Assignment 9: Evil Hangman

Due: Dec. 15, 2013 at 11:55PM.

In this assignment you will write a program that implements a version of Hangman game. In a fair Hangman game, one player chooses a secret word and writes out a number of dashes equal to the word length. The other player guesses letters. Whenever the guess is correct, the player with the secret word replaces the corresponding dashes with the correct letter. The game ends when either when all the letters have been discovered or when the guesser runs out of the available guesses.

In the Evil Hangman version, the computer is the player with the secret word. The computer plays unfairly because it does not commit to a specific word. After each guess the pool of words that can match the currently guessed pattern decreases, but the computer tries to keep that pool of available words as large as possible. For example, if the current pattern is D O - B L E, the pool of matching words is: ”doable” and ”double”. If the player guesses ”A” or ”U”, the computer marks it as wrong, and removes one word from the pool of words.

The only difference between the fair and evil games is keeping track of the pool of words that match the current pattern.

Evil Hangman Algorithm

In the first step, the computer needs to commit to the length of the word. Your program should generate a random value between 4 and 12 and make sure that there is at least one word of such length in the dictionary.

Assume that the selected length is 4 all of the 4-letter words in the dictionary are:

ALLY  BETA  COOL  DEAL  ELSE  FLEW  GOOD  HOPE  IBEX  SEAL

The player should see

----

pattern on the screen.

Now, suppose that the player selects the letter ’E’. There are more than one word with the letter ’E’ in the current pool of possible words. They can be divided into several groups matching different patterns:

---- matching words ALLY, COOL, and GOOD
-E-- matching words BETA, DEAL, and SEAL
--E- matching words FLEW and IBEX
E--E matching word ELSE
---E matching word HOPE

The computer uses approach of selecting the pattern that matches the largest number of words; in this case it can be either ---- or -E--.

SCENARIO 1: The computer selects ---- that leaves three words in the pool of possible words.

The player should see a message that ’E’ is not in the secret word and the

----

pattern should be redisplayed. From the point of view of the computer, the pool of possible words decreased from ten to three.

Suppose that the second guess is ’0’. The remaining words can be divided into two groups matching two different patterns:
----- matching word ALLY  
-00- matching words COOL and GOOD

The computer selects the pattern matching more words (which might not be the best choice in this case) and displays the next pattern as

-00-

to the player. The pool of possible choices consists of two words. As the player selects more letters the pool gets smaller and more letters appear in the pattern.

**SCENARIO 2:** The computer selects -E-- that leaves three words in the pool of possible words.

The player should see

-E--

pattern on the screen. From the point of view of the computer, the pool of possible words decreased from ten to three. Suppose that the second guess is 'A'. The remaining words can be divided into two groups matching two different patterns:

-E-A matching word BETA  
-EA- matching words DEAL and SEAL

The computer selects the pattern matching more words and displays the next pattern as

-EA-

to the player. The pool of possible choices consists of two words. As the player selects more letters the pool gets smaller and more letters appear in the pattern.

**Programming Tasks**

**Task 1:** Implement fair hangman game

Start by writing the program that implements a fair game, i.e., a game in which the secret word is chosen at the very beginning. This should be relatively easy task and it will allow you to make sure that all of the parts of the program are working before you move on to the harder parts.

**Task 2:** Implement the evil hangman algorithm

This is modification of the code from Task 1 so that instead of selecting the secret word at the beginning the computer cheats and does not commit to a specific word.

**Restrictions:**
You should not be using any of the Map interfaces/implementations provided in Java libraries.

**Task 3:** Implement the graphical user interface

Write the Processing program that provides the graphical user interface for the program that you wrote in Task 2.
Grading

The program that provides the solution to Task 1 is worth 70/100 points. Do not hand in this program if you successfully implement the code for Task 2.

The program that provides the solution to Task 2 is worth 100/100 points. Do not hand in this program if you successfully implement the code for Task 3.

The program that provides the solution to Task 3 is worth 130/100 points.

What and how to submit?

You should submit this assignment using NYU Classes. You should submit three Java source code files for all classes that you write. You should submit solution to only one of the tasks listed above (the highest number task).

The files should be combined into a single zip file called LAST_Assignment9.zip in which you replace LAST with your last name.

Do not submit files with .class extension or any other files generated by Eclipse or other IDE. If you submit the assignment past the deadline, the NYU Classes will not let you upload the file. In this case, email it to Joanna at joannakl@cs.nyu.edu.