Chapitre 1: Introduction à JAVA

Prérequis

Pour suivre ce cours sur `JAVA`, il n'y a aucun prérequis mais avoir des **bases en programmation** est un plus hormis cela la formation part du principe que vous n'avez aucune connaissance en `JAVA`

▼ Qu'est-ce que JAVA?

Histoire de Java

Le langage `Java` est un langage de programmation `multi-paradigmes`. c'est à dire qu'il est conçu pour supporter des éléments de `programmation procédurale` et de `programmation orientée objet`.

Programmation orientée objet vs programmation procédurale

La programmation orientée objet (POO) et la programmation procédurale sont deux paradigmes de programmation. Un paradigme de programmation est un style fondamental de programmation informatique, et ils diffèrent dans la façon dont les différents éléments du programme sont représentés et comment les étapes de résolution des problèmes sont définies. Comme son nom l'indique, la POO se concentre sur la représentation des problèmes à l'aide d'objets du monde réel et de leur comportement, tandis que la programmation procédurale traite de la représentation de solutions aux problèmes à l'aide de procédures, qui sont des collections de code qui s'exécutent dans un ordre spécifique. Il existe des langages de programmation qui prennent en charge les aspects clés de la POO (appelés langages OOP), de la procédure (appelés langages procéduraux) et les deux. On peut citer un le langage C++ comme exemple.

Il a été **créé** en **1994** par **James Gosling**, employé de **Sun Microsystems**. La première version officielle a été présentée le **23 mai 1995**.

Par la suite, la société **Sun** a été rachetée par **Oracle** en **2009** qui détient désormais **Java**. A noter qu'il existe une version **open source** de Java : `OpenJdk` (que nous utiliserons au cours de cette formation).

Malgré son ancienneté, le langage est toujours très utilisé actuellement et la `version Java SE 21` est sortie le **19 septembre 2023**. (`SE : Standard Edition`).

Il est réputé comme étant un langage de **programmation portable** : tout **code compilé** en Java sera **exécutable** sur **n'importe quel système d'exploitation** en utilisant la machine virtuelle Java (JVM).

Il a été nommé Java en l'honneur de la **boisson préférée des programmeurs**, c'est-à-dire le **café**, dont une partie de la production provient de **l'île Java en Indonésie**. On comprend alors pourquoi le **logo** ressemble à une **tasse de café** :

Caractéristiques principales de Java

`Java` est un langage **compilé**. Contrairement au `C`, il n'est pas **traduit** directement en **langage machine**, mais en un **code intermédiaire** formé de **bytecodes**. La particularité de ce code intermédiaire est qu'il est exécutable dans la `JVM`, ce qui permet de faire abstraction du système d'exploitation sous-jacent et de rendre le code portable.

`Java` est un langage avec un **typage statique fort**. Cela signifie que le typage des variables est **obligatoire** pendant l'écriture du code. C'est vérifié lors de la compilation en amont de l'exécution du programme (à l'inverse du typage **dynamique** d'un langage **interprété**).Il est **fortement typé** car il est **impossible** de définir une

valeur qui ne correspond pas au type choisit car cela provoque une **erreur de compilation**.

`Java` est un langage avec une **gestion automatique de la mémoire par ramasse-miettes** (`garbage collector`). Cela signifie que la **mémoire** sera **automatiquement allouée** et **libérée** à l'exécution des programmes dans la JVM.

`Java` est historiquement vu comme un **langage orienté objet (OOP)**. Aujourd'hui, il est plutôt **multi-paradigmes** suite aux très nombreuses mises à jour du langage. Il peut donc être utilisé en programmation **fonctionnelle**, **impérative ou procédural**, **générique**, etc.

`Java` est un **langage de haut niveau** avec une syntaxe simple. Cela signifie que le langage cherche à être proche du **langage naturel** pour simplifier son utilisation, contrairement à un **langage de bas niveau** qui cherche à être au plus près des **caractéristiques techniques de l'ordinateur** (**registres**, **processeurs**, **mémoire etc**).

Enfin, `Java` est un langage **multiplateformes**. En plus d'être **portable**, cela signifie que le langage est exécutable sur **différents types de machines** : téléphones portables, voitures, systèmes embarqués, etc

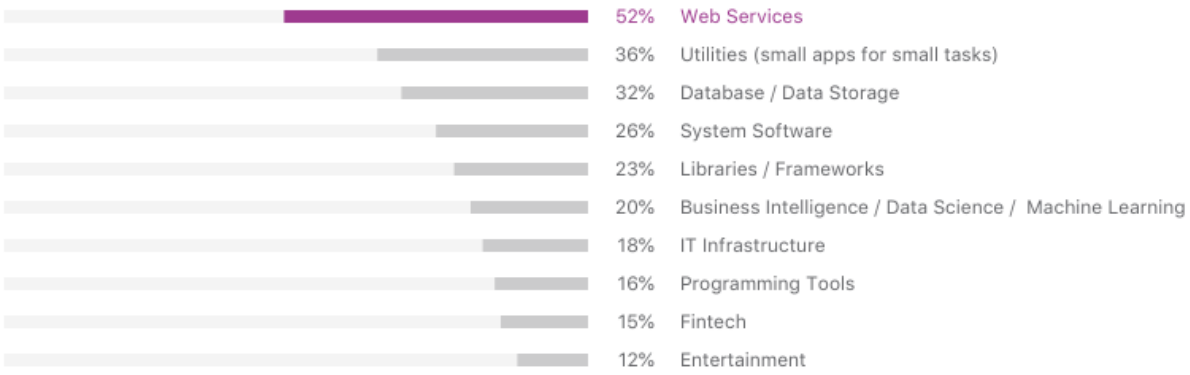
- `Java est utilisé pour créer` :
  ▪ Des applications Desktop
  ▪ Des applications Web JEE (Java Entreprise Edition)
  ▪ Des applets java (applications java destinées à s'exécuter dans une page web)
  ▪ Des applications pour les smart phones
  ▪ Des applications embarquées dans des cartes à puces

