Example of Java Regular Expressions

There are three ways to write the regex example in Java.

**import** java.util.regex.\*;

**public** **class** RegexExample1{

**public** **static** **void** main(String args[]){

//1st way

Pattern p = Pattern.compile(".s");//. represents single character

Matcher m = p.matcher("as");

**boolean** b = m.matches();

//2nd way

**boolean** b2=Pattern.compile(".s").matcher("as").matches();

//3rd way

**boolean** b3 = Pattern.matches(".s", "as");

System.out.println(b+" "+b2+" "+b3);

}}

Example2:

**import** java.util.regex.\*;

**class** RegexExample2{

**public** **static** **void** main(String args[]){

System.out.println(Pattern.matches(".s", "as"));//true (2nd char is s)

System.out.println(Pattern.matches(".s", "mk"));//false (2nd char is not s)

System.out.println(Pattern.matches(".s", "mst"));//false (has more than 2 char)

System.out.println(Pattern.matches(".s", "amms"));//false (has more than 2 char)

System.out.println(Pattern.matches("..s", "mas"));//true (3rd char is s)

}}

## **Regex Character classes**

|  |  |  |
| --- | --- | --- |
| **No.** | **Character Class** | **Description** |
| 1 | [abc] | a, b, or c (simple class) |
| 2 | [^abc] | Any character except a, b, or c (negation) |
| 3 | [a-zA-Z] | a through z or A through Z, inclusive (range) |
| 4 | [a-d[m-p]] | a through d, or m through p: [a-dm-p] (union) |
| 5 | [a-z&&[def]] | d, e, or f (intersection) |
| 6 | [a-z&&[^bc]] | a through z, except for b and c: [ad-z] (subtraction) |
| 7 | [a-z&&[^m-p]] | a through z, and not m through p: [a-lq-z](subtraction) |

## **Regular Expression Character classes Example**

1. **import** java.util.regex.\*;
2. **class** RegexExample3{
3. **public** **static** **void** main(String args[]){
4. System.out.println(Pattern.matches("[amn]", "abcd"));//false (not a or m or n)
5. System.out.println(Pattern.matches("[amn]", "a"));//true (among a or m or n)
6. System.out.println(Pattern.matches("[amn]", "ammmna"));//false (m and a comes more than once)
7. }}

## **Regex Quantifiers**

The quantifiers specify the number of occurrences of a character.

|  |  |
| --- | --- |
| **Regex** | **Description** |
| X? | X occurs once or not at all |
| X+ | X occurs once or more times |
| X\* | X occurs zero or more times |
| X{n} | X occurs n times only |
| X{n,} | X occurs n or more times |
| X{y,z} | X occurs at least y times but less than z times |

## **Regular Expression Character classes and Quantifiers Example**

1. **import** java.util.regex.\*;
2. **class** RegexExample4{
3. **public** **static** **void** main(String args[]){
4. System.out.println("? quantifier ....");
5. System.out.println(Pattern.matches("[amn]?", "a"));//true (a or m or n comes one time)
6. System.out.println(Pattern.matches("[amn]?", "aaa"));//false (a comes more than one time)
7. System.out.println(Pattern.matches("[amn]?", "aammmnn"));//false (a m and n comes more than one time)
8. System.out.println(Pattern.matches("[amn]?", "aazzta"));//false (a comes more than one time)
9. System.out.println(Pattern.matches("[amn]?", "am"));//false (a or m or n must come one time)
11. System.out.println("+ quantifier ....");
12. System.out.println(Pattern.matches("[amn]+", "a"));//true (a or m or n once or more times)
13. System.out.println(Pattern.matches("[amn]+", "aaa"));//true (a comes more than one time)
14. System.out.println(Pattern.matches("[amn]+", "aammmnn"));//true (a or m or n comes more than once)
15. System.out.println(Pattern.matches("[amn]+", "aazzta"));//false (z and t are not matching pattern)
17. System.out.println("\* quantifier ....");
18. System.out.println(Pattern.matches("[amn]\*", "ammmna"));//true (a or m or n may come zero or more times)
20. }}

