COURSEWARE



# String Manipulation

## Contents



## Overview

Strings are used for storing text. In technical terms, a string is an array of characters.
Strings are treated as objects in Java. There is a String class which can be used to create and manipulate strings.
It is important to note that String is NOT a primitive data type in Java like in some other programming languages (e.g. Python, JavaScript).
String objects are immutable in Java, so once it is created a String object cannot be changed.

## Creating Strings

Unlike in some other programming languages (e.g. Python), strings can only be created with double quotation marks ("") and not single quotation marks ('') in Java.

### Method 1

The most straightforward way to create a string in Java:

```
public class StringDemo {

    public static void main(String args[]) {
        String strLiteral = "Hello World!";
        System.out.println(strLiteral); // output will be: Hello World!
    }
}
```

### Method 2

It is also possible to create a String using the `new` operator:

```
public class StringDemo {

    public static void main(String args[]) {
        String strObject = new String("Hello World!");
        System.out.println(strObject); // output will be: Hello World!
    }
}
```

There is a subtle difference between these two methods.
Although both expressions are String objects, the `new` operator will always create a new String object.
By contrast, when you create strings using method 1, they are interned (Java

automatically calls intern() method in order to put these objects into the String pool).
The String pool is a special section of the JVM used to store **String literals** (basically any string created using `""` rather than `new`).
Caching strings in this way can save a lot of memory due to strings being such a commonly used data type.
In simple terms, this means that if you use method 2, Java will see your strings as different objects even if their values are the same:

```
public class StringDemo {

    public static void main(String args[]) {

        // Method 1
        String str1 = "I like strings";
        String str2 = "I like strings";
        System.out.println(str1 == str2); // output will be: true

        // Method 2
        String str3 = new String("I like strings");
        String str4 = new String("I like strings");
        System.out.println(str3 == str4); // output will be: false
     }
}
```

However, you could still compare the values of two strings using the `equals()` method:

```
        String str3 = new String("I like strings");
        String str4 = new String("I like strings");
        System.out.println(str3.equals(str4));  // output will be: true
```

## Creating a String through arrays

```
public class StringDemo {

     public static void main(String args[]) {
      char[] helloArray = { 'H', 'e', 'l', 'l', 'o', '.' }; // an array of chars
      String helloString = new String(helloArray);
      System.out.println(helloString); // prints out Hello.
   }
}
```

## String Length

One method that you can use with strings is the `length()` method which returns the number of characters contained in a string object.
Methods used to obtain information about an object are known as **accessor methods**.

```
public class StringDemo {

    public static void main(String args[]) {
        String myStr = "I like strings";
        int stringLength = myStr.length();
        System.out.println("String Length is " + stringLength);
        // output will be: String Length is 14
    }
}
```

## Concatenating Strings

The String class also includes a method for concatenating two or more strings:

```java
public class StringDemo {

    public static void main(String args[]) {
        //example 1
        String myStr = "My cat's name is ".concat("Copycat");
        System.out.println(myStr);
        // output will be: My cat's name is Copycat

        //example 2
        String myStr2 = "Hello, " + "World" + "!" ;
        System.out.println(myStr2);
        // output will be: Hello, World!

        //example 2.5
        String myStr3 = "Hello,"+ "World" + "!"; // Be careful, don't forget to
add spaces!
        System.out.println(myStr3);
        //output will be: Hello,World! because no spaces were included above

        //example 3
        String string1 = "Welcome ";
        String string2 = "to ";
        String string3 = "Java";
        String string4 = "!";
        System.out.println(string1 + string2 + string3 + string4);
        // output will be: Welcome to Java!
    }
}
```

## Other String methods

There are many other String manipulation methods. Here are some of the most common:

```java
public class StringDemo {

    public static void main(String args[]) {
        String str1 = "ThIsIsSoMeTeXt";
        String str2 = "thisissometext";

        // Converts text to lowercase
        System.out.println(str1.toLowerCase());// output will be: thisisometext

        // Converts text to UPPERCASE
        System.out.println(str1.toUpperCase()); // output will be:
THISISSOMETEXT

        // Returns the character at the specified index
        System.out.println(str1.charAt(3)); // output will be: s

        // Returns the first position of the character specified.
        System.out.println(str1.indexOf("I")); // output will be: 2

        // Returns true or false if the string ends with the specified character
        System.out.println(str1.startsWith("t")); // output will be: false

        // Returns true or false if the string ends with the specified character
        System.out.println(str1.endsWith("t")); // output will be: true

        // Returns true or false if the string contains the specified characters
        System.out.println(str1.contains("So")); // output will be: true

        // Returns a new string that is a substring of this string.
        // start index is inclusive and endindex is exclusive
        System.out.println(str1.substring(1, 3)); // output will be: hIs

        // Checks if the values of two strings are the same:
        System.out.println(str1.equals(str2)); // output will be: false

        // Compares two strings ignoring case considerations:
        System.out.println(str1.equalsIgnoreCase(str2)); // output will be: true

        //Searches a string for a specified value and then replaces it with
another value
        System.out.println(str1.replace("SoMe", "SOMEREALLYFUN"));  //output
will be: ThIsIsSOMEREALLYFUNTeXt


    }
}
```

## Tutorial

There are no tutorials for this module.

## Exercises

1. Create two Strings where one string has a value of “yesterday it was raining” and the other string with a value of “today it is sunny”

   Concatenate both values, capitalise both strings and print out the result. The result should be: `TODAY IT IS SUNNY, YESTERDAY IT WAS RAINING`

   ▶ Solution

   Now use the substring method to print out: `TODAY IT IS RAINING`

   ▶ Solution
2. For this task you are to implement 4 methods that manipulate `String` objects. For all parts of this section you are only allowed to use the `length()`, `substring()`, and `equals()` methods. Avoid using arrays or any methods you have yet to be taught as this defeats the purpose of the exercise:

- Method 1 - When given a `String`, count and return how many words there are in that `String`.
- Method 2 - When given a `String`, print out this `String` in a vertical fashion, each word on a different line.
- Method 3 - When given a `String`, print out this `String` in a vertical fashion in reverse order, each word on a different line.
- Method 4 - A find method, which takes 2 `String`s; the first is message and the second is the `String` you want to find in the message. A boolean value should be returned to indicate whether or not the second `String` has been found in the message.