



420-210-MV

Programmation orientée objet

Samuel Fostiné

Les tableaux

- Les tableaux permettent de faire référence à un ensemble de variables de même type par le même nom et d'utiliser un index pour les différencier.
- Un tableau peut avoir une ou plusieurs dimensions.
 - `int[] chiffres` => une dimension
 - `int[][] chiffres` => deux dimensions
- Le premier élément d'un tableau a toujours pour index zéro.
- Le nombre de cases du tableau est spécifié au moment de la création du tableau.
- Le plus grand index d'un tableau est donc égal au nombre de cases moins un.
- Après sa création, les caractéristiques d'un tableau ne peuvent plus être modifiées (nombre de cases, type des éléments stockés dans le tableau)

- La manipulation d'un tableau doit être décomposée en trois étapes :
 - Déclaration d'une variable permettant de manipuler le tableau.
 - Création du tableau (allocation mémoire).
 - Stockage et manipulation des éléments du tableau.

Déclaration du tableau

La déclaration se fait comme une variable classique sauf que l'on doit ajouter à la suite du type de données ou du nom de la variable une paire de crochets `[]`. Il est préférable, pour une meilleure lisibilité du code, d'associer les crochets au type de données. La ligne suivante déclare une variable de type tableau d'entiers.

```
int[] chiffreAffaire;
```

Création du tableau

Après la déclaration de la variable, il faut créer le tableau en obtenant de la mémoire pour stocker ces éléments. C'est à ce moment que nous indiquons la taille du tableau. Les tableaux étant assimilés à des objets, c'est donc l'opérateur `new` qui va être utilisé pour créer une instance du tableau. La valeur fournie par l'opérateur `new` est stockée dans la variable déclarée au préalable.

```
chiffreAffaire=new int[12];
```

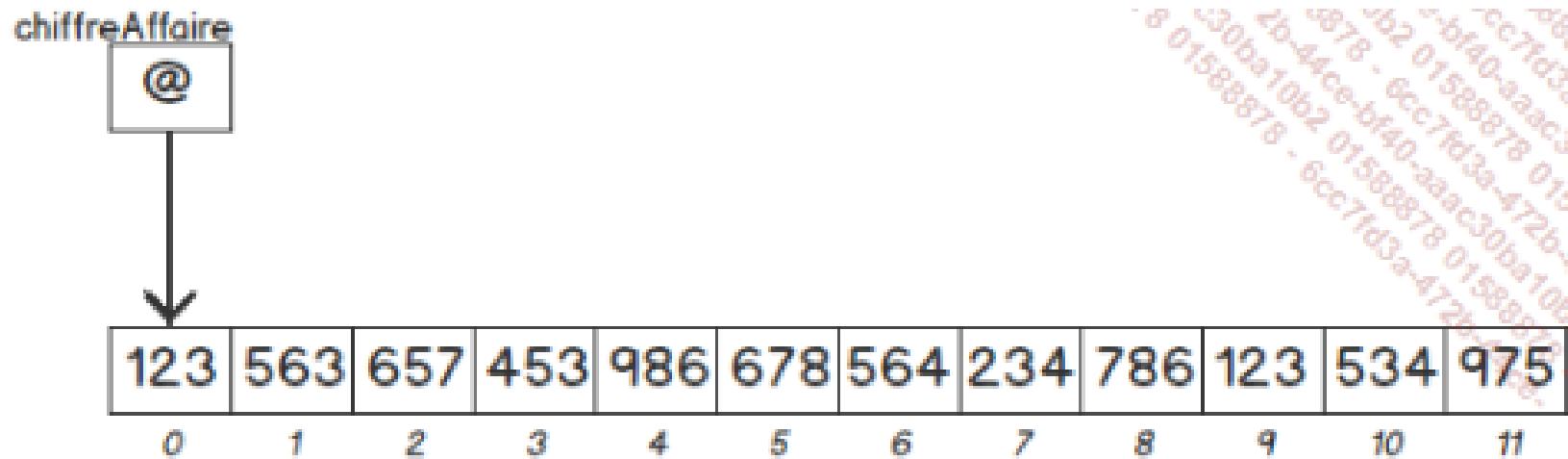
- Cette déclaration va créer un tableau avec douze cases numérotées de 0 à 11. La taille du tableau est définitive, il n'est donc pas possible d'agrandir ou de rétrécir un tableau déjà créé.
- Une autre solution est disponible pour la création d'un tableau. Elle permet la création et l'initialisation de son contenu. La syntaxe est la suivante :

```
int[] chiffreAffaire={123,563,657,453,986,678,564,234,786,123,534,975};
```

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

- Il n'y a, dans ce cas, pas besoin de préciser de taille pour le tableau. Le dimensionnement se fera automatiquement en fonction du nombre de valeurs placées entre les accolades.

Voici la représentation en mémoire que l'on peut faire de ce tableau :



La variable `chiffreAffaire` contient l'adresse mémoire du tableau. Le tableau est composé de 12 cases contiguës numérotées de 0 à 11.

Utilisation du tableau

- Les éléments des tableaux sont accessibles de la même manière qu'une variable classique. Il suffit d'ajouter l'index de l'élément que l'on veut manipuler. L'exemple suivant montre la lecture d'une case dans le tableau :
- `int chiffreAffaireMoisJanvier = chiffreAffaire[0];`
- L'exemple suivant montre la modification du contenu d'une case du tableau :
- `chiffreAffaire[0]=354;`

Attention!!!!

