calculcateHolidayBonus(data,high,low,other)

using a for loop to set through holidayBonus

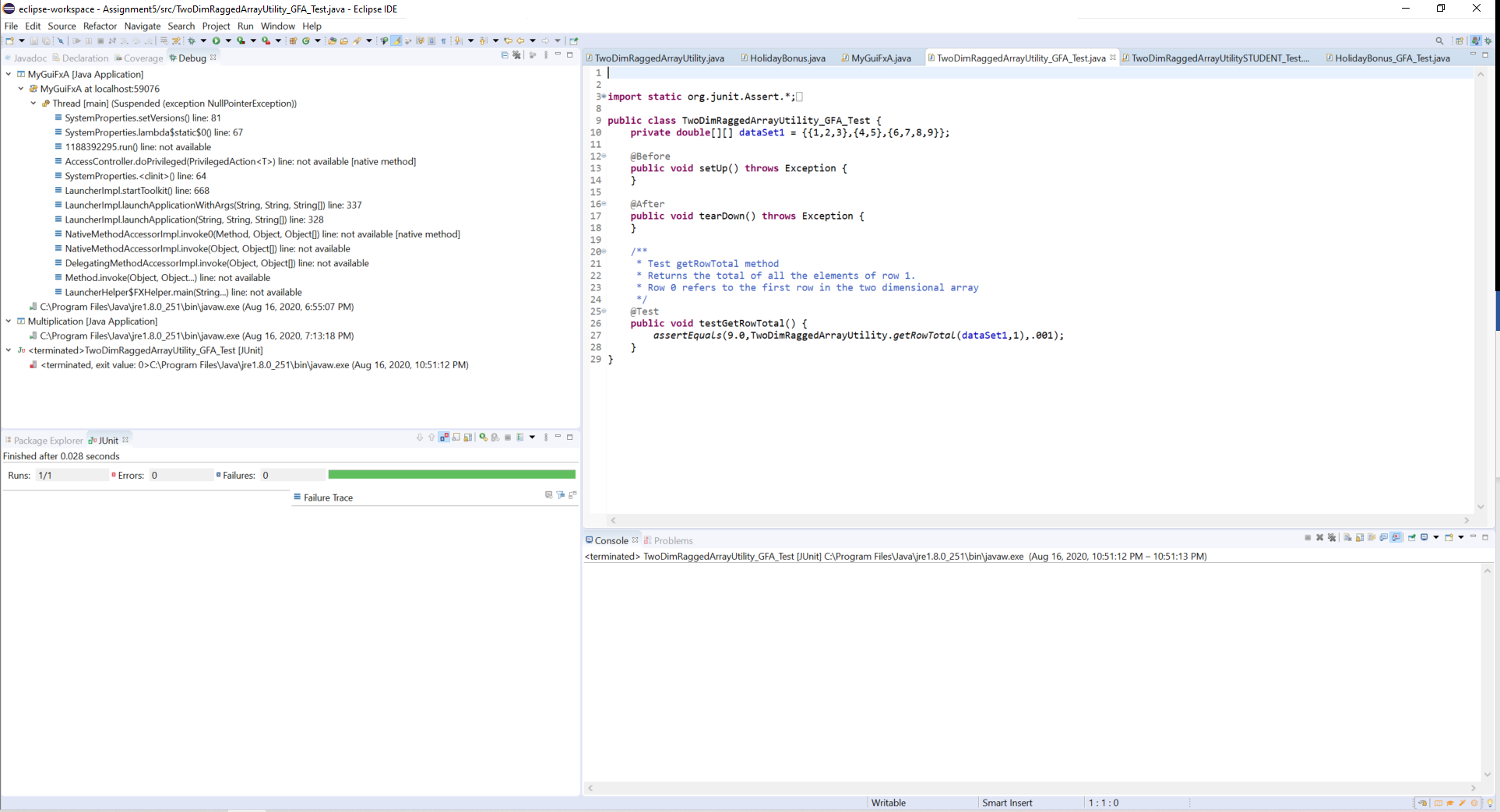
added holidayBonus[i] to totalBonus

return totalBonus

