44-642 Exam 1 - Take Home

Important: Copying and pasting code is *not allowed* (except as noted in the instructions). All code should have comments throughout, unique for each student. Code that is too similar in formatting, spacing, and variable names may be considered plagiarism/academic dishonesty. Bottom line, do your own work and you should be fine.

Create a single NetBeans project titled `YourNameExam1` to contain the problems below, each in a different package as directed.

Problem 1

1. Put all classes for this program in a package named `mapSetList`.

2. Find a full page of plain text on the web and paste it into a text file (make sure you choose your own text and are not using the same source as another student). Add this text file to your project folder. Feel free to clean up the text file as you see fit before you start using it.

3. Write a program that reads in the text file and performs several manipulations (rearranging words and letters, etc.) and calculates different statistics (measure different properties, like the number of certain letters, words that start or end a certain way, etc.). Print a report to the screen (with labels!) of the manipulations you did and the statistics you chose to calculate (include actual samples of your text file in your output as appropriate, to demonstrate what you are doing).

4. It is mostly up to you to choose what manipulations and statistics you demonstrate, but in addition to number 3 above, the following requirements **must be included**:
    a. Your program must make active use of at least one Map, one Set, and one LinkedList, and their related methods.
    b. Perform at least 5 different textual manipulation demos and 5 different statistical demos.
    c. Fully comment your program so someone can look at the code and understand quickly what it is about.
    d. It should be obvious what your program is trying to do for anyone who reads the output.

Problem 2

1. Put all classes for this program in a package named `stackQueueHash`.

2. Create a new class called Player that starts with the following code:

```
public class Player {
    private String name;
    private int id;

    public Player(String name){
        this.name = name;
```

```
        this.id = new Hashifier(name).hash();
    }
```

3. Add getters to the `Player` class.

4. Create another class called `Hashifier` with private instance variable `seed` of type String.
    a. The constructor should take a String parameter and initialize the instance variable.
    b. Add another method called `hash()` that takes no arguments. This method should return a hashcode of your own invention, using the String hashCode() of `seed` somehow.

5. In your driver class, you will be emulating some sort of card game, where different players draw cards from a machine with different values, and they get ranked according to the values they draw. Your output should show a sequence of players drawing cards (each player can draw more than one card if you want), and once every player has drawn, the output should be the final ranked list of players and their scores, according to the rules you created for your game.

6. Like before, you have freedom on the actual card values and the rules of the game, but in addition to number 5 above, certain things are **mandatory**:

    a. You must use an appropriate `queue` to contain your Player objects.
    b. You must create at least 10 Players with different names.
    c. You must use an appropriate `stack` implementation to hold your "cards", which can be represented as Integers, or some other comparable type of your choosing. You should have a sufficient number of "cards" to make it interesting.
    d. When you print your final rankings, all Player information should be displayed, including name and id, along with whatever score you have calculated for each.
    e. Fully comment your program so someone can look at the code and understand quickly what it is about.
    f. It should be obvious what your program is trying to do for anyone who reads the output.

Problem 3

1. Create a package named `heapsBST`.
2. Right-click on the package and add a new Empty File (text file). Name the file `treeEssay`**.**
3. Type your answers to the questions below directly into the text file (one paragraph each). *Be sure the answers are in your own words (do not copy-paste). Use complete sentences and well-structured paragraphs. Please supply the URL's of any websites referenced for each question.*
    a. Question 1: How are binary search trees useful in programming? Give three examples of applications/problems that can be helped by using BSTs.
    b. Question 2: How are heaps useful in programming? Give three examples of applications/problems that can be helped by using heaps.

Submit your solution by following the steps below:

- Save your files in NetBeans.
- Zip your entire Project. (It should be called YourNameExam1 where YourName is your first and last name.)

- Submit the zip file to the **Exam 1 Take Home** drop box.
- Download the zip file which you have submitted.
- Look into the zip file and verify the relevant files are correct. If not, resave your project in NetBeans and resubmit.