**Regex metacharacters Information**

Regex metacharacters are special characters in regular expressions (regex) that have specific meanings and are used to define patterns for matching strings. These characters help in creating complex search patterns.

Here’s a detailed list of regex metacharacters and their meanings:

**1. Characters for Anchors** **-** These specify positions within a string.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **^** | Matches the **start** of a string. | ^Hello matches "Hello world" only if it starts with Hello. |
| **$** | Matches the **end** of a string. | world$ matches "Hello world" only if it ends with world. |

**2. Characters for Quantifiers -** These define the number of times a pattern must occur.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **\*** | Matches **zero or more** occurrences. | a\* matches "aaa", "a", or "". |
| **+** | Matches **one or more** occurrences. | a+ matches "aaa" or "a", but not "". |
| **?** | Matches **zero or one** occurrence. | a? matches "a" or "". |
| **{n}** | Matches **exactly n** occurrences. | a{3} matches "aaa". |
| **{n,}** | Matches **at least n** occurrences. | a{2,} matches "aa", "aaa", etc. |
| **{n,m}** | Matches **between n and m** occurrences. | a{2,4} matches "aa", "aaa", or "aaaa". |

**3. Characters for Groups and Ranges -** These allow grouping patterns or specifying character ranges.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **.** | Matches **any single character** except a newline. | a.c matches "abc", "a\_c", etc. |
| **[]** | Matches **any character** within the brackets. | [aeiou] matches any vowel. |
| **[^]** | Matches **any character not** in the brackets. | [^aeiou] matches any consonant. |
| **-** | Specifies a **range** of characters. | [a-z] matches any lowercase letter. |
| **()** | Groups a subpattern. | (abc)+ matches "abc", "abcabc". |
| **`** | **`** | Acts as an OR operator. |

**4. Characters for Escaping -** Escape metacharacters to treat them as literals.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **\** | Escapes a metacharacter to match it literally. | \. matches a period (.). |

**5. Characters for Predefined Character Classes -** These represent commonly used groups of characters.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **\d** | Matches any **digit** (0–9). | \d+ matches "123". |
| **\D** | Matches any **non-digit**. | \D+ matches "abc". |
| **\w** | Matches any **word character** (alphanumeric + \_). | \w+ matches "hello\_123". |
| **\W** | Matches any **non-word character**. | \W+ matches "@#$%". |
| **\s** | Matches any **whitespace**. | \s+ matches " ", "\t", or "\n". |
| **\S** | Matches any **non-whitespace**. | \S+ matches "hello". |

**6. Characters for Assertions -** Assertions do not consume characters but assert a condition.

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| **\b** | Matches a **word boundary**. | \bcat\b matches "cat" in "a cat is here" but not "catch". |
| **\B** | Matches **not a word boundary**. | \Bcat\B matches "category". |
| **(?=...)** | Matches if the pattern is followed by ... | a(?=b) matches "a" in "ab" but not "ac". |
| **(?!...)** | Matches if the pattern is not followed by ... | a(?!b) matches "a" in "ac" but not "ab". |

**Key Components :**

1. **Input Text:**
   * String text = "Contact us at support@example.com, sales@example.org, or info@example.net.";
   * This is the input string containing email addresses to extract.
2. **Regex Pattern:**

* String regex = "[a-zA-Z0-9+\_.-]+@[a-zA-Z0-9.-]+";
* This regular expression defines the format of an email address:
* [a-zA-Z0-9+\_.-]+: Matches the username part of the email. It includes:
* Letters (a-z, A-Z), digits (0-9), and certain special characters (+, \_, ., -).
* @: Matches the @ symbol separating the username and domain.
* [a-zA-Z0-9.-]+: Matches the domain part of the email. It includes:
* Letters, digits, dots (.), and hyphens (-).

1. **Pattern and Matcher:**
   * Pattern pattern = Pattern.compile(regex);
     + Compiles the regex into a Pattern object for efficient matching.
   * Matcher matcher = pattern.matcher(text);
     + Creates a Matcher object to search the input text using the compiled pattern.
2. **Finding Matches**:
   * The while (matcher.find()) loop iterates through all matches found in the text.
   * matcher.group() retrieves the current match (email address) and prints it.

A regular expression is a special sequence of characters that helps you

to match or find the other strings or set of strings, using a specialized syntax held in a pattern. They can be used to search ,edit, or manipulate text & data.

**Package - java.util.regex**

**Pattern -** A compiled representation of a regular expression

**Matcher -** An engine that performs match operations on a character sequence by interpreting a Pattern

**PatternSyntaxException -** Unchecked exception thrown to indicate a syntax error in a regular-expression pattern.