



# Programmation Orientée Objet

Présentation de java et environnement

**Mme Hassna BENSAG**  
Email: [h.bensag@gmail.com](mailto:h.bensag@gmail.com)

# Langage de programmation Java

---

- Java est créé par James Gosling.
- La première implémentation a été publiée en 1995.
- Java est utilisé dans plusieurs types d'applications telles que :
  - Les applications mobiles (Android) ;
  - Les applications Desktop ;
  - Les applications Web ;
- Les applications Java sont compilées en bytecode qui peut s'exécuter sur n'importe quelle machine virtuelle Java.

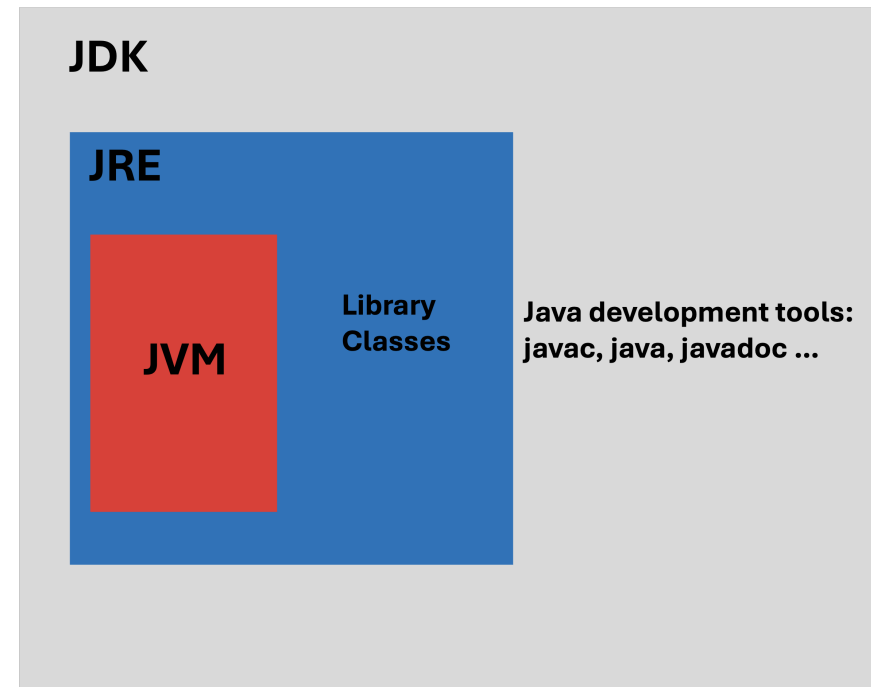
# Langage de programmation Java

---

- **Java** est un langage de programmation orienté objet (classe, héritage, polymorphisme..).
- Java est un langage **Multiplateforme**.
- **Open source**
- **Distribué** sur un ou plusieurs machines connectées via internet
- **Multithreading**: elle permet l'exécution simultanée de plusieurs programmes pour utilisation maximale des processeurs.

# Le Kit de développement Java

- **JDK** (Java Development Kit): JDK est destiné aux développeurs de logiciels et comprend des outils de développement java tels que le compilateur java, Javadoc, un débogueur.
- **JRE** (Java Runtime Environment) : JRE contient les parties des bibliothèques Java requises pour exécuter des programmes Java et est destiné aux utilisateurs finaux. JRE peut être considéré comme un sous-ensemble de JDK.
- **JVM** (Java Virtual Machine) : Il s'agit d'une spécification qui fournit un environnement d'exécution dans lequel le bytecode Java peut être exécuté. Les JVM sont disponibles pour de nombreuses plates-formes matérielles et logicielles.



# Configuration de l'environnement Java

- **Etape 1:** Télécharger Java 21 JDK depuis le lien suivant:

