Les tableaux

Les tableaux vont nous permettre 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. 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 d'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 les caractères [et]. Il est préférable, pour une meilleure lisibilité du code, d'associer les caractères [et] 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 simultanément la déclaration de la variable, la création du tableau et l'initialisation de son contenu. La syntaxe est la suivante :

```
int[] chiffreAffaire={1234,563,657,453,986,678,564,234,786,123,534,975};
```

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.

Utilisation du tableau

Les éléments des tableaux sont accessibles de la même manière qu'une variable classique. Il suffit juste d'ajouter l'index de l'élément que l'on veut manipuler.

```
chiffreAffaire[0]=12456;
```

Le contenu d'une case de tableau peut être utilisé exactement de la même façon qu'une variable du même type. 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 ArrayIndexOutOfBoundException.

Tableaux à plusieurs dimensions

Les tableaux à plusieurs dimensions sont en fait des tableaux contenant d'autres tableaux. La syntaxe de déclaration est semblable à celle d'un tableau mis à part que l'on doit spécifier autant de paires de crochets que vous souhaitez avoir de dimensions.

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

La création est également semblable à celle d'un tableau à une dimension hormis que vous devez indiquer une taille pour chacune des dimensions.

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

L'accès à un élément du tableau se fait de manière identique en indiquant les index permettant d'identifier la case du tableau concernée.

```
matrice[0][0]=99;
```

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

```
int[][] grille={{11,12,13},{21,22,23},{31,32,33}};
```

Cet exemple crée un tableau à deux dimensions de trois cases sur trois cases.

La création avec cette technique de tableaux de grande taille à plusieurs dimensions risque d'être périlleuse.

Manipulations courantes avec des tableaux

Lorsque l'on travaille avec les tableaux, certaines opérations doivent être fréquemment réalisées. Ce paragraphe décrit les opérations les plus courantes réalisées sur les tableaux. La plupart d'entre elles sont disponibles grâce à la classe java.util.Arrays fournissant de nombreuses méthodes static de manipulation de tableaux.

Obtenir la taille d'un tableau : il suffit d'utiliser la propriété length du tableau pour connaître le nombre d'éléments qu'il peut contenir. Dans le cas d'un tableau multidimensionnel, il

faut se souvenir qu'il s'agit en fait de tableaux de tableaux. La propriété length indique alors le nombre d'éléments sur la première dimension.

Pour obtenir la même information sur les autres dimensions, il faut utiliser la propriété length de chaque case du tableau de niveau inférieur.

```
matrice=new int[8][3];
System.out.println("le tableau comporte " + matrice.length +
                   " cases sur " + matrice[0].length +
                   " cases");
```

Rechercher un élément dans un tableau : la fonction binarySearch permet d'effectuer une recherche dans un tableau. Elle accepte comme paramètres le tableau dans lequel se fait la recherche et l'élément recherché dans le tableau. La valeur retournée correspond à l'index où l'élément a été trouvé dans le tableau ou une valeur négative si l'élément ne se trouve pas dans le tableau. Pour que cette fonction fonctionne correctement le tableau doit être au préalable trié.

```
int[] chiffreAffaire={1234,563,657,453,986,678,564,234,786,123,534,975};
Arrays.sort(chiffreAffaire);
System.out.println(Arrays.binarySearch(chiffreAffaire, 123));
```

Trier un tableau : la fonction sort assure le tri du tableau qu'elle reçoit en paramètre. Le tri se fait par ordre alphabétique pour les tableaux de chaîne de caractères et par ordre croissant pour les tableaux de valeurs numériques.

```
int[] chiffreAffaire={1234,563,657,453,986,678,564,234,786,123,534,975};
Arrays.sort(chiffreAffaire);
for (int i=0;i<chiffreAffaire.length;i++)
{
    System.out.print(chiffreAffaire[i] + "\t");
}
```

Affiche le résultat suivant :

```
123   234   453   534   563   564   657   678   786   975   986   1234
```

La fonction parallelSort effectue elle aussi le tri du tableau mais en utilisant un algorithme exploitant les capacités d'une machine multiprocesseur.

Afficher un tableau : la fonction toString permet d'obtenir une représentation sous forme d'une chaîne de caractères du tableau passé en paramètre.

```
System.out.println(Arrays.toString(chiffreAffaire));
```

Affiche le résultat suivant :

```
[123, 234, 453, 534, 563, 564, 657, 678, 786, 975, 986, 1234]
```

La fonction deepToString effectue la même opération mais pour un tableau à plusieurs dimensions.

```
int[][] grille={{11,12,13},{21,22,23},{31,32,33}};
System.out.println(Arrays.deepToString(grille));
```

Affiche le résultat suivant :

```
[[11, 12, 13], [21, 22, 23], [31, 32, 33]]
```

Copier un tableau : deux fonctions sont disponibles pour la copie de tableaux.

La fonction copyOf copie un tableau entier avec la possibilité de modifier la taille du tableau. La fonction copyOfRange effectue une copie d'une partie d'un tableau.

```
int[] copieChiffreAffaire;
copieChiffreAffaire=Arrays.copyOf(chiffreAffaire, 24);
System.out.println(Arrays.toString(copieChiffreAffaire));
```

Affiche le résultat suivant :

```
[1234, 563, 657, 453, 986, 678, 564, 234, 786, 123, 534, 975, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
int[] premierTrimestre;
premierTrimestre=Arrays.copyOfRange(chiffreAffaire, 0, 3);
System.out.println(Arrays.toString(premierTrimestre));
```

Affiche le résultat suivant :

```
[1234, 563, 657]
```

Remplir un tableau : la fonction fill est utilisable pour remplir toutes les cases d'un tableau avec la même valeur.