**Hangman**

Hangman is a guessing game. The computer randomly select a word to guess from an ArrayList of words. The user will try to guess it by suggesting letters within a certain number of guesses.

For our game, the user will get seven guesses.

I have provided you with a list of possible words to guess (dictionary.txt). This is only so you can have a file that has a lot of words. It is mainly for your amusement. **I should be able to pass you any file, and it should run just fine.** Do not hard code the file name! For testing purposes, I recommend you use a smaller test file that has 3-5 words in it that have different lengths, so you will know what word you get by looking at the blanks. Your life will be made easier by doing this.

The driver of your program must be called **Game.java**.

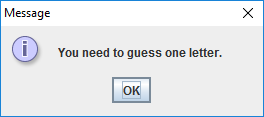
To generate the word to guess, your program will have to read a list of words in from a file. **The file name must be passed as the first command-line argument**.

For example, if dictionary.txt was in the same folder as Game.java, I would run the program by typing:

>java Game dictionary.txt

Your program needs to store each of the words in an ArrayList. You will then need to randomly select one of the words to be the word for your user to guess.

You will display the correct number of letter blanks for the randomly chosen word to the user. You will allow the user to guess a letter (meaning you need a text field and a button to submit the guess). You should only accept **one letter** from your user, and spaces should be removed. You should **not allow characters that are not letters**. If the user does not enter one letter and only one letter, inform them, but don’t count it as a wrong guess.

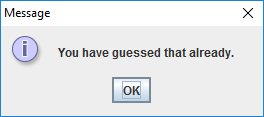


Your game **should** **not be case sensitive**. An uppercase letter guess is equivalent to a lowercase letter guess. If they guessed a letter correctly, you will fill in the correct blank with that letter. If the letter appears more than once in the letter, fill in all of the blanks for that letter. For example, if the word to guess is “scoop”, and the user guesses the letter “o”, you would fill in both o’s for that turn. If they guessed a letter incorrectly, then **draw** the correct piece of the hanged man according to the table below:

|  |  |
| --- | --- |
| **Wrong guess** | **Action** |
| First | Draw gallows |
| Second | Draw head |
| Third | Draw body |
| Fourth | Draw arm on left of body |
| Fifth | Draw arm on right of body |
| Sixth | Draw leg on left of body |
| Seventh | Draw leg on right of body |

These things should be drawn! Do not just display an image. You need to override a paintComponent() method of a JComponent class.

