# HANGMAN 2016

## MAJOR PROJECT – COMPUTER SCIENCE AP

The purpose of this project is to incorporate all of the programming skills and tools you have learned this year. The assignment is to write a game similar to HANGMAN, in which the player tries to guess a randomly selected word with less than six wrong guesses.

Hangman2016 will be implemented as an Object Oriented Program. Six classes will be involved in the implementation of the game.

- **Hangman2016** – The main program (includes a main method).
- **RandomWords** – A class that generates random words.
- **HMButtonPanel** – A specialized JPanel class.
- **HMCanvas** – A Specialized JPanel class that will be used as a drawing canvas.

GRADING CRITERIA (use these criteria to pre-grade your project before submitting the final version):

- Data file of 10 commonly used words (not too hard) that are loaded into the program into an array of strings.
- Program Sequence
  - One word is selected at random from an array of words to play the game.
  - Stars ( * ) are placed in a text field ( **JTextField sentence** ) equal to the number of letters in the sentence not counting spaces.
  - 26 dashes ( - ) are placed in a text field ( **JTextField alphabet** ) corresponding to the letters of the alphabet.
  - Player clicks on a Button corresponding to the letter he wishes to select.
  - If the letter is in the sentence, all occurrences of the letter are placed in the appropriate location in the **sentence** text field.
  - If the letter is not in the sentence, the head of the man is drawn (or whatever body part is appropriate for that guess).
  - All letters that have been used are inserted into the **alphabet** text field in their appropriate location.
  - Clicking on a button corresponding to a letter that has already been used will result in no action. In other words, it will be ignored.
  - Game ends when either the player has guessed the entire sentence, or when the number of wrong guesses equals six.
  - If the player wins, show a congratulatory message in the **sentence** text field.
  - If the player loses, tell the player he lost and show an encouraging message to do better next time in the **sentence** text field. Show the complete sentence in the **alphabet** text field.
  - If the player selects ***New Game*** from the menu begin a new game.
  - If the player selects ***Exit*** from the menu terminate the game.

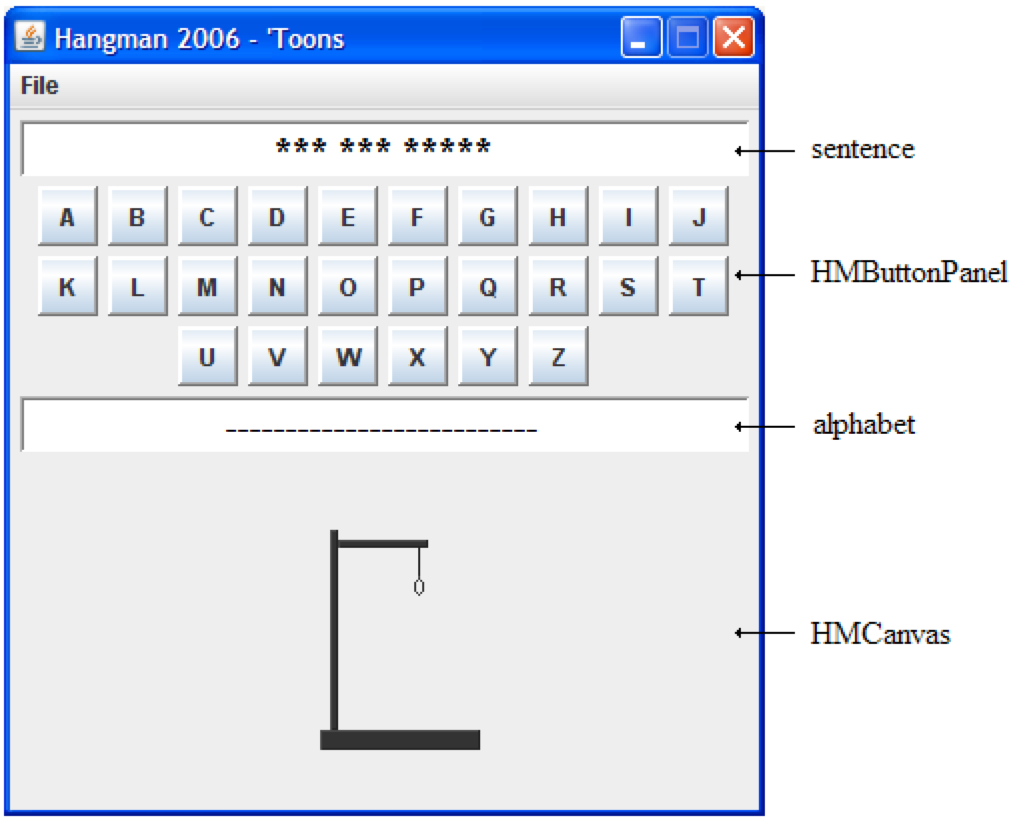
- The program should allow the player to play all ten words without any of the words repeated. Once all ten words have been used the random selection begins anew.
- Final project is due on ______________________, 2016.

