CS-200-1: Programming I
Fall 2017
Northeastern Illinois University
Programming Project: Hangman
Due: Thursday, 12/07 at 5:00 p.m.

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 lose points if the write up is not in a .pdf.
- You should use complete sentences and proper grammar in your write-up. Use spell-check! This counts as part of your grade.
- 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.
- Each group should also submit two group selfie of the meet-ups done other than in classroom.

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

- For this game, you are provided with two files: `words.txt` and `Hangman.java`. **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 methods: `loadwords()` and `stringToChar()`.
- The method `loadwords()` loads an array of words from a `words.txt` file. If you call the method correctly, you will get String array from the list of words.
- The method `stringToChar()` takes a String, which is the secret word that your computer generates, and converts the String to a char array.
- You don't have to understand these methods, but you have to know when and where to call them.

- If everything is working correctly, you should see the following printed out as you call the method `loadWords()`.

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

- You have to choose a word at random from the list of words and display the secret word as a sequence of blanks (underscores).
- The game must be interactive. You should let the user know the length of the word and then ask to input one guess(i.e letter) per round.
- A user is allowed 5 guesses. Make sure to remind how many guesses left after each round.
- You should give the feedback immediately with each guess and also display the partially guessed word, as well as letters not yet guessed.
- 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, they lose.
- In either case, you should inform the player whether they won or lost, and reveal the entire word.
- Assume that all the words are in lowercase.
- 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 and the helper methods `loadWords()` and `stringToChar()`.

Guidelines

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

1. You should prompt the player for input with the following message:
   `"You have 5 guesses left"`
   `"Please guess a letter:  "`
2. A user loses a guess only when they guess incorrectly.
3. If they guess the same letter twice, do not take away a guess - instead, print 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 again.
4. If the user enters a correct letter then print out the message:
   `"Good guess"`
5. If the user enters an incorrect one then print out:
   `"Oops!That letter is not in my word"`

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

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

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

9. The winning game should look like this:

```
Loading word list from file...
55900 words loaded.
Welcome to the game, Hangman!
I am thinking of a word that is 4 letters long.
-------------
You have 5 guesses left.
Please guess a letter:  a
Good guess:  _ a _ _
-------------
You have 5 guesses left.
Please guess a letter:  a
Oops!  You've already guessed that letter:  _ a _ _
-------------
You have 5 guesses left.
Please guess a letter:  s
Oops!  That letter is not in my word:  _ a _ _
-------------
You have 4 guesses left.
Please guess a letter:  t
Good guess:  t a _ t
-------------
You have 4 guesses left.
Please guess a letter:  r
Oops!  That letter is not in my word:  t a _ t
-------------
You have 3 guesses left.
Please guess a letter:  m
Oops!  That letter is not in my word:  t a _ t
-------------
You have 2 guesses left.
Please guess a letter:  c
Good guess:  t a c t
-------------
Congratulations, you won!
```

10. Similarly, the losing game should look like this:

```
Loading word list from file...
55900 words loaded.
Welcome to the game Hangman!
I am thinking of a word that is 4 letters long
-------------
You have 5 guesses left
Please guess a letter:  a
Oops!  That letter is not in my word:  _ _ _ _
-------------
You have 4 guesses left
Please guess a letter:  b
Oops!  That letter is not in my word:  _ _ _ _
-------------
You have 3 guesses left
Please guess a letter:  c
Oops!  That letter is not in my word:  _ _ _ _
-------------
You have 2 guesses left
Please guess a letter:  d
Oops!  That letter is not in my word:  _ _ _ _
-------------
You have 1 guesses left
Please guess a letter:  e
Good guess:  e _ _ e
-------------
You have 1 guesses left
Please guess a letter:  f
Oops!  That letter is not in my word:  e _ _ e
-------------
Sorry, you ran out of guesses.  The word was e l s e.
```

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

12. It is very important, when programming, to consider the usability of your program. If users find your program difficult to understand or operate, they won't use it!

13. When inserting underscores into your word, it is a good idea to add at least a space after each one, so it's clear to the user how many unguessed letters are left in the string **(compare the readability of _____ with _ _ _ _ _)**.

14. 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. Flow chart needs to be inside the lab-write up and not **hand-written**. 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?