CSF
Hwk10
Getting the Hang of It

Write a program that plays the game of hangperson. You are to work in groups of size two or by yourself. The program is due **Tuesday at midnight**. First think about the overall logic of how the game is played. You should have one primary method that starts the game. You will also want to make some helper methods. Please document your code.

The lab includes a Keyboard.class file to help with input. You can use Keyboard.readInt(), Keyboard.readString(), and/or Keyboard.readChar().

Your program should first print out your names, and the instructions for the game.
One person will enter a word to be guessed. Then scroll the word off the screen and start the game.

Don't forget to handle repeated letters! (E.G. he**ll**o).

Following are features to add to your program **after it is running correctly**.
Be sure to test your program after adding each feature.

1) Let the players play again without exiting the program.

2) Tell them how many more bad guesses they have left.

3) Display the set of letters guessed so far

4) Case sensitivity (treat 'E' and 'e' as the same letter)

5) Handle a player accidentally trying an already guessed letter.

6) Handle an invalid guess (such as entering '5')

7*) Have the option to let the computer choose the initial word (hint: you will want to use random numbers to choose from an array of Strings..)

8*) As wrong guesses are made, draw a text based graphic of the hanged.

9*) Give the player the ability to guess the whole word if they want to.

10*) Let two people compete against each other. You could either pick two different words or one word for both.

11*) Add any other cool features you can think of.

Some Java String methods

length()

```
public int length()
```

Returns the length of this string. The length is equal to the number of 16-bit Unicode characters in the string.

Examples:
```
 "happy".length() returns 5
```

Returns:
the length of the sequence of characters in the string.

---

charAt()

```
public char charAt(int index)
```

Returns the character at the specified index. An index ranges from `0` to `length() - 1`. The first character of the sequence is at index `0`, the next at index `1`, and so on..

Examples:
```
 "happy".charAt(1) returns 'a'
```

Parameters:
`index` - the index of the character.
Returns:
the character at the specified index of this string. The first character is at index `0`.

---

equals()

```
public boolean equals(String astring)
```

Compares this string to the specified String. The result is `true` if and only if the argument is a `String` that represents the same sequence of characters as this object.

Examples:

```
"happy".equals("happy") returns true
"happy".equals("Happy") returns false
```

Parameters:
`astring` - the String to compare this `String` against.
Returns:
`true` if the `String` are equal; `false` otherwise.

---

indexOf()

```
public int indexOf(char ch)
```

Returns the index within this string of the first occurrence of the specified character. If a character with value `ch` occurs in the character sequence represented by this `String` object, then the index of the first such occurrence is returned -- that is, the smallest value *k* such that:

```
str.charAt(k) == ch
```

is `true`. If no such character occurs in this string, then `-1` is returned.

Examples:
```
"happy".indexOf('p') returns 2
"happy".indexOf('x') returns -1
```

Parameters:
`ch` - a character.
Returns:
the index of the first occurrence of the character in the character sequence represented by this object, or `-1` if the character does not occur.

---

substring()

```
public String substring(int beginIndex)
```

Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

Examples:
```
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)
```

Parameters:
`beginIndex` - the beginning index, inclusive.
Returns:
the specified substring.

---

substring()

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:
```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:
`beginIndex` - the beginning index, inclusive.
`endIndex` - the ending index, exclusive.
Returns:
the specified substring.