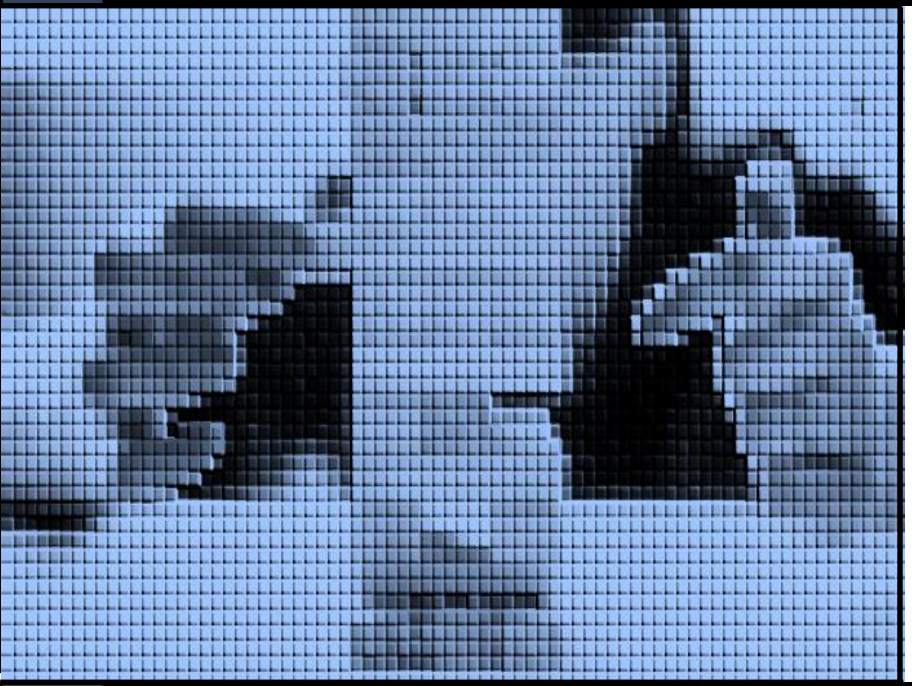
University of BRISTOL

School of Computer Science

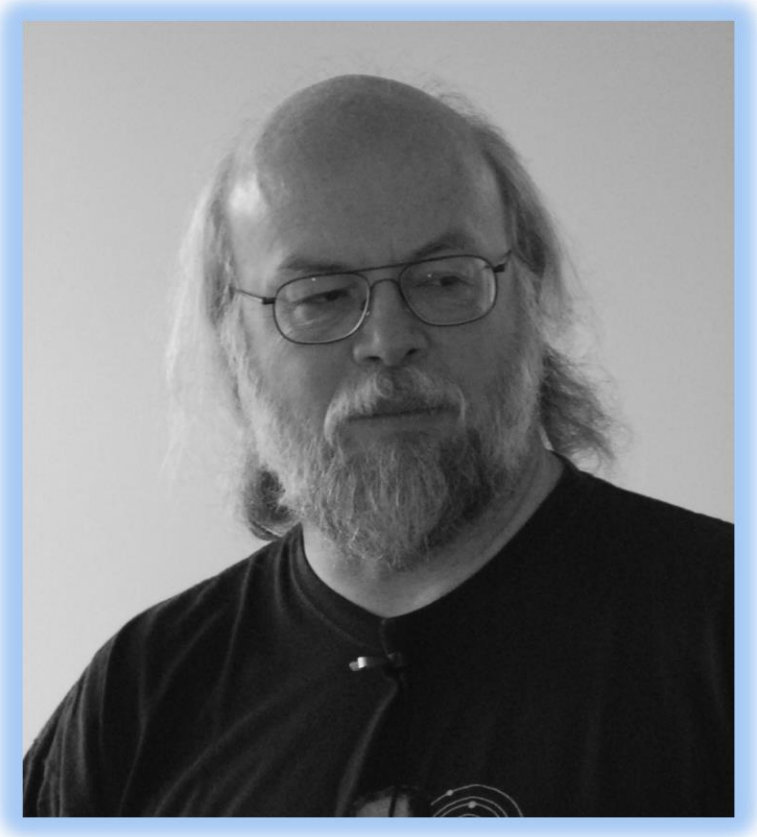
Programming in Java



Programming in Java – Lecture 01

