Les tableaux en Java

Maria Virginia Aponte

CNAM-Paris

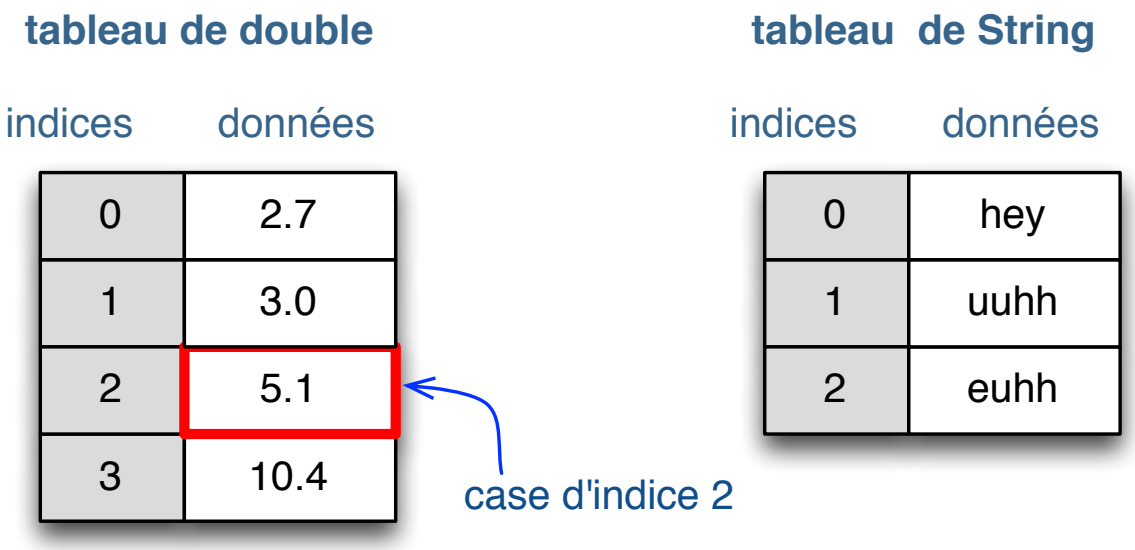
25 octobre 2017

Tableaux : qu'est-ce que c'est ?

Tableau ≈ Structure des données

Regroupement de données indexées et d'un même type.
Une donnée est une composante (ou case).

- on peut manipuler le tableau comme un tout ;
- et manipuler séparément chaque composante.



Tableaux : pourquoi faire ?

Traiter des grandes quantités de données :

- de manière uniforme (sur toute composante),
- compacte et rapide (en temps d'accès aux composantes).

Au lieu de 100 variables déclarées séparément :

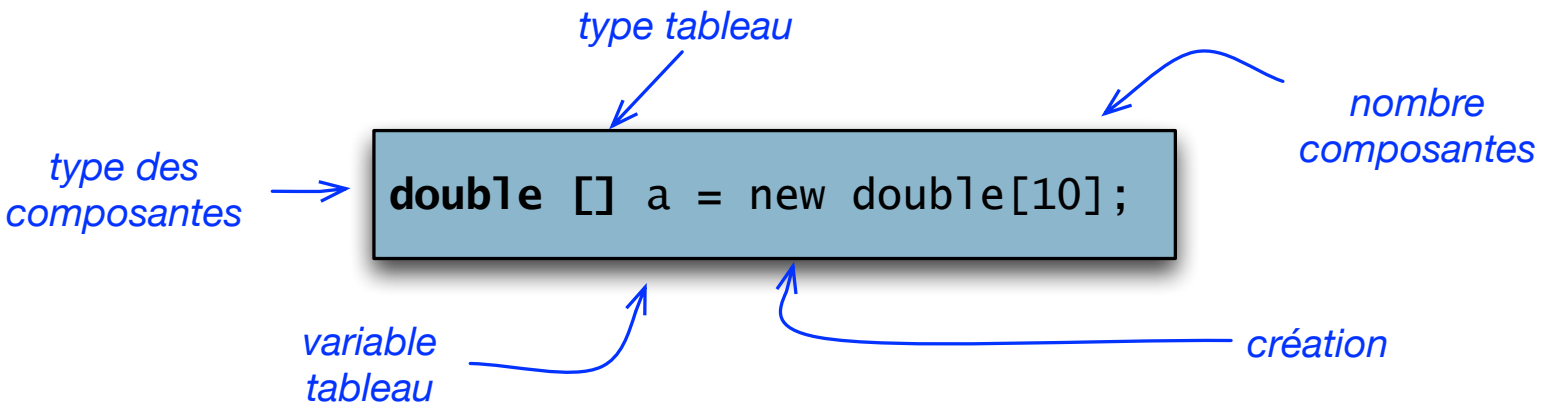
```
double a0 = 2.0;
double a1 = 5.3;
....
double a99 = -10.8;
```

une unique variable tableau a ⇒ traitements sur ses composantes :

```
double [] a = new double[100];  // une seule declaration
a[0] = 2.0;  // traitement composante d'indice 0
a[1] = 5.3;
....
```

Avant d'utiliser un tableau

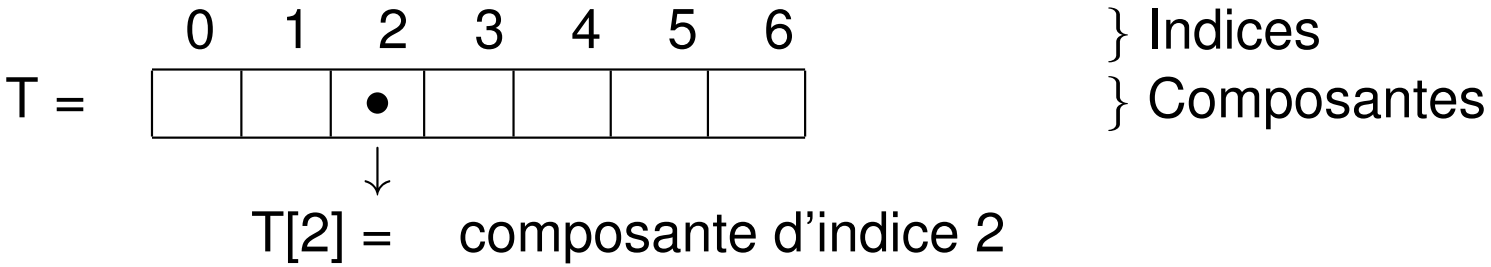
1. Déclarer une variable de type tableau (`[]`).
   - `double [] t` déclare un tableau de double.
2. Créer explicitement ses composantes en mémoire :
   - opération `new` avec nombre + type de composantes
   - `new double [10]` 10 composantes crées en mémoire
3. Initialiser les valeurs des composantes (par défaut ou explicitement).



