JAVA FINAL PROJECT - SUMMER 2018 (VER 1)

MEHDI GARROUSIAN

Please read this document very carefully and if you have any questions, feel free to contact me.

1. Instructions

- Final Deadline: Aug 18.
- Preview Deadline: On Aug 11, you'll be presenting a preliminary version of your code and will have a chance to ask me questions. Your project needs to be over 50% complete at this point.
- You'll be working in groups of 3 people.
- Use the concepts and techniques of object oriented programming to create a text processing application that performs according to the following description. The objective is to have an application that can randomly draw from a text file of inspirational quotations and offer various auxilliary features.
- Upon final submission, one member will email me the entire project with the correct folder structure in a zipped file and include the name of the members.
- **Important:** Each group will write all basic methods and components of the project and choose (or will be assigned) one stared feature to include in the project.

2. Structure

This project draws heavily from Chapter 9 on text processing and is comprised of interacting classes. See Chapter 8 for techniques such as aggregation, inheritance and so on that are going to be instrumental in the creation of this application. Each application consists of at the following three classes:

- Quote
- QuoteFile
- QuoteApp

QuoteApp is the demo class that operates on top of Quote and QuoteFile. In the main method of QuoteApp open the text file 'quote.txt'. This file consists of quotes each followed by its credit and separated by blank lines.

2.1. **QuoteFile.** In QuoteFile, design a constructor that receives the file name and creates an object to be processed as follows. Methods in QuoteFile:

- **countQuotes**: This method returns the count of the quotes by looking at the blank lines that separates them. Most likely, having a while loop would be useful here.
- **addEntry**: This allows the user to enter a new quotation entry which gets written to the bottom of the text file in the same format.
- **retrieveQuote(int num)**: This method retrieves the ith quote from the file. Put a while loop here to make sure that the input num is within the valid range. Most quotes span multiple lines. This method transforms the entire quote entry into one long String and returns it.
- (1*) **quoteSearch(String word)**: Get all quotes that contain a given word. The return type could be an array.
- (2*) **wordStats**: Return the frequency of the words that appear in the file and write them in an array. The first entry is the most frequent word with the count, the second is the second most frequent word with the count, and so on. Use structures that most meaningfully can capture this data.

In QuoteApp, create an object called 'myQuoteFile' and allow it be constructed using the same text file. Under QuoteApp, create a random number generator based on the output of 'countQuotes' and use it to draw a random quote by calling 'retrieveQuote'. Save the resulting quote in a variable called 'myQuote'. At this point, you're ready to pass your string to the methods of the Quote class.

2.2. **Quote.** Create the following methods under Quote:

- **getQuote**: This returns the quote itself.
- **getCredit**: This returns the person that the quote is attributed to.
- **getWordCount**: This gets the ouput of getQuote and counts the number of words in the quote itself. You might use the keyword 'this'.
- **(3*) scramblerGame**: Scramble the words of the quote and let the user put them back in order. If they get it right, let them know. Otherwise, show them the correct quote.
- **(4*) fillBlank**: Remove a random word (other than words such as 'a', 'the', 'to', etc) and let the user play the game of guessing what that word is.
- **(5*) tripleX**: Remove three words from the quote, randomly print them to the console and ask the user to put them in the correct places.
- **(6*) defineWords**: Use a dictionary (a text file) and look up the meaning of the all words of a quote.
- **(7*) saveToFave**: Create a file 'favorites.txt' if it doesn't exist (make sure not to overwrite an existing file if it already exists) and save the quote in the file. Make the necessary adjustments to allow the user to retrive these inputs. Ideally, you want to create as few new features and methods as possible.
- **(8*) byWhoGame**: Show the user a quote and let them guess who it's attributed to. You could give them three random names and let them choose.

2.3. **What else?**

- Make sure your coding is done according to the highest standards that we've discussed in our lectures.
- The members of each group are supposed to keep the other members accountable so thate each member contribues their fair share to the project.
- You are encouraged to exchange ideas with the other groups, however copying code from outside of your group is prohibited and counts as cheating.
- **(9*-Massive Bonus!)** Use the techniques of chapter 12 to craft a GUI (graphical user interface) for your app. You could collaborte with other teams by having them write the methods and you write the top level GUI.
- **(*)** Get creative and come up with one feature of your own that's not listed here.
- **Important:** Do not change the file and folder strcutre unless you have a very good reason for it.
- Also, present a UML for your entire project.