Procedures & Programs

Sion Hannuna and Simon Lock,
based on Tilo Burghardt's C Unit

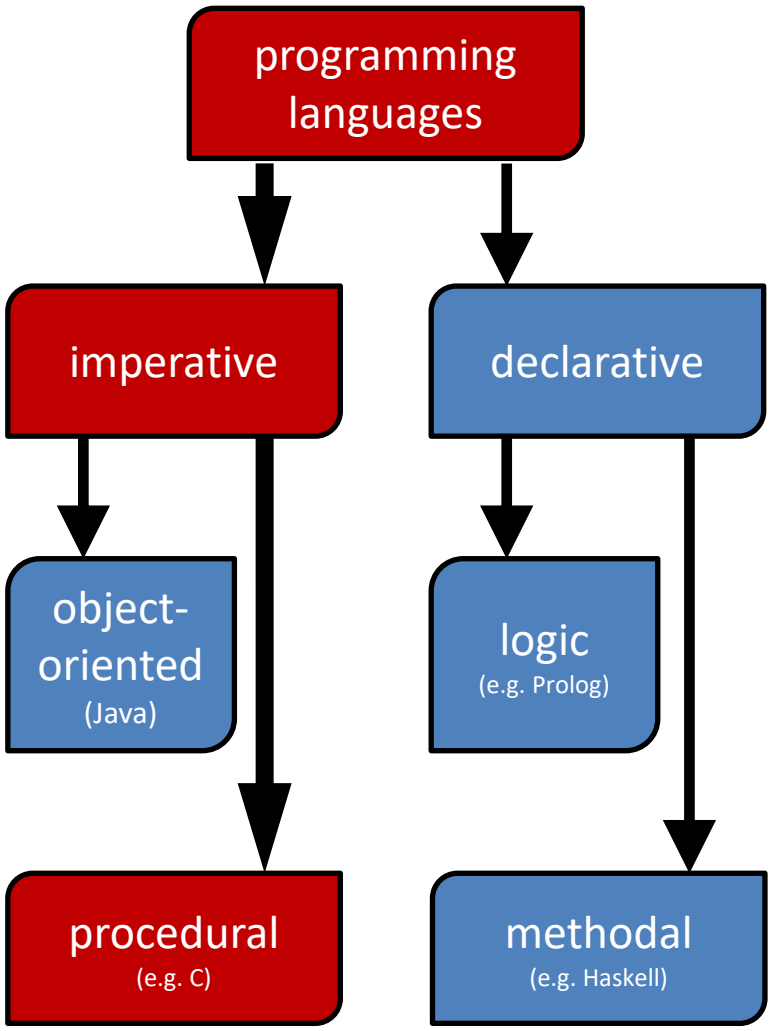


Java is C++ without the guns, clubs and knives.

