



Présentation du langage Java

I- Règles générales du langage

Le langage Java a une syntaxe proche du langage C.

1- Le langage Java est sensible à la casse ; c'est-à-dire qu'il fait la différence entre lettres majuscules (A..Z) et minuscules (a..z). Un nom contenant des majuscules est différent du même nom écrit en minuscules. **Exemple** : Le nom **varx** est différent de **varX**, **VarX** ou **VARX**.

2- En langage Java, chaque instruction ou déclaration se termine par un point-virgule ' ; '.

3- Les mots clés du langage Java

- ✓ Un mot clé est un identificateur (mot) qui a une signification particulière au langage.
- ✓ Les mots clés du langage Java sont : *abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while.*
- ✓ *true, false* et *null* sont des constantes.
- ✓ Il est interdit d'utiliser les mots clés pour nommer des variables, constantes, fonctions, ...

II- Les types de données primitifs (simples)

Le type correspond au genre d'informations utilisées. Il existe 8 types primitifs qui stockent directement des valeurs simples en mémoire (dans la pile) : byte, short, int, long, float, double, char, boolean.

Type de données	Plage (Intervalle)	Taille (Octet)	Exemples
byte	-128 à 127 [-2^7 .. $2^7 - 1$]	1	-70, 5, 18, 100
short	-32 768 à 32 767 [-2^{15} .. $2^{15} - 1$]	2	-3465, 17350
int	$[-2^{31}$.. $2^{31} - 1]$	4	-98 765, 9 345 678
long	$[-2^{63}$.. $2^{63} - 1]$	8	-5 123 456 789L, 7 654 321 098 L
float	-2^{-149} à $(2 - 2^{-23}) * 2^{127}$ Nombres flottants avec 7 chiffres décimaux	4	-2,53f, 165.78F, -3567.1456F
double	-2^{-1074} à $(2 - 2^{-52}) * 2^{1023}$ Nombres flottants avec double précision	8	-2,53, 165.78, -3567.1456
char	'\u0000' (0) à '\uffff' (65 535) (0 à $2^{16} - 1$). Des caractères Unicode	2	'A', 65
boolean	true, false	1 bit à 1 octet	true, false

Remarque : Le type String ($\equiv$ Chaîne) n'est pas un type primitif (simple) de Java. On le verra plus loin.

III- La liste des opérateurs :

- Opérateurs arithmétiques : + - * / % (modulo).
- Opérateurs d'affectation : = += -= *= /= %= ++ (incrément) -- (décrément).
- Opérateurs relationnels : == (égal) != (différent) < <= > >=
- Opérateurs logiques : && (AND ≡ ET logique) || (OR ≡ OU logique) ! (NOT ≡ NON logique).
- Opérateurs logiques bit à bit (Bitwise operators) : permettent de manipuler directement les bits (0 et 1) des types entiers (char, byte, short, int, long) :
 - & (ET par bit),
 - | (OU),
 - ^ (XOR ≡ OU Exclusif),
 - ~ (NON ≡ Complément à un),
 - << (Décalage à gauche, les bits vides à droite sont remplacés par des zéros ≡ équivalent à une multiplication par 2ⁿ),
 - >> (Décalage à droite signé, le bit de signe est conservé ≡ équivalent à une division par 2ⁿ),
 - >>> (Décalage à droite non-signé, les bits vides à gauche sont remplacés par des zéros).
 - Chaque opérateur bit-à-bit peut être combiné avec l'affectation =, ce qui donne les opérateurs suivants : &= |= ^= ~= <<= >>= >>>=

IV– Passage du langage C au langage JAVA

Concept	Langage C	Langage Java
Point d'entrée	int main()	public static void main(String[] args)
Regroupement de données	struct Point { int x; int y; };	public class Point { int x; int y; }
Allocation dynamique	p = malloc(sizeof(struct Point));	p = new Point();
Libération mémoire	free(p);	Automatique (Garbage Collector)
Affichage console	printf("Valeur : %d\n", val);	System.out.print("Valeur : " + val); System.out.println("Valeur : " + val);
Tableaux	int tab[10]; ou int *tab;	int[] tab = new int[10];
Taille d'un tableau	Passer la taille en paramètre	tab.length
Les commentaires	/* ... */ ou //.....	/* ... */ ou //.....
Les structures de contrôle	if (condition) { ... } else { ... }	if (condition) { ... } else { ... }
	if ... else if ... else if ... else ...	if ... else if ... else if ... else ...