Composantes d'un tableau

Chaque composante du tableau T :

- est désignée via son indice $i \Rightarrow$ `T[i]`,
- i correspond à la position (à partir de 0) de la case dans le tableau,
- `T[i]` peut être affectée individuellement.



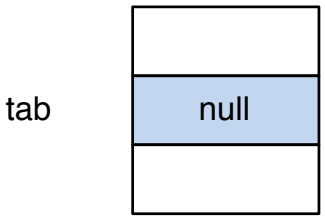
T[0], T[1], T[2], T[3], . . . , T[6] } 7 variables (cases)

En détail : déclarer un tableau

Syntaxe : `Type [] tab;`

- après cette déclaration, `tab` contient la valeur `null`.
- ET, `tab` ne possède aucune composante ;
- impossible d'accèder aux composantes, ou obtenir longueur du tableau (erreur à l'exécution).

```
int [] tab;      // variable tableau d'entiers
tab[0] = 2;      // erreur (NullPointerException)
```



En détail : création des composantes d'un tableau

Syntaxe :

```
new T[n];   // n (nbe composantes)
            // T (type composantes)
```

Cet opération se traduit par :

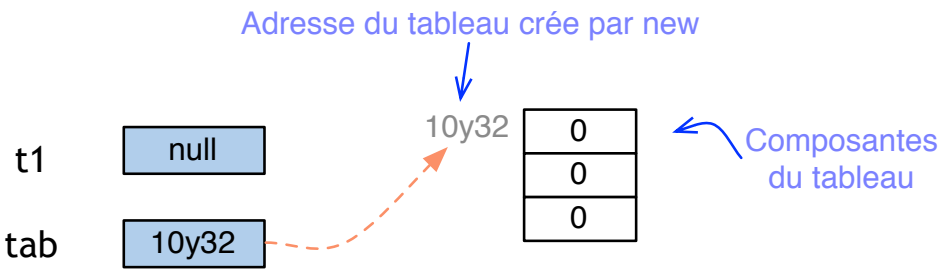
1. En mémoire : espace reservé pour n composantes de type T.
2. Initialisation des composantes du tableau avec valeurs par défaut (0 pour les types numériques, false pour bool, etc.)

Création des composantes (exemple)

```
int [] t1;
int [] tab;          // declaration
tab = new int[3]; // creation + affectation
System.pout.print(tab[0]);  // affiche 0
```

Après l'affectation `tab = new int[3]`

- `tab` : contient l'adresse d'un espace mémoire avec 3 composantes `int` initialisées à 0.
- la taille de cet espace est non modifiable ⇒ tableaux de taille fixe.



Valeurs par défaut via `new`

Les valeurs par défaut données par `new` (selon le type des composantes) :

- composantes `boolean` ⇒ initialisées à `false`.
- composantes numériques ⇒ initialisées à 0.
- composantes `char` ⇒ initialisées au caractère nul (`'\0'`)
- composantes de type *référence* ⇒ initialisées à `null` (pointeur nul).

Initialisation des composantes

- on peut initialiser avec valeurs par défaut, via `new`

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

- on peut initialiser en donnant une liste de valeurs :

```
int [] tab = {1,9,2};
```

- ou, par affectation de chaque composante :

```
int [] tab = new int [3];
tab[0] = 1;
tab[1] = 9;
tab[2] = 2;
```

Que contient une variable tableau ?

```
int []  t;        // variable t : tableau de int
```

- si t n'est pas affecté :
  - t contient la valeur `null` $\Rightarrow$ ne possède aucune composante ;
  - tout accès `t[i]` $\Rightarrow$ erreur fatale (`NullPointerException`)
- si t est affectée par :
  - une valeur de type tableau (de int),
  - ou par une opération de création de composantes (`new`) :
    - ★ tout accès `t[i]` (dans les bornes de t) réussit
    - ★ t contient l'adresse mémoire où sont stockées ses composantes.

Taille d'un tableau

Taille du tableau `t`

C'est le nombre de composantes de `t`.

- donné par : `t.lenght`
- Indices de `t` : entre 0 et `t.length-1`.

Attention : la taille d'un tableau peut-être 0.

```
/* Exemples    */
   int [] t = new int[3];              // taille 3
   Terminal.ecrireInt(t.length);       // affiche 3
   double [] m = new double[0];           // taille 0
   Terminal.ecrireInt(m.length);       // affiche 0
```

Accès en dehors des bornes du tableau

- l'accès `t[i]` est défini uniquement pour $i \in [0, \ldots, \texttt{t.length} - 1]$.
- en dehors, composante indéfinie :
  - $\Rightarrow$ erreur à l'exécution
  - nom de l'erreur (*exception)* : `ArrayIndexOutOfBoundsException`.

Exemple d'accès en dehors des bornes

```
public static void main (String args[]) {
  double [] tab = {1.0, 2.5, 7.2, 0.6};
  Terminal.ecrireString("tab[0] avant = ");
  Terminal.ecrireDoubleln(tab[0]);
  tab[0] = tab[0] + 4;
  Terminal.ecrireString("tab[0] apres = ");
  Terminal.ecrireDoubleln(tab[0]);
  tab[5] = 17; // Erreur: indice en dehors des bornes
```

```
Java/Essais> java Test
tab[0] avant = 1.0
tab[0] apres = 5.0
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 5
```

Schéma typique de parcours (il y en d'autres !)

Pour travailler avec un tableau : utiliser des boucles !

Boucle de parcours du tableau t

Permet de « visiter » les composantes en faisant varier leur indice.
Faire varier une variable *i* qui servira d'indice :

- *i* varie dans l'intervalle [0..`t.lenght` − 1].
- traiter chaque composante `t[i]`

```
for (int i=0; i< t.length; i++){
      actions sur t[i]
}
```

Les boucles `for` sont en général bien adaptées.