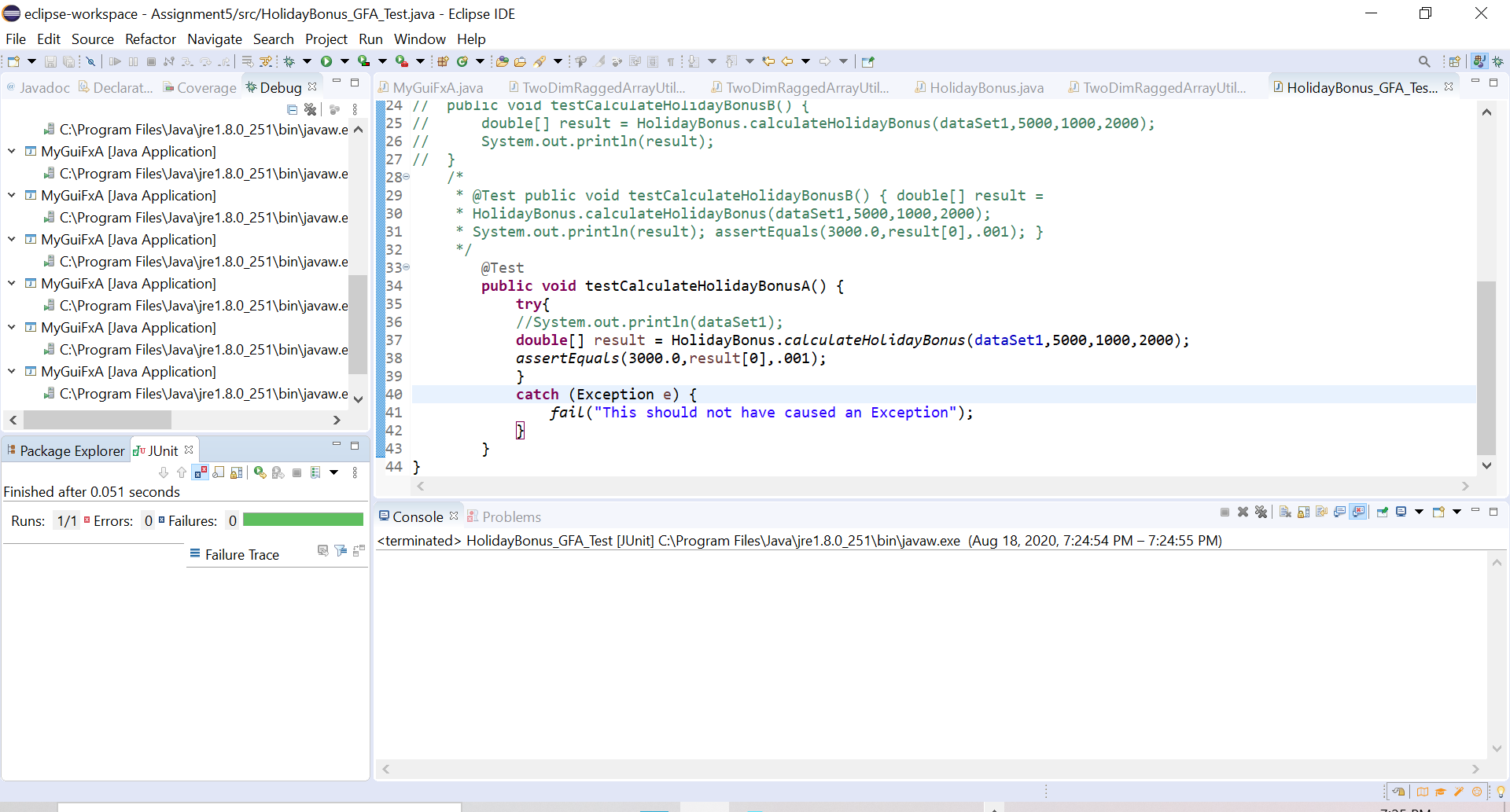
# Test Plan

The first two screenshots below are the output from the provided GFA test files

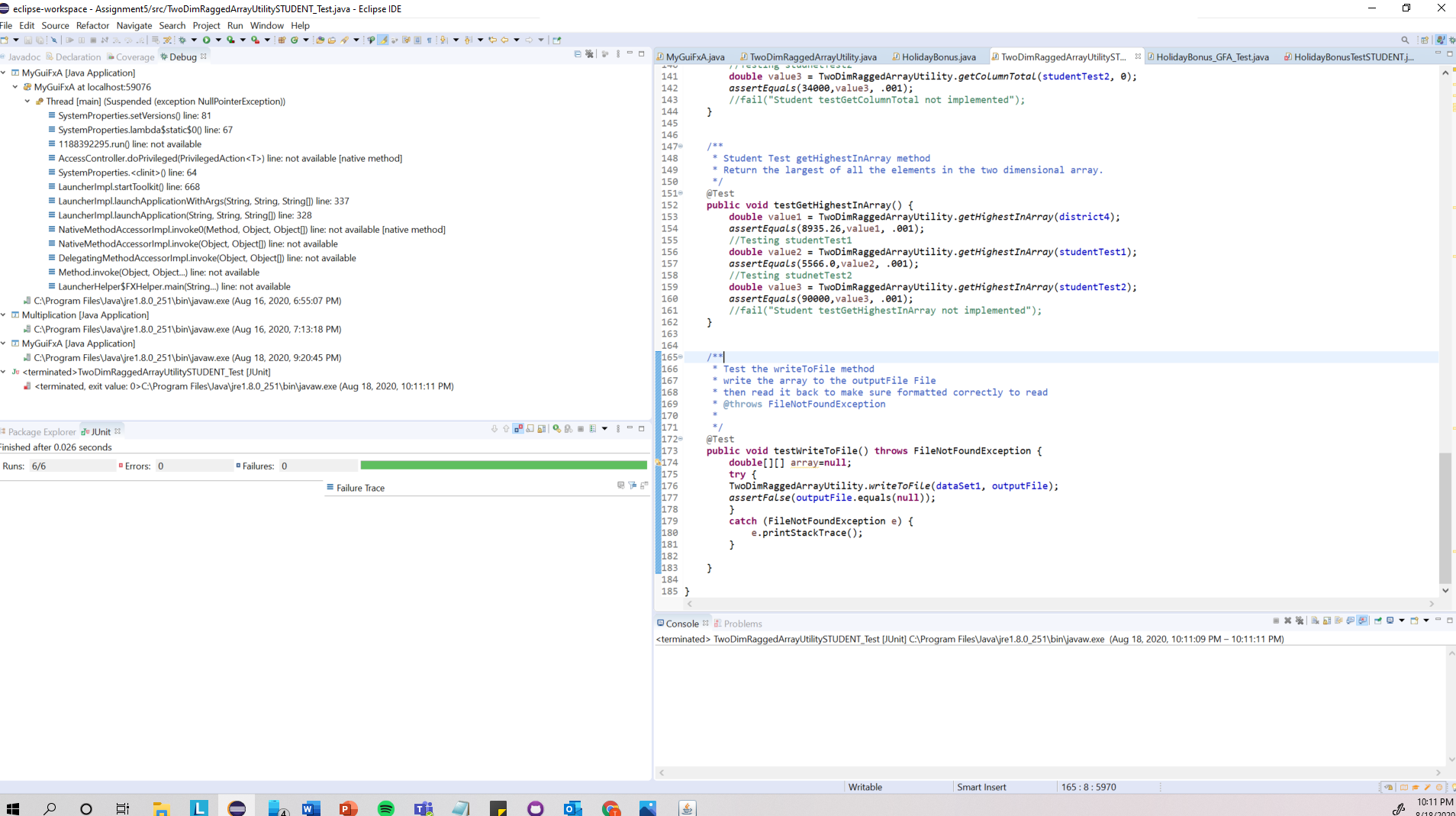
