

POO 3 (JAVA ET JAVA AVANCÉ)

Prof. Nisrine DAD

4° Ingénierie Informatique et Réseaux - Semestre I

Ecole Marocaine des Sciences d'Ingénieur

Année Universitaire : 2022/2023

PLAN

- I. Caractéristiques du langage Java
- II. Syntaxe et éléments de base du langage Java

PLAN

- I. **Caractéristiques du langage Java**
- II. Syntaxe et éléments de base du langage Java

I. CARACTÉRISTIQUES DU LANGAGE JAVA

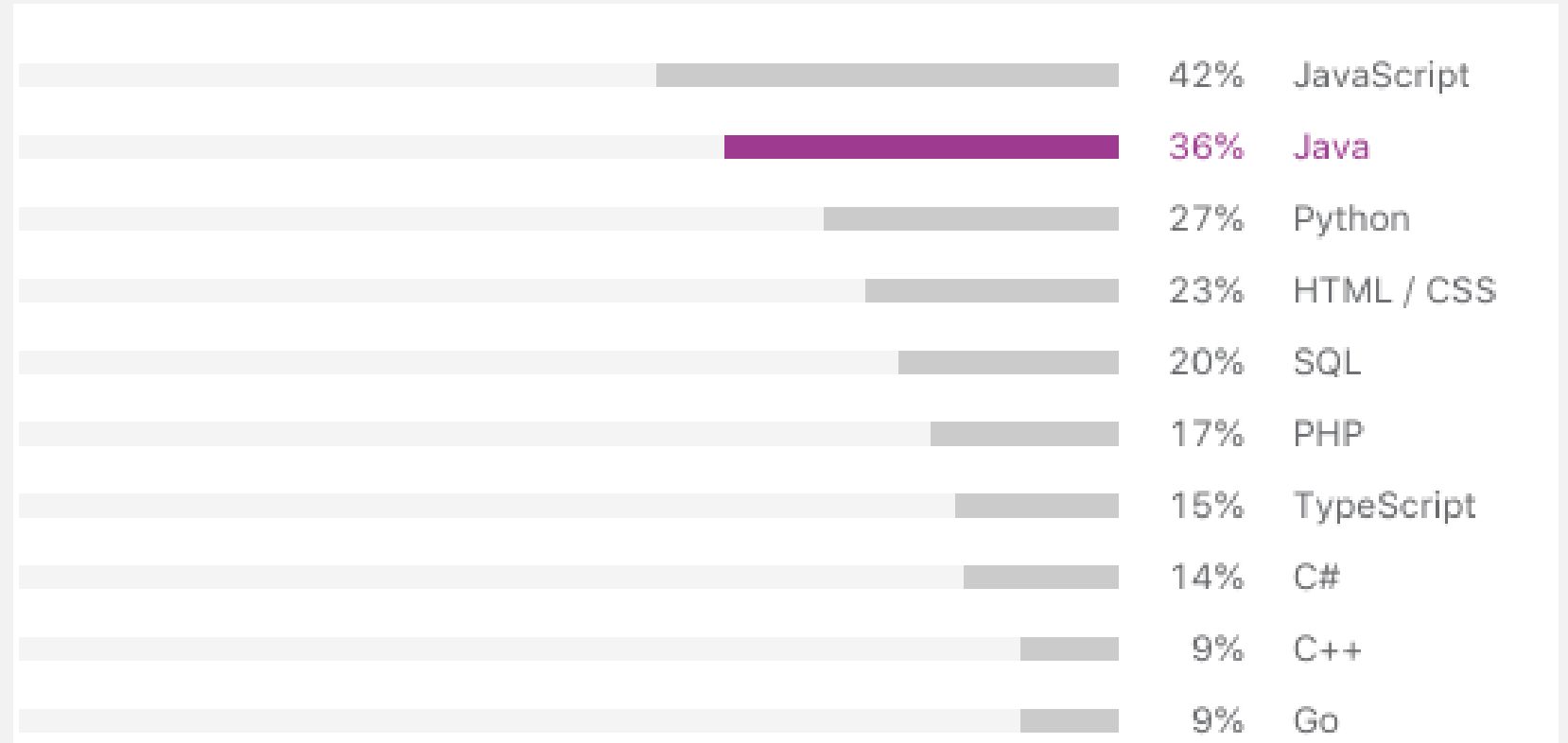
Introduction

- Java est un langage de programmation créé en 1995 par James Gosling chez **Sun Microsystems**.
- Le nom **Sun** vient de **Stanford University Network** (réseau de l'université Stanford).
- Lundi 20 avril 2009, l'éditeur américain de logiciels **Oracle** a racheté le groupe informatique Sun Microsystems.
- Donc, Java appartient aujourd'hui à l'entreprise Oracle.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Introduction

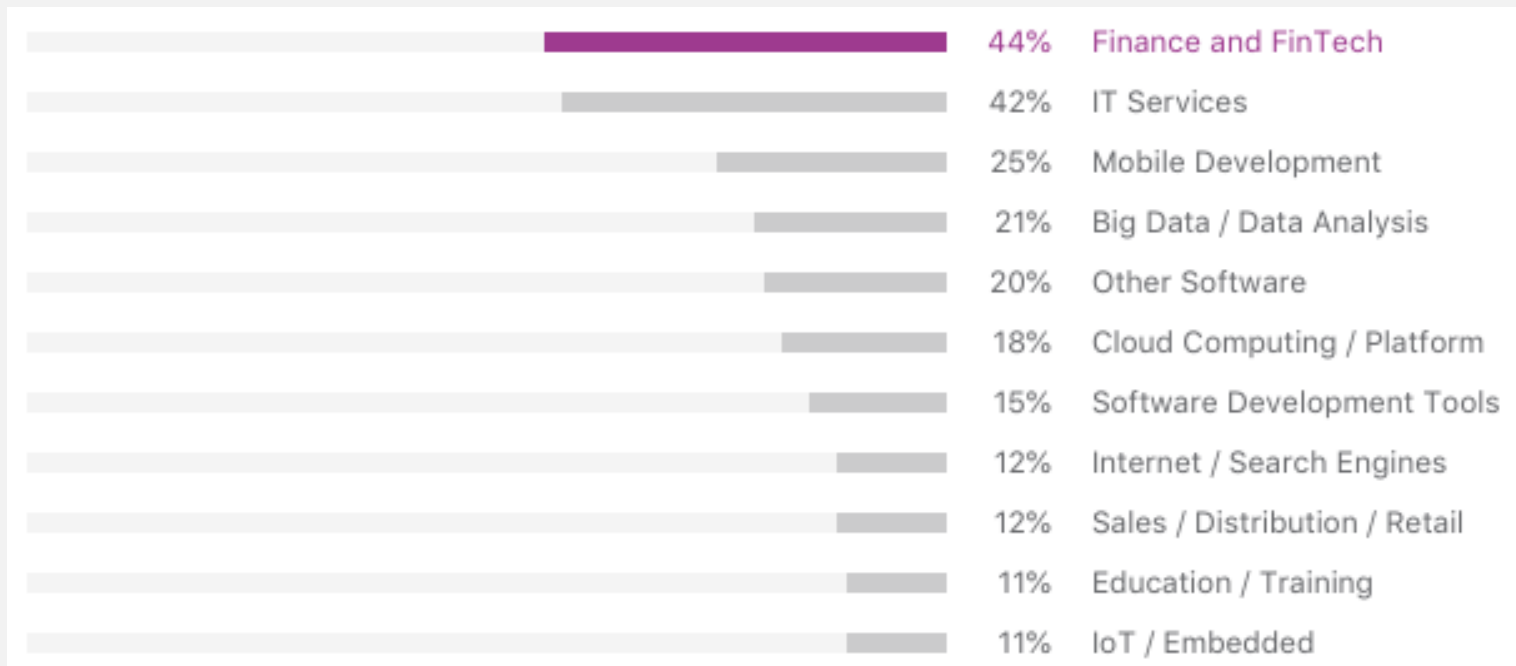
- Les pourcentages de l'utilisation de différents langage de programmation.
- Les données ont été tirées de la State of the Developer Ecosystem Survey 2020.