- `Pour créer une application java, il faut installer un kit de développement java`
  ▪ JSE : Java Standard Edition pour développer les applications Desktop. On parle de
  clients lourds.
  ▪ JEE : Java Entreprise Edition, pour développer les applications web qui vont s'exécuter dans un serveur d'application JEE (Web Sphere Web Logic, JBoss).On parle de clients légers.
  ▪ JME : Java Micro Edition, pour développer les applications pour les téléphones potables
  ▪ JCA : Java Card Edition, pour développer les applications qui vont s'éxécuter dans
  des cartes à puces.

Utilisations de Java

Java est utilisé par la plupart des **grandes entreprises** pour sa **robustesse** et sa **portabilité** : Google, Microsoft, Airbnb, Ebay, Netflix, Spotify, etc.

Java est surtout prisé aujourd'hui par les grandes entreprises pour développer des `applications côté serveurs (ou "backend")`, comme le montre le graphique suivant :



Celui-ci montre comment Java est utilisé selon les résultats de l'état de l'écosystème des développeurs 2020 produit par JetBrains.

Programmation de services Web

Java est très performant pour développer des **services Web ou API**.

Il est en effet possible de créer des services aussi bien performants que sécurisés grâce à des **frameworks** comme **Spring Boot**.

Programmation d'interfaces graphiques

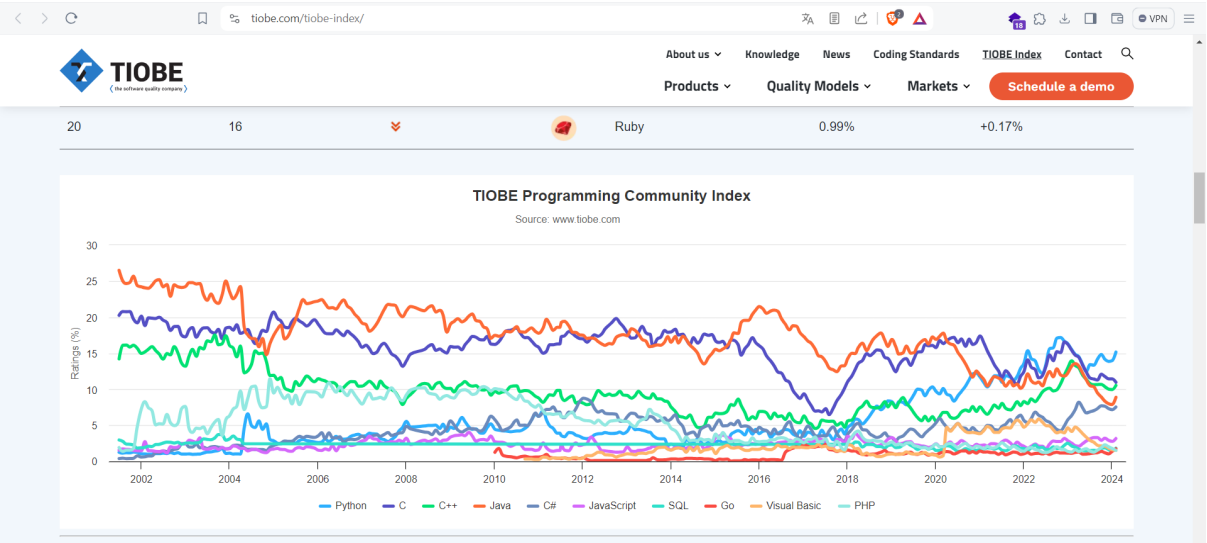
Même si Java est très prisé pour faire du code serveur, il est également possible de développer des clients lourds avec `JavaFx`

`JavaFX` est un framework et une bibliothèque d'interface utilisateur issue du projet OpenJFX, qui permet aux développeurs Java de créer une interface graphique pour des applications de bureau, des applications internet riches et des applications smartphones et tablettes tactiles.I fournit une bibliothèque d'interface utilisateur (UI) moderne ainsi que des fonctionnalités pour la création d'applications multimédias, telles que des graphiques 2D et 3D, des animations, des vidéos et des effets visuels.

Si vous souhaitez développer des clients légers pour navigateur, il est préférable d'utiliser des **frameworks Typescript / Javascript** à savoir **Angular**, **React & Vue**.

Communauté du langage Java

Java fait partie des langages les plus stables en popularité depuis le début des années 2000 :



Cette courbe décrit l'évolution de la popularité des principaux langages de programmation selon l'index TIOBE. On peut voir que Java a régulièrement été le langage le plus populaire par le passé et il se classe quatrième en Février 2024. Cela prouve une longévité remarquable de Java qui est une technologie éprouvée et compétitive.

