Les Tableaux

Les tableaux permettent de stocker

Une quantité fixe de variables de

type primitifs ou de références d'objets.

Les Tableaux
Déclaration

Déclaration du tableau

```
int [ ] monTableau;                    //déclaration d'un tableau de int

byte [ ] monAutreTableau ;             //déclaration d'un tableau de byte

String [ ] monTableauDeString;         //déclaration d'un tableau d'objets
```

Les Tableaux
Création

```
int [ ] monTableau;                    //déclaration d'un tableau de int

monTableau = new  int [ 12 ]           /*création du tableau pour
                                       contenir 12 variables de type int */


String [ ] monTableauDeString;         //déclaration d'un tableau d'objets

monTableauDeString = new String[ 3 ]   /*création du tableau pour
                                       contenir 3 références d'objet de
                                          type String*/
```

Les Tableaux
Déclaration

La déclaration et la création peuvent être regroupées

```
int [ ] monTableau = new int [ 12 ];
```

Valeur en dur

```
int a =10;
byte [ ] monAutreTableau = new byte [ a ];
```

Valeur liée à une variable

Les Tableaux
Initialisation

Initialisation à la création

```
int [ ] monTableau= {12,36,45,78,0};
```

```
/*déclaration et  création  d'un tableau pour
            contenir 5 variables de type int  et remplissage du
          tableau avec les valeurs*/
```

Les Tableaux
Initialisation

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

Chaque élément du tableau est accessible par :

monTableau [i] ou i est l'index de l'élément dans le tableau (0 à 3)

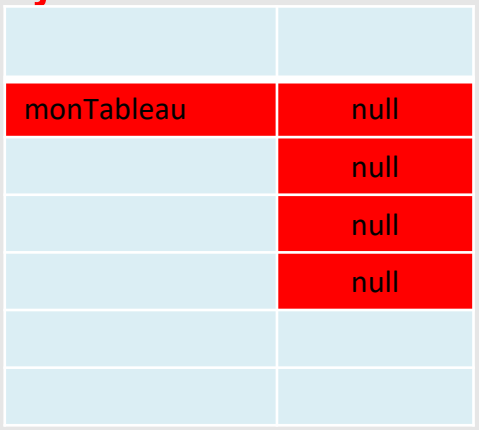
```
monTableau[ 0 ] = 458 ;
monTableau[ 1 ] = 49 ;
monTableau[ 2 ] = 22 ;
monTableau[ 3 ] = monTableau[ 2 ] ;
```

Les Tableaux
Représentation en mémoire

Les tableaux sont des objets leur déclaration crée une référence

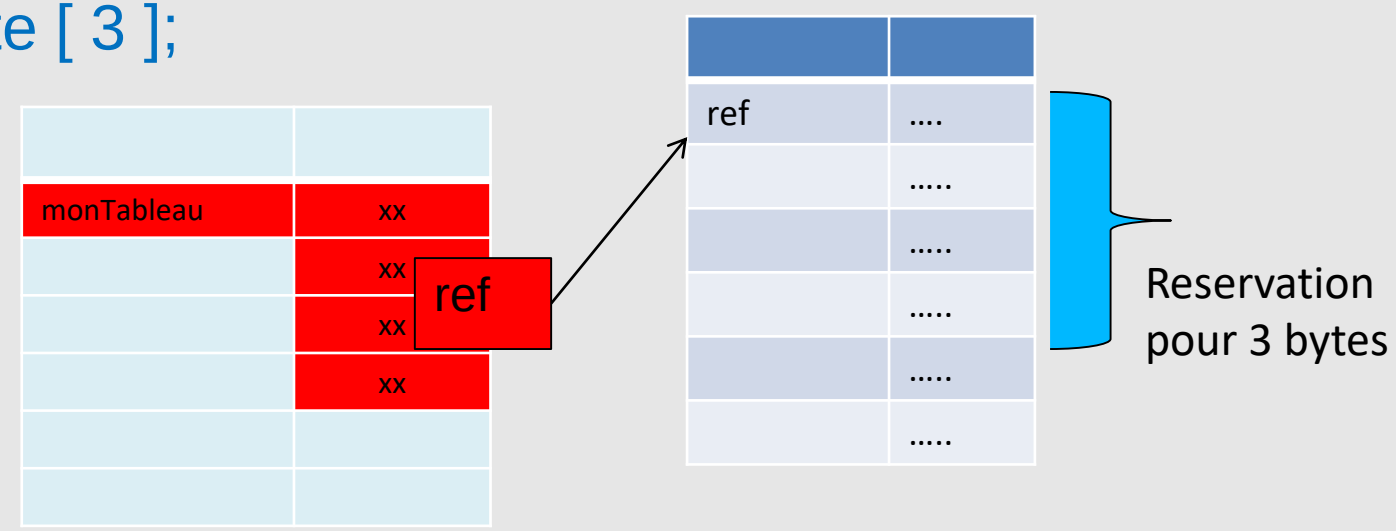
byte [ ] monTableau;