<https://www.oracle.com/java/technologies/downloads/#java21>

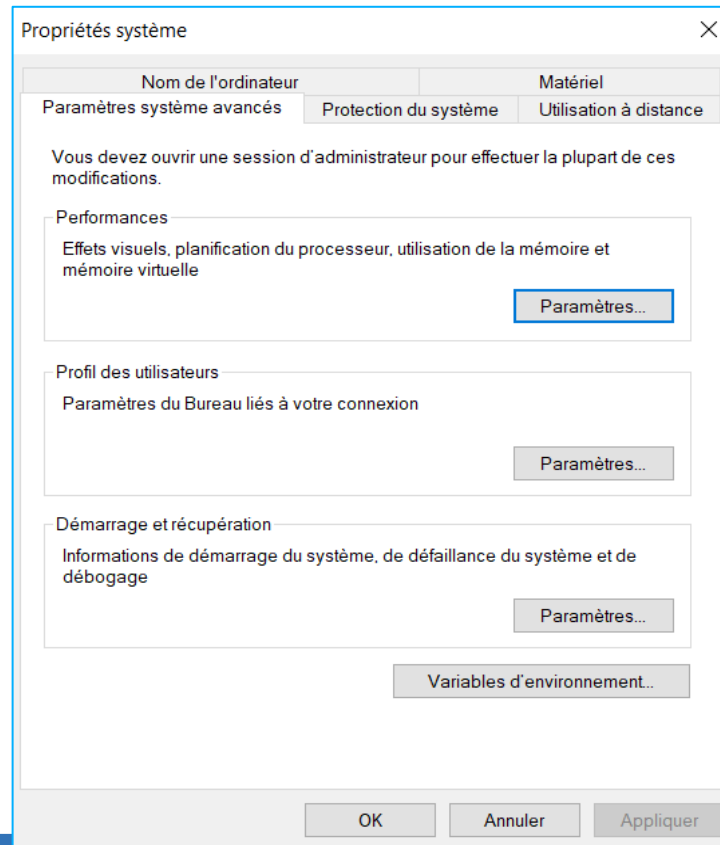
The screenshot shows the Oracle Java 21 JDK download page. The header includes the Oracle logo and navigation links: Products, Industries, Resources, Customers, Partners, Developers, and Company. There are also links for View Accounts and Contact Sales. The main navigation bar highlights Java downloads, Tools and resources, and Java archive. The page content shows the JDK 21 section, which includes a link to the JDK Development Kit 21.0.4 downloads. Below this, there is a table of download links for Linux, macOS, and Windows. The table has three columns: Product/file description, File size, and Download. The download links are provided for x64 Compressed Archive, x64 Installer, and x64 MSI Installer.

Product/file description	File size	Download
x64 Compressed Archive	185.84 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip</a> (sha256)
x64 Installer	164.23 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe</a> (sha256)
x64 MSI Installer	162.97 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi</a> (sha256)

Documentation Download

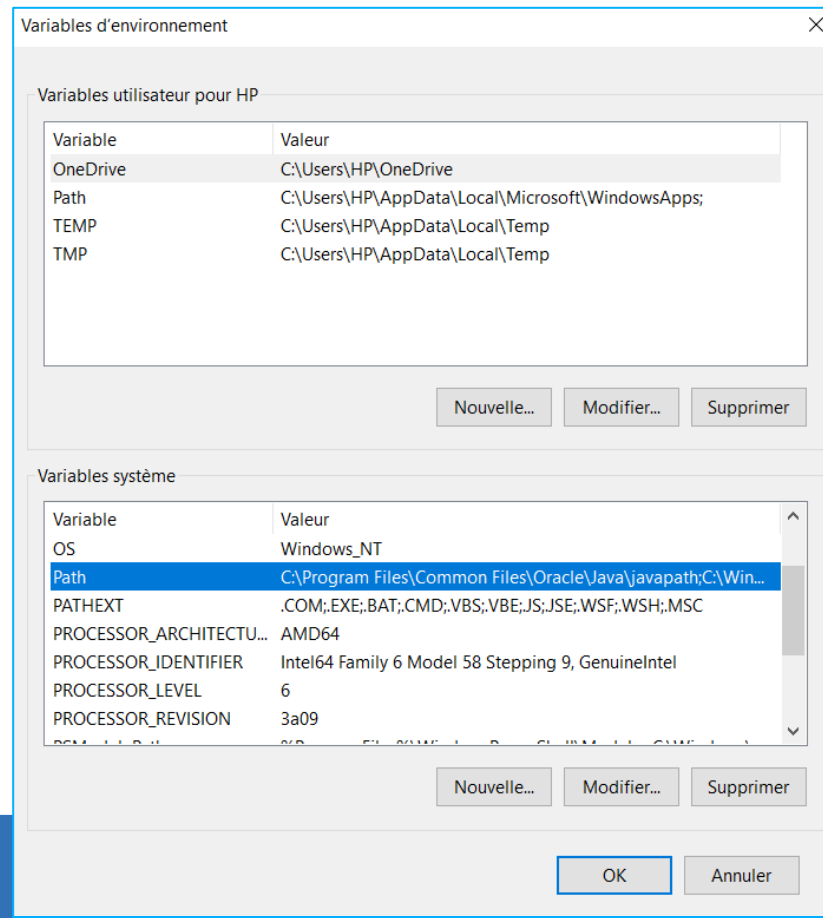
# Configuration de l'environnement Java

- **Etape 2:** Depuis Panneau de configuration -> Système et sécurité -> Système. Accédez au bouton « variables d'environnement », sous l'onglet paramètres système avancés .



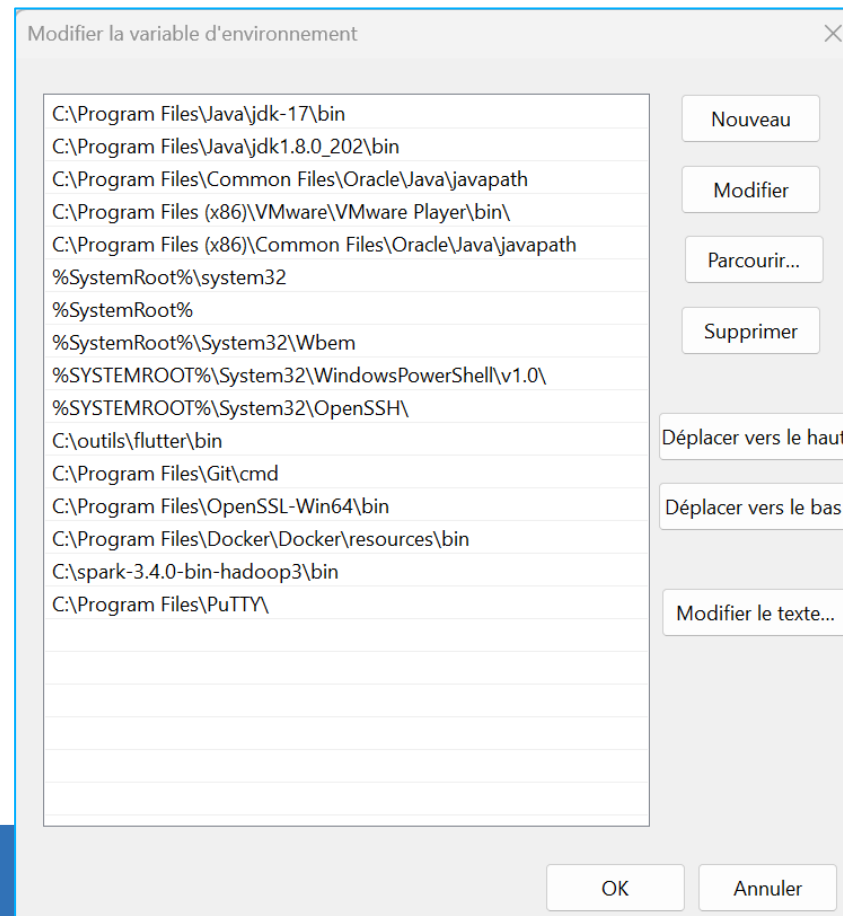
# Configuration de l'environnement Java

- **Etape 3:** Maintenant, vous devez modifier la variable "path" sous Variables système afin qu'elle contienne également le chemin d'accès à l'environnement Java. Sélectionnez la variable path et cliquez sur le bouton Modifier comme indiqué ci-dessous.



# Configuration de l'environnement Java

- **Etape 4:** Vous verrez une liste de différents chemins, cliquez sur le bouton Nouveau, puis ajoutez le chemin où Java est installé. Par défaut, Java est installé dans le dossier C:\Program Files\Java\jdk1.8.0\_202\bin. Si vous avez installé Java à un autre emplacement, ajoutez ce chemin.

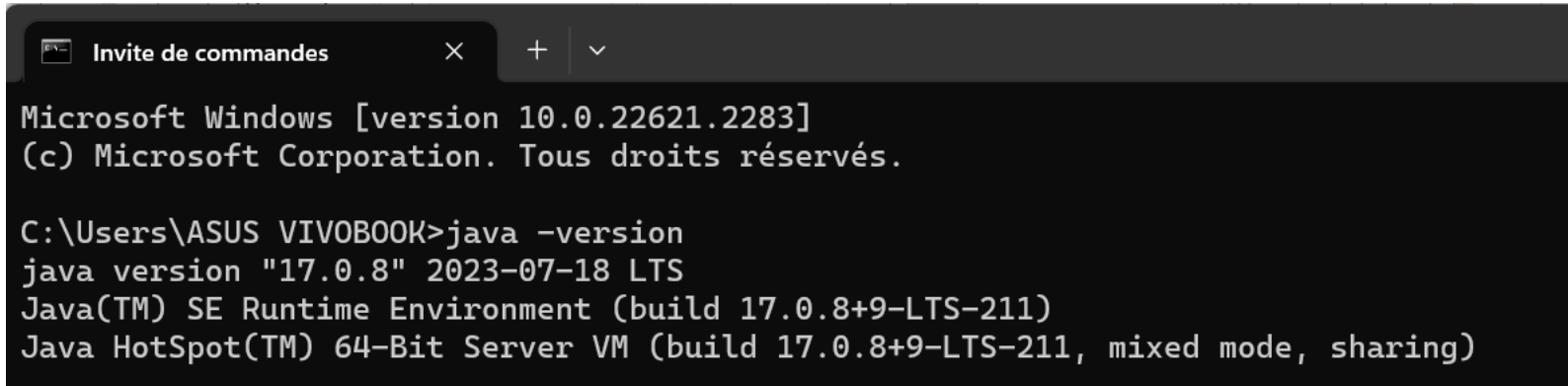




# Configuration de l'environnement Java

- **Etape 5:** Cliquez sur OK, enregistrez les paramètres, et vous avez terminé !! Maintenant, pour vérifier si l'installation est effectuée correctement, ouvrez l'invite de commande et tape:

**java -version**



```
Microsoft Windows [version 10.0.22621.2283]
(c) Microsoft Corporation. Tous droits réservés.

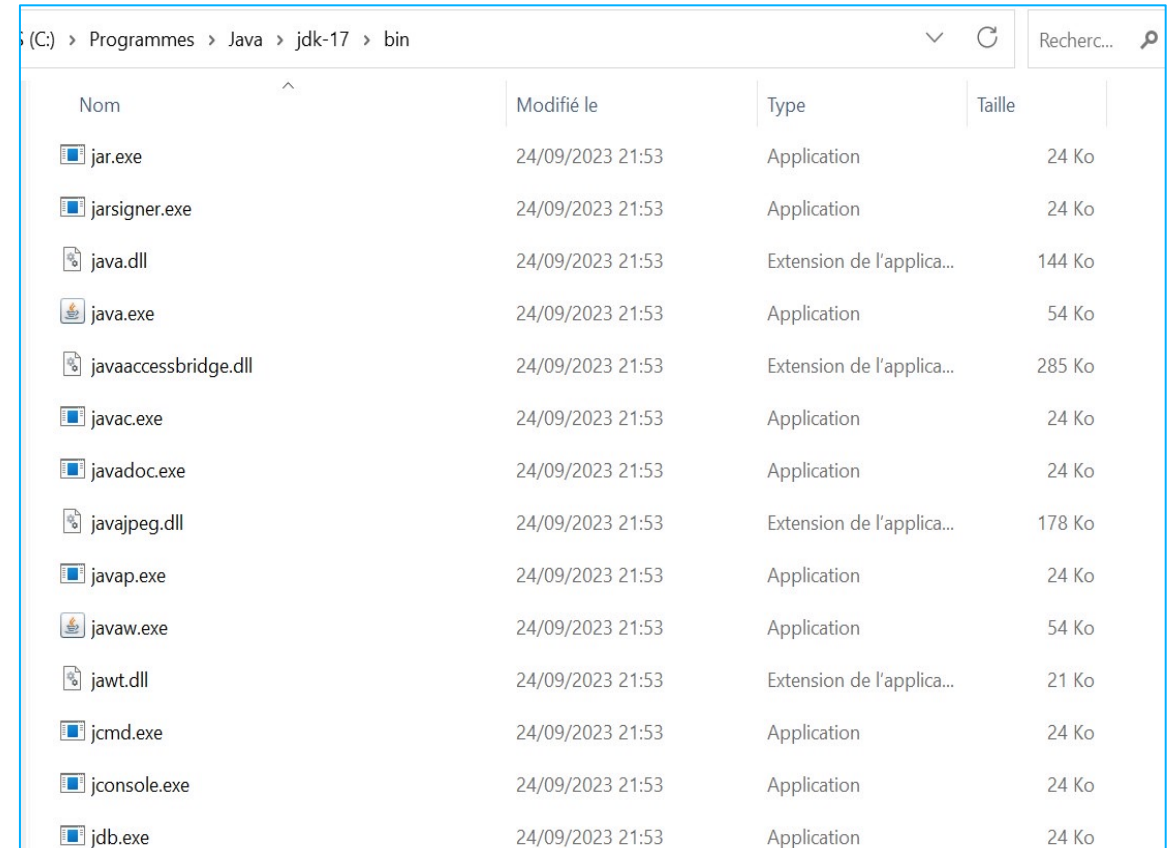
C:\Users\ASUS VIVOB00K>java -version
java version "17.0.8" 2023-07-18 LTS
Java(TM) SE Runtime Environment (build 17.0.8+9-LTS-211)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.8+9-LTS-211, mixed mode, sharing)
```

# Ce que contient le JDK

- Les programmes nécessaires au développement java sont placés dans le répertoire

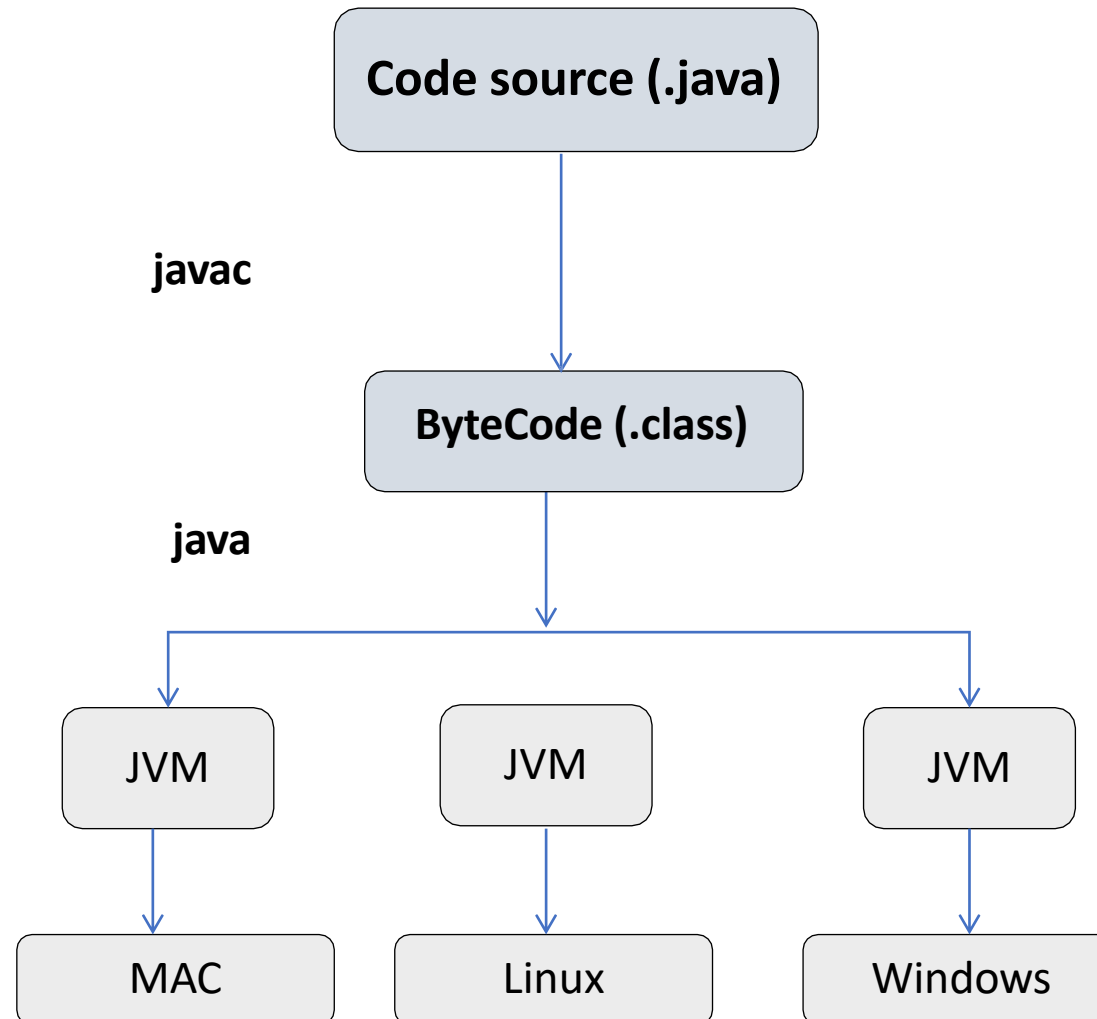
**C:\ProgramFiles\Java\jdk1.8.0\_202\bin** à savoir :

- **javac.exe** : compilateur java.
- **java.exe** : interpréteur du bytecode java.
- **jdb.exe** : débogueur java.
- **javadoc.exe** : générer la documentation de vos programmes java.
- **jar.exe** : Permet de compresser les classes Java ainsi que tous les fichiers nécessaires à l'exécution d'un programme (graphiques, sons, etc.).



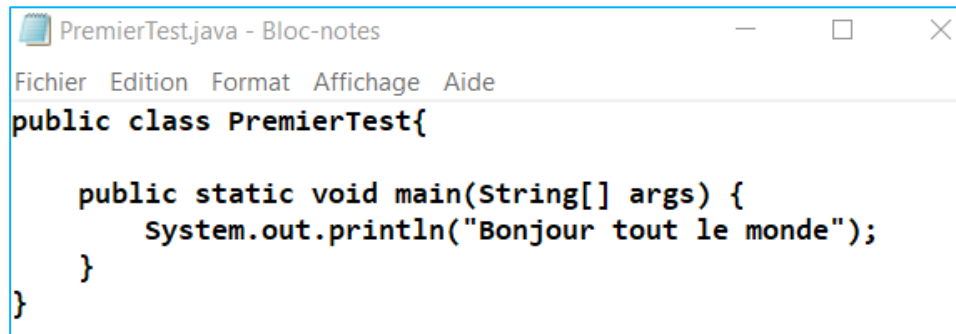
(C:) > Programmes > Java > jdk-17 > bin					Recherch...
Nom	Modifié le	Type	Taille		
jar.exe	24/09/2023 21:53	Application	24 Ko		
jarsigner.exe	24/09/2023 21:53	Application	24 Ko		
java.dll	24/09/2023 21:53	Extension de l'applica...	144 Ko		
java.exe	24/09/2023 21:53	Application	54 Ko		
javaaccessbridge.dll	24/09/2023 21:53	Extension de l'applica...	285 Ko		
javac.exe	24/09/2023 21:53	Application	24 Ko		
javadoc.exe	24/09/2023 21:53	Application	24 Ko		
javajpeg.dll	24/09/2023 21:53	Extension de l'applica...	178 Ko		
javap.exe	24/09/2023 21:53	Application	24 Ko		
javaw.exe	24/09/2023 21:53	Application	54 Ko		
jawt.dll	24/09/2023 21:53	Extension de l'applica...	21 Ko		
jcmd.exe	24/09/2023 21:53	Application	24 Ko		
jconsole.exe	24/09/2023 21:53	Application	24 Ko		
jdb.exe	24/09/2023 21:53	Application	24 Ko		

# Le processus d'exécution d'un programme Java



# Premier programme sans IDE

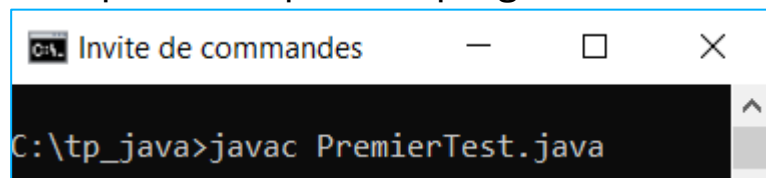
- Ecrire mon premier programme:



```
PremierTest.java - Bloc-notes
Fichier Edition Format Affichage Aide
public class PremierTest{

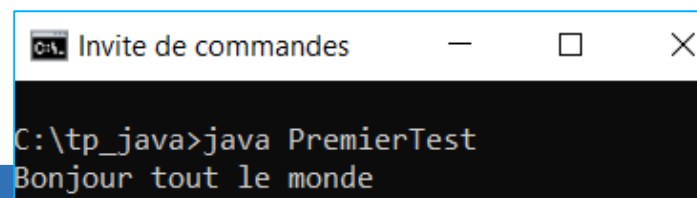
    public static void main(String[] args) {
        System.out.println("Bonjour tout le monde");
    }
}
```

- Compiler mon premier programme:



```
Invite de commandes
C:\tp_java>javac PremierTest.java
```

- Exécuter mon premier programme:



```
Invite de commandes
C:\tp_java>java PremierTest
Bonjour tout le monde
```

Création de **PremierTest.java**  
Avec **Bloc-notes**  
(Code source)

Compilation de Test.java  
Avec **javac PremierTest.java**  
Crée **PremierTest.class**  
(ByteCode)

Exécution de PremierTest.class  
Avec **java PremierTest**  
Exécute le programme

# Les fichiers .jar

---

- Un fichier jar (java archive) est une archive qui est utilisée pour envelopper des fichiers compilés .class, ainsi que toutes les ressources utilisées (images, audio, configuration, etc...).
- Un fichier .jar est basé sur le format Zip, on peut cependant renommer les fichiers .jar avec l'extension.zip et les manipuler avec les outils ZIP (7-Zip, WinZip, WinRAR, etc...).
- Le JDK fournit l'outil jar qui permet de créer ou extraire un fichier jar, visualiser son contenu, le modifier etc.
- Un fichier .jar peut être distribué et exécuté s'il contient au moins une classe avec une méthode main(). Ou peut être utilisé comme une bibliothèque qui sera utilisée par d'autres applications.
- Le fichier jar contient un fichier Manifest qui permet de préciser des informations d'exécution sur le fichier jar (classe principale de l'application, classpath, ...).

# Les fichiers .jar

- Créer un fichier JAR

```
C:\java_projects>jar cvfe myApp.jar Application Application.class
manifeste ajouté
ajout : Application.class(entrée = 437) (sortie = 298)(compression : 31 %)
```

- Afficher le contenu du fichier jar

```
C:\java_projects>jar tf myApp.jar
META-INF/
META-INF/MANIFEST.MF
Application.class
```

- Exécuter un fichier jar

```
C:\java_projects>java -jar myApp.jar
Bonjour tout le monde
```

- Extraire un fichier jar

```
C:\java_projects>jar xf myApp.jar
```

- L'option **c** est utilisée pour créer un fichier JAR.
- L'option **f** permet de spécifier le nom de JAR à créer.
- L'option **t** affiche le contenu du fichier JAR.
- L'option **x** est pour extraire le fichier JAR.
- L'option **v** produit un affichage détaillé sur la console pendant la construction du fichier JAR. L'affichage indique le nom de chaque fichier lorsqu'il est ajouté au fichier JAR.
- l'option **m** utilisé pour inclure des informations de manifeste à partir d'un fichier manifeste existant.

# Le classpath

---

- Le **classpath** en java permet de décrire au compilateur et à la JVM l'emplacement où sont disponibles les classes nécessaires pour la compilation et l'exécution d'une application.
- Si le classpath n'est pas défini par le programmeur, sa valeur par défaut sera "." (point), ce qui signifie que uniquement les classes du répertoire courant qui peuvent être chargées.
- Le classpath peut être défini avec trois méthodes :
  - à l'aide de la variable d'environnement **classpath**;
  - En utilisant l'option **-cp** ou **-classpath** dans la ligne de commandes;
  - ou de l'attribut **Class-Path** dans le fichier **Manifest.mf** à l'intérieur du fichier JAR en Java.

# Outils de développement java

---

- Pour développer des application java, on peut utiliser un simple éditeur mais il est préférable d'utiliser un éditeur conçu pour la programmation java exemples: IntelliJ IDEA , eclipse, ....
- **IntelliJ IDEA** est l'un des meilleurs IDE pour le développement Java, et parmi les plus utilisés en entreprise.
- Autres IDE java :
  - eclipse.
  - NetBeans.
  - JDeveloper.
  - JCreator.



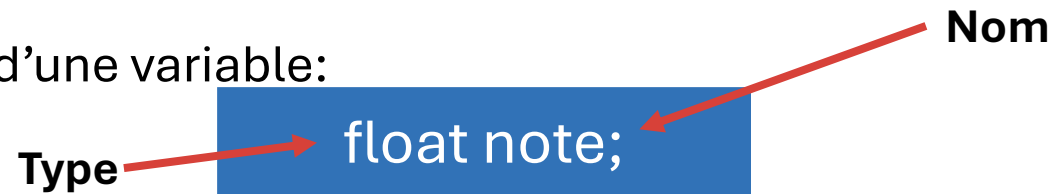


# Instructions de base en java

# Variables en Java

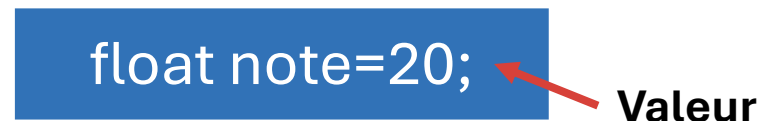
- Une variable est un nom donné à un emplacement mémoire pour stocker une valeur.
  - La valeur d'une variable peut être modifiée pendant l'exécution du programme.
  - Une variable n'est qu'un nom donné à un emplacement mémoire, toutes les opérations effectuées sur la variable affectent cet emplacement mémoire.
  - En Java, toutes les variables doivent être déclarées avant utilisation.

- Déclaration d'une variable:



The diagram shows the code `float note;` inside a blue rectangular box. A red arrow points from the word **Type** to the word `float`. Another red arrow points from the word **Nom** to the word `note`.

- Initialiser une variable:



The diagram shows the code `float note=20;` inside a blue rectangular box. A red arrow points from the word **Valeur** to the number `20`.

# Règles de nommage d'une variable

---

- Pour respecter la typologie de java, les noms des variables commencent toujours par un caractère en minuscule et pour indiquer un séparateur de mots, on utilise les majuscules. Exemples :

```
int note;
```

```
String nomPersonne;
```

# Types primitifs de données en Java

Type	Taille	Étendue
boolean	1 bit	true ou false
byte	8 bits	-128 à +127
char	16 bits	0 à 65 535
short	16 bits	-32 768 à +32 767
int	32 bits	-2 147 483 648 à + 2 147 483 647
long	64 bits	$-9,223 \times 10^{18}$ à $9,223 \times 10^{18}$
float	32 bits	$\pm 1.4 \times 10^{-45}$ à $\pm 3.4 \times 10^{38}$
double	64 bits	$\pm 4.9 \times 10^{-324}$ à $\pm 1.8 \times 10^{308}$
void	0 bit	-

# Transtypage des données primitives

- Également appelé cast, consiste à effectuer une conversion d'un type vers un autre.
- Il y a deux types de casting :
  - **sur-casting** : convertit un type de données plus particulier vers un type de données plus général, il peut se faire implicitement ou explicitement.

```
int a=8;  
long b;  
b=a;    // Casting implicite  
b=(long)a; //Casting explicite
```

- **sous-casting** : convertit un type de données général vers un type de données plus particulier et il ne peut pas se faire qu'explicitement.

```
float a=(float)7.5;  
double b=5;  
byte c=(byte)b;  
int d=6;  
byte e=(byte)d;
```