- Il faut être vigilant en manipulant un tableau et ne pas tenter d'accéder à une case du tableau qui n'existe pas, sous peine d'obtenir une exception du type `ArrayIndexOutOfBoundsException`.

- Soit le tableau suivant:

```
int tableau = { 1, 45, 65};
```

La longueur du tableau est 3

Mais si on essaie d'accéder à `tableau[3]`, on obtient une exception puisqu'il n'y a pas d'élément à cette position dans le tableau

Longueur d'un tableau

- Pour connaître le nombre d'éléments d'un tableau, utilisez la propriété `length`

- Ex:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
// Outputs 4
```

- **Boucle dans un tableau**

- Vous pouvez parcourir les éléments du tableau à l'aide de la boucle `for` et utiliser la propriété `length` pour spécifier le nombre de fois que la boucle doit s'exécuter.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
    System.out.println(cars[i]);
}
```

Boucle dans un tableau avec for-each

- Syntaxe

```
for (type variable : nomTableau) {  
    ...  
}
```

- L'exemple suivant affiche tous les éléments du tableau cars, à l'aide d'une boucle « for-each » :

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
for (String i : cars) {  
    System.out.println(i);  
}
```

Tableaux à plusieurs dimensions

- Les tableaux à plusieurs dimensions sont très utilisés pour représenter des tableaux à double entrée (lignes et colonnes).
- Ils permettent cependant de représenter des structures plus complexes à trois dimensions, voire plus.
- Pour représenter la logique d'un tableur, il est possible d'utiliser un tableau à trois dimensions faisant référence aux feuilles, aux lignes et aux colonnes.
- Les tableaux à plusieurs dimensions sont en fait des tableaux contenant d'autres tableaux.

- La représentation suivante montre un tableau à deux dimensions :

chiffreAffaireParRegion

@

0	123	563	657	453	986	678	564	234	786	123	534	975
1	45	412	854	450	658	789	425	802	356	658	500	264

- Ce tableau représente le chiffre d'affaires par région et par mois. La première dimension contient deux cases pour les deux régions où exerce l'entreprise.
- La deuxième dimension contient douze cases pour le chiffre d'affaires mensuel réalisé par l'entreprise en fonction de la région.

- Pour connaître le chiffre d'affaires réalisé en Bretagne en février, l'instruction est la suivante :

```
int CABretagneFevrier = chiffreAffaireParRegion[0][1];
```

- [0] représentant la première case de la première dimension (Bretagne).
 - [1] représentant la seconde case de la seconde dimension (Février).
-
- Pour modifier le chiffre d'affaires réalisé dans les Pays de la Loire en juin, l'instruction est la suivante :

```
chiffreAffaireParRegion[1][5]=1238;
```
 - [1] représentant la seconde case de la première dimension (Pays de la Loire).
 - [5] représentant la sixième case de la seconde dimension (Juin).

La syntaxe de déclaration est semblable à celle d'un tableau, mis à part que l'on doit spécifier autant de paires de crochets qu'il y a de dimensions.

```
int[][] matrice;
```

La création varie quelque peu. Le nombre de cases dans la première dimension doit obligatoirement être mentionné. Le nombre de cases dans les autres dimensions peut être défini plus tard si :

- on ne connaît pas au moment de la création la taille nécessaire ;
- la taille du tableau dans les autres dimensions varie.

Dans la plupart des cas, la taille des différentes dimensions est connue à l'avance et est homogène dans chaque dimension. La création du tableau se fait alors ainsi :

```
matrice=new int[2][3];
```

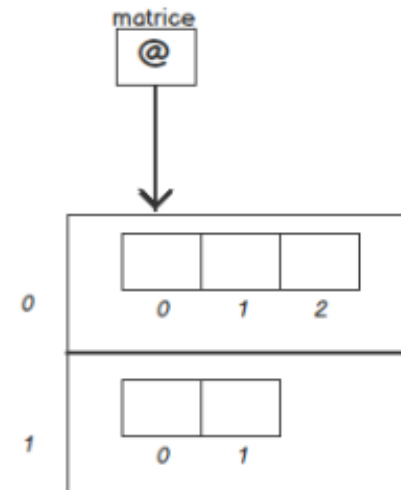

Lorsque ces conditions ne sont pas remplies, il est possible de différer la création des autres dimensions. La première étape consiste à créer la première dimension de cette manière :


```
matrice=new int[2][];
```

Ensuite, pour chaque case de la première dimension, il est possible de créer un tableau correspondant à la deuxième dimension :

```
matrice[0]=new int[3];  
matrice[1]=new int[2];
```

La représentation suivante illustre cette matrice :






La syntaxe permettant l'initialisation d'un tableau à plusieurs dimensions au moment de sa déclaration est un petit peu plus complexe.

```
int[][] matrice={{11,12,13},{21,22,23}};
```

L'utilisation imbriquée d'accolades permet de définir les différentes dimensions. Les valeurs sont situées dans les cases de la dernière dimension. L'exemple ci-dessus permet d'initialiser un tableau à deux dimensions avec deux lignes et trois colonnes.



Boucle à travers un tableau multidimensionnel

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        for (int i = 0; i < myNumbers.length; ++i) {  
            for(int j = 0; j < myNumbers[i].length; ++j) {  
                System.out.println(myNumbers[i][j]);  
            }  
        }  
    }  
}
```