**PATTERN CLASS (REGEX)**

The Pattern class is the primary Java class for compiling regular expressions (regex). In this tutorial I will demonstrate the methods from the Pattern class. If you have been watching my regex tutorial series thus far, then you should already be somewhat familiar with the .compile(String regex) method. The Pattern class and the Matcher class work in conjunction to perform many regex operations.  
Consider the following code:  
**Pattern p = Pattern.compile("the");**  
In the statement above, the reference variable **p** refers to a compiled Pattern object with state that represents the regex **the**. Regular expressions are by default case sensitive, so how can we create a compiled regex Pattern that will ingore case?   
**Pattern p = Pattern.compile("(the)|(THE)|(The)|(THe)|(tHe)|(tHE)|(thE)");**  
The regex above is crazy, so fortunetly we have a super easy way to create a case-insensitive regex Pattern object. The .compile(String regex, int flags) method is overloaded so we can pass flags to change the behavior of the compiled regex Pattern object.   
**Pattern p = Pattern.compile("the", Pattern.CASE\_INSENSITIVE);**  
Now if we invoke the .matcher("The squeaky wheel gets the grease.") method on that string literal the .find() method will return two results.

**Pattern p = Pattern.compile("the", Pattern.CASE\_INSENSITIVE);  
  Matcher m = p.matcher("The squeaky wheel gets the grease.");  
  System.out.println(m.find()); //true  
  System.out.println(m.find()); //true  
  System.out.println(m.find()); //false**  
  
The regex language is very flexible and there is a way to compile the same regex Pattern object using just pure regex and following will do the trick:  
**Pattern p = Pattern.compile("(?i)(the)");**  
However, sometimes it is just easier to read and understand the underlying pattern by using the parameter flags versus regex flags. Don't get me wrong, both ways are equally important to understand and you should strive learn each way. Let's get started.

|  |
| --- |
| **package** com.Soham;  **import** java.util.regex.\*;  **public class** Main {   **public static void** main(String[] args) {  System.***out***.println(**"Search string: The squeaky wheel gets the grease."**);  Matcher m = Pattern.*compile*(**"the"**, Pattern.***CASE\_INSENSITIVE***).matcher(**"The squeaky wheel gets the grease."**);  **while**(m.find()) {  System.***out***.println(**"Pattern.compile(\"the\", Pattern.CASE\_INSENSITIVE) index = "**+ m.start());  }  System.***out***.println();   *//? - operator makes the CAPTURED GROUP, case insensitive  // i - stands for case insensitive* m = Pattern.*compile*(**"(?i)(the)"**).matcher(**"The squeaky wheel gets the grease."**);  **while**(m.find()) {  System.***out***.println(**"Pattern.compile(\"(?i)(the)\") index = "**+ m.start());  }  System.***out***.println();  System.***out***.println(**"-------------------------------------------------------------------"**);   *//testing without i : in this case it gonna avoid the case sensitivity* Matcher m1 = Pattern.*compile*(**"(?)(the)"**).matcher(**"The squeaky wheel gets the grease."**);  **while**(m1.find()) {  System.***out***.println(**"Pattern.compile(\"(?)(the)\") index = "**+ m1.start());  }  System.***out***.println();  System.***out***.println(**"--------------------------------------------------------------------"**);    *//static boolean matches(String regex, CharSequence input)* **boolean** b = Pattern.*matches*(**"the"**,**"The squeaky wheel gets the grease."**);  System.***out***.println(**"Pattern.matches(\"the\",\"The squeaky wheel gets the grease.\") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   b = Pattern.*matches*(**"the"**,**"the"**);  System.***out***.println(**"Pattern.matches(\"the\",\"the\") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   b = Pattern.*matches*(**"the"**,**"the "**);  System.***out***.println(**"Pattern.matches(\"the\",\"the \") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   *//.the. - expects 5 characters, like (<anychar>)the(<anychar>) - exactly 5 chars.* b = Pattern.*matches*(**".the."**,**"The squeaky wheel gets the grease."**);  System.***out***.println(**"Pattern.matches(\".the.\",\"The squeaky wheel gets the grease.\") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   *// the. - expects exact 4 characters like the(<anything>),  //input string is : "the ", like "the<space>", exactly 4 char : returns TRUE;* b = Pattern.*matches*(**"the."**,**"the "**);  System.***out***.println(**"Pattern.matches(\"the.\",\"the \") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   *// .{1, } - allows one or more wild card characters at the BEGINNING and at the END as well  //quantifier CONCEPT* b = Pattern.*matches*(**".{1,}the.{1,}"**,**"The squeaky wheel gets the grease."**);  System.***out***.println(**"Pattern.matches(\".{1,}the.{1,}\",\"The squeaky wheel gets the grease.\") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   *// the + quantifier translates into one or more times  // we combine that with the . wildcard and we'll get our desired result .+  //WORKS SAME AS --> ".{1,}the.{1,}" -- ABOVE* b = Pattern.*matches*(**".+the.+"**,**"The squeaky wheel gets the grease."**);  System.***out***.println(**"Pattern.matches(\".+the.+\",\"The squeaky wheel gets the grease.\") = "** + b);  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println();  System.***out***.println(**"--------------------------------------------------------------------"**);   *// String pattern() --> COMPILES THE REAL PATTERN OBJECT* Pattern p = Pattern.*compile*(**".{1,}the.{1,}"**);  System.***out***.println(**"p.pattern() = "** + p.pattern());  System.***out***.println();  System.***out***.println(**"--------------------------------------------------------------------"**);   *// static String quote(String s) --> PUTS \Q AT BEGIN, AND \E AT THE END:* System.***out***.println(**"Pattern.quote(\".{1,}the.{1,}\"); = "** + Pattern.*quote*(**".{1,}the.{1,}"**));  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"Pattern.quote(\"20{4}\"); = "** + Pattern.*quote*(**"20{4}"**));  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"Pattern.quote(\"2\\d{4}\"); = "** + Pattern.*quote*(**"2\\d{4}"**));  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println();   *// String[] split(CharSequence input)* p = Pattern.*compile*(**"the"**, Pattern.***CASE\_INSENSITIVE***);  String sArray[] = p.split(**"The squeaky wheel gets the grease."**);  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"p = Pattern.compile(\"the\", Pattern.CASE\_INSENSITIVE);"**);  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"String sArray[] = p.split(\"The squeaky wheel gets the grease.\")"**);  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"sArray.length = "** + sArray.**length**);  System.***out***.println(**"--------------------------------------------------------------------"**);   *//RETURNS EVERYTHING UNTILL THE PATTERN MATCH IS FOUND* **for**(String s: sArray) {  System.***out***.println(**"sArray contains:"**+s);  }  **for**(String s: sArray) {  System.***out***.print(s);  }  System.***out***.println();  System.***out***.println();   *// int flags()* p = Pattern.*compile*(**"the"**, Pattern.***CASE\_INSENSITIVE***);  System.***out***.println(**"p.flags() = "**+ p.flags());  System.***out***.println(**"--------------------------------------------------------------------"**);   System.***out***.println(**"Pattern.CASE\_INSENSITIVE = "** + Pattern.***CASE\_INSENSITIVE***);  } } |

