



Introduction

Au langage JAVA



SAGIM 2024/2025



les **variables** sont des espaces de mémoire utilisés pour stocker des données que ton programme peut manipuler. Elles ont un **type**, un **nom** et une **valeur**.

int age = 25;

double prix = 19.99;

char lettre = 'A';

boolean estVrai = true;

String nom = "Ouadi";



1. Types primitifs

| Type | Taille | Description | Exemple |
|---------|----------|----------------------------|----------------------|
| int | 4 octets | Nombre entier | int x = 10; |
| double | 8 octets | Nombre à virgule flottante | double d = 3.14; |
| float | 4 octets | Moins précis que double | float f = 3.14f; |
| char | 2 octets | Caractère Unicode | char c = 'A'; |
| boolean | 1 bit | true OU false | boolean b = true; |
| long | 8 octets | Grand entier | long l = 123456789L; |
| short | 2 octets | Petit entier | short s = 100; |
| byte | 1 octet | Très petit entier | byte b = 12; |



Règles de nommage

- Doit **commencer** par une **lettre**, un **_** ou **\$**.
- Ne doit **pas contenir d'espaces**.
- Peut contenir des **chiffres** après la première lettre.
- Utilise généralement la **camelCase** :
`maVariable`, `nombreTotal`, etc.

les variables



```
public class ExempleVariables {  
    public static void main(String[] args) {  
        int age = 30;  
        double salaire = 4500.50;  
        char genre = 'M';  
        boolean estEmploye = true;  
        String nom = "Ahmed";  
        System.out.println("Nom : " + nom);  
        System.out.println("Âge : " + age);  
        System.out.println("Salaire : " + salaire);  
        System.out.println("Genre : " + genre);  
        System.out.println("Employé ? " + estEmploye);  
    }  
}
```



Règles de nommage:

- Doit **commencer** par une lettre, un _ ou \$.
- Ne doit **pas contenir d'espaces**.
- Peut contenir des **chiffres** après la première lettre.
- Utilise généralement la **camelCase** :
ma**V**ariable, nombre**T**otal, etc.

ÉCRIRE (afficher) en Java



Utiliser `System.out.print()` et `System.out.println()`

```
System.out.print("Bonjour ");
```

```
System.out.println("tout le monde");
```

Exemple :

```
System.out.print("Nom : ");
```

```
System.out.println("Ouadi");
```


Pour cela, on utilise la classe **Scanner** du paquet `java.util`.

Étapes :

1. Importer **Scanner**

2. Créer un objet **Scanner**

3. Lire les données avec la méthode appropriée

LIRE (entrer) des données de l'utilisateur

LANGAGE JAVA

10 ◀

```
import java.util.Scanner;
public class LireEcrire {
    public static void main(String[] args) {
        Scanner clavier = new Scanner(System.in);

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

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

        System.out.println("Bonjour "+nom+", vous avez"+age+" ans.");
        clavier.close();
    }
}
```



| Méthode | Ce qu'elle lit | Exemple |
|----------------------|--------------------|---|
| nextLine() | Une ligne de texte | <code>String nom = clavier.nextLine();</code> |
| next() | Un seul mot | <code>String mot = clavier.next();</code> |
| nextInt() | Un entier | <code>int x = clavier.nextInt();</code> |
| nextDouble() | Un nombre décimal | <code>double d = clavier.nextDouble();</code> |
| nextBoolean() | true ou false | <code>boolean b = clavier.nextBoolean();</code> |

S A G I M 2 0 2 4 / 2 0 2 5

▶▶▶ Les opérateurs





Les opérateurs en Java te permettent de manipuler des variables et de construire des expressions logiques, mathématiques ou relationnelles.



