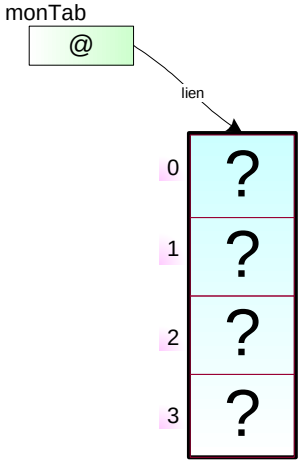
Les tableaux en Java

Table des matières :

1 Introduction

Un tableau est un outils de base pour gérer un ensemble d'éléments.
La traduction de tableau en anglais est « array ».



Le tableau est un **objet** qui mémorise un ensemble de valeurs contiguës en mémoire, auxquelles on accède grâce à un indice entier compris entre 0 et le nombre d'éléments moins un (Voir Figure suivante).

1. Les éléments d'un tableau sont tous du même type.
2. Ces éléments peuvent être des données de type primitif ou des références d'une classe donnée.
3. La taille d'un tableau est déterminée lors de sa création et ne peut être modifiée par la suite.
4. Chaque tableau a un champ public **length** contenant sa

taille.

5. La position d'un élément dans un tableau est déterminée avec un indice entier.
6. L'indice du premier élément d'un tableau est toujours 0 et l'indice du dernier élément d'un tableau est le nombre d'éléments du tableau moins 1.
7. La MV vérifie à l'exécution que les indices utilisés pour accéder à un élément figurent bien dans l'intervalle autorisé (ArrayIndexOutOfBoundsException) et que le type d'un objet à stocker est compatible avec le type du tableau (ClassCastException).

2 Syntaxe de la déclaration d'une référence vers un tableau

Syntaxe de la déclaration d'un tableau qui s'appelle « elements » :

```
TypeDeElement [] elements ;
```

Explications :

`TypeDeElement` : type primitif ou le nom d'une classe.

`[]` : Java laisse la liberté de placer les crochets avant ou après l'identificateur du tableau, contrairement à C# où les crochets `[]` doivent toujours précéder l'identificateur du tableau.

`elements` : la référence du tableau (référence de type `TypeDeElement[]`).

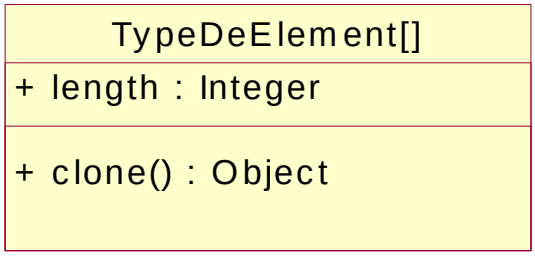
Remarques

Comme toute référence, `elements` peut aussi être égale à null.

La syntaxe « `TypeDeElement []` » peut être utilisée aussi comme type de paramètre ou comme type de retour d'une méthode retournant un tableau.

3 Représentation UML

Le diagramme de classes (UML) suivant représente un tableau d'objet:



4 Création d'un tableau sans initialisation immédiate

4.1 Utilisation de l'opérateur new avec la taille voulue du tableau.

Vous devez donner la taille du tableau à créer.
Exemple :

