Introduction à Java

## Un langage compilé

Java est un langage de programmation généraliste orienté objet. Nous n’allons pas copier Wikipedia ici (voir <https://fr.wikipedia.org/wiki/Java_(langage)>), mais relever un point important par rapport aux langages que vous avez rencontrés jusqu’à maintenant (HTML, JavaScript, PHP). Java est un langage **statiquement typé** et **compilé** :

les variables sont définies avec un type de données (une variable “String” ne pourra contenir que des valeurs de type “String”) ;

les fichiers sources devront être compilés (sorte de traduction) en fichiers “classes” avant d’être exécutés.

Cela oblige à bien définir les types des variables et à subir l’étape supplémentaire de compilation mais apporte plusieurs avantages :

* certaines erreurs sont détectées lors de la compilation et non lors de l’exécution (le code est plus “sûr”) ;
* des optimisations plus poussées peuvent être mises en œuvre ;
* les classes générées sont indépendantes du système.

Mais… même dans le développement, tout n’est pas binaire ! Même si cela est moins visible, lors de l’exécution d’un code JavaScript (on parle alors de code interprété car “lu” au fur et à mesure des besoins), ce code peut être localement compilé pour être exécuté et accéléré. Le principe est juste moins poussé que pour le langage Java et la différence est historique, les langages interprétés étaient alors exécutés comme nous le faisons en lisant le code ! De même, des extensions à JavaScript peuvent apporter du typage statique. Et dans l’autre sens, beaucoup de frameworks rendent l’opération de compilation en Java quasi invisible… On peut donc relativiser ces différences !

## Hands on!

### Installation du JDK

Pour pouvoir utiliser le langage Java (exécuter des programmes Java), il faut avoir installé l’environnement d’exécution. Cet environnement apporte la “machine virtuelle” (le programme qui exécute le code compilé), mais aussi les librairies de base (afficher un texte à l’écran, ouvrir un fichier, …). Le JRE (Java Runtime Environment) suffit pour cela.

Dans notre cas, nous allons également créer des programmes en Java : il nous faut donc installer le JDK ­- Java Development Kit.

Des extensions existent pour le JDK, selon les librairies de base installées notamment. Nous allons utiliser JavaSE (Standard Edition), version 8 : suivez les instructions depuis <http://www.oracle.com/technetwork/java/index.html>

### Un IDE ?

Différents IDE (ou même un simple éditeur de texte) sont utilisables pour écrire des programmes Java. Les références pour Java sont Eclipse (<http://www.eclipse.org/>), IntelliJ IDEA (<https://www.jetbrains.com/idea/>), Netbeans (<https://netbeans.org>)…

Nous n’entrons pas dans le débat des avantages des uns et des autres, mais ces IDE sont en général très complets, permettent le développement de gros projets et sont même parfois imposés dans des équipes pour l’utilisation de leurs fonctionnalités spécifiques. Le revers de la médaille est une courbe d’apprentissage plus raide et des mécanismes qui cachent parfois trop ce qui se passe pour un apprentissage.

D’autres IDE plus simple, comme Visual Studio Code peuvent aussi être utilisés, mais nécessitent l’installation de plugins Java.

Concrêtement :

* Vous pouvez utiliser celui que vous voulez !
* Vous trouverez en général des tutoriels qui utilisent les IDE cités ci-dessus

### Hello… World!

Comment y échapper ? Nous allons même en faire plusieurs versions ! L’objectif est d’appréhender les bases du développement en Java ; nous rentrerons dans le langage et ses principes juste après.

#### Hello 0

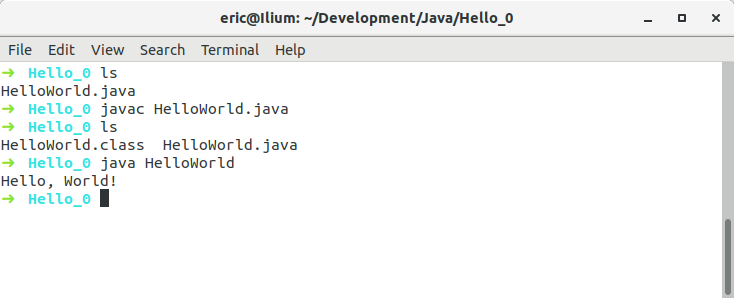
Et oui, Java est comme la plupart des langages de programmation “zero-based” (<https://en.wikipedia.org/wiki/Zero-based_numbering>).