	variable = (condition)? val1 : val2 ; Exemple : max = (x > y)? x : y ;	variable = (condition)? val1 : val2 ; Exemple : max = (x > y)? x : y ;
	switch (expression) { case valeur1: // instructions break; case valeur2: // instructions break; default: // instructions par défaut }	switch (expression) { case valeur1: // instructions break; case valeur2: // instructions break; default: // instructions par défaut }
	for (initialisation ; condition ; mise à jour) { ... }	for (initialisation ; condition ; mise à jour) { ... }
	while (condition) { ... }	while (condition) { ... }
	do { ... } while (condition) ;	do { ... } while (condition) ;

V– Les types de données

En java, les types de données sont des types primitifs (déjà vus), soit des classes.

Une classe est un **type de données défini par le programmeur** qui regroupe **données** (attributs) et **traitements** (comportement) dans une même entité. C'est le **pilier de la programmation orientée objet (POO)**.

En Java, *tout code Java doit être à l'intérieur d'une classe.*

VI– Un identificateur (identifiant)

Un identificateur est un nom (ou un mot) constitué d'une suite de lettres de l'alphabet ('A'..'Z' et 'a'..'z') et de chiffres ('0'..'9') **commençant par une lettre** (il ne peut pas commencer par un chiffre).

Le caractère '_' souligné (underscore) est considéré comme une lettre.

Remarques : ✓ L'identificateur doit être différent de tous les mots clés (char, int, float, if, while, for, ...).

✓ Il ne doit pas contenir des espaces. L'espace est, en général, remplacé par le caractère '_' souligné.

VII– Déclaration de variables

✓ Une variable est déclarée avec la syntaxe suivante : **type nomVariable ;**

✓ **type** est l'un des types définis en langage Java (byte, short, char, int, float ...).

✓ **nomVariable** est un identificateur.

✓ Une variable peut être initialisée à la déclaration : **type nomVariable = valeurInitiale ;**

✓ Plusieurs variables d'un même type peuvent être déclarées (et initialisées) dans la même déclaration.

Syntaxe : **type nomVariable1 [= valeur1], nomVariable2 [= valeur2] ... nomVariableN [= valeurN] ;**

Exemples : char a ;

```
int n=5 , nEtdudiant ;  
float x1, x2, delta=0 ;
```

VIII– Déclaration de constantes

- ✓ Une constante est déclarée avec la syntaxe suivante : **final type NOM_CONSTANTE = Valeur ;**
- ✓ **type** est l'un des types définis en langage Java (char, int, float ...).
- ✓ **NOM_CONSTANTE** est un identificateur.

Exemples : `final char CAR = 'a' ;      final char CAR_A_MAJ = 65 ;`
`final int NB_MODULES = 8 ;              final float PI = 3.14f ; // Ajouter f pour float`
`final double VIT_LUMIERE = 3E8 ; // Vitesse de la lumière 3x108 m/s. Notation exponentielle`

Remarque : Par convention, le nom d'une variable est écrit en minuscules (ou au moins commence par une minuscule). Le nom d'une constante est écrit uniquement en majuscules.

IX– L'instruction de lecture

Il faut ajouter au début du programme l'instruction suivante : **import java.util.Scanner;**

`Scanner scanner = new Scanner(System.in) ; // Ouvrir le flux System.out`

/* Décommentez la ligne ci-dessous pour utiliser le point à la place de la virgule dans les nombres réels : float et double. Exemple : 3.45 au lieu de 3,45 */

