What's a Regular Expression?

A regular expression is simply text.

It may contain **characters or character combinations** that have **special meaning**.

These are called **metacharacters**.

These combinations are interpreted by a regular expression **pattern processor**.

Most patterns you'll need, have already been written, and you'll find these, with an internet search.

Patter for:	Regular Expression	Examples of Match(es)
U.S. Phone Number	`\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}`	(800) 123-4567
HTML Tag	`<(\\w+)[^>]*>([^\\v</>]*)(</\\1>)*`	`<h1>Title</h1>` `<br/>` `<h2 class="red">Hello World</h2>`

Regular Expression

They are big time-savers!

You don't have to write a lot of looping and parsing code.

You can use a regular expression to do this work, with just a couple of lines of code.

There are really good reasons to use regular expressions.

- Verify something is formatted correctly.
- Find occurrences of patterns in text.
- Replace matching occurrences of patterns in text.
- Extract matching occurrences from the text.
- Split your text by a pattern.

Ways to use Regular Expressions in Java

There are classes with methods that take regular expression strings or patterns as parameters. A few of these are:

String, Scanner, Formatter, DateTimeFormatter, Duration.

There are also special classes in the java.util.regex package, to help you implement your own functionality.

Pattern, Matcher.

String's methods which use regular expressions

They can all be used with a String literal, that doesn't have any of the special character sequences.

They become very powerful though, when you do pass regular expression patterns.

Result	Method Name
`boolean`	`matches(String regex)`
`String`	`replaceAll(String regex, String replacement)`
`String`	`replaceFirst(String regex, String replacement)`
`String[]`	`split(String regex)`
`String[]`	`split(String regex, int limit)`