Programming Assignment 5

Getting Started

This assignment is designed as a practice on programming with loops, strings and arrays. The code you will be writing is longer and more sophisticated than in previous assignments, so be sure to get started as soon as possible and allow ample time for testing and debugging.

Programming Project

Hangman: Complete the Hangman game **worth 15 points**

Please note that the program that you write should be compiled together with HangmanDict.java, which is posted n the course website. Download HangmanDict.java and place it in the same Eclipse project with the program you are composing. You may need to refresh the project folder for the file to appear in Eclipse: select the folder and press the F5 function key.

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 total number of letters in the word by displaying a template where each letter is replaced with a dash character. At each turn of the game person B suggests a letter and person A responds by showing where in the template the letter appears.

The game stops when either person B guesses the entire word, and then B is the winner, or when B has picked 6 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. The user will be player B.

Here are a couple of sample interactions describing the required functionality. In the first one, the user wins:

```
I have picked a word.
Here's the template in which each dash denotes a single letter.
- - - - - - - - - -
There were 0 wrong guesses so far.  Please enter a letter:  t
Correct.
- - - - - - T - - -
There were 0 wrong guesses so far.  Please enter a letter:  y
Correct.
- - - - - - T - - Y
There were 0 wrong guesses so far.  Please enter a letter:  j
Incorrect.  There were 1 wrong guesses so far.  Please enter a letter:  a
Correct.
- A - - - A T - - Y
There were 1 wrong guesses so far.  Please enter a letter:  o
Correct.
- A - O - A T O - Y
There were 1 wrong guesses so far.  Please enter a letter:  l
Correct.
```

```
L A - O - A T O - Y
There were 1 wrong guesses so far.  Please enter a letter:  g
Incorrect.  There were 2 wrong guesses so far.  Please enter a letter:  b
Correct.
L A B O - A T O - Y
There were 2 wrong guesses so far.  Please enter a letter:  r
Correct.
L A B O R A T O R Y
You won!
```

In the next one the user loses.

```
I have picked a word.
Here's the template in which each dash denotes a single letter.
- - - - - -
There were 0 wrong guesses so far.  Please enter a letter:  a
Incorrect.
There were 1 wrong guesses so far.  Please enter a letter:  e
Correct.
- E - - - -
There were 1 wrong guesses so far.  Please enter a letter:  o
Incorrect.
There were 2 wrong guesses so far.  Please enter a letter:  i
Correct.
- E - I - -
There were 2 wrong guesses so far.  Please enter a letter:  u
Incorrect.
There were 3 wrong guesses so far.  Please enter a letter:  f
Incorrect.
There were 4 wrong guesses so far.  Please enter a letter:  k
Incorrect.
There were 5 wrong guesses so far.  Please enter a letter:  z
Incorrect.
You lost - the word is BEHIND
```

Picking a word. As described earlier, I will provide a class HangmanDict that contains method HangmanDict.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. The following code shows how to call the method and assign the returned value to variable *word*

```
String word = HangmanDict.pickWord();
```

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.

```
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.

Requirements. To get full credit, your program should:

1. Allow user to enter letters in either upper or lower case.
2. Check and report whether a guessed character occurs in the word, until the user either makes 6 incorrect guesses, or wins by correctly guessing all letters of the word.
3. Display the number of incorrect guesses as shown in the interactions.
4. Print the template in upper case, with extra spaces between letters/dashes.
5. Declare the win/loss correctly.

Notes and hints. This section contains a few suggestions on implementing the Hangman game.

- Use a String variable to store all letters that the user entered so far. When a new letter is entered, append the letter to that string.
- Use a char array to store the characters of the template. The length of the template should equal the length of the picked word.
- An easy way of detecting that the user has won is by verifying that the template string does not contain any dashes.
- You can use a boolean variable that designates whether the user has won.

Here's a sketch of the algorithm you could use:

```
Pick a word and create a corersponding template.
While the user has not won and has made fewer than 6 bad guesses:
{
     Get a character from user; update the String which stores all letters user has entered
     so far to include the latest letter.

     If the character occurs in this word:
        report occurrences, update and show the template as follows:
             The template can be updated by going through every character of the
             word, and checking if that character has been named by user.
             You can tell if the user has named that  character by searching
             the string of all letters user entered.

      Else, i.e. bad guess
         report that the guess was wrong,
         increment the counter of bad guesses.

     If the template string has no dashes, then the user won!
}
Report winner.
Print out the word.
```

For **extra credit** (1 point) write your program so that it does not count repeated guesses, for example if the user suggests letter Q more than once, the program should not count the second and any later attempt and should print

```
This letter was used earlier.
```