The boolean type and boolean operators

- Recall that Java provides a data type `boolean` which can take on only one of two values: `true` or `false`.

- `boolean b = true;  // stores the truth value true in b`
- `b = false;  // overwrites b with the value false`

- There are other ways to create boolean values and assign them into boolean variables besides a simplistic direct assignment of a boolean literal into a variable. Boolean operators produce true/false values.
- For example let’s assume this declaration: `int i = 10;`

- We can assign a truth value into variable `b` using boolean operators like this: `b = i < 20;`

- The expression `i < 20` is true since `I` contains the number 10. The value `true` is then assigned into the variable `b`.

- Let’s look at a summary of all the boolean operators and their behavior.
**Boolean operators**

<table>
<thead>
<tr>
<th>&amp; &amp;</th>
<th>true</th>
<th>false</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>
</tbody>
</table>

| || | true | false |
|----|------|-------|
| true| true | true  |
| false| true | false |
| true| false | false |

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

and -- ALL conditions must be true
or -- ANY condition can be true

&& is logical **and**
|| is logical **or**
! In logical negation (pronounced **not**)

true and true
true and false
false and true
false and false

true or true
ture or false
false or true
false or false
boolean variables

boolean variables can have the value true or false. That’s it.

boolean minor, foo;
int age = 21;

foo = true;
minor = ( age < 18 );  // (age<18) produces either true or false

What value is now in the variable minor?
Relational operators

- Relational operators produce boolean values
- ==  equality
- !=  inequality
- <   less than
- <=  less than or equal
- >   greater than
- >=  greater than or equal

- Relational operators have higher priority then boolean operators

- x < y && a > b evaluated as if \( \Rightarrow (x<y) && (a>b) \)

- Not a bad idea to parenthesis just for emphasis/clarity
Short circuiting practice

Short-circuiting happens when the result can be determined before the entire expression has been examined.

What do each of the following boolean expressions evaluate to?

Which of the following expressions short circuit?

```java
boolean a = true, b = false;
int c = 6, d = 5;

a && (!b)
a && b
b && a
b && (!a)
a || d < c
a || d < c
d != 10 || b
b || c == 6
d > 10 || b
```
More short-circuiting practice

```java
boolean a = true, b = false;
int c = 6, d = 5;

b && (b || c < d)

(c < d) || b

a && (!b || (c < d))

a || b

!a

!(a || b)

!((c > d) && a)

(!b && a) || (a && d < c)
```
not has a higher precedence than and/or

DeMorgan’s Law

DeMorgans Law - any expression can be equivalently expressed by
  multiplying a NOT through the boolean expression and changing || to &&
  or changing && to ||

The negation of a conjunction is the disjunction of the negations

!(p && q) → !p || !q

The negation of a disjunction is the conjunction of the negations

!(p || q) → !p && !q
and/or examples

- and examples

```java
if (age > 6 && age < 19 )
{
    System.out.println("You should be in school!");
}

if ( age < 18 && milesOverLimit > 20)
{
    System.out.println("Underage flagrant speeders get double fine!");
    fine *= 2;
}
```

- or examples

```java
if ( letter == 'A'|| letter == 'B' || letter == 'C' )  \text{RIGHT}

if (letter == 'A'|| 'B' || 'C' )  \text{WRONG}
```
More forms of the if statement

Simple conditional: use if

```java
if ( age < 21 )
{
    System.out.println("too young to drink :=( ");
}
```

Two way branch: use if else

```java
if ( age < 18 )
{
    System.out.println("too young to drink :=( ");
}
else
{
    System.out.println("Draft or bottle?");
}
```
three way branch – use an if else/if else

if ( age < 18 )
{
    System.out.println("too young to drink :-( ");
}
else if ( age < 70 )
{
    System.out.println("Draft or bottle?");
}
else
{
    System.out.println("How about some Geritol instead?");
}
Good usage of the if test

You may have your if structured like this:

```java
if ( <boolean expression here>)
{
    // nothing in the if part
}
else
    do something
```

In that case negate the test and put the action under the if instead of the else

```java
if ( !<boolean expression here> )
    do something
```

Now you don’t need the else