<([{\^-=$!|]})?\*+.>

9/13/2016

|  |  |
| --- | --- |
| Construct | Description |
| [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) |

|  |  |
| --- | --- |
| Construct | Description |
| . | Any character (may or may not match line terminators) |
| \d | A digit: [0-9] |
| \D | A non-digit: [^0-9] |
| \s | A whitespace character: [ \t\n\x0B\f\r] |
| \S | A non-whitespace character:[^\s] |
| \w | A word character: [a-zA-Z\_0-9] |
| \W | A non-word character: [^\w] |

9/14/2016

|  |  |  |  |
| --- | --- | --- | --- |
| Greedy | Reluctant | Possessive | Meaning |
| X? | X?? | X?+ | X, once or not at all |
| X\* | X\*? | X\*+ | X, zero or more times |
| X+ | X+? | X++ | X, one or more times |
| X{n} | X{n}? | X{n}+ | X, exactly n times |
| X{n,} | X{n,}? | X{n,}+ | X, at least n times |
| X{n,m} | X{n,m}? | X{n,m}+ | X, at least n but not more than m times |

## 9/16/2016

|  |  |
| --- | --- |
| Boundary Construct | Description |
| ^ | The beginning of a line |
| $ | The end of a line |
| \b | A word boundary |
| \B | A non-word boundary |
| \A | The beginning of the input |
| \G | The end of the previous match |
| \Z | The end of the input but for the final terminator, if any |
| \z | The end of the input |

## 9/22/2016

To compile a pattern with multiple flags, separate the flags to be included using the bitwise OR operator "|".

pattern = Pattern.compile("[az]$", Pattern.MULTILINE | Pattern.UNIX\_LINES);

final int flags = Pattern.CASE\_INSENSITIVE | Pattern.UNICODE\_CASE;

Pattern pattern = Pattern.compile("aa", flags);

## 9/24/2016

|  |  |
| --- | --- |
| **Constant** | **Equivalent Embedded Flag Expression** |
| Pattern.CANON\_EQ | None |
| Pattern.CASE\_INSENSITIVE | (?i) |
| Pattern.COMMENTS | (?x) |
| Pattern.MULTILINE | (?m) |
| Pattern.DOTALL | (?s) |
| Pattern.LITERAL | None |
| Pattern.UNICODE\_CASE | (?u) |
| Pattern.UNIX\_LINES | (?d) |

## 10/5/2016

**How do you force a metacharacter to act like an ordinary character?**

Precede the metacharacter with a backslash (\);

Enclose the metacharacter within the quote expressions, \Q (at the beginning) and \E (at the end).

## Backreferences

backreference is specified in the regular expression as a backslash (\) followed by a digit indicating the number of the group to be recalled. For example, the expression (\d\d) defines one capturing group matching two digits in a row, which can be recalled later in the expression via the backreference \1.

## Numbering

s described in the Pattern API, capturing groups are numbered by counting their opening parentheses from left to right. In the expression ((A)(B(C))), for example, there are four such groups:

1. ((A)(B(C)))
2. (A)
3. (B(C))
4. (C)

## Unknown Problem

\b

\B