# Java String and StringBuilder

## github-classClassroom Assignment

* [Acceptance Link](https://classroom.github.com/a/W-kycxmD)

## laptop computer01

**Points: 7**

Use the code skeleton in e01.

1. Create a new Java class called StringExercise with a main method. In the main method, check if there are any command-line arguments. If not, print an error message and exit the program.
2. If there are command-line arguments, create a StringBuilder object and append all the arguments to it, separated by spaces.
3. Print the resulting string to the console.
4. Create a class call StringUtilities.
5. Implement a static method in the StringUtilities class called reverseString that takes a string as input and returns the reverse of that string.
6. Call the reverseString method on the concatenated string from step 3 and print the result to the console.
7. Implement a static method in the StringUtilities class called countVowels that takes a string as input and returns the number of vowels (a, e, i, o, u) in that string.
8. Call the countVowels method on the concatenated string from step 3 and print the result to the console.
9. Implement a static method in the StringUtilities class called countVowels that takes a string as input and returns the number of vowels (a, e, i, o, u) in that string.
10. Call the countVowels method on the concatenated string from step 3 and print the result to the console.
11. Implement a static method in the StringUtilities class called countWords that takes a string as input and returns the number of words in that string.
12. Call the countVowels method on the concatenated string from step 3 and print the result to the console.
13. Implement a static method in the StringUtilities class called characterFrequencyMap that takes a string as input and returns a Map<Character, Integer> of containing the char and count in that string.
14. Call the characterFrequencyMap method on the concatenated string from step 3 and print the result to the console.

To run the program, you can pass one or more strings as command-line arguments, like this:

java StringExercise hello world!

This will produce the following output:

Concatenated string: hello world!

Reversed string: !dlrow olleh

Number of vowels: 3

Number of words: 2

Character frequency:

  : 1

! : 1

r : 1

d : 1

e : 1

w : 1

h : 1

l : 3

o : 2

## trophy01

**Points: 5**

Use the code skeleton in b01.

Implement a program that will play [Hangman](https://en.wikipedia.org/wiki/Hangman_(game)) with a user.

The program should:

* Print the current state of the hangman.
* Print the current hidden word.
* Prompt the user to guess a letter.
* Check if the letter has already been guessed. If it has, print an error message and prompt the user again.
* Check if the letter appears in the word. If it does, update the hidden word. If not, increment the number of wrong guesses.
* Check if the user has guessed the word. If they have, print a congratulatory message and end the game. If not, continue to the next iteration.
* If the user runs out of guesses, print a message saying they lost and reveal the word.

The program display a **\_** for each letter that is missing. The program pick a random word from a list of words stored in a **file**, eg. a word per line.

The game needs to start with java Hangman.

The flow of the game should look like this:

Play some Hangman.

 \_\_\_\_\_\_\_

 |      |

 |

 |

 |

\_||\_

\_\_\_\_\_

Guess count: 0

Guess a letter:

a

 \_\_\_\_\_\_\_

 |      |

 |

 |

 |

\_||\_

a\_\_\_\_

Guess count: 1

Guess a letter:

p

 \_\_\_\_\_\_\_

 |      |

 |

 |

 |

\_||\_

app\_\_

Guess count: 2

Guess a letter:

l

 \_\_\_\_\_\_\_

 |      |

 |

 |

 |

\_||\_

appl\_

Guess count: 3

Guess a letter:

e

apple

You got it with 4 guesses!

The HangMan states.

 // Start

 \_\_\_\_\_\_\_

 |      |

 |

 |

 |

\_||\_

// As the player guesses incorrect letters, you add elements to the hangman, like this:

 \_\_\_\_\_\_\_

 |      |

 |      O

 |

 |

\_||\_

 \_\_\_\_\_\_\_

 |      |

 |      O

 |      |

 |

\_||\_

 \_\_\_\_\_\_\_

 |      |

 |      O

 |     /|

 |

\_||\_

 \_\_\_\_\_\_\_

 |      |

 |      O

 |     /|\

 |

\_||\_

 \_\_\_\_\_\_\_

 |      |

 |      O

 |     /|\

 |     /

\_||\_

 \_\_\_\_\_\_\_

 |      |

 |      O

 |     /|\

 |     / \

\_||\_

You can store each stage of the hangman as a separate string in your code, and then print the appropriate string based on how many incorrect guesses the player has made.

light bulbYou can clear the command line between the turns with the following code:

System.out.print("\033[H\033[2J");

System.out.flush();

The escape sequence \033[H moves the cursor to the top left corner of the screen, and \033[2J clears the screen. Finally, System.out.flush() flushes the output buffer to ensure that the screen is cleared immediately.