`// scanner.useLocale(Locale.US);`

`// 1. Lecture d'un byte`

`System.out.print("Entrez un byte (-128 à 127) : ");`

`byte b = scanner.nextByte();`

`System.out.println("Vous avez saisi le byte : " + b);`

`// 2. Lecture d'un short`

`System.out.print("Entrez un short (-32768 à 32767) : ");`

`short s = scanner.nextShort();`

`System.out.println("Vous avez saisi le short : " + s);`

`// 3. Lecture d'un int`

`System.out.print("Entrez un int : ");`

`int i = scanner.nextInt();`

`System.out.println("Vous avez saisi l'int : " + i);`

`// 4. Lecture d'un long`

`System.out.print("Entrez un long : ");`

`long l = scanner.nextLong();`

```
System.out.println("Vous avez saisi le long : " + l);
```

```
// 5. Lecture d'un float
```

```
System.out.print("Entrez un float (utilisez , ou .) : ");
```

```
float f = scanner.nextFloat();
```

```
System.out.println("Vous avez saisi le float : " + f);
```

```
// 6. Lecture d'un double
```

```
System.out.print("Entrez un double : ");
```

```
double d = scanner.nextDouble();
```

```
System.out.println("Vous avez saisi le double : " + d);
```

```
// 7. Lecture d'un char
```

```
System.out.print("Entrez un caractère : ");
```

```
char c = scanner.next().charAt(0);
```

```
System.out.println("Vous avez saisi le char : " + c);
```

```
// 8. Lecture d'un boolean
```

```
System.out.print("Entrez un boolean (true/false) : ");
```

```
boolean bool = scanner.nextBoolean();
```

```
System.out.println("Vous avez saisi le boolean : " + bool);
```

```
scanner.close(); // A la fin de la lecture, fermer le flux
```

X- L'instruction d'affichage à l'écran

Les deux instructions **System.out.print()** et **System.out.println()** permettent l'affichage à l'écran. Il existe deux types d'affichage :

1- L'affichage d'un texte : **Syntaxe** : **System.out.print("texte") ;**

Exemple : **System.out.print("Donner la valeur de a : ") ;**

2- Affichage de la valeur d'une variable : **Syntaxe** : **System.out.print(nomVariable) ;**

Exemple : Si **a** est une variable, on peut écrire : **System.out.print(a) ;**

✓ L'instruction **println** ajoute automatiquement un saut de ligne à la fin de l'affichage.

✓ On peut regrouper plusieurs écritures dans la même instruction en utilisant l'opérateur +.

Exemple : **System.out.println ("Le résultat = " + a) ;**

✓ Certains caractères spéciaux ont une signification particulière pour print et println :

\n : Saut de ligne **\t** : Tabulation **** : Affiche le caractère \ **\"** : Affiche le caractère "

XI– Règles de priorité des opérateurs

Les opérateurs n'ont pas la même priorité. Ci-dessous les opérateurs sont classés par ordre décroissant de priorité. Si plusieurs opérateurs de même priorité existent dans une expression, ces opérateurs sont associatifs à gauche. C'est-à-dire l'opération la plus à gauche est évaluée (exécutée) en premier.

Remarque importante : En cas de doute, il faut utiliser des parenthèses.

1. Parenthèses et accès : `()` (parenthèses), `[]` (tableau), `.` (accès aux membres)
2. Incrémentation et décrémentation (préfixe) : `++a`, `--a`, `+`, `-` (unaires), `!`, `~`
3. Incrémentation et décrémentation (postfixe) : `a++`, `a--`
4. Cast (conversion de type) : `(type)`
5. Multiplication, division, modulo : `*`, `/`, `%`
6. Addition et soustraction : `+`, `-`
7. Décalage (shift) : `<<`, `>>`, `>>>`
8. Relationnels : `<`, `<=`, `>`, `>=`, `instanceof`
9. Égalité : `==`, `!=`
10. ET bit à bit : `&`
11. OU exclusif bit à bit : `^`
12. OU inclusif bit à bit : `|`
13. ET logique : `&&`
14. OU logique : `||`
15. Opérateur ternaire : `?:`
16. Affectation : `=`, `+=`, `-=`, `*=`, `/=`, etc. [1, 3, 4]

XII– Raccourcis utiles de l'IDE Netbeans

Navigation et recherche

- **Ctrl + F** : Rechercher dans le fichier.
- **Ctrl + H** : Remplacer dans le fichier.
- **Alt + F7** : Trouver les utilisations (Find Usages).
- **Ctrl + B** : Aller à la déclaration.
- **Ctrl + Shift + B** : Aller à la source.
- **Ctrl + G** : Aller à une ligne spécifique.

Édition et écriture de code

- **Ctrl + Space** : Activer l'autocomplétion (Code Completion).
- **Ctrl + Shift + I** : Corriger / importer automatiquement les bibliothèques manquantes.
- **Alt + Enter** : Afficher les astuces de correction rapide (Quick Fix).
- **Ctrl + E** : Supprimer la ligne actuelle.
- **Alt + Shift + F** : Formater le code source (indentation).
- **psvm + Tab** : Générer la méthode principale `public static void main(String[] args)`.
- **main + Tab** : Générer la méthode principale `public static void main(String[] args)`.
- **sout + Tab** : Générer `System.out.println("");`

Exécution et débogage

- **F6** : Exécuter le projet (Run Project).
- **Shift + F6** : Exécuter le fichier en cours (Run File).
- **Ctrl + F9** : Déboguer le projet (Debug Project).
- **Ctrl + F8** : Basculer un point d'arrêt (Breakpoint).