The Java™ Tutorials

Trail: Essential Classes
Lesson: Regular Expressions

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

Methods of the PatternSyntaxException Class

A `PatternSyntaxException` is an unchecked exception that indicates a syntax error in a regular expression pattern. The `PatternSyntaxException` class provides the following methods to help you determine what went wrong:

- `public String getDescription()`: Retrieves the description of the error.
- `public int getIndex()`: Retrieves the error index.
- `public String getPattern()`: Retrieves the erroneous regular expression pattern.
- `public String getMessage()`: Returns 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.

The following source code, `RegexTestHarness2.java`, updates our test harness to check for malformed regular expressions:

```
import java.io.Console;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.regex.PatternSyntaxException;

public class RegexTestHarness2 {

    public static void main(String[] args){
        Pattern pattern = null;
        Matcher matcher = null;

        Console console = System.console();
        if (console == null) {
            System.err.println("No console.");
            System.exit(1);
        }
        while (true) {
            try{
                pattern =
                Pattern.compile(console.readLine("%nEnter your regex: "));

                matcher =
                pattern.matcher(console.readLine("Enter input string to search: "));
            }
            catch(PatternSyntaxException pse){
                console.format("There is a problem" +
                               " with the regular expression!%n");
                console.format("The pattern in question is: %s%n",
                               pse.getPattern());
                console.format("The description is: %s%n",
                               pse.getDescription());
                console.format("The message is: %s%n",
                               pse.getMessage());
                console.format("The index is: %s%n",
                               pse.getIndex());
                System.exit(0);
            }
            boolean found = false;
            while (matcher.find()) {
                console.format("I found the text" +
                    " \"%s\" starting at " +
                    "index %d and ending at index %d.%n",
```

```
                    matcher.group(),
                    matcher.start(),
                    matcher.end());
                found = true;
            }
            if(!found){
                console.format("No match found.%n");
            }
        }
    }
}
```

To run this test, enter `?i)foo` as the regular expression. This mistake is a common scenario in which the programmer has forgotten the opening parenthesis in the embedded flag expression `(?i)`. Doing so will produce the following results:

```
Enter your regex: ?i)
There is a problem with the regular expression!
The pattern in question is: ?i)
The description is: Dangling meta character '?'
The message is: Dangling meta character '?' near index 0
?i)
^
The index is: 0
```

From this output, we can see that the syntax error is a dangling metacharacter (the question mark) at index 0. A missing opening parenthesis is the culprit.

Previous page: Methods of the Matcher Class
Next page: Unicode Support