Assignment 5

Huan Shiuan Huang

CSMC 203

Grigoriy A. Grinberg

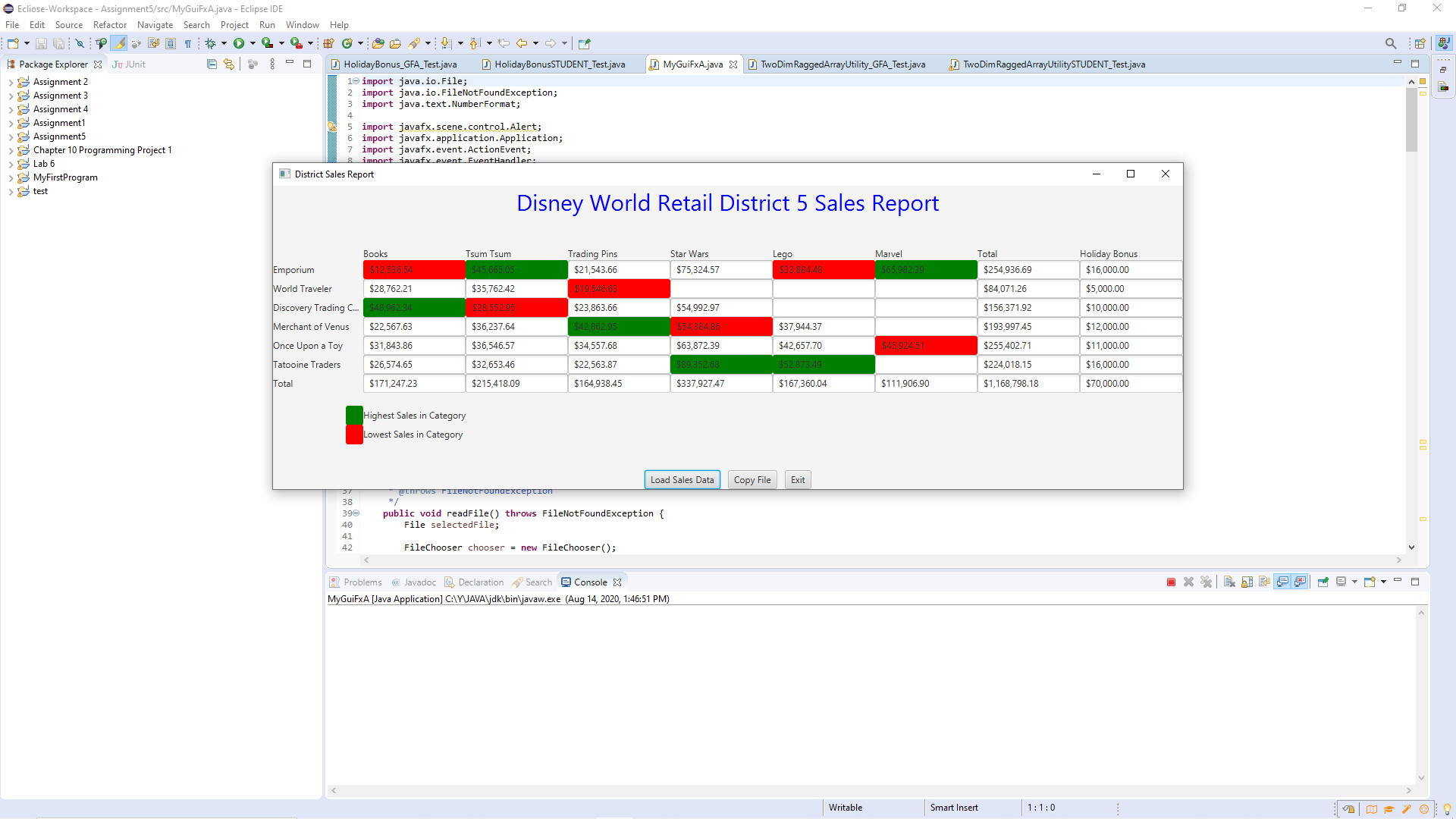
8/16/2020

Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Actual Input** | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 | { 1.1 },  { 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 },  { 8.8, 9.9 } | { 1.1 },  { 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 },  { 8.8, 9.9 } | $1.10 $1.10 $1000.00  $2.20 $3.30 $4.40 $5.50 $6.60 $7.70 $29.70 $23000.00  $8.80 $9.90 $18.70 $10000.00  $12.10 $13.20 $4.40 $5.50 $6.60 $7.70 $49.50 $34000.00 | $1.10 $1.10 $1000.00  $2.20 $3.30 $4.40 $5.50 $6.60 $7.70 $29.70 $23000.00  $8.80 $9.90 $18.70 $10000.00  $12.10 $13.20 $4.40 $5.50 $6.60 $7.70 $49.50 $34000.00 | Yes |
| 2 | { 1.1, 2.2, 3.3 },  { 4.4, 5.5, 6.6 },  { 7.7, 8.8, 9.9 } | { 1.1, 2.2, 3.3 },  { 4.4, 5.5, 6.6 },  { 7.7, 8.8, 9.9 } | $1.10 $2.20 $3.30 $6.60 $3000.00  $4.40 $5.50 $6.60 $16.50 $6000.00  $7.70 $8.80 $9.90 $26.40 $15000.00  $13.20 $16.50 $19.80 $49.50 $24000.00 | $1.10 $2.20 $3.30 $6.60 $3000.00  $4.40 $5.50 $6.60 $16.50 $6000.00  $7.70 $8.80 $9.90 $26.40 $15000.00  $13.20 $16.50 $19.80 $49.50 $24000.00 | Yes |
| 3 | { 1.0, 2.0, 3.0, 4.0 },  {5.0, 6.0 },  { 7.0, 8.0, 9.0 },  {10.0, 11.0},  {12.0, 13.0, 14.0} | { 1.0, 2.0, 3.0, 4.0 },  {5.0, 6.0 },  { 7.0, 8.0, 9.0 },  {10.0, 11.0},  {12.0, 13.0, 14.0} | $1.00 $2.00 $3.00 $4.00 $10.00 $8000.00  $5.00 $6.00 $11.00 $4000.00  $7.00 $8.00 $9.00 $24.00 $6000.00  $10.00 $11.00 $21.00 $4000.00  $12.00 $13.00 $14.00 $36.00 $15000.00  $35.00 $40.00 $26.00 $4.00 $105.00 $37000.00 | $1.00 $2.00 $3.00 $4.00 $10.00 $8000.00  $5.00 $6.00 $11.00 $4000.00  $7.00 $8.00 $9.00 $24.00 $6000.00  $10.00 $11.00 $21.00 $4000.00  $12.00 $13.00 $14.00 $36.00 $15000.00  $35.00 $40.00 $26.00 $4.00 $105.00 $37000.00 | Yes |

Screen snapshots

Loading “District 5”



