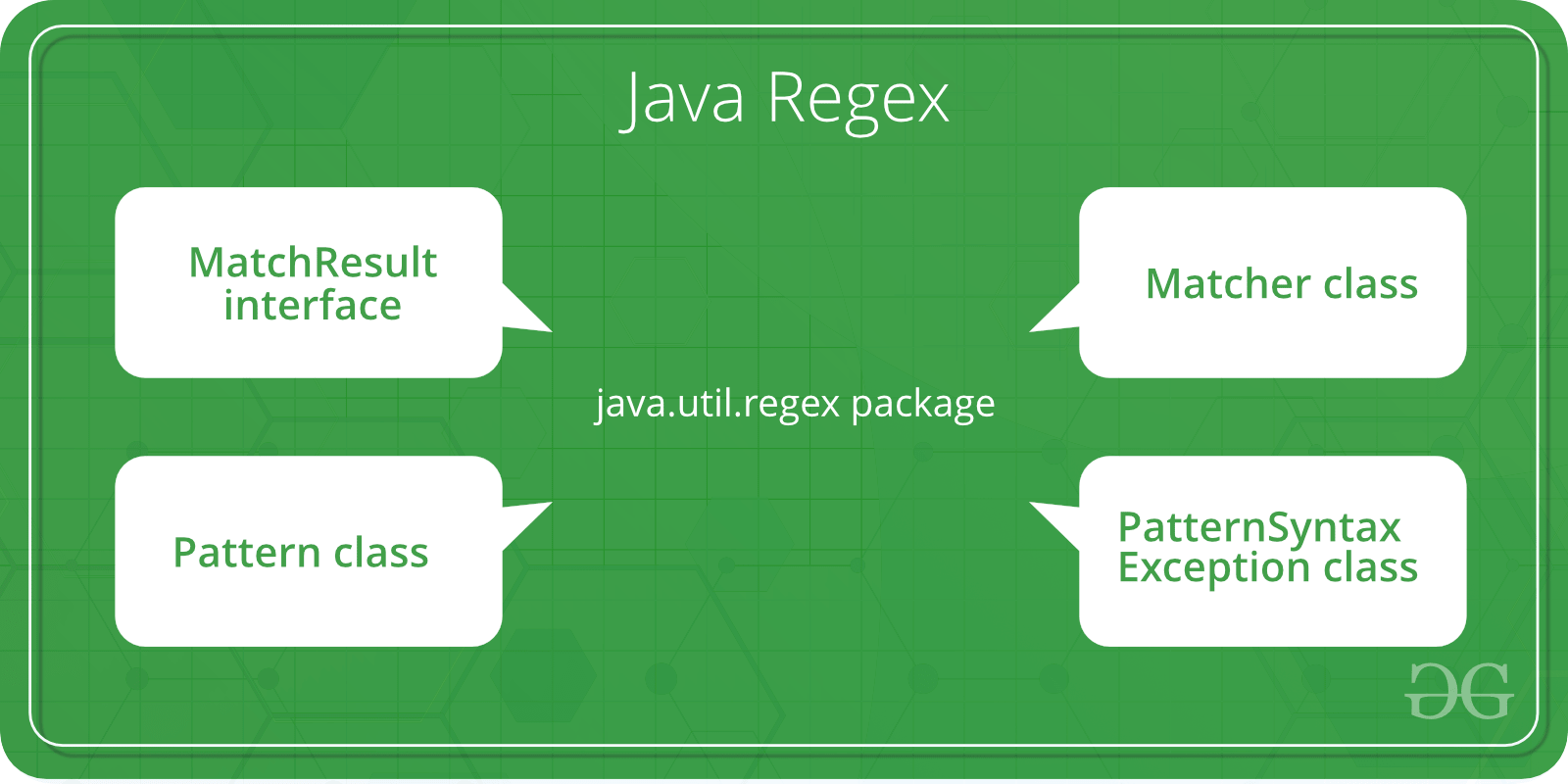
Regular Expressions in Java

Regular Expressions or Regex (in short) is an API for defining String patterns that can be used for searching, manipulating and editing a text. It is widely used to define constraint on strings such as password. Regular Expressions are provided under java.util.regex package.



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

| **CLASS** | **DESCRIPTION** |
| --- | --- |
| util.regex.Pattern | Used for defining patterns |
| util.regex.Matcher | Used for performing match operations on text using patterns |
| PatternSyntaxException | Used for indicating syntax error in a regular expression pattern |

**java.util.regex.Pattern** Class

| **MODIFIER AND TYPE** | **METHOD** | **DESCRIPTION** |
| --- | --- | --- |
| static Pattern | compile(String regex) | It is used to compile the given regular expression into a pattern. |
| static Pattern | compile(String regex, int flags) | It is used to compile the given regular expression into a pattern with the given flags. |
| int | flags() | It is used to return this pattern’s match flags. |
| Matcher | matcher(CharSequence input) | It is used to create a matcher that will match the given input against this pattern. |
| static boolean | matches(String regex, CharSequence input) | It is used to compile the given regular expression and attempts to match the given input against it. |
| String | pattern() | It is used to return the regular expression from which this pattern was compiled. |
| static String | quote(String s) | It is used to return a literal pattern String for the specified String. |
| String[] | split(CharSequence input) | It is used to split the given input sequence around matches of this pattern. |
| String[] | split(CharSequence input, int limit) | It is used to split the given input sequence around matches of this pattern. |
| String | toString() | It is used to return the string representation of this pattern. |

|  |
| --- |
| // A Simple Java program to demonstrate working of  // Pattern.matches() in Java  import java.util.regex.Pattern;    class Demo  {      public static void main(String args[])      {          // Following line prints "true" because the whole          // text "geeksforgeeks" matches pattern "geeksforge\*ks"          System.out.println (Pattern.matches("geeksforge\*ks",                                              "geeksforgeeks"));            // Following line prints "false" because the whole          // text "geeksfor" doesn't match pattern "g\*geeks\*"          System.out.println (Pattern.matches("g\*geeks\*",                                              "geeksfor"));      }  } |