# Les enveloppeurs (wrappers)

- Les types primitifs sont enveloppés dans des objets appelés enveloppeurs (wrappers ). Les enveloppeurs sont des classes.
- Le tableau ci-dessous montre le type primitif et la classe wrapper équivalente :

Type de données primitif	La classe
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

# Les enveloppeurs (wrappers) - Exemple

```
public class Application {  
    public static void main(String args[]) {  
        int a1=5; // a1 est une primitive  
        Integer a2=3; // a2 est un objet  
        Integer a3=new Integer( value: 4); // a3 est un objet  
        System.out.println(a1);  
        System.out.println(a2.intValue());  
        System.out.println(a3.intValue());  
        // exemple pour les doubles  
        double b1=6.5; // b1 est une primitive  
        Double b2=4.6; // b2 est un objet  
        Double b3=new Double( value: 8.2); // b3 est un objet  
        System.out.println(b1);  
        System.out.println(b2.doubleValue());  
        System.out.println(b3.doubleValue());  
    }  
}
```



5
3
4
6.5
4.6
8.2

# Commentaires en java

---

- Commentaire sur une seule ligne:

```
//ceci est un commentaire en java
```

- Commentaire sur plusieurs lignes:

```
/* ceci  
est un commentaire sur plusieurs lignes  
*/
```

- Commentaires de documentation:

```
/**début du commentaire  
*  
*Ceci est un commentaire de documentation  
*  
*fin du commentaire*/
```



# Les opérateurs arithmétiques

Opérateur	Description
+	opérateur d'addition
-	opérateur de soustraction
*	opérateur de multiplication
/	opérateur de division
%	opérateur de modulo (reste de la division entière)

# Les opérateurs de comparaison

Opérateur	Nom	Description
==	opérateur d'égalité	renvoie true si le côté gauche est égal au côté droit.
!=	opérateur de différence	renvoie true si le côté gauche n'est pas égal au côté droit.
>	opérateur de supériorité stricte	renvoie true si le côté gauche est supérieur au côté droit.
>=	opérateur de supériorité	renvoie true si le côté gauche est supérieur ou égal au côté droit.
<	opérateur d'infériorité stricte	renvoie true si le côté gauche est inférieur au côté droit.
<=	opérateur d'infériorité	renvoie true si le côté gauche est inférieur ou égal au côté droit.

# Les opérateurs logiques

Opérateur	Nom	Description
&&	ET logique	Renvoie True si les deux conditions sont vraies
	OU logique	Renvoie True si l'une des conditions est vraie
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur true si la variable vaut false, false si elle vaut true)

# Les opérateurs d'assignation

Opérateur	Description
=	affecte une valeur à la variable
+=	ajoute l'opérande gauche avec l'opérande droit, puis l'affecte à la variable de gauche.
-=	soustrait l'opérande gauche avec l'opérande droit, puis l'affecte à la variable de gauche.
*=	multiplier l'opérande gauche avec l'opérande droit, puis l'affecte à la variable de gauche.
/=	divise l'opérande gauche avec l'opérande droit, puis l'affecte à la variable de gauche.
%=	affecter le modulo de l'opérande gauche à l'opérande droit, puis l'affecte à la variable de gauche
++	utilisé pour incrémenter la valeur de 1
--	utilisé pour décrémenter la valeur de 1

# Les opérateurs de manipulation de bits

Opérateur	Description
&	renvoie 1 si les deux bits de même poids sont à 1.
	renvoie 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)
^	renvoie 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)
~	Il s'agit d'un opérateur unaire qui renvoie la représentation du complément à un de la valeur d'entrée,
<<	décale les bits du nombre vers la gauche et remplit 0 sur les vides laissés en conséquence
>>	décale les bits du nombre vers la droite et remplit 0 sur les vides à gauche en conséquence

# Les instructions conditionnelles

- If:

```
if (condition) {  
    //bloc d'instruction  
}
```

- If ... else ...:

```
if (condition) {  
    //Exécuter ce bloc si la condition est vraie  
} else {  
    //Exécuter ce bloc si la condition est fausse  
}
```

- Opérateur ternaire:

```
(condition) ? instruction si vrai : instruction si faux
```

# Les instructions conditionnelles

- switch

```
switch (input) {  
    case value1:  
        //liste d'instructions  
        break;  
    case value2:  
        //liste d'instructions  
        break;  
    .  
    .  
    .  
    default:  
        //liste d'instructions  
        break;  
}
```

Cette variable peut être de type byte, short, int, char ou énumération. A partir de JDK7, la variable peut également être de type String.

# Les instructions itératives

- La boucle while:

```
while (condition) {  
    //bloc d'instructions  
}
```

- La boucle do ... While:

```
do {  
    //bloc d'instructions  
}while(condition);
```

- La boucle for:

```
for (int j = 0; j < notes.length; j++)  
{  
    //bloc d'instructions  
}
```