Conditional Statements
if - else

Conditional?

“If” statements are called “conditional” because what happens depends on conditions.

For example: If the weather is good I’ll play soccer, otherwise the game will be cancelled and I’ll go to board game club instead.

```
if (weather == good) {
    play soccer
} else {
    soccer game cancelled
    go to board game club
}
```

"We flew down weekly to meet with IBM, but they thought the way to measure software was the amount of code we wrote,
when really the better the software,
the fewer lines of code."

-Bill Gates

Syntax:

```
if (condition) {
   statement;
   statement;
   …
}
```

```
// if the ball hits the top wall, make it go faster and bounce it off the top

if (ball.y < 0) {
   xspeed +=5;
   yspeed = -yspeed;
}
```

All IF statements have:

- if
- ( ) with a condition inside them

Most IF statements have a code block too: { }

Syntax: One line If statement

```
if (condition) statement;
```

```
if (lives == 0)  endGame = true;
```

- these should only be used for short statements, things that are clear.

Simple Mistake #1

```
if (redButton.isPressed); {
    detonateExplosives();
}
```

NEVER EVER put a semicolon after the (). What this does is:

```
if (redButton.isPressed);  //this no longer does anything
{
    detonateExplosives();
}
```

which is identical to:

```
if (redButton.isPressed);  //this no longer does anything
detonateExplosives();    //no matter what happens, this is going to blow up .... 💥
```

Simple Mistake #2

You can write a one-line if statement, and **you do not need a code block.**

```
if (redButton.isPressed)   detonateExplosives();  //This if statement will only do one thing.
```

Technically, you could also write it like this:

```
if (redButton.isPressed)
       detonateExplosives();
```

But one day you decide to add something …

```
if (redButton.isPressed)
       sendWarning();
       detonateExplosives();
```

There is NO CODE block.
It looks like the second line is part of the IF statement, BUT IT IS NOT!

Now the explosives will detonate 💥 EVERY time that the program runs.

Solution: NEVER SPLIT A ONE LINE IF STATEMENT INTO TWO LINES.

Comparison Operators

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=`	does not equal	`3.2 != 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`
`>=`	greater than or equal to	`5.0 >= 5.0`	`true`

- You cannot write “=>” “equal to or greater than”
- It must always be “<=” “greater than or equal to”
- **!** means NOT

Recall that for **objects** (Strings etc.) you have to use **.equals()**

```
if ( answer.equals("YES") ) { ...
if ( ! answer.equals("NO") ) { ...      // if the answer is NOT NO.
```

Comparison Operators

- You cannot write things the way that you do in math
  (no surprise here since 3=x doesn't work)

- To check if a number is a single (positive) digit
  - if (0 < x < 10) { ...}   DOES NOT WORK
  - if (1 < x && x < 10) {...} IS THE CORRECT WAY

- What does && mean? It means AND
- You have to put the variable in twice. Once for each < sign.

- So our if statement says : if x is greater than 0 AND x is less than 10

More about && on the next slide ...

Logical Operators

These are used to join comparison operators together in IF statements and in FOR loops

&&	means AND
||	means OR
!	means NOT

eg. if (date == 29 && month == 2) System.out.println("Leap day!!");

★ Always use && or ||

Do not use & or | – *this will be explained at some point in the future*

Logical Operators ... combining

Example: *When the ball hits the left wall, bounce it back*

```
        if ( ball.xspeed < 0 && ball.x < 0)  ball.xspeed *= -1;
```

If you combine AND and IF, it’s best to use parentheses to get the logic correct.

Example: Everyday *at 9am you have to do something, except on Sundays when you do it at 11am.*

```
        if (time == 9.0 || day.equals("Sunday") && time == 11.0 ) {
```

Problem: does OR happen before or after AND? Solution: use more ()

```
        if ( time == 9.0 || (day.equals("Sunday") && time == 11.0) ) {
```

Comments - bad and good

Poor style:

```
// Test whether student 1's GPA is better than student 2's
if (gpa1 > gpa2) {
    // print that student 1 had the greater GPA
    System.out.println("The first student had the greater GPA.");
} else if (gpa2 > gpa1) {
    // print that student 2 had the greater GPA
    System.out.println("The second student's GPA was higher.");
} else {
    // there was a tie
    System.out.println("There has been a tie!");
}
```

Comments - bad and good

Describe why you are performing that test, and/or what you intend to do based on its result.

Good style:

```
// Print a message about which student had the higher GPA.
if (gpa1 > gpa2) {
    System.out.println("The first student had the greater GPA.");
} else if (gpa2 > gpa1) {
    System.out.println("The second student's GPA was higher.");
} else {   // gpa1 == gpa2 (a tie)
    System.out.println("There has been a tie!");
}
```

Refactoring - making code more efficient

The following example has a lot of redundant code in the if/else:

```
if (money < 500) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Caution!  Bet carefully.");
    System.out.print("How much do you want to bet? ");
    bet = scanner.nextInt();
} else if (money < 1000) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Consider betting moderately.");
    System.out.print("How much do you want to bet? ");
    bet = scanner.nextInt();
} else {
    System.out.println("You have, $" + money + " left.");
    System.out.print("You may bet liberally.");
    System.out.print("How much do you want to bet? ");
    bet = scanner.nextInt();
}
```

All the red code is repeated.

Repeated code is a sign that code is poorly written.

Refactoring - making code more efficient

- If the beginning of each if/else branch is essentially the same, try to move it out *before* the if/else.
- If the end of each if/else branch is the same, try to move it out *after* the if/else.

```
System.out.println("You have, $" + money + " left.");

if (money < 500) {
    System.out.print("Caution!  Bet carefully.");
} else if (money < 1000) {
    System.out.print("Consider betting moderately.");
} else {
    System.out.print("You may bet liberally.");
}

System.out.print("How much do you want to bet? ");
bet = scanner.nextInt();
```

This is much cleaner.

Nested IF

… still to be done …

Else-if

When you use else-if, the order of the if statements is critical:

```
if (x > 50) {
    you get $50
} else if ( x%2 == 0) {  //x is an even number
   you get $10
}
```

Which numbers get $10?
Which numbers get no money?

Else-if reversed conditions

Same example, but the order is reversed.

```
if ( x%2 == 0) {  //x is an even number
    you get $10
} else if (x > 50) {
   you get $50
}
```

Which numbers get $10?
Which numbers get no money?

Else - if

.: Do not use "else if" unless you absolutely have to.

```
if (score == 100) lives++;
else if (score == 200) lives +=2;
```

NO! If score is 100, then it cannot be 200. You do not need **else-if** here.

YES: Write it like this:

```
if (score == 100) lives++;
if (score == 200) lives +=2;
```

Else - if

The only reason to use else-if with a lot of == conditions is if you need an “else” clause.

```
if (day == Saturday)  {
    go to work;
} else if (day == Sunday) {
    sleep in;
} else {
    go to school;
}
```

Doing this without using else-if would be really awkward.

Else - if $\longrightarrow$ switch

But you can code the previous example using SWITCH

```
switch (day) {
case "Saturday":
    go to work;
   break;
case "Sunday":
    sleep in;
    break;
default:
    go to school;
}
```

You should learn how to do this, and why we need "break"

For this situation, SWITCH is better than ELSE-IF chains.