Exemple 1 : parcours + affichage d'un tableau

```
public class AfficheTab {
 public static void main (String args[]) {
    int[] tab = {10,20,30,40};
    for (int i=0; i<= tab.length -1; i++)  {
       Terminal.ecrireStringln("tab["+i+ "] = "+ tab[i]);
    }
}}
```

```
Java/Essais> java AfficheTab
tab[0] =  10
tab[1] =  20
tab[2] =  30
tab[3] =  40
```

Attention aux bornes de l'indice

- Erreur commune : fixer le dernier indice à `tab.length`,
- produit une erreur : cette composante (4ème ici), n'existe pas dans le tableau.

```
Java/Essais> java AfficheTabErr
tab[0] =  10
tab[1] =  20
tab[2] =  30
tab[3] =  40
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 4
```

Exemple 2 : Initialisation notes lues+ affichage

Problème : initialiser un tableau avec des notes lues au clavier.

```
Java/Essais> java Notes
Nombre de notes a lire? 4
Note no. 1? 7.6
Note no. 2? 11
Note no. 3? 14
Note no. 4? 5

Notes dans le tableau:
**********************
Note no. 1 = 7.6
Note no. 2 = 11.0
Note no. 3 = 14.0
Note no. 4 = 5.0
```

Initialisation notes lues + affichage (2)

Solution :

1. Demander le nombre N de notes à lire ;
2. Créér un tableau `notes` de cette taille ;
3. Une première boucle initialise le tableau ;
4. la boucle suivante affiche son contenu.
5. Les itérations se font de `i=0` jusqu'à `i <= notes.length-1`.

Initialisation notes lues + affichage (3)

```
    Terminal.ecrireString("Nombre de notes a lire? ");
    int N = Terminal.lireInt();
    double [] notes = new double[N];
    // Initialisation
    for (int i=0; i< notes.length; i++)  {
          Terminal.ecrireString("Note no. "+(i+1)+"? ");
          notes[i] = Terminal.lireDouble();
    }
    // Affichage
    Terminal.sautDeLigne();
    Terminal.ecrireStringln("Notes dans le tableau:");
    Terminal.ecrireStringln("**********************");
    for (int i=0; i< notes.length; i++)  {
         Terminal.ecrireString("Note no. " + (i+1) + " = ");
         Terminal.ecrireDoubleln(notes[i]);
    }
}}
```

Exemple 3 : Recherche des min/max d'un tableau (1)

Problème : Afficher les minimum et le maximum d'un tableau.

Solution :

- Deux variables `min` et `max` initialisées avec le premier élément du tableau,
- La boucle compare chaque élément avec min et max : si un élément est plus petit que le min ou plus grand que le max, leurs valeurs sont modifiées.
- La comparaison se fait à partir du deuxième élément (pourquoi ?) $\Rightarrow$ i débute à `i=1`.

min/max d'un tableau (2)

```
Terminal.ecrireString("Combien␣de␣nombres?␣");
int n = Terminal.lireInt();
int [] tab = new int[n];
// Initialisation par lecture de composantes
for (int i=0; i< tab.length; i++)  {
     Terminal.ecrireString("Composante␣"+(i+1)+"?␣");
     tab[i] = Terminal.lireInt();
}
// Recherche de min et max
// min et max initialises au premier du tableau
int min = tab[0]; int  max = tab[0];
// Comparaison a partir de i=1
for (int i=1; i<= tab.length -1; i++)  {
     if (tab[i] < min) { min = tab[i];}
     if (tab[i] > max) { max = tab[i];}
}
Terminal.ecrireStringln("Le␣minimum␣est:␣" + min);
Terminal.ecrireStringln("Le␣maximum␣est:␣" + max);
```

min/max d'un tableau (3)

```
Java/Essais> java MinMax
Combien des nombres? 5
Composante 1? 7
Composante 2? 0
Composante 3? -2
Composante 4? 67
Composante 5? 3
Le minimum est: -2
Le maximum est: 67
Java/Essais>
```

Exemple 4 : Moyenne de notes

Problème : Calculer et afficher la moyenne des notes, les notes maximale et minimale d'un tableau de notes.

Solution : adaptation code d'initialisation, et de min/max.

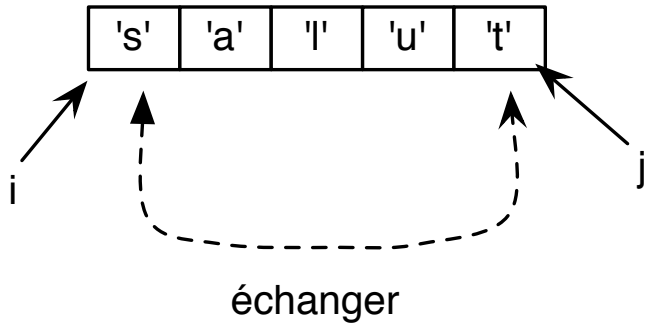
```
Java/Essais> java Notes
Nombre de notes a` lire? 4
Note no. 1? 5
Note no. 2? 8
Note no. 3? 10
Note no. 4? 15
La moyenne des notes est: 9.5
Le nombre de notes >= 10 est: 2
La note minimum est: 5.0
La note  maximum est: 15.0
```

Moyenne de notes (2)

```
Terminal.ecrireString("Nombre de notes a lire? ");
int nbeNotes = Terminal.lireInt();
double [] notes = new double[nbeNotes];
for (int i=0; i<= notes.length -1; i++)  {
     Terminal.ecrireString("Note no. "+(i+1)+"? ");
     notes[i] = Terminal.lireDouble();
}
double min = notes[0]; double max = notes[0];
double somme = 0; int sup10 = 0;
for (int i=0; i<= notes.length -1; i++)  {
    if (notes[i] < min) { min = notes[i];}
    if (notes[i] > max) { max = notes[i];}
    if (notes[i] >= 10) { sup10++;}
    somme = somme + notes[i];
}
Terminal.ecrireStringln("Moyenne= "+ somme/nbeNotes);
Terminal.ecrireStringln("Nombre de notes >= 10: "+sup10
Terminal.ecrireStringln("Note minimum: " + min);
Terminal.ecrireStringln("Note maximum: " + max);
```

