*Ref - https://www.digitalocean.com/community/tutorials/java-11-features*

1. *New String Methods*

*Java 11 adds a few new methods to the String class: isBlank, lines, strip, stripLeading, stripTrailing, and repeat. Let’s see how we can make use of the new methods to extract non-blank, stripped lines from a multi-line string:*

**String** multilineString = "Baeldung helps \n \n developers \n explore Java.";

List<String> lines = multilineString.lines().filter(line -> !line.isBlank()) .map(String::strip) .collect(Collectors.toList()); assertThat(lines).containsExactly("Baeldung helps", "developers", "explore Java.");

*These methods can reduce the amount of boilerplate involved in manipulating string objects, and save us from having to import libraries.*

strip() is “Unicode-aware” evolution of trim(). When trim() was introduced, Unicode wasn’t evolved.

1. *New File Methods - Additionally, it’s now easier to read and write Strings from files. We can use the new readString and writeString static methods from the Files class:*

**Path** filePath = Files.writeString(Files.createTempFile(tempDir, "demo", ".txt"), "Sample text");

**String** fileContent = Files.readString(filePath); assertThat(fileContent).isEqualTo("Sample text");

1. *The java.util.Collection interface contains a new default toArray method which takes an IntFunction argument.*

*This makes it easier to create an array of the right type from a collection:*

**List** sampleList = Arrays.asList("Java", "Kotlin");

String[] sampleArray = sampleList.toArray(String[]::**new**); assertThat(sampleArray).containsExactly("Java", "Kotlin");

1. *The Not Predicate Method - A static not method has been added to the Predicate interface. We can use it to negate an existing predicate, much like the negate method:*

List<String> sampleList = Arrays.asList("Java", "\n \n", "Kotlin", " ");

**List** withoutBlanks = sampleList.stream() .filter(Predicate.not(String::isBlank)) .collect(Collectors.toList()); assertThat(withoutBlanks).containsExactly("Java", "Kotlin");

1. *Local-Variable Syntax for Lambda - Support for using the local variable syntax (var keyword) in lambda parameters was added in Java 11.*

*We can make use of this feature to apply modifiers to our local variables, like defining a type annotation.*

List<String> sampleList = Arrays.asList("Java", "Kotlin");

**String** resultString = sampleList.stream()

**.**map((@Nonnull **var** x) -> x.toUpperCase()) .collect(Collectors.joining(", ")); assertThat(resultString).isEqualTo("JAVA, KOTLIN");

1. *Running Java Files - A major change in this version is that we don’t need to compile the Java source files with javac explicitly anymore. we can directly run the file using the java command:*
2. *Performance Enhancements*

*7.1 Improved Aarch64 Intrinsics - Java 11 optimizes the existing string and array intrinsics on ARM64 or AArch64 processors. Additionally, new intrinsics are implemented for sin, cos, and log methods of java.lang.Math.*

*We use an intrinsic function like any other; however, the intrinsic function gets handled in a special way by the compiler.*

*7.2 No-Op Garbage Collector*

*A new garbage collector called Epsilon is available for use in Java 11 as an experimental feature.*

*It’s called a No-Op (no operations) because it allocates memory but does not actually collect any garbage. Thus, Epsilon is applicable for simulating out of memory errors.*

*Obviously, Epsilon won’t be suitable for a typical production Java application; however, there are a few specific use-cases where it could be useful.*