## **Regex Metacharacters**

The regular expression metacharacters work as shortcodes.

|  |  |
| --- | --- |
| **Regex** | **Description** |
| . | Any character (may or may not match terminator) |
| \d | Any digits, short of [0-9] |
| \D | Any non-digit, short for [^0-9] |
| \s | Any whitespace character, short for [\t\n\x0B\f\r] |
| \S | Any non-whitespace character, short for [^\s] |
| \w | Any word character, short for [a-zA-Z\_0-9] |
| \W | Any non-word character, short for [^\w] |

## **Regular Expression Question 1**

1. /\*Create a regular expression that accepts alphanumeric characters only.
2. Its length must be six characters long only.\*/

import java.util.regex.\*;

class RegexExample3{

public static void main(String args[]){

System.out.println(Pattern.matches("\\w{6}","ram"));

System.out.println(Pattern.matches("\\w{6}","ramcha"));

System.out.println(Pattern.matches("\\w{6}","ram123"));

System.out.println(Pattern.matches("\\w{6}","123ram"));

System.out.println(Pattern.matches("\\w{6}","123456"));

System.out.println(Pattern.matches("\\w{6}","123"));

System.out.println(Pattern.matches("\\w{6}","ramCha"));

System.out.println(Pattern.matches("\\w{6}","Ram123"));

System.out.println(Pattern.matches("\\w{6}","RAMCHA"));

System.out.println(Pattern.matches("\\w{6}","RAM"));

}}

1. /\*Create a regular expression that accepts 10 digit numeric characters
2. starting with 7, 8 or 9 only.\*/

import java.util.regex.\*;

class RegexExample3{

public static void main(String args[]){

System.out.println(Pattern.matches("[789][0-9]{9}","ram"));

System.out.println(Pattern.matches("[789][0-9]{9}","012"));

System.out.println(Pattern.matches("[789][0-9]{9}","0123456789"));

System.out.println(Pattern.matches("[789][0-9]{9}","7891234567"));

System.out.println(Pattern.matches("[789][0-9]{9}","8791010100"));

System.out.println(Pattern.matches("[789][0-9]{9}","87910101001"));

}}

false  
false  
false  
true  
true  
false

## **Java Regex Finder Example**

1. **import** java.util.regex.Pattern;
2. **import** java.util.Scanner;
3. **import** java.util.regex.Matcher;
4. **public** **class** RegexExample8{
5. **public** **static** **void** main(String[] args){
6. Scanner sc=**new** Scanner(System.in);
7. **while** (**true**) {
8. System.out.println("Enter regex pattern:");
9. Pattern pattern = Pattern.compile(sc.nextLine());
10. System.out.println("Enter text:");
11. Matcher matcher = pattern.matcher(sc.nextLine());
12. **boolean** found = **false**;
13. **while** (matcher.find()) {
14. System.out.println("I found the text "+matcher.group()+" starting at index "+
15. matcher.start()+" and ending at index "+matcher.end());
16. found = **true**;
17. }
18. **if**(!found){
19. System.out.println("No match found.");
20. }
21. }
22. }
23. }

Output:

Enter regex pattern: java

Enter text: this is java, do you know java

I found the text java starting at index 8 and ending at index 12

I found the text java starting at index 26 and ending at index 30

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Main {

public static void main(String[] args) {

Pattern pattern = Pattern.compile("w3schools", Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher("Visit W3Schools!");

boolean matchFound = matcher.find();

System.out.println(Pattern.matches("[abc]+","ab"));

if(matchFound) {

System.out.println("Match found");

} else {

System.out.println("Match not found");

}

String s="ram is great, king is god. god is king";

System.out.println(s);

s=s.replaceAll("\\s+"," ");

System.out.println(s);

s=s.replaceAll("[^0-9a-zA-Z ]+","");

System.out.println(s);

}

}

o/p

true

Match found

ram is great, king is god. god is king

ram is great, king is god. god is king

ram is great king is god god is king