



UNIVERSITY OF MARYLAND

DEPARTMENT OF COMPUTER SCIENCE

CMSC131 - OBJECT-ORIENTED PROGRAMMING I

SECTION 010X, 030X, AND 040X

DR. ROGER EASTMAN AND DR. ILCHUL YOON

Project 3

Hangman in Eclipse

Assigned: Mar. 22th, 2018

Due: Apr. 2nd, 2018 11:00 PM

Late Due: Apr. 3rd, 2018 11:00 PM — 20% penalty

Good Faith Attempt Due: Apr. 9th, 2018 11:00 PM

Drop Option: You can drop this project. Note that you can only drop one throughout the semester.

1 Overview

For this project you will implement the classic game of Hangman in Java. The main objective of this project is to reinforce your understanding of the String.

Use Eclipse to work on this project. Check out first the *Hangman* project from the CVS repository and then write code in the provided methods. You may add more methods if you need, but you are not allowed to modify the provided method signatures in the starter file. The methods will be called from the *main* method.

2 Requirements

The flow of the game is given below. You must implement the features described in the flow and must also consider the constraints for each step in the flow in your implementation.

1. At start, your program will first call the *chooseWord* method that selects a word from a text file and returns the word selected (i.e., a String is returned from the method). This method is already implemented in the starter code. You must NOT modify the method but instead will call it from the *main* method or from a method you create. The *chooseWord* method takes two parameters. When you call the method, use *f* and *rand* (already initialized in the *main* method) as the parameters.

The word selected from the *chooseWord* method should be validated by calling the *isValidWord* method, which returns, *false* if the word does not satisfy any of the following constraints:

- No alphabet occurs three or more times (e.g., *sadness* is invalid).
- At least 6 alphabets in the word (e.g., *enjoy* is invalid).
- Only alphabets in the word (e.g., *bona fide* is invalid).

The selection-validation process should be repeated until a word satisfying the constraints is selected.

2. After a secret word is selected, your program will display a prompt message for a game mode. The player will enter either 'i' (or 'I') for the *case insensitive* mode or 's' (or 'S') for the *case sensitive* mode. Then the player will type the 'enter' key to start to game. The following shows an example interaction with the player in the console. In the example, the player chooses the case insensitive mode. (We will use this mode in the running example.) If the player enters any other key (e.g., 'k'), repeat displaying the prompt again and receive the user input.

```
Choose the game mode i/I for case insensitive mode, s/S for case sensitive mode): i
You selected case insensitive mode.
```

3. The program will then display a prompt message to receive a guess from the player. It will display the selected word in console by a row of underscore characters, where each underscore character represents each letter of the word (say this *hidden word*). Now, the program is ready to interact with the player. For example, if the word is *practice* (7 unique alphabets),

```
_____ 10 Remaining guesses
Enter a letter or string:
```

4. If the guessing player enters a letter or a string, followed by an enter key,
 - (a) If the letter or string occurs in the word, your program shows it in all its correct positions. In specific, the program prints one more line that reveals the correct letter(s) or string(s). Set the initial number of guesses to 3 more than the number of unique alphabets in the selected word. For example, if the word is *practice* (7 unique alphabets) and the player first enters *ice*, the following message should be printed:

```
____ice 10 Remaining guesses
Enter a letter or string:
```

In this example, the number of guesses left is unchanged because *ice* occurs in the word. When you write code that prints the messages, put 3 space characters between the word and the remaining guess message, and put a space after ':' in the second line. Note that this example assumes the case-insensitive mode. Even though the player enters *ICE*, the output message should be identical. The second line will only be displayed if the word is not fully revealed after the guess.

- (b) If the suggested letter or string does not occur in the word, the number of allowed guesses will be decreased by 1. In the example above, if the player first enters *k*, the following message should be printed:

```
k does not occur in the word.
----- 9 Remaining guesses
Enter a letter or string:
```

5. Repeat the step 3 and 4 until all alphabets of the word is fully revealed or until the number of allowed guesses is decreased to 0.
6. If all guesses are used up and the word is still not fully revealed, display the following message. Again, the word we use in the example is *practice*.

```
You lost! (The word was practice)
```

7. If there are 0 or more remaining guesses and if the word is fully revealed, display a message.

```
Great! You saved the man!
```

8. After the player uses up all guesses or the word is revealed, your program asks if the player wants to play again. If the answer is yes, the game begins again. Otherwise, the game ends. Use the following messages:

```
Do you want to play again? (y/n):
```

The player will enter 'y' (or 'Y') for another game, or 'n' (or 'N') to end the game. Again, put a space after '.' in the message. If the player wants to play another game, repeat the flow from the step 1. If the player enters any other key (e.g., 'k'), repeat displaying the prompt again and receive the user input.

```
Invalid command. Do you want to play again? (y/n):
```

3 Grading and Good Faith Attempt Policy

Your submission will be evaluated based on the public, release, and secret tests in the submit server. All public tests must be passed to meet the GFA requirements for this project.

