**CMSC203**

**Assignment #5**



Minnie and Mickey are getting ready to send out Holiday bonuses to their hard-working employees in Retail District #5. The bonuses are calculated based on how much each retail store sold in each category. The retail store with the highest amount sold in a category will receive $5,000. The retail store with the lowest amount sold in a category will receive $1,000. All other retail stores in district #5 will receive $2,000. If a retail store didn’t sale anything in a category, or they have a negative sales amount, they are not eligible for a bonus in that category. If only one retail store sold items in a category, they are eligible to receive only the bonus of $5000 for that category.

**Assignment Description**

Create a utility class that manipulates 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. Follow the Javadoc provided.

Create a utility class that will calculate holiday bonuses given a ragged array of doubles which represent the sales for each store in each category. It will also take in bonus amount for the store with the highest sales in a category, the bonus amount for the store with the lowest sales in a category and the bonus amount for all other stores.

These utility classes will be used with an existing GUI class to create a sales report and display holiday bonuses.

Testing of these utility classes will be done with the JUnit tests and the GUI class provided for you.

**Concepts tested by this assignment**

* Creating classes based on Javadoc
* Two Dimensional Ragged Arrays
* Passing arrays to and from methods
* Creating a Utility class (static methods)
* JUnit testing
* Reading from a file
* Writing to a file
* Using methods of the utility class within an existing GUI driver class
  + Must follow Javadoc to implement correctly

**Classes**

**Utility class - TwoDimRaggedArrayUtility**

The class *TwoDimRaggedArrayUtility* will follow the provided Javadoc and will contain the following methods:

1. Method **readFile** – pass in a file and return a two-dimensional ragged array of doubles
2. Method **writeToFile** – pass in a two-dimensional ragged array of doubles and a file, and writes the ragged array into the file. Each row is on a separate line and each double is separated by a space.
3. Method **getTotal** – pass in a two-dimensional ragged array of doubles and returns the total of the elements in the array.
4. Method **getAverage** – pass in a two-dimensional ragged array of doubles and returns the average of the elements in the array (total/num of elements).
5. Method **getRowTotal** – pass in a two-dimensional ragged array of doubles and a row index and returns the total of that row. Row index 0 is the first row in the array.
6. Method **getColumnTotal** - pass in a two-dimensional ragged array of doubles and a column index and returns the total of that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.
7. Method **getHighestInRow** - pass in a two-dimensional ragged array of doubles and a row index and returns the largest element in that row. Row index 0 is the first row in the array.
8. Method **getHighestInRowIndex** - pass in a two-dimensional ragged array of doubles and a row index and returns the index of the largest element in that row. Row index 0 is the first row in the array.
9. Method **getLowestInRow** - a two-dimensional ragged array of doubles and a row index and returns the smallest element in that row. Row index 0 is the first row in the array.
10. Method **getLowestInRowIndex** - a two-dimensional ragged array of doubles and a row index and returns the index of the smallest element in that row. Row index 0 is the first row in the array.
11. Method **getHighestInColumn** - pass in a two-dimensional ragged array of doubles and a column index and returns the largest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.
12. Method **getHighestInColumnIndex** - pass in a two-dimensional ragged array of doubles and a column index and returns the index of the largest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.
13. Method **getLowestInColumn** - pass in a two-dimensional ragged array of doubles and a column index and returns the smallest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.
14. Method **getLowestInColumnIndex** - pass in a two-dimensional ragged array of doubles and a column index and returns the index of the smallest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.
15. Method **getHighestInArray** - pass in a two-dimensional ragged array of doubles and returns the largest element in the array.
16. Method **getLowestInArray** - pass in a two-dimensional ragged array of doubles and returns the smallest element in the array.

