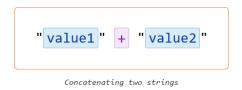
# **Examples of working with strings**

### 1. Concatenation (merging strings)

There's this slick and simple thing that you can do with strings in Java: you can glue them together. This operation is called *concatenation*. Here's how we remember it: Con-Cat-en-Nation. It is often called "joining strings" or "combining strings".

To concatenate two lines, you use the + sign. It's very easy:



### Examples:

Statement	Note
String name = "Steve" + "Steve";	name contains the string SteveSteve
String city = "New York" + "Steve";	city contains the string New YorkSteve
String message = "Hello!" + "Steve";	message contains the string Hello! Steve

And, of course, you can join lots of strings at the same time, and you can also join strings and variables.

#### **Examples:**

```
String name = "Steve";
String city = "New York";
String message = "Hello!" + city + name + city;

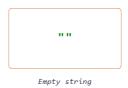
Mote

name contains the string Steve city contains the string New York
message contains the string
Hello!New YorkSteveNew York
```

In the last example, you can see that the text in the message is difficult to read, because it is missing spaces. To indicate one or more spaces, you just need to write them in code and then wrap them in double quotes. It's easier than it sounds:



By the way, if you don't put any spaces between the quotes (i.e. you write two double quotes in a row), you get the so-called "empty string":



On the one hand, it seems we have a string. But on the other hand, when we display this string, nothing is displayed. And when we join it with other strings, nothing happens. It's kind of like a zero in addition, only for strings.

## 2. Converting to a string

As mentioned above, Java developers have made sure that absolutely every variable, object, and expression in Java can be converted to the String type.

What's more, this happens automatically when we concatenate a String with some *other type*. Examples:

```
Statement

int a = 5;
String name = "Steve" + a;

int a = 5;
String city = a + "New York" + a;

int number = 10;
String code = "Yo";
String message = "Hello!" + number + code;
```

In all three instances, we calmly combined **int** and String variables, and the result is always a String.

```
You can't perform arithmetic operations with the String type. Even if the entire string consists of digits.
```

#### **Examples:**

```
Statement

int a = 5;
String name = "1" + a;

int a = 5;
String city = a + "9" + a;

int number = 10;
String code = "10";
String message = "" + number + code;
```

The plus operations are executed from left to right, so the result may be somewhat unexpected. Example:

```
Statement

Int a = 5;
String name = a + a + "1" + a;

Order of operations: ((a + a) + "1") + a
```

## 3. Converting a string to a number

Converting a number to a string in Java is as easy as concatenating it to an empty string:

```
String str = "" + number;

Converting a number to a string
```

But what if you need to convert a string to a number? Well, not every string can be converted to a number. But if the string consists only of numbers, then you can. There is a special *method* for this in the **Integer** class.

The corresponding statement looks like this:

```
int x = Integer . parseInt (string);
```

Where int x is the declaration of an x integer variable, and string is a string that represents a number (i.e. a string consisting of digits).

### Examples:

```
String str = "123";
int number = Integer .parseInt(str);

int number = Integer .parseInt("321");
number contains the number 321

int number = Integer .parseInt("321" + 0);
number contains the number 3210

int number = "321";
This won't compile: the variable is an int, but the value is a String
```

### 4. Converting an object/primitive to a string

To convert an instance of any Java class or any primitive data type to a string, you can use the String.valueOf() method:

```
1
     public class StringExamples {
 2
         public static void main(String[] args) {
              String a = String.valueOf(1);
 3
 4
              String b = String.valueOf(12.0D);
 5
              String c = String.valueOf(123.4F);
 6
              String d = String.valueOf(123456L);
              String s = String.valueOf(true);
 8
 9
              System.out.println(a);
              System.out.println(b);
10
11
              System.out.println(c);
              System.out.println(d);
12
13
              System.out.println(s);
14
15
16
             Output:
17
             12.0
18
              123.4
19
              123456
20
21
              true
               */
22
23
          }
24
     }
```

# 5. Some methods for working with strings

And finally, I would like to talk about several methods of the String class.

### length() method

The length() method lets you get the length of a string, i.e. how many characters it contains.

#### Examples:

Statement	Note
<pre>String name = "Rome"; int count = name. length();</pre>	count contains the value 4
<pre>int count = "".length();</pre>	count contains the value Ø
<pre>string name = "Rom"; int count = (name + 12).length();</pre>	count contains the value 5

You can call these methods on anything whose type is String, even an expression:

```
( name + 12).length()

Calling the Length() method on an expression whose type is String
```

### toLowerCase() method

The toLowerCase() method lets you convert all characters in a string to lowercase:

#### Examples:

Statement	Note
<pre>String name = "Rom"; String name2 = name. toLowerCase();</pre>	name2 contains the string rom
String name = "". toLowerCase();	name contains an empty string
String name = "ROM123"; String name2 = name. toLowerCase();	name2 contains the string rom123

# toUpperCase() method

The toUpperCase() method lets you convert all characters in a string to uppercase:

### Examples:

Statement	Note
String name = "Rom"; String name2 = name. toUpperCase();	name2 contains the string ROM
<pre>String name = "rom123"; String name2 = name. toUpperCase();</pre>	name2 contains the string ROM123