James Gosling
creator of Java

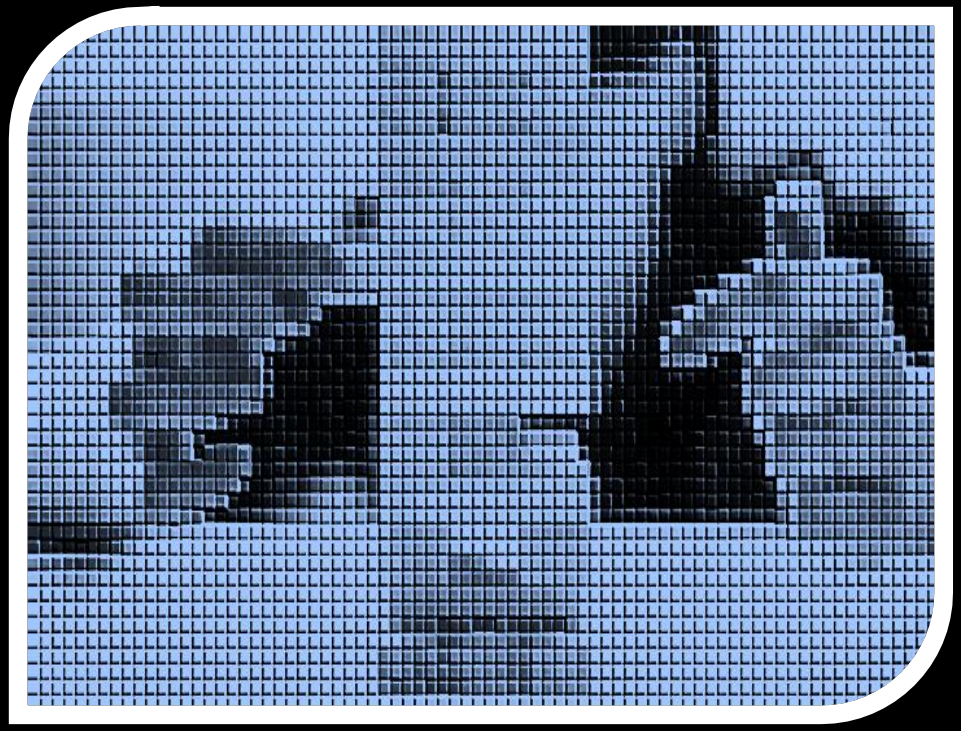
Java

Language Context



- *Java* is an *imperative* and *object-oriented* language.
- Imperative means `telling the computer *how* to do things' – using sequences of commands (i.e. *statements*) to step-by-step define and change a computer's *state* to calculate a result…
- Object-oriented means data and its associated code are coupled into well-defined, reusable chunks called *objects*.
