Arithmetic Expressions

- Operations on numerical types

- Operations:
  - + “addition”
  - - “subtraction”
  - * “multiplication”
  - / “division” (different for int vs. double)
  - % “remainder”

- Precedence (in order):
  - () highest
  - *, /, %
  - +, - lowest

  Operators in same precedence category evaluated left to right
Type Casting

- Treat one type as another for one operation

```java
int x = 3;
double y;

y = x / 2;        // y = 1.0

y = (double)x / 2; // y = 1.5

y = 5.9;
x = (int)y;       // x = 5

x = 7;
y = x;            // fine: y = 7.0
x = y;            // error
```
Expression Short-hands

```c
int x = 3;

x = x + 1;   x += 1;   x++;

x = x + 5;   x += 5;

x = x - 1;   x -= 1;   x--;

x = x * 3;   x *= 3;

x = x / 2;   x /= 2;
```
Boolean Expressions

• Boolean expression is just a *test* for a condition
  • Essentially, evaluates to *true* or *false*

• Value comparisons:
  
  == “equals” (note: not single =)

  != “not equals” (cannot say <>)

  > “greater than”

  < “less than”

  >= “greater than or equal to”

  <= “less than or equal to”
More Boolean Expressions

- **Boolean comparisons (in order of precedence):**
  
<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>“not”</td>
</tr>
<tr>
<td>!p</td>
<td>if p is true, then !p is false, and vice versa</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>“and”</td>
</tr>
<tr>
<td>p &amp;&amp; q</td>
<td>only true if p and q are both true</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
</tr>
</tbody>
</table>

```java
boolean p = (x != 1) || (x != 2);
note: p is always true, you really want:
```

```java
boolean p = (x != 1) && (x != 2);
```
Short Circuit Evaluation

• Stop evaluating boolean expression as soon as we know the answer

• Consider:

\[ p = (5 > 3) \text{ || } (4 \leq 2); \]

The test \((4 \leq 2)\) is not performed!

• Example of useful case:

\[ p = (x \neq 0) \text{ && } ((y / x) == 0); \]

Avoid division by 0, since \(((y / x) == 0)\) is not performed