**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 class members:

Attributes:

* Three constant values of bonus amounts; 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.

Methods:

1. Method calculateHolidayBonus– pass in a two-dimensional ragged array of doubles. 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. 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.

**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. When the *Copy File* button is selected the loaded file will be copied to the path you select.
4. Each file contains a row for each store and each category in the row is separated by a space

**JUnit Test**

1. For each class listed above, a corresponding GFA test has been provided. GFA (Good Faith Attempt) is the minimum set of requirements for the project. Run each provided JUnit test file and ensure that all tests succeed. Do not modify any of these JUnit tests files, since the instructor will be using the original file(s).
2. For each assignment class that you create, you must create a JUnit test file. Name your test file as the following format: [classname]TestStudent; for example; HolidayBonusTestStudent
3. Make sure your test files cover as much as possible test cases. Ensure your test cases all succeed. Since the instructors will be using their own JUnit test files that thoroughly covers each public method. If you have not tested every single method, your chance of failing a test case would be high.

**Assignment Details**

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

**Do not modify GUI.**

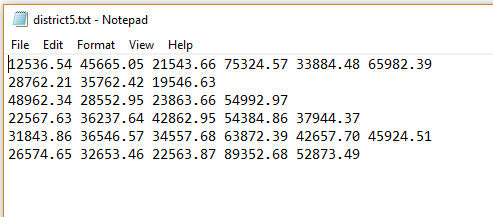
User selects *Load Sales Data* to select the file containing the sales data. The application then displays the sales for each store and each item as well as the totals for the store and the totals for the item. The store with the highest sales for each item will be highlighted.

By selecting *Copy File* button, user has the option of copying the sales data, storing it, modifying the values and loading it.

*Exit* will exit the application.

