My Richard The Third The Video Game is going to be FANTASTIC.

MAN I HEAR THAT

All I need to do is program it!

LATER:
Um, APPARENTLY, programming is for folks who are thrilled when a computer reminds them they're missing a bracket or semicolon? It must be, because they make that happen SO OFTEN.

So it's not going well?
I CAN'T EVEN GET RICHARD THE THIRD TO MOVE.
You know what my game is now? My game is NIBBLES, with the text changed from "Copyright Microsoft 1990" to "man, forget this"

Programming's a skill!

I just thought it was a skill I could pick up easily. I don't need to know everything! I don't need to know the difference between friggin' binary and B+ search trees! ALL I WANT TO KNOW is how to make Richard III's sucky horse do double jumps, you know?

Hey, here's a tip!

You've got to learn to crawl before you can run, T-Rex.

Crawling sucks!
Flow Control

So Far: Writing simple statements that get executed one after another.

Flow control allows us to change program behavior based on different situations

(Sometimes called “Selection statements”)
Booleans

A boolean is a value that is either true or false.
A boolean is a value that is either true or false. Can you think of an expression that would use booleans?
Booleans

A boolean is a value that is either true or false. Can you think of an expression that would use booleans?

Java has a data type for boolean values.
Booleans

A boolean is a value that is either true or false. Can you think of an expression that would use booleans?

Java has a date type for boolean values. Java boolean values are written as “true” or “false” (notice the capitalization)
Booleans

Examples:

boolean isRaining = true;
boolean isRaining = false;
Comparison Operators

Allow us to compare values of variables, literals, or expressions and return a boolean value (lower precedence than arithmetic operators).

== equals
!= not equals
< less than
> greater than
<= less than or equal
>= greater than or equal
Comparison (equals)

- \( p == q \)

- **Do not confuse this with assignment** (single equals '==') where we are *setting values* equal to something
  
  \( p = q \), is not the same as \( p == q \)

  \( p = q \) sets the variable \( p \) to the variable \( q \)

  \( p == q \) returns whether \( p \) is equal to \( q \)
Boolean Logic Operators

We've talked about arithmetic operators, and comparison operators, but Java provides boolean logic operators as well.

Boolean logic operators return...
Boolean Logic Operators

We've talked about arithmetic operators, and comparison operators, but Java provides boolean logic operators as well.

Boolean logic operators return... booleans!
Boolean Logic Operators

• Negation: !
• And: &&
• Or: ||

• Examples (given two boolean values p & q):
  • p && q
  • P || q
  • !p
Complex Boolean Expressions

• Just as you can have multiple arithmetic operators in a given expression (3+4-3*2), you can have multiple boolean operators in an expression (False and not False or True)

• Order of operations, highest priority first:
  • 1st: !
  • 2nd: &&
  • 3rd: ||
Short-Circuit Evaluation

- As soon as Java can determine the outcome of a boolean logic operator (unlike comparison operators) it returns the result
  - Faster code
- False and Anything: is always False
- True or Anything: is always True
- Java will not execute the code in 'Anything'
  - true || (p && q)
    - (p && q) is never checked
  - false && (p || q)
    - (p || q) is never checked
- Can lead to bugs if you rely on code in 'Anything' being run
If - else if - else statements

• We can use all these boolean operators to control the flow of the program.

• The if-else if-else statement looks like:

```java
if (<boolean condition>) {
    <statements>
} else if (<boolean condition>) { #repeat as needed
    <statements>
} else { # this block is optional
    <statements>
}
```
Conditional expressions

- We can also do a shorthand version of this that allows assignment (ternary operator):

  ```java
  boolean isEven = (number%2 == 0) ? true : false;
  ```

Sets `isEven` to true if `number` is divisible by 2 otherwise, sets `isEven` to false.
## Order Of Operations
(highest to lowest)

<table>
<thead>
<tr>
<th>Type</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>postfix</td>
<td>expr++, expr--</td>
</tr>
<tr>
<td>unary</td>
<td>++expr, –expr, +expr, -expr, ...</td>
</tr>
<tr>
<td>multiplicative</td>
<td>*, /, %</td>
</tr>
<tr>
<td>additive</td>
<td>+, -</td>
</tr>
<tr>
<td>comparison</td>
<td>==, !=, &lt;=, &gt;=, &lt;, &gt;</td>
</tr>
<tr>
<td>logical NOT</td>
<td>!</td>
</tr>
<tr>
<td>logical AND</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>logical OR</td>
<td></td>
</tr>
<tr>
<td>assignment</td>
<td>=, +=, -=, *=, /=, %=, ...</td>
</tr>
</tbody>
</table>
Nested If statements
Nested If[-elif][-else] Statements
Nested If[-elif][-else] Statements
Nested 'if' statements

• The if-elif statement looks like:

```python
if (<condition>) {
    if (<condition>) {
        <statements>
    } else {
        <statements>
    }
} else if (<condition>) { // repeat as needed
    <statements>
} else {
    // this block is optional
    <statements>
}
```