A couple of people had problems getting Eclipse to recognize/find JUnit
  – I suspect all of those people were using Eclipse 3.2
  – I’ll switch to a way of using JUnit that is compatible with Eclipse 3.2
  – but you should probably upgrade

Submission from inside Eclipse is simpler than web submission
Office hours, initial pass

- David An
  - Tuesday, 1:30pm - 2:30pm
  - Thursday, 10:45am - noon
- Bill Pugh
  - Monday, 1:30pm - 3pm
  - Friday, 10:30am - noon
- Plus by appointment
- Plus additional office hours in the days before an assignment is due
Still need a note taker

• We still need a volunteer to help another students with taking notes for the course

  – The campus (via Disability student services) requests your help and will pay you.
Project 1

• Complete implementation of a web server
• Use a simplification of Servlets
  – Allowing Java code to generate responses, rather than just serving up static web pages
• Use reflection to map request strings to servlets
• Worry about security
• Use custom mocking framework for server sockets
• Due Wednesday, Sept 19th
  – posted later today
## 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>
Pattern p = Pattern.compile("[0-9]+");
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
Implementing a simple web server

• Before we dive into project 1, we are going to implement a simple web server