TwoDimRaggedUtility\_GFA\_Test.java



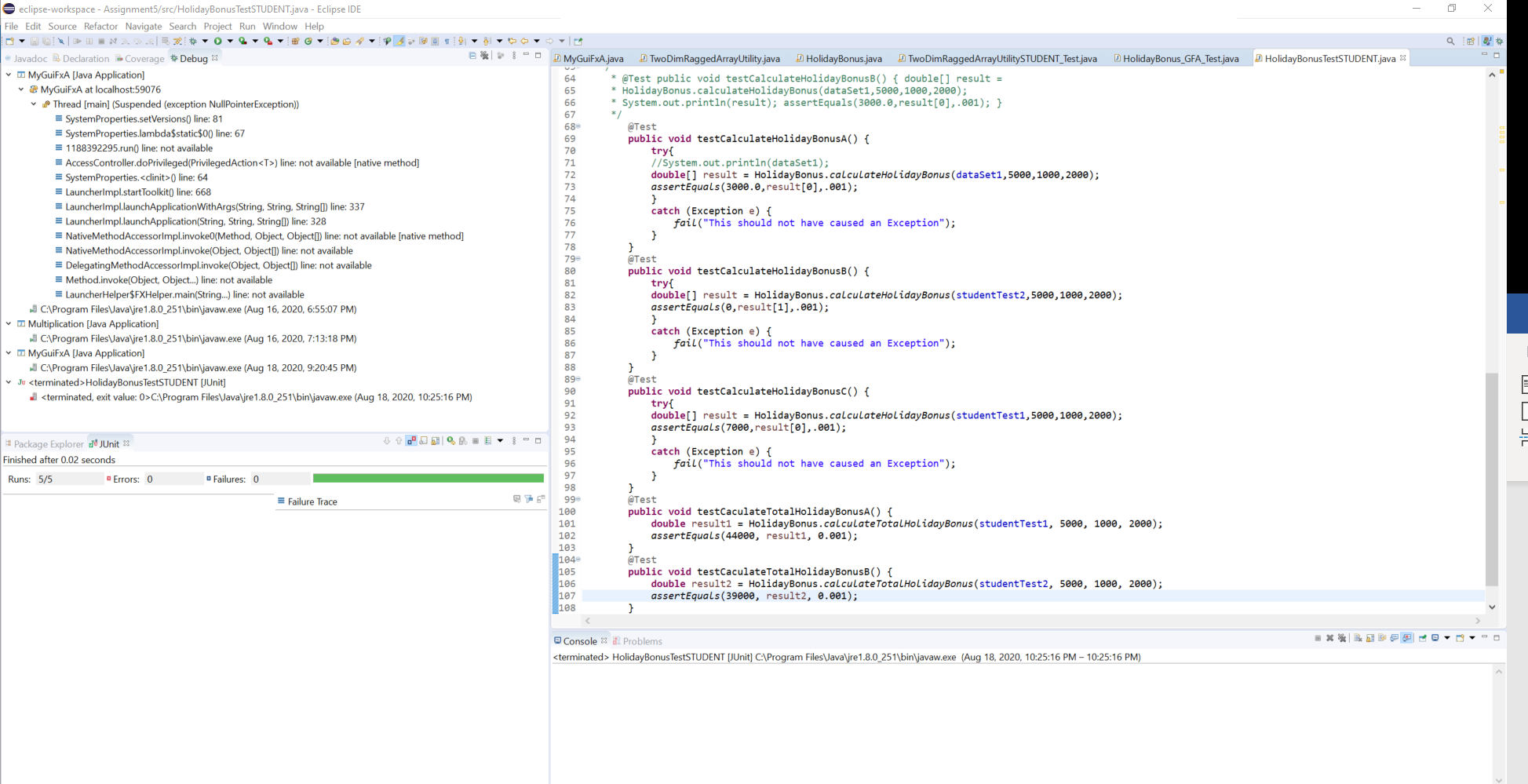
HolidayBonus\_GFA\_Test.java



In order to test TwoDimRaggedArrayUtility.java, I took the arrays provided and implemented them into my TwoDimRaggedArrayUtilitySTUDENT.java. I was able to successfully pass all the tests created, please see the attached TwoDimRaggedArrayUtilitySTUDENT.java file for the tests.

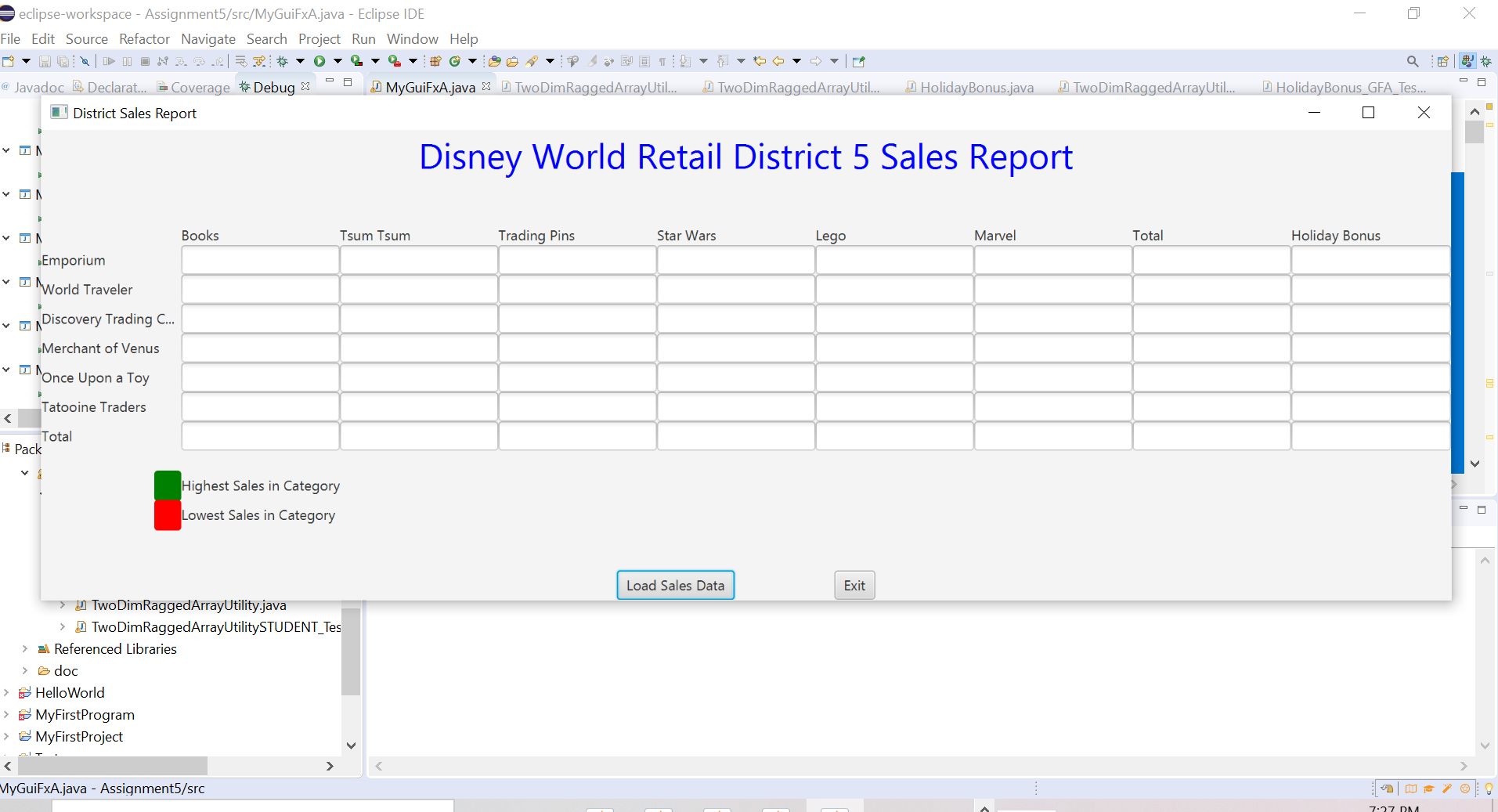


In order to test HolidayBonus.java, I copied all of the arrays implemented in the other test file and dropped them into HolidayBonusTEST.java. Using their values, I tested the two functions within HolidayBonus. See HolidayBonusTEST.java for the tests.

