Assignment 5: Reading and Programming Project due 10/9

Reading Assignment

Read Sections 3.2, 3.3 and 3.4 on Loops and using boolean variables to control loops completing self-test exercises.

Programming Assignment

This assignment consists of one project, Hangman1, that involves programming with loops and static methods.

Programming Project

**Hangman1:** Play the game of Hangman - part 1 due **11:00 p.m. on Thursday, 10/9**
worth 14 points

The game. Hangman is a two-person game in which one person (person A) picks a word that the other person (person B) has to guess. Person A originally reveals only the first and the last letters, and the total number of letters in the word. At each turn of the game person B suggests a letter and person A responds by identifying

1. whether the letter occurs in the word, and

2. if an occurrence of the letter is found, all positions at which the letter occurs in the word, not including the word’s first (number 0) and last positions. The game stops when either person B guesses the entire word, and then B is the winner, or when B has picked 7 letters not found in the word. In the latter case the winner is person A.

The program that you write will assume the role of player A. For this week’s assignment you will complete only a part of the game. You will complete the game in the next week’s assignment.

Picking a word. For this project I will provide a class HangmanDict that contains a static method pickWord(). When called, this will return a randomly selected word. You should use this method in the final version of your program to imitate the program “picking” a random word. You will have to download HangmanDict.java from the course website and place it into this project’s directory, so that the compiler can find the method.

While implementing your algorithm you should know the picked word. It is also a good idea to use the same word on the early stages of development to simplify the process of tracing and debugging your code. In the very beginning, I recommend you “hardwire” the word into the code, e.g.

```java
String word = "temperature";
```

and use different guessing scenarios to test your code. Once you get your program to work on just one word, modify your code so that the word is obtained by random selection using the pickWord() method. Again,
while you’re testing, print the word immediately after it has been picked, so that you are aware of what it is and can test the program properly.

When you develop the final version, remove the part that prints out the word so that it is not revealed to the person playing against your program.

For this week’s assignment you have to implement a main method and two other methods the functionality of which is described below. When working on your program, make sure you follow the instructions on what parameters should be passed to these methods, what the return value is, and the content of all printed messages.

- (3 points) Implement method main that picks a random word using method HangmanDict.pickWord(), prints out a message showing the first and the last letters (see examples in the very end of this document) and then uses method hangmanTemplate() described below to create and display a template to the user, showing the first and last letters of the picked word and a dash for every other letter. For example the template for the word temperature is T - - - - - - - - - E (note the spaces around and between the dashes).

Then, the main method should ask the user to enter a single letter (lower- or uppercase) and, using method countLetter() described later in this document, print how many times the letter the user entered occurs in the picked word. The program should stop at this point.

- (4 points) Implement method hangmanTemplate() that accepts one string parameter representing the word, and returns a template showing the first and last letters of the picked word and a dash for every other letter. For example the template for the word program is P - - - - - M. The easiest way to create a template is to create a sequence of dashes using a loop, and then concatenate the sequence with the first and last letters of the word, converted to uppercase.

- (4 points) Implement method countLetter() that accepts a word (parameter of type String) and a character (parameter of type char). It should print out a message about the occurrences of the character in the word, not taking into account the first and the last letters. For instance, when passed the word temperature and character 'E', it prints

  Correct. Letter E occurs in this word at
  position 1
  position 4

When passed the same word and letter 'Q', it should print

  Incorrect. Letter Q does not occur in this word.

This method should return the number of occurrences of the character in the word, not counting the first and the last positions. For instance, in the first example, the return value should be 2, while in the second it should be 0.

I recommend you start developing your program by implementing and testing this method and then using it in the overall algorithm.

A simple way to implement this method is to go through each position in the word, comparing the character at that position to the letter that has been passed.

Take a look at the following example that demonstrates how the program should work, assuming the word picked was loop:
I have picked a word that starts with letter L and ends with P. 
Here’s the template in which each dash denotes a single letter: 
L - - P
Please enter a letter:  A
Incorrect guess. Letter A does not occur in this word.

Here is another example using the same word:

I have picked a word that starts with letter L and ends with P. 
Here’s the template in which each dash denotes a single letter: 
L - - P
Please enter a letter:  O
Correct. Letter O occurs in this word at
position 1
position 2