Algorithmes et Pensée Computationnelle

Introduction à Java

1 Objectifs du document

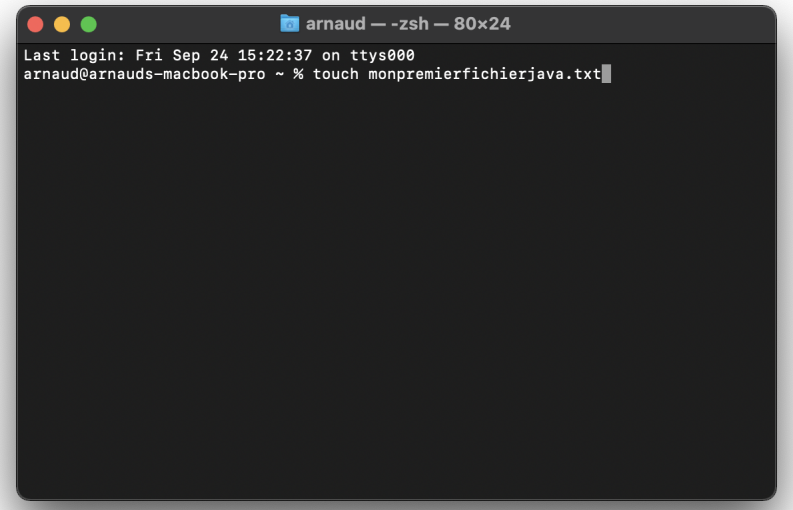
Ce document constitue un guide pour débuter à programmer en utilisant le langage Java. Les objectifs de ce guide sont les suivants :

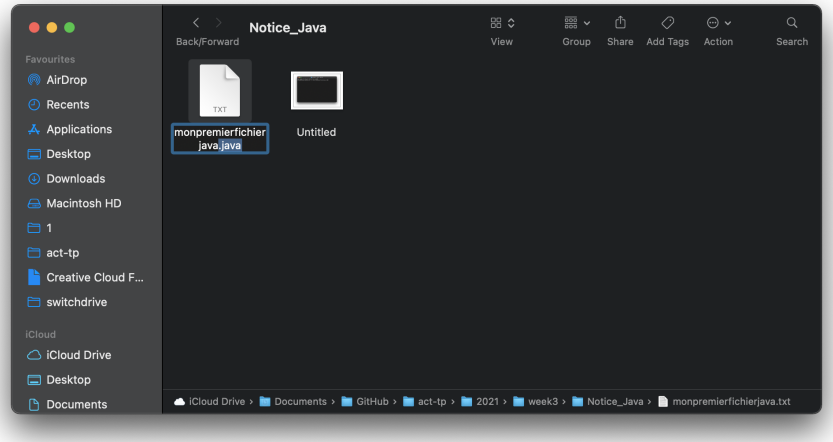
- Comprendre comment créer, éditer et lancer un programme en Java.
- Découvrir l'environnement de travail qui sera utilisé
- Se familiariser avec quelques notions de base du langage.