Exemple 5 : Inversion (en place) d'un tableau

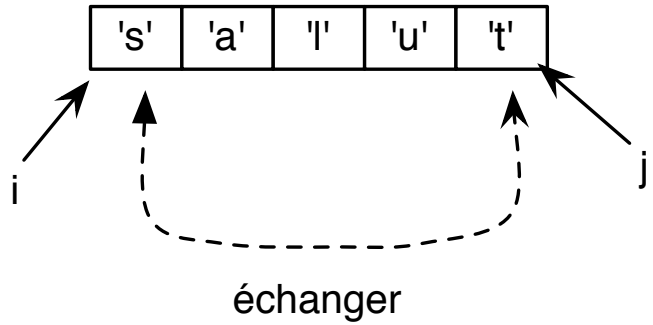
Problème : Inverser l'ordre des éléments d'un tableau de caractères, sans utiliser un autre tableau.



Solution :

- 2 variables d'itération i, j, initialisées avec premier et dernier indices du tableau ;
- à chaque itération, les valeurs dans les positions i et j sont échangés, puis i est incrémenté et j décrémenté,

5 : Inversion (en place) d'un tableau



Solution (suite) :

- i, j, initialisées aux bornes du tableau ;
- échanger valeurs dans i et j ; incrémenter i, décrémenter j ;
- 2 cas d'arrêt possibles selon taille du tableau :
  - taille impair : on doit arrêter lorsque i=j ;
  - taille pair : arrêt si `j < i`.
  - Conclusion : la boucle doit se poursuivre tant que `i < j`.

Initialisation + affichage avant inversion

```
// Initialisation
Terminal.ecrireString("Combien␣de␣caracteres␣?␣");
char [] t = new char[Terminal.lireInt()];
for(int i=0; i<=t.length-1; i++) {
   Terminal.ecrireString("Un␣caractere?␣␣");
   t[i] = Terminal.lireChar();
}
// Affichage avant inversion
Terminal.ecrireString("Tableau␣avant␣inversion:␣");
for(int i=0; i<=t.length-1; i++){
   Terminal.ecrireChar(t[i]);
}
Terminal.sautDeLigne();
```

Boucle d'inversion

```
// Inversion: arret si (i >= j)
char tampon;
for(int i=0, j= t.length-1; i < j; i++, j--) {
    tampon = t[i];
    t[i] = t[j];
    t[j] = tampon;
}
Terminal.ecrireString("Le tableau inverse: ");
for(int k=0; k<= t.length-1; k++) {
    Terminal.ecrireChar(t[k]);
}
```

Inversion d'un tableau : affichages

```
Java/Essais> java Inversion
Combien de caracteres? 5
Un caractere?  s
Un caractere?  a
Un caractere?  l
Un caractere?  u
Un caractere?  t
Le tableau avant inversion: salut
Le tableau inverse: tulas
```

Comprendre la représentation des données en mémoire

C'est utile pour ...

- comprendre les opérations : sur les données. Parfois, le résultat n'est pas celui que l'on inmagine ;
- en tirer parti : certaines opérations seront + ou - simples/efficaces selon la représentation interne.
- éviter les erreurs : une certaine représentation peut s'avérer délicate à manipuler (erreurs difficlles à détecter).

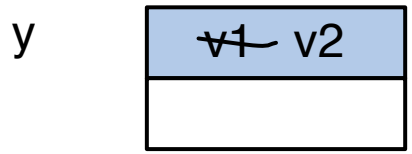
Deux catégories de données en Java

Données de ...

- type primitif : valeurs élémentaires
  - int, boolean, char, double, etc.
- type référence : valeurs composites, formées (possiblement) de plusieurs données plus élémentaires
  - tableaux, String, objets.

$\Rightarrow$ leur représentation en mémoire est différente,
$\Rightarrow$ leur utilisation en programmation aussi...

Représentation des variables en Java

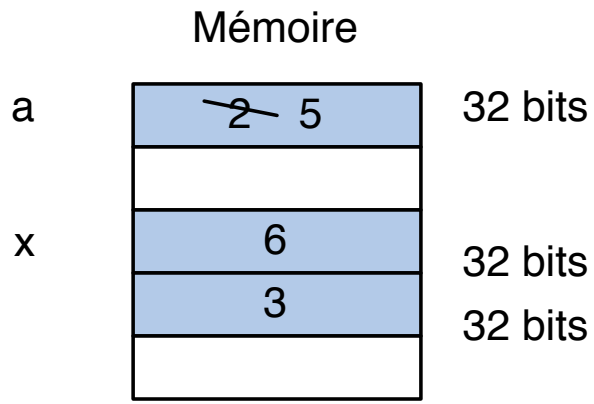


A toute variable correspond un emplacement de stockage :

- Il est fixe :
  - même emplacement tout le long du programme,
  - il est de taille fixe,
  - il contient la valeur courante de la variable ;
- taille + contenu $\Rightarrow$ dépendent de son type !
  - type primitif.
  - type référence.

Emplacement de stockage : types primitifs

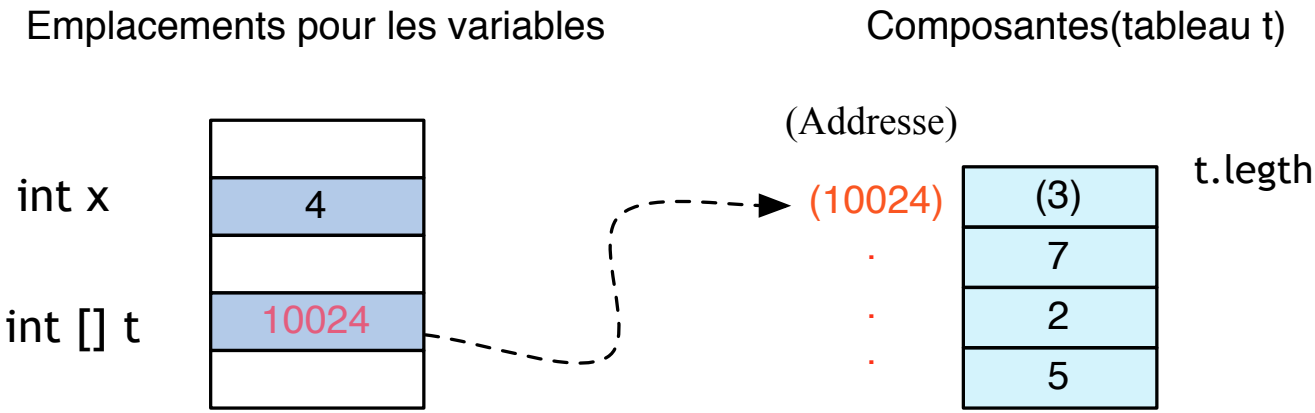
- taille : variable selon le type.
  - `int` ⇒ 32 bits
  - `double` ⇒ 64 bits
  - `char` ⇒ 16 bits
  - ...
- contenu stocké : la donnée *en place*, un entier, un double, etc.




```
public static void main(….) {
  int  a = 2;
  double x = 6.3;
  a = a+2;
  ….
}
```

