I. Déclaration

II. Construction

III. Initialisation et Manipulation

IV. Tableaux à plusieurs dimensions

V. Dépassement de bornes

Chapitre III – Les tableaux simples

- I. Déclaration
- II. Construction
- III. Initialisation et Manipulation
- IV. Tableaux à plusieurs dimensions
- V. Dépassement de bornes

Chapitre III – Les tableaux simples

- I. Déclaration
- II. Construction
- III. Initialisation et Manipulation
- IV. Tableaux à plusieurs dimensions
- V. Dépassement de bornes

I. Déclaration

II. Construction

III. Initialisation et Manipulation

IV. Tableaux à plusieurs dimensions

V. Dépassement de bornes

Chapitre III – Les tableaux simples

- I. Déclaration
- II. Construction
- III. Initialisation et Manipulation
- IV. Tableaux à plusieurs dimensions
- V. Dépassement de bornes

I. Déclaration

II. Construction

III. Initialisation et Manipulation

IV. Tableaux à plusieurs dimensions

V. Dépassement de bornes

Chapitre III – Les tableaux simples

- I. Déclaration
- II. Construction
- III. Initialisation et Manipulation
- IV. Tableaux à plusieurs dimensions
- V. Dépassement de bornes

I. Déclaration

II. Construction

III. Initialisation et Manipulation

IV. Tableaux à plusieurs dimensions

V. Dépassement de bornes

Chapitre III – Les tableaux simples

- I. Déclaration
- II. Construction
- III. Initialisation et Manipulation
- IV. Tableaux à plusieurs dimensions
- V. Dépassement de bornes

I. Déclaration

II. Construction

III. Initialisation et Manipulation

IV. Tableaux à plusieurs dimensions

V. Dépassement de bornes

tableau

Un tableau est une collection de valeurs qui sont tous de même type (primitif ou objet).
En Java on distingue la déclaration, la construction et l'initialisation d'un tableau.
Dans ce chapitre on ne verra que des tableaux d'entiers.

I. Déclaration
II. Construction
III. Initialisation et Manipulation
IV. Tableaux à plusieurs dimensions
V. Dépassement de bornes

Déclaration

La syntaxe de la déclaration d'un tableau nomTableau contenant des éléments de type `typeElement` est :

`typeElement` [ ] nomTableau ;

Construction

La syntaxe de construction d'un tableau nomTableau déjà déclaré est :

nomTableau = `new typeElement [nbreElement]` ;

`nbreElement` correspond au nombre d'éléments de nomTableau. C'est *une valeur de type entier qui doit pouvoir être évaluée au moment de l'exécution*.

Cette construction correspond à l'allocation mémoire des `nbreElement` espaces mémoire pour contenir les éléments du tableau. Une fois cette dimension fixée *elle ne peut pas être modifiée*. A ce moment, nomTableau est une référence à ces espaces mémoire ; de plus chacun de ces espaces mémoire est initialisée selon `typeElement` : soit à `null` pour un type objet, soit 0 pour un type entier et 0.0 pour un type flottant.

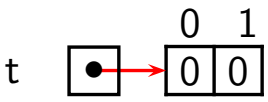
numérotation

Un tableau est **toujours numéroté depuis 0**.
Tout tableau a un attribut `length` permettant de connaître sa longueur soit le nombre d'éléments qu'il contient. Si `tab` est l'identificateur du tableau `tab.length` sera le nombre d'éléments de `tab`.

Exemple

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

peut s'illustrer ainsi



Initialisation et Manipulation

On peut regrouper ces différentes étapes de plusieurs façons :

- `int[ ] t ={1,2,3};` : cela définit directement un tableau de 3 entiers contenant 1, 2, 3.
-
  - `char[] v ;`
  - `v = new char[2];`
  - `v[0]='o'; v[1]='h';`

Dans l'exemple suivant, on va utiliser la classe `Random` de `java.util.` pour créer une méthode qui initialise un tableau d'entiers par des valeurs engendrées au hasard.

Exemple

```
static void initTableau(int [] t){
Random randomGenerator = new Random() ;
int randomInt = randomGenerator.nextInt(100);
for (int i=0 ;i<t.length ; i++){
t[i]=randomInt ;//t[i] appartient à [0,99]
randomInt = randomGenerator.nextInt(100); }
}
```

Tableaux à plusieurs dimensions

Il est possible de manipuler des tableaux de tableaux. Par exemple,

```
int [][] tab ;
t =new int [2][3];
for (int i=0 ;i<tab.length ; i++){
for (int j=0 ;j<tab[i].length ; j++){ t[i][j] = i+j ;
}}
for (int i=0 ;i<t.length ; i++){
for (int j=0 ;j<tab[i].length ; j++){
System.out.print(tab[i][j] + " ");}
System.out.println() ;
}
```

donne comme affichage
0 1 2
1 2 3

On peut aussi avoir des lignes de taille différente par exemple

```
int [][] tab ;
t =new int [3][];
for (int i=0 ;i<t.length ; i++){
t[i] = new t[i+1] ;
for (int j=0 ;j<=i ; j++){ t[i][j] = i+j ;
}}
for (int i=0 ;i<t.length ; i++){
for (int j=0 ;j<t[i].length ; j++){
System.out.print(t[i][j] + " ");}
System.out.println() ;
}
```

donne comme affichage

0
1 2
2 3 4

Dépassement de bornes

Comme dans tout langage, en Java, faire appel à un élément t[i] où i n'appartient pas à l'intervalle [0... t.length-1] provoque une erreur à l'exécution.
Pour ne pas interrompre un programme dans lequel une telle erreur puisse être commise on peut utiliser un mécanisme de gestion des exceptions ; c'est donc un premier exemple d'introduction à cette notion.

Exemple d'exception

```
int [] t ;
t =new int [3];
for (int i=0 ;i<t.length ; i++){
t[i] = i+1;}
for (int i=0 ;i<t.length ; i++){
System.out.print(t[i]);}
try { t[3] = 4 ;
}
catch (ArrayIndexOutOfBoundsException e) {
System.out.println("erreur interceptée ");
}
```

donne l'affichage suivant

1 2 3

erreur interceptée