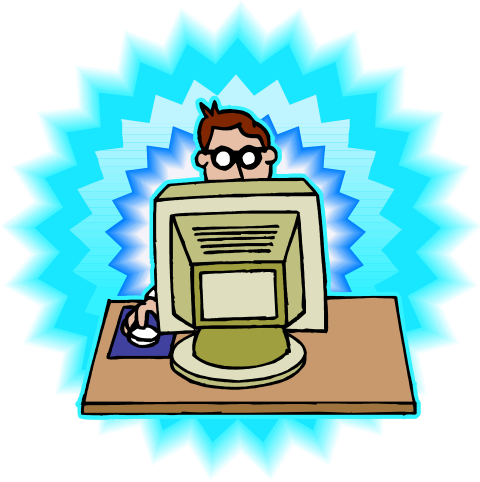
COMP 248

ASSIGNMENT 3



DESCRIPTION: In this final assignment, you will be working with objects and arrays. You will be developing a simple game that allows people to guess English words. The idea is as follows:

The user will be asked to guess 4-letter words. To make this realistic, we must utilize a large list of possible words. For this purpose, I have provided a text file containing 4030 4-letter English words. To make things a little easier, I also provide a short program called StringDatabase.java that reads the file from disk and loads all strings into an array of Strings. Note that you will have to change the file path in this source file to reflect the location of the file on your machine. To simplify things, you should place the file with the 4-letters words in the same folder as your source files.

Your program will randomly select a string and ask the user to guess the string. Initially, the string will be listed as "----". The user will have the choice to try to guess the string directly or to guess a letter at a time. The initial screen might look like this:

```
** The great guessing game **

Current Guess: ----


g = guess, t = tell me, l for a letter, and q to quit
```

If the letter option is chosen ("l"), the user will enter a possible letter and the program will indicate how many such letters there are in this string. It will then replace one or more of the "-" values with this letter and redisplay the result. It might look like this, if 'a' is chosen:

```
l
Enter a letter:
a
You found 1 letters

Current Guess: --a-


g = guess, t = tell me, l for a letter, and q to quit
```

This process continues until the user either gets the answer right or gives up. In both cases, the proper string will be displayed and an appropriate message will be printed to the screen. At this point, a new string will be randomly selected from the database and the game will continue with a new guessing round. Of course, "quit" can be selected at any time.

To make the game more interesting, you must keep score. We will assume that a user can play up to one hundred games (you will get tired of this after 2 or 3 rounds). For each game/word, you must do the following. Each letter in the English language is used with a certain frequency ("e" is the most common letter, "z" the least). The table of frequencies is given on the following page.

The letters that are still blank at the time of a correct guess will be summed together to give a total. Basically, the point value for a given letter is equal to its frequency. Note that it is easiest to guess a word if you first uncover the most common letters. However, this leaves the least points in the remaining word. Of course, the number of letters that you turn over also affects your score. So you should divide the sum by the number of times you request a letter. The more letters you must turn over, the lower your total score. Finally, an incorrect guess costs you 10% of your score.

What happens if you give up? Then you should lose points. Here, the total points lost should simply be the sum the uncovered letters.

The idea is that your score from each round will be recorded, along with the original word, the status of the game (correct guess or not), the number of guesses required, and the final score for that game. Once the user has finished playing, they will select quit. At that point, you will print a short report that summarizes the info. It would look something like this, assuming the user had played two games:

Game	Word	Status	Bad Guesses	Missed Letters	Score
1	yule	Success	0	2	2.02
2	dipt	Gave up	0	4	-17.96
3	laid	Success	2	1	12.20
4	defi	Success	1	1	5.83

Final Score: 2.09

In this assignment, you really only need three classes. The first will be called **Guess** and will represent the game itself (including the menu). The second will be called **Game** and will maintain information about a specific game. These game objects will be stored in an array as required. The third class will be a possibly modified version of the **StringDatabase** class that I have provided. You can add whatever code you think you need here.

Note that you will have to use the Random class to generate the random numbers. In addition, proper Javadocs documentation **must** be provided for the system. In other words, it should be possible to generate a full web-based API for the application.

Letter	Frequency
a	8.17%
b	1.49%
c	2.78%
d	4.25%
e	12.70%
f	2.23%
g	2.02%
h	6.09%
i	6.97%
j	0.15%
k	0.77%
l	4.03%
m	2.41%
n	6.75%
o	7.51%
p	1.93%
q	0.10%
r	5.99%
s	6.33%
t	9.06%
u	2.76%
v	0.98%
w	2.36%
x	0.15%
y	1.97%
z	0.07%

DELIVERABLES: You will submit three sources files and the word list file. No other files should be submitted! The source files will be named **Guess.java**, **Game.java**, and **StringDatabase.java**. Each file will contain one class, with the same name as the file. The **main** method will be found in Guess.java. Once you are finished writing your code, you will combine the three source files into a single zip file called:

A3_name_ID.zip

where "name" is your name and "ID" is your student ID. So if you your name is John Smith and your Student ID is 12345, then you would create a zip file called

A3_John_Smith_12345.zip

The file should be uploaded to the Moodle server via the Assignment 3 link on the web page. You are free to submit multiple versions (new versions just overwrite the last one) until the deadline. You may still submit after the deadline (up to two days late) but the server will record your submission time and an appropriate penalty will be assessed (as per the syllabus). Absolutely NO submission will be accepted by email.

As always, it is your responsibility to verify that your assignment has been submitted correctly.

Good Luck!