**Utility class – HolidayBonus**

The class *HolidayBonus* will contain the following methods:

1. Method **calculateHolidayBonus** – pass in a two-dimensional ragged array of doubles, and the bonus amount for the store with the highest sales in a category, the bonus amount for the store with the lowest sales in a category and bonus amount for all other stores. It will return an array of doubles which represents the holiday bonuses for each of the stores in the district. The first entry in the returned array [0] will represent the holiday bonus for the store at [0] in the two-dimensional ragged array of doubles. You will be using methods from the TwoDimRaggedArrayUtility when needed.
2. Method **calculateTotalHolidayBonus** – pass in a two-dimensional ragged array of doubles, and the bonus amount for the store with the highest sales in a category, the bonus amount for the store with the lowest sales in a category and bonus amount for all other stores. It will return a double which represents the total of all Holiday Bonuses for the District. You will be using methods from the TwoDimRaggedArrayUtility when needed.

**For**

**GUI Application – provided for you**

1. Uses methods of *TwoDimRaggedArrayUtility* and *HolidayBonus*
2. When the *Load Sales Data* button is selected the sales data is read from a file and displayed on the screen with the sales data as well as the totals for each store and the totals for each category. The largest sales for each category is highlighted in green. The smallest sales for each category is highlighted in red. The holiday bonus for each store is displayed as well as the total of holiday bonuses.
3. The file contains a row for each store and each double in the row is separated by a space
4. Student must provide two additional input files and a screenshot of the results of each. Each file will have at least 4 rows and up to 6 numbers on each row. They must represent ragged arrays.

**JUnit Test**

1. Student will implement and submit TwoDimRaggesArrayUtilityTestSTUDENT.java (Template is provided)
2. Student will implement and submit HolidayBonusTestSTUDENT.java

**Assignment Details**

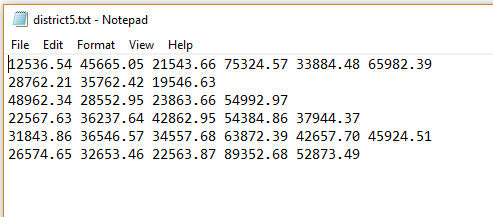
When GUI application starts (provided), user is shown display of Store Names and Item Names

User selects *Load Sales Data* to select the file containing the sales data. The application then displays the sales for each store and each item as well as the totals for the store and the totals for the item. The store with the highest sales for each item will be highlighted.

*Exit* will exit the application.

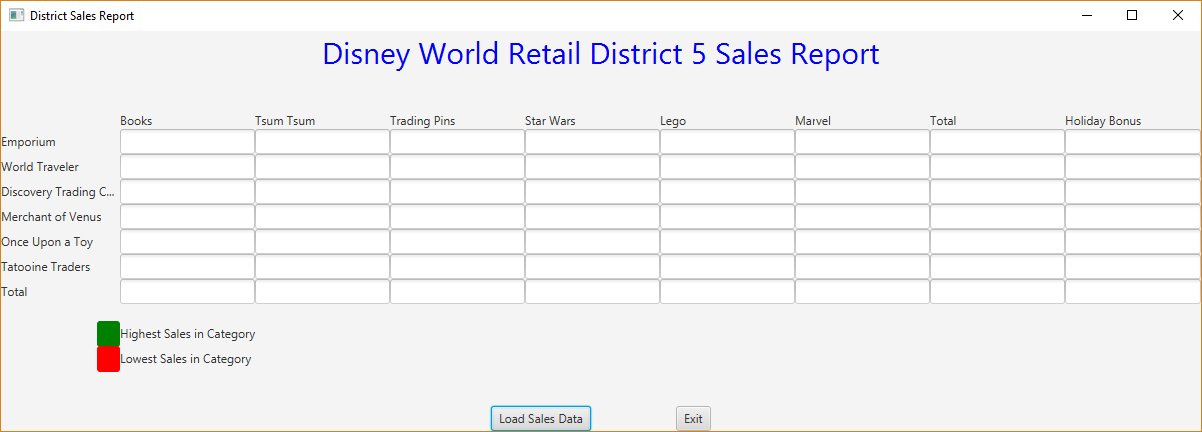
File Format

The file will be in the following format: one store per line, each sales figure is separated by a space.