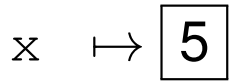
Emplacement de stockage : types référence

- donnée de type référence ⇒ toujours composite (plusieurs) ;
- emplacement de stockage ⇒ ne contient pas les données ;
- il contient :
  - adresse mémoire d'un espace ailleurs pour les données.

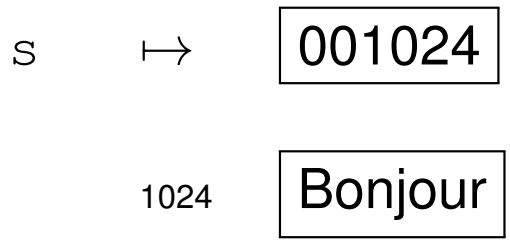


Synthèse : représentation des données

- variable types primitif : contient sa valeur **sur place**. En mémoire, `int x = 5;`



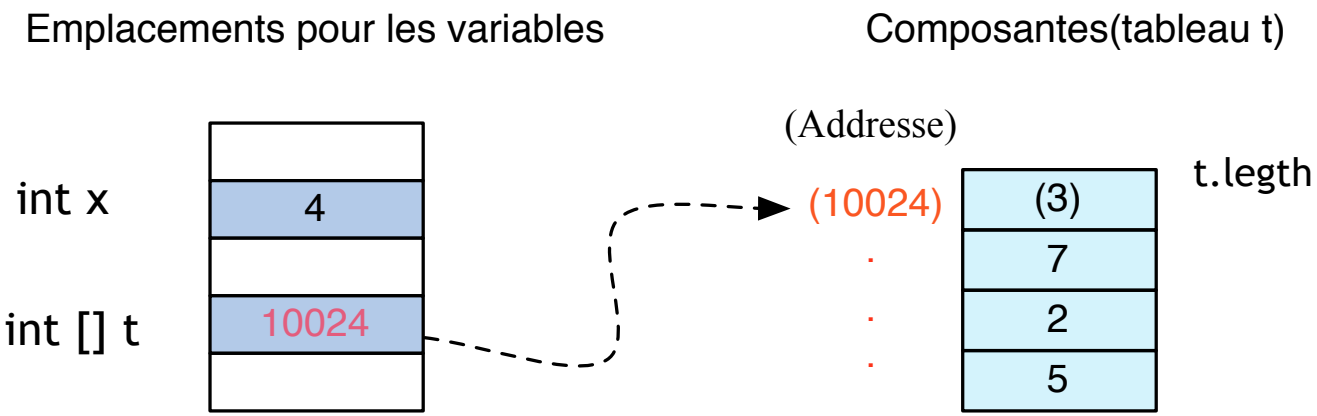
- variables type référence : (objets, tableaux) **ne contiennent pas** leurs valeurs, mais **une adresse** vers celles-ci. En mémoire, `String s = "Bonjour";`



Trouver composantes ⇒ aller à l'adresse référencée

Exemple

```
int x = 4;
int[] t = {7,2,5};
```



- `x` est de type primitif : elle contient **directement** sa valeur.
- `t` est de type référence : elle ne contient pas le tableau, mais l'adresse où se trouvent ses composantes.
- `t` est un *pointeur ou référence*.

Exemples de données de types référence

- Une variable de type String, ne contient pas la chaîne elle-même, mais l'adresse mémoire où se trouve la chaîne.
- La variable `int [] t = {4, 6, 3}` ne contient pas le tableau, mais l'adresse où se trouvent ses composantes.
- Chacune de ces variables est un *pointeur ou référence*.

Retour sur la création de tableaux

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

1. Déclaration `int [] t` = `new int [3]`

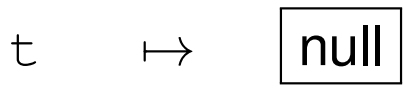
⇒ Réserve un emplacement pour `t`, initialisé à l'adresse `null`.

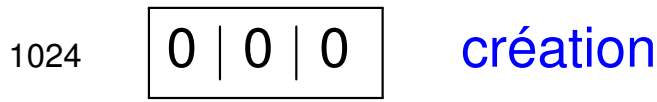
$$t \mapsto \boxed{\text{null}} \quad \text{déclaration}$$

Retour sur la création de tableaux (2)

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

2. Création `int [] t =` `new int [3]`
   1. Réserver un espace ailleurs pour stocker 3 composantes `int` ;
   2. Initialiser avec valeurs par défaut (0).



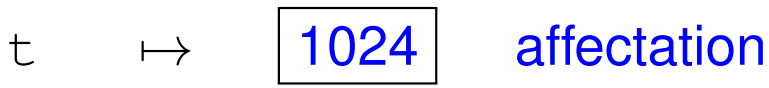


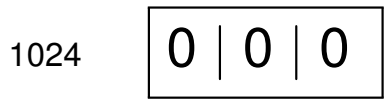
Retour sur la création de tableaux (2)

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

3. Affectation `int [] t` **`=`** `new int [3];`

   - Copier l'adresse où se trouvent les composantes dans l'emplacement de stockage pour `t`.