2 Introduction

2.1 Création d'un programme en Java

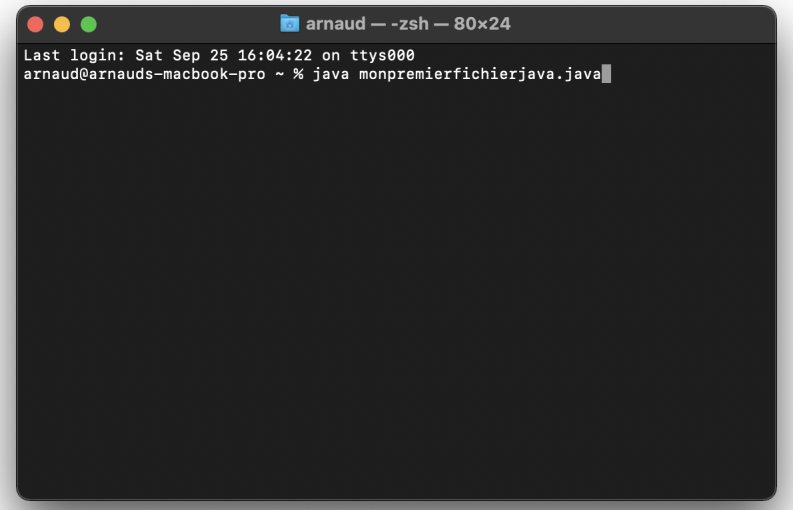
Pour commencer à programmer en Java, il est nécessaire de créer un fichier ayant une extension **.java**. Pour ce faire, il suffit de créer un simple fichier vide et de changer l'extension du fichier. La manière la plus simple est de passer par le terminal (ou invite de commande). Une fois ouvert, entrer la commande **touch monpremierfichierjava.txt**pour créer un fichier texte vide nommé **monpremierfichierjava.txt**, puis de changer l'extension du fichier directement depuis l'explorateur de fichier de .txt à .java.
Pourquoi renommer le fichier quand on peut le créer directement en .java en faisant **touch mon_fichier.java** ? Cela est une étape additionnelle qui n'est pas nécessaire.





Utiliser d'un terminal ayant un fond blanc

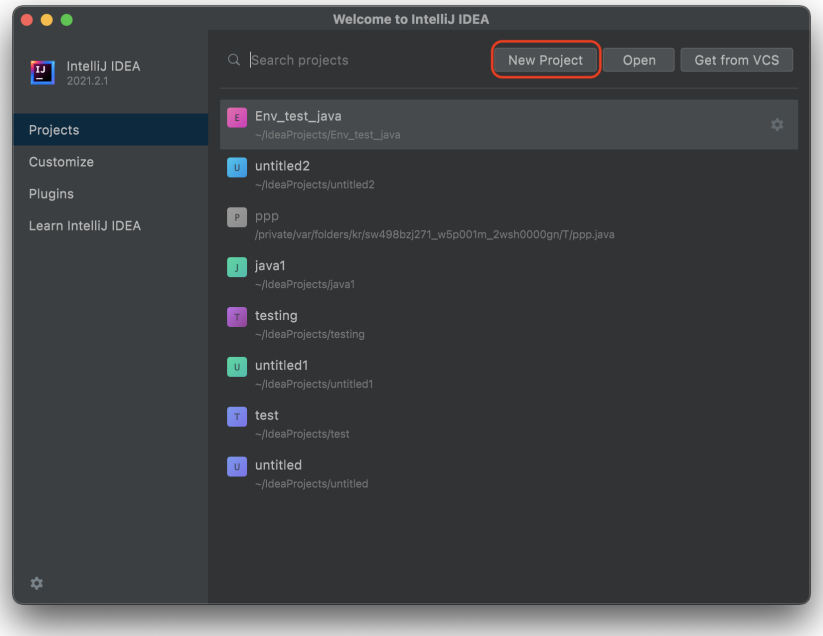
Pour éditer votre programme java, il suffit simplement de l'éditer avec n'importe quel éditeur de texte Vous pouvez ensuite lancer votre programme Java en entrant la commande **java monpremierfichierjava.java**. Ce dernier va s'exécuter dans le terminal.