SUGGESTIONS

Save Often

Test your program often, and debug as you go.

Plan your time wisely, and set completion goals for yourself.



## REQUIRED METHOD COMPLETION SEQUENCE

**Complete each step in the order listed below. Remember to DOCUMENT as you go!**

1. WORDFILE: Create a data file of 10 words and call it "Hangman.dat".

2. Create a class called **RandomWords**. The RandomWords class should contain three private instance variables:

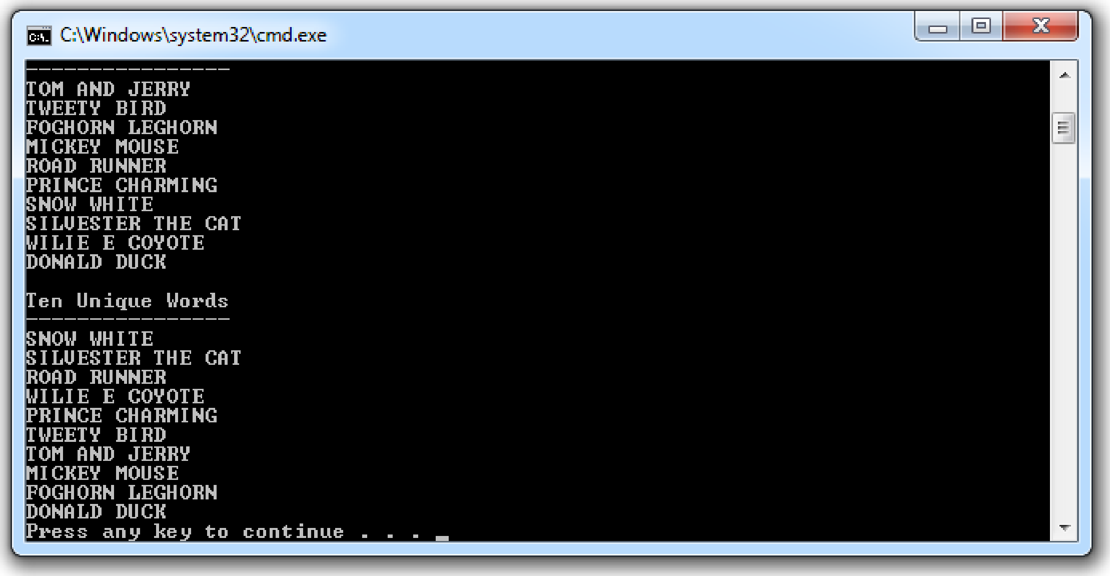
   | | |
   |---|---|
   | **wordList[]** | *An array of 10 strings that will be read from a data file.* |
   | **wordsUsed[]** | *An array of 10 booleans to keep track of words that have been used* |
   | **totalWordsUsed** | *An int to keep track of the number of words that have been used* |

   RandomWords should also contain two methods: A constructor and **chooseRandomWord()**. In the constructor write the instructions that will load the data file into **wordList** (the array of Strings).