Cette popularité se traduit par un engouement mondial du langage selon les résultats de l'état de l'écosystème des développeurs 2020 produit par JetBrains :

On voit bien que la communauté Java représente des millions de développeurs. Vous n'aurez aucun mal à trouver des développeurs expérimentés pour vous aider si vous êtes bloqués dans le développement de vos programmes Java.

▼ Comment fonctionne Java ?

Fonctionnement d'un compilateur

Un `compilateur` est responsable de **convertir** du **code source** qui respecte une certaine syntaxe en **code machine** exécutable par l'ordinateur.

Voici les principaux avantages qu'offre un langage **compilé** :

- Ils sont plus performants que les langages dits "interprétés" qui font toutes les étapes d'interprétation et d'exécution du code à la volée.
- Le code source écrit ne compile pas s'il contient des erreurs. Il n'y par exemple pas besoin d'exécuter son programme pour détecter des erreurs de typage, il suffit de compiler.

D'un autre côté, un langage compilé possède généralement le désavantage suivant :

- Un **compilateur** fabrique du langage machine **compatible** avec le **type de machine utilisé**. Le programme obtenu ne sera donc pas exécutable sur un autre type de machine (**OS ou architecture différente**).

Java est-il un langage compilé ?

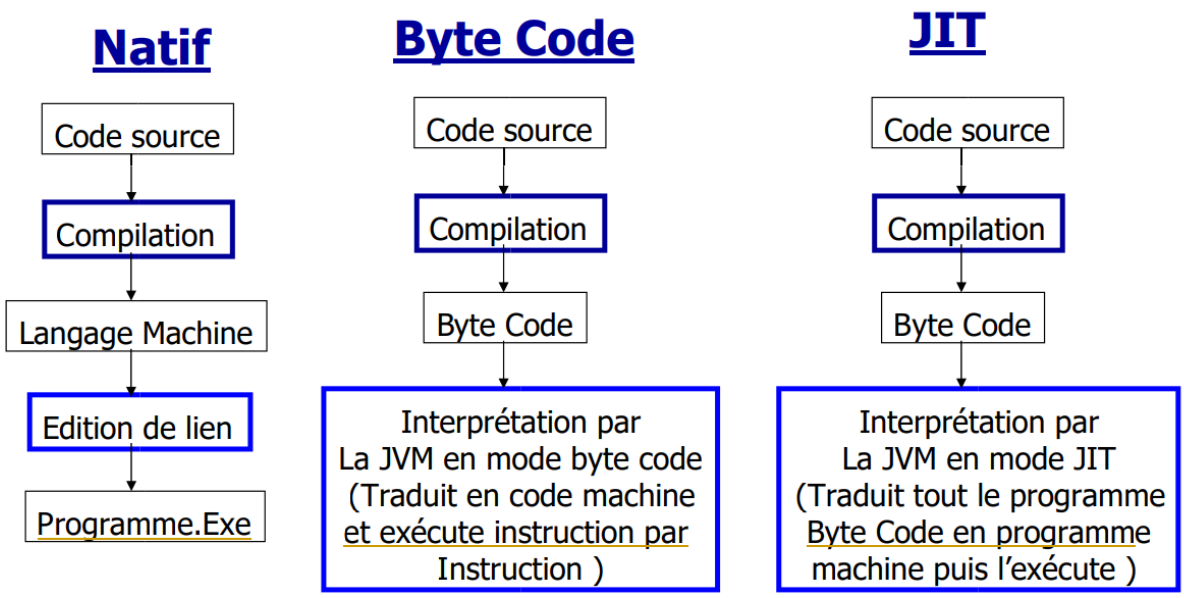
Oui et non, il y a bien une étape de compilation, mais ce n'est pas un langage **uniquement compilé**. On dit que Java est plutôt un **langage "semi interprété"**.

Pour le comprendre, voici les **étapes** de compilation et d'exécution de Java :

1. Écriture du `code source Java`
2. Compilation du code source : Le résultat n'est ici pas du **code machine** comme pour un langage compilé classique. Le résultat obtenu est du `bytecode intermédiaire`
3. Exécution du `bytecode` dans la `JVM (Java Virtual Machine)`

Le `bytecode` est un l**angage intermédiaire entre le code source et le code machine** qui permet l'exécution d'applications Java multiplate-forme puisque il est indépendant de tout système d'exploitation.

C'est alors que la `JVM` entre en compte pour rendre cette exécution possible. Elle est disponible sur la plupart des OS pour interpréter le bytecode. Elle est aussi disponible sur nombre de systèmes embarqués.



Les outils du JDK (Java Development Kit)

Le `JDK (Java Développement Kit)` désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code

Java peut être **compilé**, **transformé** en byte code **destiné** à la machine virtuelle Java. Le JDK contient **JRE (Java Runtime Environnement).**

Le **JRE** permet l'exécution des programmes écrits en langage Java sur de **nombreux types d'appareils** (ordinateurs personnels, téléphones mobiles,...) en faisant **abstraction des caractéristiques techniques de la plateforme informatique sous-jacente**. Il se **compose** d'une **machine virtuelle (JVM)**, de bibliothèques logicielles utilisées par les programmes Java et d'un plugin pour permettre l'exécution de ces programmes depuis les

navigateurs web.

En résumé le **JDK** fournit un ensemble d'outils pour compiler et exécuter du code Java.

Le compilateur `Javac`

Cet **outil** est le **compilateur** : il utilise des **fichiers source Java** pour créer des **fichiers contenant le bytecode Java** correspondant. Pour chaque fichier source avec l'extension `.java`, un fichier portant le même nom avec l'extension `.class` est créé si la **compilation se déroule bien**. En cas d'erreur d'écriture du code source, il y aura des erreurs relevées par le compilateur et **aucun bytecode ne sera alors généré**.

> Exemple de compilation en ligne de commande :

```
javac MaPremiereClasse.java
```

L'exécution de cette **commande résulte en la création** d'un fichier `MaPremiereClasse.class` qui contient le **bytecode** du code compilé.

L'interpréteur `java / javaw`

Cet **outil** est **l'interpréteur de bytecode** : il lance la **JVM**, charge les classes nécessaires et exécute la **méthode main de la classe** passée en paramètre. La méthode **main** et le **point d'entrée** des programmes Java et doit **exister dans le code source**.

> **Remarque:**
> Le nom du fichier java doit être le même que celui de la classe qui

> contient la fonction
> principale main.

> Exemple d'exécution de bytecode en ligne de commande :

```
java MaPremiereClasse/ javaw MaPremiereClasse
```

La commande `java` **ouvre** une console et exécute le **bytecode MaPremiereClasse.class**.

La commande `javaw` exécute le **bytecode MaPremiereClasse.class** mais **n'ouvre pas** de console.

L'outil `jar`

Jar est le diminutif de `Java ARchive`. C'est un format de fichier qui permet de regrouper des fichiers contenant du bytecode Java (`fichier .class`) ou des données utilisées en tant que ressources (**images**, **son**, **etc**).

Ce format est compatible avec le format **ZIP** : les fichiers contenus dans un **jar** sont **compressés** de façon **indépendante du système d'exploitation**. On peut par exemple créer un seul **fichier .jar** exécutable qui sera le fichier final de l'application que l'on souhaite créer.

> Exemple de commande jar :

```
jar cf monJar.jar *.class
```

Cette commande résulte en la création d'un jar ayant comme nom `monJar.jar` avec tous les **fichiers .class**

Installation de l'environnement

▼ Environnement Windows

▼ Installation de Git

Lors de l'installation de `Git`, deux programmes très utiles sont installés :

Le programme `Git` lui-même. Il s'agit d'un outil de gestion de **versions décentralisé**. Il permet notamment de partager du code et sera très utile pour réaliser les exercices et projets tout au long de ce cours.

Le programme
`Git Bash`. Il s'agit d'un **Shell** qui sera capable d'interpréter les mêmes instructions de base que sur un environnement **Linux** ou **MacOs**.

Or, sur **Windows**, le terminal par défaut est le **Powershell**. Il possède une syntaxe spécifique aux environnements Windows et c'est très contraignant pour les développeurs en général.

Grâce au **Git Bash,** tous les étudiants qui suivent cette formation pourront donc exécuter les mêmes commandes dans leur console, peu importe qu'ils soient sur **Windows**, **Linux** ou **MacOs**.

Nous allons donc installer git pour pouvoir l'utiliser : téléchargez le ici.

> Git - Downloads
> The entire Pro Git book written
> by Scott Chacon and Ben Straub is available to read online for free.
> Dead tree versions are available on Amazon.com.
> https://git-scm.com/downloads

▼ Installation d'OpenJdk

Vous avez 2 possibilités pour installer JAVA sur votre machine:

Aujourd'hui, la distribution du JDK d'Oracle n'est plus le choix privilégié quand on souhaite développer en Java.

La raison vient d'OpenJdk : Il s'agit de la distribution officielle et libre de Java SE, tel que défini par le Java Community Process. Pour résumer, cette communauté est responsable de coordonner les évolutions du langage Java

- La première manière c'est de l'installer via son **exécutable disponible sur le site openJDK**, mais cette méthode vous impose à avoir une version spécifique de PHP installer si plus tard vous avez besoin d'une autre vous serez obligé de refaire la même procédure et ceci peut entrainer des conflits de versions suivant vos différents projets en PHP.

  Pour installer l'éditeur PHP, téléchargez l'exécutable ici.

> OpenJDK JDK 21.0.2 GA Release
>
> This page provides production-ready open-source builds of the Java Development Kit,
> version 21, an implementation of the Java SE 21
>
> https://jdk.java.net/21/

Cliquez sur le lien de téléchargement du Zip de la **version Windows.**

Dézippez le dossier téléchargé par exemple dans `C > Programmes > java`.

- La deuxième méthode que je vous recommande fortement consiste à installer `SDKMAN`.

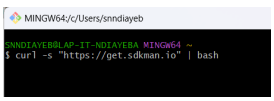
  SDKMAN est un outil permettant de gérer des versions parallèles de plusieurs SDK, que **SDKMAN** appelle "**candidats**".
  Il fournit une interface de
  ligne de commande (CLI) et une **API** pratiques pour **lister**, **installer**, **changer** et **supprimer** des **candidats**. De plus, il se charge de définir les variables d'environnement pour nous.
  Il permet également aux développeurs d'installer des
  SDK basés sur la **JVM** comme **Java**, **Groovy**, **Scala**, **Kotlin** et **Ceylon**. **Maven**, **Gradle**, **SBT**, **Spring Boot**, **Vert.x** et bien d'autres sont également pris en charge. **SDKMAN** est un utilitaire **gratuit**, léger et **open-source** écrit en **Bash**.

> NB: Pour pouvoir l'installer sur windows il faut obligatoirement un terminal Zsh ou bash comme Git Bash qui vient avec l'installation de Git

Nous allons donc installer `SDKMAN` pour pouvoir l'utiliser : téléchargez le ici.

> Home - SDKMAN! the Software Development Kit Manager
>
> SDKMAN! is a tool for managing parallel versions of multiple Software Development Kits on most Unix based systems.
>
> https://sdkman.io/

Ouvrez le terminal **Git Bash** et lancer cette commande



```
curl -s "https://get.sdkman.io" | bash
```

Ensuite, suivez les instructions à l'écran pour terminer l'installation.

Il se peut que les paquets **zip** et **unzip** soient nécessaires pour compléter le processus d'installation.

Vous pouvez télécharger leurs exécutables et les installer via le lien ci dessous:

zip and unzip for the windows command line

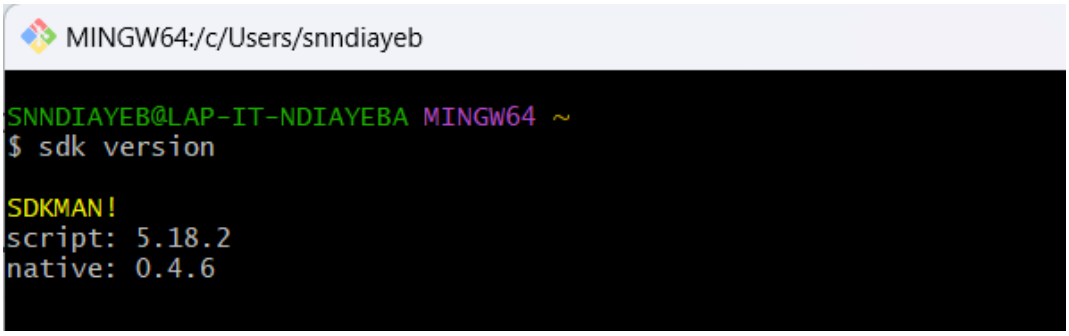
List biggest files
List newest files
Show subdir sizes
http://stahlworks.com/dev/index.php?tool=zipunzip

Ensuite, ouvrez un nouveau terminal ou exécutez la commande suivante dans le même shell :

```
$ source "$HOME/.sdkman/bin/sdkman-init.sh"
```

Enfin, exécutez l'extrait suivant pour confirmer le succès de l'installation :

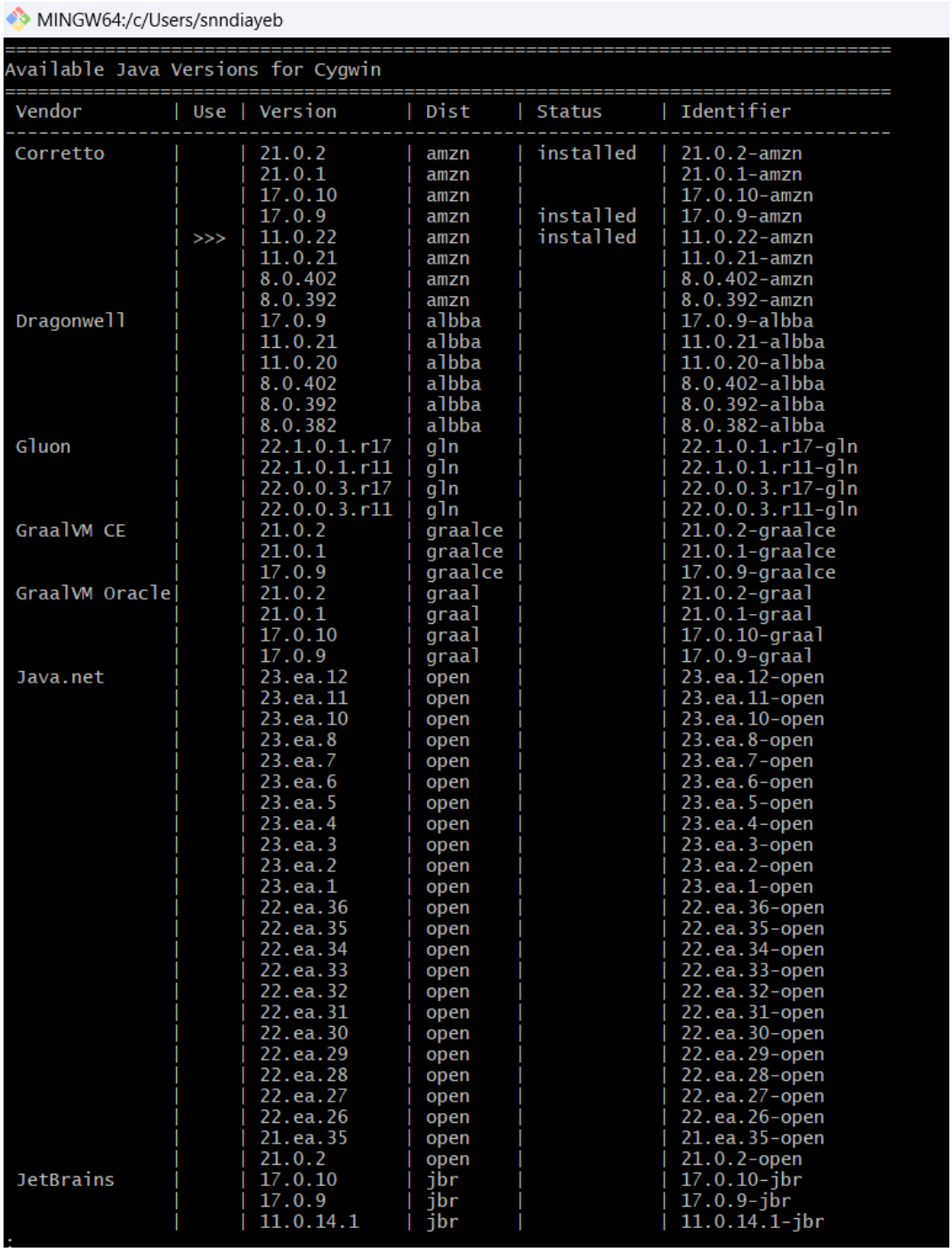
```
$ sdk version
```



Pour dresser la liste des versions disponibles de Java, utilisez la commande:

```
$ sdk list java
```

Le résultat est un tableau d'entrées regroupées par fournisseur et triées par version :



Maintenant une version de java spécifique vous prenez une valeur au niveau de la colonne **Identifier** et vous lancez la commande :

```
$ sdk install java 11.0.22-amzn
```

Passer d'une version à l'autre

Avec SDKMAN on peut installer plusieurs version différentes et switcher entre elle selon nos besoins.

Nous pouvons contrôler le passage d'une version à l'autre sous deux formes :

1. Temporairement

   ```
   $ sdk use java 21.0.2-amzn
   ```

2. Permanente

   ```
   $ sdk default java 11.0.22-amzn
   ```

Supprimer une version

Pour supprimer une version installée, exécutez la commande de désinstallation avec la version ciblée :

```
$ sdk uninstall java 14.0.1.j9-adpt
```

Afficher la version utilisée

Pour vérifier la version actuelle de Java, nous exécutons la commande suivante :

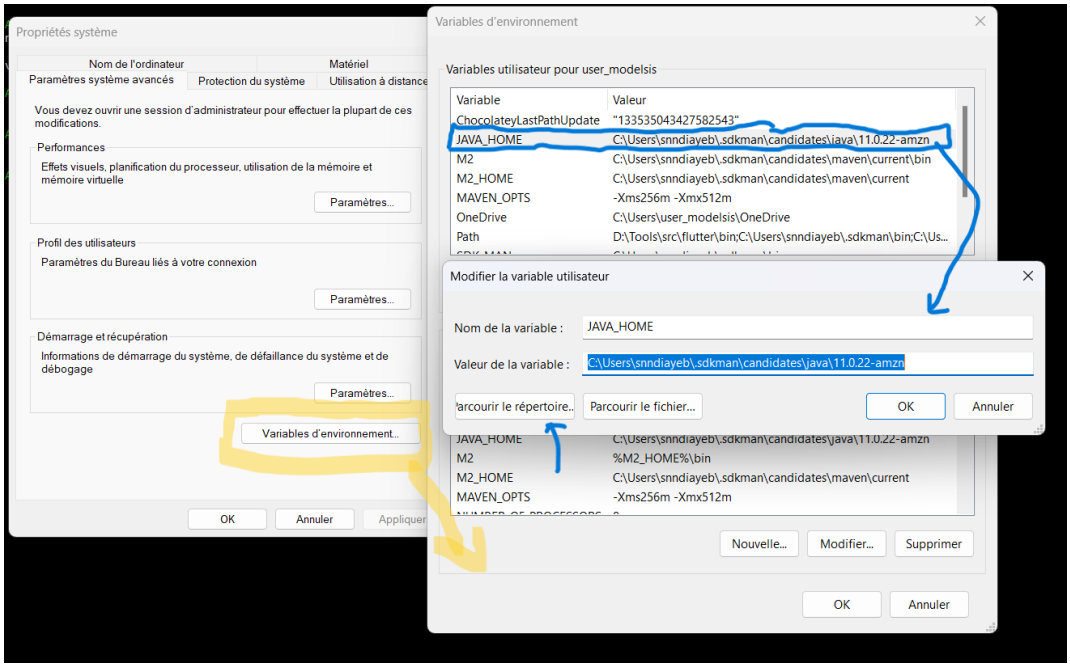
```
$ sdk current java

Using java version 11.0.22-amzn
```

