Final Project: Hangman!

Objective

- Apply knowledge acquired in class, including:
  - Handling user input and output
  - Drawing in Java
  - Manipulating Strings
  - Using arrays
  - Decomposing a problem
  - Designing a Java application
  - Commenting code

What to hand in:

Submit the following to the dropbox no later than 11:59pm the night of March 18th.

- NetBeans project
  - Compress into a .zip or .rar
- Readme.txt
  - This file should describe:
    - How to compile and run the program
    - How to play the game

Note: Late submissions will be penalized at a rate of 20% per weekday.

Marking Scheme:

Your solution will be assessed based on:

- Following project specifications (see below for detailed specifications)
- Commenting
  - Javadoc method comments
  - logic and variables commented
  - name, date and file descriptions
- Format and Conventions
  - appropriately named fields and methods
  - appropriate visibility of fields and methods
  - appropriate use of constants
  - decomposition into meaningful methods
  - decomposition into appropriate classes and objects
- Compiles and runs without errors
- Matches given output (creativity is allowed, but general output requirements should be met)
- Correctly handling user input
  - Handling invalid input gracefully as needed

Specifications:

You are tasked with designing and implementing a Hangman game in Java (see https://en.wikipedia.org/wiki/Hangman_%28game%29 for detailed rules and examples). Your game should work as follows:

- Upon launch, output the rules of the game to the user (can be in the Java console)
- In the Java console, ask the user if they wish to play or quit
  - If play is selected, start game.
  - If quit is selected, output a message to the console indicating they have quit

    - If neither is selected, give the user an error and have them enter play or quit
- Select a word at random from the game's internal dictionary
    - The dictionary can be an array of Strings. There should be at least 10 different words in the dictionary. The words should be composed solely of lower case letters, a-z.
- Launch a Frame with a canvas that will display 3 elements:
    - The Current State of the word they're guessing
        - Starts as _ _ _ _ _ with exactly as many blanks as there are letters in the word
    - The Gallows
        - This is where your program will draw the hangman as needed
    - The letters missed so far
        - As users select letters that aren't in the word, list them
- In the console, ask the user to enter a letter
    - You can assume the user will enter a valid letter following by the "Enter" key
    - The user can try a letter more than once. You can decide if they should be penalized or not
    - If the letter is in the word, update the Current State
    - If the letter is not in the word, add a part to the hangman, and update missed letters
- Continue with previous step until end of game
- The game ends when either:
    - The hangman runs out of parts (Game is lost)
    - The word is discovered correctly (Game is won)
- At the end of the game, let the user know if they won or lost by outputting a message to the console

Hangman:

- The hangman should have a head, 2 arms, a body and 2 legs. As such, the user loses when they make 6 misses

Bonus, one of:

- (1 mark) Let the user play as many games as they wish by prompting them after winning/losing if they'd like to play again. Keep score.
- (1 mark) Implement *an option* to play a 2-player version, where player 1 enters a word in the console (may assume it contains only valid characters a-z), and player 2 tries to guess the word. For this version the game **must** still be playable as 1-player with internal word dictionary.

Tips:

- Start thinking about this early!
- Test early, and test often
- Think about your design carefully, and try to make use of methods to decompose the problem
- You may need more than one Class
- When the panel needs to be refreshed (something has changed) you can alert the panel by calling its "repaint()" method. All code in the paint(Graphics g) method will get run. You should not call paint directly!
- In Java, you cannot change a letter within a String. If I want to change "cat" to "cot", for example, I must make a new string, and copy the c to it, add the o to it, then copy the t to it. An alternate approach is to store an array of chars instead of a string. So, if I had the array: String myWord = {'c', 'a', 't'}, I could change the 'a' to a 'o' by using array access and assignment. myWord[1] = 'o' would result in myWord now containing: {'c', 'o', 't'}.

Sample gameplay:

For sample 1, the game is not played because the user selects quit. It is up to you whether or not the panel will display in this case;

Sample Console Output 1 :
```
Welcome to hangman!
Rules of the game:
Your job is to try to find out the secret word, by guessing one letter at
a time.  But beware!  If you incorrectly select a letter that is not in
the secret word, you will be one step closer to hanging on the gallows!
The secret word is made up of letters a-z.
Do you want to play? (Enter 'q' to quit, 'p' to play)

q

K, thanks bye!
```

For sample output2, the panel should display (as described in specifications, above) before the user is prompted to enter a letter, so they can see the length of the word. In this case it would be: _ _ _. After the first selection (e) the Current State would be _ e _, and the gallows would be empty. After the second selection (r) the Current State would remain unchanged, the gallows would have a head added, and the letter 'r' is added to the missed letters. After the third selection of (f) the Current State would be: f e _, and the gallows would remain unchanged with just a head. After the fourth selection (z) the Current State would be: f e z, the gallows would remain unchanged with just a head, and the game would be over. Player wins!

Sample Console Output 2:
```
Welcome to hangman!
Rules of the game:
Your job is to try to find out the secret word, by guessing one letter at
a time.  But beware!  If you incorrectly select a letter that is not in
the secret word, you will be one step closer to hanging on the gallows!
The secret word is made up of letters a-z.
Do you want to play? (Enter 'q' to quit, 'p' to play)

p

Let's Play!

Pick a letter (a-z):
e
Good choice! Pick a letter (a-z):
r
Too bad, try again.  Pick a letter (a-z):
f
Good choice! Pick a letter (a-z):
z
Good choice!

You win!
```