I. CARACTÉRISTIQUES DU LANGAGE JAVA

Introduction

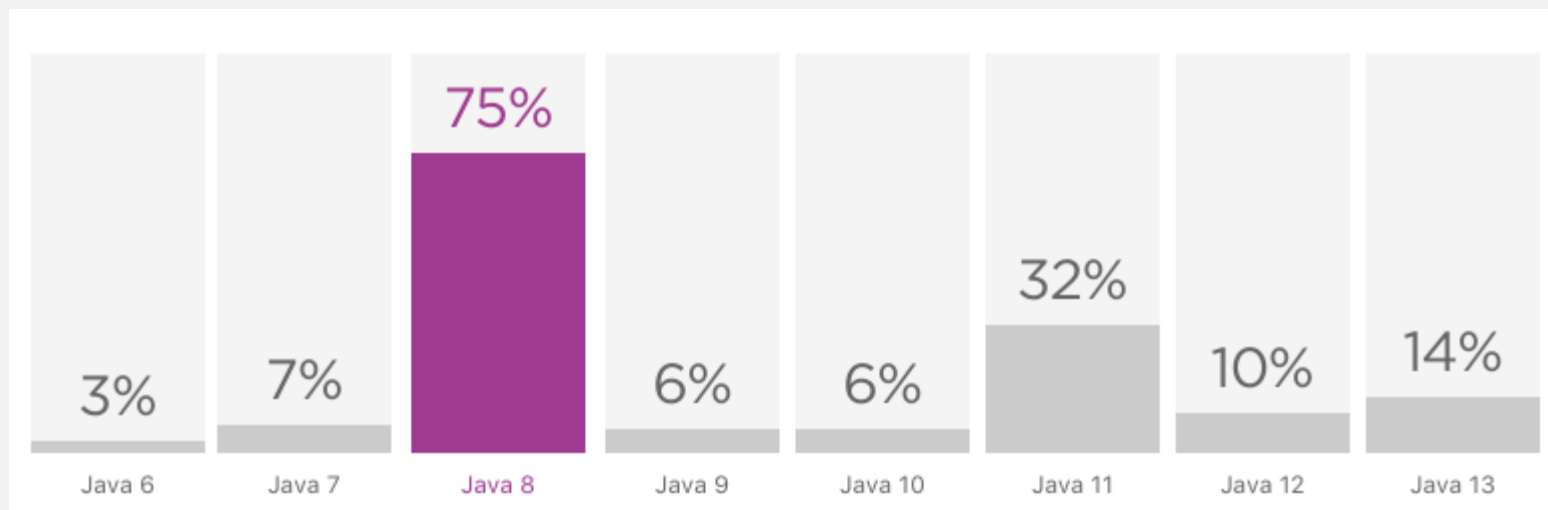
- Principaux secteurs où Java est utilisé selon le *Developer Ecosystem Survey 2020*.



I. CARACTÉRISTIQUES DU LANGAGE JAVA

Java et ses versions

- **Dernière version de Java:** Java SE 19 (20 Septembre 2022).
- Java 8 reste la version la plus populaire.



Popularité des versions Java selon Developer Ecosystem Survey 2020

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Java et ses versions

- Version Java installée sur mon ordinateur.
- Commande *java -version*

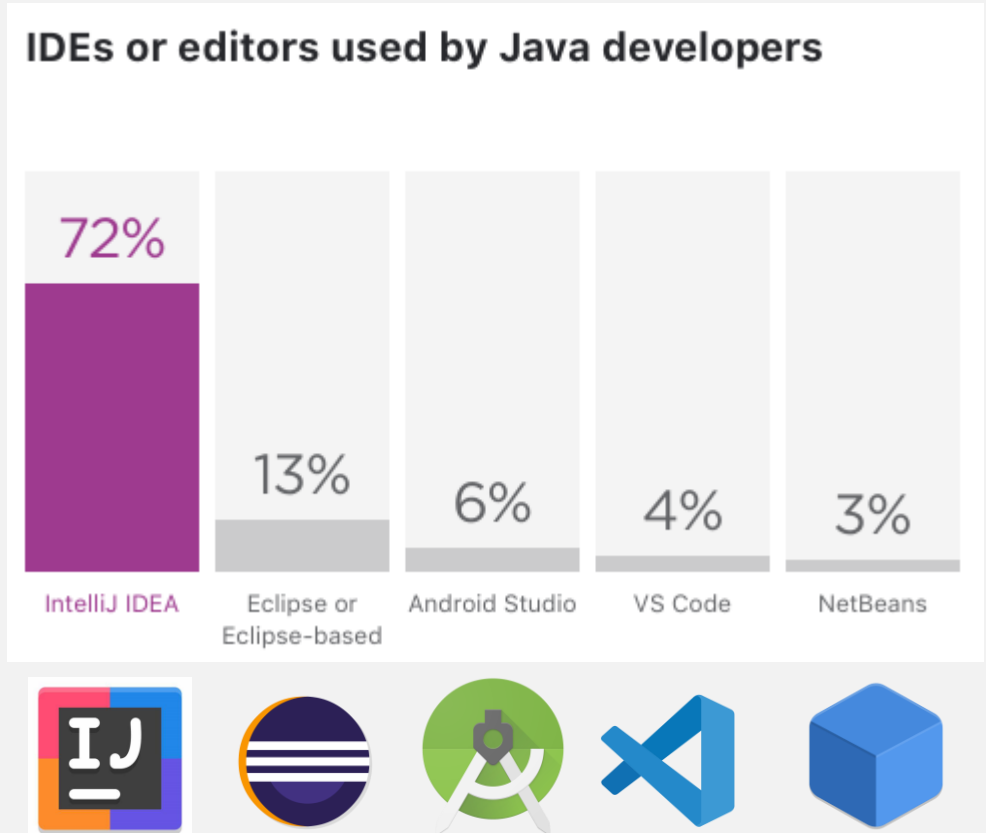
```
C:\Windows\system32>java -version
java version "17.0.4.1" 2022-08-18 LTS
Java(TM) SE Runtime Environment (build 17.0.4.1+1-LTS-2)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.4.1+1-LTS-2, mixed mode, sharing)
```

```
C:\Windows\system32>javac -version
javac 17.0.4.1
```


I. CARACTÉRISTIQUES DU LANGAGE JAVA

IDE Java

- l'IDE: *Integrated Development Environment*.
- Il regroupe un ensemble d'outils spécifiques dédiés aux programmeurs pour faciliter la mise en œuvre de projets tels que le développement de logiciels..
- IntelliJ IDEA occupe la première place, suivi de Eclipse.
- L'un des principaux produits de JetBrains State of Developer Ecosystem est IntelliJ IDEA.



I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage simple

- On peut l'apprendre facilement :
 - Peu de mots-clés
 - Facilité des fonctionnalités principales.
 - Se base sur les concepts fondamentaux des langages C et C++: En reprenant une grande partie de la syntaxe de ces deux langages, Java est vite pris en main par les développeur C et C++.

Langage orienté objet

- Java ne permet d'utiliser que des objets, sauf les types de base.
- Les grandes idées reprises sont : encapsulation, classe /instance, attribut, méthode / message, visibilité, héritage simple, interface/implémentation, surcharge des méthodes, polymorphisme.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

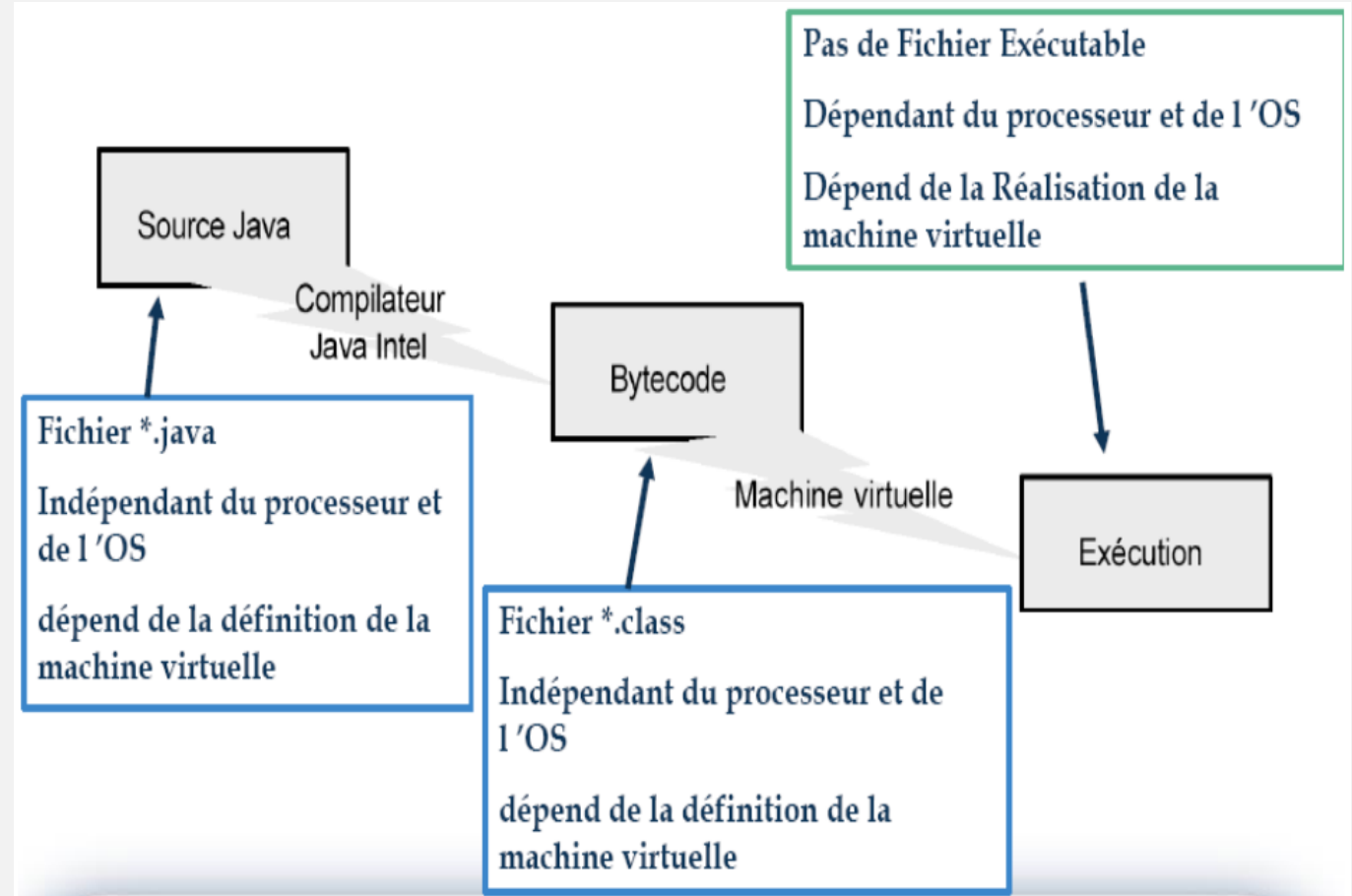
Langage fiable

- Sources d'erreurs limitées.
 - Pas de pointeurs,
 - Pas de structures,
 - Pas d'héritage multiple,
 - Pas de surcharge des opérateurs,
 - Vérifications faites par le compilateur facilitant une plus grande rigueur du code.
- Gestion automatique de la mémoire (ramasse-miette ou "*garbage collector*").
- Gestion des exceptions: Pour la détection et localisation des bugs.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage semi interprété

- Après compilation de la source, on obtient un byte-code.
- Il suffit de disposer d'une «Machine virtuelle» Java pour pouvoir exécuter tout programme Java (byte-code) même s'il a été compilé avec un autre système d'exploitation.



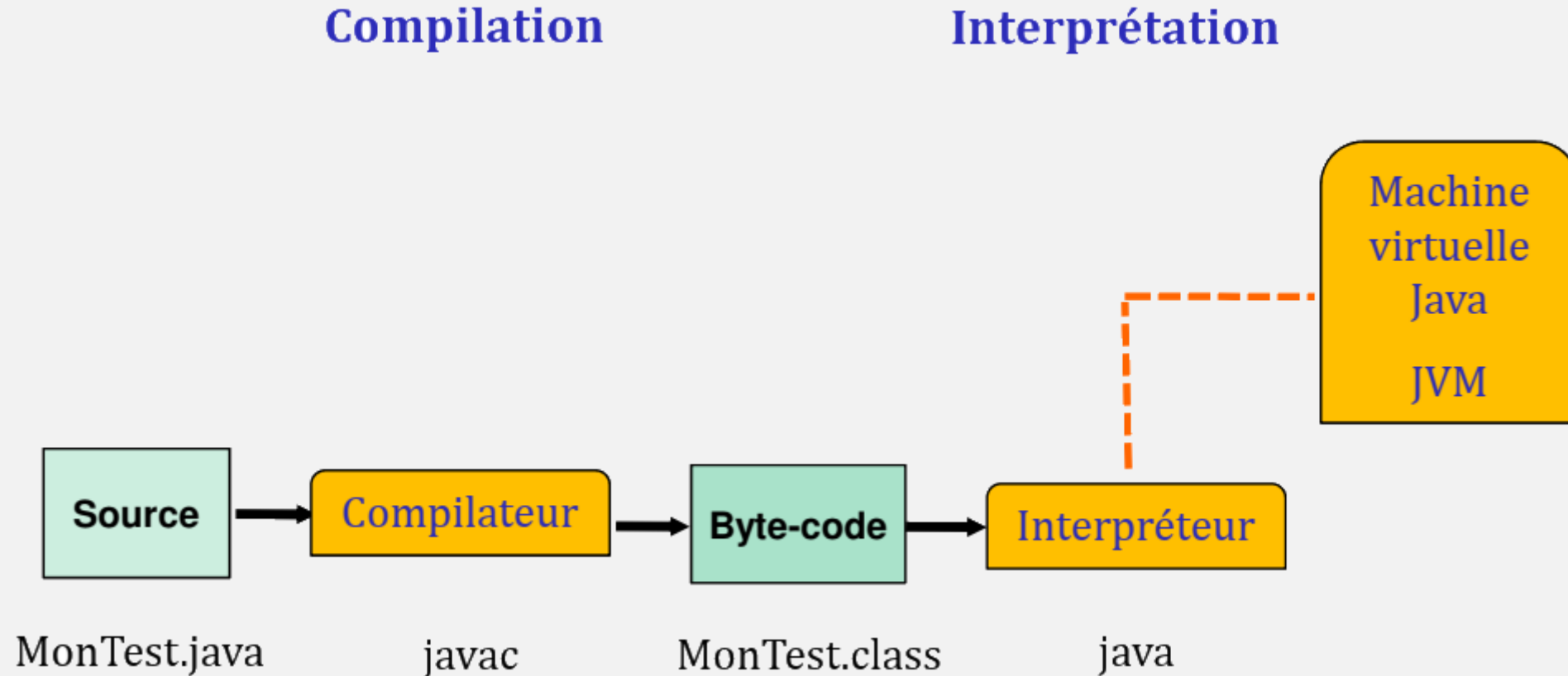
I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage semi interprété

- **Source Java**
 - Fichier utilisé lors de la phase de programmation.
 - Le seul fichier réellement intelligible par le programmeur.
- **Byte-code Java**
 - Code objet destiné à être exécuté sur toute « Machine virtuelle » Java
 - Provient de la compilation du code source.
- **Machines Virtuelles Java**
 - Des interpréteurs de byte-code indépendants (pour exécuter les programmes Java).
 - Contenues au sein d'un navigateur (pour exécuter des applets Java).

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage semi interprété



I. CARACTÉRISTIQUES DU LANGAGE JAVA

Étapes de développement

- **Création du code source**
 - A partir des spécifications (par exemple en UML) → **Outil** : éditeur de texte, IDE
- **Compilation en Byte-code**
 - A partir du code source → **Outil** : compilateur Java
- **Diffusion sur l'architecture cible**
 - Transfert du Byte-code seul → **Outils** : réseau, disque, ...
- **Exécution sur la machine cible**
 - Exécution du Byte-code → **Outil**: Machine Virtuelle Java

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage sûr

- Des fonctions de sécurité intégrées au langage et au système d'exécution:
 - La vérification du bytecode pour le code douteux,
 - la prise en charge de l'authentification et de la confidentialité, etc.

Indépendant de la plateforme

- Le byte-code généré par le compilateur est indépendant de toute plateforme.
- Toute application Java peut donc tourner sur une plate-forme implémentant une machine virtuelle Java.

« *Write Once Run Anywhere* »

« Ecrire une fois, exécuter partout »

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Langage multi-tâches

- Exécution de plusieurs processus effectuant chacun une tâche différente simultanément.
- Mécanismes de synchronisation.
- Fonctionnement sur des machines multiprocesseurs.

Langage multi-thread

- La gestion de plusieurs tâches prévues.
- Un thread est une portion de code capable de s'exécuter en parallèle à d'autres traitements.
- Exécution de plusieurs threads d'un processus simultanément

I. CARACTÉRISTIQUES DU LANGAGE JAVA

JDK vs JRE vs JVM

- JVM (Java Virtual Machine),
- JDK (Java Development Kit) et
- JRE (Java Runtime Environment)
- Ce sont trois composants indispensables de la plate-forme Java.
- Ils fonctionnent ensemble dans les applications Java.
- Il est possible de télécharger le JRE indépendamment du JDK.

JDK

`javac, jar, debugging tools,
javap`

JRE

`java, javaw, libraries,
rt.jar`

JVM

Just In Time
Compiler (JIT)

I. CARACTÉRISTIQUES DU LANGAGE JAVA

JDK vs JRE vs JVM

- La JVM est le composant de la plate-forme Java qui exécute les programmes.
- Le JRE crée la JVM et s'assure que les dépendances sont disponibles pour les programmes Java.
- Le JDK permet de créer des programmes Java qui peuvent être exécutés par la JVM et le JRE.
- JDK contient en plus du JRE et la JVM:
 - Les classes de base de l'API java.
 - La documentation au format HTML
 - Le compilateur : javac
 - Le générateur de documentation : javadoc, etc.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

API Java

- Java fournit de nombreuses bibliothèques de classes remplissant des fonctionnalités très diverses : c'est l'API Java.
- Une API (*Application Programming Interface*) est un ensemble normalisé de **classes**, de **méthodes**, de **fonctions** et de **constantes** qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.
- Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur.
- Ces classes sont regroupées, par catégories, en paquetages (ou "*packages*").
- Un *package* est un ensemble de classes. En fait, c'est un ensemble de dossiers et de sous-dossiers contenant une ou plusieurs classes.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

API Java: Exemples

- Les principaux packages:
 - **java.util** : structures de données classiques (vecteurs, hashtable).
 - **java.io** : entrées / sorties (flux, fichiers).
 - **java.lang** : chaînes de caractères, interaction avec l'OS, threads.
 - **java.math** : gestion de grands nombres.
 - **javax.swing** : Pour la création d'interfaces graphiques.
 - **java.sql** : fournit le package JDBC

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Documentation Java

- La documentation de Java est standard, que ce soit pour les classes de l'API ou pour les classes utilisateur → Possibilité de génération automatique avec l'outil **Javadoc**.
- Elle est au format HTML → Intérêt de l'hypertexte pour naviguer dans la documentation .
- Pour chaque classe, il y a une page HTML contenant :
 - L'hierarchie d'**héritage** de la classe.
 - Une **description** de la classe et son but général.
 - La liste des **attributs** de la classe (locaux et hérités).
 - La liste des **constructeurs** de la classe (locaux et hérités).
 - La liste des **méthodes** de la classe (locaux et hérités).
 - Puis, chacune de ces trois dernières listes, avec la description détaillée de chaque élément.

I. CARACTÉRISTIQUES DU LANGAGE JAVA

Documentation Java: Exemple

- Documentation de la classe String:
- <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>
- Javadoc:

```
/** Classe Hello Etudiant  
 *  
 * @author Nisrine DAD  
 *  
 */
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Premier exemple de programme Java

Fichier PremierTest.java

```
public class PremierTest
//La classe est l'unité de base de nos programmes.
{ //Accolade débutant la classe PremTest
    public static void main(String[] args)
    { //Accolade débutant la méthode main
        /* Pour l'instant juste une instruction. Ce commentaire s'étale sur
        plusieurs lignes */
        System.out.println("bonjour"); // Une instruction se termine par une ;
    } //Accolade fermant la méthode main
} //Accolade fermant la classe PremTest
```


II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Exemple de programme Java

```
public class HelloEtudiant {  
    public static void main(String[] args) {  
        sayHelloTo("Amine Amini"); // sayHelloTo(args[0]) si on veut  
        passer le nom en paramètre lors de l'exécution  
    }  
    // affiche le message "hello" à l'étudiant fourni  
    private static void sayHelloTo(String nomEtudiant) {  
        System.out.println("Hello " + nomEtudiant);  
    }  
}
```

Cas de l'utilisation de args[o]

```
d:\EMSI>javac helloEtudiant.java  
  
d:\EMSI>java HelloEtudiant  
Hello Amine Amini
```

```
D:\EMSI\JavaFiles>javac HelloEtudiant.java  
  
D:\EMSI\JavaFiles>java HelloEtudiant "Amini Amine"  
Hello Amini Amine
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Contenu du byte-code

- Byte-code (HelloEtudiant.class) (ou code binaire – langage machine).

```

HelloEtudiant.class
1  Êþº%NULNULNUL=NUL:
2  NULSTXNULETXBETNULEOTFFNULENQNULACKSOHNULDLEjava/lang/ObjectSOHNULACK<init>SOHNULETX()V
3  NULBSNUL    BETNUL
4  FFNULVTNULFSOHNUL
5  HelloEtudiantSOHNUL
6  sayHelloToSOHNULNAK(Ljava/lang/String;)V
7  NULSONULSTBETNULDLEFFNULDC1NULDC2SOHNULSTjava/lang/FloatSOHNUL
8  parseFloatSOHNULNAK(Ljava/lang/String;)F    NULDC4NULNARBETNULSYNFFNULETBNULCANSOHNULDLE
9  java/lang/SystemSOHNULETXoutSOHNULNAKLjava/io/PrintStream;
10 NULSUBNULESCBETNULFSFFNULGSNULRSOHNULDC3java/io/PrintStreamSOHNULBELprintlnSOHNULEOT
(F)VDC2NULNULNUL    FFNUL!NUL"SOHNULETBmakeConcatWithConstantsSOHNUL
&(Ljava/lang/String;)Ljava/lang/String;
11 NULSUBNUL$FFNULGSNULFFSOHNULEOTCodeSOHNULSTLineNumberTableSOHNULEOTmainSOHNULSYN
([Ljava/lang/String;)VSOHNUL
12 SourceFileSOHNULDC2HelloEtudiant.javaSOHNULDLEBootstrapMethodsSTACKNUL-
NUL.NUL/BETNUL0FFNUL!NUL1SOHNUL$java/lang/invoke/StringConcatFactorySOHNUL
{Ljava/lang/invoke/MethodHandles$Lookup;Ljava/lang/String;Ljva/lang/invoke/MethodType;Ljava/lang/
String;[Ljava/lang/Object;)Ljava/lang/invoke/CallSite;BSNUL3SOHNULBELHello SOHSOHNULFF
InnerClassesBELNUL6SOHNUL$java/lang/invoke/MethodHandles$LookupBETNUL8SOHNULRS
java/lang/invoke/MethodHandlesSOHNULACKLookupNUL
!NULBSNULSTXNULNULNULNULETXNULSOHNULENQNULACKNULSOHNUL
%NULNULNULGSNULSOHNULSOHNULNULNULENQ*·NUL/SOH+NULNULNULSOHNUL
&NULNULNULACKNULSOHNULNULNULSTXNULNUL'NUL(NULSOHNUL%NULNULNUL@NULETXNULSTXNULNULNULFS*ETX
2,NULBET*EOT2,NUL
*NENQ2,NUL
13 bD°NULDC3#¶NULEM+NULNULNULSOHNUL
&NULNULNULDC2NULEOTNULNULNULENQNULACKNULACKNULDC4NULBELNULESCNULBSNUL
NULVTNULFFNULSOHNUL%NULNULNUL)NULSTXNULSOHNULNULNUL
²NULDC3*°NULUSNULNUL¶NUL#+NULNULNULSOHNUL&NULNULNUL
NULSTXNULNULNULVTNULEFFNULFFNULETXNUL)NULNULNULSTXNUL*NUL+NULNULNULBSNULSOHNUL,NULSOHNUL
2NUL4NULNULNUL
18 NULSOHNUL5NUL7NUL9NULEM

```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Fonction main

- Le nom du fichier doit être le même que celui de la classe avec l'extension .java;
- Java est sensible à la casse des lettres (PremierTest # premierTest).
- Pour pouvoir faire un programme exécutable, il faut toujours une classe contenant une méthode particulière, la méthode « main ».
- C'est le point d'entrée dans le programme : le microprocesseur sait qu'il va commencer à exécuter les instructions à partir de cet endroit.
- Les principes du code propre exigent qu'aucune logique ne soit écrite à l'intérieur de la méthode `main` . Tout le travail doit être délégué à des fonctions bien nommées.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Fonction main

- **public** : obligatoire pour que votre programme puisse s'exécuter.
- **Static**: précise que la méthode main de la classe *PremierTest* n'est pas liée à une instance (objet) particulière de la classe.
- **String[] args** : permet de récupérer des arguments transmis au programme au moment de son lancement. On peut lancer un programme sans fournir d'arguments, mais l'indication `String args[]` est obligatoire.
- **System.out.println ("Bonjour")** ; cette ligne sert à définir le contenu de notre programme qui va afficher le message Bonjour suivi d'un retour à la ligne suivante dans la fenêtre de console.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Instructions, blocs, blancs et commentaires

- Les instructions Java se terminent par un ;
- Les blocs sont délimités par { pour le début de bloc } pour la fin du bloc.
- Un bloc permet de définir un regroupement d'instructions.
- La définition d'une classe ou d'une méthode se fait dans un bloc.
- Les espaces, tabulations, sauts de ligne sont autorisés. Cela permet de présenter un code plus lisible.
- La séquence // permet d'insérer un commentaire sur une seule ligne.
- La séquence /* permet d'insérer un commentaire sur plusieurs lignes. La séquence */ marque la fin du commentaire.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

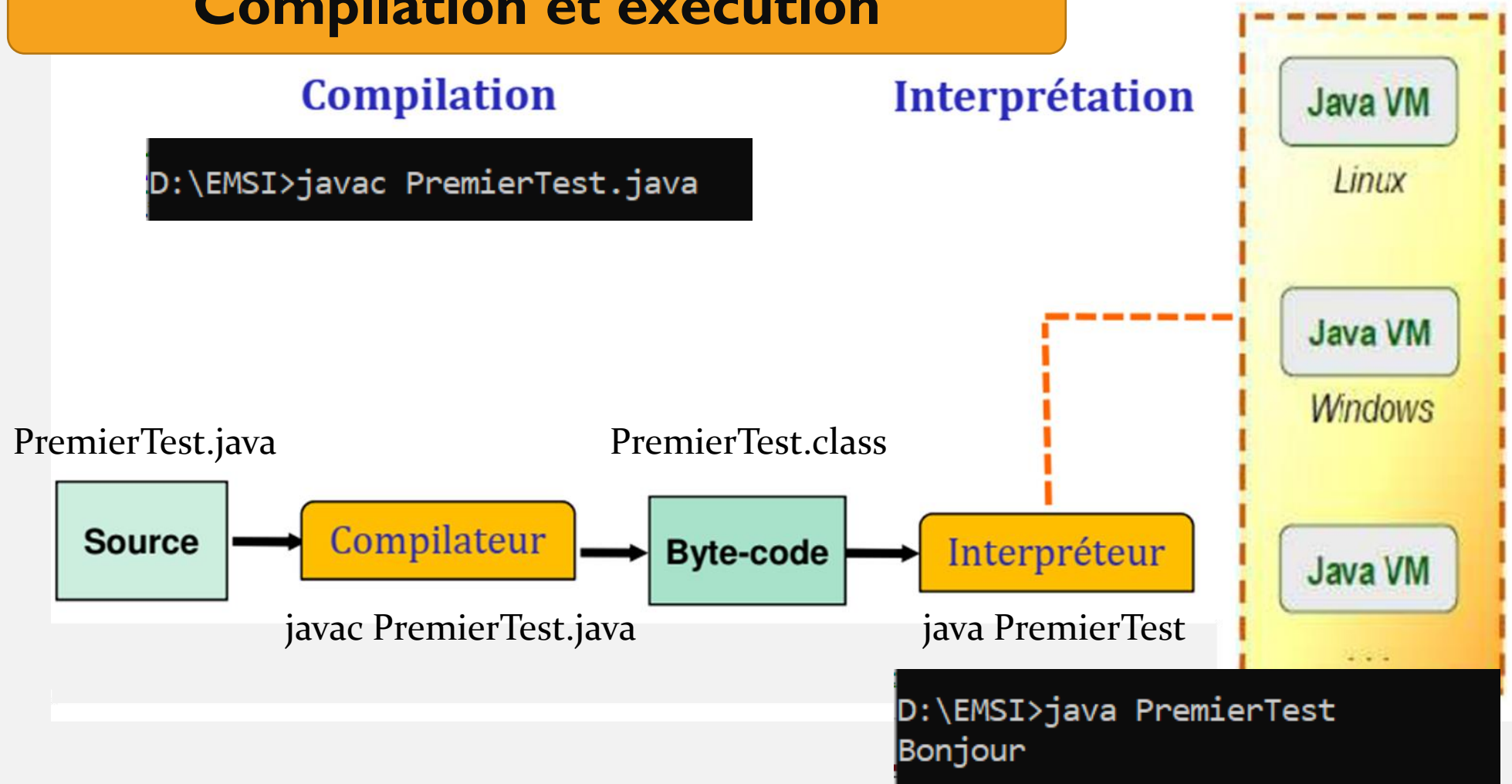
Compilation et exécution

- **Etape 1:** Compilation en bytecode java dans une console DOS:
javac PremierTest.java → Génère un fichier PremierTest.class
- **Etape 2:** Exécution du programme (toujours depuis la console DOS) sur la JVM :
java PremierTest → Affichage de “Bonjour” dans la console

```
D:\EMSI>javac PremierTest.java  
  
D:\EMSI>java PremierTest  
Bonjour
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Compilation et exécution



II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Compilation et exécution

- **Résumé:**
- `javac PremTest.java` :
 - Compilation en Byte-code java.
 - Indication des erreurs de syntaxe éventuelles.
 - Génération d'un fichier `PremierTest.class` si pas d'erreurs.
- `java PremierTest`
 - java est la machine virtuelle.
 - Exécution du Byte-code.
 - Nécessité de la méthode `main`, qui est le point d'entrée dans le programme.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Exécution dans un IDE

- Le traitement et l'exécution des différents programmes Java seront effectués par le fameux IDE: Eclipse.
- Donc, on va utiliser désormais la dernière version d'Eclipse (Eclipse IDE for Java Developers), qu'on peut la télécharger de la page:

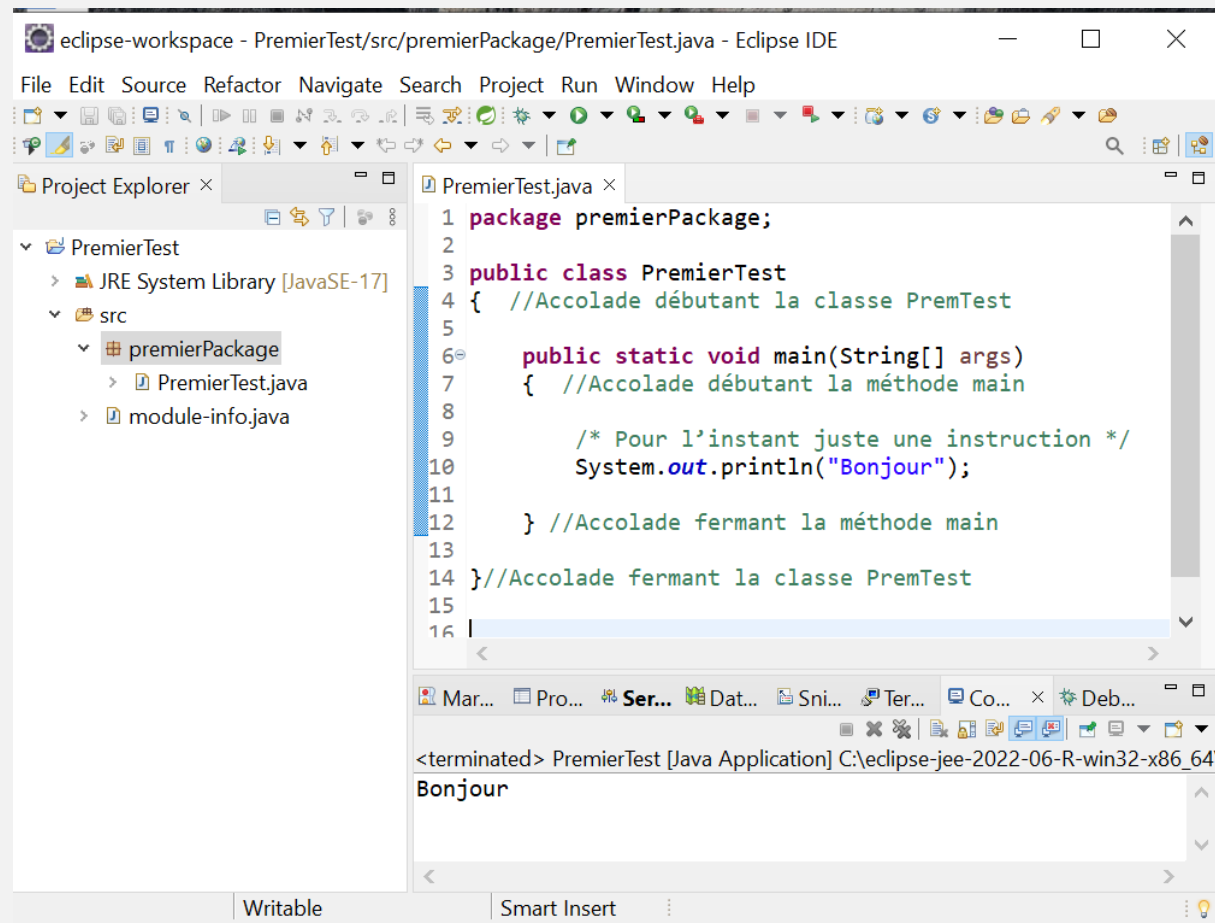
<http://www.eclipse.org/downloads>

- On peut télécharger la version de la JDK (8) sur le site suivant:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Exécution dans un IDE



```
eclipse-workspace - PremierTest/src/premierPackage/PremierTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
PremierTest
  JRE System Library [JavaSE-17]
  src
    premierPackage
      PremierTest.java
      module-info.java
PremierTest.java
1 package premierPackage;
2
3 public class PremierTest
4 { //Accolade débutant la classe PremTest
5
6     public static void main(String[] args)
7     { //Accolade débutant la méthode main
8
9         /* Pour l'instant juste une instruction */
10        System.out.println("Bonjour");
11
12    } //Accolade fermant la méthode main
13
14 } //Accolade fermant la classe PremTest
15
16
<terminated> PremierTest [Java Application] C:\eclipse-jee-2022-06-R-win32-x86_64
Bonjour
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions

- Une **fonction** peut être considérée comme un bloc de code avec un nom, qui exécute un service.
- Lorsqu'une fonction est située à l'intérieur d'une classe, elle s'appelle une méthode.
- Tout le code Java doit être situé à l'intérieur de classes.
- Les classes sont contenues dans des packages.
- Lorsqu'on lance le programme, c'est la fonction **main()** qui se lance.
- La fonction **main()** est le morceau de code que l'interpréteur Java recherche lorsque vous démarrez un programme.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions

- Si on écrit toute la logique de notre programme à l'intérieur du `main`, cela pourrait donner une trop grande quantité de code à un seul endroit.
- Ce serait difficilement compréhensible pour nous, les humains, et compliquerait la maintenance de votre programme.
- C'est la raison pour laquelle on doit organiser notre code en classes.
- Dans les bonnes pratiques, la méthode `main` doit être la plus courte possible, en appelant uniquement les méthodes nécessaires.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions prédéfinies

- Les classes prédéfinies en java offrent plusieurs méthodes.
- Il faut utiliser le site web Javadoc pour trouver toutes les méthodes d'une classe.
- Cela permet de gagner du temps par la suite.
- Par exemple, **String** est une classe qui définit un état et un comportement :
 - Son état est la chaîne de caractères que vous stockez. La valeur réelle est définie pour chaque objet lorsque vous l'instanciez.
 - Son comportement est l'ensemble des méthodes que la classe `String` définit, et qui vous permettent d'opérer sur la chaîne que vous stockez.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions prédéfinies

- Exemple:

```
String exemple="hello";  
exemple=exemple.toUpperCase();
```

String	replace (char oldChar, char newChar)	Returns a string resulting from replacing all occurrences of oldChar in this string with newChar.
String	toUpperCase ()	Converts all of the characters in this String to upper case using the rules of the default locale.
boolean	contains (CharSequence s)	Returns true if and only if this string contains the specified sequence of char values.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions

- Les fonctions sont considérées comme des modules qui permettent d'éviter de tout rassembler dans un seul et même bloc de code, afin **d'améliorer la lisibilité** de son programme.
- Cela permet aussi de diviser le programme en sous parties, qui pourront être ensuite **réutilisées** dans d'autres programmes.

```
public static <type_de_la_variable_de_retour> <nom_de_la_fonction>(<paramètre1>,  
<paramètre2>, ...) {  
    // Instructions ici  
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions

Exemple:

```
public class Test {  
    public static int MultiplierParDeux(int entree) {  
        int sortie; sortie = entree * 2;  
        return sortie; }  
  
    public static void main(String args[]) {  
        int x = 3;  
        int y = MultiplierParDeux(x);  
        System.out.println(y); }  
}
```


II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions

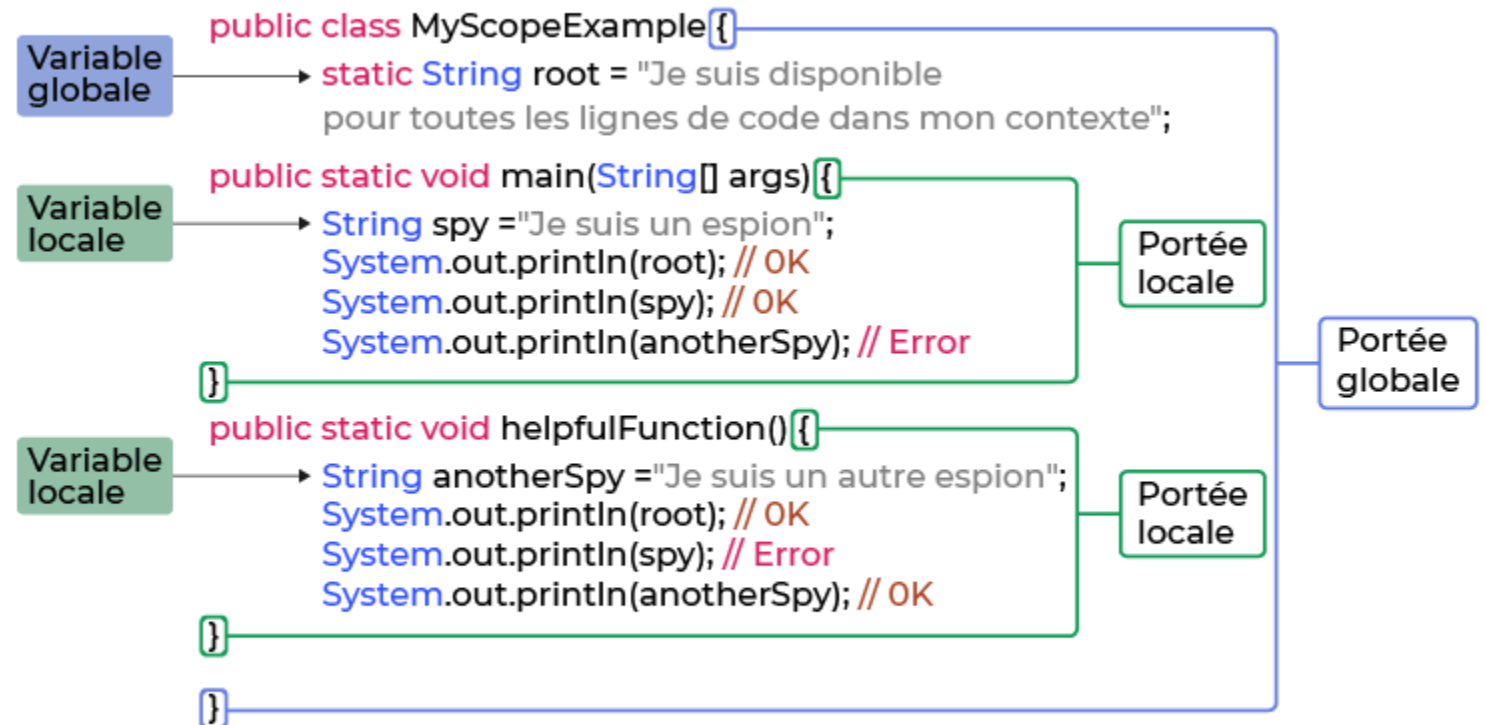
- Les principes du code propre exigent qu'aucune logique ne soit écrite à l'intérieur de la méthode `main` .
- Tout le travail doit être délégué à des fonctions bien nommées.

```
public class HelloEtudiant {  
    public static void main(String[] args) {  
        sayHelloTo("Amine Amini");  
        // affiche le message "hello" à l'étudiant fourni  
    private static void sayHelloTo(String nomEtudiant) {  
        System.out.println("Hello " + nomEtudiant);  
    }  
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Portée des variables

- La portée (scope) d'une variable est la zone de code où elle a été déclarée.
- La portée variable générale s'applique à tous les blocs de code, y compris les classes.



Deux portées locales au sein d'une portée globale

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Portée des variables

- Un autre moyen nécessaire pour contrôler l'accès aux variables et aux fonctions est d'utiliser des niveaux de contrôle : `public`, `protected`, `package-protected` et `private`.
- **public** : visible pour tous et par conséquent le moins restrictif ;
- **protected (protégé)** : visible pour le package et l'ensemble de ses sous-classes ;
- **package-protected (protégé par paquet)** : généralement visible uniquement par le package dans lequel il se trouve (paramètres par défaut). Ne pas mettre de mot clé déclenche ce niveau de contrôle ;
- **private (privé)** : accessible uniquement dans le contexte dans lequel les variables sont définies (à l'intérieur de la classe dans laquelle il est situé).

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Portée des variables: Exercice

```
public class Rectangle {  
    public int width=4;  
    private int borderWidth=1;
```

```
    public static void main(String[] args) {  
        int mainWidth=28;  
    }
```

```
    public void printValues() {  
        final int newWidth=12;  
        if (true) {  
            int pixelSize=5;  
        }
```

```
        //Quelle variable pourrait être utilisée ici ?  
        System.out.println(?);  
    }
```

```
}
```

- La variable **newWidth** est **accessible** car elle se trouve dans la même fonction.
- Les variables **width** et **borderWidth** sont aussi **accessibles** car elles se trouvent dans la classe Rectangle.
- La variable **pixelSize** n'est pas **accessible** car elle déclarée dans le bloc if.

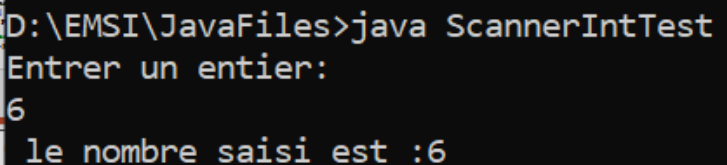
II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Lecture d'informations au clavier

- Pour la saisie de données au clavier, on va utiliser la classe **Scanner** (`java.util.Scanner`) qui regroupe plusieurs méthodes pratiques :
- `nextInt()`, `nextDouble()`, `nextFloat()`, `nextLong()`, `nextLine()`, ...

```
import java.util.Scanner;
public class ScannerIntTest{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);//System.in: Entrée Standard

        System.out.println("Entrer un entier:");
        int a=sc.nextInt();
        System.out.println(" le nombre saisi est : " + a);
    }
}
```



```
D:\EMSI\JavaFiles>java ScannerIntTest
Entrer un entier:
6
le nombre saisi est :6
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Lecture d'informations au clavier

- **Remarque:** si vous avez appelé `nextInt()`, `nextDouble()` et que vous appelez directement après `nextLine()`, celle-ci ne vous invite pas à saisir une chaîne de caractères : elle vide la ligne commencée par les autres instructions.
- En effet, celles-ci ne repositionnent pas la tête de lecture, l'instruction `nextLine()` le fait à leur place.

