

# Les tableaux à plusieurs dimensions en Java

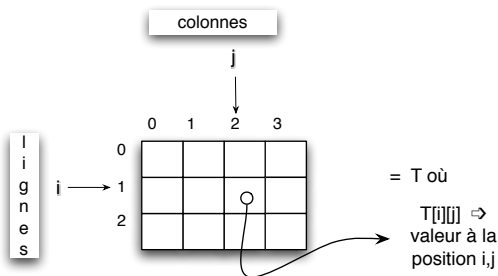
Maria Virginia Aponte

CNAM-Paris

23 octobre 2020

# Tableau à deux dimensions ou matrices

- vu comme une grille composée de lignes et de colonnes,
- chaque élément est désigné par sa position dans cette grille : (numéro de ligne, numéro de colonne),
- Si  $T$  est un tableau à deux dimensions, l'élément à la ligne  $i$  et colonne  $j$  est donné par  $T[i][j]$ .



# Déclaration d'un tableau à plusieurs dimensions

En Java, un tableau de **n dimensions** et composantes de type `TyBase` est déclaré par :

```
TyBase [] []...[] tab; // n fois le symbole []
```

Chaque occurrence du symbole `[]` permet d'obtenir une dimension supplémentaire :

```
int [] t;           // 1 dimension  
int [] [] m;       // 2 dimensions  
char [] [] [] p;   // 3 dimensions
```

## Création et initialisation avec `new`

- création avec `new`, en donnant la **taille** de chacune des dimensions,
- toutes les composantes sont initialisées avec des valeurs par défaut.

```
int [][] T=new int [3][4]; //creation avec 3 lignes
                             //et 4 colonnes
T[1][2]= 7;    // modification composante (1,2)
```

	0	1	2	3
0	0	0	0	0
1	0	0	7	0
2	0	0	0	0

T[0][0] T[0][1] T[0][2] T[0][3]

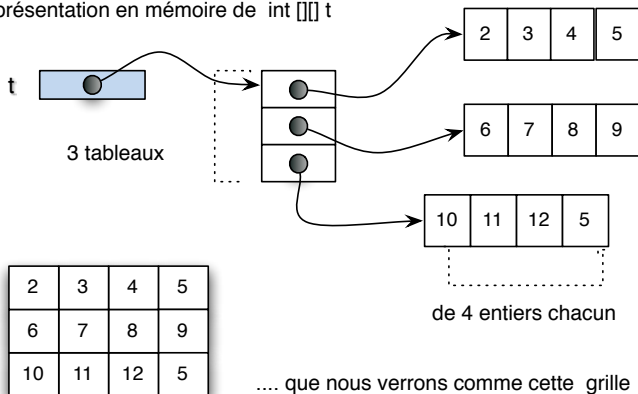
T[1][0] T[1][1] T[1][2] T[1][3]

T[2][0] T[2][1] T[2][2] T[2][3]

# Représentation de matrices en mémoire

- En Java, une matrice est en réalité **un tableau de tableaux**.
- **Exemple** : `int[][] t = new int[3][4]` est formé de :
  - ▶ 3 tableaux de `int`,
  - ▶ où chacun de ces 3 tableaux a 4 composants de type `int`.

