## 

## Les tableaux 2D ou Matrices

Les tableaux Java peuvent comporter plusieurs dimensions. Il est ainsi possible de créer des tableaux rectangulaires, cubiques, ou à un nombre quelconque de dimensions. Par contre, pour le cours de 201, nous nous contenterons d’étudier les tableaux à deux dimensions, plus communément appelés matrice. Pour déclarer un tableau 2D, vous devez utiliser la syntaxe suivante :

int[][] x;

L'initialisation du tableau peut être effectuée à l'aide de tableaux littéraux :

int[][] x = {{1, 2, 3, 4},{5, 6, 7, 8}};

ou à l'aide de l'opérateur **new** :

//Ici, il n’y a pas d’affectation!!!  
int[][] x = new int[2][4];

Ces exemples mettent en évidence la véritable nature des tableaux 2D. Il s'agit en réalité de tableaux de tableaux. En fait, la syntaxe :

int[][] x = new int[3][2];

crée un tableau de 3 tableaux. Ces 3 tableaux ont tous 2 éléments.

### Un exemple pour illustrer ce dernier concept

A) Si on a la déclaration et l’initialisation suivantes :

int[] tab = {12, 8, -3} ;

On se trouve à déclarer et à initialiser une variable tab. Cette variable est de type int[]. Ceci signifie donc que la variable tab est une référence vers un objet de type tableau de int.



Or, lors de la création du tableau tab, il nous est possible de placer un tableau dans chaque position du tableau, au lieu d’y placer les entiers 12, 8 et -3.

int[] t1 = new int[2];

int[] t2 = new int[2];

int[] t3 = new int[2];

int[][] tab = {t1, t2, t3};

En effet, puisqu’un tableau est représenté, tout comme n’importe quel autre objet, par une variable, qui est en fait une référence à un objet en mémoire. Nous pouvons donc placer les références t1, t2, t3 vers leur tableau respectif dans notre tableau tab.   
On se trouve donc à déclarer et à initialiser une variable tab. Cette variable est de type int[]. Ceci signifie donc que la variable tab est une référence vers un objet de type tableau de référence à des objets de type int[].

On peut donc dire que la variable tab est donc une référence à un tableau de tableaux de int. On peut donc déclarer la variable tab de façon plus concise en utilisant la syntaxe suivante :

int[][] tab = new int[3][2];

Ceci signifie donc bien que tab est un tableau de 3 tableaux de 2 int chacun.

Références:

**tab**

objet

tableau

**t1**

**t2**

**t3**

objet

t2

objet

t3

objet

t1

## Manipulation des tableaux

Habituellement, on en fait une représentation plus simplifiée d’un tableau 2D, qui en facilite la compréhension car elle est plus près de notre réalité. Donc, pour l’exemple précédent, on aurait un tableau de la forme suivante avec la déclaration et l’allocation de mémoire :

int[][] tab = new int[3][2];

tab

C’est-à-dire un tableau de 3 lignes et 2 colonnes !

Pour accéder aux éléments du tableau, on utilise leurs indices :



Remarquez bien que les indices, aussi bien pour les lignes que les colonnes, commencent à zéro tout comme pour les vecteurs.



Donc, on peut faire l’affectation du tableau précédent en utilisant les 6 affectations suivantes (une pour chaque élément) :

tab[0][0] = 4;  
tab[0][1] = 8;  
tab[1][0] = 2;  
tab[1][1] = 3;  
tab[2][0] = 0;  
tab[2][1] = 5;

Évidemment, cette façon de procéder est impensable si on a une matrice de 20 par 20, cela nous demanderait de faire 400 affectations! Il existe une façon de procéder plus générique, on procédera comme pour les vecteurs, non pas avec une, mais bien avec deux bonnes vieilles boucles **POUR** imbriquées…

Pour accéder à tous les éléments du tableau, vous devez utiliser des boucles imbriquées. Une boucle (celle à l’intérieure) vous permettra **de parcourir chacune des colonnes d’une ligne**. Une autre boucle (celle à l’extérieure) vous permettra **de parcourir chacune des lignes** de notre tableau.

int[][] tab = new int[3][2];

int nbLignes = 3;

int nbColonnes = 2;

//Traitement de toutes les lignes

for (int i = 0; i < nbLignes; i++)

{

//Traitement de tous les éléments d’une ligne, soit de chacune de ses colonnes

for (int j = 0; j < nbColonnes; j++)

{

x[i][j] = 5;

}

}

On peut également ne pas utiliser les variables nbLignes et nbColonnes et plutôt utiliser **length**. Cela nous permet de ne pas créer des variables inutilement.

int[][] tab = new int[3][2];

//Traitement de toutes les lignes

for (int i = 0; i < tab.length; i++)

{

//Traitement de tous les éléments d’une ligne

for (int j = 0; j < tab[i].length; j++)

{

x[i][j] = 5;

}

}

Ce qui nous donnerait, dans les deux cas, le tableau suivant :

# 