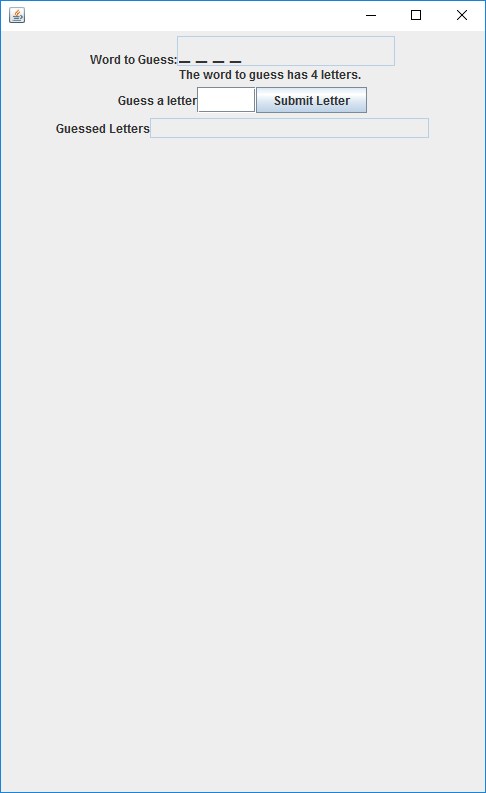
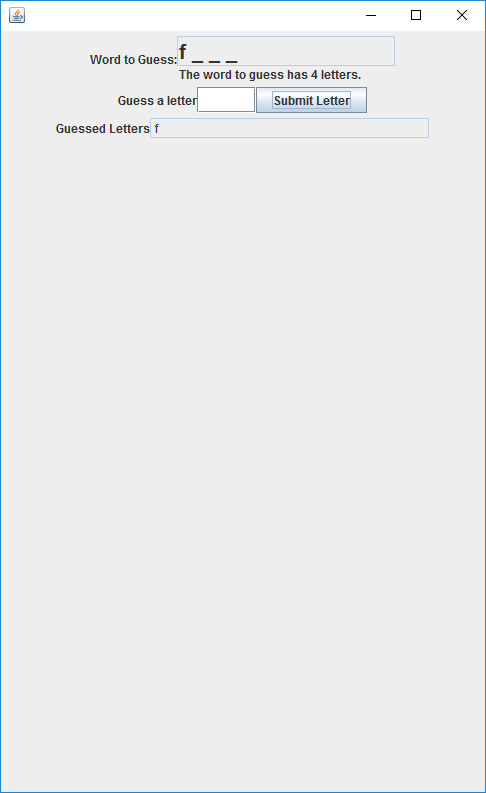
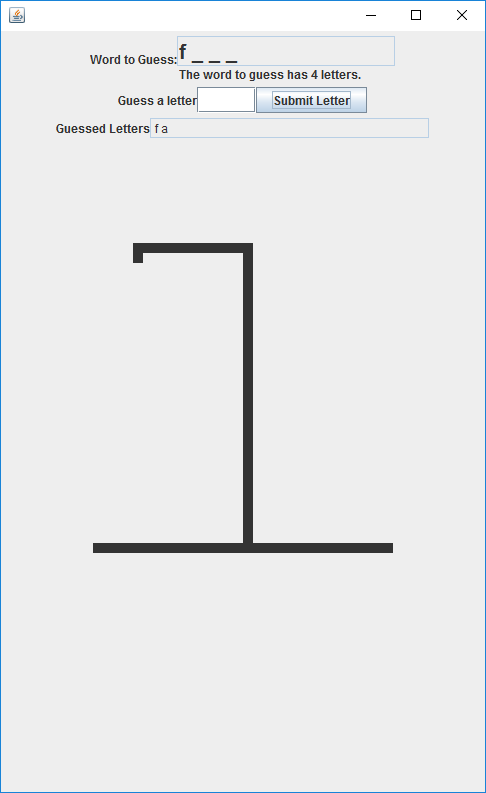
**Show the user the past letter guesses.** If they guess a letter twice, they should not be penalized. Inform the user that they guessed that letter already:

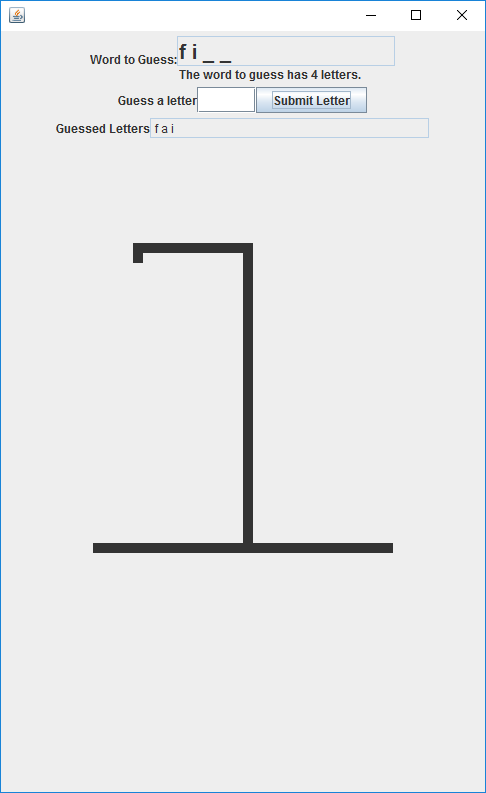
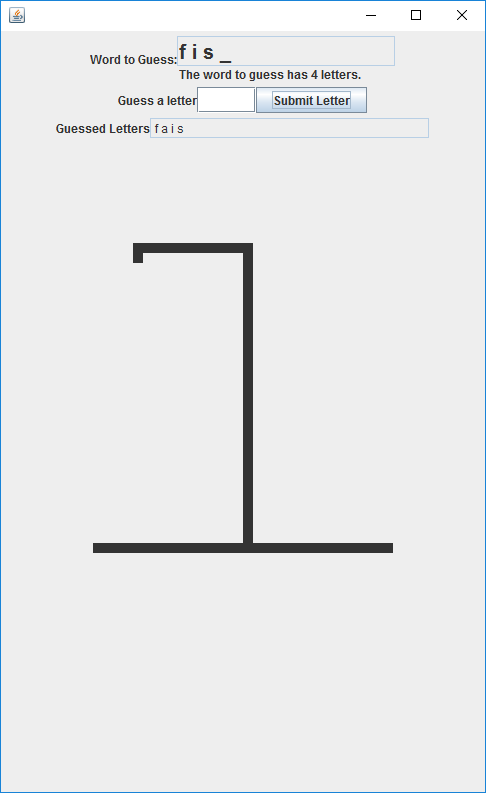
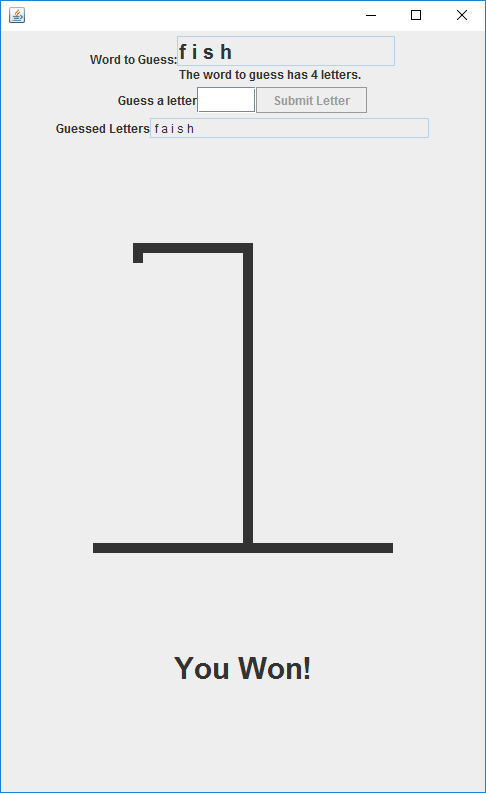


