**Lab 23 – Hangman**

Open BlueJ, and create a new BlueJ project titled **Lab23-Hangman** in your CS\LABS folder.

Create a new class and start with this code:

//Name:

import java.util.\*;

public class PracticeProblems

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

**String[] s = {"Hello", "Goodbye", "Computer", "Science", "Um", "Do", "What", "Now"};**

}

}

In this lab, we’ll be working with an array of Strings. Arrays of Strings work just like arrays of ints, except that each element in the list of elements is a String! The element s[0] is “Hello”. REMEMBER – the length property of an array is different from the length() method used on Strings! Examples:

s.length = 8 //number of Strings in the array of Strings *s*

s[1].length() = 7 //number of characters in the element at index 1 (which is a String!)

s.length() <- this will not work! The variable *s* is an array!

s[1].length <- this will not work! The element at index 1 is a String, not an array!

**Before each problem, insert a COMMENT with the problem number.**

1. Note the bolded addition to our skeleton code above. Add this variable declaration to your program – the variable s will be used to test the code you write in the following problems.
2. Print the number of elements in s to the console.
3. Print the first letter of each String in s, separated by space and on the same line.
4. Print the number of Strings in s that end in an “e”.
5. Print the total number of characters in all the Strings in s to the console.
6. Print “I see some long words!” if at least two Strings in s have a length of 7 or more. Print “Mostly short” if not.
7. (Riddle) Someone at a party introduces you to your mother's only sister's husband's sister in law.  He has no brothers.  What do you call this lady?
8. Print how many times you see the letter “e” in the array.

//if you need to check every letter of every String, you'll need nested loops

1. Print the **number of Strings** in s that contain an “e”.

**Hangman app**

Create a new class and start with this code:

//Name:

import java.util.\*;

public class Hangman

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

Each time you start the program, you will be asked to enter a secret word. That secret word will be used as the solution to a game of Hangman.

You will first need to break that secret word into an array of characters. If the secret word is “salsa” we want the character array to contain [‘s’, ‘a’, ‘l’, ‘s’, ‘a’]. There is a cool String method that can quickly do this for us!

//create a char array based on the characters in String str

char[] secretWord = str.toCharArray();

After the secretWord has been entered, we need to hide that from our player so they can play the game. In BlueJ, you can force the terminal window to clear with this line of code: (Only works in BlueJ)

System.out.println("\u000c");

We also need to create a different char array that contains blanks (the underscore character \_). That will be what we show to the player. I recommend calling that char array displayWord.

Now you have 2 parallel arrays: secretWord (containing the correct characters) and displayWord (containing what the player sees while they are guessing.)

When you print displayWord to the screen, you should put spaces to indicate how many letters there are.

Word: \_ \_ \_ \_ \_

Guesses remaining: 6

Please guess a letter >>> **s**

If the user guesses a letter found in secretWord, change the corresponding element in displayWord.

**(But there isn’t a nextChar() in the Scanner class! How do I get a single character from the user? Answer: Get creative! Get their input as a String and then “break off” the first letter. Use that character as the guess.)**

Word: s \_ \_ s \_

Guesses remaining: 6

Please guess a letter >>>

There are many ways we could check and see if player has won the game – for example, when all the elements in displayWord are the same as secretWord*.* The player loses the game if out of guesses.

**For now, we won’t prevent users from guessing the same letter more than once, and we won’t worry about storing the already guessed letters. If the user loses, you should print out what the secret word was.**

Check out this video to see the program running. [Sample Video](https://youtu.be/n99Ky-AiHyo)

Sample input / output:

Enter a secret word >>> **salsa**

(\*\*\*screen clears\*\*\*)

Word: \_ \_ \_ \_ \_

Guesses remaining: 6

Please guess a letter >>> **s**

Word: s \_ \_ s \_

Guesses remaining: 6

Please guess a letter >>> **l**

Word: s \_ l s \_

Guesses remaining: 6

Please guess a letter >>> **o**

Word: s \_ l s \_

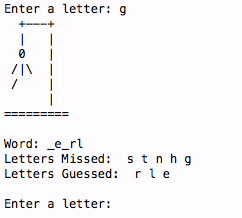
Guesses remaining: 5

Please guess a letter >>> **a**

You got it! The word was: salsa

**(Advanced) Better Hangman app**

In this version of Hangman, store the letters the user has already guessed, and prevent them from entering the same letter twice. In addition, create an ASCII (picture comprised of letters, numbers, and other characters) art version of the ‘gallows’. Example:



Note – you can use the Unicode character for a form feed, \u000c, to "clear" the console between guesses. We do this by putting the escape sequence inside a print statement, like this:

System.out.println("\u000c");

Each time the print statement above is run, the console (terminal) will clear.