Introduction à Java

Rezak AZIZ
rezak.aziz@lecnam.net

CNAM Paris

Un peu d'histoire

- Java créé en 1995 par Sun Microsystems (racheté par Oracle en 2009).
- **Java reprend en grande partie la syntaxe vu en C.**
- Aujourd'hui utilisé dans le Web, Android, serveurs d'entreprise.
- Exemples d'application en Java:
  - Jeux videos: Minecraft
  - Applications de Big Data: Hadoop Apache
  - Applications de Gestion de Projets: Jira Atlassian
  - Editeurs logicielle: Eclipse IDE

Ne pas confondre avec Javascript (langage de scripts utilisé sur les sites web)

Écosystème Java : JDK, JRE, JVM

JDK

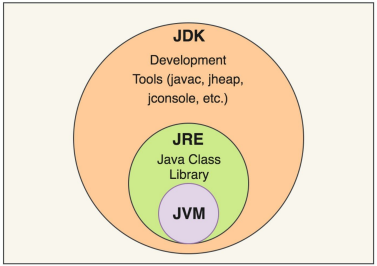
Java Development Kit : outils pour compiler et exécuter du code Java.

JRE

Java Runtime Environment : contient la JVM et les bibliothèques de base.

JVM

Java Virtual Machine : exécute le bytecode Java.



Comparaison avec le C

C

- Compilation → exécutable natif.
- Gestion mémoire manuelle avec `malloc/free`.
- Pointeurs.

Java

- Compilation → bytecode (langage intérmidiaire).
- Le bytecode est executé par une machine virtuelle (JVM)
- Avec la JVM, Java est portable avec Windows, Mac, Linux, etc.
- Gestion mémoire automatique (Garbage Collector).

Le Garbage Collector

- En C : le programmeur doit libérer la mémoire (`free`).
- En Java : la JVM libère automatiquement les objets non utilisés.
- Avantages : moins d'erreurs (fuites mémoire, segmentation fault).

L'IDE Eclipse

- **Eclipse** est un **environnement de développement intégré** (IDE) pour Java.
- Outils inclus :
  - Éditeur de code avec coloration syntaxique.
  - Auto-complétion du code.
  - Gestion de projets (packages, classes).
  - Compilation et exécution intégrées.
  - Débogueur pas-à-pas.
- Alternatives populaires : IntelliJ IDEA, NetBeans, VS Code.

Premier Programme en Java

Exercice 1 : Créer un projet

1. Ouvrir Eclipse.
2. Menu : **File → New → Project → Java Project**.
3. Nommer le projet : `PremierProjet`.
4. Vérifier la présence du dossier `src/`.

Structure minimale d'un programme Java

- Un fichier Java contient une **classe publique** (même nom que le fichier).
- La méthode `main` est le **point de départ** de l'exécution.
- Chaque instruction se termine par un `;`.
- Les { et } délimitent les blocs de code.

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

- **class Bonjour** : la "boîte" qui contient le code (nom = fichier).
- **main** : point d'entrée du programme.
- **System.out.println(...)** : affiche un message à l'écran.

Exercice 2 : Créer une classe

1. Clic droit sur `src/` → **New → Class**.
2. Nommer la classe : `Hello`.
3. Cocher la case : **public static void main(String[] args)**.

Exercice 3 : Premier programme

Dans la méthode main de la classe `Hello`, écrire :

```
System.out.println("Bonjour Eclipse !");
```

Puis exécuter avec : **Run → Run As → Java Application**.

L'autocomplétion dans un IDE

Qu'est-ce que l'autocomplétion ?

- Fonctionnalité de l'IDE qui propose automatiquement des mots-clés, méthodes, ou variables.
- Elle aide à écrire plus vite et à éviter les erreurs de syntaxe.
- Elle affiche souvent la **documentation** d'une méthode ou d'une classe.

Exercice 4 : Autocomplétion

- Taper `System.` et observer les suggestions.
- Utiliser l'autocomplétion pour écrire `System.out.println`.
- Tester aussi avec :
  - Les fonctions mathématiques : `Math.` (exemple : `Math.sqrt(16)`)
  - Les fonctions sur les chaines de caractères : `String.` (exemple : `"test".length()`)

Les variables en Java

Qu'est-ce qu'une variable ?

Une **variable** sert à stocker une valeur temporairement en mémoire. Elle a un **type**, un **nom** et une **valeur**.

Exemples

```
int age = 20;
double taille = 1.75;
String nom = "Amine";
```

Syntaxe générale

```
type nom = valeur;
```

Types les plus courants :

- `int` → nombres entiers
- `double` → nombres à virgule
- `String` → texte
- `boolean` → vrai ou faux

Remarque

Le nom de variable commence toujours par une lettre, et respecte la casse : `Nom` $\neq$ `nom`.

Afficher du texte et des variables

Afficher à l'écran

En Java, on utilise :

```
System.out.println("Bonjour !");
System.out.println("Âge : " + age);
System.out.println("Nom : " + nom);
```

Résultat à l'écran

```
Bonjour !
Âge : 20
Nom : Amine
```

Astuce

L'opérateur + sert à **concaténer** du texte et des variables.

À retenir

Chaque ligne d'affichage utilise `println()`
$\rightarrow$ elle saute une ligne après le message.

Exercice 5 : Variables et affichage

Déclarer et afficher des variables :

```
int age = 20;
double taille = 1.75;
String nom = "Alice";

System.out.println("Nom: " + nom
  + ", Âge: " + age
  + ", Taille: " + taille);
```

Résultat attendu : `Nom: Alice, Âge: 20, Taille: 1.75`

Lire une variable au clavier en Java

Pour lire une valeur saisie par l'utilisateur, il faut utiliser un Scanner:

```
import java.util.Scanner;

Scanner clavier = new Scanner(System.in);

String nom = clavier.nextLine();

int age = clavier.nextInt();

clavier.close();
```

Explications

- `Scanner` est un objet qui permet de lire les entrées clavier.
- `nextLine()` $\rightarrow$ lit une ligne de texte.
- `nextInt()` $\rightarrow$ lit un nombre entier.