If the user wins, report it to the user. If the user loses, report it to the user and tell them the word they were supposed to guess. Prevent any more letters from being submitted after the game ends via a win or a loss (disable the submit button).

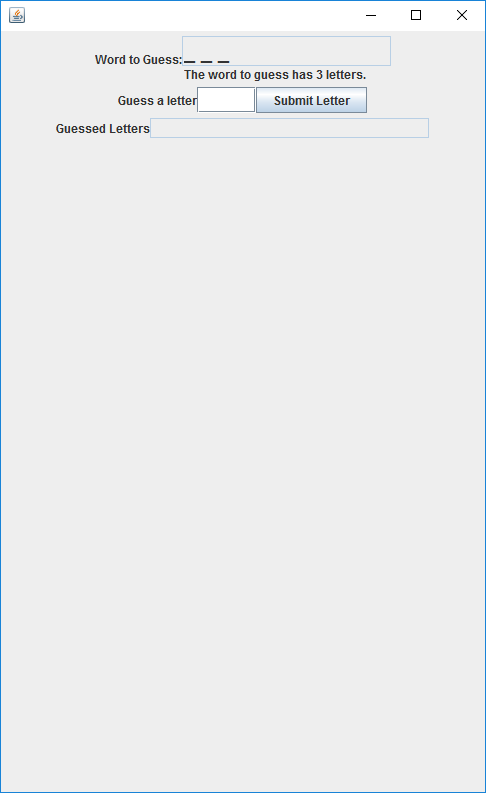
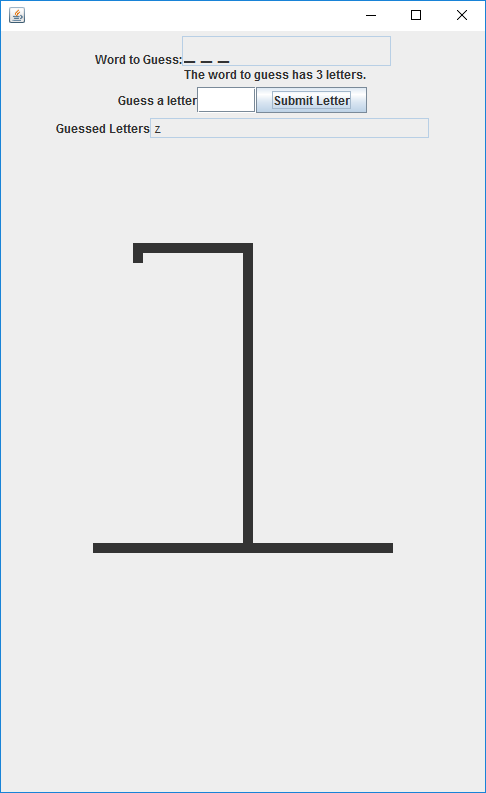
Below are screen shots of a game being won and another game being lost.