Affectation entre variables de type référence

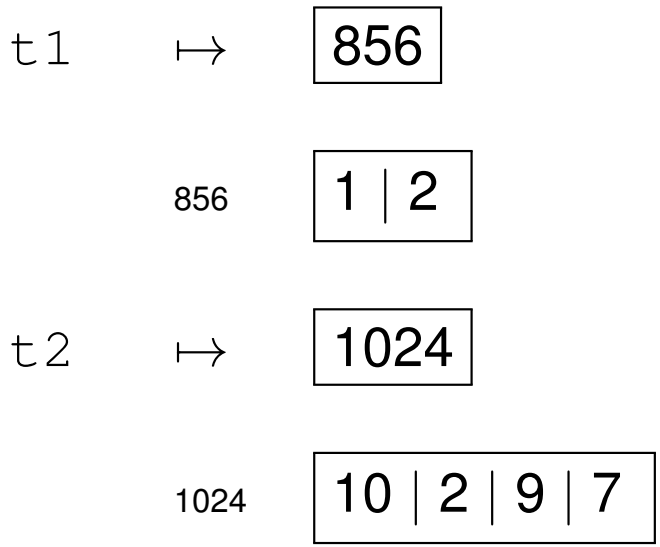
- l'affectation entre variables de type pointeur est possible, à condition que les types de ces variables soient compatibles,
- par exemple, entre deux tableaux de int, deux Strings, etc.
- Quelle est le résultat d'une telle affectation ?

```
int [] t1, t2;
t1 = {1,2};
t2 = {10,2, 9, 7};
t1 = t2;
```

$\Rightarrow$ On copie le contenu d'une variable dans l'autre.
$\Rightarrow$ Ce contenu est **une adresse**.

Affectation entre variables de type référence

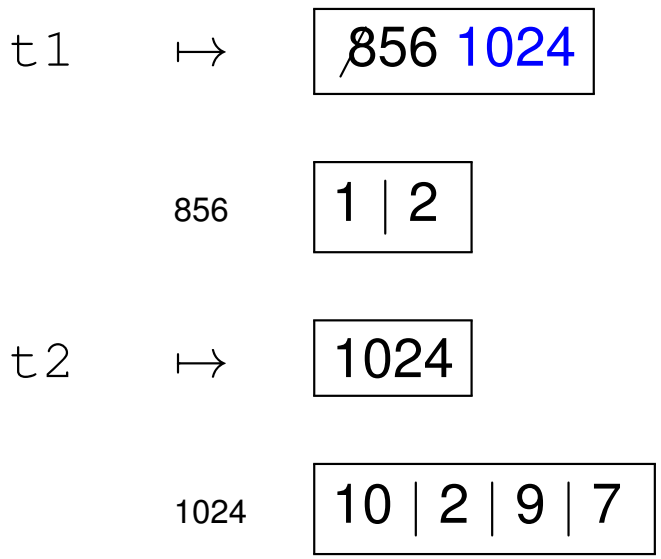
```
int [] t1, t2;
t1 = {1,2};
t2 = {10,2, 9, 7};
t1 = t2;
```



On recopie le contenu d'une variable dans l'autre.
On recopie une adresse.

Affectation entre variables de type référence

```
int [] t1, t2;
t1 = {1,2};
t2 = {10,2, 9, 7};
t1 = t2;
```



⇒ t1 et t2 contiennent la **même adresse**.

Affectation entre variables de type référence

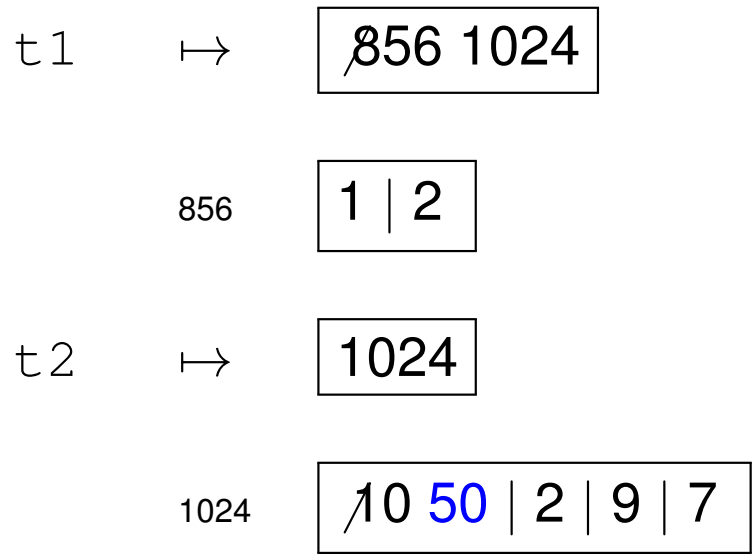
```
int [] t1, t2;
t1 = {1,2};
t2 = {10,2, 9, 7};
t1 = t2;
t1[0] = 50;
Terminal.ecrireInt(t2[0]);
```

- On copie le contenu d'une variable dans l'autre. Ce contenu est **une adresse**.
  $\Rightarrow$ t1 et t2 contiennent la **même adresse**.
- Elles pointent vers le même emplacement physique de la mémoire.
  $\Rightarrow$ tout changement dans l'une modifie ce qui est pointé par l'autre.

On dit de t1 et t2 qu'elles **partagent** le même espace.

Affectation entre variables de type référence

```
int [] t1 = {1,2};
int [] t2 = {10,2, 9, 7};
t1 = t2;
t1[0] = 50;
Terminal.ecrireInt(t2[0]);
```



`Terminal.ecrireInt(t2[0])` ⇒ affiche 50

Tableaux synonimes ou partagés

```
int [] t;
int [] m = {2,3,4,5,6};
t = m;     // t et m designent un meme tableau
Terminal.ecrireStringln("t[0] = " + t[0]);
Terminal.ecrireStringln("m[0] = " + m[0]);
t[0] = 9;
Terminal.ecrireStringln("Nouveau t[0] = " + t[0]);
Terminal.ecrireStringln("Nouveau m[0] = " + m[0]);
```

```
t[0] = 2
m[0] = 2
Nouveau t[0] = 9
Nouveau m[0] = 9
```

