Java - Regex

The Java Regex or Regular Expression is an API to define a pattern for searching or manipulating strings

The Matcher and Pattern classes provide the facility of Java regular expression

Regex Character classes	
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z](subtraction)

Regex Quantifiers	
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	
.	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]
\b	A word boundary
\B	A non word boundary

Pattern.**matches() -→ Returns Boolean**

Below are a few examples to familiarize with Regex

Pattern.matches("[amn]", "abcd") -------> false (not a or m or n)

Pattern.matches("[amn]", "a") -------> true (among a or m or n)

Pattern.matches("[amn]", "ammmna") -------> false (m and a comes more than once)

Pattern.matches("[amn]?", "a") -------> true (a or m or n comes one time)

Pattern.matches("[amn]?", "aaa") -------> false (a comes more than one time)

Pattern.matches("[amn]?", "aammmnn") -------> false (a m and n comes more than one time)

Pattern.matches("[amn]?", "aazzta") -------> false (a comes more than one time)

Pattern.matches("[amn]?", "am") -------> false (a or m or n must come one time)

Pattern.matches("[amn]+", "a") -------> true (a or m or n once or more times)

Pattern.matches("[amn]+", "aaa") -------> true (a comes more than one time)

Pattern.matches("[amn]+", "aammmnn") -------> true (a or m or n comes more than once)

Pattern.matches("[amn]+", "aazzta") -------> false (z and t are not matching pattern)

Pattern.matches("[amn]*", "ammmna") -------> true (a or m or n may come zero or more times)

Pattern.matches("\\d", "abc") -------> false (non-digit)

Pattern.matches("\\d", "1") -------> true (digit and comes once)

Pattern.matches("\\d", "4443") -------> false (digit but comes more than once)

Pattern.matches("\\d", "323abc") -------> false (digit and char)

Pattern.matches("\\D", "abc") -------> false (non-digit but comes more than once)

Pattern.matches("\\D", "1") -------> false (digit)

Pattern.matches("\\D", "4443") -------> false (digit)

Pattern.matches("\\D", "323abc") -------> false (digit and char)

Pattern.matches("\\D", "m") -------> true (non-digit and comes once

Pattern.matches("\\D*", "mak") -------> true (non-digit and may come 0 or more times)

Pattern.matches("[a-zA-Z0-9]{6}", "arun32") -------> true

Pattern.matches("[a-zA-Z0-9]{6}", "kkvarun32") -------> false (more than 6 char)

Pattern.matches("[a-zA-Z0-9]{6}", "JA2Uk2") -------> true

Pattern.matches("[a-zA-Z0-9]{6}", "arun$2") -------> false ($ is not matched)

Pattern.matches("[789]{1}[0-9]{9}", "9953038949") -------> true

Pattern.matches("[789][0-9]{9}", "9953038949") -------> true

Pattern.matches("[789][0-9]{9}", "99530389490") -------> false (11 characters)

Pattern.matches("[789][0-9]{9}", "6953038949") -------> false (starts from 6)

Pattern.matches("[789][0-9]{9}", "8853038949") -------> true

Pattern.matches("[789]{1}\\d{9}", "8853038949") -------> true

Pattern.matches("[789]{1}\\d{9}", "3853038949") -------> false (starts from 3)

PatternSyntaxException Class

While writing regular expressions, to identify and throw any syntactical errors in the pattern, PatternSyntaxException class is used. This is an unchecked exception available in java.util.regex package

constructor of PatternSyntaxException Class:

Syntax:

PatternSyntaxException(String desc, String regex, int index)

Parameters:

desc: Description of the error.

regex: The pattern which is incorrect.

index: The approximate index of the error in the regex pattern, or -1 if the index is not known.

```
try {  write the regex code here}
catch(PatternSyntaxException pse  ){
System.out.println("PatternSyntaxException: ");
        System.out.println("Description: "   + pse.getDescription());
        System.out.println("Index: " + pse.getIndex());
        System.out.println("Message: " + pse.getMessage());
        System.out.println("Pattern: "   + pse.getPattern());
        System.exit(0);
    }
```