# Theory: Replacing characters

16 minutes 0 / 5 problems solved

Skip this topic Start practicing

1133 users solved this topic. Latest completion was about 9 hours ago.

Sometimes we may need to replace a substring with another string. Java provides us with several convenient methods to do this, and regular expressions play an important role in this process.

## §1. The methods replaceFirst and replaceAll of a string

There are two strings methods for substring replacement:

- `String replaceFirst(String regex, String replacement)` replaces the first occurrence of `regex` in the string with `replacement`;
- `String replaceAll(String regex, String replacement)` replaces all occurrences of `regex` in the string with `replacement`;

where

- `regex` is a regular expression matching the substrings that are to be replaced;
- `replacement` is a string that will replace a substring matching the regex (it should be just a string, not a regex!).

Since strings are immutable objects, both methods return a new string with necessary modifications.

Be careful, the `replace` method performs a similar kind of operation but it does NOT support regular expressions.

Let's look at the examples below.

```
String digitRegex = "\\d"; // a regex to match a digit

String str = "ab73c80abc9"; // a string consisting of letters and digits

String result1 = str.replaceAll(digitRegex, "#"); // it replaces each digit with #

System.out.println(result1); // "ab##c##abc#"

String result2 = str.replaceFirst(digitRegex, "#"); // it replaces only the first digit with #

System.out.println(result2); // "ab#3c80abc9"
```

It is possible to use any regex as the first argument of these methods. The following example demonstrates how to replace all sequences of uppercase Latin letters in a string with a single dash character.

```
String regex = "[A-Z]+";

String str = "aBoeQNmDFEFu";

String result = str.replaceAll(regex, "-"); // "a-oe-m-u"
```

The Matcher class, however, has the same methods. Let's take a quick look at them.

## §2. The methods replaceFirst and replaceAll of a matcher

An object of `Matcher` also has two methods for replacing a substring found by means of a regular expression:

Current topic:

Replacing characters

Topic depends on:

Patterns and Matcher

- `Matcher replaceFirst(String replacement)`;
- `Matcher replaceAll(String replacement)`.

The difference between string methods and these ones is that the `Matcher`'s methods do not take regexes as their arguments: any `Matcher` object gets its regex when being initiated. See the example below.

```
Pattern pattern = Pattern.compile("\\d"); // a regex to match a digit

String str = "ab73c80abc9"; // a string consisting of letters and digits

Matcher matcher = pattern.matcher(str);

System.out.println(matcher.replaceAll("#"));   // ab##c##abc#
System.out.println(matcher.replaceFirst("#")); // ab#3c80abc9
```

As you can see, the replacement works quite simple, the main goal is to write a correct regular expression.

## §3. Conclusions

In this topic, we've learned:

- we can replace a substring with any other string by means of `replaceFirst` and `replaceAll` methods belonging either to `String` objects or `Matcher` objects;
- string methods accept two arguments: a regular expression matching substrings that have to be replaced and a string that's going to replace them;
- `Matcher` methods accept only the second argument (the replacement string).

Report a typo

108 users liked this theory. 2 didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion