

TP 02

INSTALLATION IDE ET PREMIERE APPLICATION

INSTALLER ECLIPSE

- Cliquez sur le lien suivant : <https://www.eclipse.org/downloads/packages/>
- Télécharger et installer Eclipse pour Développeur Java

OUVREZ VOTRE IDE

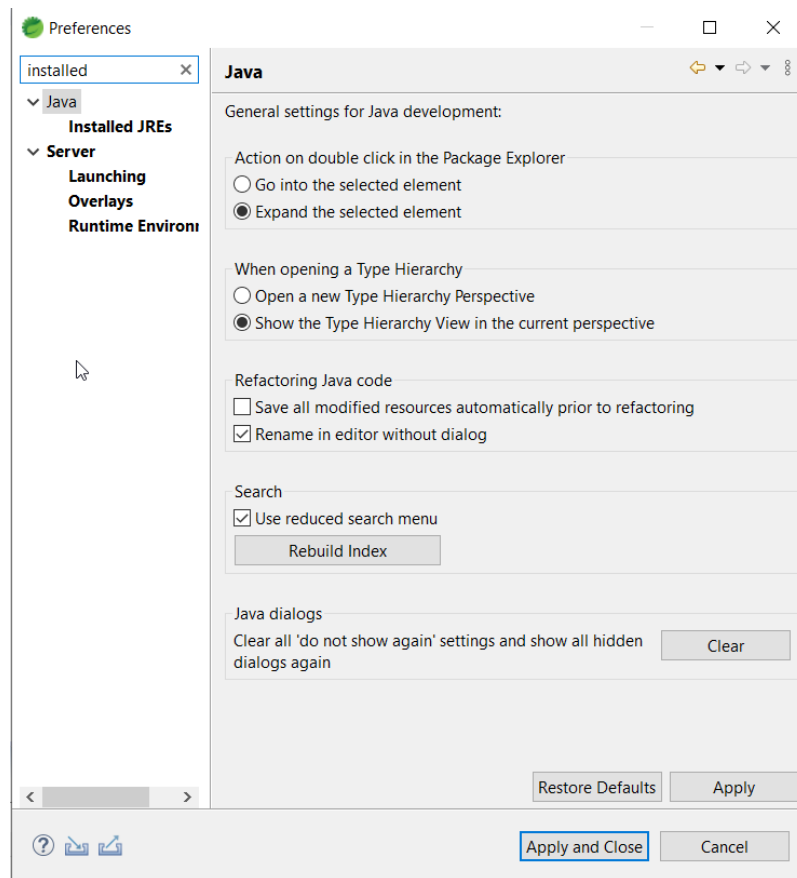
- Lancer Eclipse.
- Par défaut un espace de travail devrait être créé dans votre espace utilisateur.
Exemple sous Windows :

C:\Users\myAccount\Documents\workspace-xxx

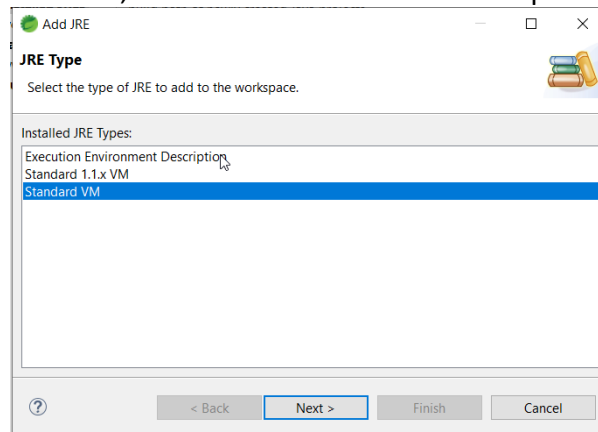
VERIFIONS QUE LE JDK17 EST PRIS EN COMPTE PAR VOTRE IDE

- 1) Dans la barre de menu de votre IDE, cliquez sur **Window**, puis **Preferences**