- Java is a versatile, object-oriented programming language known for platform independence and wide applicability in web, mobile, and enterprise development..

Procedures



Methods

- Mathematical *methods* take arguments from some well-defined sets and `return` a result element from some other well defined set, for instance:

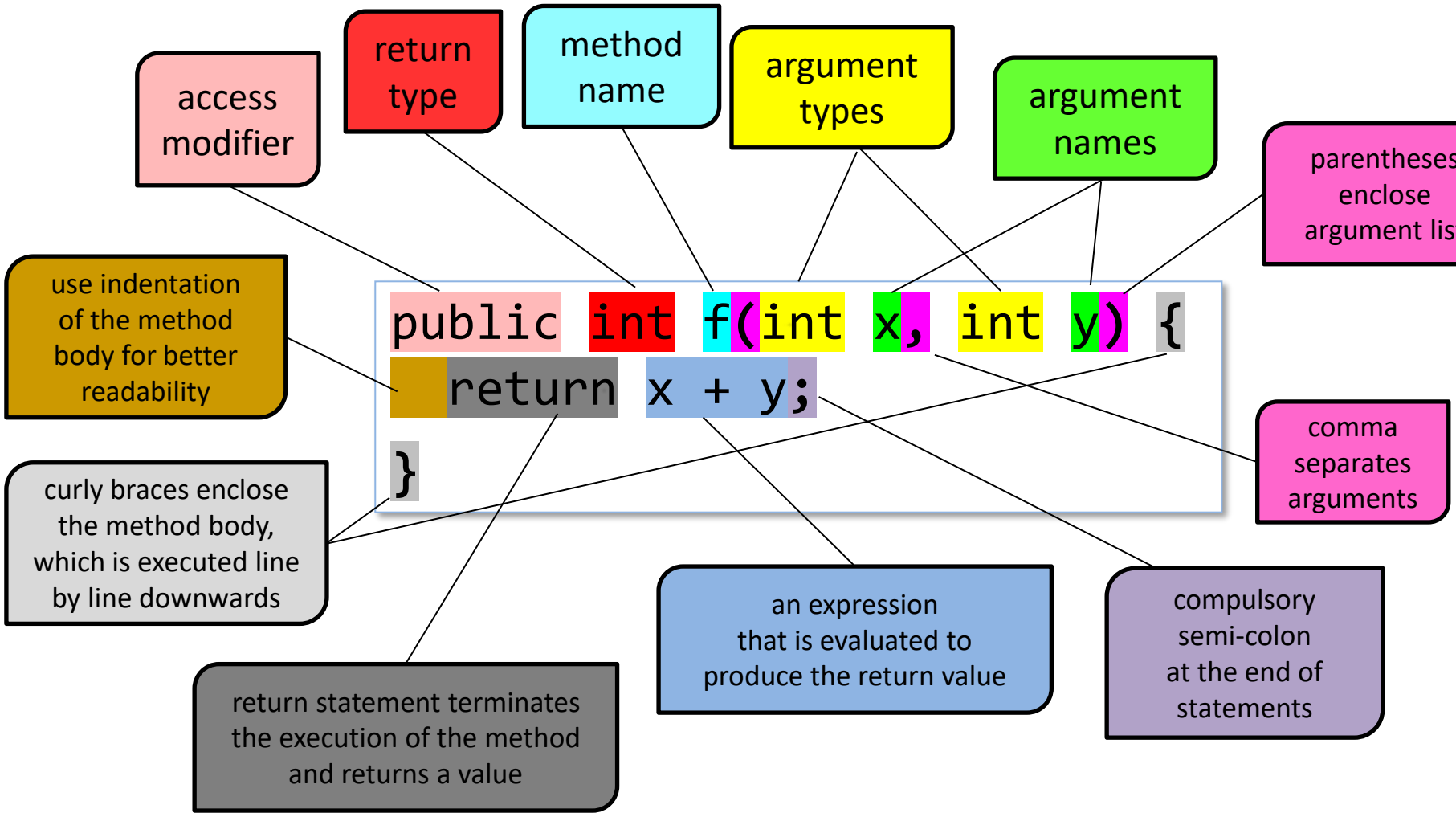
$$f: Z \times Z \to Z \qquad f(x, y) = x + y$$

