CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION

ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 3: String Handling, Regular Expression and Wrapper class

Chapter 3

Objective: After completing this lesson you will be able to : * Gain an understanding of Java regular expressions, regex class, matcher class and pattern class	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 120 minutes	**Practical Duration:** 0 minute
Total Duration: 120 minutes	

Chapter 3

3.1 Java Regular Expression- Regex Class

What is a Java Regular Expression?

Regular Expressions or Regex is the API used for defining String patterns to be used for locating, editing and a text. These expressions are used for defining string constraints. Regular Expressions are available within the java.util.regex package.

Example of the three methods for writing regular expressions in Java –

```
import java.util.regex.*;
public class RegexExample1{
public static void main(String args[]){
```

1st method

```
Pattern p = Pattern.compile('.s');//. represents single character
Matcher m = p.matcher('as');
boolean b = m.matches();
```

2nd method

```
boolean b2=Pattern.compile('.s').matcher('as').matches();
```

3rd method

```
boolean b3 = Pattern.matches('.s', 'as');
System.out.println(b+' '+b2+' '+b3);
}}
```

Output: true true true

Character classes of Regex

[abc] – a, b, c

[^abc] – Any character other than a, b, or c

[a-zA-Z] – a to z, or A to Z, inclusive

[a-d[m-p]] – a to d, or m to p: [a-dm-p]

[a-z&&[def]] – d, e, f

[a-z&&[^bc]] – a to z, but not b and c: [ad-z]

[a-z&&[^m-p]] – a to z, but not m to p: [a-lq-z]

Regex Quantifiers

Regex quantifiers point to the number of character occurrences –

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

Regex Metacharacters

These function as short regex codes

. – Any character

\d – Any digits (0-9)

\D – Any non-digit (^0-9)

\s – Any whitespace character (\t\n\x0B\f\r)

\S – Any non-whitespace character (^\s)

\w – Any word character (a-zA-Z_0-9)

\W – Any non-word character (^\w)

\b – A word boundary

\B – A non-word boundary

Example of regular expression –

```
import java.util.regex.*;
class RegexExample1{

public static void main(String args[]){
System.out.println(Pattern.matches('.t', 'bt'));//true (2nd char is t)  System.out.println(Pattern.matches('.t',
'pk'));//false (2nd char is not t)  System.out.println(Pattern.matches('.t', 'lot'));//false (has upwards of 2 char)
System.out.println(Pattern.matches('.t', 'boat'));//false (has upwards of 2 char)
System.out.println(Pattern.matches('..t', 'rat'));//true (3rd char is t)
}}
```

2nd **Example of regular expression –**

Regex : ^(.+)@(.+)%

```
List emails = new ArrayList();
emails.add('user@domain.com');
emails.add('user@domain.co.in');
emails.add('user1@domain.com');
emails.add('user.name@domain.com');
emails.add('user#@domain.co.in');
emails.add('user@domaincom');
emails.add('user#domain.com');
emails.add('@gmail.com');

String regex = '^(.+)@(.+)% ';

Pattern pattern = Pattern.compile(regex);

for(String email : emails){
   Matcher matcher = pattern.matcher(email);
   System.out.println(email +' : '+ matcher.matches());
}
```

Output:

```
user@domain.com :       true
user@domain.co.in :     true
user1@domain.com :      true
user.name@domain.com :  true
user#@domain.co.in :    true
user@domaincom :        true
```

user#domain.com :   false

@gmail.com :   false

Matcher class

Matcher class is an expression used to implement the MatchResult interface. The regex engine of Java performs the necessary actions to match character chain operations.

Matcher class methods

boolean matches() – tests to check if a regular expression matches a pattern

boolean find() – locates the next expression matching a pattern

boolean find(**int start**) – searches the next expression matching the pattern, starting from a specified number

String group() – returns matching sub-sequence

int start() – returns starting index of a matching sub-sequence

int end() – returns ending index of a matching sub-sequence

int groupCount() – returns the count of total matching sub-sequences

Pattern class

The pattern class refers to the compiled variant of a regular expression. This form of expressions is used for defining a regex engine pattern.

Pattern class methods

static Pattern compile(**String regex**) – performs specific regex compilations and returns the Pattern instance.

Matcher matcher(**CharSequence input**) – creates a matcher matching the input with a pattern

static boolean matches(**String regex, CharSequence input**) – operates as a combination of matcher and compile methods. It performs regular expression compilation and matching of provided input with the pattern.

String[] split(**CharSequence input**) – splits input string based on designated pattern matches

String pattern() – returns the regex pattern

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. Regular expressions in Java are also known as _______.

a) reg exp

b) regex

c) regexp

d) None of the mentioned

2. What are used for defining string constraints?

a) irregular expressions

b) core expressions

c) regular expressions

d) None of the mentioned

3. Which Java package includes regular expressions?

a) java.util.exp

b) java.util.regex

c) java.util.structure

d) None of the mentioned

4. What does the [^abc] class entail?

a) a and b, or c

b) a or b or c

c) a+b+c

d) characters other than a,b,c

5. What does the [a-zA-Z] class entail?

a) a to z, or A to Z

b) a and z, or A and Z,

c) a with z, or A with Z,

d) None of the mentioned

6. What does the X? quantifier signify?

a) X occurs once or not at all

b) X occurs once or twice

c) X occurs once then multiple times

d) None of the mentioned

7. What does the X{n,} quantifier signify?

a) X occurs n times only

b) X occurs n or more times

c) X occurs n+1 times

d) None of the mentioned

8. Which interface does the Matcher class implement?

a) MatchStrings

b) MatchResult

c) MatchClass

d) None of the mentioned

9. int start() returns the starting index of a matching ___________

a) sequence

b) sub-sequence

c) wrapper

d) None of the mentioned

10. Pattern class refers to _________ regular expression variants.

a) compiled

b) combined

c) constricted

d) None of the mentioned