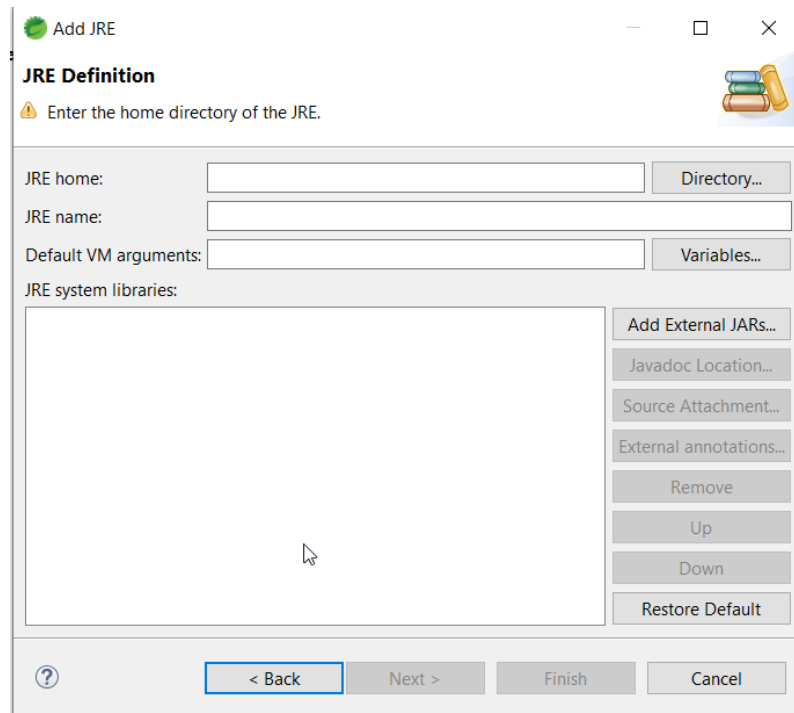
Une fenêtre s'ouvre :



- 2) Dans la zone de saisie de texte située en haut à gauche, saisissez le mot « **installed** » qui va permettre de trier le menu vertical du dessous.
- 3) Vous devriez alors voir apparaître « Installed JREs ». **Cliquez sur cette option de menu.**
- 4) Cliquez sur le bouton « **Add** » situé sur la droite.
- 5) Une fenêtre s'ouvre alors, sélectionnez « Standard VM » puis cliquez sur **Next**:



- 6) Une nouvelle fenêtre d'ouvre qui va vous permettre de sélectionner le répertoire d'installation :



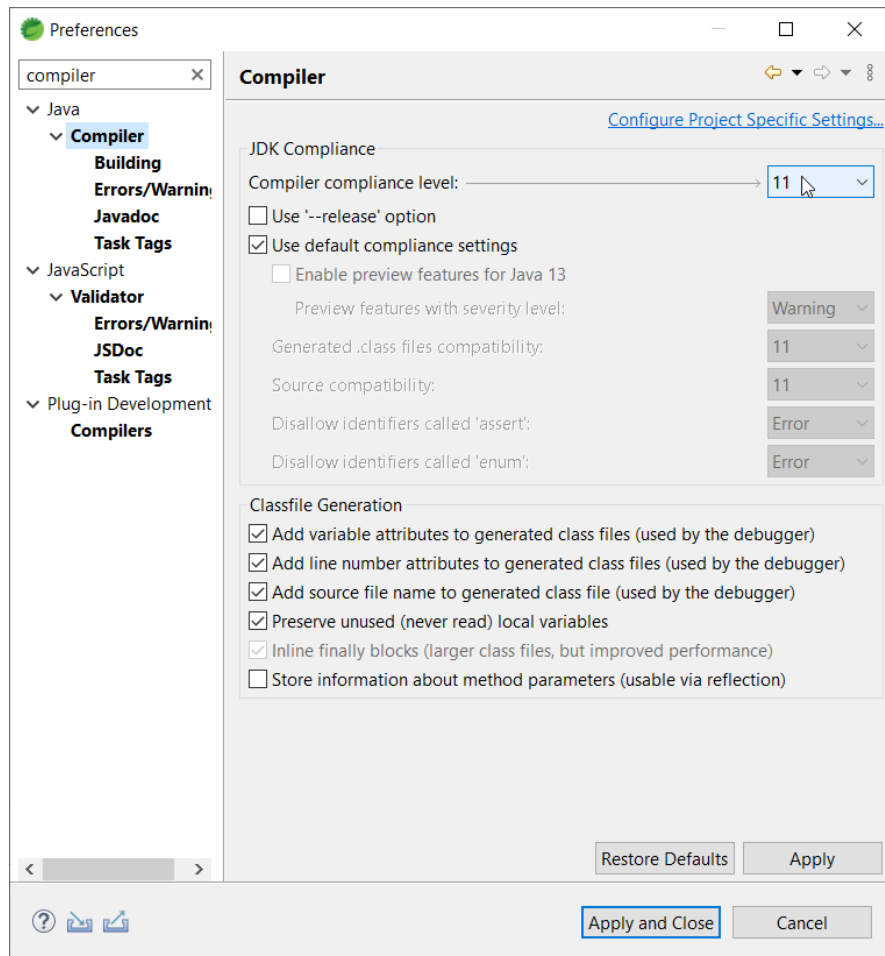
- 7) Cliquez sur le bouton « Directory » et sélectionnez le répertoire d'installation de votre JDK 11.

