CS-200-1: Programming I
Fall 2017
Northeastern Illinois University
Programming Project: Hangman
Due: Thursday, 12/26 at 5:00 p.m.

Background:
Hangman is a very popular word game with two-players. First one being a "chooser", who comes up with a secret word and the second one being "guesser" who attempts to guess this word one letter at a time. In our implementation, the computer will be a "chooser" who secretly chooses the word from word-list that will be provided to you. The rules are as follows:

- For this game, you are also provided with `words.txt` and `Hangman.java` files. **Be sure to save them in the same directory.** Open and run `Hangman.java` without making any modifications.
- Inside the program, you are provided with two helper codes: `loadwords()` and `stringToChar()`.
- `loadwords()` loads an array of words from a `words.txt`file. As you can see the return type of the method is a String array, if you call the methods properly you will get String array of word-list.
- `stringToChar()` takes a String, which in this case is the secretWord that your computer generates, and converts the String to a char array.
- You don't have to understand the codes inside these methods, but depending upon the requirement of the game, you have to know when and where to call these methods.
- If everything is working correctly, after calling the `loadWords()`, you should see the following printed out:

```
Loading words from the file......
55909 words loaded.
--
---------
```

- The computer will secretly choose a word at random from String array and show the player how many letters are in the word by displaying a sequence of blanks (underscores).
- The game must be interactive. You should let the user know the length of the secret word.
- You should then ask the user to input one guess(i.e letter) per round.
- The feedback should be given immediately with each guess if their guess is in the secret word or not.
- After each round, you should also display to the user the partially guessed word so far, as well as letters that the user has not guessed yet.
- If the player guesses all the letters in the word, they win.

- The game ends in one of the two ways
  - If the player has guessed the complete secret word, they win.
  - Otherwise, if the player runs out of guesses.
- In either case, the computer should stop asking for questions.
- You should inform the player whether they won or lost, and reveal the entire secret word.
- Assume that all the words are lowercase.

Initial Instructions:

- You should work in groups of 2-3 individuals. Groups of more than 3 are **not** permitted.
- Each group should submit ONE project write-up. It is the responsibility of each group member to ensure that their name is on the write-up.
- The lab write-up should be typed! Type each question (and the question number) followed by your group's answer. **Convert your lab write-up to a .pdf.**
- You should use complete sentences and proper grammar in your write-up. Use spell-check! This counts as part of your grade.
- Your code should be your own - no plagiarism is permitted! You should have at least 3 methods in your code, not including the `main` method.
- Submit the pdf and your .java files to D2L by the specified due date.
- Each member of the group must turn in a `readable` digital copy of the peer assessment to an individual Dropbox by the assigned due date and time. The peer assessment counts as a significant part of your grade and you will receive a **zero** for that portion of the research lab grade if you do not turn it in.

Guidelines
You should adhere to the following guidelines when designing your program.

1. A user is allowed 8 guesses. Make sure to remind the user of how many guesses s/he has left after each round.
2. You should prompt the player for input with the following prompt:
   `"You have 8 guesses left"`
   `"Please guess a letter:  "`
3. A user loses a guess only when s/he guesses incorrectly.
4. If the user guesses the same letter twice, do not take away a guess - instead, print a message as:
   `"Oops!  You've already guessed that letter"` displaying the word to let them know that they've already guessed that letter and prompt them to try again.

5. If the user enters a correct letter then print out the message:
   `"Good guess"`

6. If the user enters an incorrect letter then print out:
   `"Oops!  That letter is not in my word"`

7. The word should be printed out following each guess to show the user(s) the remaining letters to guess.

8. If the user guesses all the letters of the secret word then print out the message:
   `"Congratulations, you won!"`

9. On the other hand, if the user runs out of guesses, then display:
   `"Sorry, you ran out of guesses.The word was secretWord."`

10. Your output should be easy to read and You will be graded on usability of the program.

11. You should thoroughly test all aspects of the program. You will be graded on the correctness of your program.

Project write-up questions
Answer the following questions in your lab write-up. Make sure to include each question in the write-up followed by your group's answer.

Q1: Create a flow chart - a graphical representation of the sequence of steps needed to implement the Hangman algorithm. For additional information and details on flow charts, see the following sites:
`http://www.computerhope.com/jargon/f/flowchar.htm`
`http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl_flow.htm`

Q2: Describe how you stored the user entries for the letters.

Q3: What are the methods that your group created in your code? Describe each method in detail and why you chose to create each particular method.

Q4: What was the most challenging part of this project for your group?

Q5: What did your group learn/find the most useful by doing this project?

Q6: What was the most fun aspect of doing this project?