**Regular Expressions in Java: Mastering Pattern Matching and Text Manipulation**

* Regular expressions are a way to describe a string or a pattern
* You have probably already used regular expressions, possibly without actually knowing it.
* Some methods in the String class, accept a regular expression as a parameter, for example, the matches method, the replaceAll and replaceFirst methods, as well as the split methods, all work with regular expressions.
* These expressions are often used to search strings for a specific pattern, or to validate data.
* As an example, you may want to verify that a user has provided an email address using a valid email address format. You can do this by checking the input against a regular expression pattern, that describes how an email address string should look.
* Regular expressions can also be very useful for parsing log files, to extract or search for specific types of information.

**What's a Regular Expression?**

* A regular expression is simply text.
* It may contain characters or character combinations that have special meaning.
* These characters are called metacharacters.
* These combinations are interpreted by a regular expression pattern processor.
* Most patterns, have already been written and we can find them or internet with a search

|  |  |  |
| --- | --- | --- |
| Pattern for: | Regular Expression | Example 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!
* We don’t always have to write a lot of looping and parsing code
* We 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 text
  + Split your text by a pattern

**Ways to use Regular Expressions in Java**

There are classes with methods that take regular expression string 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 us implement our own functionality

Patter, Matcher

**String’s methods which use regular expressions**

They can all be used with a String literal that does not have any of the special character sequences.

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

|  |  |
| --- | --- |
| Result | Method Name |
| Boolean | Matches(String regex) |
| String | replaceAll(String regex, String replacement) |
| String[] | Split(String regex) |
| String[] | Split(String regex, int limit) |

The Parts of a Regular Expression

A regular expression can be made up of combinations of the following:

* **Literals**. These have no additional meaning and are a one to one match. If you specify the literal "abc", the code will match on the first occurrence of "abc", in your string.
* **Character Classes**. Some of these are predefined, others you can define yourself. The period is an example of a predefined character class.
* **Quantifiers**. These metacharacters identify the number of occurrences of a character class or literal, required to make a match. I used the asterisk, but there are several others I'll review shortly.
* **Boundary matchers**, or anchors. These specify the position in the text. For example, at the start of the text or the end.
* **Groups**. These identify and allow for the capturing of subexpressions. More on this later.

The table on this slide displays some common metacharacters that fall into these categories.

|  |  |
| --- | --- |
| **Type** | **Examples** |
| Character Classes | **.**  **[**[**abc**](https://intrinsicservices.atlassian.net/wiki/pages/createpage.action?spaceKey=OJ1R&title=abc&linkCreation=true&fromPageId=3844505601)**] [**[**a-g**](https://intrinsicservices.atlassian.net/wiki/pages/createpage.action?spaceKey=OJ1R&title=a-z&linkCreation=true&fromPageId=3844505601)**] [**[**A-Z**](https://intrinsicservices.atlassian.net/wiki/pages/createpage.action?spaceKey=OJ1R&title=A-Z&linkCreation=true&fromPageId=3844505601)**] [**[**0-9**](https://intrinsicservices.atlassian.net/wiki/pages/createpage.action?spaceKey=OJ1R&title=0-9&linkCreation=true&fromPageId=3844505601)**] [^**[**abc**](https://intrinsicservices.atlassian.net/wiki/pages/createpage.action?spaceKey=OJ1R&title=abc&linkCreation=true&fromPageId=3844505601)**]  \d  \s \w** |
| Quantifiers | **\* + ?** |
| Boundary matchers (or anchors) | **^ $ \b** |
| Groups | **()** |

You can find these examples and more by looking at Java's Pattern Class API.

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/regex/Pattern.html>

**Character classes defined with square brackets**

Characters in square brackets may have a different meaning.

Any character except ^, -, ] or \ is a **literal**, when it's in the **square brackets**.  
As an example, a period in square brackets will represent a literal period, and not a meta character to match any character.

|  |  |
| --- | --- |
| . | Any character except ^, -, ] or \ is a **literal**, when it's in the **square brackets**. |
| [.] | Means a single period |

**Quantifiers**

There are six different quantifiers you can use in your regular expressions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Quantifier** | **Meaning** | **Pattern Example** | **Match Examples** |
| \* | pattern appears zero or more times | b\* | empty string, b, bb, bbb |
| + | pattern appears one or more times | b+ | b, bb, bbb |
| ? | pattern appears zero or one time | colou?r | color, colour |
| { n } | pattern must appear exactly n times | b{3} | bbb |
| { n, } | pattern must appear at least n times | b{2,} | bb, bbb, bbbb |
| { n, m } | pattern must appear at least n but not more than m times | b{3, 4} | bbb, bbbb |

**Boundary Matchers**

There are three common boundary matchers or anchors.

|  |  |  |  |
| --- | --- | --- | --- |
| **metacharacter** | **Meaning** | **Pattern String** | **Match Notes** |
| ^ | matches to start of text | "^." | Matches first character in a string |
| $ | matches to end of text | ".$" | Matches last character in a string |
| \b | matches to word | "\\b" | Matches first word in a string. |