3. RANDOM SELECTION. Write a method called **chooseRandomWord()** that returns a String**.** In **chooseRandomWord()** write the instructions that randomly select a word from the array of words (**wordList[]**) and returns the word to the calling method/Class. The method should return a different random word each time it is called, never repeating a word, until all ten words have been used. (HINT #1: Use the boolean array **wordsUsed[]** to keep track of which words have already been used.) (HINT #2: Use a while loop that checks the boolean array **wordsUsed[]** until it finds a word that has not been used. Once an unused word has been found, use it and mark it as used, then increment **totalWordsUsed**). When all the words have been used reinitialize **totalWordsUsed** back to zero and reset all the elements of **wordsUsed[]** back to false. Refer to the *pseudocode* on page 10.

60 Points

4. Test the **RandomWords** class by double clicking on *RandomWords.bat*. A unique word from your word list should be displayed each time **chooseRandomWord()** is called. All ten words should be displayed with no words being repeated. The word list will be randomly displayed twice.



```
----------------
TOM AND JERRY
TWEETY BIRD
FOGHORN LEGHORN
MICKEY MOUSE
ROAD RUNNER
PRINCE CHARMING
SNOW WHITE
SILVESTER THE CAT
WILIE E COYOTE
DONALD DUCK

Ten Unique Words
----------------
SNOW WHITE
SILVESTER THE CAT
ROAD RUNNER
WILIE E COYOTE
PRINCE CHARMING
TWEETY BIRD
TOM AND JERRY
MICKEY MOUSE
FOGHORN LEGHORN
DONALD DUCK
Press any key to continue . . . _
```

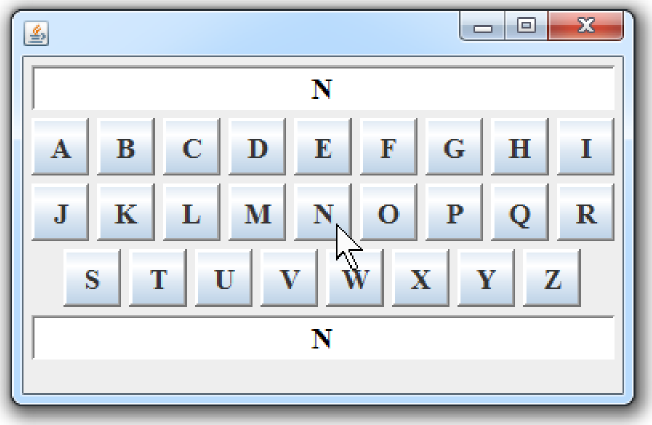
5. Create a class called **HMButtonPanel** that inherits from the JPanel class. The HMButtonPanel class should contain two private instance variables:

| | |
|---|---|
| **sentence** | *An instance of the HMTextField class* |
| **alphabet** | *An instance of the HMTextField class* |

The HMButtonPanel class should contain three methods: A *constructor* that receives an ActionListener as an argument and a set() method for each of the two instance variables (**setSentence(String s)** and **setAlphabet(String s)**). The constructor should add sentence to the HMButtonPanel, add 26 instances of a JButton (A – Z: A count controlled loop would work well here), and then add alphabet. (HMButtonPanel extends JPanel so it has an **add()** method. Set the preferred size of the HMButtonPanel to an acceptable size. (As you add the buttons to the HMButtonPanel don't forget to add the ActionListener to each button.)

70 Points

6. Test the **HMButtonPanel** class by double clicking on *HMButtonPanel.bat*. Your HMButtonPanel will be displayed in a Window. Clicking on any of the buttons should display the appropriate character in both text fields.



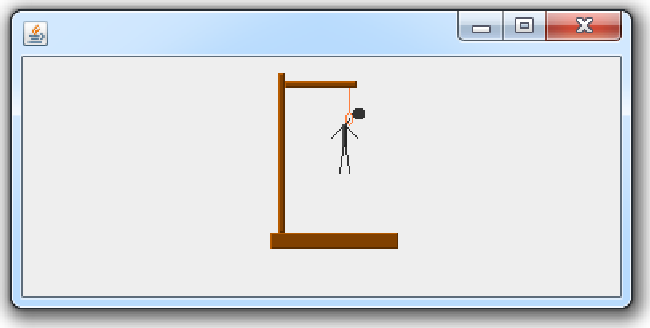
7. Create a class called **HMCanvas** that inherits from the JPanel class. The HMCanvas class will serve as a canvas to draw on. The HMCanvas class will include one instance variable:

| | |
|---|---|
| **misses** | *An int that stores the number of wrong guesses (misses)* |

In the constructor set the preferred size of the panel to an appropriate size. The width should be the same as the width for the HMButtonPanel. The HMCanvas class should contain three additional methods. There should be a **setMisses(**int n**)** and a **getMisses()** methods. In addition, the HMCanvas class should overwrite the JPanal's paintComponent method. (**public void paintComponent(Graphics g)**). This method should draw the gallows and the parts of the body. Which body parts are drawn should be determined by **misses**. The gallows should be redrawn every time **paintComponent()** is called. (HINT: Use a switch statement that switches on **misses.** If **misses** is equal to 0 then draw the gallows, if **misses** is equal to 1 draw the head, 2 draw the body, etc. List them in descending order and have no break statements.).

80 Points

8. Test the **HMCanvas** class by double clicking on *HMCanvas.bat*. Your hangman images should be displayed in the Window (repeatedly cycling from 0 misses through 6 misses.)



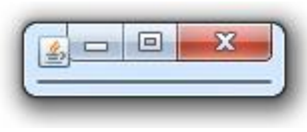
9. Create a class called **Hangman2016** that inherits from the JFrame class. Write a main method that instantiates a Hangman object. I.E. **new Hangman2015()**. At this point your program should look like this:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Hangman2016 extends JFrame
{
    public static void main(String args[]) {
        new Hangman2016().setVisible(true);
    }
}
```

Your program should compile as it is. Because you have **extended** (inherited from) the JFrame class your program has all the properties of a JFrame object. We will instantiate a nested ActionListener to allows you to click on components (JButtons) using the mouse.

10. Write a constructor that calls three methods: `initFrame()`, `initMenu()`, `initGame()`, and `pack()`. You do not need to write any code for those methods yet. Write stubs for `initFrame()`, `initMenu()`, and `initGame()`. Compile and execute your Hangman2015 class and you should see a window that looks like this:



11. Add the following private instance variables to the Hangman class:

| | |
|---|---|
| **lettersUsed[]** | *A boolean array to keep track of which characters have been used* |
| **thisWord** | *A String to hold the current word(s) being used to play the game.* |
| **displayWord** | *String to be displayed. One asterik (*) for each character in the sentence. Blank spaces should appear as blank spaces.* |
| **lettersGuessed** | *String to hold the letters that have been guessed (Initialized with 26 hyphens ( - ) - one for each letter in the alphabet.* |
| **charCount** | *An integer to keep track of the number of characters in the sentence, excluding spaces.* |
| **charsReplaced** | *An integer to keep track of the number of characters that have been correctly guessed.* |
| **wordList** | *An instance of your RandomWords class that will be used to generate random words to be used in the game.* |

| | |
|---|---|
| **top** | ***An instance of the HMButtonPanel() class.*** |
| **canvas** | ***An instance of the HMCanvas() class.*** |

12. In the **initFrame()** method perform the following actions:

    1. Set the frame title
    2. Turn off the Resizable attribute
    3. Set the Default Close Operation to **JFrame.EXIT_ON_CLOSE**
    4. Instantiate an instance of an **ActionListener** to process HMButton clicks.
       **`ActionListener (HMButton)`***:*
       - Call e.getSource() to determine the button pressed. Type cast the result of the call into an HMButton object.
       - If the number of **misses** is less than six and the number of characters replaced is less than the number of characters in the word (spaces do not count) call the method **thisGuess()** sending it the letter of the button that was clicked. (**`button.getText().charAt(0)`** will give you the char value of the button)
       - 
    5. Initialize all instance variables that require it (all objects). The HMButtonPanel constructor will require the ActionListener defined above as an argument.
    6. add top (HMButtonPanel) to the North area of the frame.
    7. add canvas (HMCanvas) to the Center of the frame.

13. In the **initMenu()** method create a **File** menu with two menu items: **New Game** and **Exit**.

    Add **ActionListeners** for each menu item.
    **`ActionListener (New Game)`***:*
    - Begin a new game by calling **initGame()** and repainting the frame.

    `ActionListener (`**`Exit`**`)`**:**
    - Terminate the program

14. The **initGame()** method should initialize all instance variables to their initial states. Make a call to **chooseRandomWord()** (which is a method in the **RandomWords** class) and store the random word in **thisWord**. You will need a FOR loops to initialize the boolean array to *false*. Make the following assignment: **lettersGuessed = "--------------------------"** (one dash for each letter in the alphabet) and display it in **alphabet** (which is a **JTextField** in the **HMButtonPanel** class). **displayWord** should be assigned a String that will consist of a series of asterisks for every letter in the **word** being used. Blank spaces should remain unchanged. Increment **charCount** once for each letter in the word not counting spaces. Hence the word *Big River* becomes *** *****. Display the **displayWord** in **sentence** (which is a **JTextField** in the **HMButtonPanel** class).

15. Write a method **findCharacters(char ch)** that receives a character and searches the **word** for every occurrence of that character. Refer to the *pseudocode* on page 10.

16. Write a method **thisGuess(*char ch*)** that determines if the current letter has been used. Refer to the *pseudocode* on page 10.

17. Write a **gameOver()** method that checks for the end of the game. Refer to the *pseudocode* on page 10.

90 Points

18. BETA test your final version until all bugs are worked out. Go to TURN-IN CHECKLIST.

## TURN-IN REQUIREMENTS

100 Points

- GRADING CRITERIA
  - ✓ _____ Quality of program structure, with appropriate use of parameters, procedures not too long, etc.
  - ✓ _____ Program runs correctly, with appropriate credit for level of "user-friendliness" of the program.

## Pseudocode

### public String chooseRandomWord( )