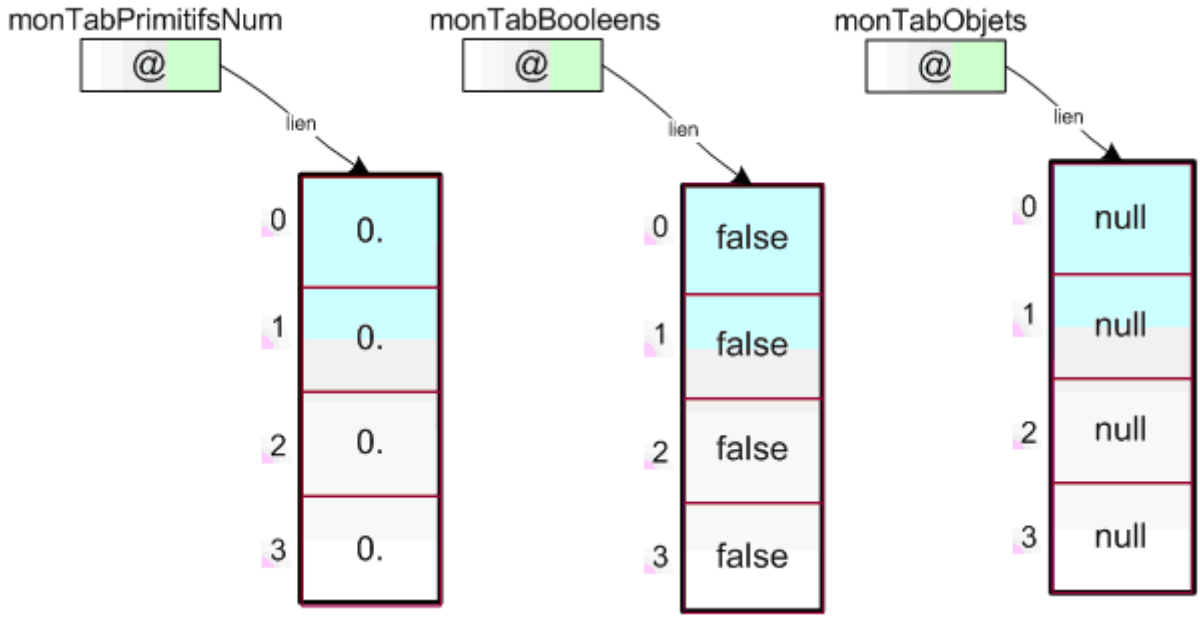
```
float [] tabNombres = new float [4]


String [] textes;//tableau  d'objets
…
int taille = 10;
textes = new String [taille];
textes[1] = «azerty »
```

Un tableau peut avoir une taille nulle.

A la création du tableau, ses éléments sont initialisés avec une valeur par défaut.
Suivant le type d'élément, voici la nature de la valeur par défaut:

Type d'éléments	Valeur par défaut
numérique ou caractère	0
boolean	false
type objet (références)	null.



Donc, la création d'un tableau de type objet ne génère pas d'objet pour chaque élément du tableau, mais uniquement des références initialisées à null par défaut.

5 Création d'un tableau avec initialisation immédiate

On peut effectuer la création avec initialisation d'un objet tableau de deux façons différentes.

5.1 Utilisation d'accolades

Lors de la déclaration du tableau, si les valeurs initiales des éléments à stocker sont connues, la liste de ces valeurs peut être citée entre accolades après l'opérateur =
Exemple :

```
int [] nombresPremiers = {1, 2, 3, 5, 7, 11, 13, 17};
String [] accueils = {"Bonjour", "Hello", "Guten Tag"};
Livre [] etagere = { new Livre(1), new Livre(2), new
      Livre(3) } ;
```

5.2 Utilisation d'accolades sans référence locale créée

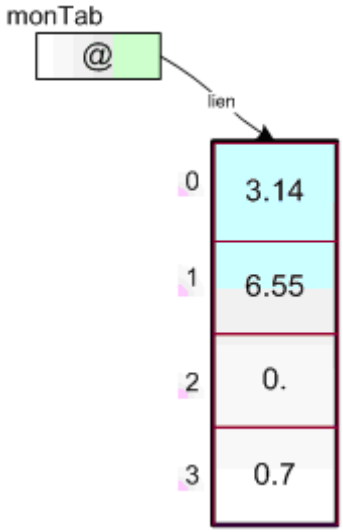
A tout moment, en utilisant l'opérateur new et une liste entre accolades de valeurs. Cette notation est très pratique pour passer un tableau en paramètre à une méthode sans qu'il faille créer une variable locale pour ce tableau.
Exemple :

```
char lettreT = 't';
char lettreO = 'o';
String s = new String ( new char [] {'t', 'o', 't', 'o' });
```

