# Assignment 3

# Hangman Game

Due at the Start of Class Dec 6th

**Hangman Application**

For this assignment we will be creating a game of hangman. To play hangman, one player chooses a word to be guessed. Then the other player guesses letters that the word might contain. If they successfully guess a letter that is a part of the word, the position of that letter is revealed to the player. If they guess a letter than isn’t in the word, then we tally a miss. Play continues until a player finally guesses all letters in the word, in which case they win, or until they have tallied 6 misses, in which case they lose.

At the start of the game, one player will choose their word. You must then take this word and convert it into a character array. Then, the other player will be allowed to guess specific characters. If a guess is incorrect, you should add a miss. In either case, whether the guess was correct or not, you should keep note of the guessed letter by using an array full of booleans. This array will contain one boolean value for each letter of the alphabet, representing whether or not that letter has been guessed. At the start of the game, all entries in this array will false, indicating that no letter have been guessed yet.

You should create your program within the provided HangmanGame.java file, inside of which you will find some code has already been provided. To create our program we should add the following methods:

**char[] toCharArray(String word) –**

You should write a method that takes as input a String representing a word, and converts that string into an array of characters, where each character in the word is placed in order into an array. The method should then return that array.

Note: The only String methods you are allowed to use are length() and charAt(). Any other methods will cause you to lose marks.

**char toUpperCase(char c)** –

The game would be extra difficult if we didn’t accept characters based on their case. In order to ensure that our program is case insensitive, we will write a method called “toUpperCase” which takes as input a character, and returns that character in uppercase, regardless of the case of the original character.

When we compare a guess character against a character in the word in the **isLetterInWord** method, you should use the **toUpperCase** method to convert both characters to uppercase first, such that it doesn’t matter whether the input was uppercase or lowercase.

**boolean isLetterInWord(char[] word, char c) –**

You should write a method that takes as input a char[] representing a word, and a char containing a character we would like to find in that word. The method should return true if the character exists in the word.

**void printGuesses(boolean[] guesses) –**

You should write a method called printGuesses that takes as input an array of booleans representing the letters than a player has guessed. You can assume this array has 26 entries, with the first representing the letter ‘A’, and the last representing the letter ‘Z’. A value of true indicates that the letter has been guessed already, a value of false means the player has not yet chosen this letter. This method should go through the array of booleans and print out all letters that the player has chosen so far.

Ex: For guesses: {false, true, true, false, true, false, true, … , false}

The method should print: B, C, E, G, …

**void printWord(char[] word, boolean[] guesses) –**

The printWord method will display the letters of the word that the player has successfully guessed. This method will take as input the char[] representing word they are trying to guess, and a boolean[] representing which letters have been guessed. For each letter of the word, you should check whether or not that letter has been guessed by consulting the boolean[]. If it has been guessed, you should print the letter, if it has not been guessed you should print an underscore (‘\_’).

Ex: For the word: {‘C’,’A’,’B’,’B’,’A’,’G’,’E’}

And guesses: {false, true, true, false, false, false, true, … , false}

The method should print: C \_ B B \_ G \_

**boolean isGameOver(char[] word, boolean[] guesses, int misses) –**

The isGameOver method will determine whether or not the game has ended. This can happen either because the word has been guessed, or because the player has made too many misses. This method will take as input the char[] representing word they are trying to guess, a boolean[] representing which letters have been guessed, and an int representing how many misses the player has made.

The first thing this method should check is whether or not 6 misses have been made. If there are 6 misses, the method should return true right away.

If not, the general structure of the rest of this method is almost identical to that of the printWord method. For each letter of the word, you should check whether or not that letter has been guessed by consulting the boolean[]. If any given letter in the word has not been guessed, this method should return false. If all the letters have been guessed, it should return true.

Ex: For the word: {‘C’,’A’,’B’,’B’,’A’,’G’,’E’}

And guesses: {false, true, true, false, false, false, true, … , false}

And misses: 3

The method should return false.

For the word: {‘C’,’A’,’B’,’B’,’A’,’G’,’E’}

And guesses: {true, true, true, false, true, false, true, … , false}

And misses: 3

The method should return true.

(Hint: This method is almost identical to the printWord method, except we are keeping track of a boolean value instead of printing)

**void runGame()**

The actual gameplay itself should take place in a method called **runGame**. This method should take no input, and has no return value.

In this method, you should create and initialize a boolean[] with size 26 to represent the letters which have been guessed. The entry at index 0 will represent the letter ‘A’, index 1 will represent ‘B’, and so on until 25 which will represent the letter ‘Z’. For each boolean, false indicates that that letter has yet to be guessed, while true indicates that that letter has been previously guessed. To start with, they should all be false, but as the player guesses letters they should be set to true.

You should also generate a char[] to represent the word being guessed from user input. You should take in a String and convert into a char[] using the **toCharArray** method. You should also create an int variable to keep track of the number of misses a player has made.

To run the game, this method should have a loop which continuously asks the player to enter a letter. Then, once a letter has been provided, a call should be made to **isLetterInWord**, which will determine whether or not the letter occurs in the word. If **isLetterInWord** returns false, we should instead increment the number of misses the player has made. If it returns true, nothing happens. In either case, you should update the guesses boolean[] to indicate which letter was guessed.

After every single guess, you should print out the current state of the game so players know what is happening. To do this, you will start by printing out the guessed letters of the word by using the **printWord** method. You should then print out the full list of letters that have been guessed by using the **printGuesses** method. In addition to these 2 methods, the method **printHangman** has been provided. **printHangman** takes as a parameter the number of misses the player has made so far, and prints out a stickman representation of the misses. You should call this method as well, providing to it the number of misses.

The game loop should continue until either all of the letters have been guessed or until the player has made 6 misses. You should check these conditions using the **isGameOver** method. After the game has ended, if the player has made 6 or more misses you should print out the message “You lose!”, if fewer than 6 misses have been made you should print “You win!”.

**void main(String[] args)**

Your main method should call the runGame method, and do nothing else.

What to submit:

* A hard copy of the source code of your program.
* The source code (.java files) are to be submitted online.