Par exemple le répertoire **C:\Program Files\Java\jdk-17.0.1_1**

- 8) Cliquez sur Finish.
- 9) Enfin, sélectionnez ce JDK parmi ceux affichés et cliquez sur « **Apply** » puis « **Apply and close** ».

VERIFIONS QUE L'IDE UTILISE LA BONNE VERSION DU COMPILATEUR, A SAVOIR LE 17

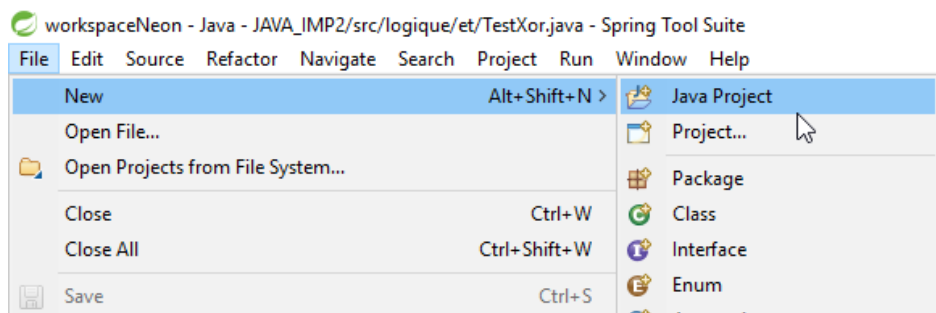
- 1) Dans la barre de menu de votre IDE, cliquez sur **Window**, puis **Preferences**
- 2) Dans la zone de saisie de texte située en haut à gauche, saisissez « compiler » :



- 3) Dans la liste de sélection « JDK compliance » sélectionnez **17**.
- 4) Enfin, cliquez sur « **Apply** » puis « **Apply and close** ».

CREEZ LE PROJET APPROCHE-IMPERATIVE

- Une fois l'IDE démarré, cliquez sur **File -> new -> Project** pour créer un nouveau projet appelée **approche-imperative**.
- Ignorez la création du fichier **module-info.java** en cliquant sur **Don't create**
- Créez un dépôt local avec la commande `git init`
- Créez le fichier `.gitignore` et ajoutez les exclusions suivantes
 - *.class
 - /.settings
 - .classpath
 - .project
- Créez un dépôt distant sur **GitHub** portant le nom **approche-imperative**
- Utilisez la commande **git remote** (cf instructions fournies par GitHub une fois votre dépôt distant créé) pour synchroniser votre dépôt local avec votre dépôt distant situé sur **GitHub**.



