

Regex Support in the Pattern and Matcher Classes



Victor Grazi

ORACLE JAVA CHAMPION, SPEAKER AND GEEK

@vgrazi



Introduction



- **Pattern class flags and equivalent Regex flags**
- **Pattern class methods**
- **Unicode support**
- **Matcher class methods**



Pattern Class Flag	Value	Explanation	Regex Flag	Performance Penalty
<u>CANON_EQ</u>	128	Enables canonical equivalence	None	Yes
<u>CASE_INSENSITIVE</u>	2	Enables case-insensitive matching	(?i)	Yes
<u>COMMENTS</u>	4	Permits whitespace and comments in pattern	(?x)	
<u>DOTALL</u>	32	Enables dotall (single-line) mode	(?s)	
<u>LITERAL</u>	16	Enables literal parsing of the pattern	None	
<u>MULTILINE</u>	8	Enables multiline mode	(?m)	
<u>UNICODE_CASE</u>	64	Enables Unicode-aware case folding	(?u)	Yes
<u>UNICODE_CHARACTER_CLASS</u>	256	Enables the Unicode version of <i>Predefined character classes</i> and <i>POSIX character classes</i>	(?U)	Yes
<u>UNIX_LINES</u>	1	Enables Unix lines mode	(?d)	



Pattern Compile

```
Pattern.compile("my-regex",  
    Pattern.CASE_INSENSITIVE| Pattern.DOTALL)
```



Pattern Compile

```
Pattern pattern = Pattern.compile("abc((?ism)DEF)ghi");
```


```
Pattern pattern = Pattern.compile("abcDEFghi",  
    Pattern.CASE_INSENSITIVE |  
    Pattern.DOTALL |  
    Pattern.MULTILINE);
```



`\w`

`[A-Za-z0-9_]`



Syntax	Meaning	Explanation
(?<=regex)	Look-behind	Asserts that the regex pattern immediately precedes the current capture position
(?<!regex)	Negative look-behind	Asserts that the regex pattern does not immediately precede the current capture position
(?=regex)	Look-ahead	Asserts that the regex pattern immediately follows the current capture position
(?!regex)	Negative look-ahead	Asserts that the regex pattern does not immediately follow the current capture position
		

Replace, using Look-arounds

Replace “cane” with “bar”...
but only if it follows “candy ”

Sugar cane, candy **cane** consumes

anchors

(?<=candy)cane

“Sugar cane, candy cane”

.replaceAll("(?<=candy)cane","bar")

➡ Sugar cane, candy bar



Multiple look-arounds in sequence

- Legal! (Since they just assert a position)
- Must not contradict...
- or no match is found...
- (No exception thrown)



Extracting Matched Results

```
Pattern pattern = Pattern.compile("(\\w+
```



Summary



- Pattern class flags and equivalent Regex flags
- Pattern class methods
- `Matcher.matches()`, `find()`, `lookingAt()`
- `Matcher.group()`, `region()`, `useTransparentBounds()`...
- `Matcher.usePattern()`