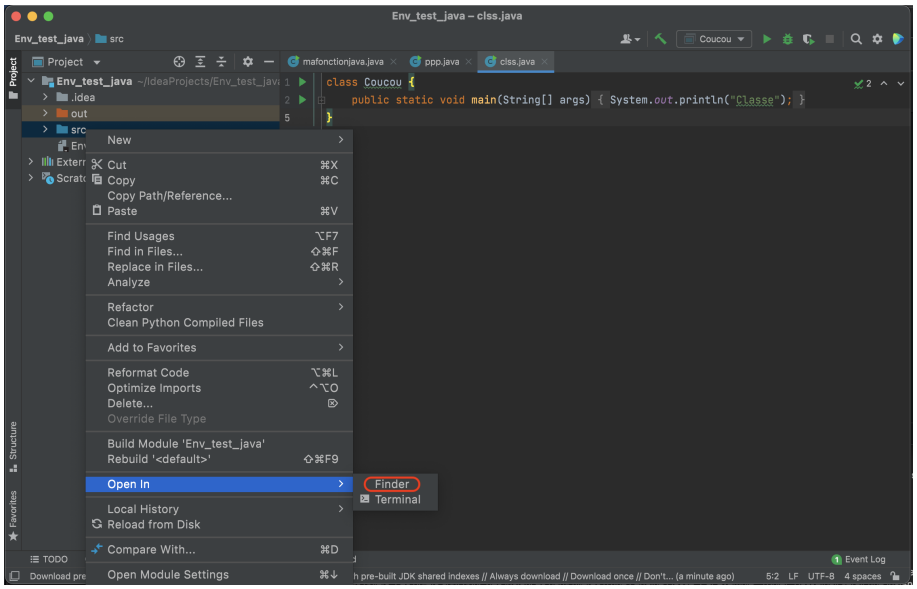
2.2 IntelliJ IDEA

Dans le cours, il vous a été demandé d'installer IntelliJ IDEA, l'un des environnements de développement le plus populaire de ces dernières années. Un environnement de développement ou IDE (Integrated Development Environnement) est un programme qui combine différents outils de développement et qui facilite le travail d'un programmeur. Pour ouvrir un fichier **.java** avec le programme, il est nécessaire de créer un nouveau projet et de copier le fichier **.java** dans le dossier contenant le projet.

Commencez par ouvrir IDEA et créez un nouveau projet.



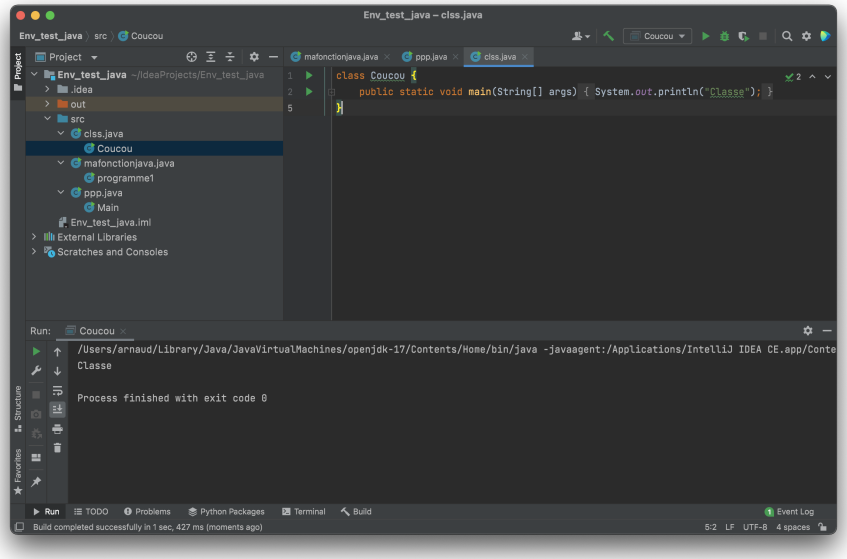
Dans le menu à gauche, vous allez retrouver un sous-dossier nommé src. Ouvrez l'emplacement de ce dossier puis copiez le fichier .java que vous souhaitez lancer.



La fenêtre qui vient de s'ouvrir est similaire à un éditeur de texte basique. C'est ici que le code peut être entré et modifié.
Vous pouvez maintenant lancer le fichier en allant dans **Run... > Run lenomdevotreclasse**

2.3 Les Classes en Java

Contrairement à d'autres langages de programmation, tout code en Java doit être écrit à l'intérieur d'une classe. Une classe en Java est semblable à une sorte de schéma pour créer ce qu'on appelle des objets. Les objets sont donc des instances particulières d'une certaine classe. Par exemple, "l'objet" Basketball serait une instance de la "classe" Sport. Pour créer une classe en Java, il suffit d'écrire **class Lenomdemaclasse**