Représentation en mémoire de `int [][] t`

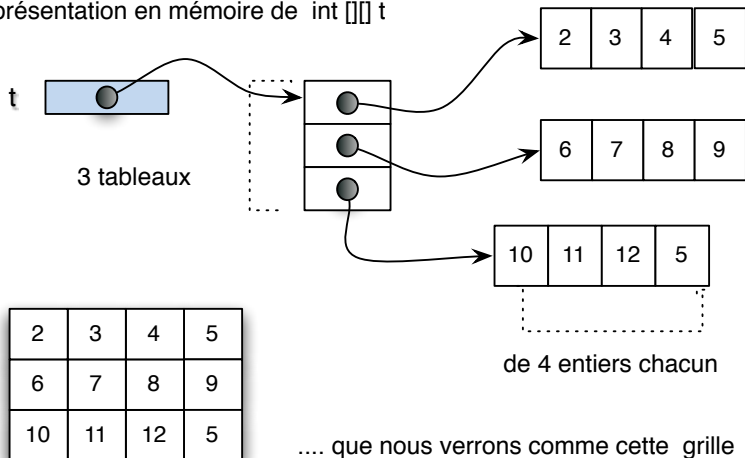


.... que nous verrons comme cette grille

## Représentation en mémoire (2)

Souvent il nous suffira de penser aux matrices comme des grilles.

Représentation en mémoire de `int [][] t`



# Longueur d'une dimension

Si `t` est une matrice :

- `t.length` : donne la longueur de la première dimension (nombre de lignes du tableau).
- `t[i].length` : donne la longueur de la ligne `i` de `t`, autrement dit, le nombre de colonnes de cette ligne.

---

```
int [][] t = new int [3][4];           // 3 lignes, 5 col  
Terminal.afficheIntln(t.length);      // affiche 3  
Terminal.afficheIntln(t[1].length);   // affiche 4
```

---

# Création et initialisation par une liste de tableaux

- Une matrice est un tableau dont les composantes sont elles-mêmes des tableaux.
- On peut donc créer une matrice en donnant la liste de ses composantes, à savoir, la liste de tous les tableaux correspondant à ses composantes.

Exemple : une matrice de 3 lignes et 4 colonnes pourra être créée par une liste de 3 tableaux (un pour chaque ligne).

Chacun des 3 tableaux sera composé de 4 éléments (c'est le nombre de colonnes).



## Création et initialisation par une liste de tableaux (2)

```
int [][] tab = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
int [] t = tab[1];
for (int j = 0; j<= t.length-1; j++) {
    Terminal.ecrireInt(t[j]);
}
Terminal.sautDeLigne();
```

Le programme extrait la composante `tab[1]` de la matrice : c'est la ligne 1 de `tab` **qui est un tableau** de 4 entiers. Ce tableau est mis dans la variable `t` qui est ensuite affiché :

```
Java/Essais> java Test
5678
```

- Réalisé en général avec 2 boucles imbriquées :
  - ▶ une boucle externe pour parcourir les lignes pour des indices compris entre 0 et `mat.length-1` ;
  - ▶ une boucle interne qui, pour chaque ligne, fait le parcours des éléments de toutes les colonnes de cette ligne. Les indices des colonnes seront alors compris entre 0 et `mat[i].length-1`.

## Parcours de matrices (2)

```
int n,m;
Terminal.ecrireString("Nombre_de_lignes?_");
n = Terminal.lireInt();
Terminal.ecrireString("Nombre_de_colonnes?_");
m = Terminal.lireInt();
int [][] mat = new int [n][m];
// Initialisation
for (int i=0; i<= mat.length -1; i++) {
    for (int j=0; j<= mat[i].length -1; j++) {
        Terminal.ecrireString("Element_( " + i + ",_ " + j +
        mat[i][j] = Terminal.lireInt();
    }
}
```

## Parcours des matrices (3)

Ce programme affiche :

```
Java/Essais> java initMatrice
Nombre de lignes? 2
Nombre de colonnes? 3
Element (0, 0)? 1
Element (0, 1)? 2
Element (0, 2)? 3
Element (1, 0)? 4
Element (1, 1)? 5
Element (1, 2)? 6
```

# Exemple : gestion de notes d'une classe

**Problème** : Gestion de plusieurs notes par élève, pour tous les élèves d'une classe.

**Solution** : Initialiser une matrice de  $n$  élèves avec  $m$  notes par élève, puis calculer dans un tableau de taille  $n$ , la moyenne de chaque élève.