Créer un répertoire Hello\_0, avec un fichier HelloWorld.java qui contient :

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

Ce fichier est la définition d’une classe HelloWorld, qui a une méthode statique (attachée à la classe et non aux instances de cette classe) main (nous passons sur les arguments) et qui utilise la méthode statique println de la classe System.out pour faire l’affichage tant attendu. Les principes ne sont pas vraiment différent de la POO en PHP.

Pour exécuter ce programme, il faut d’abord le compiler. Ouvrez un shell (cmd, bash, …) et positionnez-vous dans le répertoire du fichier. Compilez et exécutez le programme comme ceci :



La commande javac (Java Compiler) produit le fichier compilé HelloWord.class. La commande java HelloWorld exécute ces étapes :

* cherche la classe HelloWorld dans l’environnement (par défaut dans le répertoire courant => définie dans HelloWord.class
* charge cette classe
* cherche la méthode par défaut, main et l’exécute

#### Hello ++

L’opérateur de post-incrément est disponible en Java, comme en PHP ou JavaScript !

Créez l’arborescence de répertoires Hello\_1/src/main/java/hello. En utilisant Bash, vous pouvez le faire avec la seule commande

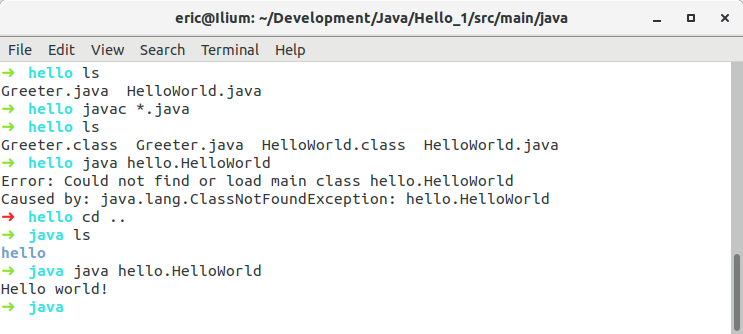
|  |
| --- |
| mkdir -p Hello\_1/src/main/java/hello |

Créez dans le répertoire hello les deux fichiers Greeter.java et HelloWorld.java qui contiennent respectivement :

|  |
| --- |
| package hello;  public class Greeter {  public String sayHello() {  return "Hello world!";  }  } |

|  |
| --- |
| package hello;  public class HelloWorld {  public static void main(String[] args) {  Greeter greeter = new Greeter();  System.out.println(greeter.sayHello());  }  } |

Le programme utilise maintenant deux classes et peut être compilé et exécuté comme suit :



Vous remarquerez dans ce cas que les classes indiquent qu’elles appartiennent au “package” hello - et cela correspond au répertoire hello. Lors de l’exécution, la classe ne peut être trouvée que depuis le répertoire java car la classe cherchée hello.HelloWorld (autrement dit la classe HelloWorld du package hello ne peut être trouvée que dans un répertoire hello).

Cette notion de package est très utile pour organiser son code.

## Livrable : On déménage !

Vous allez mettre en œuvre ces récents acquis sur Java. L’objectif est de produire un programme Java, disponible sur Github, avec la documentation décrivant la préparation et l’exécution.

Nous allons pour ce premier livrable, reprendre ce que nous avions développé (en java, déjà !!!), lors du module d’Algo. Nous avions travaillé sur le projet “déménagement”, que nous avions développé à l’aide de processing (projet déménagement, étape 2).

Nous allons écrire ce programme maintenant dans un environnement pur Java.

Le programme doit contenir une classe principale dont le seul but est l’exécution du point d’entrée (main).

Pour cela, nous utiliserons les ressources suivantes :

### Ressources :

* *Openclassrooms :* 
  + - <https://openclassrooms.com/courses/apprenez-a-programmer-en-java/les-variables-et-les-operateurs>
    - <https://openclassrooms.com/courses/apprenez-a-programmer-en-java/lire-les-entrees-clavier>
    - https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/20757-definissez-des-conditions
    - https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/20841-apprenez-a-creer-des-boucles

## 