Pattern Matching and Regular Expressions

Go over the material in the other textbook, 4th edition. The programs from this other textbook are below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Checks whether the input string is a valid identifier.

File: Ch9MatchJavaIdentifier.java

\*/

import javax.swing.\*;

class Ch9MatchJavaIdentifier {

private static final String STOP = "STOP";

private static final String VALID = "Valid Java identifier";

private static final String INVALID = "Not a valid Java identifier";

private static final String VALID\_IDENTIFIER\_PATTERN

= "[a-zA-Z][a-zA-Z0-9\_$]\*";

public static void main (String[] args) {

String str, reply;

while (true) {

str = JOptionPane.showInputDialog(null, "Identifier:");

if (str.equals(STOP)) break;

if (str.matches(VALID\_IDENTIFIER\_PATTERN)) {

reply = VALID;

} else {

reply = INVALID;

}

JOptionPane.showMessageDialog(null,

str + ":\n" + reply);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Checks whether the input string conforms to the phone number

pattern xxx-xxx-xxxx.

\*/

import javax.swing.\*;

class Ch9MatchPhoneNumber {

private static final String STOP = "0";

private static final String VALID = "Valid phone number";

private static final String INVALID = "Not a valid phone number";

private static final String VALID\_PHONE\_PATTERN

= "[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]";

// = "[0-9]{3}-[0-9]{3}-[0-9]{4}";

// = "\\d{3}-\\d{3}-\\d{4}";

public static void main (String[] args) {

String phoneStr, reply;

while (true) {

phoneStr = JOptionPane.showInputDialog(null, "Phone#:");

if (phoneStr.equals(STOP)) break;

if (phoneStr.matches(VALID\_PHONE\_PATTERN)) {

reply = VALID;

} else {

reply = INVALID;

}

JOptionPane.showMessageDialog(null,

phoneStr + ":\n" + reply);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The following is from the Pattern class, from http://java.sun.com/javase/6/docs/api/

**Summary of regular-expression constructs**

|  |  |
| --- | --- |
| **Construct** | **Matches** |
|  |  |
| **Characters** | |
| *x* | The character *x* |
| \\ | The backslash character |
| \0*n* | The character with octal value 0*n* (0 <= *n* <= 7) |
| \0*nn* | The character with octal value 0*nn* (0 <= *n* <= 7) |
| \0*mnn* | The character with octal value 0*mnn* (0 <= *m* <= 3, 0 <= *n* <= 7) |
| \x*hh* | The character with hexadecimal value 0x*hh* |
| \u*hhhh* | The character with hexadecimal value 0x*hhhh* |
| \t | The tab character ('\u0009') |
| \n | The newline (line feed) character ('\u000A') |
| \r | The carriage-return character ('\u000D') |
| \f | The form-feed character ('\u000C') |
| \a | The alert (bell) character ('\u0007') |
| \e | The escape character ('\u001B') |
| \c*x* | The control character corresponding to *x* |
|  |  |
| **Character classes** | |
| [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) |
|  |  |
| **Predefined character classes** | |
| . | Any character (may or may not match [line terminators](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html" \l "lt)) |
| \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] |
|  |  |
| **POSIX character classes (US-ASCII only)** | |
| \p{Lower} | A lower-case alphabetic character: [a-z] |
| \p{Upper} | An upper-case alphabetic character:[A-Z] |
| \p{ASCII} | All ASCII:[\x00-\x7F] |
| \p{Alpha} | An alphabetic character:[\p{Lower}\p{Upper}] |
| \p{Digit} | A decimal digit: [0-9] |
| \p{Alnum} | An alphanumeric character:[\p{Alpha}\p{Digit}] |
| \p{Punct} | Punctuation: One of !"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~ |
| \p{Graph} | A visible character: [\p{Alnum}\p{Punct}] |
| \p{Print} | A printable character: [\p{Graph}\x20] |
| \p{Blank} | A space or a tab: [ \t] |
| \p{Cntrl} | A control character: [\x00-\x1F\x7F] |
| \p{XDigit} | A hexadecimal digit: [0-9a-fA-F] |
| \p{Space} | A whitespace character: [ \t\n\x0B\f\r] |
|  |  |
| **java.lang.Character classes (simple** [**java character type**](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#jcc)**)** | |
| \p{javaLowerCase} | Equivalent to java.lang.Character.isLowerCase() |
| \p{javaUpperCase} | Equivalent to java.lang.Character.isUpperCase() |
| \p{javaWhitespace} | Equivalent to java.lang.Character.isWhitespace() |
| \p{javaMirrored} | Equivalent to java.lang.Character.isMirrored() |
|  |  |
| **Classes for Unicode blocks and categories** | |
| \p{InGreek} | A character in the Greek block (simple [block](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#ubc)) |
| \p{Lu} | An uppercase letter (simple [category](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#ubc)) |
| \p{Sc} | A currency symbol |
| \P{InGreek} | Any character except one in the Greek block (negation) |
| [\p{L}&&[^\p{Lu}]] | Any letter except an uppercase letter (subtraction) |
|  |  |
| **Boundary matchers** | |
| ^ | 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](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#lt), if any |
| \z | The end of the input |
|  |  |
| **Greedy quantifiers** | |
| *X*? | *X*, once or not at all |
| *X*\* | *X*, zero or more times |
| *X*+ | *X*, one or more times |
| *X*{*n*} | *X*, exactly *n* times |
| *X*{*n*,} | *X*, at least *n* times |
| *X*{*n*,*m*} | *X*, at least *n* but not more than *m* times |
|  |  |
| **Reluctant quantifiers** | |
| *X*?? | *X*, once or not at all |
| *X*\*? | *X*, zero or more times |
| *X*+? | *X*, one or more times |
| *X*{*n*}? | *X*, exactly *n* times |
| *X*{*n*,}? | *X*, at least *n* times |
| *X*{*n*,*m*}? | *X*, at least *n* but not more than *m* times |
|  |  |
| **Possessive quantifiers** | |
| *X*?+ | *X*, once or not at all |
| *X*\*+ | *X*, zero or more times |
| *X*++ | *X*, one or more times |
| *X*{*n*}+ | *X*, exactly *n* times |
| *X*{*n*,}+ | *X*, at least *n* times |
| *X*{*n*,*m*}+ | *X*, at least *n* but not more than *m* times |
|  |  |
| **Logical operators** | |
| *XY* | *X* followed by *Y* |
| *X*|*Y* | Either *X* or *Y* |
| (*X*) | X, as a [capturing group](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#cg) |
|  |  |
| **Back references** | |
| \*n* | Whatever the *n*th [capturing group](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#cg) matched |
|  |  |
| **Quotation** | |
| \ | Nothing, but quotes the following character |
| \Q | Nothing, but quotes all characters until \E |
| \E | Nothing, but ends quoting started by \Q |
|  |  |
| **Special constructs (non-capturing)** | |
| (?:*X*) | *X*, as a non-capturing group |
| (?idmsux-idmsux) | Nothing, but turns match flags [i](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#CASE_INSENSITIVE) [d](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#UNIX_LINES) [m](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#MULTILINE) [s](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#DOTALL) [u](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#UNICODE_CASE) [x](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#COMMENTS) on - off |
| (?idmsux-idmsux:*X*) | *X*, as a [non-capturing group](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#cg) with the given flags [i](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#CASE_INSENSITIVE) [d](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#UNIX_LINES) [m](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#MULTILINE) [s](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#DOTALL) [u](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#UNICODE_CASE) [x](http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html#COMMENTS) on - off |
| (?=*X*) | *X*, via zero-width positive lookahead |
| (?!*X*) | *X*, via zero-width negative lookahead |
| (?<=*X*) | *X*, via zero-width positive lookbehind |
| (?<!*X*) | *X*, via zero-width negative lookbehind |
| (?>*X*) | *X*, as an independent, non-capturing group |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Some more programs that use the Pattern and Matcher classes.

