Java 3: Conditions

The Heart of Turning Computing

Conditions and Branching

Condition statements use True or False to make decisions. The most common condition statement is the “if” statement. When you use an “if” statement you can think of it like asking “If this is true, do this”. With these “if” statements, we also use “else”. “else” is like saying “If this is true, do this, otherwise do this”. We will see in the coding example how these are used.

Branching isn’t something we can do explicitly in Java, but its concept does exist. Branching is any time your program breaks its execution sequence to do something else, such as a function call, and if statement, etc.

Testing For True and False

There is many ways to check a variable, for our example let’s say we are testing two integers. Our options available are: == to test for equality, > for greater than, < for less than, >= for greater than or equal, and <= for less than or equal. When we want to check for not equal we use !=. An ! in programming typically means “not” and can be used to negate variables, or tests. Note that negating a variable doesn't mean making it negative, so applying it to a integer wont do anything.

A Coding Example

We once again will use our code from before

```
public class HelloWorld{
     public static void main(String args[]){
          int x = 1;
          int y = 2;
          if(x > y){
               x = 10;
          }
          else{
               y = 10;
          }
          System.out.println(x + “ “ + y);
     }
}
```

Code Analysis

We have quite the complex code now. At the start we simply declare our variables x and y, giving them the values 1 and 2 respectively. Then we test the condition that x is greater than y. We know this is false, so x shouldn't be set to 10. You can see then we have our else statement, which will execute because the previous if was false. In that we are setting y to 10, which we do by stating that y = 10. This is like the declaration, except we are not recreating the variable, we are resetting its value.

Finally in our println, you see something strange: (x + " " + y). This is because using + on a string is considered a concatenation. So what we are doing is concatenating our variables together, with a space in between them . You'll see our output is "1 10".

You’re now a conditions master!

You can now experiment with more complex conditions, a common one being if a number is even. Think about how you would do it… What properties do we know about even number that we can apply here… well, first a even number is divisible by two so we can use that fact to help us, but how can we check if a number is divisible by another number… ah, if the remainder is 0.

Using all this we can say something like: if we have a int x and we want to test if its divisible by two, we can use x % 2 == 0

Think about this, we’re saying how many times does 20 go into 2, it seems backwards right? This is the kind of slips that beginners make that cause them to struggle.

Next: loops

Loops are a crucial part in programming. It is one of the essential components that you should get comfortable with.