Important

Pour utiliser cette classe, il faut importer au préalable le package (bibliothèque) qui contient la classe Scanner avec l'instruction `import java.util.Scanner;`.

Exercice 6 : Lire et afficher des variables

Lire les variables au clavier et les afficher :

```
Scanner sc = new Scanner(System.in);

System.out.print("Entrez votre nom : ");
String nom = sc.nextLine();

System.out.print("Entrez votre âge : ");
int age = sc.nextInt();

System.out.print("Entrez votre taille : ");
double taille = sc.nextDouble();
System.out.println("Nom: " + nom
  + ", Âge: " + age
  + ", Taille: " + taille);
```

Les instructions conditionnelles en Java

Principe

Une **instruction conditionnelle** permet d'exécuter du code uniquement si une condition est vraie.

```
if (condition) {
    // instructions si vrai
} else if (autre_condition) {
    // instructions si la première est fausse
    // et la deuxième vraie
} else {
    // instructions si toutes sont fausses
}
```

Remarque importante

Les instructions conditionnelles en Java fonctionnent exactement comme en **langage C** : même syntaxe, mêmes opérateurs logiques et de comparaison (==, !=, <, >, &&, ||, !).

Les différentes formes d'instructions conditionnelles en Java

1. Structure simple

```
if (condition) {
    // instructions si la condition est vraie
}
```

2. Structure avec sinon

```
if (condition) {
    // si vrai
} else {
    // si faux
}
```

3. Structure avec plusieurs cas

```
if (condition1) {
    // si condition1 vraie
} else if (condition2) {
    // si condition1 fausse et condition2 vraie
} else {
    // sinon
}
```

4. L'opérateur ternaire (comme en C)

```
switch (variable) {
    case valeur1:
        // instructions si variable == valeur1
        break;
    case valeur2:
        // instructions si variable == valeur2
        break;
    default:
        // instructions si aucune des valeurs
        ne correspond
}
```

Exercice 7 : Conditions

Exécuter un code selon une condition:

```
Scanner sc = new Scanner(System.in);
int age = sc.nextInt();

if(age >= 18) {
   System.out.println("Majeur");
} else {
   System.out.println("Mineur");
}
```

Les boucles en Java

Principe

Une **boucle** permet de répéter plusieurs fois un même bloc d'instructions tant qu'une condition est vraie.

Remarque importante

Les boucles en Java fonctionnent exactement comme en **langage C** : même syntaxe, même logique, mêmes règles de portée.

Les différentes formes de boucles en Java

1. Boucle `while`

```
initialiser la variable;

while (condition) {
    // effectuer une tâche répétée
    // mettre à jour la variable
}
```

2. Boucle `for`

```
for (initialisation; condition; mise à jour) {
    // exécuter une série d'instructions
}
```

4. Boucle `do...while`

```
initialiser la variable;

do {
    // répéter une action
    // mettre à jour la variable
} while (condition);
```

Exercice 8 : Boucles

Afficher les nombres de 1 à 10 :

```
for(int i = 1; i <= 10; i++) {
    System.out.println(i);
}
```

Afficher les nombres pairs entre 1 et 20 :

```
for(int i = 2; i <= 20; i += 2) {
    System.out.println(i);
}
```

Exercice 9 : Table de multiplication

Demander un nombre et afficher sa table :

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();

for(int i = 1; i <= 10; i++) {
   System.out.println(n + " x " + i
      + " = " + (n * i));
}
```

Déclaration d'un tableau en Java

Forme générale

```
// Déclaration d'un tableau sans initialisation
type[] nomTableau;

// Déclaration avec taille fixe
type[] nomTableau = new type[taille];

// Déclaration avec initialisation directe
type[] nomTableau = {val1, val2, val3, ...};
```

Important

En Java, un tableau est un **objet**. Il faut le créer explicitement avec `new`, sauf en cas d'initialisation directe.

Manipuler les cases

Même syntaxe qu'en C

```
// Accéder à un élément du tableau
nomTableau[indice];

// Modifier un élément du tableau
nomTableau[indice] = nouvelleValeur;
```

Rappel

- Un tableau regroupe des valeurs du même type, accessibles par un indice.
- Les indices commencent à **0**.

Exercice 10 : Tableaux

Déclarer et parcourir un tableau :

```
int[] notes = {12, 15, 8, 19};
int somme = 0;

for(int i = 0; i < notes.length; i++) {
    somme += notes[i];
}

double moyenne = (double)somme / notes.length;
System.out.println("Moyenne = " + moyenne);
```

Manipulation des chaînes de caractères en Java

Forme générale

```
// Déclaration d'une chaîne sans initialisation
String nomChaine;

// Déclaration avec affectation
String nomChaine = "texte";

// Déclaration avec le mot-clé new
String nomChaine = new String("texte");
```

Important

En Java, une chaîne de caractères est un **objet** de la classe `String`. Elle est **immuable** : une fois créée, son contenu ne peut pas être modifié directement.

Manipuler les caractères

```
// Accéder à un caractère
nomChaine.charAt(indice);

// Connaître la longueur
nomChaine.length();

// Comparer deux chaînes
nomChaine.equals(autreChaine);

// Concaténer deux chaînes
nomChaine + autreChaine;
```

Des exemples plus détaillés sont développé en TP8, TP9 et TP10 à travers des exercices.