▼ Configuration de la variable PATH

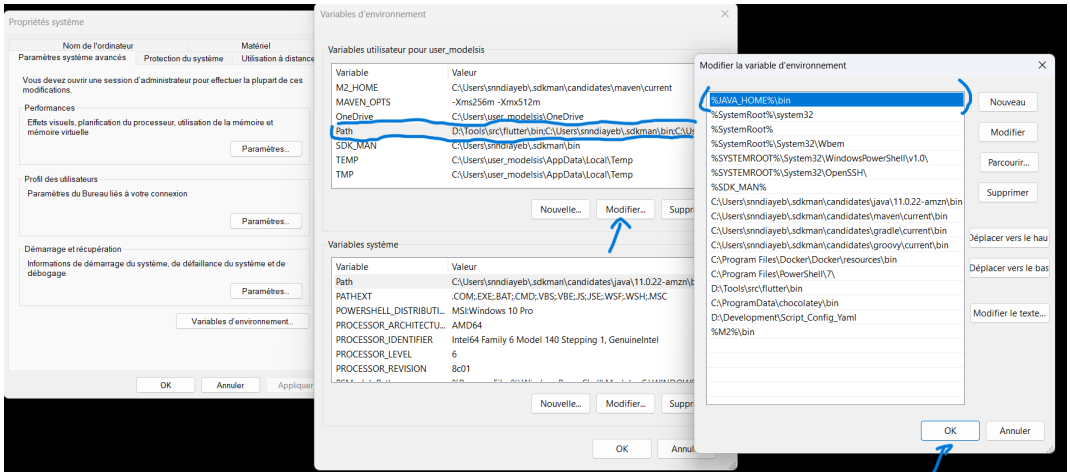
Lors de l'installation d'OpenJdk, il est important de rendre les programmes exécutables sur l'ensemble de la machine, peu importe le dossier dans lequel on se trouve.

Pour cela, il faut modifier les **variables d'environnement** du système de la manière suivante :

- Créer une variable `JAVA_HOME` : Sa valeur doit pointer sur le dossier qui contient le JDK installé sur la machine.



- Modifier la variable **PATH** existante : Ajouter `%JAVA_HOME%\bin`
  Il faut s'assurer que chaque valeur dans la variable PATH est bien séparée par un ;
  Le dossier bin contient les programmes javac et java, que nous avons vu dans la partie précédente.



L'installation de **JAVA** se valide avec la commande suivante dans le terminal :

```
java -version
```

Si la commande affiche la version comme attendu, c'est que l'installation s'est bien passée. Il est désormais possible d'utiliser **javac** et **java** dans le terminal.

Installation d'IntelliJ CE (Community Edition)

IntelliJ CE est sans conteste l'un des meilleurs éditeurs de code pour le monde Java. Cet EDI (Environnement de Développement Intégré) est produit par l'éditeur de logiciel JetBrains. Il a l'avantage d'être gratuit et possède de nombreux plugins pour améliorer son utilisation.

Pour installer IntelliJ CE allez sur le site officiel de JetBrains via le lien en dessous puis vous scrollez en bas pour télécharger la Community Edition

Download IntelliJ IDEA – The Leading Java and Kotlin IDE

Download the latest version of IntelliJ IDEA for Windows, macOS or Linux.

https://www.jetbrains.com/idea/download/?section=windows

Configurer Git Bash dans IntelliJ

Ouvrir les paramètres projet : `Ctrl + Alt + S` ou le **menu File / Project Settings**.

Naviguer ensuite dans **Tools / Terminal.**

Ouvrir la liste déroulante dans **Shell Path** et sélectionner le **Git Bash** à la place de **Powershell**.

Configurer le Jdk Java dans IntelliJ

Ouvrir les paramètres de structure du projet : `Ctrl + Alt + Shift + S` ou le **menu File / Project structure settings**

Dans **SDK**, sélectionner le **JDK** dans la liste déroulante.

Si rien n'est présent, aller dans le menu **SDKs** et ajouter le **jdk** installé sur la machine. C'est peut-être nécessaire si **IntelliJ** n'a pas été capable de détecter automatiquement le JDK.

▼ Environnement MacOs

1. ## Installation de Homebrew

   `Homebrew` est un gestionnaire de paquets développé en Ruby pour faciliter l'installation de logiciels sur MacOs.

   La commande suivante permet de l'installer :

   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
   ```

2. ## Installation SDKMAN

   SDKMAN est un outil permettant de gérer des versions parallèles de plusieurs SDK, que **SDKMAN** appelle "**candidats**".
   Il fournit une interface de
   ligne de commande (CLI) et une **API** pratiques pour **lister**, **installer**, **changer** et **supprimer** des **candidats**. De plus, il se charge de définir les variables d'environnement pour nous.
   Il permet également aux développeurs d'installer des
   SDK basés sur la **JVM** comme **Java**, **Groovy**, **Scala**, **Kotlin** et **Ceylon**. **Maven**, **Gradle**, **SBT**, **Spring Boot**, **Vert.x** et bien d'autres sont également pris en charge. **SDKMAN** est un utilitaire **gratuit**, léger et **open-source** écrit en **Bash**.

   Nous allons donc installer `SDKMAN` pour pouvoir l'utiliser : téléchargez le ici.

   Home - SDKMAN! the Software Development Kit Manager
   SDKMAN! is a tool for managing parallel versions of multiple Software Development Kits on most Unix based systems.
   https://sdkman.io/

   Ouvrez le terminalet lancer cette commande

   ```
   curl -s "https://get.sdkman.io" | bash
   ```

   Ensuite, suivez les instructions à l'écran pour terminer l'installation.

   Il se peut que les paquets **zip** et **unzip** soient nécessaires pour compléter le processus d'installation.

   Ensuite, ouvrez un nouveau terminal ou exécutez la commande suivante dans le même shell :