4 Example

An example output from a fully implemented project is given below for your reference.

```
Choose the game mode i/I for case insensitive mode, s/S for case sensitive mode):
```

```
i
You selected case insensitive mode.
----- 8 remaining guesses.
Enter a letter or string: t
__t___ 8 remaining guesses.
Enter a letter or string: i
__ti__ 8 remaining guesses.
Enter a letter or string: o
o_tio_ 8 remaining guesses.
Enter a letter or string: p
optio_ 8 remaining guesses.
Enter a letter or string: n
Great! You saved the man!(The word wasoption)
Do you want to play again? (y/n):
n
```

Another execution example output is given below.

Choose the game mode i/I for case insensitive mode, s/S for case sensitive mode):

```
s
You selected case sensitive mode.
----- 8 remaining guesses.
Enter a letter or string: 0
0 does not occur in the word.
----- 7 remaining guesses.
Enter a letter or string: p
_p---- 7 remaining guesses.
Enter a letter or string: n
_p___n 7 remaining guesses.
Enter a letter or string: I
I does not occur in the word.
_p___n 6 remaining guesses.
Enter a letter or string: K
K does not occur in the word.
_p___n 5 remaining guesses.
Enter a letter or string: 0
0 does not occur in the word.
_p___n 4 remaining guesses.
Enter a letter or string: o
op__on 4 remaining guesses.
Enter a letter or string: T
T does not occur in the word.
op__on 3 remaining guesses.
Enter a letter or string: t
opt_on 3 remaining guesses.
Enter a letter or string: L
L does not occur in the word.
opt_on 2 remaining guesses.
Enter a letter or string:
E
E does not occur in the word.
opt_on 1 remaining guesses.
Enter a letter or string: e
e does not occur in the word.
You lost! (The word was option)
Do you want to play again? (y/n):
n
```

5 Requirements on readability and coding style

Follow a good coding style (e.g., <http://www.cs.umd.edu/~nelson/classes/resources/javastyleguide/>, sections 14, 16, and 18 can be ignored). Your code should be properly indented (use the AutoFormat feature in Eclipse), and well documented with appropriate comments. It is mandatory to use meaningful variable names, rather than “magic numbers” (literal constants). There are no explicit requirements for how many and what type of variables. For readability including variable naming, we suggest you to use these rules:

- Format the code clearly with blank lines, spacing and indents
- Use self-documenting, meaningful variable names

- Use variable names that begin with lower case and then camel case (e.g., roomTemp) if two words
- Use capital letters (e.g., `ALL_CAPS`) for variables that encodes constant values that aren't updated
- Use appropriate comments that indicate what a section, or a complicated line, does

As you have done for other projects, first few lines of your code should be the comments with the project, date, and author (you) information. You are **required** to write your name, directory ID, university ID, and section number as a comment at the top of each file you submit from this project forward. Additionally, we expect you to write the honor pledge (***I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.***) as a comment near the top of each file you submit.

6 Submission

Submit your work to P3 in the submit server by the due date/time (Check the submit server). It is strongly recommended to submit directly using the 'Submit Project' menu in Eclipse. After submission, check in the submit server that your work is submitted correctly and also that your code passed all tests. Note that you will need to pass all release/secret tests to get the full credits.

7 Extra point (20 points) – Hangman in Processing

What you have developed in this project is a text based game. For extra credits, create a separate, visual version of Hangman in Processing. The Processing version cannot be claimed as a substitute submission of the main project (in Java, using Eclipse) implementation expected. For submission, compress all files needed to run your code into a zip file and submit the compressed file to **P4Extra** in the submit server. The best submission will be given a separate prize announced after the project deadline.

8 Academic Integrity

Make sure you read the academic integrity section of the syllabus so you understand what you must not do. Note that we check your submission against other students' submissions, and that we are required to report academic dishonesty cases to the University's Office of Student Conduct. As stated in section 2, you are **required** to write the honor pledge as a comment near the top of each file you submit.