Java Regular Expressions

Tutorial: http://docs.oracle.com/javase/tutorial/essential/regex/

```
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexTester {
        public static void main(String[] args){
                Scanner in = new Scanner(System.in);
                while (true) {
                        System.out.print("Regular expression (or Enter to quit): ");
                        String line = in.nextLine();
                        if (line.equals("")) break;
                        Pattern pattern = Pattern.compile(line);
                        System.out.println("Enter lines to search for this regular expression, or press
Enter to try another.");
                        while (true) {
                                line = in.nextLine();
                                if (line.equals("")) break;
                                Matcher matcher = pattern.matcher(line);
                                boolean found = false;
                                while (matcher.find()) {
                                        System.out.printf("I found \"%s\" from index %d to %d.\n",
                                                        matcher.group(), matcher.start(), matcher.end());
                                        found = true;
                                }
                                if (!found)
                                        System.out.println("No match found.");
                        }
                }
        }
}
```

(Based on: http://docs.oracle.com/javase/tutorial/essential/regex/examples/RegexTestHarness.java)

Basic concepts: http://en.wikipedia.org/wiki/Regular_expression#Basic_concepts

- Alphabet of symbols
- Concatenation (this then that)
- Decisions (this or that)
- Grouping (parenthesis)
- Looping, i.e. Quantification (e.g., 0 or 1, 0 or more, 1 or more)

Alphabet of symbols

- Most characters are interpreted as-is.
- Characters <([{\^-=$!|]})?*+.> are “metacharacters”. These may be escaped with a backslash (\).
- Character classes: (from http://docs.oracle.com/javase/tutorial/essential/regex/char_classes.html)

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)

- Predefined character classes: (from http://docs.oracle.com/javase/tutorial/essential/regex/pre_char_classes.html

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]`

- Quantifiers: (from http://docs.oracle.com/javase/tutorial/essential/regex/quant.html)

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

- *Capturing groups* are defined by parentheses, and are numbered by left-to-right order of left parenthesis in the pattern. The leftmost parenthesis starts group 1 (denoted \1 when backreferencing a matched capturing group). Group 0 is the entire string matched.

- Boundary matchers: (from http://docs.oracle.com/javase/tutorial/essential/regex/bounds.html)

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