```
$ source "$HOME/.sdkman/bin/sdkman-init.sh"
```

Enfin, exécutez l'extrait suivant pour confirmer le succès de l'installation :

```
$ sdk version
```

Pour dresser la liste des versions disponibles de Java, utilisez la commande:

```
$ sdk list java
```

Le résultat est un tableau d'entrées regroupées par fournisseur et triées par version :

```
================================================================================
Available Java Versions for Cygwin
================================================================================
 Vendor        | Use | Version      | Dist    | Status     |
--------------------------------------------------------------------------------
 Corretto      |     | 21.0.2       | amzn    | installed  |
               |     | 21.0.1       | amzn    |            |
               |     | 17.0.10      | amzn    |            |
               |     | 17.0.9       | amzn    | installed  |
               | >>> | 11.0.22      | amzn    | installed  |
               |     | 11.0.21      | amzn    |            |
               |     | 8.0.402      | amzn    |            |
               |     | 8.0.392      | amzn    |            |
 Dragonwell    |     | 17.0.9       | albba   |            |
               |     | 11.0.21      | albba   |            |
               |     | 11.0.20      | albba   |            |
               |     | 8.0.402      | albba   |            |
               |     | 8.0.392      | albba   |            |
               |     | 8.0.382      | albba   |            |
 Gluon         |     | 22.1.0.1.r17 | gln     |            |
               |     | 22.1.0.1.r11 | gln     |            |
               |     | 22.0.0.3.r17 | gln     |            |
               |     | 22.0.0.3.r11 | gln     |            |
 GraalVM CE    |     | 21.0.2       | graalce |            |
               |     | 21.0.1       | graalce |            |
```

```
               |     | 17.0.9       | graalce |             |
 GraalVM Oracle|     | 21.0.2       | graal   |             |
               |     | 21.0.1       | graal   |             |
               |     | 17.0.10      | graal   |             |
               |     | 17.0.9       | graal   |             |
 Java.net      |     | 23.ea.12     | open    |             |
               |     | 23.ea.11     | open    |             |
               |     | 23.ea.10     | open    |             |
               |     | 23.ea.8      | open    |             |
               |     | 23.ea.7      | open    |             |
               |     | 23.ea.6      | open    |             |
               |     | 23.ea.5      | open    |             |
               |     | 23.ea.4      | open    |             |
               |     | 23.ea.3      | open    |             |
               |     | 23.ea.2      | open    |             |
               |     | 23.ea.1      | open    |             |
               |     | 22.ea.36     | open    |             |
               |     | 22.ea.35     | open    |             |
               |     | 22.ea.34     | open    |             |
               |     | 22.ea.33     | open    |             |
               |     | 22.ea.32     | open    |             |
               |     | 22.ea.31     | open    |             |
               |     | 22.ea.30     | open    |             |
               |     | 22.ea.29     | open    |             |
               |     | 22.ea.28     | open    |             |
               |     | 22.ea.27     | open    |             |
               |     | 22.ea.26     | open    |             |
               |     | 21.ea.35     | open    |             |
               |     | 21.0.2       | open    |             |
 JetBrains     |     | 17.0.10      | jbr     |             |
               |     | 17.0.9       | jbr     |             |
               |     | 11.0.14.1    | jbr     |             |
:
```

Maintenant une version de java spécifique vous prenez une valeur au niveau de la colonne **Identifier** et vous lancez la commande :

```
$ sdk install java 11.0.22-amzn
```

Passer d'une version à l'autre

Avec SDKMAN on peut installer plusieurs version différentes et switcher entre elle selon nos besoins.

Nous pouvons contrôler le passage d'une version à l'autre sous deux formes :

1. Temporairement

```
$ sdk use java 21.0.2-amzn
```

2. Permanente

```
$ sdk default java 11.0.22-amzn
```

Afficher la version utilisée

Pour vérifier la version actuelle de Java, nous exécutons la commande suivante :

```
$ sdk current java

Using java version 11.0.22-amzn
```

Supprimer une version

Pour supprimer une version installée, exécutez la commande de désinstallation avec la version ciblée :

```
$ sdk uninstall java 14.0.1.j9-adpt
```

Installation d'IntelliJ CE (Community Edition)

IntelliJ CE est sans conteste l'un des meilleurs éditeurs de code pour le monde Java. Cet EDI (Environnement de Développement Intégré) est produit par l'éditeur de logiciel JetBrains. Il a l'avantage d'être gratuit et possède de nombreux plugins pour améliorer son utilisation.

Pour installer IntelliJ avec HomeBrew, lancer la commande suivante :

```
brew install --cask intellij-idea-ce
```

Lancement de l'éditeur :

```
intellij-idea-community
```

Configurer le Jdk Java dans IntelliJ

Ouvrir les paramètres de structure du projet : `Cmd +` ; ou le **menu File / Project structure settings**

Dans **SDK**, sélectionner le **JDK** dans la liste déroulante.

Si rien n'est présent, aller dans le menu **SDKs** et ajouter le **jdk** installé sur la machine. C'est peut-être nécessaire si **IntelliJ** n'a pas été capable de détecter automatiquement le JDK.