Chapitre 2 : Les tableaux

Table des matières

Un tableau est un objet composé d'une séquence d'éléments numérotés de même type. Ces éléments sont numérotés à partir de 0 et peuvent être référencés par leur numéro grâce à l'opérateur d'index [].

1 les tableaux à une dimension

1.1 déclaration

on peut déclarer des références à des tableaux de deux manière :

```
int[] tabEntiers ; //recommendé
int tabEntiers[] ;
```

tabEntiers est une référence sur un tableau d'entier, mais qui ne référence rien pour le moment.

La déclaration `int[] t1, t2 ;` est équivalentes à `int  t1[], t2[] ;`

1.2 création

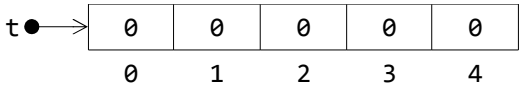
La création effective d'un tableau utilise l'opérateur **new,** on doit préciser le type et nombre d'éléments.
La création d'un tableau avec l'opérateur **new**, initialise chaque élément de ce tableau par une valeur par défaut, cette valeur est :

- **0 :** pour les types numériques (y compris char)
- **false :** pour le type boolean
- **null :** pour les objets et les tableaux. (dans le cas de tableau d'objets ou tableau de tableaux)

Soit les instructions suivantes :

```
int[] t ;
t=new int[5]
```

Cette instruction réserve 5 espaces mémoires de type **int** et retourne l'adresse mémoire dans la référence **t**.



1.3 initialisation

Java utilise l'opérateur d'indexation ([]) pour affecter des valeurs aux éléments d'un tableaux. En général, les indices d'un tableau doivent être le résultat d'une expression qui renvoie des entiers positifs comprise entre 0 et la taille du tableau moins un.

```
int[] t ;
t=new int[3] ;
```

```
x=5 ;
t[3]=-1 ;
t[x]=2 ; //erreur de compilation, x>4
t[(x*2)-8]=4 ; //donc t[2]=4
```

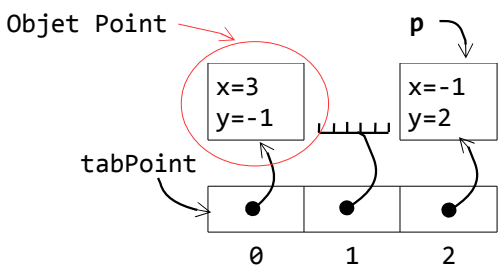
Initialisation avec {...}

Lors de la déclaration d'un tableau, java donne la possibilité initialiser ce tableau avec une liste d'expressions entre accolades, comme dans :

```
int y = 2;
int t[] = {1,y,3};
```

1.4 tableau d'objet

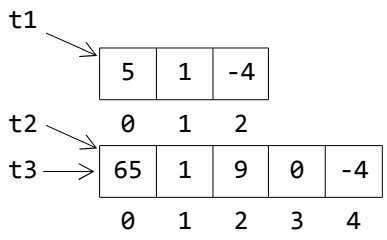
Code tableau de Points	Représentation en mémoire

`Point p=new Point(-1,2) ;`
`Point[] tabPoint =new Point[3] ;`
`tabPoint[0]=new Point(3,-1) ;`
`tabPoint[2]=p ;`
 	
Le code suivant est équivalent :
 	
`Point p=new Point(-1,2) ;`
`Point[] tabPoint ={new Point(3,-1), null, p}`
 	
Chaque élément de **tabPoint** est une référence d'objet de type **Point**
On peut accéder au éléments du tableaux (les objets points), et par exemple, faire appel à la méthode **distance()** pour calculer une distance entre deux points :
 	
`double d=tabPoint[0].distance(tabPoint[2]) ;`
	

1.5 Un tableau est identifié par une référence

L'affectation entre deux variables tableaux de même type, ne réalise pas une copie du tableau :

Code	État de la mémoire

`int [] t1={5,1,-4} ;`
`int [] t2={65,1,9,0,-4};`
 	
`int [] t3 =t2 ;`
	

1.6 parcours d'un tableau à une dimension

Les structures **for** et **while** permettent de parcourir un tableau. Pour cela, la propriété **length** donne le nombre d'éléments d'un tableau

```
String persTab={"ali","oussama","brahim","mohamed"};
for (int i=0; i<persTab.length; i++){
      String nomPersonne=persTab[i];
      //traitement…
}
```

