**String**

## Creating a String

Strings in Java are objects. Therefore you need to use the new operator to create a new Java String object. Here is a Java String instantiation (creation) example:

String myString = new String("Hello World");

The text inside the quotes is the text the String object will contain.

### Java String Literals

Java has a shorter way of creating a new String:

String myString = "Hello World";

Instead of passing the text "Hello World" as a parameter to the String constructor, you can just write the text itself inside the double quote characters. This is called a String literal. The Java compiler will internally figure out how to create a new Java String representing the given text.

### String class function

The following methods are some of the most commonly used methods of String class.

#### charAt()

charAt() function returns the character located at the specified index.

String str = "studytonight";

System.out.println(str.charAt(2));

**Output:** u

**NOTE:** Index of a String starts from 0, hence str.charAt(2) means third character of the String str.

#### equalsIgnoreCase()

equalsIgnoreCase() determines the equality of two Strings, ignoring thier case (upper or lower case doesn't matters with this fuction ).

String str = "java";

System.out.println(str.equalsIgnoreCase("JAVA"));

true

#### indexOf()

indexOf() function returns the index of first occurrence of a substring or a character. indexOf() method has four forms:

* int indexOf(String str): It returns the index within this string of the first occurrence of the specified substring.
* int indexOf(int ch, int fromIndex): It returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
* int indexOf(int ch): It returns the index within this string of the first occurrence of the specified character.
* int indexOf(String str, int fromIndex): It returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

**Example:**

public class StudyTonight {

public static void main(String[] args) {

String str="StudyTonight";

System.out.println(str.indexOf('u')); //3rd form

System.out.println(str.indexOf('t', 3)); //2nd form

String subString="Ton";

System.out.println(str.indexOf(subString)); //1st form

System.out.println(str.indexOf(subString,7)); //4th form

}

}

2 11 5 -1

**NOTE:** -1 indicates that the substring/Character is not found in the given String.

#### length()

length() function returns the number of characters in a String.

String str = "Count me";

System.out.println(str.length());

8

#### replace()

replace() method replaces occurances of character with a specified new character.

String str = "Change me";

System.out.println(str.replace('m','M'));

Change Me

## Substrings

You can extract a part of a String. This is called a substring. You do so using the substring() method of the String class. Here is an example:

String string1 = "Hello World";

String substring = string1.substring(0,5);

After this code is executed the substring variable will contain the string Hello.

The substring() method takes two parameters. The first is the character index of the first character to be included in the substring. The second is the index of the character *after* the last character to be included in the substring. Remember that. The parameters mean "from - including, to - *excluding*". This can be a little confusing until you memorize it.

The first character in a String has index 0, the second character has index 1 etc. The last character in the string has has the index String.length() - 1.

## Comparing Strings

Java Strings also have a set of methods used to compare Strings. These methods are:

* equals()
* equalsIgnoreCase()
* startsWith()
* endsWith()
* compareTo()

### equals()

The equals() method tests if two Strings are exactly equal to each other. If they are, the equals() method returns true. If not, it returns false. Here is an example:

String one = "abc";

String two = "def";

String three = "abc";

String four = "ABC";

System.out.println( one.equals(two) );

System.out.println( one.equals(three) );

System.out.println( one.equals(four) );

The two strings one and three are equal, but one is not equal to two or to four. The case of the characters must match exactly too, so lowercase characters are not equal to uppercase characters.

The output printed from the code above would be:

false

true

false

### equalsIgnoreCase()

The String class also has a method called equalsIgnoreCase() which compares two strings but ignores the case of the characters. Thus, uppercase characters are considered to be equal to their lowercase equivalents.

### startsWith() and endsWith()

The startsWith() and endsWith() methods check if the String starts with a certain substring. Here are a few examples:

String one = "This is a good day to code";

System.out.println( one.startsWith("This") );

System.out.println( one.startsWith("This", 5) );

System.out.println( one.endsWith ("code") );

System.out.println( one.endsWith ("shower") );

This example creates a String and checks if it starts and ends with various substrings.

The first line (after the String declaration) checks if the String starts with the substring "This". Since it does, the startsWith() method returns true.

The second line checks if the String starts with the substring "This" when starting the comparison from the character with index 5. The result is false, since the character at index 5 is "i".

The third line checks if the String ends with the substring "code". Since it does, the endsWith() method returns true.

The fourth line checks if the String ends with the substring "shower". Since it does not, the endsWith() method returns false.

### compareTo()

The compareTo() method compares the String to another String and returns an int telling whether this String is smaller, equal to or larger than the other String. If the String is earlier in sorting order than the other String, compareTo() returns a negative number. If the String is equal in sorting order to the other String, compareTo() returns 0. If the String is after the other String in sorting order, the compareTo() metod returns a positive number.

Here is an example:

String one = "abc";

String two = "def";

String three = "abd";

System.out.println( one.compareTo(two) );

System.out.println( one.compareTo(three) );

This example compares the one String to two other Strings. The output printed from this code would be:

-3

-1