```
➢ Increment totalWordsUsed
➢ If totalWordsUsed is greater than 10
    • Make totalWordsUsed equal 1
    • Reset the wordsUsed  array to false
➢ Get a random number between 0 and 9
    • While wordsUsed at index <random number> is true get another random number
      between 0 and 9
➢ Make wordsUsed at index position <random number> equal to true
➢ Return wordList at index position <random number>
```

### private boolean findCharacters(char thisChar)

```
➢ Declare a boolean datatype foundChar and assign it the value false.
➢ Compare thisChar with each character in thisWord. (Use a for loop)
    • if the current character equals thisChar
        ▪ thisChar should replace the appropriate asterisk(s) in displayWord. (Use
          substring())
        ▪ Increment charsReplaced each time the character is found.
        ▪ Assign foundChar the value true.
➢ Make sentence (JTextField in the HMButtonPanel) equal to displayWord
➢ Return foundChar which is true if thisChar occurs in thisWord otherwise it is false.
```

### private void thisGuess(char ch)

```
➢ Calculate the index value by subtracting 65 from char (Store it in a variable index)
➢ If lettersUsed at index is false
    • Make lettersUsed at index equal to true
    • Replace the appropriate dash ( - ) with ch in the String lettersGuessed
    • Make alphabet (JTextField in the HMButtonPanel) equal to lettersGuessed
    • Call the method findCharacter passing ch as an argument. If findCharacter returns
      false
    • Increment the number of misses
    • Repaint the HMCanvas
    • Call gameOver()
```

### private void gameOver()

```
➢ If misses (from the HMCanvas class) is equal to 6 (Player has lost!)
    • Display an appropriate message that the player lost in sentence.
    • Display the correct word in alphabet.
➢ Else if charsReplaced is equal to charCount (Player has won!)
    • Display an appropriate message that the player won in sentence.
    • Display the correct word in alphabet.
```