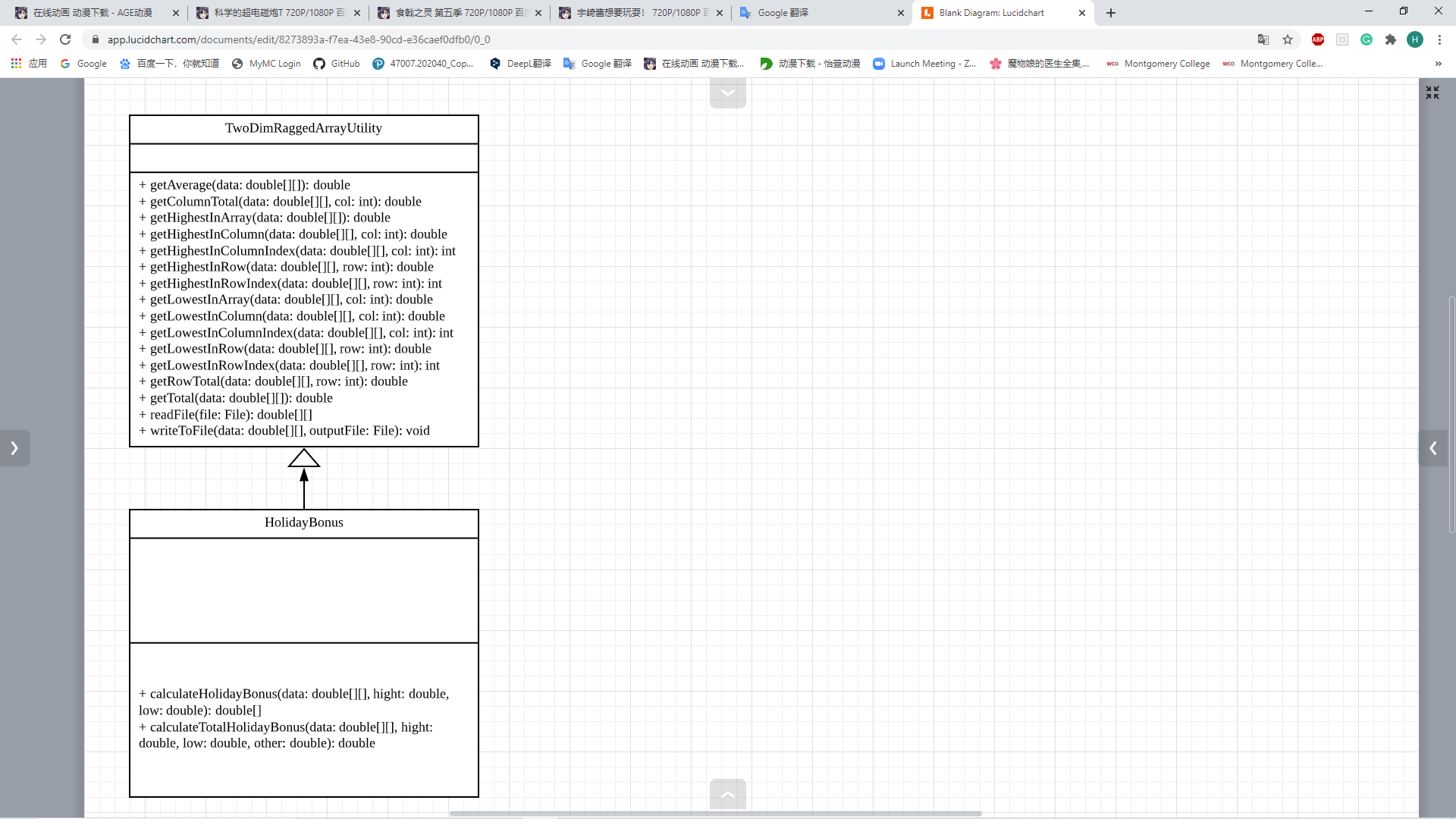
STUDENT Junit test







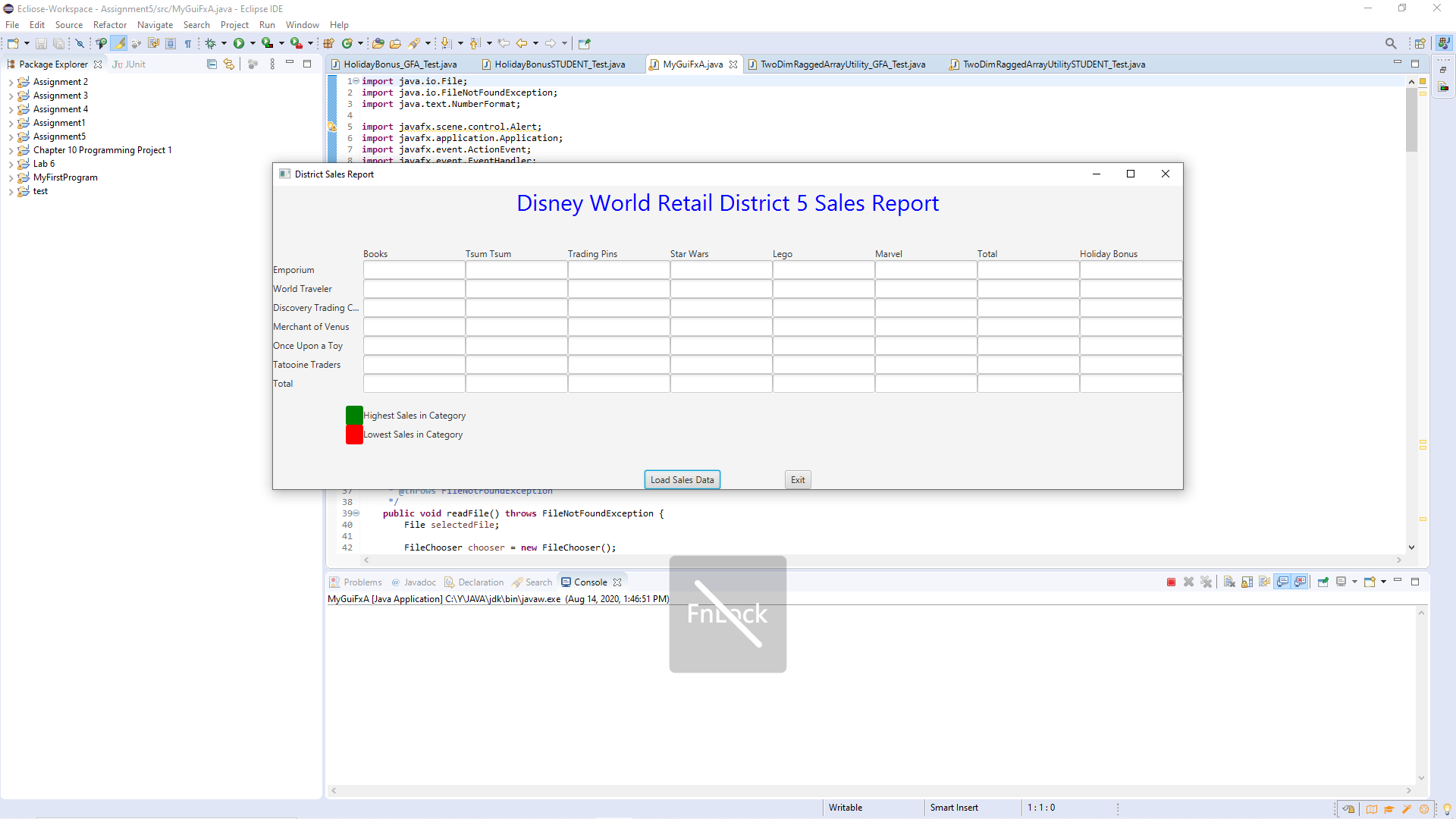
UML Class Diagram



Algorithm

1. TwoDimRaggedArrayUtility Class
2. getAverage (double[][] data) method
   1. declare a double variable called total initialize to 0
   2. declare a double variable called num initialize to 0
   3. for every i from 0 to length of data
      1. for every j from 0 to length of data in row i
         1. num++
         2. total equals to total + data[i][j]
      2. end for
   4. end for
   5. return total / num
3. end method
4. getColumnTotal(double[][] data, int col) method
   1. declare a double variable called total initialize to 0
   2. for every i from 0 to length of data
      1. if there is no this col in this row
         1. continue
      2. end if
      3. total equals to total + data[i][j]
   3. end for
   4. return total
5. end method
6. getHighestInArray(double[][] data) method
   1. declare a double variable called large initialize to data[0][0]
   2. for every i from 0 to length of data
      1. for every j from 0 to length of data in row i
         1. if large is lower than data[i][j]
            1. large = data[i][j]
         2. end if
      2. end for
   3. end for
   4. return large
7. end method
8. getHighestInColumn(double[][] data, int col) method
   1. declare a double variable called large initialize to -999999999
   2. for every i from 0 to length of data
   3. if there is no this col in this row
      1. continue
   4. end if
   5. if large is lower than data[i][j]
      1. large equal to data[i][j]
   6. end if
   7. return large
9. end method
10. getHighestInColumnIndex(double[][] data, int col)
    1. declare a double variable called large initialize to -999999999
    2. declare a int variable called index intialize to 0
    3. for every i from 0 to length of data
       1. if there is no this col in this row
          1. continue
       2. end if
       3. if large is lower than data[i][j]
          1. large equal to data[i][j]
          2. index = i
       4. end if
    4. end for
    5. return index
11. end method
12. getHighestInRow(double[][] data, int row)
    1. declare a double variable called large initialize to -999999999
    2. for every i from 0 to length of row in data
       1. if large is lower than data[row][i]
          1. large equal to data[row][i]
       2. end if
    3. end for
    4. return large
13. end method
14. getHighestInRowIndex(double[][] data, int row) method
    1. declare a double variable called large initialize to -999999999
    2. declare a int variable called index intialize to 0
    3. for every i from 0 to length of row in data
       1. if large is lower than data[row][i]
          1. large equal to data[row][i]
          2. index = i
       2. end if
    4. end for
    5. return index
15. end method
16. getLowestInArray(double[][] data) method
    1. declare a double variable called smallest initialize to data[0][0]
    2. for every i from 0 to length of data
