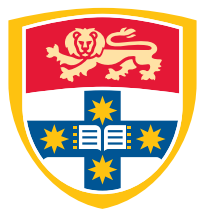
# INFO1103: Introduction to Programming

School of Information Technologies, University of Sydney



# Lecture 1: Programming basics

# What is a program?

A program is a set of instructions that were written, probably by a human, in such a way that can be converted to instructions that a computer can understand and then execute.
Wikipedia has this to say:

> *A computer program (also software, or just a program) is a sequence of instructions written to perform a specified task with a computer.*

# Your First Program

The *classic* program of all time is "HelloWorld". In Java it looks like this:

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```

# Wait, that was just some text.

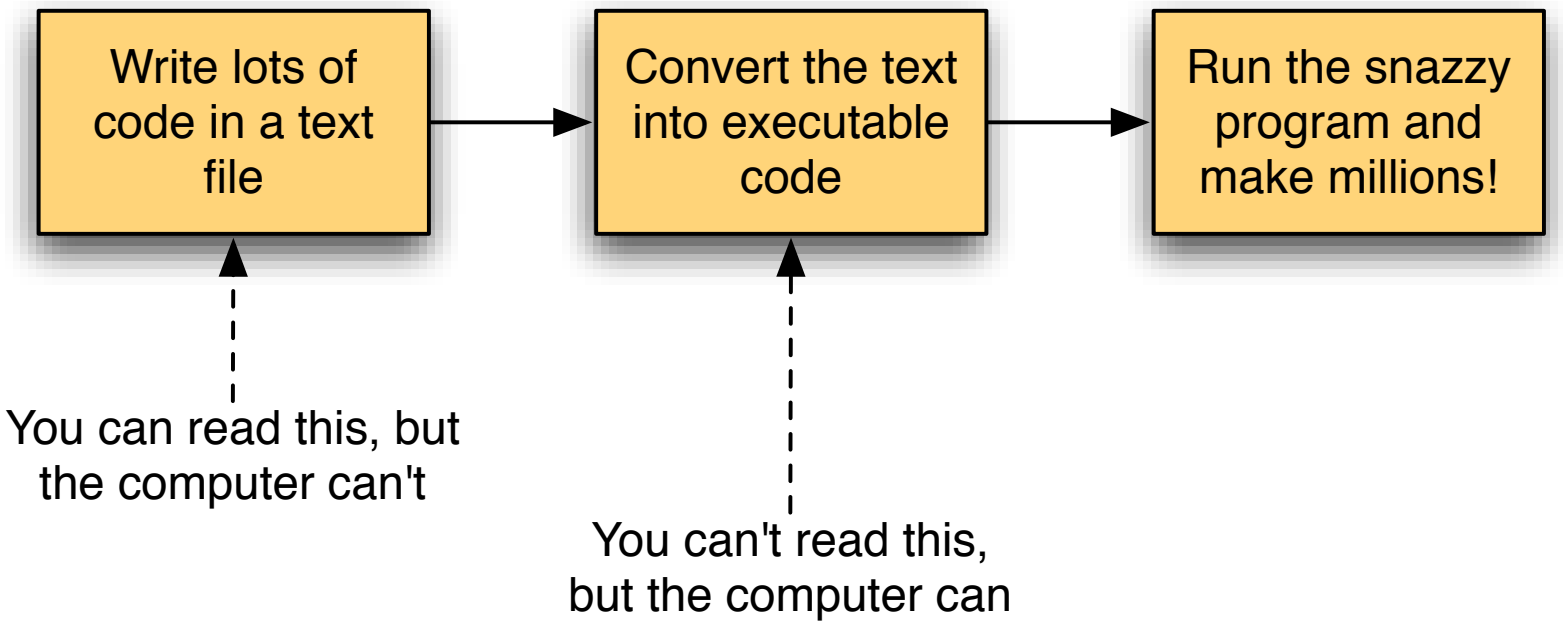
Ok, true.

When you write a program you really are writing a text file, or a set of text files, which you might even link with other programs.

The text file you write has to be in a special, very particular format, such that it obeys the *syntax* of the programming language you are using.

Once the text file is written, a *compiler* is used to process the file and turn it into something that a computer can run: an *executable*, a *program*, or an *application* or *app*.

# Wait, that was just some text. (cont.)



# There are many programming languages

. . . and no “best one”. We have chosen Java for this course as it is a major language used in industry, is cross-platform and has some nice features that make it very suitable for this course and INFO1105 (Data Structures), in Semester 2.

