Operators

5x - 2 = 13
then
x = 3

But professor, you said yesterday that x was equal to 2!
Variables are objects in your method (or class)

To use variables two things must be done:
- Declaration
- Initialization

See: Uninitialized.java

C example:
I am 68882420 inches tall.
I am -1094369310 inches tall.
Variables

```
int x, y, z; // Declaration
x = 2;
{  y = 3;
  z = 4;
}
```

Same as:

```
int x=2, y=3, z=4;
```

Variables can be declared anywhere (preferably at start)
Constants

Constants do not need to be initialized or declared and cannot change value.

Example:

2 or Math.PI

(Constants are sometimes called literals)

To make a constant, put final before the type:

```java
final int x;
```

(can only set the value of x once)
Assignment operator

= is the assignment operator

The object to the right of the equals sign is stored into the object in the left

```c
int x, int y;
y = 2;
x = y + 2;
y = 10;
```
Assignment operator

= is NOT a mathematic equals

x=3;
x=4;

This does not mean 3=4
Assignment operator

To the left of = needs to be a valid object that can store the type of data on the right

```c
int x;
x=2.6; // error, 2.6 is not an integer
x+2 = 6; // x+2 not an object
2 = x; // 2 is a constant, cannot store x
```
Assignment operator

What does this code do?

```c
int x = 2, y = 3;
y=x;
x=y;
```

What was the intention of this code?
Increment operators

What does this code do?

```java
int x = 2;
x = x + 1;
```

Same as:

```java
x += 1;
```

or

```java
x ++ ;
```
Increment operators

Two types of increment operators:

\[ x++ \quad \text{// increments after command} \]
\[ ++x \quad \text{// increments before command} \]

(see IncrementOperator.java)
Complex assignments

The following format is general for common operations:

variable (operator)= expression
variable = variable (operator) expression

Examples:

\[
x += 2 \quad \leftrightarrow \quad x = x + 2
\]

\[
x *= y + 2 \quad \leftrightarrow \quad x = x * (y + 2)
\]

(see AssignmentOperator.java)
Order of operations

Order of precedence (higher operations first):
- - , ++, -- and ! (unary operators)
* , / and % (binary operators)
+ and - (binary operators)

% is remainder operator
(see Remainder.java)
Order of operations

Binary operators need two arguments
Examples: (see BinaryOperators.java)
2+3, 5/2 and 6%2

Unary operators require only one argument:
Examples: (see UnaryOperators.java)
+x, x++, !x

(! is the logical inversion operator for booleans)
Order of operations

When multiple operations have the same precedence level:

Binary operations go from left to right

Unary operations go right to left;
boolean values

Sometimes this might cause an error, such as:

```java
int x = 2;
if( ! x>5 ) will not compile
Why?
```
Sometimes this might cause an error, such as:

```java
int x = 2;
if( ! x>5 ) will not compile
```

Why?

A: order of operations will do the unary operator first (the '!=')

if (! x>5) will become if ( !(2) > 5)

...and Java does not understand !2
Logical operators

These are all the operators that result in a boolean:

> (greater than), e.g. 7 > 2.5 is true
== (equals), e.g. 5 == 4 is false
< (less than), e.g. 1 < 1 is false
>= (greater than or equal to), e.g. 1 <= 1 is true
!= (not equal to), e.g. 8 != 7 is true
<= (less than or equal to), e.g. 6 <= 2 is false
! (not, negation), e.g. !true is false
Complex expressions

Two boolean operators:

- `&&` is the AND operations
- `||` is the OR operations

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p &amp;&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>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |

| p | q | p || q |
|---|---|-------|
| T | T | T     |
| T | F | T     |
| T | T | T     |
| F | T | T     |
| F | T | F     |
Complex expressions

AND operation removes Ts from the result
The OR operation adds Ts to the result

Evaluate (!p OR q) AND (p)

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>!p</th>
<th>!p OR q</th>
<th>(!p OR q) AND (p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Write an boolean expression that is true if a variable (int) x is a positive odd number.

Hint: You may want to use the remainder (also called modulus) operator (the % sign).

For example, 5 % 3 = 2
Complex expressions

Humans tend to use the english word OR to describe XOR (exclusive or)

“We can have our final exam on the scheduled day (May 13) or the last day of class (May 6).”

Did you think the statement above meant final exams on both days was a possibility?
Complex expressions

Write boolean expressions for each of the following truth tables:

1. \[
\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

2. \[
\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

XOR

3. \[
\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

4. \[
\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
Complex expressions

```c
int x = 9, y = 7;

x < 12 && y > 10

9 < 12 && 7 > 10

T && F

F
```
Complex expressions

If statements for when $x$...

... is between 10 and 20 (inclusive)

$$(10 \leq x \land x \leq 20)$$

Cannot say: $10 \leq x \leq 20$ (why?)

$$(x == 'a' \lor x == 'e' \lor x == 'i' \lor x == 'o' \lor x == 'u')$$

... is a vowel ($x$ is type `char`)
Complex expressions

Be careful when negating, that you follow De Morgan's Law:

```cpp
bool a, b;
!(a OR b) is equivalent to (!a) AND (!b)
!(a AND b) is equivalent to (!a) OR (!b)
```

“Neither rainy or sunny” means
“Both not rain and not sunny”