       1. for every j from 0 to length of data in row i
       2. if smallest is larger than data[i][j]
          1. smallest equal to data[i][j]
       3. end if
    3. end for
    4. return smallest
17. end method
18. getLowestInColumn(double[][] data, int col) method
    1. declare a double variable called small initialize to 999999999
    2. for every i from 0 to length of data
       1. if there is no this col in this row
          1. continue
       2. end if
       3. if small larger than data[i][col]
          1. small equal to data[i][col]
       4. end if
    3. end for
    4. return index
19. end method
20. getLowestInColumnIndex(double[][] data, int col) method
    1. declare a double variable called small initialize to 999999999
    2. declare a double variable called index initialize to 0
    3. for every i from 0 to length of data
       1. if there is no this col in this row
          1. continue
       2. end if
       3. if small larger than data[i][col]
          1. small equal to data[i][col]
          2. index = i
       4. end if
    4. end for
    5. return index
21. end method
22. getLowestInRow(double[][] data, int row) method
    1. declare a double variable called small initialize to 999999999
    2. for every i from 0 to length of row in data
       1. if small is larger than data[row][i]
          1. small = data[row][i]
       2. end if
    3. return small
    4. return small
23. end method
24. getLowestInRowIndex(double[][] data, int row)
    1. declare a double variable called small initialize to 999999999
    2. declare a int variable called index intialize to 0
    3. for every i from 0 to length of row in data
       1. if small is larger than data[row][i]
          1. small = data[row][i]
          2. index = i
       2. end if
    4. end for
    5. return index
25. end method
26. getRowTotal(double[][] data, int row) method
    1. declare a double variable called total initialize to 0
    2. 1. total equal to total + data[row][i]
    3. end for
    4. reutrn total
27. end method
28. getTotal(double[][] data) method
    1. declare a double variable called total initialize to 0
    2. for every i from 0 to length data
       1. for every j in length row in data
          1. total equal to total + data[i][j]
       2. end for
    3. end for
    4. return total
29. end method
30. readFile(java.io.File file) method
    1. create a Scanner object called sc
    2. declare a final int variable called MAX\_ROW\_COL initialize to 10
    3. declare a int variable called row initialize to 0
    4. create a 2D ragged array called str with MAX\_ROW\_COL row
    5. for every i if sc.hasNextLine
       1. str[i] reference the sc.nextLine().split(“ “)
       2. row ++
    6. end for
    7. create a 2D ragged array called data initialize “row” row
    8. for ever i from 0 to row
       1. data[i] reference double array initialinze length of str[i]
       2. for every j from 0 to length of str[i]
          1. data[i][j] = str[i][j]
