**TCSS 143A**

**AUTUMN 2016**

**Lab 4**

* **Today’s Lab:**
  + File Processing, Arrays – Parallel, 2D
  + Objects and Classes
  + JavaDocs
* **Lab 4a [15 points]: File Processing**

Write a program that accepts the name of the file (DavidCopperfield.txt) to be read from the user and counts the number of words, and lines from the given input file and prints it:

1) To the console and

2) To an output file called ‘Results.txt’.

Make sure your program has at least one method apart from main() method.

Sample Output:

Enter the name of the file: **DavidCopperfield.txt**

Total number of words: **1000**

Total number of line: **95**

*Use the workspace to figure out the logic (Input -> Process -> Output):*

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Lab 4b [15 points]: File Processing**

Write a program that reads a file containing data about the changing popularity of various baby names over time and displays the data about a particular name. Each line of the file stores a name followed by integers representing the name’s popularity in each decade: 1990, 1910, 1920, and so on. The rankings range from 1 (most popular) to 1000 (least popular), or 0 for a name that was less popular than the 1000th name. A sample file called ‘population.txt’ has been given to you. Your program should prompt the user for a name and search the file for that name. If the name is found, the program should display the data about the name on the screen. If the name is not found, then display the message ‘Name not found in the file’

Sample Output:

Name? **Sam**

Statistics on the name “Sam”

1900: 58

1910: 69

1920: 99

1930: 131

…

*Use the workspace to figure out the logic (Input -> Process -> Output):*

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + **Lab 4c [20 points]: Arrays in Parallel**

In this program you will use two arrays in parallel and perform some useful tasks that can be done using arrays.

1. Create an array to hold the ID’s (int) of 10 sales employees. Then create an array to hold the sales amounts for each employee that will parallel the array of IDs.
2. Write a loop to read in (Scanner) data for both arrays. (what type of loop is this?)
3. Write a second loop to display the data from both arrays lined up **with column headings**.

ID Sales

2121 $3456

2235 $5324

1. highestSales(): Method that accepts ID array and Sales array to find which salesperson had the highest sales. Print the employee’s ID and sales amount.
2. lowestSales(): Method that accepts ID array and Sales array to find which salesperson had the lowest sales. Print the employee’s ID and sales amount.
3. averageSales(): Method that takes in the Sales array and returns and prints the average sales record.
4. In main, print the IDs of the salespeople who have below average sales and print the IDs of the salespeople who have above average sales.
5. searchID(): Method accepts the ID array, Sales array and the target ID that needs to be searched (Accept the ID to be searched from the user in Main and pass it as input). Search the ID array for that target ID and then print the ID and the sales for that salesperson. If the ID is not present in the array of IDs print a message stating the search ID was not found.

*Use the workspace to figure out the logic (Input -> Process -> Output):*

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + **Lab 4d [20 points]: Two-dimensional Arrays**

**Magic Square**

Write a program that accepts 9 values from the user in the range of 1 to 9. Store these values in a two-dimensional array of size 3 \* 3. The program should display if the two-dimensional array is a magic square or not. A square is considered to be a magic square if all of its row, column and diagonal sums are equal. Create a method called magicSquare() that accepts the two-dimensional array with integers as input and returns true or false.

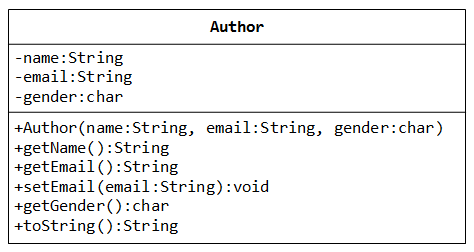
For example:

[[2,7,6], [9,5,1], [4,3,8]] is a magic square because all eight of the sums are exactly 15.

*Use the workspace to figure out the logic (Input -> Process -> Output):*

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + **Lab 4e [20 points]: Creating simple objects and testing it out**

A class called Author is designed as shown in the class diagram. It contains:

* + Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
  + One constructor to initialize the name, email and gender with the given values;

**public Author (String name, String email, char gender) {......}**

(There is no default constructor for Author, as there are no defaults for name, email and gender.)

* + public getters/setters**: getName(), getEmail(), setEmail(),** and **getGender()**;  
    (There are no setters for name and gender, as these attributes cannot be changed.)
  + A **toString()** method that returns "*author-name (gender) at email*", e.g., "Tan Ah Teck (m) at [ahTeck@somewhere.com](mailto:ahTeck@somewhere.com)".

Write the Author class. Also write a *test program* called TestAuthor to test the constructor and public methods. Try changing the email of an author, e.g.

**Author anAuthor = new Author(“Tan Ah Teck”,** [**ahteck@somewhere.com**](mailto:ahteck@somewhere.com)**”, ‘m’);**

**System.out.println(anAuthor); // Call toString()**

**anAuthor.setEmail(“paul@nowhere.com”);**

**System.out.println(anAuthor);**

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **Lab 4f [10 Points]: JavaDocs**

Look at the Documentation Requirement provided with this lab. Read through the documentation and type in the sample code given. Once done typing the code, follow these steps to generate the Java Documentation using jGrasp:

1. File -> Generate Documentation
2. A HTML documentation will be generated. Save it and show it to your TA.

**TA Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_