Plan

• Today
  • Servers
  • Regular expressions
  • Threads
  • Interview with Josh Bloch

• rest of week
  • more concurrency, web server project
For Wednesday

• learn and experiment with regular expressions, web server examples, threads
  • links/resources will be posted
  • demo code posted to examples/ServerDemo
# Regular expressions

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>X*</td>
<td>Zero or more X’s</td>
</tr>
<tr>
<td>X+</td>
<td>One or more X’s</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>XY</td>
<td>X followed by Y</td>
</tr>
<tr>
<td>(X)</td>
<td>X (indicates capturing group, and useful for precedence)</td>
</tr>
<tr>
<td>.</td>
<td>Any one character</td>
</tr>
<tr>
<td>^</td>
<td>The beginning of a line</td>
</tr>
<tr>
<td>$</td>
<td>The end of a line</td>
</tr>
<tr>
<td>[A-Za-z]</td>
<td>Any ASCII letter</td>
</tr>
<tr>
<td>[^0]</td>
<td>Any character other than 0</td>
</tr>
</tbody>
</table>
Using regular expressions

Pattern p = Pattern.compile("[0-9]+"webdriver);

while (true) {
    String s = in.readLine();
    if (s == null) break;
    Matcher m = p.matcher(s);
    while (m.find())
        System.out.println(" -> " + m.group());
}


Looking for matches

- Matcher.matches tries to match Pattern against entire String
- Matcher.find tries to find the next occurrence of the pattern in the String
- keeps track of where it has been, can be applied iteratively
Capturing groups

• Each group of parentheses defines a capturing group

  • numbered started at 1, depending on position of left parentheses

• After a match, m.group(i) returns the portion of the string matched by the i’th capturing group

• m.group() matches the entire string that matched the pattern