“You’ve got to be very careful if you don’t know where you are going, because you might not get there.”

Yogi Berra
Computers are more than calculators

• So far your programs have just done mathematical calculations
• Most programs need to make decisions
• The Java “if” statement is used to make decisions in a program
if Statement

```java
if (paid > bill) {
    change = paid - bill;
}
```

If the contents of paid are greater than the contents of bill, then the change statement will be executed.
if Logic

Boolean Expression

Statement(s)
if Syntax

```
if ( true or false decision )
    next statement;
```

- The program will execute the next statement if and only if the decision is true
- The next statement can be a single Java statement or a block
- Whitespace is optional
Logical Expressions

• The logical decision of an if statement must be a logical expression
• The most common logical expression is the comparison of two variables
• The result of a logical expression must be either true or false
• Comparisons are of the format
  cat \textit{operator} dog
  
  cat > dog
<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
<tr>
<td><code>==</code></td>
<td>equal to</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>not equal to</td>
</tr>
</tbody>
</table>
Watch Out for Double Equals

if (ant = bat)        /* Incorrect */

if (ant == bat)       /* Correct */
Blocks

• In Java, a **block** is a bunch of code surrounded by 
  {  curly brackets  }

• Almost anyplace you might put a single statement, you can put a block of statements

  *hint:*

• *When you type a left curly bracket, immediately type a right curly bracket and then backup between them*
Blocks and Statements

• The statement following an `if` can be a single statement or a block

```plaintext
if (cow < bird) {
    cat = dog;
    tree = bird;  // executed only if cow < bird
}
```

```plaintext
// always executed
```
Write some Java

```java
int dog = ?, cat = ?, cow = 7;

if dog is greater than cat, set cow to 5;
```
Write some Java

```java
int dog = ?, cat = ?, cow = 7;

if dog is greater than cat, set cow to 5;
if ( dog > cat ) {
    cow = 5;
}
```
What is displayed?

```
int ant = 3, bird = 5;
int cat = 7, dog = 47;
if (bird >= cat) {
    dog = 13;
}
System.out.println(dog);
```

A. 5
B. 13
C. 47
D. none of the above
Now what is displayed?

```java
int ant = 3, bird = 5;
int cat = 7, dog = 47;
if (ant + bird >= cat) {
    dog = 13;
}
System.out.println(dog);
```

A. 5
B. 13
C. 47
D. none of the above
Indenting

• Although the Java compiler does not care, it is traditional to indent the statements that are executed only when the if is true

```java
if ( dog > cat ) {
    cow  = 5;
    bull = 37;
}
```
Various Formats

```c
if ( bull > cow ) goat = 17;

if ( bull > cow )
    goat = 17;

if ( bull > cow ) {
    goat = 17;
}
```
Indenting Required in all Assignments

• Indenting makes it easier for humans to read
• Remember that indenting does not determine the logic
• *Your programs will receive a reduced grade if they are not indented correctly*
What is displayed?

```java
int ant = 3, bird = 5;
int cat = 7, dog = 47;
if (bird >= cat)
    System.out.print(bird);
    System.out.print(cat);
System.out.println(dog);
```

A. 5 7 47  
B. 7 47  
C. 47  
D. none of the above
Brackets would fix the problem

```java
int ant = 3, bird = 5;
int cat = 7, dog = 47;
if (bird >= cat) {
    System.out.print(bird);
    System.out.print(cat);
}
System.out.println(dog);
```
Compound Logical Statements

• You can combine relational expressions with logical or Boolean operators
• Expressions can be combined with AND, OR, XOR and NOT
## Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>exclusive or</td>
</tr>
</tbody>
</table>

George Boole
19th century British mathematician
inventor of Boolean logic
Truth Table for NOT

<table>
<thead>
<tr>
<th>dog &gt; 5</th>
<th>!(dog &gt; 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

! (dog > 5) *is* dog <= 5
# Truth Table for AND

<table>
<thead>
<tr>
<th>dog &gt;5</th>
<th>cat &lt;2</th>
<th>dog&gt;5 &amp;&amp; cat&lt;2</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

dog > 5 && cat < 2 is true only when both dog is more than 5 and cat is less than 2
**Truth Table for OR**

| dog >5 | cat <2 | dog >5 || cat <2 |
|--------|--------|----------------|
| true   | true   | true           |
| true   | false  | true           |
| false  | true   | true           |
| false  | false  | false          |

**dog > 5 || cat < 2** is true when
either dog is more than 5 or cat is less than 2
Truth Table for XOR

<table>
<thead>
<tr>
<th>dog &gt;5</th>
<th>cat &lt;2</th>
<th>dog &gt;5 ^ cat &lt;2</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</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>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

This is not used very often
Compound Logic in if

- Set grade to 4.0 if both study is greater than avg and work is equal to done

```cpp
if( (study > avg) && (work == done) ) {
    grade = 4.0;
}
```
What is displayed?

```java
int ant = 3, bird = 5, cat = 7;
int dog = 47;
if ((ant != bird) && (cat > bird)) {
    dog = 13;
}
System.out.println(dog);
```

A. 5  
B. 7  
C. 13  
D. 47
Now what is displayed?

```java
int ant = 3, bird = 5, cat = 7;
int dog = 47;
if ((ant == bird) || (cat > bird)) {
    dog = 13;
}
System.out.println(dog);
```

A. 5
B. 7
C. 13
D. 47
Caution

Adding a semicolon at the end of an if clause is a common mistake
if (radius >= 0) ; { area = radius*radius*PI; System.out.println(radius); } 
• This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error
• This error often occurs when you use the next-line block style
Try It

• Print an error message if the variable loan or the variable years are less than zero

double loan, years;

// input the value of loan and year

if ( ? ) {
    System.out.println("Bad stuff");
}

Possible Solution

• Print an error message if the variable loan or the variable years are less than zero

double loan, years;

// input the value of loan and year

if ( loan < 0.0 || years < 0.0 ) {
    System.out.println("Bad stuff");
}
Incorrect Solution

- Print an error message if the variable loan or the variable years are less than zero

```java
double loan, years;
// input the value of loan and year

if ( loan || years < 0.0 ) {
    System.out.println("Bad stuff");
}
```