Java 11 features

Swipe Left →

Java 11 has been out for a couple of years. Yet, it is the most widely used Java version, ahead of Java 8 and Java 17. So I have gathered a few new features that you must know if you are going to use Java 11.

New String Methods

Many additional methods that make working with strings easier are introduced in Java 11.

Do you want to verify that a string contains nothing but spaces? Make use of ***isBlank()***!

Do you want the leading or trailing whitespace to go? Make use of ***stripTrailing()*** and ***stripLeading()***.

Additionally, there are functions to repeat a string a predetermined number of times (***repeat()***) and break strings into lines (***lines()***).

Example:

```
String message = "Hello World!";

if (message.isBlank()) {
  System.out.println("This string is empty!");
} else {
  System.out.println(message.stripTrailing() + "How cool is this?");
  // Output: Hello World! How cool is this?
}

String repeated = "Java";
System.out.println(repeated.repeat(3));  // Output: JavaJavaJava
```

Local-Variable Syntax for Lambdas

Lambda arguments can now be declared just like ordinary local variables, which simplifies and improves the readability of your code.

EXAMPLE :

```
Comparator<String> byLength = (var str1, var str2) ->
                                    str1.length() - str2.length();
```

HTTP Client

With Java 11, you can easily make and receive requests thanks to its built-in HTTP client. Because of this, interacting with online services and APIs from your Java code is really easy.

EXAMPLE:

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
  .uri(URI.create("https://api.example.com/data"))
  .GET()
  .build();

HttpResponse<String> response = client.send(request,
    HttpResponse.BodyHandlers.ofString());
System.out.println(response.body());
```

Reading and Writing Strings from Files (Even Easier)

File handling is streamlined in Java 11. Files.lines() may now be used to read a file line by line, and Files.writeString() can be used to write strings to a file.

EXAMPLE:

```
String fileName = "abc.txt";

try (Stream<String> lines = Files.lines(Paths.get(fileName))) {
  lines.forEach(line -> System.out.println(line));
} catch (IOException e) {
  e.printStackTrace();
}

String content = "This is some awesome content!";
Files.writeString(Paths.get(fileName), content);
```