PW 01 : Introductory Java and Eclipse

Exercise 01: Getting Started with Java

Part 1: Install the Developer Tools

Developers who are new to Java often wonder what is the difference between the Java Virtual Machine (JVM), the Java Development Kit (JDK), and the Java Runtime Environment (JRE)?

JDK Java Development Kit allows you to create Java programs that can be run by the JVM on any operating system. It is important to know that Java is a pseudo-compiled language, the result of the compilation of a program by jdk is a bytecode file with a .class extension.

Java Runtime Environment or JRE creates the Java virtual machine and ensures that dependencies are available to Java programs.

Java virtual machine is the component of Java platform that executes programs (the conversion of byte-codes into machine-specific code).

Each time the developer downloads a JDK, it includes a version-compatible JRE, and that JRE itself includes a default JVM. However, it is possible to download the JRE independently of the JDK.

- ***javac*** is the compiler provided in the JDK for compiling a .java source file into a language called *bytecode* and saves the result to a file with a .class extension.

- the *bytecode* is not directly executable. It must be interpreted by a Java virtual machine that translates it into a language suitable for the operating system. Java program provided in the JDK launches a virtual machine to interpret a compiled class.

Link to download JDK: https://www.oracle.com/java/technologies/javase-downloads.html

Part 2: MS-DOS

MS-DOS (**M**icrosoft **D**isk **O**perating **S**ystem) is an operating system developed by Microsoft for the IBM PC first, then for PC-compatible PCs. It is a system that operates in real, single-task, single-user mode, and is equipped with a command-line interface by default.

Here are some examples of commands:

CD Allows you to move from one directory to another. (e.g. cd directory).
**CD ** Allows you to access the root of a drive. (e.g. c:>cd).
MKDIR Creating a folder. (e.g. mkdir PW_OOP)
RMDIR Delete a folder. (*ex : rmdir PW_OOP*)
DIR Lists the contents of the current directory. (*ex : dir*)

DEL Delete a file. (*ex : del c:\ MaClasse.java*)
EXIT Closes the MS-DOS window. (*ex : exit*)

Part 3: First Program with Java

1. Open a text editor, such as Notepad or NotePad, write the above program in the text editor.

```
File  Edit  Format  View  Help
public class MaClasseJava
{
  public static void main(String[] args)
  {
    System.out.println("Bonjour");
  }
}
```

2. Save the file under MaClasseJava.java in a directory named PW_OOP which is located in C. Since MaClasseJava is the name of our class, it is essential to save this file under the same name MaClasseJava.java (another name will cause an error when compiling).
3. Open the Command Prompt (CMD) and compile the program using javac, type the following command and press Enter.

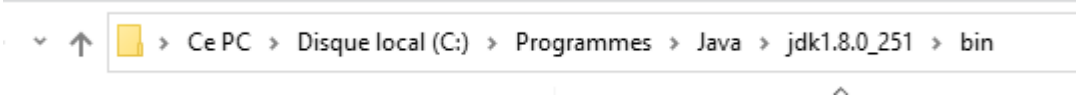
```
C:\TP_POO>javac MaClasseJava.java
```

4. You may get this error:

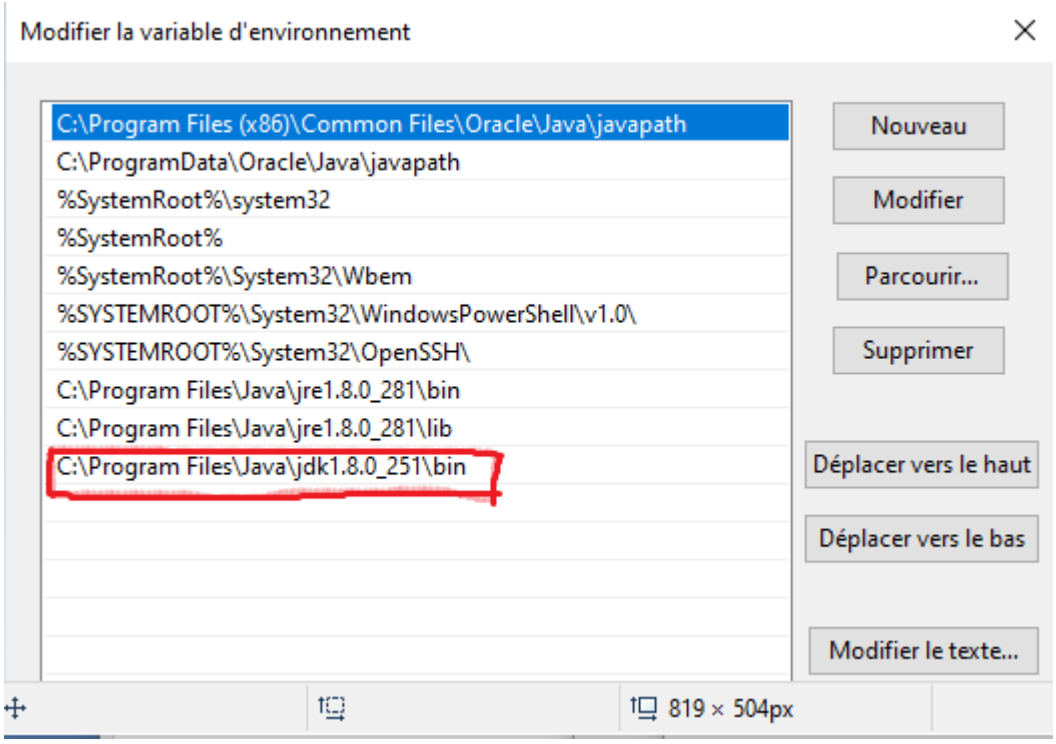
```
C:\TP_POO>javac MaClasseJava.java
'javac' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.
```