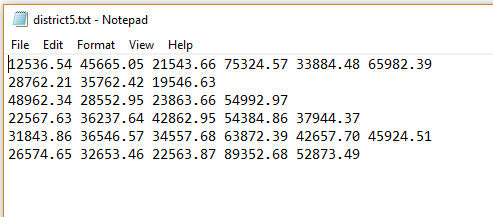


**Examples**

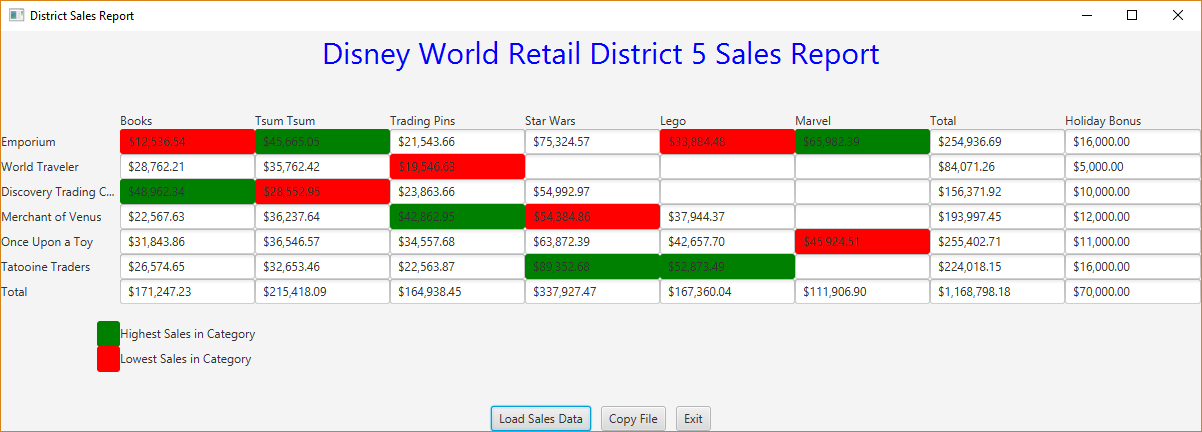
When application starts:



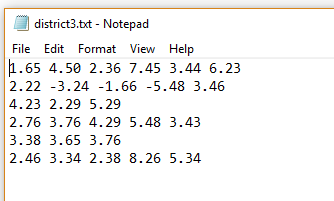
File containing sales data:



Result after selecting Load Sales Data:



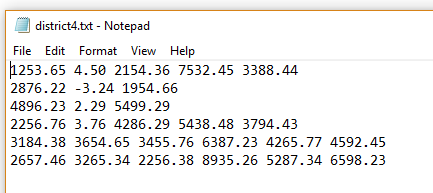
File containing sales data (including negative numbers):



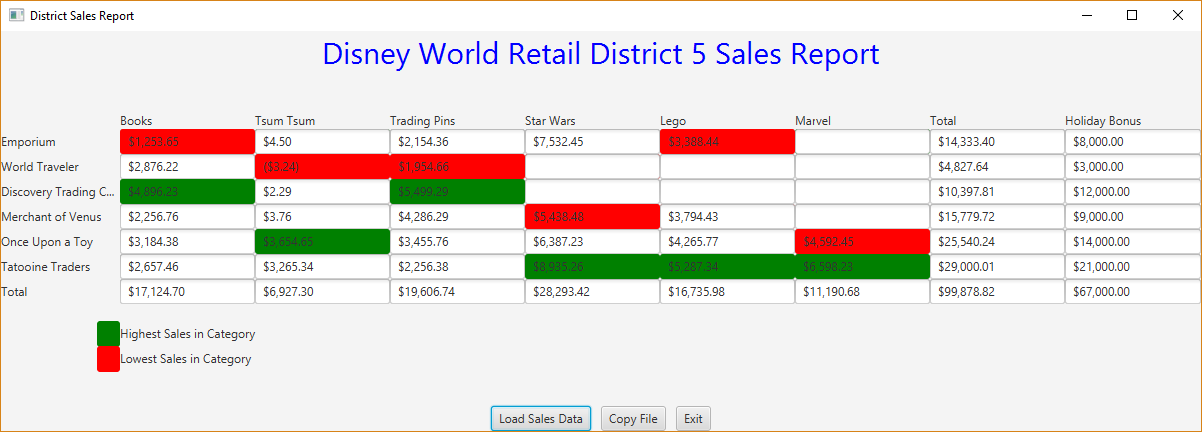
Result after selecting Load Sales Data:



File containing sales data (including negative numbers):



Result after selecting Load Sales Data:



**Deliverables**

**Deliverables / Submissions:**

Design: Turn in a UML diagram that includes box that contains pseudo-code for each of the methods specified in TwoDimRaggedArrayUtility and HolidayBonus classes.