|  |
| --- |
| Search string: The squeaky wheel gets the grease.  Pattern.compile("the", Pattern.CASE\_INSENSITIVE) index = 0  Pattern.compile("the", Pattern.CASE\_INSENSITIVE) index = 23  Pattern.compile("(?i)(the)") index = 0  Pattern.compile("(?i)(the)") index = 23  -------------------------------------------------------------------  Pattern.compile("(?)(the)") index = 23  --------------------------------------------------------------------  Pattern.matches("the","The squeaky wheel gets the grease.") = false  --------------------------------------------------------------------  Pattern.matches("the","the") = true  --------------------------------------------------------------------  Pattern.matches("the","the ") = false  --------------------------------------------------------------------  Pattern.matches(".the.","The squeaky wheel gets the grease.") = false  --------------------------------------------------------------------  Pattern.matches("the.","the ") = true  --------------------------------------------------------------------  Pattern.matches(".{1,}the.{1,}","The squeaky wheel gets the grease.") = true  --------------------------------------------------------------------  Pattern.matches(".+the.+","The squeaky wheel gets the grease.") = true  --------------------------------------------------------------------  --------------------------------------------------------------------  p.pattern() = .{1,}the.{1,}  --------------------------------------------------------------------  Pattern.quote(".{1,}the.{1,}"); = \Q.{1,}the.{1,}\E  --------------------------------------------------------------------  Pattern.quote("20{4}"); = \Q20{4}\E  --------------------------------------------------------------------  Pattern.quote("2\d{4}"); = \Q2\d{4}\E  --------------------------------------------------------------------  --------------------------------------------------------------------  p = Pattern.compile("the", Pattern.CASE\_INSENSITIVE);  --------------------------------------------------------------------  String sArray[] = p.split("The squeaky wheel gets the grease.")  --------------------------------------------------------------------  sArray.length = 3  --------------------------------------------------------------------  sArray contains:  sArray contains: squeaky wheel gets  sArray contains: grease.  squeaky wheel gets grease.  p.flags() = 2  --------------------------------------------------------------------  Pattern.CASE\_INSENSITIVE = 2 |