```
import java.util.Scanner;
public class ScannerLineTest {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Saisissez un entier : ");
        int i = sc.nextInt();
        System.out.println("Saisissez une line : ");
        String str = sc.nextLine();
        System.out.println("FIN ! "); } }
```

```
D:\EMSI\JavaFiles>java ScannerLineTest
Saisissez un entier :
5
Saisissez une line :
FIN !
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Lecture d'informations au clavier

- Le programme précédent ne vous demande pas de saisir une chaîne et affiche directement « Fin ». Pour pallier ce problème, il suffit de vider la ligne après les instructions ne le faisant pas automatiquement :

```
import java.util.Scanner;
public class ScannerLineTest {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Saisissez un entier : ");
        int i = sc.nextInt();
        System.out.println("Saisissez une ligne : ");
        //On vide la ligne avant d'en lire une autre
        sc.nextLine();
        String str = sc.nextLine();
        System.out.println("FIN ! "); } }
```

```
D:\EMSI\JavaFiles>java ScannerLineTest
Saisissez un entier :
6
Saisissez une ligne :
Nouvelle Ligne
FIN !
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Lecture d'informations au clavier

Résumé:

- L'affichage des données se fait par *System.out.println ()*.
- La lecture des entrées clavier se fait via l'objet *Scanner*.
- Scanner se trouve dans le package *java.util* qu'il faut importer.
- Pour pouvoir récupérer ce que vous allez taper dans la console, vous devrez initialiser l'objet Scanner avec l'entrée standard, *System.in*.
- Il y a une méthode de récupération de données pour chaque type (sauf les char) : *nextLine()* pour les String, *nextInt()* pour les int ...

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Identificateurs

- **Attention** : Java distingue les majuscules des minuscules .
- **Conventions sur les identificateurs** :
- Si plusieurs mots sont accolés, utilisez le **Camel Case**.
- D'autre part suivez ces directives:
 - La première lettre est majuscule pour les classes et les interfaces. exemples : **MaClasse**, **MonInterface**.
 - La première lettre est minuscule pour les méthodes, les attributs et les variables. exemples : **setLongueur**, **i**, **unAttribut**.
 - Les constantes sont entièrement en majuscules. exemple : **LONGUEUR_MAX**

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les mots réservés de Java

- Les mots réservés du langage Java ne peuvent pas être utilisés comme identifiant.

<code>abstract</code>	<code>default</code>	<code>goto</code>	<code>null</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>package</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>private</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>protected</code>	<code>throws</code>
<code>case</code>	<code>extends</code>	<code>instanceof</code>	<code>public</code>	<code>transient</code>
<code>catch</code>	<code>false</code>	<code>int</code>	<code>return</code>	<code>true</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>short</code>	<code>try</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>static</code>	<code>void</code>
<code>continue</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>volatile</code>
<code>const</code>	<code>for</code>	<code>new</code>	<code>switch</code>	<code>while</code>

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Types de base

- En Java, tout est objet sauf les types de base.
- Il y a **huit** types de base :
 - Un type booléen pour représenter les variables ne pouvant prendre que 2 valeurs (vrai ou faux : 0 ou 1, etc.) : **boolean** avec les valeurs associées true et false.
 - Un type pour représenter les caractères : **char**
 - **Quatre** types pour représenter les entiers de diverses tailles : **byte** (8bits dont 1 bit de signe), **short** (16bits), **int** (32 bits) et **long** (64bits).
 - **Deux** types pour représenter les réelles : **float** (32 bits) et **double** (64 bits)

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Types de base

- Les opérateurs d'incrémentation ++ et de décrémentation -- ajoute ou retranche 1 à une variable. **Exemple:** `int n = 12; n ++; //Maintenant n vaut 13`
- `n++;` « équivalent à » `n = n+1;` `n--;` « équivalent à » `n = n-1;`
- Peut s'utiliser de manière suffixée : `++n`. La différence avec la version préfixée se voit quand on les utilisent dans les expressions.
- En version suffixée l'incrémentation ou la décrémentation s'effectue en premier.
- **Exemples:**

```
int m=7; int n=7;  
int a=2 * ++m; //a vaut 16, m vaut 8  
int b=2 * n++; //b vaut 14, n vaut 8
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les opérateurs

	Symbole	Description	Exemple
Opérateurs arithmétiques	-	<i>soustraction</i>	<i>x-y</i>
	*	<i>multiplication</i>	<i>3 *x</i>
	/	<i>division</i>	<i>4/2</i>
	%	<i>modulo (reste de la division)</i>	<i>5%2</i>
Opérateurs d'affectation	=	<i>Affectation</i>	<i>x=2</i>
	-=	<i>Soustraction et affectation</i>	<i>x-=2</i>
	+=	<i>Addition et affectation</i>	<i>x+=2</i>

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les opérateurs

	Symbole	Description	Exemple
Opérateurs d'incrémentations et décrémentations	<i>++</i>	<i>Pré-incrémentation</i>	<i>++x</i>
	<i>++</i>	<i>Post-incrémentation</i>	<i>x++</i>
	<i>--</i>	<i>Pré-décrémentation</i>	<i>--x</i>
	<i>--</i>	<i>Post-décrémentation</i>	<i>x--</i>
Opérateurs relationnels	<i>==</i>	<i>égal à</i>	<i>x==2</i>
	<i><</i>	<i>inférieur à</i>	<i>x<2</i>
	<i><=</i>	<i>inférieur ou égal à</i>	<i>x<=3</i>
	<i>></i>	<i>supérieure à</i>	<i>x>2</i>
	<i>>=</i>	<i>supérieur ou égal à</i>	<i>x>=3</i>
	<i>!=</i>	<i>différent de</i>	<i>a !=b</i>

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les opérateurs

	Symbole	Description	Exemple
Opérateurs logiques	<code>&&</code>	<i>et</i>	<code>a && b</code>
	<code>//</code>	<i>ou</i>	<code>a // b</code>
	<code>!</code>	<i>non</i>	<code>!a</code>
Opérateurs relationnels	<code>&</code>	<i>et</i>	<code>a & b</code>
	<code> </code>	<i>ou</i>	<code>a b</code>
	<code>^</code>	<i>ou exclusif</i>	<code>a ^ b</code>
	<code>~</code>	<i>non</i>	<code>~x</code>
	<code><<</code>	<i>décalage à gauche</i>	<code>a << 3</code>
	<code>>></code>	<i>décalage à droite</i>	<code>b >> 2</code>

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

- ***for***: Exécute un ensemble d'instructions un nombre de fois fixe, basé sur les valeurs limites inférieure et supérieure de l'énumérateur.

```
for (initialisation; condition ; incrémentation ou décrémentation) {  
    lesInstruction;  
}
```

- ***break***: Permet de terminer l'exécution d'une boucle (interrompre une séquence d'exécution).
- ***continue***: Permet l'interruption d'une itération en cours et retourne au début de la boucle avec exécution de la partie incrémentation (Ignorez quelques instructions à l'intérieur d'une boucle)

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

```
for ( int i=0; i <10; i++) { // déclarations exécutées à chaque itération
    if(i == 2 || i == 5) {        continue; }
    System.out.println("Valeur de i : " + i + ".");
}
```

```
int [] myArray = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
for (int i =0; i<myArray.length;i++) {
    if (myArray[i] == 50) {
        System.out.println ("J'ai trouvé ce que je cherche: "
            +myArray[i]+ " !");
        break; }
    System.out.println ("Je cherche encore. Je suis arrivé à "+myArray[i]);
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

- **while** : Exécute un ensemble d'instructions jusqu'à ce qu'une condition définie soit remplie.

```
while (expression booléenne) {  
    instructions;  
}
```

```
int nombreArbres = 0;  
while (nombreArbres < 10) {  
    nombreArbres += 1;  
    System.out.println("J'ai" + nombreArbres +  
        " arbres");  
}
```

```
J'ai 1 arbres  
J'ai 2 arbres  
J'ai 3 arbres  
J'ai 4 arbres  
J'ai 5 arbres  
J'ai 6 arbres  
J'ai 7 arbres  
J'ai 8 arbres  
J'ai 9 arbres  
J'ai 10 arbres
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

- **do, while:** Elle est très similaire à **while**, mais la condition est placée à la fin du bloc de code correspondant. De cette façon, le bloc de code sera toujours exécuté au moins une fois.

```
do {  
    instructions;  
} while (condition);
```

```
int pushUpGoal = 10;  
do{  
    System.out.println ("Push up " + pushUpGoal);  
    pushUpGoal -= 1;  
} while(pushUpGoal > 0);
```

```
Push up 10  
Push up 9  
Push up 8  
Push up 7  
Push up 6  
Push up 5  
Push up 4  
Push up 3  
Push up 2  
Push up 1
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

- *If, else if, else*

```
if (expression logique){
    instructions;
}

else if (expression logique){
    instructions;
}

else{
    instructions;
}
```

```
if (args.length==1) {
    sayHelloTo(args[0]);
}
else if (args.length==2) {
    sayHelloTo(args[0] + "-" + args[1]);
}
else if (args.length==3) {
    sayHelloTo(args[0] + "-" + args[1] +
        "-" + args[2]);
}
else {
    sayHelloTo("world");
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

- *switch, case, break, default*: Switch utilise ce qu'on appelle des cas, pour comparer une valeur et décider si un bloc de code associé doit être exécuté.

```
switch(variable) {  
    case Valeur 1: instruction 1;  
    break;  
    case ...:  
    break;  
    case Valeur n: instruction n;  
    break;  
    default: instruction;  
}
```

```
switch(args.length) {  
    case 0: sayHelloTo("world");  
    break;  
    case 1: sayHelloTo(args[0]);  
    break;  
    case 2: sayHelloTo(args[0] + "-" +  
        args[1]);  
    break;  
    default: System.out.println("« Vous  
        devez fournir au plus 2 arguments!");  
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

Exercice: Quel est le résultat du programme suivant?