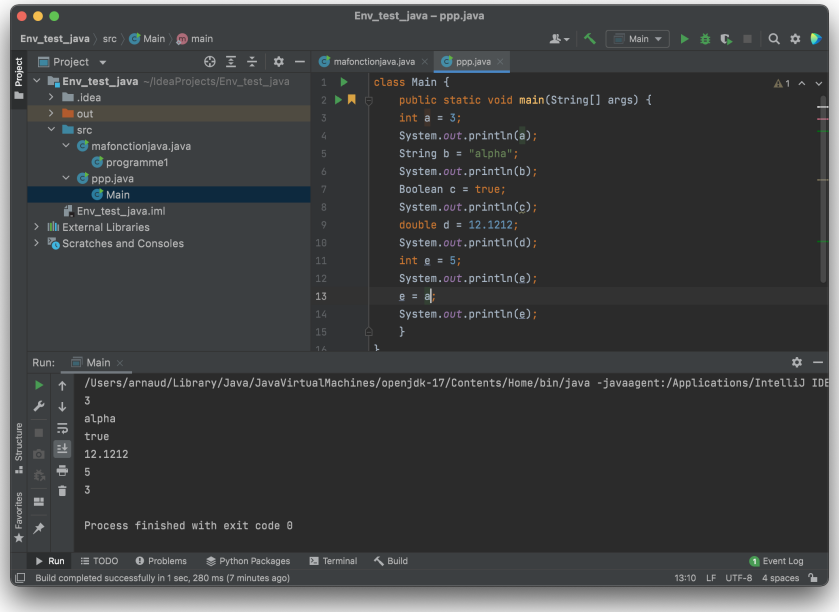


2.4 La méthode main

Dans tout programme Java, on doit toujours retrouver au moins une fois la fonction **public static void** main**(String[] args)** dans une de nos classes. Cette fonction est un point d'entrée dans le programme qui ne s'exécutera pas s'il ne retrouve pas cette fonction.

2.5 La création de variable

Les variables dans les langages de programmation sont similaires à des noms données à une valeur précise Pour assigner une valeur à une variable en Java, il suffit de respecter la forme suivante **type variable=valeur**.



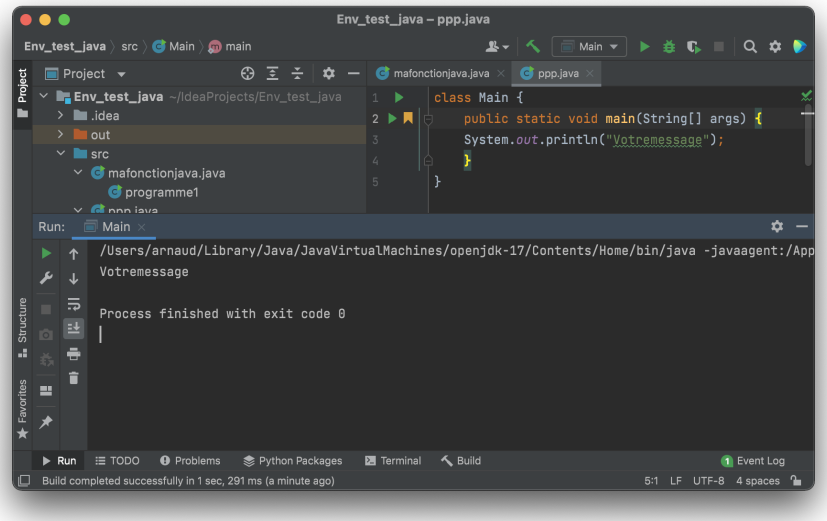
Conseil

- En Java, une variable est généralement dynamique (elle peut changer), mais il est possible de rendre une variable statique en ajoutant un type **final** avant le type de la variable.
- On peut assigner n'importe quelle suite de caractères non-réservée en tant que variable
- Il est aussi possible d'assigner des chaînes de caractères (strings en anglais) ou encore des valeurs booléennes à des variables
- En réassignant une nouvelle valeur à une variable déjà définie, la valeur de la variable va être écrasée et remplacée par la nouvelle valeur
- Il est possible d'additionner les variables du même type
- Plusieurs variables peuvent avoir la même valeur
- Le nom d'une variable doit toujours commencer par une lettre