.java. Your pseudo-code should be part-way between English and java. There is no need to spell out all the details of variable declaration, etc., but by the same token, the pseudo-code needs to have enough detail that a competent Java programmer could implement it.

Implementation: Submit a compressed file containing the follow (see below): The Java application (it must compile and run correctly); Javadoc files in a directory; a write-up as specified below. Be sure to review the provided project rubric to understand project expectations. The write-up will include:

* Test Cases (These are the ones you will use in your STUDENT test methods in the JUnit test)
  + Prepare a test table with a list of test cases (expected versus actual results) that you are testing the application with
* Finalized UML diagram
* Any assumptions that you are making for this project
* In three or more paragraphs, highlights of your learning experience (see notes)

**Deliverable format:** The above deliverables will be packaged as follows. Two compressed files in the following formats:

**1st zip file:** FirstInitialLastName\_Assignment5\_Complete.zip, a compressed file containing the following:

* Word document with a name FirstInitialLastName\_Assignment4.docx should include:
  + Finalized UML Class Diagram for all classes
  + Screen snapshots of the GUI with several properties (like screenshots in Assignment 5 Descriptions
  + Screen snapshot of Junit (display test for each method)
  + A screenshot of a computer

    Description automatically generated with medium confidence
  + A screenshot of a computer

    Description automatically generated with medium confidence
  + A screenshot of a computer screen

    Description automatically generated with medium confidence
  + A screenshot of a computer

    Description automatically generated with medium confidence
  + Screen snapshot of GitHub submission
  + Graphical user interface, text, application, email

    Description automatically generated
  + Lessons Learned
  + How to write junit test classes, and how to sort through ragged arrays
  + Check List
* doc (a directory) containing your javadoc HTML files for your classes:
  + TwoDimRaggedArrayUtility.html
  + HolidayBonus.html
  + TwoDimRaggesArrayUtilityTestSTUDENT.html
  + HolidayBonusTestSTUDENT.html
* src (a directory) *contains your files:*
  + TwoDimRaggedArrayUtility.java
  + HolidayBonus.java
  + TwoDimRaggesArrayUtilityTestSTUDENT.java
  + HolidayBonusTestSTUDENT.java

**2nd zip file:** LastNameFirstName\_Assignment5\_Moss.zip, a compressed file containing

the following Java files only:

* + TwoDimRaggedArrayUtility.java
  + HolidayBonus.java
  + TwoDimRaggesArrayUtilityTestSTUDENT.java
  + HolidayBonusTestSTUDENT.java

This folder should contain Java source files that you created or edited only.

**Notes:**

* Learning Experience: highlight your lessons learned and learning experience from working on this project.
* What have you learned?

How to write junit test classes, and how to sort through ragged arrays.

* What did you struggle with?
* What will you do differently on your next project?
* Include what parts of the project you were successful at, and what parts (if any) you were not successful at.
* GitHub: In your repository (see Assignment0), upload your Word file and java file. You will want to upload these files as contents of a directory so that future uploads can be kept separate. Take and submit a screen shot of the GitHub repository.
* Proper naming conventions: All constants, except 0 and 1, should be named. Constant names should be all upper-case, variable names should begin in lower case, but subsequent words should be in title case. Variable and method names should be descriptive of the role of the variable or method. Single letter names should be avoided.
* Documentation: The documentation requirement for all programming projects is one block comment at the top of the program containing the course name, the project number, your name, the date and platform/compiler that you used to develop the project. If you use any code or specific algorithms that you did not create, a reference to its source should be made in the appropriate comment block. Additional comments should be provided as necessary to clarify the program.

Indentation: It must be consistent throughout the program and must reflect the control structure

**Grading Rubric**

See attachment: CMSC203 Assignment 5 Rubric.xlsx

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