```
int n = 0;
do {
    if (n % 2 == 0) {
        System.out.println(n + " est pair");
        n += 3;
        continue;}

    if (n % 3 == 0) {
        System.out.println(n + " est multiple de 3");
        n += 5;}

    if (n % 5 == 0) {
        System.out.println(n + " est multiple de 5");
        break;}

    n += 1;
} while (true);
```

0 est pair
3 est multiple de 3
9 est multiple de 3
15 est multiple de 3
20 est multiple de 5

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les structures de contrôle

Exercice: Quel est le résultat du programme suivant?

```
int n, p;  
  
n = p = 0;  
while (n < 5) n += 2;  
p++;  
System.out.println("A : n = " + n + ", p = " + p);  
  
n = p = 0;  
while (n < 5) { n += 2; p++;}  
System.out.println("B : n = " + n + ", p = " + p);
```

A : n = 6, p = 1
B : n = 6, p = 3

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

- Les tableaux Java sont des structures pouvant contenir un nombre fixe d'éléments de même nature. Il peut s'agir d'objets ou de primitives.

Tableaux unidimensionnels

- **Déclaration** : `type nomTableau []` ; ou `type [] nomTableau` ;
- **Création**: `nomTableau = new type [dimension]`;

```
int[] tableauInt; // Déclaration d'un tableau de type int
Eleve tableauInt []; // Déclaration d'un tableau de type Eleve
String[] tableauString=new String[2]; // Déclaration et création d'un tableau de String
```


II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

- **Initialisation automatique:** Chaque élément du tableau est initialisé selon son type par l'instruction **new**:
 - 0 pour les numériques, \0 pour les caractères, false pour les booléens null pour les chaînes de caractères et les autres objets.
- **Initialisation au moment de la déclaration:** `int [] x={1, 2, 3, 4, 5} ;// tableau de cinq éléments de type int initialisé par{1, 2, 3, 4, 5}`

```
String[] tableauString=new String[2]; /* initialisation automatique grâce à  
new */  
int[] tableauInt ={1,2,3,4,5}; // Initialisation au moment de la déclaration
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

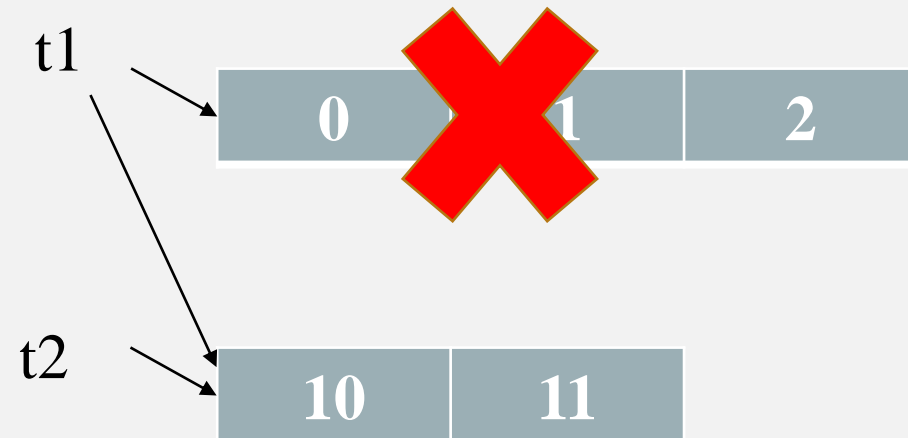
Les tableaux

- Affectation des tableaux:

```
int [] t1 = new int[3] ;  
for (int i=0 ; i<3 ; i++)  
    t1[i] = i ;  
int [] t2 = new int[2] ;  
for (int i=0 ; i<2 ; i++)  
    t2[i] = 10 + i ;  
t1=t2;
```

```
t1[0] = 5; // Quel serait le contenu de t2[0]?
```

- `System.out.println(t2[0]);` // La valeur 5 est affichée



II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

- **Taille d'un tableau:** Un tableau est un objet possédant l'attribut `length` : c'est la taille du tableau.

```
int t[] = new int[5] ;  
System.out.println ("taille de t : " + t.length) ; // affiche 5  
t = new int[3] ;  
System.out.println ("taille de t : " + t.length) ; // affiche 3
```

- **Utilisation d'un tableau:**
- **Exemple:** Un programme qui utilise un tableau de réels pour déterminer le nombre d'élèves d'une classe ayant une note supérieure à la moyenne de la classe.

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

```
public class Moyenne{
    public static void main (String args[]) {
        int i, nbEl, nbElSupMoy ; double somme ; double moyenne ;
        Scanner sc=new Scanner (System.in);
        System.out.print ("Combien d'élèves ") ;
        nbEl = sc.nextInt();
        double notes[] = new double[nbEl] ;
        for (i=0 ; i<nbEl ; i++){
            System.out.print ("donnez la note numéro " + (i+1) + " : " ) ;
            notes[i] = sc.nextDouble() ;}
        for (i=0, somme=0 ; i<nbEl ; i++) {
            somme += notes[i] ;}
        moyenne = somme / nbEl ;
        System.out.println ("\n moyenne de la classe " + moyenne) ;
        for (i=0, nbElSupMoy=0 ; i<nbEl ; i++ )
            if (notes[i] > moyenne)
                nbElSupMoy++ ;
        System.out.println (nbElSupMoy + " élèves ont plus de cette moyenne") ;
    }
}
```

```
Combien d'élèves 5
donnez la note numéro 1 : 11
donnez la note numéro 2 : 14
donnez la note numéro 3 : 13
donnez la note numéro 4 : 10
donnez la note numéro 5 : 16
moyenne de la classe 12.8
3 élèves ont plus de cette moyenne
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

Tableaux multidimensionnels

- **Déclaration** : `type nomTableau [][]` ; ou `type [][] nomTableau` ;
- **Création**: `nomTableau = new type [n1][n2]`;

```
// Déclaration et création d'un tableau à 2 dimensions de String  
String[][] tableauString=new String[5][10];
```

- Un tableau à 2 dimensions peut être initialisé comme suit:

```
int t21[][] = {{1, 2, 3}, {4, 5, 6}} // {{ligne 1},{ligne 2}}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les tableaux

Tableaux multidimensionnels

- **Déclaration et création:** Toutes les lignes d'un tableau à 2 dimensions n'ont pas forcément le même nombre d'éléments :

```
int t22[][] ;  
t22 = new int[5][];  
for( int i = 0; i< t22.length; i++){  
    t22[i]= new int [i+1]; }  
  
for( int i = 0; i< t22.length; i++){  
    for( int j = 0; j<t22[i].length; j++){  
        //accès à t22[i][j] pour le remplir par exemple }  
    }  
}
```

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Les fonctions mathématiques

- **Math.pow ()** : x puissance a
- **Math.sqrt()** : détermine la racine carrée d'un nombre
- **Math.abs ()** : $|x|$ (valeur absolue de x)
- **Math.min ()** : détermine le minimum de deux nombres
- **Math.max()** : détermine le maximum de deux nombres
- **Math.random()** : donne un nombre au hasard entre 0 et 1
- **Math.sin ()** : calcule le sinus d'un angle.
- **Math.cos ()** : calcule le cosinus d'un angle.
- **Math.tan ()** : calcule la tangente d'un angle.
- **Math.exp ()** : calculer l'exponentielle d'un nombre.
- **Math.log ()** : calcule le logarithme d'une valeur.
- **Math.floor()** : arrondit à l'entier inférieur

II. SYNTAXE ET ÉLÉMENTS DE BASE DU LANGAGE JAVA

Exercice: tableaux et fonctions

Exemple d'exécution du programme:

3 7 1 9 2 8 9 1 2 5

Il y a 2 éléments de valeur 9 dans le tableau

Il y a 7 éléments plus grand que 4 ou plus petit que 2

La somme des éléments est 47

La moyenne arithmétique des éléments est 4.7

Le plus grand élément vaut: 9