Programming with Java (2/2)
~ Tutorial ~

Baptiste Vergain[1]
INFO0062 - Object Oriented Programming

2023

[1] Greatly inspired by the work of former assistant, Jean-François Grailet.

Important remark

If you are following this tutorial, this means that you went through the Programming with Java (1/2) tutorial dealing with the installation of Java on your machine. If not, please follow the first part before starting this one.

Regarding IDE

There are two main approach when programming in Java:

- either your use an IDE (integrated development environment) which are very complete software that let you edit code and set up complex project, and even compile and run them for you,
- or you simply use a code editor of your liking (Visual Studio Code, Emacs, Atom, Vim, ...) and then, compile and run your programs by hand using command-line.

We highly encourage you to chose the latter solution, as it will allow you to have a better understanding of what happens during the compilation or at runtime, and will probably help you debug your code at some point. For this reason, this tutorial will solely focus on dealing with your code *via* command-line interface (CLI).

Getting used to CLI

- **Please note that commands can greatly differ between Windows and Linux/MacOS.**
- Commands under Linux/MacOS are well documented online:
  - A tutorial is available here.
- Windows' command prompt tutorials are usually technical, but exceptions exist.
  - A tutorial is available here.
- For this course, all you really need is to be able to browse your filesystem.
  - You can do the rest (managing/creating source files) the usual way.
- Most essential commands:
  - **Linux/MacOS**: `cd`, `ls`
  - **Windows**: `cd`, `dir`

A very simple program (I)

- To show how to use `javac` and `java`, we are going to use a simple example.
- Next slide gives you a program simply displaying "*Hello world !*".
- The details of implementation are left for later.
  - E.g., what the `args` of the `main()` can be used for.
- For now, just copy the code in a `HelloWorld.java` file.
- For convenience, place it in a new folder you will use only for this course.
  - Name it `INFO0062`, for instance.
  - Under Windows, you can place it in your `Documents` folder.

A very simple program (II)

HelloWorld.java

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

Compiling and running our `HelloWorld` class (I)

- Open your terminal (Linux/macOS) or command prompt (Windows).
- Place yourself in the directory containing your code.
  - macOS/Linux: `cd` (*Change Directory*) followed by the path to said directory
  - Windows: `cd` (idem) followed by the path to said directory
  - In both cases, use "`./`" to denote the path w.r.t.[2] your current directory
- You can also use one command to check the content of the current directory.
  - macOS/Linux: `ls` (with `-l` for more details)
  - Windows: `dir`

```
cd ./Documents/INFO0062
```

[2] with respect to, FR: vis-à-vis de

Compiling and running our `HelloWorld` class (II)

- To compile `HelloWorld.java`, run this command:

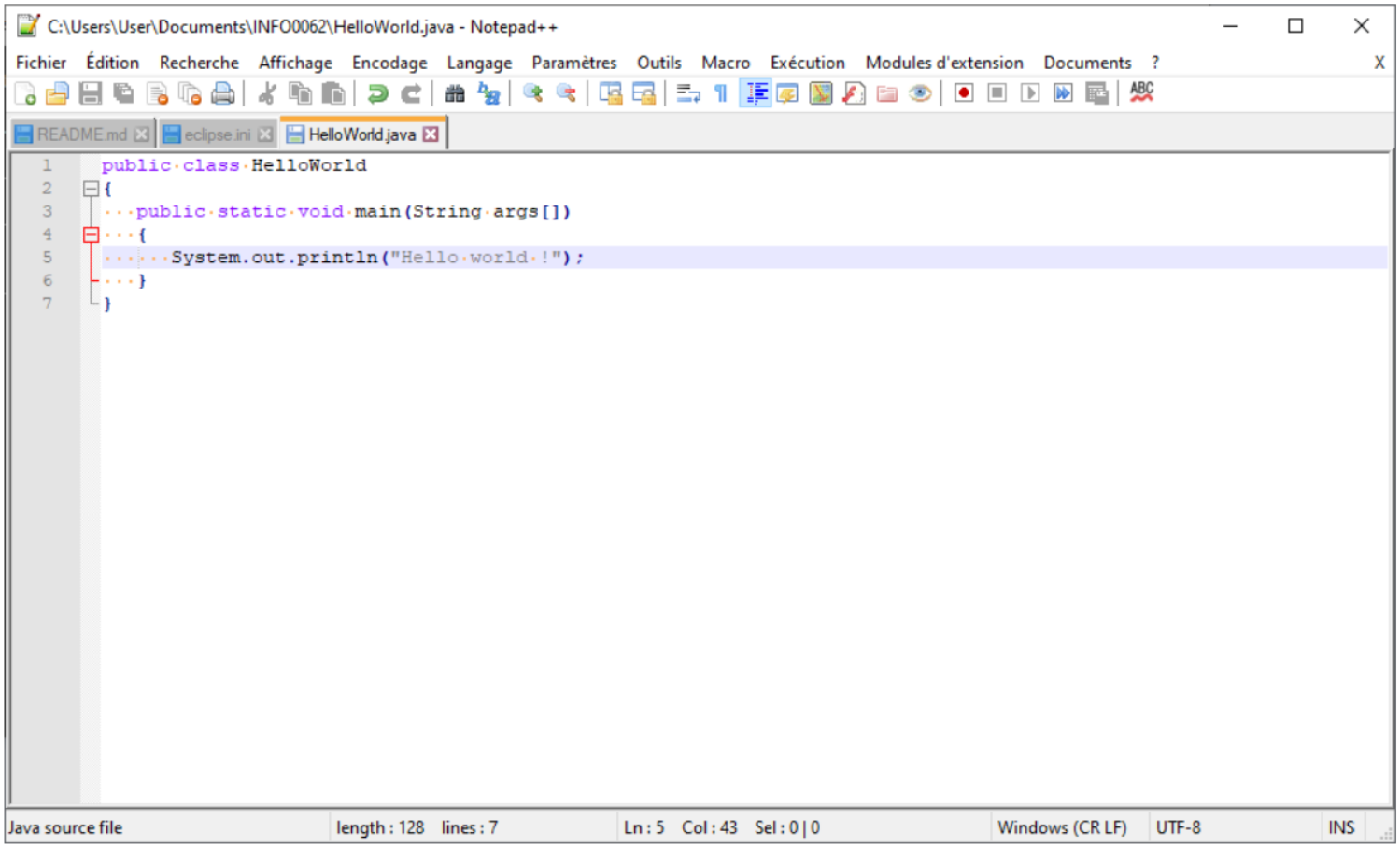
```
javac HelloWorld.java
```

- No message should be displayed if compilation worked.
- If compilation is successful, you can run it with:

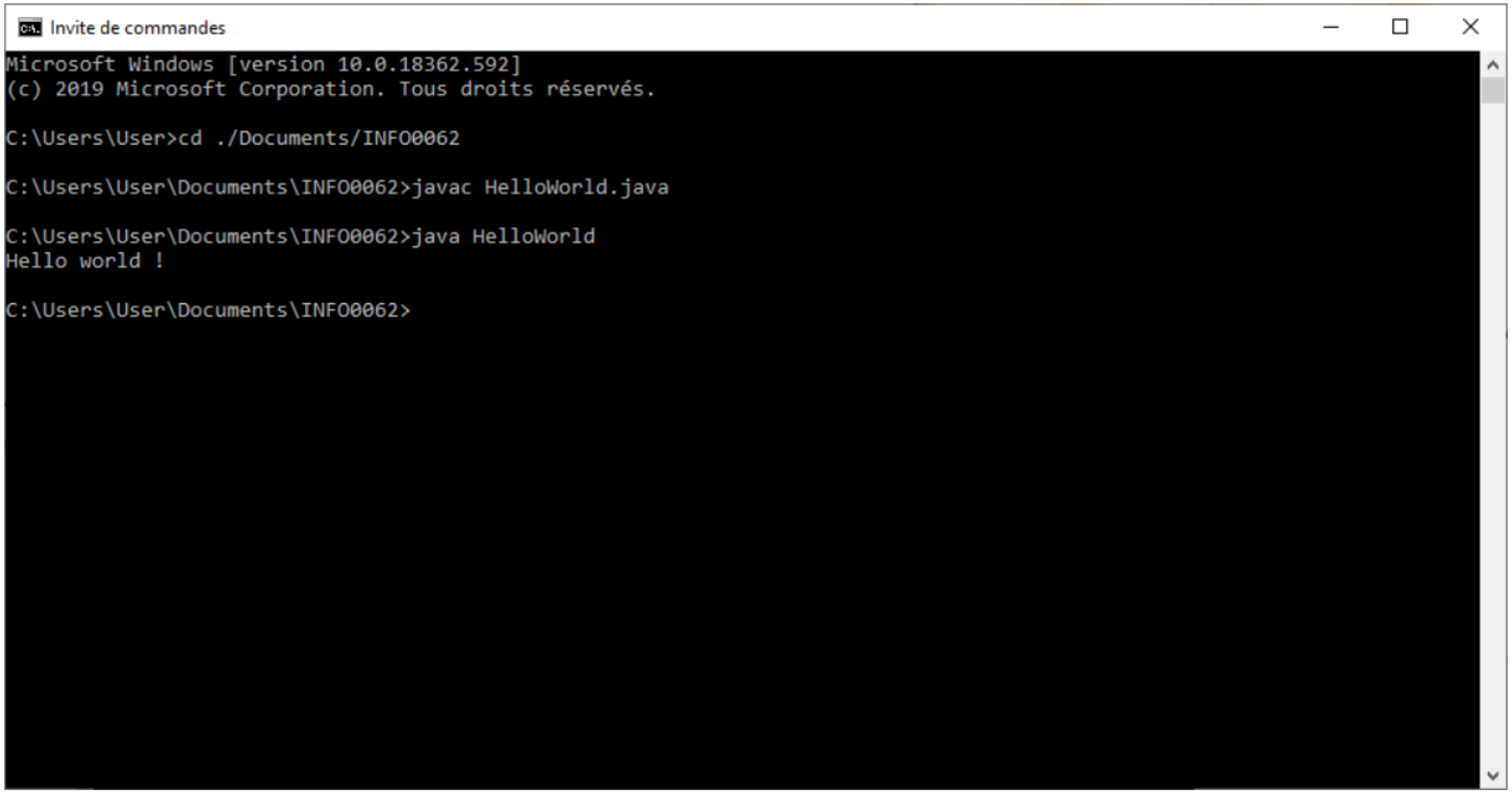
```
java HelloWorld
```

- That's it !

Compiling and running our `HelloWorld` class (III)



Compiling and running our `HelloWorld` class (IV)



Compiling and running a Java program made of several classes

- At compilation, you must provide all source files to `javac`.
- Upon running the program, you only need to provide the main class.
- For example, let's compile exercise 2 from the first exercise series:
  - `GroceriesList.java` (main class)
  - `Groceries.java`, `Item.java` (additional classes)
- Run these commands:

```
javac Item.java Groceries.java GroceriesList.java
java GroceriesList
```

Compilation options (I)

- `javac` places `.class` files in the same folder as the `.java` files.
- If you want to separate both kinds of files, you can create an output directory.
- Use the `-d` flag of `javac` to do so.
- If we want `MyClass.class` in a `bin/` folder:

```
javac -d bin MyClass.java
```

- To run a .class file located in a `bin/`, you will have to type

```
java -cp bin MyClass
```

- `-cp` is another flag of `javac` used to specify where to look for `.class`.

Compilation options (II)

- Note that you can provide paths to `javac`.
- For instance, if you place your `.java` files in a folder `src/`:

```
javac -d bin src/MyClass.java
```

- In fact, placing `.java` in `src/` and `.class` in `bin/` is a common practice.
  - Both kinds of files are clearly isolated from each other.
  - Home folder can consist of documentation, usage examples, etc.
  - This convention is followed by Eclipse IDE to manage projects.