Affectation entre tableaux

Les tableaux d'une affectation peuvent avoir des longueurs différentes

Pourquoi ?

```
int [] t = {10, 20};
int [] m = {2,3,4,5,6};
Terminal.ecrireStringln("Longueur de t = "+ t.length);
t = m;  // t contient maintenant un tableau de 5 element
// Nouvelle longueur de t
Terminal.ecrireString("Nouvelle longueur t = "+ t.length
```

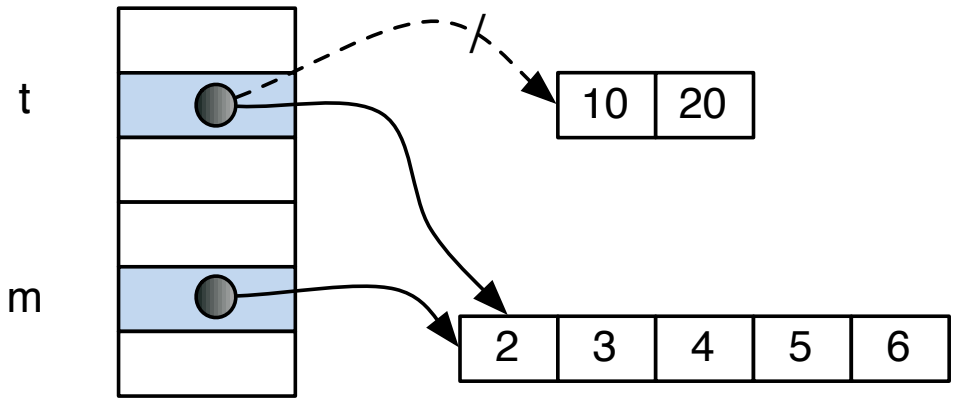
```
Longueur de t = 2
Nouvelle longueur de t = 5
```

Affectation entre tableaux

Les tableaux d'une affectation peuvent avoir des longueurs différentes

Pourquoi ?

```
int [] t = {10, 20};
int [] m = {2,3,4,5,6};
Terminal.ecrireStringln("Longueur de t = "+ t.length);
t = m;  // t contient maintenant un tableau de 5 element
// Nouvelle longueur de t
Terminal.ecrireString("Nouvelle longueur t = "+ t.length
```



Comparer des types référence

Qu'affiche ce programme ?

```
int [] t1 = {10, 20};
int [] t2 = {10, 20};
int [] t3 = t1;
if (t1==t2){ Terminal.ecrireStringln("t1==t2");
} else {
     Terminal.ecrireStringln("t1!=t2");  }
if (t1==t3){
     Terminal.ecrireStringln("t1==t3");
} else {
     Terminal.ecrireStringln("t1!=t3");  }
```

Egalité des types référence

L'exécution de ce programme produit :

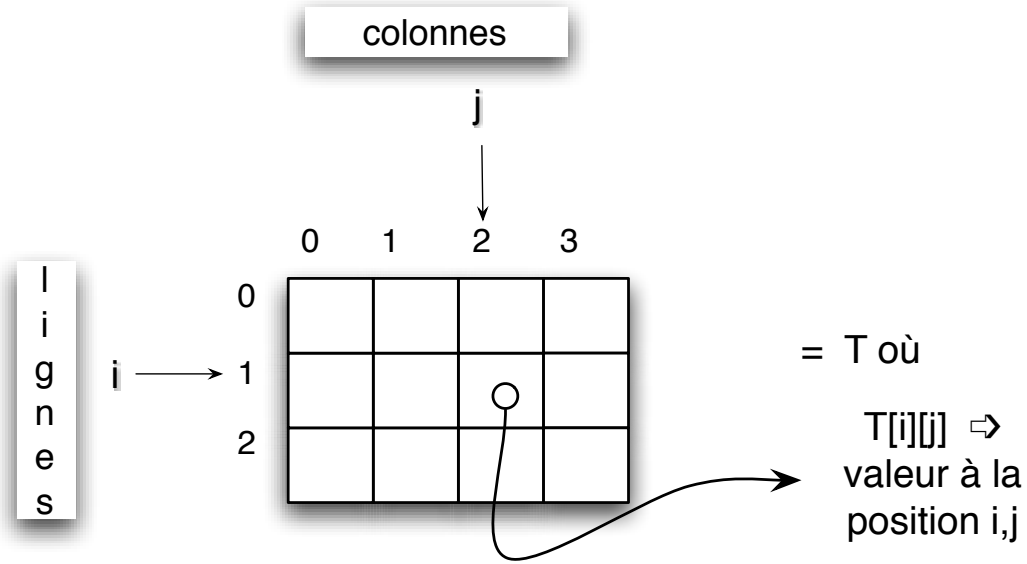
```
> java Chap12d
t1!=t2
t1==t3
```

Comparer tableaux, Strings

- Tableaux et Strings sont des types référence : **ce sont des pointeurs**.
- L'opérateur == utilisé pour les comparer, **compare leurs adresses**, autrement dit, cela teste s'ils pointent vers le même emplacement en mémoire.
- Ce n'est pas la bonne méthode si l'on veut comparer **leur contenu**, c.a.d, si leurs valeurs internes sont identiques.
- On doit donc utiliser ou écrire des méthodes qui comparent une à une chacune de leurs composantes internes.

