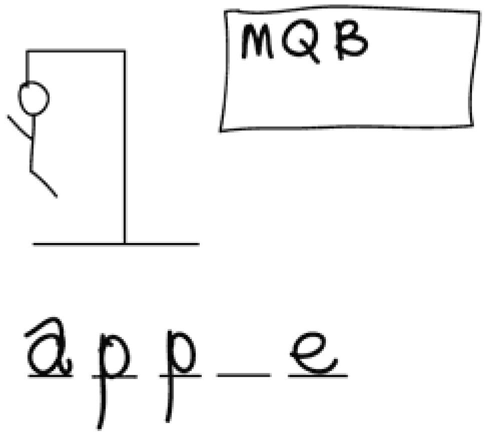
Assignment 05

CS 120: Software Design I

Program 1: `Hangman`

Hangman[1] is a game where one player thinks of a word, and the other person tries to figure out what word it is. The guesser knows how many letters the word is; if they guess a letter that is in the word, they are shown all the spots where that letter occurs. Below, you can see a partially-completed game of hangman:



For your program, the computer will choose a word, and you will write the code for the user to guess that word, playing the game one or more times.

This document is split into two parts. **Part 1** discusses the extra credit portion of this assignment[2]. **Part 2** discusses the particulars of the implementation. Note that Part 1 is only first to reflect it's earlier due date. However, completion of Part 1 will depend on a careful reading of Part 2. Even if you opt not to submit Part 1 for credit, it is recommended to work through it, as I will likely ask you many of the same questions should you come to office hours.

[1]Read more: `https://en.wikipedia.org/wiki/Hangman_(game)`

[2]Where the extra credit will be applied (e.g., assignments) will be determined shortly. However, it can't hurt to start this assignment too early!

Part 1: Plan of Attack

Although this game seems relatively simple at first glance, there are multiple uses of conditionals and loops (often nested) that can make debugging difficult. For a point of comparison, I count 11 loops in my implementation of the program, and at one point, I have loops nested four levels deep, with a conditional inside that innermost loop[3]. Thus, to help facilitate a smooth implementation of this program, I **highly recommend** doing significant planning before opening up Eclipse. You have the option of turning in this work for extra credit by **October 25 at 11 PM**; late work will not be accepted for this part of the assignment. You may turn in any or all subparts of Part 1. Depending on the number of submissions I get, I will strive to give feedback to people over the weekend before the assignment is due.

For Part 1, create a folder called `cs120-assign05-<lastName>-part1`. You will put all documents for this part in this folder.

Considering Subproblems (1 extra credit point)
Begin by considering subproblems you might first solve in attempting to consider your solution. Identify three subproblems. In a document file (e.g., Google Docs, Microsoft Word), name and describe your three subproblems in a numbered list. Subproblems should be ordered in the order you would consider them, and your descriptions should identify why you have ordered the subproblems in this way. **Export your document as a PDF called `1-subproblems.pdf` and put it in the folder described earlier in the section.**

Considering Cases (2 extra credit points)
Looking at the example runs shown in Part 2, identify what can happen each time the user enters a guess during any given game. Also consider what data structures you might have for tracking information pertinent to a guess; how will those data structures change in each one of the cases you identify? Note that I am interested only in arrays and not variables. In a document file (e.g., Google Docs, Microsoft Word), describe the data structures you envision maintaining to track information pertinent to a guess. Then identify **all** possible outcomes from the user entering a guess in a numbered list, and identify how the data structures you described earlier will change for each possible scenario. **Export your document as a PDF called `2-cases.pdf` and put it in the folder described earlier in the section.**

Activity Diagram (5 extra credit points)
Before implementing your program, it would be beneficial to sketch out a high-level solution. For this part, you will be developing an activity diagram of the program you will later implement. In particular, I am interested in where and how you will be using loops to control gameplay (e.g., why do we have this loop?). Activities and decisions can be shown in less detail. For example, you might have an activity that says something like “sorting characters”, or “checking to see if the user has correctly guessed the word” without showing the need to check each individual character, or making a decision using a conditional. On the other hand, each loop should be annotated on (1) what it is used for (e.g., controls whether or not to play another game, and (2) what its boolean condition is (this can be in English, e.g., runs if user wants to play another game).

