CM1 - Introduction

Category	Envoi 1

La programmation c'est quoi ?

Cela consiste à **planifier et organiser la résolution d'une tâche** par une machine, via à un **langage commun entre l'humain et la machine**, afin de répondre à un besoin. Cela passe par plusieurs étape :

- **Analyse** : étude des besoins
- **Algorithme** : conception des méthodes de résolution
- **Code** : implémentation des instructions dans le langage
- **Test** : vérification du bon comportement du programme
- **Portabilité** : déploiement du programme sur différents environnements
- **Maintenance** : (Sur)vie du programme sur le long terme

Les langages de programmation

Comme les langues étrangères ont une orthographe et une grammaire propre, les langages de programmation ont **des règles et des syntaxes (parfois communes, souvent semblables)** qui permettent aux humains de communiquer leurs instructions à la machine qui les exécute. On distingue cependant deux grandes catégories de langages :

- **Bas niveau** (proches du matériel) :
  - **Assembleur** : très proche du code machine (suite d'opérations binaires/numériques), difficile à lire pour les humains.
  - **C** : permet un contrôle fin sur le matériel et les ressources, utilisé pour les systèmes embarqués ou les OS.

- **Haut niveau** (proches de l'humain) :
    - **Python, Java** : Plus faciles à lire et à écrire, abstraient les détails matériels.

Principales différences entre langages

- **Performance/rapidité et typage des variables**
    - **Langages compilés avec typage statique** (ex : C, C++, Java)
        - les instructions sont transformées en code machine avant exécution → plus rapides/robustes
        - types de variable vérifiés à la compilation → prévention des erreurs
    - **Langages interprétés avec typage dynamique** (ex : Python, JavaScript)
        - les instructions sont exécutées ligne par ligne → plus lents mais plus flexibles
        - types de variables vérifiés à l'exécution → possibles surprises
- **Genre d'applications/support**
    - **Web** : JavaScript, HTML/CSS, PHP
    - **Data Science** : Python, R
    - **Interfaces graphiques** : Java
    - **Systèmes embarqués/Jeux vidéos** : C, C++, C#
    - **Applications mobiles** : Swift (iOS), Kotlin (Android)

La petite Histoire du Java

Java est issu d'un projet lancé en 1990 par **Sun Microsystems** sous la direction de James Gosling, Patrick Naughton et Mike Sheriden comme alternative au C++ et dont la philosophie et les innovations clés sont :

- **"Write Once, Run Anywhere"** (WORA) : grâce à la **machine virtuelle Java (JVM)**, les programmes sont compilés (en bytecode) et peuvent s'exécuter sur n'importe quel système (disposant d'une JVM), sans recompilation

- **Sécurité et robustesse** : par l'intégration d'une gestion automatique de la mémoire (ramasse-miettes) et de mécanismes avancés de sécurité et de réduction/détection des erreurs
- **Orientation entièrement objet** : expurgé des finesses/difficultés du C++ (pointeur et héritage multiple), la volonté d'être est donc modulaire, *théoriquement* facile à maintenir et réutilisable.

Un langage en perpétuelle évolution

De nombreuses mises à jour et améliorations depuis la version 1.0 en 1995

- 1.2 en 1998 : améliorations de performances (*compiler*) ; introduction des collections *List, Set, Map*
- 1.4 en 2002 : Regex
- 6 (1.6) en 2006 : désormais appelé Java SE (version encore stable) ; Intégration de langages de script (JavaScript, Python)
- 7 en 2011 : Try-with-resources ; Switch fonctionne avec String
- 8 en 2014 : Introduction des lambdas (programmation fonctionnelle)
- 9 en 2017 : apparition du JShell
- 10 à 16 (2018-2021) : Innovations incrémentales
- 25 en 2025, actuellement dernière LTS (après 17 en 2021 et 21 en 2023)

L'émergence d'**un écosystème via une communauté active** partageant une immense bibliothèque de classes/programmes réutilisables a permis à Java de se populariser dans le paysage de l'informatique, notamment pour les applications avec interface graphiques. Cette communauté veillent à la définition de standards, grâce au *Java Community Process* (https://jcp.org/en/participation/), organisation communautaire qui définit ou révise les spécification des technologies Java, même si le dernier mot reste à Oracle depuis le rachat en 2009.

Premier pas avec Java

Pour programmer en Java, comme dans beaucoup d'autres langages, il existe trois mode de développement, que nous vous invitons à tester, pour réaliser à quel

point le dernier est le plus confortable :

- **JShell** (mode calculette) : dans un terminal, tapez la commande *JShell* afin de faire apparaître le shell Java. Ce prompt vous permet de tester des instructions mais ne sauvegarde pas votre programme lorsque vous quitter (en tapant `/exit`). Il affiche également le résultat de l'opération même si pas spécialement demandé/sauvegardé dans une variable.
- **Editeur + Terminal** : créer, ouvrez et modifiez un fichier avec un éditeur de texte (e.g. *Gedit*, *Sublime*, ...). Les code que vous allez développer sera sauvegardé. Pour l'exécuter, vous devrez cependant ouvrir indépendamment (dans une autre fenêtre) un terminal et soit vous rendre dans le même répertoire que le fichier, soit indiquer le chemin vers celui-ci à la commande permettant de compiler votre fichier, puis à celle pour l'éxecuter.
- **Environnement de développement** (*integrated development environment* ; IDE) : les logiciels de type IDE intégrèrent de nombreux outils pour augmenter la productivité et le confort des programmeurs. Ils offrent la possibilité d'organiser différents panneaux où s'affichent par exemple un éditeur de texte, une console, un navigateur de fichier, l'état des variables en mémoire, etc. Il est surtout possible de les personnaliser via l'installation de nombreuses extensions. Dans la suite de ce cours, nous prenons l'exemple de *Virtual Studio Code* (et de l'extension Run Coder) offrant un bon compromis de fonctionnalités et intégrant Java et Python. Des IDEs plus spécifiques/adaptés à chaque langage existent et vous êtes vivement invités à les tester/comparer afin de travailler avec celui sur lequel vous vous sentirez le plus à l'aise (e.g. *IntelliJ*, ou *Eclipse* pour Java, *Spyder* ou *Pycharm* pour Python).

Pour découvrir les bases de la syntaxe du langage Java, nous allons commencer par voir comment :

- **compiler et exécuter** vos futurs programmes, via un tutoriel "*Hello World !*"
- **déclarer une variable** avec le typage statique, via une illustration de traduction de Python à Java

Notez toutefois que l'aspect POO conduit à certaines lourdeurs dans la syntaxe que nous ne justifierons pas dans les premiers exemples (qu'il faudra accepter telles quelles) avant de les expliquer en détail plus tard, lorsque l'on introduira la notion d'objet/classe.