- In Java, using the keyword `int` to represent small-ish integers, the above method could be written like this:

```
public int f(int x, int y) {
  return x + y;
}
```

- Java methods consist of two parts: a *signature* such as public `int f(int x, int y)` and a *body* surrounded by {…}.

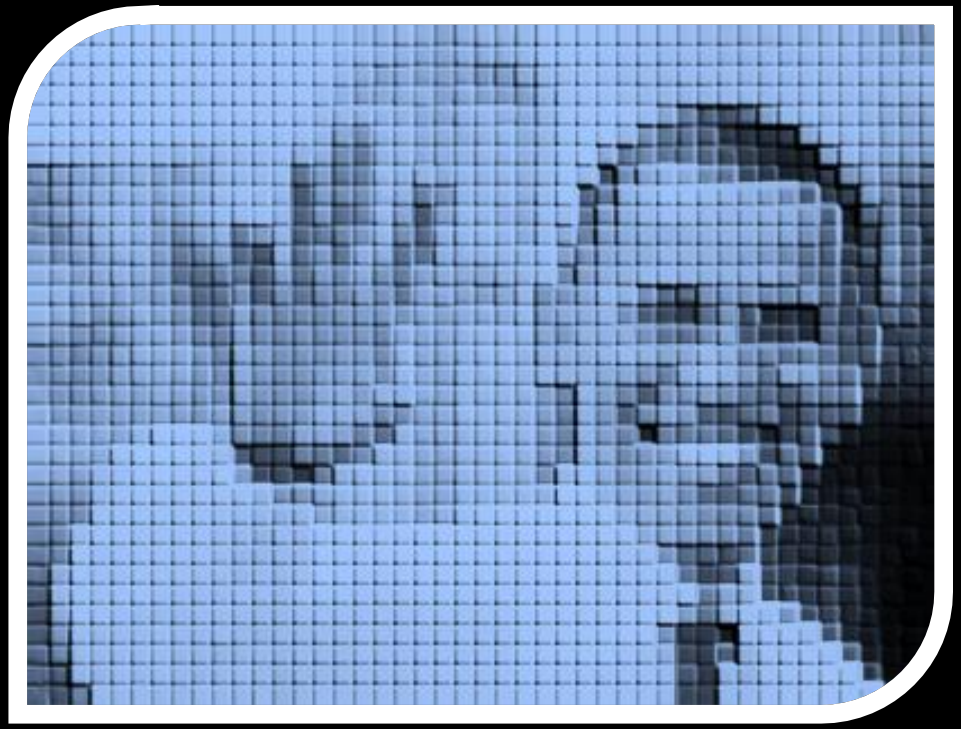
Basic Elements of a Method



Procedures

- A method that only calculates the result using its arguments is called *pure* (such as *f*).
- *Procedures* are methods, yet they are more flexible than pure methods: they may also read, write or manipulate data outside the method or interact with computer resources such as files or devices.
- Procedures in general can (but do not have to) *return* a result; they may or may not take *arguments*.
- The body of a procedure contains a sequence of *statements* that are executed line-by-line downwards.

Hello



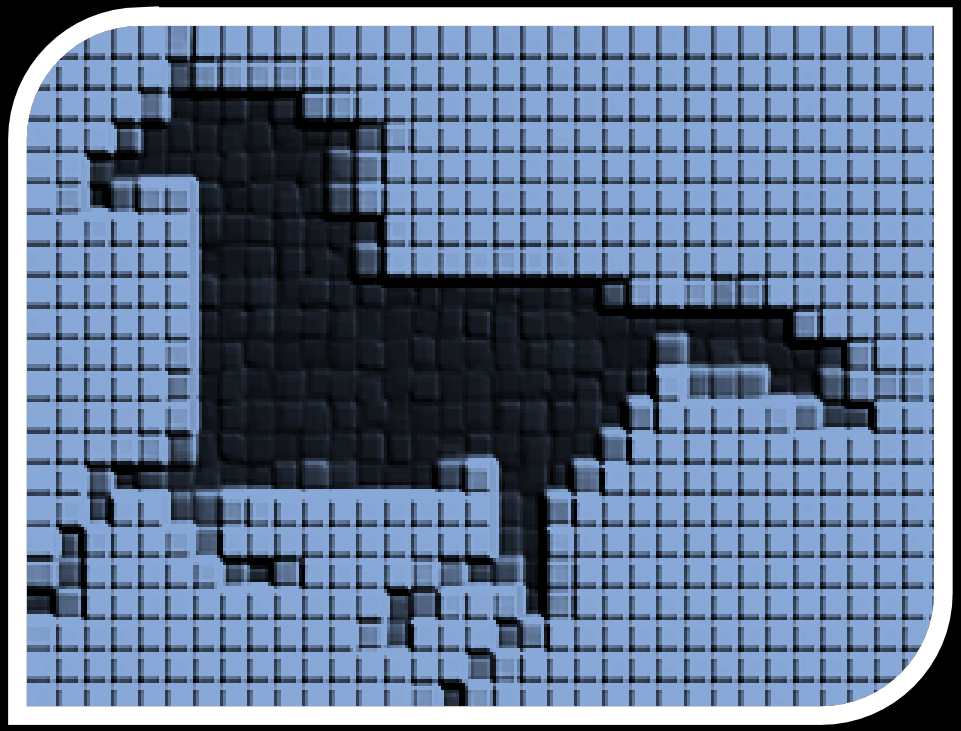
Hello World!

```
/* Example program that prints Hello World! */
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!!");
   }
}
```

HelloWorld.java

- public class HelloWorld : Defines a class named HelloWorld.
- public static void main(String[] args) : Main method, the entry point of the program.
- System.*out*.println("Hello World!!"); : Prints "Hello, World!" to the console.

