CIS351

Homework 3

Submission Instructions

1. Submit completed **DoubleList.java** through in Blackboard.
2. DO NOT forget to mention your full name in the Java class documentation.
3. submit a filled in **cover sheet** in Blackboard.

Objectives:

- Writing moderately complicated algorithms.
- Using loops of various kinds.
- Constructing arrays.
- It is also intended to reinforce what you have already learned about integer arithmetic and the nesting of if statements.
- Finally, it is intended to demonstrate the value of reusable methods and classes (i.e., to make you appreciate the fact that methods/classes can, if designed and implemented properly, be used for multiple projects).

Download Materials

For this assignment you will implement six methods to support the development of "Word Guess" games like Hangman. For this homework, you need to download the following starter code from Blackboard. The following five files represent a complete command-line hangman program:

- WordGuess.java - create from scratch
- ReferenceTests.javaPreview the document - Code for testing the correctness of WordGuess.java methods
- Dictionary.javaPreview the document - Code for reading words from Words.txt
- Hangman.javaPreview the document - Driver/User-interface

- Sample.txt - Sample output from one the game we are developing
- Words.txtPreview the document - A list of guessable words
- Words.csvPreview the document - This is the same as Words.txt, but this needs to be used with the jar file
- PA4.jar - Execute this jar file to see how a correct implementation of this PA will look like (command to execute jar file: java - jar PA4.jar)

Background

When Hangman.java is executed, a player is presented with a series of blanks representing a hidden word. The player guesses letters; if the letter is in the word, the word is updated to reveal the letter of that guess filled into the correct spots in the word. If the player does not guess a letter in the word, he or she receives a strike. If the player guesses the word (reveals the last letter) in fewer than 6 guesses, they win. If it takes more than 6 guesses, they lose. In this assignment, if a player guesses a letter that they have already guessed, it is not counted as a strike. See the sample output from one such game in Sample.txt.

The last three files are provided *only as an example*, and they will not be used in grading your assignment. You should build and test WordGuess.java incrementally -- as you are writing each method -- rather than trying to test with Hangman.java all at once.

Required Methods for WordGuess.java

The following terms will be used consistently throughout the method descriptions below.

`theWord`
: The current word to guess. It will be preserved throughout play.

`userWord`
: A pattern that represents the progress the user is making on the word.

`guess`
: The letter that the user is guessing.

`strikes`
: The number of bad guesses the user has made.

`guesses`

A list of characters that the user has guessed.

Method Specifications

`public static String makeUserWord(String theWord)`

Takes the word the player is trying to guess and generates a pattern of '*' values to represent letters that have not yet been guessed. For example, if the word were "dog" this method would return "***".

`public static boolean isInWord(char guess, String theWord)`

Returns true if the guessed character is in the word, false otherwise. For example, if the guess were 'x' and the word were "xylophone" this method would return true.

`public static String updateUserWord(char guess, String userWord, String theWord)`

Returns a userWord with all occurrences of '*' corresponding to the current guess replaced with that guess. For example, if the word was "fetch" and the user word was "****h" and the user guessed 'e', the return string would be "*e**h".

`public static String updateGuesses(String guesses, char guess)`

Updates the list of guesses with the current guess. The update should only add the guess if it does not already exist in the list. The new guess must be placed at the end of the existing list of guesses. For example, if guesses were "tabg" and the current guess were 'f', this method would return "tabgf".

`public static String displayUserWord(String userWord)`

Returns a String that is the userWord with spaces between each letter and each '*' replaced with an '_'. For example, if the userWord is "fe***" this method would return "f e _ _ _". Note that there is no space before the 'f' and after the last '_'.

`public static String displayGuesses(int strikes, String guesses)`

Returns a String in the form "Strikes: %d\tGuesses: %s", with the list of guesses separated by spaces. For example, if there were 3 strikes and guesses were "bcd", this method would return"Strikes: 3\tGuesses: b c d".

Grading Criteria

Total points: 100 points

Correctly compiles and executes without any failure - 15 pts
Instructor provided correctness testcases - 70 pt
Documentation and styling - 15 pt

Don't forget
the submission process or
the grading criteria or
the cover sheet.

Farzana Rahman / frahman@syr.edu