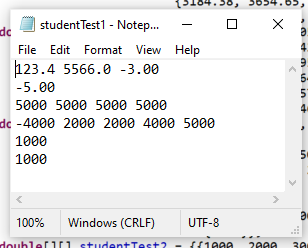


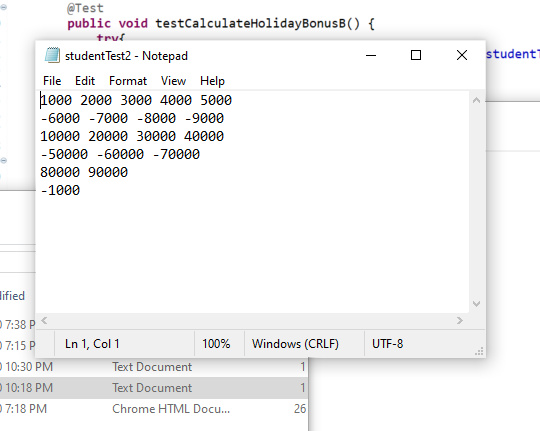
Test Cases

Screenshot of Load Data Popup



In order to test the Sales Report output, I created two txt files, studentTest1.java and studentTest2.java. They are shown below

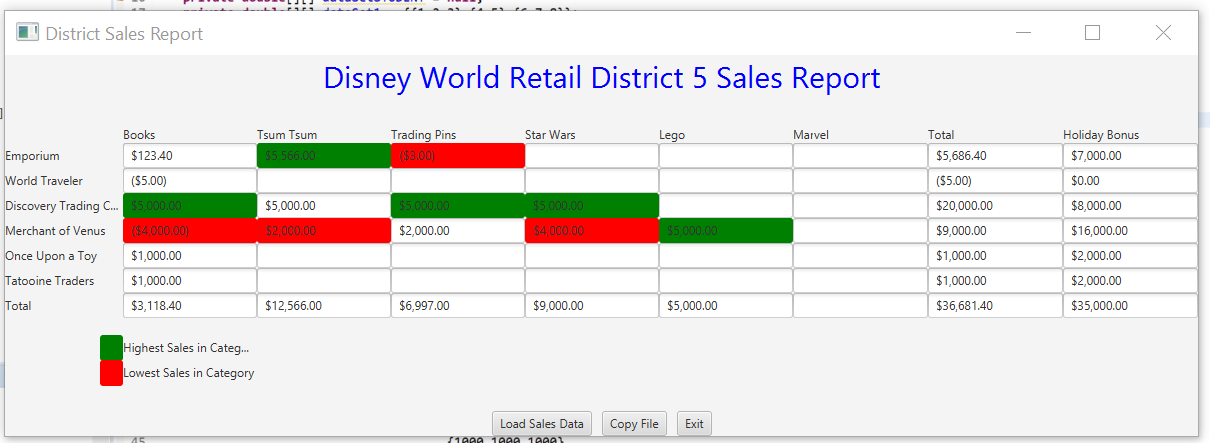




TEST CASE 1

For Test Case 1, the input for load file will be studentTest1.java. The output should look like the table below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Books | Tsum Tsum | Trading Pins | Star Wars | Lego | Marvel | Total | Holiday Bonus |
| Emporium | $123.40 | $5,566.00 | ($3.00) |  |  |  | $5,686.40 | $7,000.00 |
| World Traveler | ($5.00) |  |  |  |  |  | ($5.00) | $0.00 |
| Discovery Trading Company | $5,000.00 | $5,000.00 | $5,000.00 | $5,000.00 |  |  | $20,000.00 | $8,000.00 |
| Merchant of Venus | ($4,000.00) | $2,000.00 | $2,000.00 | $4,000.00 | $5,000.00 |  | $9,000.00 | $16,000.00 |
| Once Upon a Toy | $1,000.00 |  |  |  |  |  | $1,000.00 | $2,000.00 |
| Tatooine Traders | $1,000.00 |  |  |  |  |  | $1,000.00 | $2,000.00 |
| Total | $3,118.40 | $12,566.00 | $6,997.00 | $6,997.00 | $5,000.00 |  | $36,681.40 | $35,000.00 |