Here are some major languages:
C, C++, C#, BASIC, VisualBasic, Java, Python, Perl, Pascal, Haskell, Fortran, Cobol.

# Languages

In old (1978) C it would be more like this:

```c
int main() {
   printf("Hello, World!\n");
}
```

in ISO-C++ it would be like this,

```cpp
#include <iostream>

int main()
{
   std::cout << "Hello World!" << std::endl;
}
```

in Perl like this:

```
      print "Hello World!\n";
```

Python:

```
   print "Hello World"
```

# In general...

What we have for all of these programs is something like the following:

```
1 <load things we need>    // "optional"
2 <begin a method>         // explicit or implicit, "main"
3 <print the string>       // definitely needed
4 <end the printed line somehow>
5 <end the method>         // as in line #2
```

# Example program

This program will just print out a couple of messages.
Don't worry if you don't know what each line does; that's why you're here!

```
1 public class Surprise {
2     public static void main(String[] args) {
3         System.out.println("Boo!");
4         System.out.println("Argh!");
5     }
6 }
```

# Compiling and running

After making the text file, a.k.a. “the .java file”, we can compile it by invoking the java compiler.
On the command line (or “terminal”) you do this:

```
~> javac Surprise.java
```

And *run* it like this:

```
~> java Surprise
Boo!
Argh!
```

☺

When you type `java` and then the name of a program on the command-line, you’re invoking the Java Virtual Machine (JVM) to run your program. The JVM is itself a program, which enables java programs to run on many different computer platforms.