Les opérateurs en java

Les opérateurs arithmétiques

| Opérateur | Signification | Exemple (int a = 10; int b = 3;) |
|-----------|------------------|----------------------------------|
| + | Addition | $a + b \rightarrow 13$ |
| - | Soustraction | $a - b \rightarrow 7$ |
| * | Multiplication | $a * b \rightarrow 30$ |
| / | Division entière | $a / b \rightarrow 3$ |
| % | Modulo (reste) | $a \% b \rightarrow 1$ |



Les opérateurs en java

Les opérateurs relationnels (comparaison)

| Opérateur | Signification | Exemple (int a = 10; int b = 3;) |
|-----------|-------------------|-----------------------------------|
| == | Égal à | $a == b \rightarrow \text{false}$ |
| != | Différent de | $a != b \rightarrow \text{true}$ |
| < | Inférieur à | $a < b \rightarrow \text{false}$ |
| > | Supérieur à | $a > b \rightarrow \text{true}$ |
| <= | Inférieur ou égal | $a <= b \rightarrow \text{false}$ |
| >= | Supérieur ou égal | $a >= b \rightarrow \text{true}$ |



Les opérateurs en java

Les opérateurs logiques (booléens)

| Opérateur | Signification | Exemple (int a = 10; int b = 3;) |
|-----------|---------------|----------------------------------|
| && | ET logique | (a > 5 && b < 5) → true |
| | OU logique | (a > 5 b > 5) → true |
| ! | NON logique | !(a > 5) → false |

Les opérateurs en java

Les opérateurs d'affectation

| Opérateur | Équivaut à | Exemple (int x = 5;) |
|-----------|----------------------|----------------------|
| = | Affectation | x = 10; |
| += | Addition et affect. | x += 2; → x = x + 2; |
| -= | Soustraction et aff. | x -= 3; → x = x - 3; |
| *= | Multiplication | x *= 4; |
| /= | Division | x /= 2; |
| %= | Modulo | x %= 3; |



Les opérateurs en java

Les opérateurs d'affectation

| Opérateur | Équivaut à | Exemple (int x = 5;) |
|-----------|----------------------|----------------------|
| = | Affectation | x = 10; |
| += | Addition et affect. | x += 2; → x = x + 2; |
| -= | Soustraction et aff. | x -= 3; → x = x - 3; |
| *= | Multiplication | x *= 4; |
| /= | Division | x /= 2; |
| %= | Modulo | x %= 3; |



Les opérateurs en java

Les opérateurs d'incrémentation et de décrémentation

| Opérateur | Signification | Exemple (int x = 5;) |
|------------|-------------------------|---|
| ++x | Incrémente puis utilise | System.out.println(++x); // 6 |
| x++ | Utilise puis incrémente | System.out.println(x++); // 5, puis x=6 |
| --x | Décrémente puis utilise | --x |
| x-- | Utilise puis décrémente | x-- |



Les opérateurs en java

```
public class OperateursJava {
    public static void main(String[] args) {
        int a = 10, b = 3;
        System.out.println("Addition : " + (a + b));
        System.out.println("Modulo : " + (a % b));
        System.out.println("a > b : " + (a > b));
        System.out.println("a est pair ? " + (a % 2 == 0));
        boolean condition = (a > 5 && b < 5);
        System.out.println("Condition logique:" + condition);
        a += 2;
        System.out.println("a après += 2 : " + a);
    }
}
```



Les structures de controle :



if, else if, else

```
if (condition) {
```

```
    // instructions si condition vraie
```

```
} else if (autreCondition) {
```

```
    // instructions si autreCondition vraie
```

```
} else {
```

```
    // instructions si aucune condition n'est vraie
```

```
}
```



```
int age = 20;
if (age < 18) {
    System.out.println("Mineur");
} else {
    System.out.println("Majeur");
}
```

SWITCH (choix multiple)



```
int jour = 3;

switch (jour) {
    case 1:
        System.out.println("Lundi");
        break;
    case 2:
        System.out.println("Mardi");
        break;
    default:
        System.out.println("Autre jour");
}
```


LES BOUCLES : (while)

```
int i = 1;
while (i <= 5) {
    System.out.println("i= " + i);
    i++;
}
```

LES BOUCLES : (do...while)



```
int i = 1;

do {
    System.out.println("i =" + i);
    i++;
} while (i <= 5);
```

LES BOUCLES : (for)

```
for (int i = 1; i <= 5; i++) {  
    System.out.println("i= " + i);  
}
```

SAGIM 2024/2025

▶▶▶ EXERCICES



Exercice 1 :



Lire un entier et
afficher s'il est
pair ou impair.