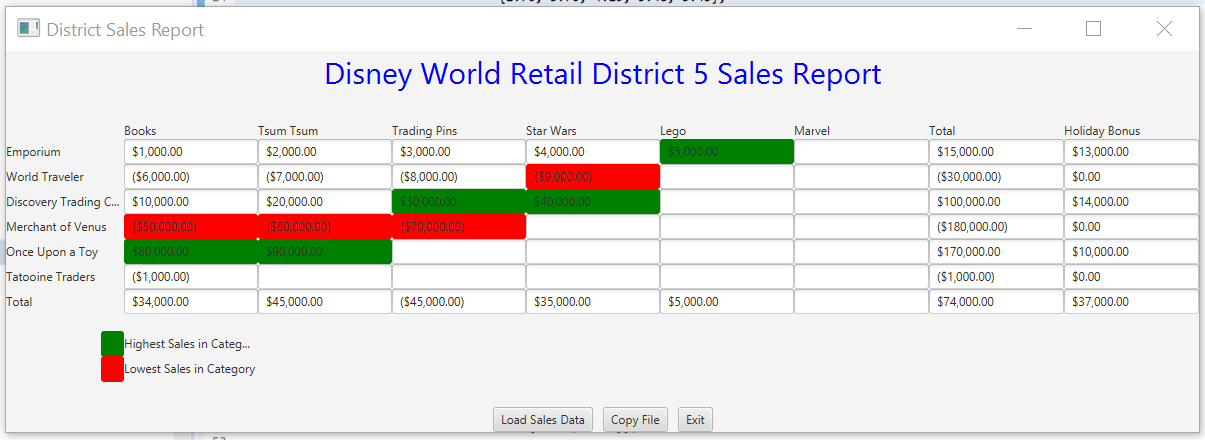


TEST CASE 2

For Test Case 2, the input for load file will be studentTest2.java. The output should look like the table below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Books | Tsum Tsum | Trading Pins | Star Wars | Lego | Marvel | Total | Holiday Bonus |
| Emporium | $1,000.00 | $2,000.00 | $3,000.00 | $4,000.00 | $5,000.00 |  | $15,000.00 | $13,000.00 |
| World Traveler | ($6,000.00) | ($7,000.00) | ($8,000.00) | ($9,000.00) |  |  | ($30,000.00) | $0.00 |
| Discovery Trading Company | $10,000.00 | $20,000.00 | $30,000.00 | $40,000.00 |  |  | $100,000.00 | $14,000.00 |
| Merchant of Venus | ($50,000.00) | ($60,000.00) | ($70,000.00) |  |  |  | ($180,000.00) | $0.00 |
| Once Upon a Toy | $80,000.00 | $90,000.00 |  |  |  |  | $170,000.00 | $10,000.00 |
| Tatooine Traders | ($1,000.00) |  |  |  |  |  | ($1,000.00) | $0.00 |
| Total | $34,000.00 | $45,000.00 | ($45,000.00) | $35,000.00 | $5,000.00 |  | $74,000.00 | $37,000.00 |

Actual Output for Test Case 1



Github Submission

# Learning Experience

For this project, I really struggled with passing arrays into the methods. I think that is the only thing that stands in between me and a functioning project. I created a main method within HolidayBonus to run the example provided in the test file and that showed that a pointer to the array was passed into my program, not the array itself. And using 2 for loops to extract the individual elements of the array was not successful. In programming courses, I need to ensure that I complete more of my project early so that when I run into an issue I cannot solve, I have time to reach out for help.

# Checklist

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