# Regular Expressions - Regex

Java provides the java.util.regex package for pattern matching with regular expressions. Java regular expressions are very similar to the Perl programming language and very easy to learn.

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. They can be used to search, edit, or manipulate text and data.

The java.util.regex package primarily consists of the following three classes −

* **Pattern Class** − A Pattern object is a compiled representation of a regular expression. The Pattern class provides no public constructors. To create a pattern, you must first invoke one of its public static **compile()** methods, which will then return a Pattern object. These methods accept a regular expression as the first argument.
* **Matcher Class** − A Matcher object is the engine that interprets the pattern and performs match operations against an input string. Like the Pattern class, Matcher defines no public constructors. You obtain a Matcher object by invoking the **matcher()** method on a Pattern object.
* **PatternSyntaxException** − A PatternSyntaxException object is an unchecked exception that indicates a syntax error in a regular expression pattern.

**EXAMPLE:**

|  |  |
| --- | --- |
| **package** com.Soham;  **import** java.util.regex.\*;  **public class** Main {   **public static void** main(String[] args) {  *regix*(**"pussy"**, **"you have a small pussy bitch"**);  *regix*(**"penis"**, **"my penisis too bab for you"**);  *regix*(**"boobs"**, **"I like nipple"**);  *regix*(**"108"**, **"108.365.358.369"**);  *regix*(**"nigga"**, **"you are a dirty asshole bitch"**);  *regix*(**"-"**, **"ivy is called tipu-tipu"**);  *regix*(**"D:\\\\"**, **"D:\\\\fuck\\\\goat"**);  }   **public static void** regix(String regex, String find\_me){  **boolean** getmatch = **false**;  Pattern p = Pattern.*compile*(regex);  Matcher m = p.matcher(find\_me);   **while** (m.find()){  System.***out***.println(**"Matcher found "** + m.group() + **" at index "** + m.start() + **" for regex "** + regex + **" with "** + find\_me);  getmatch = **true**;  }   **if** (!getmatch){  System.***out***.printf(**"no match for "** + regex + **" on "** + find\_me);  }  System.***out***.println();  } } | Matcher found pussy at index 17 for regex pussy with you have a small pussy bitch  Matcher found penis at index 3 for regex penis with my penisis too bab for you  no match for boobs on I like nipple  Matcher found 108 at index 0 for regex 108 with 108.365.358.369  no match for nigga on you are a dirty asshole bitch  Matcher found - at index 18 for regex - with ivy is called tipu-tipu  Matcher found D:\ at index 0 for regex D:\\ with D:\\fuck\\goat |
| ***THIS WILL BE COUNTED AS ESCAPE CHARACTER :***  *regix*(**"D:\\"**, **"D:\\fuck\\goat"**); | D:\  ^  at java.base/java.util.regex.Pattern.error(Pattern.java:2010)  no match for nigga on you are a dirty asshole bitch  at java.base/java.util.regex.Pattern.compile(Pattern.java:1779)  Matcher found - at index 18 for regex - with ivy is called tipu-tipu  at java.base/java.util.regex.Pattern.<init>(Pattern.java:1422)  at java.base/java.util.regex.Pattern.compile(Pattern.java:1082)  at com.Soham.Main.regix(Main.java:19)  at com.Soham.Main.main(Main.java:14) |

#### **java.util.regex package**

It provides following classes and interface for regular expressions. The Matcher and Pattern classes are widely used in java regular expression.

1. MatchResult interface
2. Matcher class
3. Pattern class
4. PatternSyntaxException class

## **Matcher class**

It implements **MatchResult** interface. It is a *regex engine* i.e. used to perform match operations on a character sequence.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | boolean matches() | test whether the regular expression matches the pattern. |
| 2 | boolean find() | finds the next expression that matches the pattern. |
| 3 | boolean find(int start) | finds the next expression that matches the pattern from the given start number. |
| 4 | String group() | returns the matched subsequence. |
| 5 | int start() | returns the starting index of the matched subsequence. |
| 6 | int end() | returns the ending index of the matched subsequence. |
| 7 | int groupCount() | returns the total number of the matched subsequence. |

## **Pattern class**

It is the *compiled version of a regular expression*. It is used to define a pattern for the regex engine.

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | static Pattern compile(String regex) | compiles the given regex and return the instance of pattern. |
| 2 | Matcher matcher(CharSequence input) | creates a matcher that matches the given input with pattern. |
| 3 | static boolean matches(String regex, CharSequence input) | It works as the combination of compile and matcher methods. It compiles the regular expression and matches the given input with the pattern. |
| 4 | String[] split(CharSequence input) | splits the given input string around matches of given pattern. |
| 5 | String pattern() | returns the regex pattern. |