Lecture 1.2 — Java intro

Variables [Starting with python]

variable = a box with a name...

```
# x does not exist
x=1
print(x)
x=1.7
print(x)
x="Hello"
print(x)
```

Var name is ...

left of =	*put something into the box*
elsewhere	*get something out of the box*

In python

- Assigning a value into a name is enough to make it exist.
- Different sorts of things could go in the same variable.

Java — static types

Variables have types as well as names so they need to be "declared" before they can be used.

```
int x;   // x will store whole numbers
x=1;     // legal
x=-5000; /* legal */
x=1.0;   // illegal: not a whole number
x="15"; // illegal: not a whole number
```

Primitive types

Java distinguishes two sorts of types:

- "primitive types":

`boolean`		`true` or `false`
`byte`		-100, 120
`char`	one unicode character	'A', '6'
`short`	small whole number	
`int`	whole numbers	1, -500
`long`	larger range of whole numbers	
`float`	floating point number	
`double`	higher precision floating point	$27.8, -1.9e200$

- "classes" (reference types): everything else

For example, Strings and arrays are classes.
Because literals and variables have types, expressions using them do too. (Watch out — division between ints gives an int answer).

scope

A variable's scope describes where your program can use that name to refer to that variable. In Java, a variable's scope ends at the end of the block it is declared in*.

This works in python:

```
# z does not exist
if 5>4:
    z=2
else:
    z=3
print(z)
```

This does not work in Java:

```
if (5>4) {
    int z=2;
} else {
    int z=3;
}
System.out.println(z);
```

Uninitialised variables

Q: What is the value of y after the following?

```
int x, y; // declaring multiple variables

...          // nothing affecting x or y in here
```

y=x+1;

A: It depends.
It either sets it to something which looks like zero or won't compile (depending on where the variable is declared).
You are better off explicitly initialising your variables.

Dynamic typing (Python)

```
x=1
print(x,type(x))          1 <class 'int'>
x=1.7
print(x,type(x))          1.7 <class 'float'>
x="Hello"
print(x,type(x))          "Hello" <class 'str'>
```

Python knows that 1 and 1.7 are different sorts of values (different "dynamic" / "runtime" types) but doesn't restrict variables on that basis.
Java can do this too, but in a more restricted way.

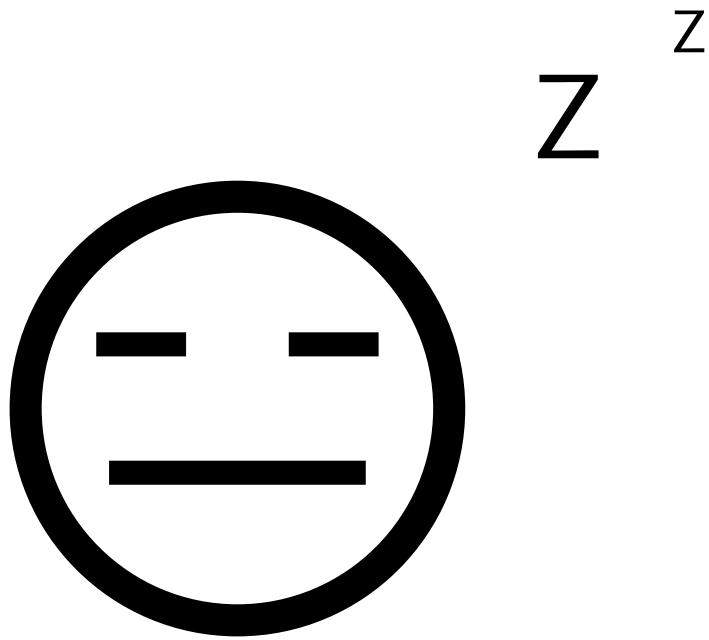
Control flow — if

Hello.java

for

```
for (Init; Test; Update) { Body }
```

Init	OR	Init	OR	Init	...
Test		Test		Test	
		Body		Body	
		Update		Update	
		Test		Test	
				Body	
				Update	
				Test	


z
z

Java (temporary) "Magic"

So your most basic Java program `XYZ.java` will be:

```
public class XYZ {

public static void main(String[] args) {
    // your code here
}

}
```

Examples

Program	
`Loops.java`	`for`, `while`, `do-while`, increment operator
`ForEach.java`	for-each
`Switch.java`	`switch`, array literal, `exit()`
`Rec.java`	simple recursion