# Mise en place d'un environnement de programmation avec Visual Studio Code

Université Paris Cité

Objectifs:

- Faire la différence entre éditeur de texte et IDE.
- Installer VSCode et les extensions utiles pour programmer en Java.
- Connaître quelques fonctionnalités basiques de VSCode.

Pour programmer, on a juste besoin d'un éditeur de texte, tel que kwrite, gedit, vim, Emacs, Notepad++, Sublime Text, etc. Ensuite, on peut exécuter ses programmes dans le terminal avec les commandes `javac` et `java`. C'est suffisant au début, mais on se retrouve vite à jongler entre plusieurs fenêtres, surtout quand notre programme se retrouve découpé en plusieurs fichiers.

Pour se faciliter la vie, il existe des logiciels appelés IDE (Integrated Development Environment) qui permettent de regrouper au même endroit tous les outils qui permettent aux programmeurs d'écrire, tester, et débugger leur code :

- Un éditeur de texte avec des fonctionnalités spécifiques à chaque langage de programmation (coloration syntaxique, auto-complétion, indentation automatique, ...).
- Un terminal intégrer pour exécuter des commandes.
- Un gestionnaire de projet pour organiser les différents fichiers.
- Et plein d'autres fonctionnalités utiles.

Tous ces outils combinés rendent la programmation plus fluide et efficace. L'objectif de ce document est de présenter l'IDE VSCode, et quelques fonctionnalités de base utiles pour programmer en Java.

# 1 Installation

## 1.1 Installation du Java Development Kit (JDK)

Le kit de développement Java (Java Development Kit, en anglais), ou JDK, permet de compiler et d'exécuter le code Java sur votre ordinateur.

Comment l'installer ?

### 1.1.1 Linux (Ubuntu/Debian).

Sur les versions récentes d'Ubuntu, le JDK est déjà installé par défaut. Pour vérifier si c'est le cas, tapez dans un terminal la commande `java -version`. Si Java est installé, cette commande affichera le numéro de version de Java qui est installée. Il n'y a rien de plus à faire. Sinon :

1. Ouvrez une fenêtre de terminal.
2. Mettez à jour la liste des paquets en utilisant la commande :
   `sudo apt update`

3. Installez le JDK par défaut à l'aide de la commande :
   `sudo apt install default-jdk`
4. Vérifiez le succès de l'opération en utilisant la commande :
   `java -version`

De nombreux programmes écrits avec JAVA utilisent la variable d'environnement `JAVA_HOME` pour déterminer l'emplacement d'installation de JAVA.

1. Ouvrez une fenêtre de terminal.
2. Déterminez l'endroit où JAVA est installé à l'aide de cette commande :
   `sudo update-alternatives --config java`
   Cette commande affiche chaque installation de JAVA ainsi que son chemin d'installation.
   **Par exemple**, il s'agit de quelque chose comme `/usr/lib/jvm/java-21-openjdk-amd64/bin/java`.
3. Définissez `JAVA_HOME` à l'aide de la commande suivante : **Par exemple**,
   `echo 'export JAVA_HOME="/usr/lib/jvm/java-21-openjdk-arm64/"' >> ~/.bashrc`
   **Il est important de supprimer la partie `/bin/java`.**
   **Utilisez votre propre chemin d'installation, voir l'étape 2 !**
4. Fermez et ouvrez le terminal, JAVA devrait être prêt à être utilisé. Vous pouvez confirmer le chemin à l'aide de la commande :
   `echo $JAVA_HOME`

### 1.1.2 Mac

**Variante 1**

1. Téléchargez l'installateur (DMG Installer) pour JDK 21 sous https://www.oracle.com/fr/java/technologies/downloads/#jdk21-mac.
   Si vous avez un Mac M1 ou M2, choisissez l'installateur Arm 64. Si vous avez un Mac basé sur Intel, choisissez l'installateur x64.
2. Lancez l'installateur et suivez les instructions.

**Variante 2 : Si vous utilisez Homebrew**

1. Ouvrez une fenêtre de terminal.
2. Installez le JDK par défaut à l'aide de la commande :
   `brew install openjdk`

**Définition de la variable `JAVA_HOME`.** De nombreux programmes écrits avec JAVA utilisent la variable d'environnement `JAVA_HOME` pour déterminer l'emplacement d'installation de JAVA.

1. Ouvrez une fenêtre de terminal.
2. Effectuez l'une des opérations suivantes :
   Si vous utilisez le shell bash
   `echo 'export JAVA_HOME=$(/usr/libexec/java_home)' >> ~/.bashrc`
   `echo 'export PATH="$JAVA_HOME:$PATH"' >> ~/.bashrc`
   Si vous utilisez le shell zsh
   `echo 'export JAVA_HOME=$(/usr/libexec/java_home)' >> ~/.zshrc`
   `echo 'export PATH="$JAVA_HOME:$PATH"' >> ~/.zshrc`
3. Fermez et ouvrez le terminal, JAVA devrait être prêt à être utilisé. Vous pouvez confirmer le chemin à l'aide de la commande :
   `echo $JAVA_HOME`

### 1.1.3 Windows

1. Téléchargez l'installateur (x64 MSI Installer) pour JDK 21 sous https://www.oracle.com/fr/java/technologies/downloads/#jdk21-windows.
2. Lancez l'installateur et suivez les instructions.

De nombreux programmes écrits avec JAVA utilisent la variable d'environnement `JAVA_HOME` pour déterminer l'emplacement d'installation de JAVA.

1. Trouvez l'endroit où JAVA est installé. Si vous n'avez rien changé, il devrait s'agir de quelque chose comme `C:\Program Files\Java\jdk-21.0.4\`
   Vous pouvez déplacer le logiciel vers un autre emplacement si vous le souhaitez.
2. Effectuez l'une des opérations suivantes :
   Windows 7 – Cliquez avec le bouton droit de la souris sur Poste de travail et sélectionnez Propriétés > Avancées.
   Windows 8 – Allez dans Panneau de configuration > Système > Paramètres système avancés
   Windows 10 – Recherchez Variables d'environnement puis sélectionnez Modifier les variables d'environnement du système.
3. Cliquez sur le bouton Variables d'environnement.
4. Sous Variables système, cliquez sur Nouveau.
5. Dans le champ Nom de la variable, entrez : `JAVA_HOME`
6. Dans le champ Valeur de la variable, entrez le chemin d'installation de votre JDK.
7. Cliquez sur OK et appliquez les modifications comme demandé.

## 1.2 Installation de Visual Studio Code

**Linux / Mac / Windows.** Téléchargez l'installateur correspondant à votre système d'exploitation à l'adresse suivante, et exécutez-le pour lancer l'installation.
https://code.visualstudio.com/download

Sous Linux (Ubuntu/Debian), on peut aussi simplement taper dans un terminal la commande :
```
sudo apt install code
```

# 2 Interface

L'interface utilisateur de VSCODE est typiquement divisée en trois parties (Voir Figure 1) :

(1) La colonne de gauche est l'explorateur de fichier. Il affiche l'arborescence des fichiers et dossiers ouverts dans le projet courant.

(2) La partie du bas est le terminal intégré. Vous pouvez y entrer les mêmes commandes que dans le terminal de Linux ; par exemple, `javac` et `java` permettent de compiler et exécuter un programme JAVA.
L'onglet « Problèmes » vous sera aussi très utile : vous y trouverez les messages d'erreur générés par votre programme, sans avoir à recompiler avec la commande `javac`.

(3) La zone principale, au centre, est l'éditeur de texte, où vous écrivez et modifiez votre code. Les onglets en haut indiquent les fichiers actuellement ouverts : par exemple, sur la Figure 1, on voit que les fichiers `Affiche.java`, `Ligne.java` et `Carre.java` ont été ouverts. Le rond blanc à droite de `Ligne.java` signifie que ce fichier a été modifié mais pas sauvegardé. Le chiffre "1" dans un rond bleu, tout en haut à gauche de l'écran, indique aussi qu'il y a un fichier non sauvegardé dans le répertoire courant.

De nombreux éléments de l'interface de VSCODE sont personnalisables, n'hésitez pas à fouiller les paramètres dans le menu `Fichier > Préférences`.

# 3 Extensions recommandées

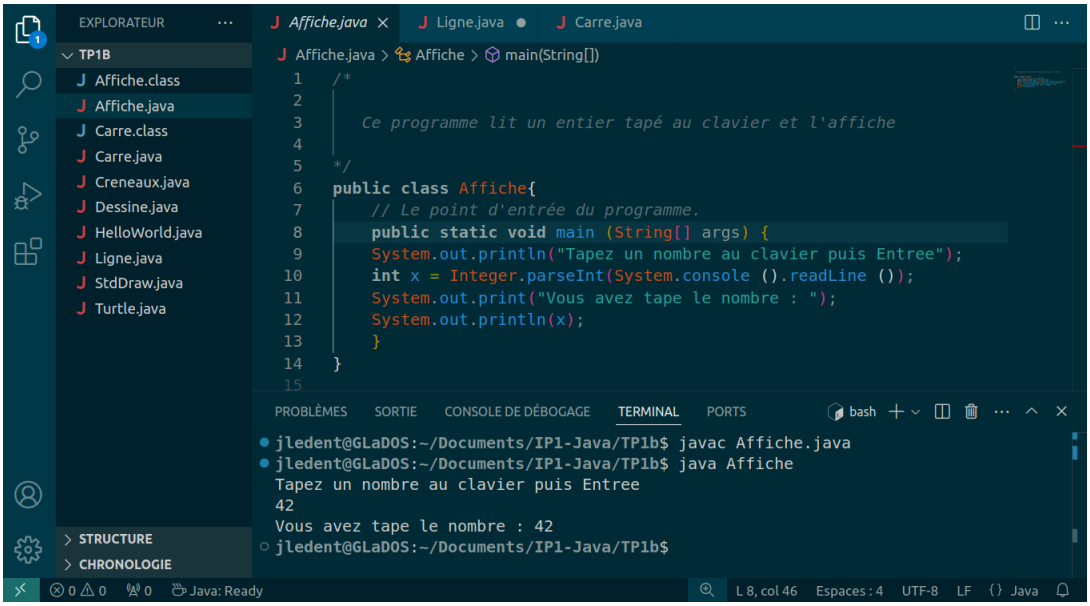



FIGURE 1 – Exemple d'interface utilisateur de VSCODE

VSCODE utilise un système *d'extensions* : des modules additionnels qui permettent de rajouter des fonctionnalités à l'éditeur. Dans un premier temps, on vous recommande de n'installer que le strict minimum. Vous pourrez en rajouter par la suite quand les besoins se feront sentir.

✓ **Language Support for Java(TM) by Red Hat.** C'est l'extension la plus importante pour programmer en JAVA : elle fournit la coloration syntaxique, l'auto-complétion de code, des messages d'erreur intégrés à l'éditeur, et bien d'autres fonctionnalités. Pour débuter, c'est la seule extension dont vous avez besoin.

✓ **French Language pack for Visual Studio Code.** Ce n'est pas nécessaire, mais si vous voulez que les menus de VSCODE s'affichent en Français, installez cette extension.

✗ **Extension Pack for Java.** C'est un pack d'extensions, qui contient divers outils utiles pour la gestion de projets, les batteries de tests, le debugging. Ne l'installez pas pour l'instant, ça va vous rajouter trop d'informations à l'écran. Mais jetez-y un œil à partir du deuxième semestre !

✗ **Désactivez les Inlay Hints.** *Important !* Par défaut, VSCODE affiche des informations « fantômes » qui s'insèrent dans votre code mais n'en font pas vraiment partie. L'utilité est discutable pour un programmeur, mais c'est carrément trompeur pour les étudiants. Désactivez les impérativement !

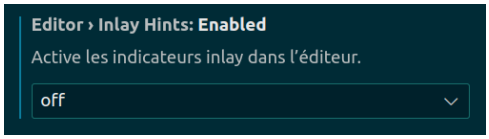
Par exemple, si vous tapez la ligne `System.out.println("Hello World!");` elle s'affichera comme suit dans l'éditeur :

Avec inlay hints : `System.out.println (x:"Hello World!");`

Sans inlay hints : `System.out.println ("Hello World!");`

Pour les désactiver :

1. Ouvrir les paramètres : en allant dans le menu `Fichier > Préférences > Paramètres`, ou bien avec le raccourci clavier `Ctrl` + `,`
2. Taper "inlay hints" dans la barre de recherche des paramètres.
3. Désactiver les inlay hints :




✗ **Désactivez l'ampoule d'action de code.** C'est un petit symbole d'ampoule jaune qui s'affiche à gauche de la ligne courante. Cliquer dessus propose diverses actions pour améliorer ou corriger votre code. Ce n'est pas une solution magique : si vous l'utilisez sans comprendre ses suggestions, cela va bien souvent empirer la situation !

Pour la désactiver : comme pour les Inlay Hints, ouvrir les paramètres, taper "lightbulb" dans la barre de recherche, et sélectionner `off` dans le menu déroulant.

✗ **IA de génération de code.** VSCode permet d'intégrer des outils d'Intelligence Artificielle (IA) très puissants, capables de générer des blocs entiers de code. Ces IAs sont capables de résoudre la plupart des exercices de niveau L1/L2. **On vous déconseille très fortement de les utiliser.** Ces outils ne sont pas intégrés par défaut dans VSCode ; il vous suffit donc de ne pas en installer.

```
3     public static void eratosthene(int n) {
4         boolean[] isPrime = new boolean[n + 1];
          for (int i = 2; i <= n; i++) {
              isPrime[i] = true;
          }
          for (int i = 2; i * i <= n; i++) {
              if (isPrime[i]) {
                  for (int j = i * i; j <= n; j += i) {
                      isPrime[j] = false;
                  }
              }
          }
          for (int i = 2; i <= n; i++) {
              if (isPrime[i]) {
                  System.out.println(i);
              }
          }
5     }
```

Figure 2 – Un nom de fonction suffit pour que l'IA devine la fonction complète.

Citons quelques raisons qui font que les IA génératrices de code sont un gros frein à l'apprentissage.

(a) Les suggestions de code intégrées à l'éditeur de texte empêchent de se concentrer. C'est comme avoir quelqu'un qui vous souffle en permanence les réponses au lieu de vous laisser réfléchir.

(b) Chaque exercice n'a pas une seule, mais plusieurs solutions. Votre manière de procéder sera toujours un peu différente de ce que suggère l'IA. Il est beaucoup plus satisfaisant de réussir à faire fonctionner **votre** solution. Lire une solution proposée par l'IA bride votre créativité.

(c) Les IA utilisent parfois des fonctions avancées pour résoudre un exercice, alors qu'une solution bien plus basique était possible. De manière générale, les exercices sont conçus pour travailler les notions qui viennent d'être vues en cours ; mais l'IA ne peut pas savoir dans quel contexte l'exercice a été proposé.

(d) Faire des erreurs est une partie cruciale de l'apprentissage de la programmation. Le processus de recherche et de correction de ses propres erreurs est essentiel pour devenir un programmeur compétent. Il faut comprendre les failles de son raisonnement, comprendre les messages d'erreur du compilateur, acquérir des automatismes pour écrire du code correct, apprendre à débugger son code.

(e) Les IA font parfois des erreurs, souvent subtiles, même sur des exercices de niveau L1. Pour les utiliser efficacement, il faut non seulement comprendre le code qu'elles produisent, mais aussi se l'approprier pour pouvoir le corriger et le modifier selon ses besoins. C'est un travail difficile, qui demande un certain recul.

(f) Au final, faire générer du code par IA, c'est se priver de la partie la plus intéressante (l'écriture du programme), pour ne garder que le plus pénible (le débuggage) !

# 4 Tips & Tricks

VSCode inclut de nombreuses fonctionnalités pour faciliter la vie des programmeurs. On en liste quelques unes ci-dessous, mais bien sûr, la liste n'est pas exhaustive.

## 4.1 Indentation automatique

En Java, l'indentation[1] est optionnelle : elle permet de rendre le code plus lisible, mais elle ne modifie pas son fonctionnement. Imaginons que suite à une mauvaise manip, votre code se retrouve mal indenté :

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

Il suffit d'utiliser le raccourci clavier Ctrl + Shift + I pour réindenter automatiquement votre code :

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

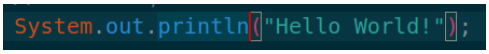
## 4.2 Commenter du code

Il est parfois utile de commenter une ou plusieurs lignes de code, pour les désactiver temporairement sans pour autant les supprimer. Le raccourci clavier Ctrl + / permet de commenter la ou les lignes courantes. Pour commenter un bloc entier de code, on peut aussi utiliser Ctrl + Shift + A.

## 4.3 Trouver une parenthèse manquante

Quand on programme, de nombreux symboles vont par deux : les parenthèses ( ), les accolades { }, les crochets [ ], les chevrons < >, les guillemets " ", etc. Une erreur courante, surtout chez les débutants, est d'oublier de fermer une parenthèse ou une accolade. Plusieurs indices peuvent vous aiguiller :

(1) Placez le curseur juste avant une parenthèse ouvrante : celle-ci sera surlignée, ainsi que la parenthèse fermante qui va avec :




Si ce n'est pas celle à laquelle vous vous attendiez qui s'allume, le problème n'est pas loin.

(2) Si la coloration syntaxique se comporte de façon inattendue, c'est sans doute un problème de ponctuation. Par exemple, si on supprime un guillemet :

1. Le mot *indentation* désigne les espaces en début de ligne qui permettent de mettre en évidence différents blocs de code comme les fonctions, les boucles, les conditions.

```
6   public class Affiche {
7       // Le point d'entrée du programme.
8       public static void main(String[] args) {
9           System.out.println("Tapez un nombre au clavier puis Entree");
10          int x = Integer.parseInt(System.console().readLine());
11          System.out.print("Vous avez tape le nombre : ");
12          System.out.println(x);
13      }
14  }
```

(3) Si l'indentation automatique se comporte de façon inattendue (des lignes se décalent alors qu'elles ne devraient pas), l'accolade manquante n'est pas loin.

## 4.4 Toujours ouvrir un dossier, pas seulement un fichier `.java`

Lorsque vous travaillez en JAVA sur VSCODE, il faut toujours passer par le menu « Ouvrir le dossier », puis sélectionner le dossier qui contient les fichiers `.java` sur lesquels vous voulez travailler. Si vous ouvrez uniquement un fichier `.java`, vous n'aurez accès qu'à des fonctionnalités réduites. Si vous faites l'erreur, vous verrez s'afficher le message suivant dans l'onglet « Problèmes » du terminal intégré de VSCODE :