It is likely that your diagram will require a few iterations. I would recommend sketching it out in a format that is easily modifiable and allows room for additions, such as on a whiteboard. Your final version should be easily readable. I envision this happening in one of two ways: (1) using a computer program (e.g., Power Point, Microsoft Word) to draw the shapes for the diagram, or (2) copying down your final solution by hand to paper (perhaps a few pieces of paper) and scanning

[3] I do not expect you to match the number of loops I have and whatnot, this is only to demonstrate that this is a complex problem!

it in. Poorly drawn or difficult to read diagrams will not be graded. **Regardless of how you digitize your diagram, export your document as a PDF called `3-activityDiagram.pdf` and put it in the folder described earlier in the section.**

Part 2: The Program

A basic outline of hangman was given previously. In your version of hangman, the user (henceforth referred to as "the player") will play one or more games of hangman with the computer, with the computer responsible for picking a word, displaying the game board, and responding to guesses.

The computer will start by picking a word, and displaying a number of blanks (represented with asterisks) to the player equal to the length of the chosen word. The player will then issue guesses, one guess at a time. The player cannot guess a letter they have previously guessed (regardless of whether or not that letter was in the word), but they do have unlimited guesses (i.e., they cannot lose a game by taking too many guesses). You can assume that players will **only** guess letters, but that those letters could be in upper or lower case (i.e., your program should interpret both 'A' and 'a' as the same letter). After each guess, the computer will display the updated board. When the player has successfully guessed all the letters in the word, they have won the game. The computer will congratulate the player, display how many guesses it took (note that guesses of previously-guessed letter should not count), and display all the letters guessed **in sorted order**. The computer will then ask if the player would like to play again. If they answer yes, the computer will pick a new word, and a new round of the game will start. If they answer no, the computer will thank the player and display the number of games they won (which should be equal to all the games played, as the player cannot quit partway through a game).

To start, create a new class in your project called `Hangman`. Then, go to D2L and download the `Hangman.java` file. Open up the file in a plain text editor (e.g., Notepad), and copy all the contents of the file into the `Hangman` class you created in Eclipse; **note that what you copy over should replace everything that Eclipse auto-generated for you**. This starting code provides you with a way to obtain random words for each round of the game. The only line pertinent to you is line 7; this line sets `char[] word` to an array of `char` values representing the random word given to you. You can continue to fill in `main` with your code, as you would any other program. You might find it useful to comment out the `char[] word` line while working on your program, and replacing it with a word of your choosing, so that you can work with the same input every time. See below:

```
// char[] word = randomWord();
char[] word = {'h', 'o', 'r', 's', 'e'};
```

However, when you submit your program, please be sure your declaration and use of word is the line originally given to you in the program. While you do not need to keep this line of code on line 7, omitting or modifying this line in your program will prevent Autolab from being able to test your code.

Below are several runs of my program, with input `Highlighted`. Make sure that your output matches the examples below (except for the sections determined by user input). Note that there are 30 dashes between each guess (you can hardcode these). Also note that the last two examples span more than one page. Finally, note that the guesses displayed in the last example only span two lines due to the limitations of the page width; your program should display these all on one line.

Example 1

```
Starting board: ***

Enter your guess: e

------------------------------

Your current board: e**

Enter your guess: a

------------------------------

Your current board: ea*

Enter your guess: t

------------------------------

Your current board: eat

Congratulations! You win!
You took 3 guesses!
You guessed the following letters: a e t

Would you like to play again? [ y / n ] n

------------------------------

Thank you for playing! You won 1 game!
```

Example 2

```
Starting board: ***

Enter your guess: P

------------------------------

Your current board: p**

Enter your guess: E

------------------------------

Your current board: pe*

Enter your guess: T

------------------------------

Your current board: pet

Congratulations! You win!
You took 3 guesses!
You guessed the following letters: e p t

Would you like to play again? [ y / n ] N

------------------------------

Thank you for playing! You won 1 game!
```

