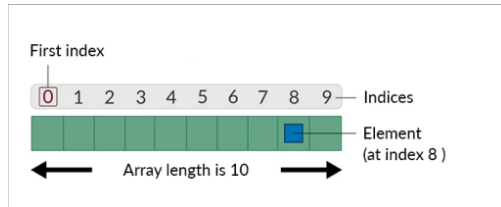


Les Tableaux en Java



Les **tableaux en Java** (ou arrays) sont des structures de données qui permettent de stocker plusieurs valeurs du même type sous une seule variable. Voici une introduction complète à leur sujet :

1. Déclaration d'un tableau

```
int t[]; // peut s'écrire
int[] t;
...
int[] t1, t2; // t1 et t2 sont des références à des tableaux d'entiers
int t1[], t2[]; // écriture équivalente
...
int t1[], n, t2[]; // t1 et t2 sont des tableaux d'entiers, n est entier
Point tp[]; // tp est une référence à un tableau d'objets de type Point
Point a, tp[]; // a est une référence à un objet de type Point
... // tp est une référence à un tableau d'objets de type Point
int t[5]; // erreur : on ne peut pas indiquer de dimension ici
```

Pour déclarer un tableau en Java, on utilise la syntaxe suivante :

type[] nomTableau;

Exemples :

```
int[] tableauEntiers;
String[] tableauChaines;
```

2. Création d'un tableau

```
int[] myInts = new int[10];
int[] intValues = {1, 5, 18, 3, 7, 9, 23, 5, 11, 2};
```

Pour créer un tableau, on utilise l'opérateur **new** et on spécifie sa taille.

Exemple :

```
int[] tableauEntiers = new int[5]; // Crée un tableau pour 5 entiers
```

Remarque : Les indices d'un tableau commencent toujours à 0 en Java.

3. Initialisation d'un tableau

Diagram illustrating the components of array declarations:

```
int[] myInts = new int[10];
int[] intValues = {1, 5, 18, 3, 7, 9, 23, 5, 11, 2};
```

Annotations:

- Type of array:** `int[]`
- array variable name:** `myInts`
- element type:** `int`
- length:** `10`
- List of Initialization Values:** `{1, 5, 18, 3, 7, 9, 23, 5, 11, 2}`
- Type of array:** `int[]`
- array variable name:** `intValues`

Un tableau peut être initialisé lors de sa déclaration ou après.

a) Initialisation directe :

```
int[] nombres = {1, 2, 3, 4, 5};
```

b) Initialisation après déclaration :

```
int[] nombres = new int[3];
nombres[0] = 10;
nombres[1] = 20;
nombres[2] = 30;
```

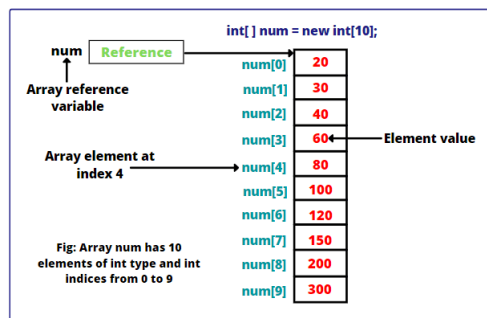


Figure 1: “Exemple d'allocation en mémoire”

4. Accès aux éléments

Les éléments d'un tableau sont accessibles via leur **indice**.

Exemple :

```
System.out.println(nombres[0]); // Affiche le premier élément (10)
```

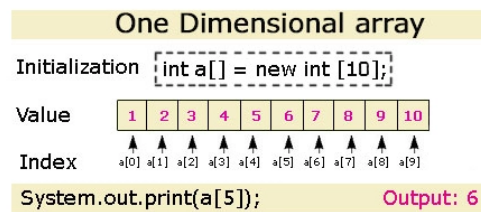


Figure 2: “Exemple de Tableau à une Dimension”

5. Longueur d'un tableau

Pour obtenir la taille d'un tableau, utilisez la propriété `.length`.

Exemple :

```
System.out.println(nombres.length); // Affiche 3
```

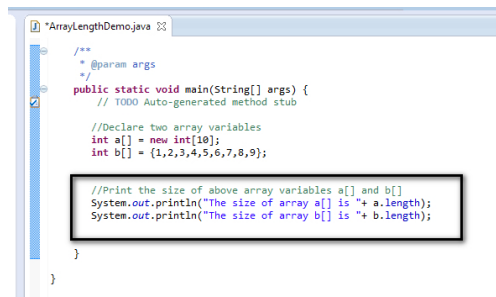
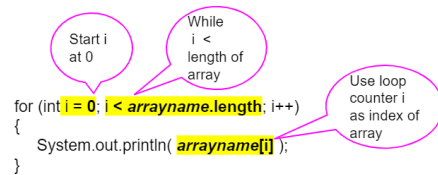


Figure 3: “Exemple d’Affichage de Longueur de Tableaux”

6. Parcours d'un tableau



Vous pouvez parcourir un tableau à l'aide d'une boucle `for` ou d'une boucle `for-each`.

Exemple avec `for` :

```
for (int i = 0; i < nombres.length; i++) {  
    System.out.println(nombres[i]);  
}
```

Exemple avec `for-each` :

```
for (int nombre : nombres) {  
    System.out.println(nombre);  
}
```

7. Tableaux multidimensionnels

Les tableaux peuvent avoir plusieurs dimensions (comme des matrices).

Exemple :

```
int[][] matrice = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9}  
};  
System.out.println(matrice[1][2]); // Affiche 6
```

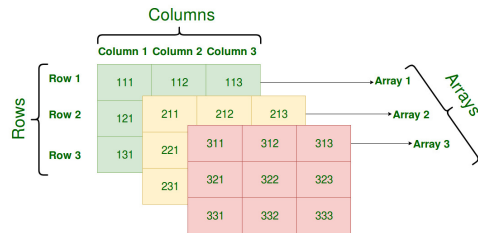
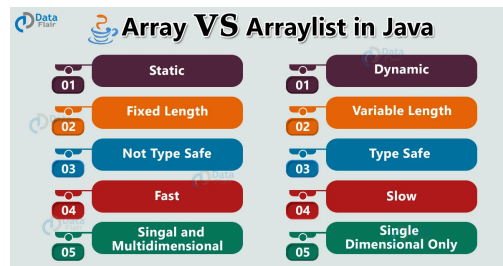


Figure 4: “Exemple de Tableau Tri-Dimensionnel”

8. Limites des tableaux



1. **Taille fixe** : La taille d'un tableau ne peut pas être modifiée après sa création.
2. **Types homogènes** : Tous les éléments doivent être du même type.
3. **Non dynamique** : Pour des structures plus flexibles, utilisez des collections comme ArrayList.

9. Utilisation avancée avec Arrays

La classe utilitaire `java.util.Arrays` propose des méthodes pratiques pour manipuler les tableaux.

Exemple :

```
import java.util.Arrays;

int[] nombres = {5, 2, 8, 1};
Arrays.sort(nombres); // Trie le tableau
System.out.println(Arrays.toString(nombres)); // Affiche [1, 2, 5, 8]
```

A screenshot of a Java code editor window. The window has a title bar with three colored buttons (red, yellow, green) and the text "Java". The code is as follows:

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4
5 public class Main {
6     public static void main(String[] args) {
7         // Sample array
8         Integer[] array = {1, 2, 3, 4, 5};
9
10        List<Integer> list = Arrays.asList(array);
11
12        ArrayList<Integer> arrayList = new ArrayList<>(list);
13
14        System.out.println(arrayList);
15    }
16 }
```

Figure 5: “Example avec Array et ArrayList”