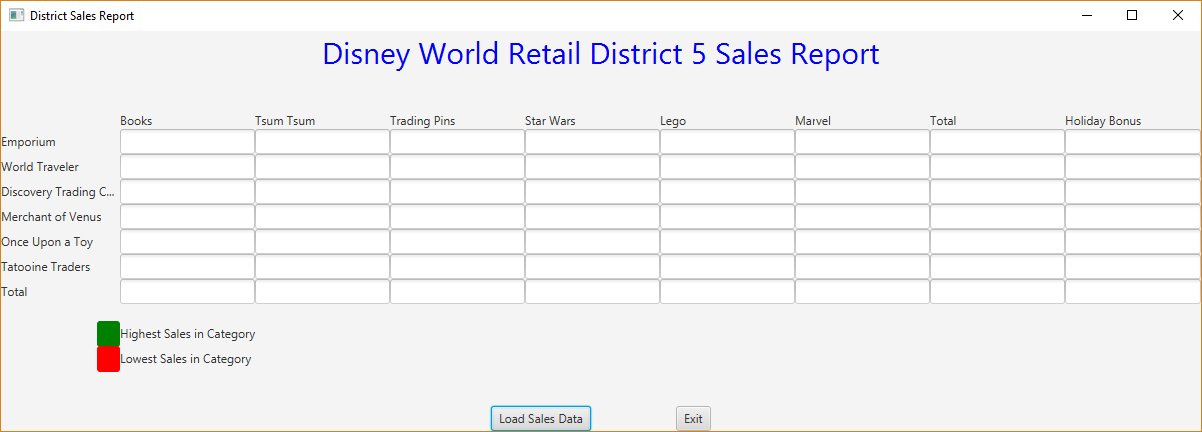
File Format

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

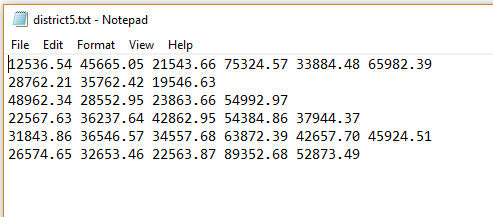


**Examples**

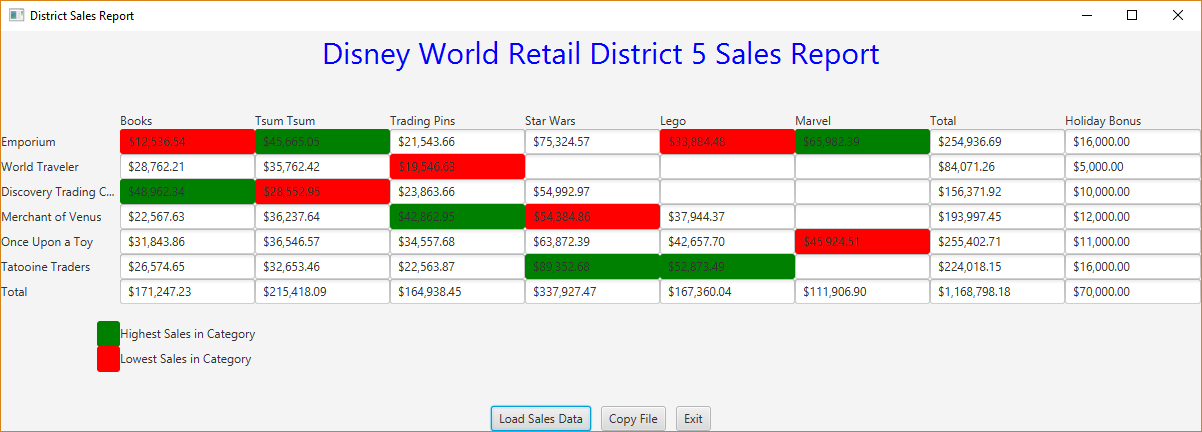
When application starts:



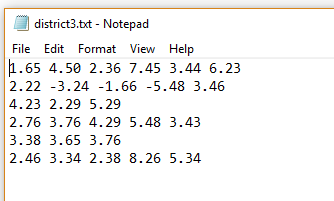
File containing sales data:



Result after selecting Load Sales Data:



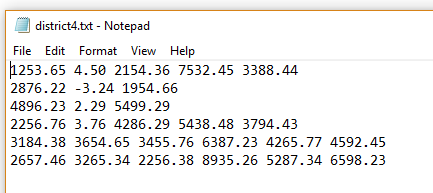
File containing sales data (including negative numbers):



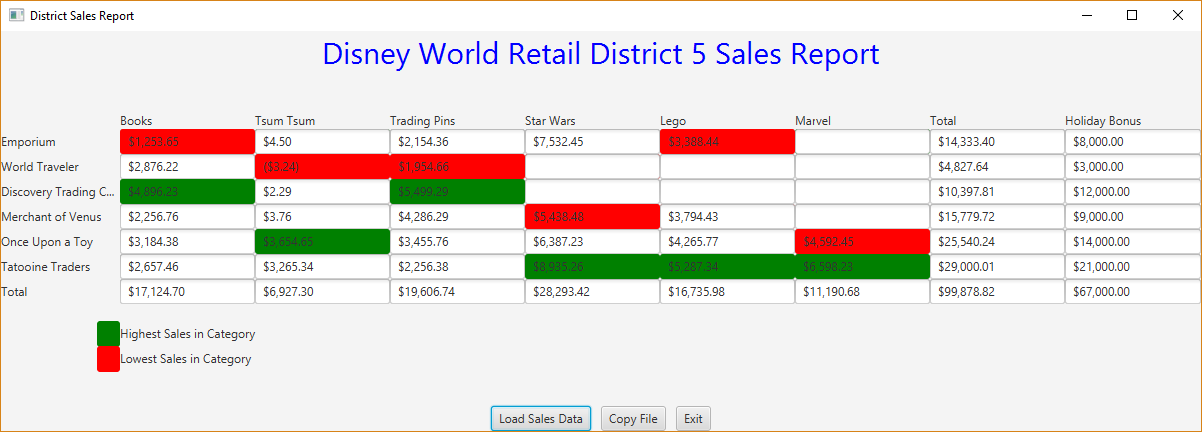
Result after selecting Load Sales Data:



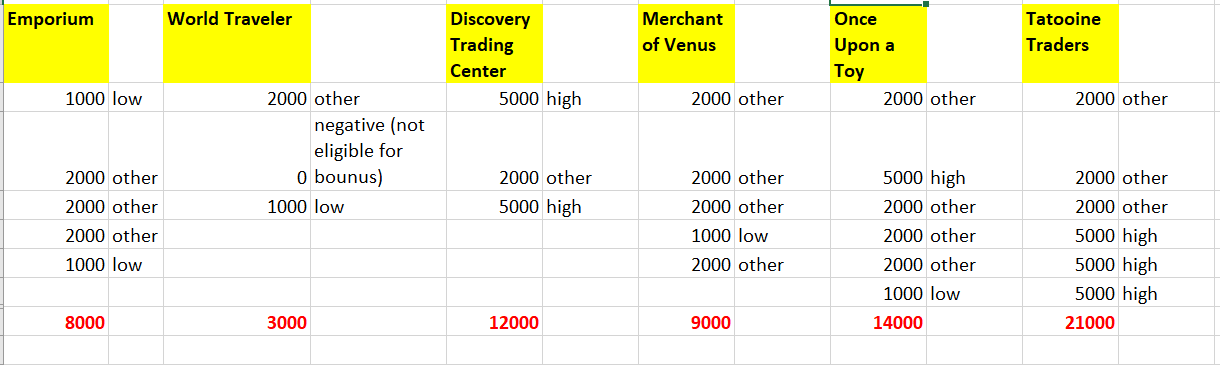
File containing sales data (including negative numbers):



Result after selecting Load Sales Data:



An example of calculating bonus for the above district is shown below:



**Deliverables**

**Deliverables / Submissions and Deliverable format:**

* The Java application must compile and run correctly, otherwise project grade will be zero.
* The detailed grading rubric is provided in the assignment rubric excel file.
* Your source code should contain proper indentation and documentation.
* Documentation within a source code should include
  + additional Comments to clarify a code, if needed
  + class description comments at the top of each program containing the course name, the project number, your name, the date, and platform/compiler that you used to develop the project, for example:

/\*

 \* Class: CMSC203

 \* Instructor:

 \* Description: (Give a brief description for each Class)

 \* Due: MM/DD/YYYY

\* Platform/compiler:

 \* 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.

   Print your Name here: \_\_\_\_\_\_\_\_\_\_

\*/

**Design**

* Turn in a UML class diagram for all classes that you created in a Word document (or .uml file if you use UmlScluptor).

**Implementation**

**Note**: Only submit the files that are created/modified by per requirement. DO NOT submit the files that are already provided for you.

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

* **FirstInitialLastName\_Assignment5\_Complete.zip**, a compressed file in the zip format, with the following:
* src folder*:*
  + HolidayBonus.java
  + TwoDimRaggedArrayUtility.java

**JUnit Test Files:**

* + HolidayBonusTestSudent.java
  + TwoDimRaggedArrayUtilityTestStudent.java
* Word document that includes (use provided template):
  + - 1. Finalized UML Class Diagram for all classes
      2. Screenshots:
         1. Screen snapshots of the GUI with several district data (similar to screenshots in Assignment Description).
         2. Screen shot of src folder files in your GitHub repository
      3. You 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.
      4. If you have added any public methods in addition to the ones listed in the provided Javadoc, you must submit an updated version of your Javadoc.
      5. Lessons Learned: Provide answers to the questions listed below:
         1. Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.
         2. What have you learned?
         3. What did you struggle with?
    - **FirstInitialLastName\_Assignment5\_JavaFiles.zip**, a compressed file containing one or more Java files **(This folder SHOULD NOT contain any folders and it SHOULD contain Java source file only** **that are created/modified by you per requirement.)**
  + HolidayBonus.java
  + TwoDimRaggedArrayUtility.java
  + HolidayBonusTestSudent.java
  + TwoDimRaggedArrayUtilityTestStudent.java