Running



Running HelloWorld (in the terminal): The Java Toolchain

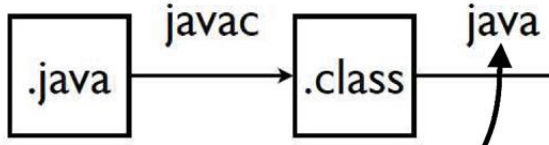
```
$ ls
HelloWorld.java

$ javac HelloWorld.java
$ ls
HelloWorld.class
HelloWorld.java

$ java HelloWorld
Hello World!
```

compilation of sources into 'bytecode' stored in .class files

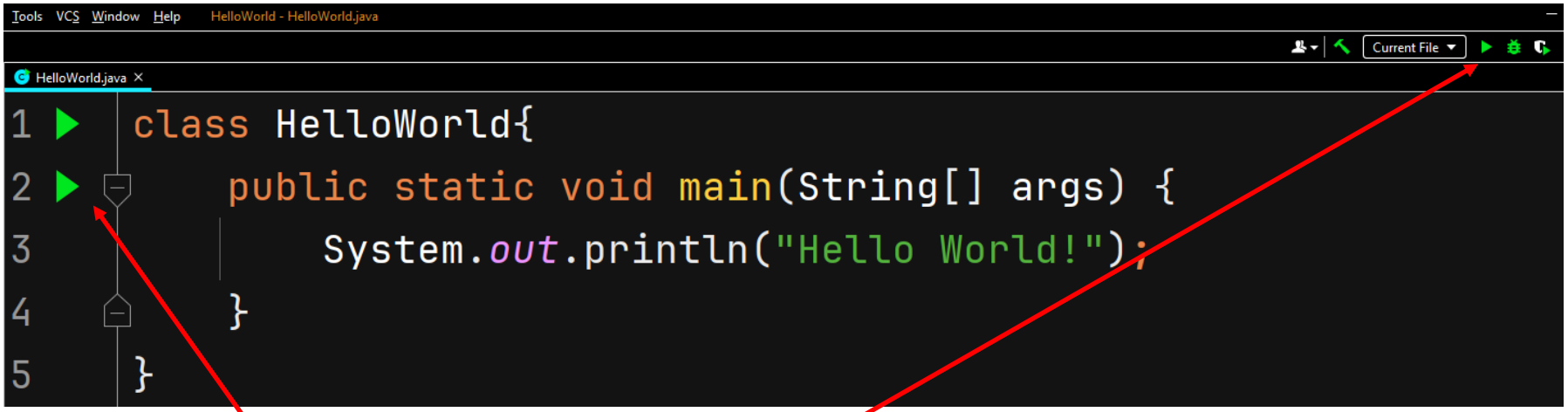
C3PO says 'Even I run Java!'



execution via interpretation of .class files by the Java Virtual Machine (JVM)

some slide content used from N. Wu

Running HelloWorld (in IntelliJ)



click the play button in Intellij (requires setup)

Source Code and Compilation

- The text of a source program (like `HelloWorld.java`) is called its *source code* (or just code).
- Java source code cannot be run directly on a system.
- The code needs to be translated by the *compiler* (`javac`) into a *java bytecode* program compatible which in turn runs on the *JVM (Java virtual machine)*, which interprets the byte code for the system on which it resides
- During compilation the *syntax* (or detailed structure) of the program is checked to be valid.
- The *semantics* (or detailed meaning) are not checked.

Compilation and Running

- To compile and then run `HelloWorld.java` via a Linux terminal, navigate to its directory and then just type:

```
$ ls
HelloWorld.java

$ javac HelloWorld.java

$ java HelloWorld
Hello World!
```

Comments

Let's dissect the Hello World program:

- The first line is a *comment*.

```
/* Example program that prints Hello World! */
...
```

- Comments and blank lines are for human readers only, they are ignored by the computer.
- Use comments before a program or more complex procedure to *explain succinctly* what they are for.
- Comments `/*…*/` can run over many lines, alternatively, comments `//…` run over one line only.

Procedure Calls

- The rest of the program is our familiar `main` method.

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

- public class HelloWorld: Defines a class named HelloWorld.
- public static void main(String[] args): Main method, the entry point of the program.
- System.out.println("Hello, World!");: Prints "Hello, World!" to the console.