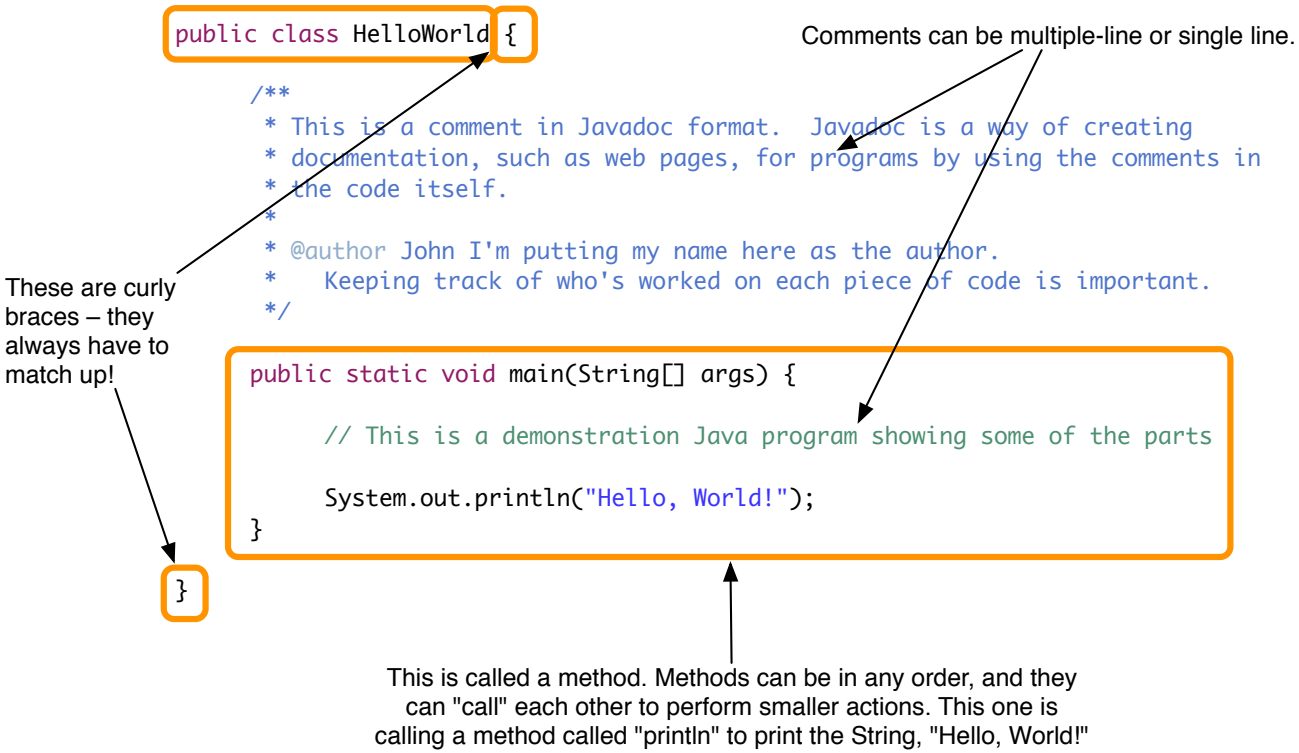
# Hello, World! in Java

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

- What you need to observe here is the text on line 3. That is instructing the Java Virtual Machine to print the right message when the program is run.
- For the moment note that the *name* of the program is given after `class` on line 1, and that you must have lines 2, 4 and 5 as they are.

# Anatomy of the Hello World program



# Java syntax

In order for the compiler to turn your code into a working program, the code has to obey a lot of syntax rules. You will pick up many of these as you go along, but let's just list a few now too:

- Some words are *reserved* — e.g., `public`, `static`, `void`, `int`, `float`. You can't use these for variable names.
- Variable names can't begin with numbers.
- Blocks of code are delineated with braces `{ }`
- Expressions are delineated with parentheses `( )`
- Array items are accessed with brackets `[ ]`
- Statements should end with semi-colons `;`
- Strings are limited by double quotes `" "` (not single ones or curly ones)

It's very easy to get things wrong. Don't worry. Most things are fixable. DIVE IN!

# The important bits in Hello, World

The main part that you have to worry about and understand is the line

```
3       System.out.println("Hello, World!");
```

which does the real work. `System.out` is a special variable, an *object* of a *class*, that has a *method* defined for it called `println`. It's the `println` method that actually prints a string of characters to the console.

`println` also puts a newline at the end of whatever it prints, so you don't have to add a newline character, '`\n`'.

You don't have to remember all this new terminology now

# Making a Program Go

In order to make a Java (or C, C++, but not Python) program actually do anything you have to *compile* it. This is a process done by a *compiler* that converts your human-readable (Java) code into machine-readable *byte code* or *machine code*.

To *compile* a single Java file, e.g., `HelloWorld.java`, you type:

```
> javac HelloWorld.java
```

(followed by a Return) and that should produce no output, which means there aren't any errors detected by the compiler.

This makes a *class file* for your program, which is where the bytecode is (all being well).

In this case the class file would be called `HelloWorld.class`, and would *not* be human-readable (nor can I represent it here on a slide).

# Running it

To *run* the program you type

```
> java HelloWorld
Hello, World!
```

where it's assumed that `HelloWorld.java` has a proper `main` method in it.

# Testing Code

We will spend more time on this later, but for now, just try to remember that you *cannot* guarantee your code will work!

- Not the first time
- Not for every case
- Even if it compiles beautifully

So you need to run your program many times with different inputs and make sure it works. *Testing is really important.*

# Syntax and Logic and Compiler

When you write programs you have to write them in such a way that they make sense to the compiler – there's a *syntax*, that is, a set of rules, that you must follow in order for this to be true.

We have the same thing in English: we can't just make up a sentence of random words and expect it to make sense.

*randomizing it's of enough words bad order the just the.*

*Tguhoh aaltpeprny if you lvaee the fsrit and lsat ltetres in pacle it's ok*

# Syntax is not Logic

However even if we have the correct *syntax* we may still not make sense. This is a very famous syntactically correct yet meaningless sentence:

*Colorless green ideas sleep furiously*

*Noam Chomsky*

In most cases your compiler will complain because of *syntax* errors, but it will also sometimes give you an error message if a variable is unused, or used before it's been initialised, or if there is part of the code that is unreachable. These are *logic* errors. The compiler won't be able to pick up on every logic error (else there would be no need for debugging!) but it will help you find simple logical errors. Pay attention to those compiler messages!

## Live Coding

*Hello, World and using Scanner to get items from the keyboard*

# Hello World

We will

1. Put this into a text file called `HelloWorld.java`:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

2. *Compile* it to turn it from human-readable text into *bytecode*
3. Run it!

# Scanner to get input from the keyboard

Next we will

1. Put this into a text file called `Echo.java`:

```java
import java.util.Scanner;

public class Echo {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please enter a word");
        String word = keyboard.next();
        int n = keyboard.nextInt();
        System.out.println("You entered the word " + word
            + " and the number " + n);
    }
}
```

2. Compile it
3. Run it
4. Answer a couple of questions about it