Evaluating boolean Expressions

- The simple numeric comparison operators and equality relations are easy to evaluate.
- To determine whether a more complex logical formula is true or false, we use truth tables for logical operators.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P &amp;&amp; Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Evaluating boolean Expressions

- Using the basic truth tables, and our precedence rules, we can evaluate any complex logical expression we like...

| P | Q | !P | P || Q |
|---|---|----|------|
| T | T | F  | T    |
| T | F | T  | T    |
| F | T | T  | T    |
| F | F | F  | F    |
Evaluating boolean Expressions

For example, suppose we have variables:
\[ x = -3, \quad y = 0 \]

Then, we can evaluate the following:
\[ (x \leq y) \land !((y > 10) \lor y = 1) \]

Some Legal and Illegal Expressions

```java
int i1 = 3;
int i2 = 4;
char c = 'e';

if (i1 != i2) {
    ...
} else {
    ...
}

if (!(i1 != i2)) {
    ...
} else {
    ...
}

if ((i1 < i2) && (i2 >= 4)) {
    ...
} else {
    ...
}

if ((c == 'a') || (c == 'e')) {
    ...
} else {
    ...
}

if (i1 <= i2) {
    ...
} else {
    ...
}

if (0 <= i1 <= 5) {
    ...
} else {
    ...
}
```

Some Exercises

Suppose `num1` and `num2` are integer variables:

1. Write code that prints out the sum of the variables if they are both less than 7.

2. Write code that prints out the sum of the variables if `num1` is less than 7, but `num2` is not.

3. Write code that prints out the largest of the two variables; if they are equal, it should print out the word `EQUAL`.

Short-Cut Evaluation

Programmers are always trying to make programs run faster, by never doing more work than is needed.

For example, the creators of Java wanted to make sure that we could get the final output value of a complicated boolean expression as fast as possible.

Therefore, when a program executes, it will use some short-cuts in evaluating these expressions:

1. If either side of a conjunction (\&\&) is false, the entire expression is also false. **We can stop.**

2. If either side of a disjunction (\|\|) is true, the entire expression is also true. **We can stop.**
Short-Cut Evaluation

- Because of the short-cuts, things that are logically equivalent in do not always do the same thing in a program.
- In logical terms, e.g., these two things always mean exactly the same:

\[
\text{if } ( X \land Y ) \\quad \text{if } ( Y \land X )
\]

- But in Java execution, there is a difference—since we go left-to-right in evaluating the expression, they each work differently:

```
if ( X && Y ) {
    ...
}
if ( Y && X ) {
    ...
}
```

Using Short-Cut Evaluation

- We can sometimes use this feature to our advantage.
- As an example, we can order our instructions so that the first one that is false causes the evaluation of the boolean condition to stop immediately.
- This allows us to write code that is more compact and less buggy:

```
String data = input.getText();
if ( data.charAt(5) == 'a' && data.length() > 5 ) {
    System.out.println( "One thing" );
}
```

This Week & Next

- Meetings this week:
  - Monday/Tuesday/Wednesday: regular classroom
  - Friday: in the CS Lab (16 Wing)
- Program 03: due 11:59 PM, Thursday, 3 October
- Reading Assignment 3: due Thursday, 26 September at Noon
- Midterm 1: Monday, 07 October
- Office Hours: Wing 212
  - Monday/Friday: 2:15 PM–3:15 PM
  - Tuesday: 1:30 PM–2:30 PM
  - Wednesday: 12:05 PM–1:00 PM
- Lab and Tutor Hours: On my website