/\*

Checks whether the input string is a valid identifier. This version

uses the Matcher and Pattern classes.

\*/

import javax.swing.\*;

import java.util.regex.\*;

class Ch9MatchJavaIdentifier2 {

private static final String STOP = "STOP";

private static final String VALID = "Valid Java identifier";

private static final String INVALID = "Not a valid Java identifier";

private static final String VALID\_IDENTIFIER\_PATTERN

= "[a-zA-Z][a-zA-Z0-9\_$]\*";

public static void main (String[] args) {

String str, reply;

Matcher matcher;

Pattern pattern

= Pattern.compile(VALID\_IDENTIFIER\_PATTERN);

while (true) {

str = JOptionPane.showInputDialog(null, "Identifier:");

if (str.equals(STOP)) break;

matcher = pattern.matcher(str);

if (matcher.matches()) {

reply = VALID;

} else {

reply = INVALID;

}

JOptionPane.showMessageDialog(null, str + ":\n" + reply);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Count the number of times the word 'java' occurs

in input using pattern matching technique.

The program terminates when the word STOP (case-sensitive)

is entered.

\*/

import javax.swing.\*;

import java.util.regex.\*;

class Ch9PMCountJava {

public static void main (String[] args) {

String document;

int javaCount;

Matcher matcher;

Pattern pattern = Pattern.compile("java",

Pattern.CASE\_INSENSITIVE);

document = JOptionPane.showInputDialog(null, "Sentence:");

javaCount = 0;

matcher = pattern.matcher(document);

while (matcher.find()) {

javaCount++;

}

JOptionPane.showMessageDialog(null,

"The word 'java' occurred " +

javaCount + " times.");

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Displays the positions the word 'java' occurs

in a given string using pattern matching technique.

The program terminates when the word STOP (case-sensitive)

is entered.

\*/

import javax.swing.\*;

import java.util.regex.\*;

class Ch9PMCountJava2 {

public static void main (String[] args) {

String document;

int javaCount;

Matcher matcher;

Pattern pattern = Pattern.compile("java",

Pattern.CASE\_INSENSITIVE);

document = JOptionPane.showInputDialog(null, "Sentence:");

javaCount = 0;

matcher = pattern.matcher(document);

while (matcher.find()) {

System.out.println(document.substring(matcher.start(),

matcher.end())

+ " found at position "

+ matcher.start());

}

}

}