6 Représentation schématique d'un tableau initialisé

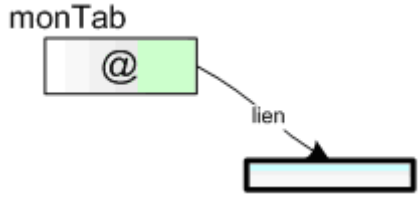
6.1 Tableau de type primitif

Un tableau qui contient des valeurs de type primitif, stocke directement ses valeurs les unes après les autres.



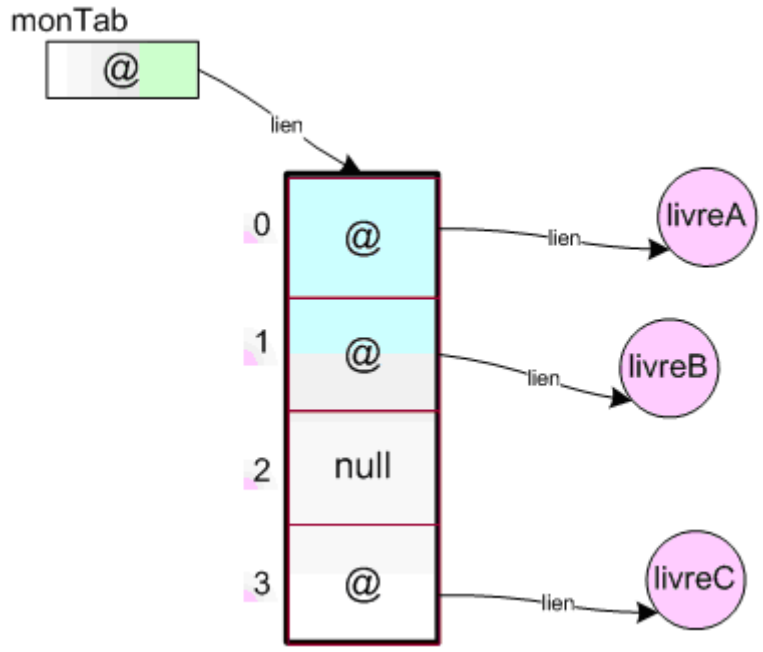
6.2 Tableaux de longueur nulle

Java permet de créer des tableaux vides comme int [] tableauVide = {};
Le champ length d'un tel tableau est égal à 0. Par exemple, si aucun argument n'est passé à la méthode main d'une application, args est un tableau vide.



6.3 Tableau de type objet

Un tableau, qui contient des objets, stocke les références des éléments.



7 Afficher un tableau grâce à showMessageDialog

Si l'objet en paramètre de la méthode showMessageDialog est un tableau, cette méthode affiche les éléments du tableau les uns en dessous des autres.

8 Utiliser un tableau

Une fois que l'on a créé un objet tableau, pour consulter ou modifier un élément, on utilise son indice dans le tableau.

Exemple :

```
tabNombres [1] = 3.14f; //stockage sur le 2ème élément

textes [0] = accueils[2]; //copie de référence du 3ème élément

int plusPetit = nombresPremiers [0];  //Tableau des n. Premiers
int plusGrand = nombresPremiers [nombresPremiers.length - 1];
```

En tant qu'objet Java, un tableau hérite de la classe java.lang.Object et de toutes ses méthodes. La méthode `clone` d'un tableau est notamment redéfinie pour retourner une copie du tableau. En revanche, la méthode `equals` fonctionne mal car elle n'est pas redéfinie pour comparer le contenu de deux tableaux. Quant à la méthode `toString`, elle n'**affiche pas la liste des valeurs**.

Exemple :

```
int [] tabValeurs = {0, 5, 10, 15, 20};
int [] copieTab = (int [])tabValeurs.clone();

boolean b = tabValeurs.equals(copieTab);
```

Attention, ici, b vaudra toujours false !!!!!!!!!!

