String Methods

Pg 138-140 in TB

Name of method	Description	Example
length()	Put the name of a string and a "." before length, to find the length of the string.	System.out.println(word3.length()); -Will print out the length of the word3 string.
substring(int start, int end)	Put two numbers It will then print out the letter in the position of the first number. **POSITIONING STARTS AT 0!	String word3 = ("Hello"); System.out.println(word3.substring(2, 3)); Output: l --- String word3 = ("Hello"); System.out.println(word3.substring(2, 4)); Output: ll
substring(int start)	This will start printing the string at whatever starting position you tell it to. **Numbering starts at zero. H- 0 E- 1 L- 2 L- 3 O- 4	String word3 = ("Hello"); System.out.println(word3.substring (0)); Output: Hello System.out.println(word3.substring (1)); Output: ello And so on...
toLowerCase()	If the whole word is written in uppercase letters, this will print out the word in lowercase letters. If some letters of the word is uppercase and others are lowercase, the uppercase letters will become lowercase, and the lowercase letters will remain the same.	String word3 = "HELLO"; System.out.println(word3.toLowerCase()); Output: hello String word3 = "HeLlO"; System.out.println(word3.toLowerCase()); Output: hello
toUpperCase()	If the whole word is written in lowercase letters, this will print out the word in uppercase letters. If some letters of the word is lowercase and others are uppercase, the lowercase letters will become uppercase, and the uppercase letters will remain the same.	String word3 = "hello"; System.out.println(word3.toUpperCase()); Output: HELLO String word3 = "HelLO"; System.out.println(word3.toUpperCase()); Output: HELLO

trim()	Will remove all leading or trailing white spaces in a string	Without trim() String word3 = "hello "; System.out.print(word3); System.out.print("world"); Output: hello world With trim(): String word3 = "hello "; System.out.print(word3.trim()); System.out.print("world"); Output: helloworld
replaceFirst (String str, String str2)	Will replace the first instance of a letter (first time a letter appears in a word) with another one.	String word3 = "hello "; System.out.println(word3.replaceFirst ("l","j")); Output: hejlo ** replaced the first “l” with j.
replaceAll (String str, String str2)	Will replace a certain letter in a word with another one. This one replaces the letter every time it comes up in the word, not just the first time like the replaceFirst method	String word3 = "hello "; System.out.println(word3.replaceAll ("l" , "j")); Output: hejjo ** replaced all l’s with j.
equals(string str)	If the same string is in the condition it will output true. If not it will output false.	String word3 = "hello "; String word = "hello "; System.out.println(word.equals (word3)); Output: true --- String word = “test”; String word3 = "hello "; System.out.println(word3.equals (word)); Output: false
equalsIgnoreCase(String str)	If the same string is in the condition it will output true. If not it will output false. Same as equals(string str), expect it ignores casing when comparing strings.	String word3 = "hello "; String word = HELLO; System.out.println(word.equals (word3)); Output: true --- String word = “test”; String word3 = "hello "; System.out.println(word3.equals (word)); Output: false

compareTo(String str)	-Will output a 0 if the two strings being compared are exactly the same, by having the same word and casing. -If the string in the bracket comes comes alphabetically before the string a positive number is displayed. -If the string in the bracket comes comes alphabetically after the string a negative number is displayed.	Sample output on next page String word3 = "hello "; String word2 = "hello "; System.out.println(word3.compareTo(word2)); Output: 0 --- String word = "a"; String word3 = "b "; System.out.println(word3.compareTo(word)); Output: 1 --- String word = "a"; String word3 = "b "; System.out.println(word.compareTo(word3)); Output: -1
compareToIgnoreCase(string str)	Same as compareTo expect uppercase and lowercase differences are ignored. Must only be same word. Same rules with alphabetically before and after, main difference is when the strings casing are different.	String word = "a"; String word3 = "A"; System.out.println(word.compareToIgnoreCase(word3)); Output: 0
indexOf (string str)	Tells you where the position of a letter is in a string is. If there is two of the same letter in a word it will give you the FIRST occurrence of that letter. If you enter a letter that is not in a string the output will be -1 meaning this does not exist in the string. ** Numbering starts at 0.	String word3 = "apple"; System.out.println(word3.indexOf("l")); Output: 3 --- String word3 = "apple"; System.out.println(word3.indexOf("z")); Output: -1 Since there is no letter “z” in apple
lastIndexOf (string str)	Similar to indexOf. Main difference, this will give you the LAST occurrence of the letter in a word. If you enter a letter that is not in a string the output will be -1 meaning this does not exist in the string. ** Numbering starts at 0.	String word3 = "apple"; System.out.println(word3.lastIndexOf("p")); Output: 2 ** If you used indexOf the output would be 1 since the FIRST occurrence of “p” is at position 1.
startsWith (string str)	If the word starts with the letter you	

	input it will output true. If it does not start with the letter you input it will output false.	String word3 = "apple"; System.out.println(word3.startsWith("a")); Output: True --- String word3 = "apple"; System.out.println(word3.startsWith("z")); Output: false
endsWith (string str)	Similar to startsWith (string str) but this looks at the ending letter. If the word ends the letter you input it will output true. If it does not end with the letter you input it will output false.	String word3 = "apple"; System.out.println(word3.endsWith("e")); Output: true --- String word3 = "apple"; System.out.println(word3.endsWith("z")); Output: false