       3. end for
    9. end for
    10. sc.close()
31. end method
32. writeToFile(double[][] data, java.io.File outputFile) method
    1. create a PrintWriter object called out
    2. for every I from 0 to length of data
       1. for every j from 0 to length of data[i]
          1. print data[i][j]
       2. end for
    3. end for
    4. out.cloce()
33. end method
34. end class
35. HolidayBonus Class
36. calculateHolidayBonus(double[][] data, double high, double low, double other)
    1. create a double ragged 2D array called bonus
    2. Declare two int variable called highest, and lowest
    3. For every I from 0 to length of data[i]
       1. For every j from 0 to length of data[i]
          1. Highest equal to TwoDimRaggedArrayUtility.getHighestInColumnIndex(data, j)
          2. Lowest equal to TwoDimRaggedArrayUtility.getLowestInColumnIndex(data, j)
          3. If data[i][j] is lower or equal to 0
             1. Continue
          4. End if
          5. If i is not equal to highest and i is not equal to lowest
             1. Bonus[i] += other
          6. Else if i equsl to highest
             1. Bonus[i] += high
          7. Else
             1. Bonus[i] += low
          8. End if
       2. End for
    4. End for
    5. Return bonus
37. End method
38. calculateTotalHolidayBonus(double[][] data, double high, double low, double other) method
    1. Declare a double variable called toal initialize to 0
    2. Create a double array object called arr by called calculateHolidayBonus method
    3. For every i in arr
       1. Total += i
    4. End for
    5. Return total
39. End method
40. End class

Assumptions



Assume Emporium does not sell Tsum Tsum but does sell Trading Pins. So how to skip this product when reading txt data? Or we can choose a symbol to represent none for that product.

learning experience

I learned how to use ragged array in this project. I need to determine how many rows there are before creating a general array for each row.

The suffering I encountered is that I sometimes confuse row and colnum. Especially in the bonus category. I need to draw a picture to solve it.

In my next project, I will optimize my code. Try to name the variables more appropriately. I successfully completed all the requirements although this project was very challenging.

Check List

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment 5\_Moss.zip | **Yes** |  |
|  | * FirstInitialLastName\_Assignment5\_Completed.zip | **Yes** |  |
|  | **Program compiles** | **Yes** |  |
|  | **Program runs with desired outputs related to a Test Plan** | **Yes** |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Yes** |  |
|  | * Screenshots for each Test case listed in the Test Plan | **Yes** |  |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Yes** |  |
|  | * UML Diagram | **Yes** |  |
|  | * Algorithms/Pseudocode | **Yes** |  |
|  | * Lessons Learned | **Yes** |  |
|  | * Checklist is completed and included in the Documentation | **Yes** |  |