Example 3

```
Starting board: ***

Enter your guess: y

------------------------------

Your current board: y**

Enter your guess: Z

------------------------------

Your current board: y**

Enter your guess: z

Sorry, you already guessed this character, please pick again.

------------------------------

Your current board: y**

Enter your guess: a

------------------------------

Your current board: y**

Enter your guess: o

------------------------------

Your current board: yo*

Enter your guess: l

------------------------------

Your current board: yo*

Enter your guess: j

------------------------------

Your current board: yo*

Enter your guess: u

------------------------------

Your current board: you
```

```
Congratulations! You win!
You took 7 guesses!
You guessed the following letters: a j l o u y z

Would you like to play again? [ y / n ] n

------------------------------

Thank you for playing! You won 1 game!
```

Example 4

```
Starting board: ****

Enter your guess: a

------------------------------

Your current board: ****

Enter your guess: e

------------------------------

Your current board: *ee*

Enter your guess: t

------------------------------

Your current board: *ee*

Enter your guess: n

------------------------------

Your current board: *ee*

Enter your guess: k

------------------------------

Your current board: *ee*

Enter your guess: d

------------------------------

Your current board: dee*

Enter your guess: p

------------------------------

Your current board: deep

Congratulations! You win!
You took 7 guesses!
You guessed the following letters: a d e k n p t

Would you like to play again? [ y / n ] Y

------------------------------
```

```
Starting board: *******

Enter your guess: a

------------------------------

Your current board: **a****

Enter your guess: e

------------------------------

Your current board: *ea****

Enter your guess: y

------------------------------

Your current board: *ea***y

Enter your guess: h

------------------------------

Your current board: hea**hy

Enter your guess: d

------------------------------

Your current board: hea**hy

Enter your guess: p

------------------------------

Your current board: hea**hy

Enter your guess: l

------------------------------

Your current board: heal*hy

Enter your guess: t

------------------------------

Your current board: healthy

Congratulations! You win!
You took 8 guesses!
You guessed the following letters: a d e h l p t y
```

```
Would you like to play again? [ y / n ] y

------------------------------

Starting board: ********

Enter your guess: a

------------------------------

Your current board: ********

Enter your guess: b

------------------------------

Your current board: ********

Enter your guess: c

------------------------------

Your current board: c*******

Enter your guess: d

------------------------------

Your current board: c*******

Enter your guess: e

------------------------------

Your current board: c**e*e**

Enter your guess: f

------------------------------

Your current board: c**e*e**

Enter your guess: g

------------------------------

Your current board: c**e*e**

Enter your guess: h

------------------------------

Your current board: c*he*e**
```

```
Enter your guess: i

------------------------------

Your current board: c*he*e**

Enter your guess: j

------------------------------

Your current board: c*he*e**

Enter your guess: k

------------------------------

Your current board: c*he*e**

Enter your guess: l

------------------------------

Your current board: c*he*e**

Enter your guess: m

------------------------------

Your current board: c*he*e**

Enter your guess: n

------------------------------

Your current board: c*he*en*

Enter your guess: o

------------------------------

Your current board: cohe*en*

Enter your guess: p

------------------------------

Your current board: cohe*en*

Enter your guess: q

------------------------------

Your current board: cohe*en*
```

```
Enter your guess: r

------------------------------

Your current board: coheren*

Enter your guess: s

------------------------------

Your current board: coheren*

Enter your guess: u

------------------------------

Your current board: coheren*

Enter your guess: v

------------------------------

Your current board: coheren*

Enter your guess: w

------------------------------

Your current board: coheren*

Enter your guess: x

------------------------------

Your current board: coheren*

Enter your guess: y

------------------------------

Your current board: coheren*

Enter your guess: z

------------------------------

Your current board: coheren*

Enter your guess: t

------------------------------

Your current board: coherent
```

```
Congratulations! You win!
You took 26 guesses!
You guessed the following letters: a b c d e f g h i j k l m n o p q r s t u v
    w x y z

Would you like to play again? [ y / n ] n

------------------------------

Thank you for playing! You won 3 games!
```

The following is a high level checklist of requirements for your program:

- ☐ Your class is named `Hangman` and is placed in the project created for this assignment
- ☐ Your code is commented according to the guidelines in the Java Style Guide found on D2L
- ☐ Your code is formatted according to the guidelines in the Java Style Guide found on D2L
- ☐ Your code fulfills the functionality outlined above
- ☐ You have the `char[] word = randomWord();` line in your program
- ☐ You have not modified any of the other code given to you.