La MV vérifie lors de l'exécution si les tableaux sont manipulés correctement :

- Si l'indice utilisé pour accéder à un élément d'un tableau n'est pas compris entre 0 et la taille du tableau moins 1, une exception de classe java.lang.ArrayIndexOutOfBoundsException est déclenchée.
- Si la taille requise pour un nouveau tableau est négative, une exception de classe java.lang.NegativeArraySizeException est déclenchée.
- Si vous tentez de stocker dans un tableau un objet dont le type est incompatible avec le type du tableau, une exception java.lang.ArrayStoreException est déclenchée.

9 Tableau multidimensionnel

On accède à un élément d'un tableau multidimensionnel par le biais de plusieurs indices. Les tableaux bidimensionnels sont par exemple utilisés pour, mémoriser en informatique les matrices mathématiques.

Remarque sur C#
Il n'existe pas en Java l'équivalent des tableaux multidimensionnels C# déclarés avec [,] dont tous les éléments sont rangés dans une seule zone mémoire.

Java permet de créer et initialiser un objet tableau multidimensionnel de façons différentes.

9.1 Utilisation de l'opérateur new et des tailles voulues du tableau.

A tout moment, en utilisant l'opérateur new et la taille dans chaque dimension du tableau. Les éléments sont alors initialisés avec leur valeur par défaut.
Exemple d'un cube :

```
int [][][] tab3Dimensions;
tab3Dimensions = new int [3] [3] [3] ;

tab3Dimensions[2][0][1] = 12 ;//init d'un éléments
```

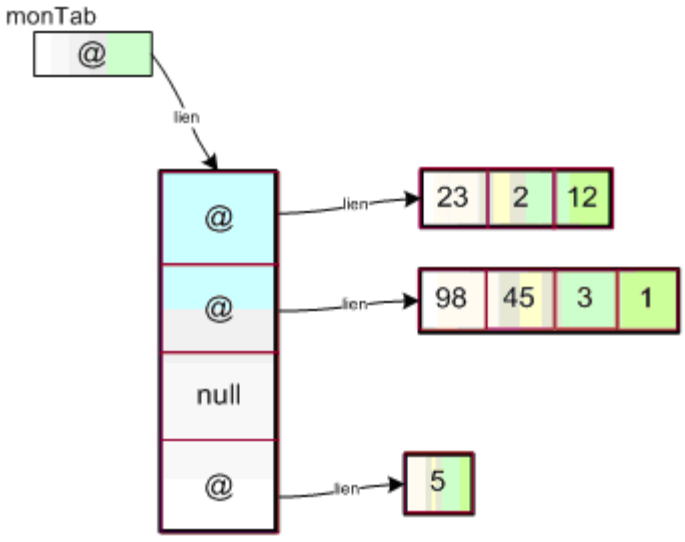
9.2 Utilisation d'accolades

Lors de la déclaration du tableau avec la liste des valeurs initiales citées entre accolades imbriquées.
Exemple d'une matrice :

```
double [][] matriceIdentite = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
```

9.2.1 Création et initialisation par étapes

Java manipule les tableaux multidimensionnels comme des tableaux de tableaux ; chaque sous-tableau peut avoir une taille différente si nécessaire. On crée le tableau principal puis ses sous-tableaux.



Exemple d'un triangle de Pascal de quatre lignes:

```
int [][] trianglePascal = new int [4] [] ;

for (int i =0; i < trianglePascal.length; i++)
     trianglePascal [i] = new int [i +1]; //Création d'une ligne du tableau

//initialisation
TrianglePascal[0][0]=1; TrianglePascal[1][0]=1; …,etc
```

Ce qui revient à la déclaration suivante :

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

10 Manipuler les tableaux

Une fois les tableaux instanciés et initialisés, on souhaitera probablement les manipuler.