Par ailleurs, deux exceptions peuvent survenir lors du traitement d'un tableau en java :

- **NullPointerException :** si la référence du tableau est **null (table==null)**
- **ArrayIndexOutOfBoundsException :** si l'indice qui parcoure le tableau n'appartient pas à {0,.., table.length -1}

1.7 Le for amélioré ou foreach

Depuis la version 1.5, Java donne la possibilité d'utiliser une syntaxe améliorée du for qui permet un parcours systématique de tous les valeurs d'un tableau. Exemple :

Code	Exécution

`int tab[]={2,43,1,0,3};`
 	
`for (int element : tab){`
`  System.out.println(element);`
`}`
	2
	43
	1
	0
	3

Dans chaque nouvelle itération de la boucle for amélioré, la variable **element** contient une copie de **tab[i]**.
Avec des tableaux contenant des variables primitives, toute modification de **element** n'aura aucun effet sur le contenu du tableau.

2 les tableaux à plusieurs dimensions

En java, les tableaux à plusieurs dimensions sont des tableaux de tableaux

2.1 déclaration et création

Déclaration
Soit t un tableau de deux dimensions d'entier, les trois déclarations suivantes sont équivalentes :

```
int t[][] ;
int[] t[] ;
int[][] t ;
```

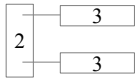
Création
Comme les tableaux à une seule dimension, la création d'un tableau à plusieurs dimension passe

par l'opérateur **new**, en donnant le type des éléments et la taille de chaque dimensions :

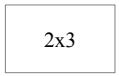
```
int[][] t =new int[2][3] ;
```

Le tableau ainsi crée peut être considéré soit :

- un tableau de **deux (2)** lignes, chacune d'entre elle est un tableau de **trois (3)** entiers.



- un tableau rectangulaire de **deux** lignes et de trois colonnes

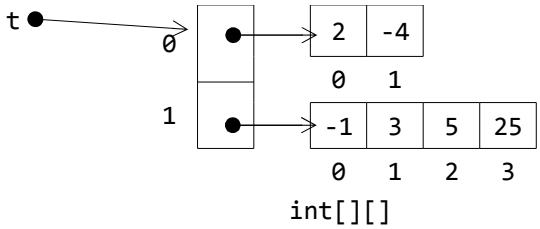


En java, lors de la création d'un tableau à plusieurs dimensions, seule la taille de la première dimension est obligatoire :

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

Cela permettra de donner des tailles différentes pour les tableaux définissant les autres dimensions

Exemple tableau int[][]	Représentation en mémoire

`int[][] t =new int[2][];`
`t[0]=new int[2];`
`t[1]=new int[4];`
 	
`t[0][0]=2; t[0][1]=-4;`
 	
`t[1][0]=-1; t[1][1]=3; t[1][2]=5 ;`
`t[1][3]=25;`
	

Comme pour un tableau à une seule dimension, un tableau à plusieurs dimensions peut être créée comme un ensemble de tableau entre accolades, on peut avoir le même tableau *t* de l'exemple précédent :

```
int[][] t ={{2,-4},{-1,3,5,25}}
```

2.2 Parcours d'un tableau à plusieurs dimensions

Un tableau à plusieurs dimensions se parcours avec autant de boucle imbriquées que de dimensions.