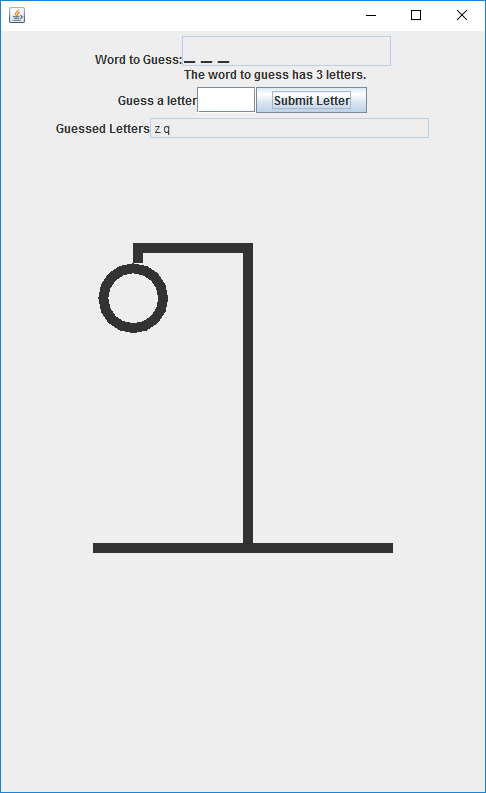
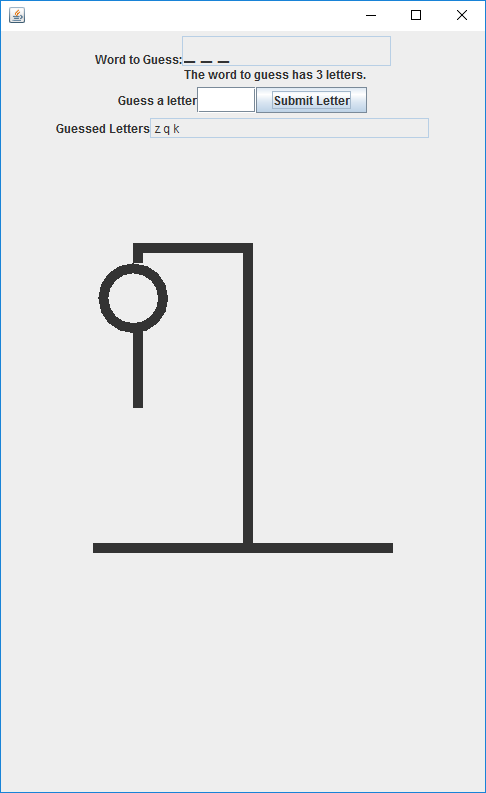
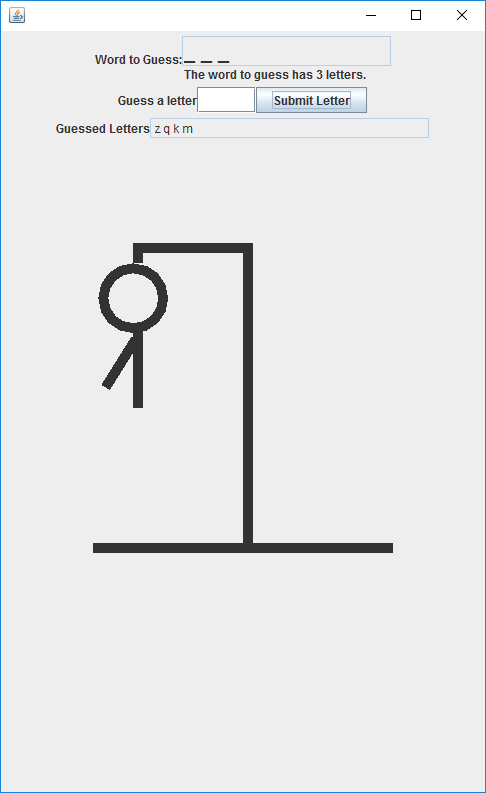
Winning game:

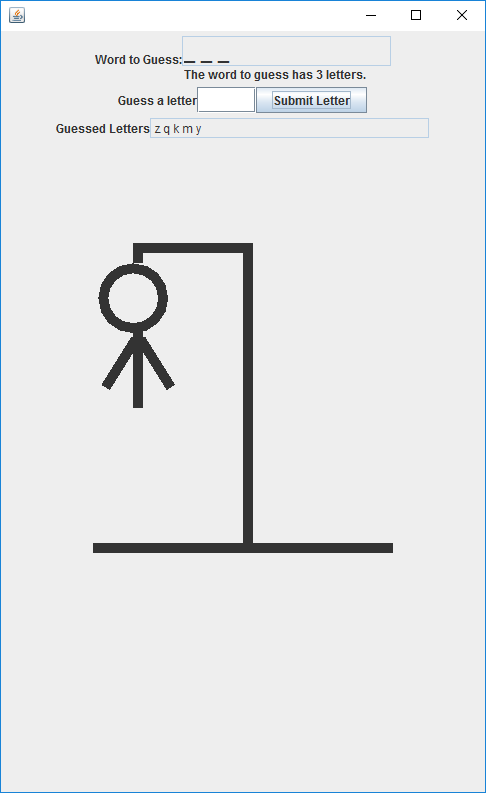
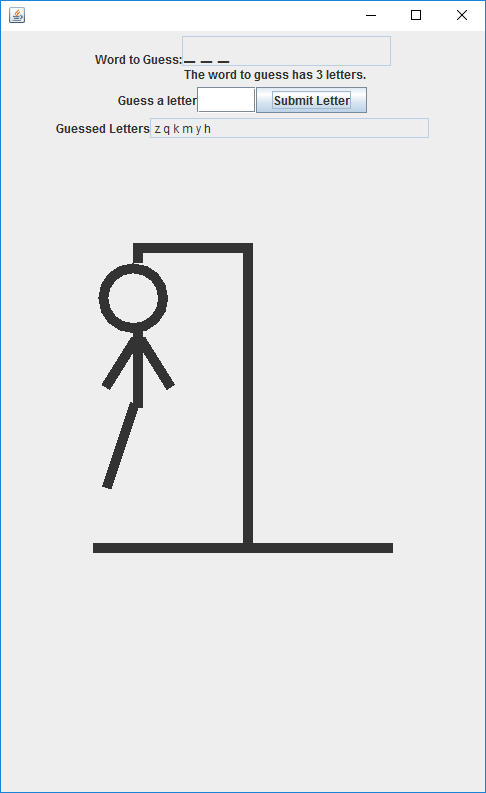
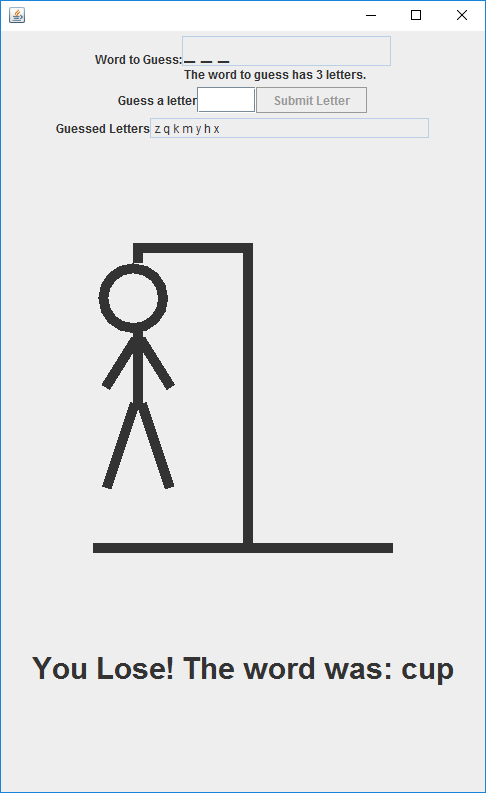
  

Losing Game:

Feel free to embellish your hangman figure and other aspects of the GUI. I will be grading on the ability to display the word to guess correctly (filling in the blanks correctly as well), the ability to get a letter from the user (one letter only, and not penalizing if already guessed the letter before), drawing the hangman figure, and correctly notifying the user of a win or loss.

Turn in your .java files zipped and your readme. Do not turn in your test files or the dictionary.