10.1 La classe java.util.Arrays

La classe java.util.Arrays contient un ensemble de méthodes **de classe** (static) qui permettent d'effectuer des traitements sur les tableaux de type primitif ou du type Object .
Le diagramme de classes suivant présente la classe java.util.Arrays (J2SE 1.3):

Nom de la méthode	Explications
equals	compare les éléments de deux tableaux
fill	remplit tout ou partie d'un tableau avec une valeur donnée
sort	trie les éléments d'un tableau dans l'ordre ascendant
binarySearch	renvoie l'indice du premier élément égal à une valeur dans un tableau **trié**

Ces méthodes sont largement surchargées pour chaque type primitif mais aussi pour des élément de type Object.
Dans le cas d'Object, Arrays.sort() vérifie si les éléments de la collection sont Comparable (instanceof) ensuite il s'appuie sur le compareTo() qui définit l'ordre de tri par défaut.

Aparté sur la manipulation des tableaux en C#

La classe System.Array de C# correspond à la classe java.util.Arrays en Java et pas à la classe java.lang.reflect.Array utilisée pour les opérations de réflexion sur les tableaux (manipulation des tableaux avec des méthodes).

10.2 java.lang.System.arraycopy : la copie de tableaux

La méthode pour copier des éléments d'un tableau dans un autre est dans la classe java. lang.System :

```
public static void arraycopy(0bject src, int srcPos, Object
dest, int destPos, int length)
```

11 La méthode main

Voici la signature de la méthode main:

```
public static void main (String [] args)
```

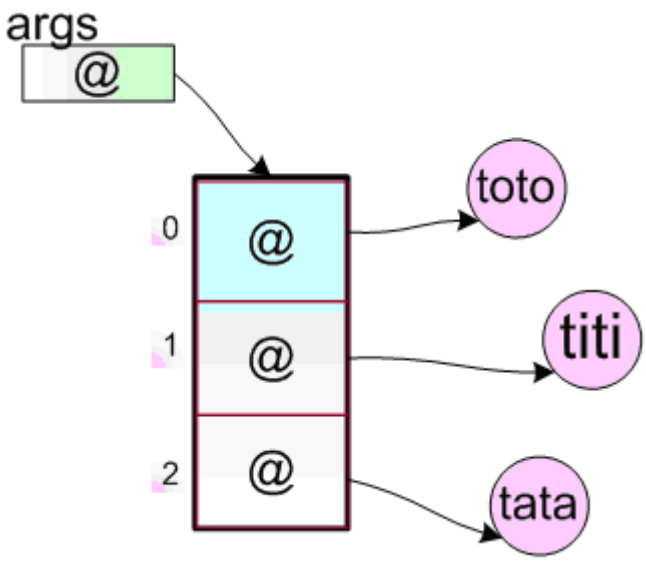
- main est une méthode de classe est appelée par la MV sans créer d'instance de la classe,
- main ne renvoie pas de valeur,
- main reçoit en paramètre un tableau de chaînes de caractères contenant certains arguments de la ligne de commande (voir paragraphe suivant).

12 La ligne de commandes de lancement

Soit l'exemple suivant d'une commande « java » avec comme classe éxécutée : MaClasse

java -classpath ..\classes biblio.tests.MaClasse toto titi tata

Le premier élément du tableau d'arguments de la méthode main est le paramètre qui suit la classe dans la commande « java ».
Schéma du tableau args récupéré dans le main :



Sur cette commande « java » et après la classe exécutée, si aucun argument n'est fourni, alors le tableau args est un tableau vide. (champ length à 0)

13 Signature de main depuis JAVA 5.0

Une liste d'arguments variable est une fonctionnalité de Java 5.0 qui permet à une méthode de recevoir en **dernier** paramètre un nombre variable de valeurs (zéro ou plus).
Une telle liste est déclarée en précédant le dernier paramètre du symbole ..., ce paramètre étant en fait un tableau.
Donc, voici une autre signature de la méthode main:

```
public static void main (String … args)
```