CREEZ UNE APPLICATION

- Créez un package : « fr.declaration.variable »
- Créez une classe « **DeclarationApp** » dans le package « fr.declaration.variable ».
- Ajoutez une méthode `public static void main(String[] args)` dans cette classe afin qu'elle soit exécutable.

DECLAREZ LES VARIABLES

- Dans la méthode `main`, déclarez une variable de chacun des types suivants avec un littéral différent de 0 pour les chiffres :
 - byte
 - short
 - int
 - long
 - float

- double
- char
- boolean
- String

AFFICHAGE DE LA VALEUR D'UNE VARIABLE

En Java pour afficher un résultat on utilise `System.out.println(...)` et on met entre parenthèses ce qu'on veut afficher :

- une variable
- directement un littéral.

Testez cette méthode pour afficher une variable déclarée.

Retour à la ligne dans une chaîne de caractère

Déclarer la variable suivante :

`String randomString = "Voici le résultat d'un calcul : 1+5=6";`

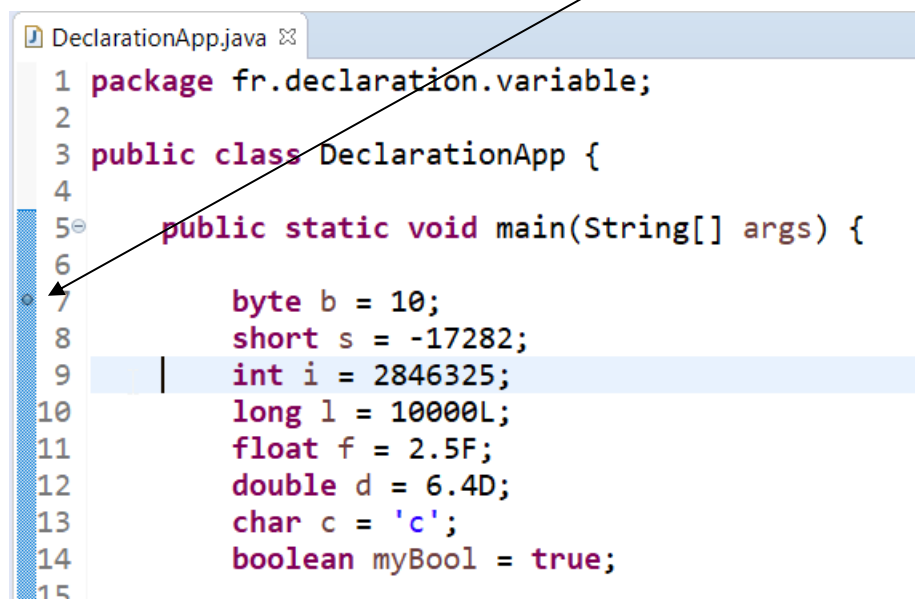
Utilisez la méthode `System.out.println(...)` pour afficher `randomString`.

Modifiez la chaîne de caractères afin de mettre un saut de ligne après le caractère « : ».

Utilisation du Debugger


Nous allons maintenant apprendre à utiliser un outil très important : le debugger.

- 1) Reprenez votre classe `DeclarationApp` et double-cliquez dans la marge au niveau de la première variable déclarée :

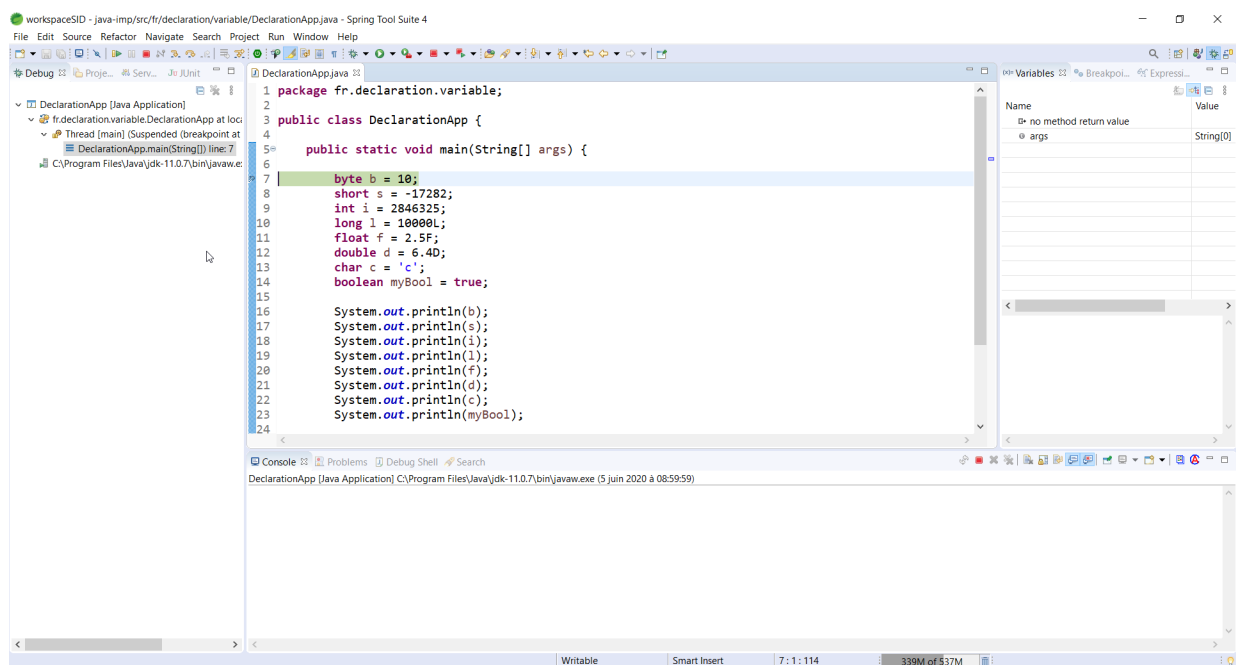


Cette action fait apparaître ce qu'on appelle un point d'arrêt.

2) Nous allons désormais démarrer l'application en mode **debug**. Pour cela il y a 2 possibilités :


- première possibilité : vous faites un clic droit sur votre classe et vous sélectionnez « Debug as » puis « Java application ».
- seconde possibilité : vous cliquez sur l'icône en forme de punaise :  dans la barre du menu. Attention si vous cliquez sur cette icône vous lancez en mode **debug** la dernière application exécutée. Pour sélectionner une application spécifique vous pouvez être amenée à cliquer sur la flèche noire descendante située à droite de la punaise.

3) Une fois cette action effectuée, l'IDE va vous proposer de passer sur la perspective DEBUG. Acceptez cette proposition. L'application se lance et au bout de quelques secondes vous voyez l'exécution du code suspendue au niveau de votre point d'arrêt. La ligne correspondante est surlignée en vert :





Pour avancer dans le code, vous disposez de plusieurs icônes de contrôle du debugger :



 : La flèche verte, qu'on appelle « **Resume** » permet d'aller jusqu'au prochain point d'arrêt, ou de terminer l'exécution de l'application s'il n'y en a pas d'autre.

 : Le bouton rouge permet de stopper le debugger.

 : La flèche descendante, appelée « **Step into** » permet de rentrer dans une méthode, s'il y en a une qui est appelée sur la ligne courante.

 : La flèche qui fait un demi-tour, appelée « **Step over** » permet d'avancer d'une ligne.

- 4) Avancez maintenant d'une ligne et consultez la vue « Variables » située en haut à droite. Vous devriez constater que votre première variable est apparue avec sa valeur affichée.
- 5) Alors que le debugger continue de tourner, ajoutez un nouveau point d'arrêt dans le code. Cliquez sur Resume pour aller directement jusqu'à ce point.
- 6) Vérifiez que toutes vos variables apparaissent bien dans la vue « Variables ».

Par la suite, utilisez le debugger autant que vous le pouvez. Une bonne maîtrise du debugger est requise en milieu professionnel.

COMMITEZ SUR GITHUB