ECE 122

Feb. 15, 2005
for Repetition Statement

• for (int i=0; i<5; i++)
  System.out.println(i);
• Generalization
  for(initialization; loopContinuationCondition; increment)
    statement;
• Better use integer as counter
• Demo for statement
Common Programming Error

- Floating-point value may be approximate. Controlling loops (counter) with floating point value may result in imprecise counter values and inaccurate termination error.
- Using an incorrect relational operator or an incorrect counter final value may result in an off-by-one error.
- Infinite loops occur when the loop continuation condition never becomes false.
break statement (revisit)

- Break statement can be used in while, do..while, for statement, or switch statement. It causes immediate exit from that statement.
- Execution continues with the first statement after the control statement.
- Can be used to exit early from a loop.
continue statement

• The continue statement can be executed within while, do..while, and for statement.
• It skips the remaining statements in the loop body and proceeds with the next iteration of the loop.
• In while, and do..while statements, the program evaluates the loop-continuation condition immediately after the continue statement.
• In a for statement, the increment expression executes, then the program evaluates the loop continuation condition.
Logical Operators

• Can be used to form more complex conditions by combining simple conditions.
  •   && conditional AND
  •   || conditional OR
  •   & boolean logical AND
  •   | boolean logical OR
  •   ^ boolean logical exclusive OR
  •   ! Logical NOT
Conditional AND (&&)

- If \((I \gt 5 \&\& j \gt 5)\)

  System.out.println("Both i and j are greater than 5.");

<table>
<thead>
<tr>
<th>expression1</th>
<th>expression2</th>
<th>&amp;&amp;</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Conditional OR (||)

- If \((I > 5 \text{ || } j > 5)\)
  System.out.println("Either i is greater than 5 or j is greater than 5");

| Expression 1 | Expression 2 | \(||\) |
|--------------|--------------|--------|
| false        | false        | false  |
| true         | false        | true   |
| false        | true         | true   |
| true         | true         | true   |
Short-circuit evaluation

- The parts of the expression containing && or || will be evaluated only until it is known whether it is true or false. Once this is known, other parts of the expression may not be evaluated (short_circuited).
- \((i>1) \&\& (j > 1)\)

If \(i>1\) is false, then the condition is false no matter what \(j\) is. The second part \((j>1)\) will not be evaluated at all.
boolean logical AND, OR

• Boolean logical AND: \&
e.g. (i>1) \& (j>1)
• Boolean logical OR: |
e.g. (i>1) | (j>1)
• Also called bitwise AND, OR
• Work identically to their conditional logical counterpart, && and ||, except, they don’t have short-circuit behavior, I.e. both parts will be evaluated at all times.
• Demo comparison of && and &
Boolean logical exclusive OR

• \(^\)

e.g. \((i > 1) \land (j > 1)\)

• The condition evaluates to true only if two parts are different. i.e. one is true, the other is false.

• The condition evaluates to false if two parts are of the same (both are true, or false).
Boolean logical exclusive OR

<table>
<thead>
<tr>
<th>Expression 1</th>
<th>Expression 2</th>
<th>^</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
## Logical NOT

<table>
<thead>
<tr>
<th>expression</th>
<th>!expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Assignment

• Practice ForStatement.java