Tableau à deux dimensions ou matrice

- 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

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)
```

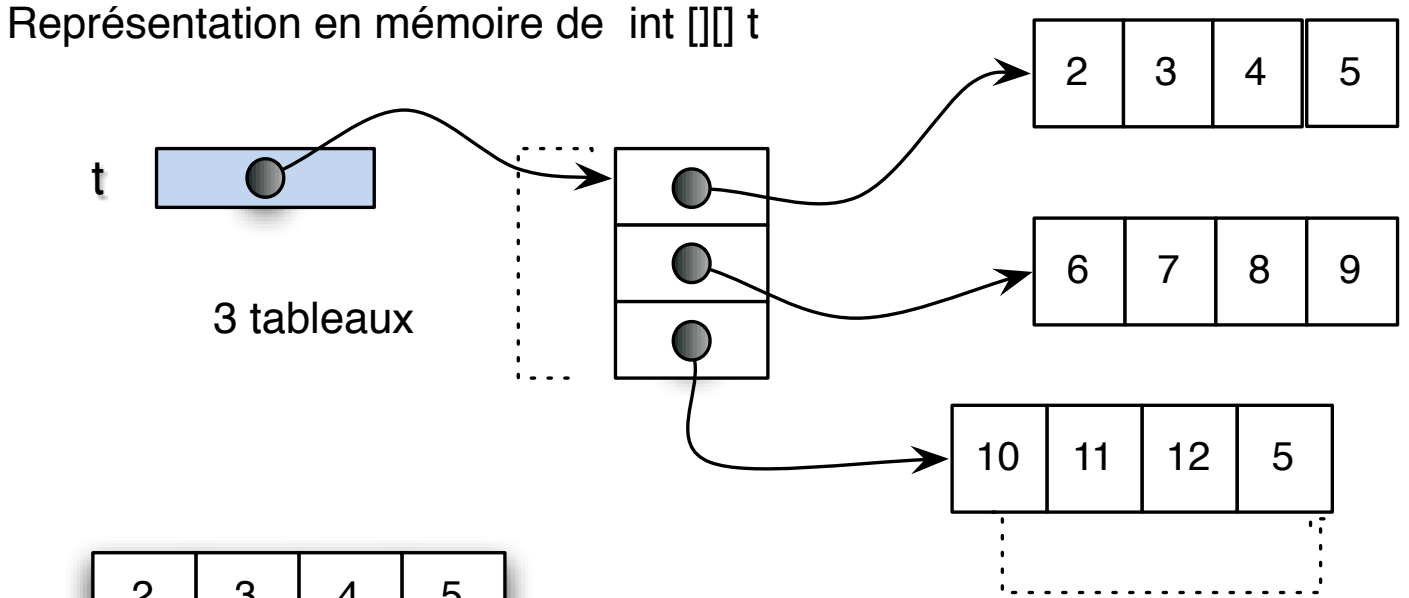
T =

	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 en mémoire des matrices

- 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 composantes de type `int`.

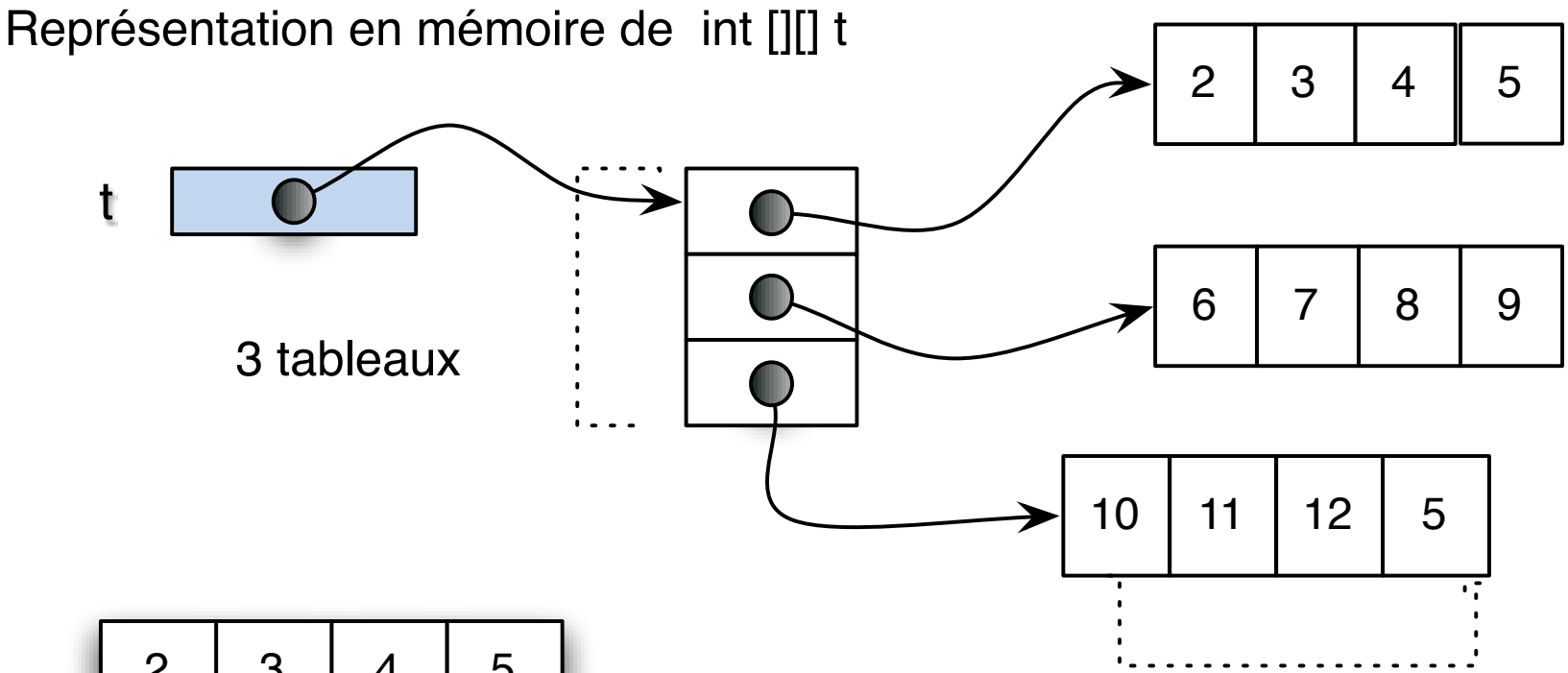


2	3	4	5
6	7	8	9
10	11	12	5

.... que nous verrons comme cette grille

Représentation en mémoire des matrices

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



2	3	4	5
6	7	8	9
10	11	12	5

.... que nous verrons comme cette grille

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.ecrireIntln(t.length);         // affiche 3
Terminal.ecrireIntln(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 crée 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

```
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
```

Parcours des matrices

- Réalisé en général avec 2 boucles imbriquées :
  - une boucle externe pour parcourir les lignes pour des indices compris entre 0 et `mat.lenght-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].lenght-1`.

Parcours des matrices

```
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

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
```

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 séléctionant dans T à nouveau le plus petit élément que l'on recopie à la deuxième place de res.
- On continue jusqu'à avoir séléctionne 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 séléctionant dans T à nouveau le plus petit élément que l'on recopie à la deuxième place de res.
- On continue jusqu'à avoir séléctionne et recopié N éléments où N est la taille de T.