**java.util.regex.Matcher** Class

| **MODIFIER AND TYPE** | **METHOD** | **DESCRIPTION** |
| --- | --- | --- |
| boolean | find() | It is mainly used for searching multiple occurrences of the regular expressions in the text. |
| boolean | find(int start) | It is used for searching occurrences of the regular expressions in the text starting from the given index. |
| int | start() | It is used for getting the start index of a match that is being found using find() method. |
| int | end() | It is used for getting the end index of a match that is being found using find() method. It returns index of character next to last matching character. |
| int | groupCount() | It is used to find the total number of the matched subsequence. |
| String | group() | It is used to find the matched subsequence. |
| boolean | matches() | It is used to test whether the regular expression matches the pattern. |

Note that Pattern.matches() checks if whole text matches with a pattern or not. Other methods (demonstrated below) are mainly used to find multiple occurrences of pattern in text.

**Java Programs to demonstrate workings of compile(), find(), start(), end() and split() :**

1. **Java Program to demonstrate simple pattern searching**

|  |
| --- |
| // A Simple Java program to demonstrate working of  // String matching in Java  import java.util.regex.Matcher;  import java.util.regex.Pattern;    class Demo  {      public static void main(String args[])      {          // Create a pattern to be searched          Pattern pattern = Pattern.compile("geeks");            // Search above pattern in "geeksforgeeks.org"          Matcher m = pattern.matcher("geeksforgeeks.org");            // Print starting and ending indexes of the pattern          // in text          while (m.find())              System.out.println("Pattern found from " + m.start() +                                 " to " + (m.end()-1));      }  } |

Output:

Pattern found from 0 to 4

Pattern found from 8 to 12

1. **Java Program to demonstrate simple regular expression searching**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // A Simple Java program to demonstrate working of  // String matching in Java  import java.util.regex.Matcher;  import java.util.regex.Pattern;    class Demo  {      public static void main(String args[])      {          // Create a pattern to be searched          Pattern pattern = Pattern.compile("ge\*");            // Search above pattern in "geeksforgeeks.org"          Matcher m = pattern.matcher("geeksforgeeks.org");            // Print starting and ending indexes of the pattern          // in text          while (m.find())              System.out.println("Pattern found from " + m.start() +                                 " to " + (m.end()-1));      }  } |

Output:

Pattern found from 0 to 2

Pattern found from 8 to 10

Pattern found from 16 to 16

1. **Java program to demonstrate Case Insensitive Searching**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // A Simple Java program to demonstrate working of  // String matching in Java  import java.util.regex.Matcher;  import java.util.regex.Pattern;    class Demo  {      public static void main(String args[])      {          // Create a pattern to be searched          Pattern pattern = Pattern.compile("ge\*", Pattern.CASE\_INSENSITIVE);            // Search above pattern in "geeksforgeeks.org"          Matcher m = pattern.matcher("GeeksforGeeks.org");            // Print starting and ending indexes of the pattern          // in text          while (m.find())              System.out.println("Pattern found from " + m.start() +                                 " to " + (m.end()-1));      }  } |

Output:

Pattern found from 0 to 2

Pattern found from 8 to 10

Pattern found from 16 to 16

1. **Java program to demonstrate working of split() to split a text based on a delimiter pattern**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // Java program to demonstrate working of splitting a text by a  // given pattern  import java.util.regex.Matcher;  import java.util.regex.Pattern;    class Demo  {      public static void main(String args[])      {          String text = "geeks1for2geeks3";            // Specifies the string pattern which is to be searched          String delimiter =  "\\d";          Pattern pattern = Pattern.compile(delimiter,                                          Pattern.CASE\_INSENSITIVE);            // Used to perform case insensitive search of the string          String[] result = pattern.split(text);              for (String temp: result)              System.out.println(temp);      }  } |

Output:

geeks

for

geeks

**MatchResult** Interface

| **MODIFIER AND TYPE** | **METHOD** | **DESCRIPTION** |
| --- | --- | --- |
| int | end() | It is used to return the offset after the last character matched. |
| int | end(int group) | It is used to return the offset after the last character of the subsequence captured by the given group during this match. |
| String | group() | It is used to return the input subsequence matched by the previous match. |
| String | group(int group) | It is used to return the input subsequence captured by the given group during the previous match operation. |
| int | groupCount() | It is used to return the number of capturing groups in this match result’s pattern. |
| int | start() | It is used to return the start index of the match. |
| int | start(int group) | It is used to return the start index of the subsequence captured by the given group during this match. |

**PatternSyntaxException** Class

| **MODIFIER AND TYPE** | **METHOD** | **DESCRIPTION** |
| --- | --- | --- |
| String | getDescription() | It is used to retrieve the description of the error. |
| int | getIndex() | It is used to retrieve the error index. |
| String | getMessage() | It is used to return a multi-line string containing the description of the syntax error and its index, the erroneous regular-expression pattern, and a visual indication of the error index within the pattern. |
| String | getPattern() | It is used to retrieve the erroneous regular-expression pattern. |

**Important Observations/Facts:**

1. We create a pattern object by calling Pattern.compile(), there is no constructor. compile() is a static method in Pattern class.
2. Like above, we create a Matcher object using matcher() on objects of Pattern class.
3. Pattern.matches() is also a static method that is used to check if given text as a whole matches pattern or not.
4. find() is used to find multiple occurrences of pattern in text.
5. We can split a text based on a delimiter pattern  
   using split()