5. Explain why ?
6. Install Java (64-bit Java for 64-bit Windows and 32-bit Java for 32-bit Windows).
7. Go to the location where you installed java on your system and locate the bin directory (which contains the java and javac commands).

Note: *In partition C:*
. The Programs folder: contains 64-bit programs.
. The Programs folder (x86): contains 32-bit programs.



8. Create a new PATH environment variable by adding the path.



9. Restart your computer.
10. Check the Java version with the command:

```
C:\>java -version
java version "1.8.0_281"
Java(TM) SE Runtime Environment (build 1.8.0_281-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode)
```

11. If you still have the same compile error, check that JDK has been installed correctly. Otherwise, in the bin directory checks that there is a binary named javac.exe. If it does not exist then reinstall Java properly.
12. Otherwise, if the compilation was successful, a **.class** file MaClasseJava.class is created, this file consists of byteCode. Check the PW_OOP directory.
13. To run the program, you need to call the Java interpreter followed by the name of the class containing the main method.

```
C:\TP_POO>java MaClasseJava
```

14. What is the result?

Part 4: Command Line Arguments

A Java program must have at least one class containing a main function definition, named main ; the execution of the program begins with this function. It is this function that will eventually invoke the other functions. Its minimal definition is as follows:

```
public static void main(String[] args) {}
```

The main() function can access the command line passed to the command interpreter to execute the program. This command line is used to start a Java Virtual Machine, to designate the main class, and to specify the value of a number of program parameters. These parameters are represented by an array of strings, conventionally called args, which is the only parameter of the main() function. When main() is called, the elements of this array, args[0], ..., args[args.length-1], are initialized by the strings (space-separated words) appearing on the command line after the main class.

Q1 : Modify the class MaClasseJava so that the program displays: Hello Algeria. (Algeria is placed on the command line).

Q2 : Modify the MyJavaClass class so that all arguments passed by the command line are written to the program output, one per line. If the arguments are "Hello" and "Hi", the program should write:

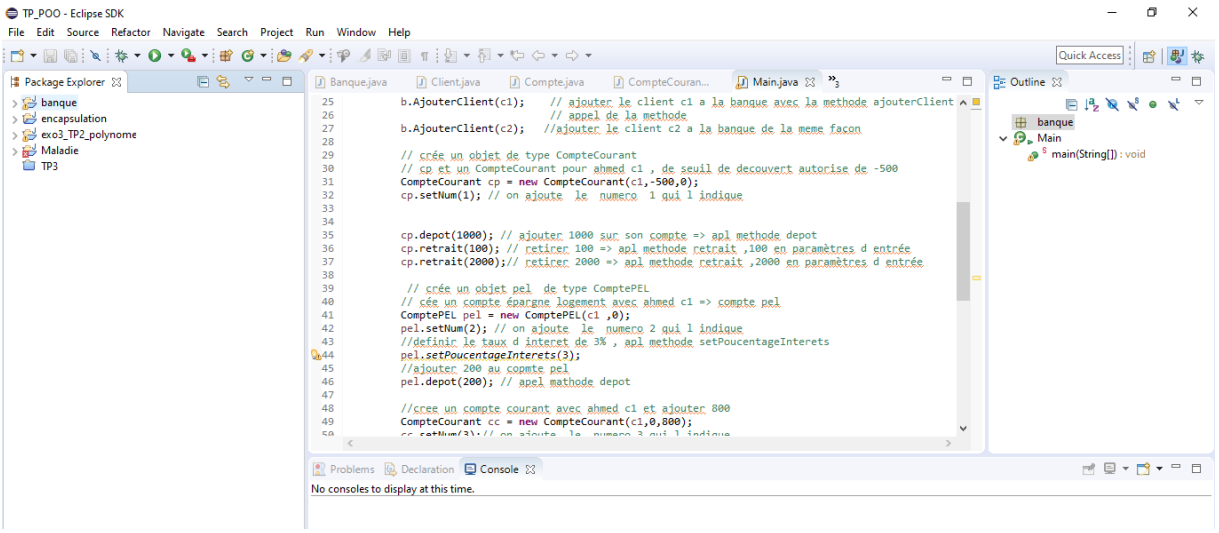
Hello
Hi

Q3 : It is often necessary to convert args[i], which are always of type String, to other data types. Integer.parseInt(s) converts the argument s, a string that is the notation of an integer. Modify the program so that it adds three integers read from the command line.

Exercise 02: Getting started with Eclipse

Eclipse is an Integrated Development Environment whose goal is to provide a modular platform for carrying out computer developments.

1. Ouvrez l'IDE Eclipse, on vous demandera de choisir l'espace de travail dans lequel vous allez trouver vos projets. Si vous n'avez pas encore créé d'espace de travail, Eclipse le créera pour vous.
2. Close the welcome window that appears.
3. You are currently in your workspace. The following figure shows you the different views available.
4. Create a new Java project. To do this, tap File → New → Java Project.
5. Créez un nouveau projet Java. Pour cela, appuyez sur File → New → Java Project.
6. Write the name of your project as shown. Keep the default settings, and press Finish.
7. 8. In the workspace, under the Packages tab, you will see that the project is created, and it already contains a src directory (which should contain all the source files you create), and JRE System Library, which is used to compile your code.



8. To create a package under the src directory, click on it, then on the icon, or right-click on src, and choose New -> Package. Choose a name for the package.
9. To create a class:
   - Click on the package that should contain the class then on the icon  , or right click on the package, and choose New -> Class. In the window that appears, choose the name of the class.
   - If the class is not defined in a package (what is not recommended), Repeat the above operation from the src directory.
   - You can automatically generate the main method by clicking on the box **public static void main(String[] args).**
10. The generated class will appear under the package you have chosen. Double-click on it to modify its code in the editing part. You will see that a skeleton of the class is proposed to you. You will only have to finish the rest of the code.

11. With Eclipse, you don't need to explicitly compile your code: compilation is done in real time. In addition, syntax errors will be displayed while writing the code, with suggestions for corrections.

New project: Hello Algeria

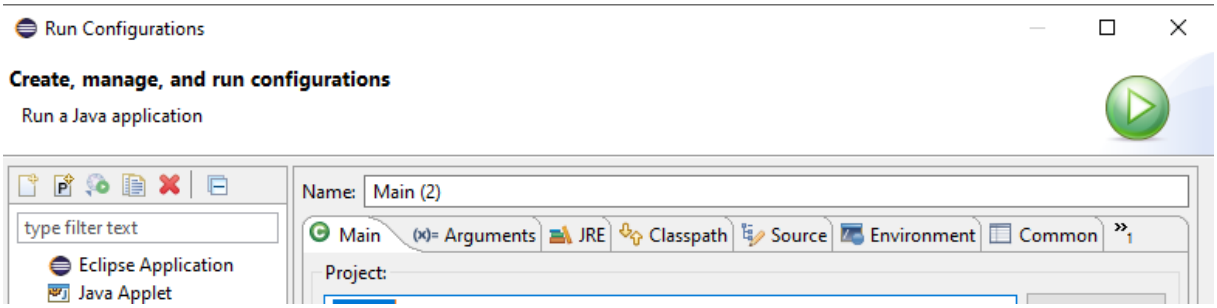
I. Hello Algeria – Simplified version

1. Create a new Algeria project as indicated in the previous part. Create a package named helloDZ, containing a HelloAlgeria class which contains a main method.
2. In the main method, write "System.out.println("Hello Algeria!");"
3. Run your program by clicking directly on the icon  . The display appears at the bottom, under the Console tab.

II. Hello Algeria – Adding arguments

1. Create in the same Algeria project, a new package named argsHelloDZ, containing a HelloAlgeria class with a main method.

1. In the main method code, write: "System.out.println("Hello "+args[0]+"!");"
2. To set arguments to the class, click the arrow next to the run icon  and select Run Configurations, or click on the class you want to run, and go to Run -> Run Configurations... The following window will appear.



3. On the left side, select Java Application, then click on the icon  (top left), to add a new configuration. You will see that your HelloAlgeria class has been added under Java Application.
4. Select the Arguments tab (above outlined in red) and, in the Program Arguments box, simply write your name.
5. Then click on Run. You will see in the console the display "Hello your_name!".