EXERCICE 1 :



```
import java.util.Scanner;
public class PairOuImpair {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Entrez un entier : ");
        int n = sc.nextInt();
        if (n % 2 == 0) {
            System.out.println("Le nombre est pair.");
        } else {
            System.out.println("Le nombre est impair.");
        }
        sc.close();
    }
}
```

Exercice 2 :



**Afficher les
10 premiers
entiers**

EXERCICE 2 :



```
public class Entiers1a10 {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.println(i);  
        }  
    }  
}
```


Exercice 3 :



**Calcul de la
somme de 1 à
nombreEnt**

EXERCICE 3 :



```
import java.util.Scanner;

public class SommeLaEntier {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Entrez une valeur entiere : ");
        int nombreEnt = sc.nextInt();
        int somme = 0;
        for (int i = 1; i <= nombreEnt; i++) {
            somme += i;
        }
        System.out.println("Somme = " + somme);
        sc.close();
    }
}
```

Exercice 4 :



Table de multiplication d'un nombre

EXERCICE 4 :



```
import java.util.Scanner;

public class TableMultiplication {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Entrez un nombre : ");
        int n = sc.nextInt();
        for (int i = 1; i <= 10; i++) {
            System.out.println(n + "x" + i + " = " + (n * i));
        }
        sc.close();
    }
}
```



▶▶▶ Les méthodes en JAVA



Une **méthode** est un bloc de code qui effectue une tâche spécifique. Elle peut recevoir des **paramètres** (entrées), retourner un **résultat**, ou ne rien retourner du tout.

Qu'est-ce qu'une méthode en Java ?



| Terme | Utilisation en Java | Explication |
|-----------------|---|--|
| Méthode |  Oui (terme officiel) | Bloc de code défini dans une classe , qui peut être appelé par un objet ou statiquement |
| Fonction |  Pas un terme officiel | Le mot "fonction" est utilisé par abus de langage, souvent en comparaison avec d'autres langages comme C, Python, etc. |

Qu'est-ce qu'une méthode en Java ?



Les différents types de méthodes

| Type de méthode | Exemple |
|------------------------------|---|
| Avec retour et paramètres | <code>int addition(int a, int b)</code> |
| Sans retour, avec paramètres | <code>void afficher(String nom)</code> |
| Avec retour, sans paramètres | <code>String salutation()</code> |
| Sans retour, sans paramètres | <code>void direBonjour()</code> |

Qu'est-ce qu'une méthode en Java ?



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

Qu'est-ce qu'une méthode en Java ?



```
public class FonctionsDemo {  
  
    public static void direBonjour() {  
        System.out.println("Bonjour !");  
    }  
  
    public static int addition(int a, int b) {  
        return a + b;  
    }  
  
    public static String salutation() {  
        return "Bienvenue dans le monde Java !";  
    }  
  
    public static void main(String[] args) {  
  
        direBonjour();  
        int somme = addition(4, 7);  
        System.out.println("Somme : " + somme);  
        System.out.println(salutation());  
    }  
}
```

S A G I M 2 0 2 4 / 2 0 2 5

▶▶▶ Exercices



Exercice 1 :



Créer une méthode **carre** qui prend un nombre entier en paramètre et retourne son carré. Appeler cette méthode depuis `main`.

EXERCICE 1 :



```
import java.util.Scanner;

public class CarreNombre {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Entrez un entier : ");
        int n = sc.nextInt();
        int resultat = carre(n);
        System.out.println("Le carré de " + n + " est : " + resultat);
        sc.close();
    }

    public static int carre(int x) {
        return x * x;
    }
}
```

EXERCICE 1 :

Que signifie static en Java ?

Le mot-clé **static** indique qu'un **membre** (méthode ou **variable**) appartient à la **classe elle-même** et **non** à une **instance** (objet) de la classe.

En clair :

- **static** = pas besoin de créer un objet pour y accéder
- utilisé surtout pour les **méthodes utilitaires**, les **constantes**, ou le **point d'entrée** du programme (**main**)