CSE12- Fall 2015 Project 1: Warm up.

Due: 29 September, 2015, at 23:59:59

Points: 50 (+5 for Extra Credit)

Project 0 is designed just to make sure you are able to use the UNIX machines at UCSD, survey site and submit your work via Vocareum.

Logistics:
In EACH AND EVERY FILE that you turn in, we need the following in comments at the top of each file. These are essential so that we can more easily process your submissions and insure that you receive proper credit. This is a very large class with about 350 students when combining both lectures.

NAME: <your name>
ID: <your student ID>
LOGIN: <your class login>

Turn in:
See the submission instructions at the end of this document. You will need a Vocareum account for submission, so if you have not received your login information, post on the appropriate post on Piazza.

Getting Started
We strongly recommend that you work on the lab machines, but if you choose to work on your own machine, you can. Just make sure you know what you are doing. We'll be doing all of our testing from the lab machines. Lab accounts should be set up sometime during week 1. Instructions below assume you are using the lab machines.

Create a subdirectory call "HW1" in your class account. All of your files should be placed in that subdirectory. If you cannot remember how to create directories, refer to a unix tutorial

Problem #0 (3 points)

1. First read and sign the Integrity of Scholarship agreement for CSE 12 here: http://quizstar.4teachers.org/index.jsp
    a) You need to sign up first
    b) Please use your name, so that I can use the reports later
    c) Search the class by my last name: "Langlois".

   You cannot earn any credit in CSE 12 until you have done so.

2. For another 2 points, fill out a short pre-test. Important:
    - You need your QuizStar account to take this quiz (see above).
    - You will get your two points for any reasonable attempt at the quiz.

Problem #1 (7 points)

The purpose of this problem is to get your comfortable both on the command line as well as using the Eclipse IDE. For parts A you may use any program you like to edit your java files (e.g., Dr. Java, vim, Notepad++, or even Eclipse), but we would like you to compile the program and generate Javadoc via the command line.

Download following Files from the Class Website and save them to your HW1 directory:

Dictionary.java
Dictionary.pdf

Look at the file Dictionary.pdf. This is a PDF of documentation created using javadoc. Using whatever editor you like (vim, Dr. Java, or even Eclipse), modify Dictionary.java with appropriate javadoc comments, so that it generates similar documentation. Replace the author field with your name.

Next generate the javadocs for this file **via the command line**, and place all of the documentation files in a subdirectory called **doc** in your HW1 directory. If you do not know how to do this, and don't know where to start, try Googling "javadoc command line" (without the quotes). I recommend skipping the StackOverflow link and going to the official Java page. The section on "options" will be particularly useful.

Look at the generated Dictionary.html file to be sure it was generated appropriately, and matches what is in Dictionary.pdf (with your name as the author). When you turn in Dictionary.java, we will run javadoc on your file to create the required documentation.

In addition, place the following information in your HW1-Answers.pdf file

- What command line is used to create the javadoc documentation in HW1/doc?
- What command-line flag(s) is/are used to create the author and version entries for the class?

Problem #2 (40 points)

Use Eclipse to complete this programming assignment. You job is to create a Java program that simulates a hangman game. If you are unfamiliar with the game, here are a few links: link1, link2.

We have provided a tiny bit of starter code in the files Dictionary.java which you can download from the course website. You will write your game Hangman.java in the main method and add more code to Dictionary.java. When the user starts your game, it should play the game of Hangman with the user until the user types 'n'. Here is an example run. User input is shown in blue.

```
How long is the word? 3-6: 4
- - - -
Misses: []
Attempts Left: 6
your guess? o
- - - -
Misses: [o, ]
Attempts Left: 5
your guess? a
- - - -
```

```
Misses: [a, o, ]
Attempts Left: 4
your guess? e
- - - -
Misses: [e, a, o, ]
Attempts Left: 3
your guess? t
- - - -
Misses: [t, e, a, o, ]
Attempts Left: 2
your guess? r
- - r -
Misses: [t, e, a, o, ]
Attempts Left: 2
your guess? l
- - r -
Misses: [l, t, e, a, o, ]
Attempts Left: 1
your guess? u
bird
again? y/n
y
How long is the word? 3-6: 3
- - -
Misses: []
Attempts Left: 6
your guess? a
- - -
Misses: [a, ]
Attempts Left: 5
your guess? o
- o -
Misses: [a, ]
Attempts Left: 5
your guess? d
- o d
Misses: [a, ]
Attempts Left: 5
your guess? r
rod
again? y/n
n
```

Your exact formatting doesn't have to match ours, but the game play and the info shown should match.

Here are some detailed requirements of the game play and specifics about the program:

- You will write your code in the main method, but you should use good style and helper functions as needed.
- **Assume an intelligent player** (for this assignment).

- The game should repeat until the user enters ‘n’
- The game should track the full guess history for the player. It should store the missed guesses history of the user in a LinkedList of Strings. These variables are already set up in the starter code. You just need to use them.
- At the beginning of each game the user is given an option on the length of the word. It varies from 3 to 6.
- The words are stored in a predefined “dictionary”, it is provided. You need to pick a word of the given length at random.
- There are 6 attempts to guess a word.
- Print the following info after each attempt:
  - Phrase (with dashes and correctly guessed letters)
  - List of missed letters
  - How many attempts left

Problem #3 (5 extra points):

- Count the number of played games and report the overall statistics after the player presses ‘n’:

Example:

```
Number of games played: 10
Words of length 3: 3 Games played. Won 2 games, Lost 1 Game
Words of length 4: 1 Game played. Won 1 Game.
Words of length 5: 3 Games played. Lost 3 Games
Words of length 6: 3 Games played. Won 1 Game, Lost 2 Games.

Overall: 40% wins, 60% losses.
```

How to submit your homework
Below are instructions for submitting all of your homeworks for CSE12.
You should have received an email from support@vocareum.com indicating your userid and password for this website. This is our submission site.

1. Follow the link https://labs.vocareum.com/home/login.php and log in.

2. You should be able to see the course info and click on “details”

3. You should be able to see HW1. Click on “Upload Assignment”.

4. You will see a pop-up page. Click on the ”Upload files” on the upper left corner. You can simply drag the files to window (but also READ STEP 5 to create the necessary directories).

5. If you want to add a new directory, you need to create a folder in Vocareum first, and then drag those files that belongs to that directory into the folder. **You should preserve the directory structure you used when testing your homework. Do not upload zipped file.** If you are not sure if the directory structure is correct or not, go to step 7.

6. We are setting the number of submission to be unlimited. So feel free to submit the assignment multiple times.

7. Once you submit the assignment, you should be able to see a new yellow button called “check submission”. We have created a check submission script for you to check if your submission has the correct directories, files and their structures.

8. Once you submit and check, you are done with submitting.

We are going to grade the latest version you submit.

What to Submit:

Dictionary.java
Hangman.java
doc/Dictionary.html
HW1-Answers.pdf