# Initialisation

Toutes les notes de l'élève  $i$  se trouvent à la ligne  $i$  de la matrice `notes`, alors que sa moyenne est dans `moyennes[i]`.

```
int n,m;
Terminal.ecrireString("Nombre_d'eleves?_");
n = Terminal.lireInt();
Terminal.ecrireString("Nombre_de_notes_par_eleve?_");
m = Terminal.lireInt();
double [][]notes= new double[n][m]; // les notes
double []moyennes= new double[n]; // les moyennes
// Initialisation
for (int i=0; i<= notes.length -1; i++) {
    Terminal.ecrireStringln("Notes_eleve_" + (i+1) + "?");
    for (int j=0; j<= notes[i].length -1; j++) {
        Terminal.ecrireString("_Note_" + (j+1) + "?_");
        notes[i][j] = Terminal.lireDouble();
    }
}
```

# Calcul des moyennes

---

```
// Calcul des moyennes
for (int i=0; i<= notes.length -1; i++) {
    for (int j=0; j<= notes[i].length -1; j++) {
        moyennes[i] = moyennes[i] + notes[i][j];
    }
    moyennes[i] = moyennes[i]/notes[i].length;
}
//Affichages
for (int i=0; i<= moyennes.length -1; i++) {
    Terminal.ecrireString("Moyenne_eleve_" + (i+1) + "= ");
    Terminal.ecrireDoubleln(moyennes[i]);
}
```

---

# Affichages

```
Java/Essais> java matriceNotes
Nombre d'eleves? 3
Nombre de notes par eleve? 2
Notes pour l'eleve 1?
  Note 1? 2
  Note 2? 2
Notes pour l'eleve 2?
  Note 1? 6
  Note 2? 17
Notes pour l'eleve 3?
  Note 1? 10
  Note 2? 15
Moyenne de l'eleve 1= 2.0
Moyenne de l'eleve 2= 11.5
Moyenne de l'eleve 3= 12.5
```



# Exemple d'algorithme sur tableaux : tri par sélection

**Problème** : Trier **en place** un tableau T d'entiers (ordre croissant).

- **trier en place** un tableau, signifie que l'on ne s'autorise pas à prendre un 2ème tableau afin de faire le tri.
- Cela équivaudrait à **dupliquer** l'espace utilisé (inefficace pour les grands tableaux, ex : bases des données).

**Question** : en quoi prendre un 2ème tableau simplifie le travail de tri ? Quel serait l'algorithme dans ce cas là ?

## Solution avec deux tableaux (interdite !)

- On se donne 2 tableaux :  $T$  (tableau d'origine),  $res$  (pour le résultat triée).
- On sélectionne l'élément le plus petit de  $T$ . Supposons qu'il se trouve à l'indice  $i$  :
  - ▶ on recopie  $T[i]$  dans  $res[0]$ ,
  - ▶ on remplace  $T[i]$  par une valeur  $V$  qui est plus grande que n'importe élément de  $T$ .
- On recommence en sélectionnant dans  $T$  à nouveau le plus petit élément que l'on recopie à la deuxième place de  $res$ .
- On continue jusqu'à avoir sélectionné et recopié  $N$  éléments où  $N$  est la taille de  $T$ .

# Solution en place

- On se donne 2 tableaux :  $T$  (tableau d'origine),  $res$  (pour le résultat triée).
- On sélectionne l'élément le plus petit de  $T$ . Supposons qu'il se trouve à l'indice  $i$  :
  - ▶ on recopie  $T[i]$  dans  $res[0]$ ,
  - ▶ on remplace  $T[i]$  par une valeur  $V$  qui est plus grande que n'importe élément de  $T$ .
- On recommence en sélectionnant dans  $T$  à nouveau le plus petit élément que l'on recopie à la deuxième place de  $res$ .
- On continue jusqu'à avoir sélectionné et recopié  $N$  éléments où  $N$  est la taille de  $T$ .