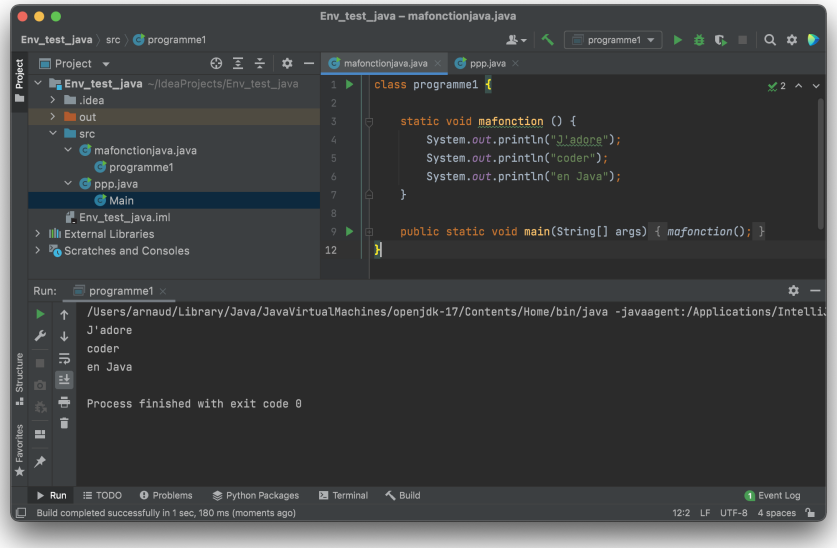
A noter que les dernières versions de Java supportent l'inférence du type des variables. On pourrait autant écrire **String a="Alpha"** que **a="Alpha"** car le type de la variable serait automatiquement attribué. Il est cependant important de quand même toujours prendre la peine de déclarer vos types de variables dans le cadre de ce cours.
Si vous ouvrez et exécutez le programme Java **Variables.java** qui contient quelques exemples d'attribution de variables, vous pourrez observer comment vous pouvez créer différentes variables mais aussi comment le programme les traite.

2.6 Les fonctions

Dans les langages de programmation, on retrouve un très grand nombre de fonctions. Ces fonctions sont des blocs de code qui, lorsqu'ils sont invoqués avec certains paramètres, effectuent certaines actions. Une des fonctions basique et plutôt importante en Java est la fonction **System.out.println("Votremessage");**. Cette fonction permet d'afficher à l'écran le contenu de la parenthèse.
Si vous entrez la ligne de code **System.out.println("Votremessage");** et que vous lancez le programme, IntelliJ va vous afficher une ligne de texte. Le message qui apparaît est celui que vous avez entré entre guillemets (à la place de "Votremessage"). C'est le but d'une fonction **System.out.println("Votremessage");**



En plus des fonctions incluses dans les librairies de base, il est possible de créer des fonctions complètement personnalisées (généralement plusieurs fonctions sont réunies en une seule) au moyen de la fonction **static void nomdelafonction()** et de l'utiliser à n'importe quel moment en l'invoquant avec **nomdelafonction()**.
Si vous ouvrez et exécutez le programme **programme1.java**, vous pourrez voir l'exemple de la fonction personnalisée **mafonction** qui imprime 3 suite de caractères différentes.


Env_test_java – mafonctionjava.java
Env_test_java
src
programme1
Project
Env_test_java ~/IdeaProjects/Env_test_java
.idea
out
src
mafonctionjava.java
programme1
ppp.java
Main
Env_test_java.iml
External Libraries
Scratches and Consoles
mafonctionjava.java
ppp.java
class programme1 {
static void mafonction () {
System.out.println("J'adore");
System.out.println("coder");
System.out.println("en Java");
}
public static void main(String[] args) { mafonction(); }
}
Run:
programme1
/Users/arnaud/Library/Java/JavaVirtualMachines/openjdk-17/Contents/Home/bin/java -javaagent:/Applications/Intelli
J'adore
coder
en Java
Process finished with exit code 0
Run
TODO
Problems
Python Packages
Terminal
Build
Event Log
Build completed successfully in 1 sec, 180 ms (moments ago)
12:2
LF
UTF-8
4 spaces