Exemple: La méthode **print()** permet d'afficher les élément d'un tableau d'entier de deux dimensions passé en paramètre :

```
public static void print(int[][] t){
    System.out.print("[");
    for(int i=0;i<t.length;i++){
        if (t[i]!=null){
            System.out.print("[");
            for(int j=0;j<t[i].length;j++){
                System.out.print(t[i][j]+" ");
            }
            System.out.print("]");
        }
```

Appel de la méthode print :

```
int[][] v={{1,2},null,{3,4,5,6},{7,8,9,10},
{11,12}} ;
print(v);
```

Exécution :

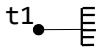
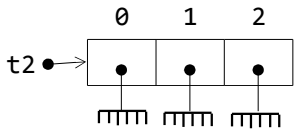
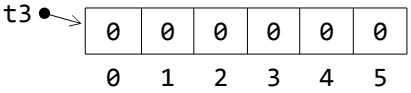
```
[1 2 ]
```

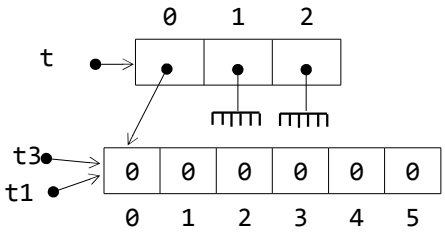
```
        else System.out.print(" -- ");
    }
    System.out.println("]");
}
```

```
--
[3 4 5 6 ]
[7 8 9 10 ]
[11 12 ]
```

2.3 Affectation de référence de tableau à plusieurs dimensions

l'exemple suivant illustre l'affectation entre des tableaux :

Code	Représentation en mémoire
`int[] t1;`	
`int[][] t2 = new int[3][];`	
`t1 = t2;`	Erreur, t2 est tableau de deux dimensions alors de t1 est un tableau de int
`int[] t3 = new int[6];`	

`t1 = t3;  // OK, t3 est un tableau int`
`t2[0] = t1;  // OK, t1 est un tableau int`
	

2.4 Tableau comme argument et comme type de retour d'une méthode

La spécification d'un paramètre tableau se fait en écrivant autant de couples de [] que de dimensions du tableau, mais sans donner la taille de chaque dimension. Cette taille peut être obtenue dans la méthode à l'aide de l'attribut *length*.

2.5 Tableau anonyme

Un tableau dit anonyme est un tableau non affecté à une variable et destiné à ne servir qu'une seule fois. Par exemple, un tableau anonyme peut être créé dans le but de figurer dans l'appel d'une méthode :

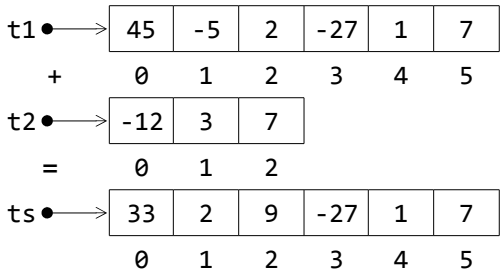
```
public class TestTableauxAnonyme{

 public static void traitementTab(int[] param){
 //...
 }

 public static void main(String[] args){
       traitementTab(new int[]{0,-42,59});//appel d'une méthode avec un tableau anonyme comme argument
  }
}
```

2.6 programme réalisant la somme de deux tableaux

On veut réaliser la somme de deux tableaux comme le montre la figure suivante :




```
public class SommeTableaux{

 public static int[] somme(int[]t1,int[]t2){
     if(t1==null) return t2;
     else if (t2==null) return t1;
     int[] r=new int[Math.max(t1.length,t2.length)];
     for(int i=0;i<Math.min(t1.length,t2.length);i++){
         r[i]=t1[i]+t2[i];
     }
     if (t1.length!=t2.length){
         int[] t=(t1.length>t2.length)?t1:t2;
         for(int i=Math.max(t1.length,t2.length)-1;i>=Math.min(t1.length,t2.length);i--){
             r[i]=t[i];
         }
     }
     return r;
 }

 public static void print(int[] t){
    System.out.print("[");
    for(int i=0;i<t.length;i++){
        if (i==t.length-1) System.out.print(t[i]+"]");
        else System.out.print(t[i]+" ");
    }
    System.out.println();
 }

 public static void print(String message,int[] t){
    System.out.print(message);
    print(t);
 }
```

```
 public static void main(String[] args){
       print("1er exemple, somme de deux tableaux: ",
       somme(new int[]{1955,-4215,599},new int[]{-5471,4752,553,4,5,6,7}));

       print("2eme exemple, somme de trois tableaux: ",
       somme(new int[]{19,55,-42,15,5,99},
       somme(new int[]{155,-15,599},new int[]{55,-45})));
 }
}
```