Déclaration d'un nom de variable associé à un tableau de bytes



monTableau = new byte [3];

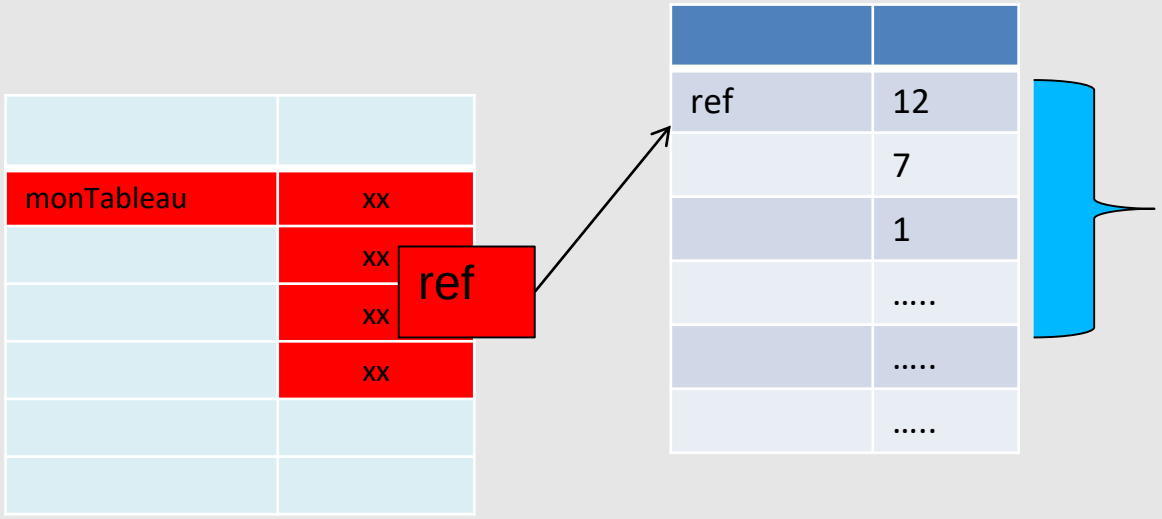
Création d'une référence vers un objet de type tableau et réservation de mémoire pour 3 bytes



Les Tableaux
Représentation en mémoire

```
monTableau[ 0 ] = 12;
monTableau[ 1 ] = 7;
monTableau[ 2 ] = 1;
```

Initialisation des valeurs des 3 bytes



Les Tableaux
Utilisation

```
double [ ] tabDouble = new double[50];
```

Ecriture dans le tableau

```
tabDouble[ 10 ] = 12.25;
```

Lecture d'une valeur

```
double d = tabDouble[ 6 ];
System.out.println(tabDouble [ 0 ]);
```

Taille du tableau

Elle est mémorisée dans l'attribut length du tableau

```
int laTaille = tabDouble.length;  // laTaille contiendra 50
```

Les Tableaux
Utilisation

La taille d'un tableau (mémorisée dans l'attribut length) est fixe, on ne peut la modifier.
Si on veut modifier la taille d'un tableau il faut en créer un autre et recopier les valeurs.

Comparaison

Les tableaux étant des objets on en peut pas les comparer en utilisant les opérateurs de comparaison (tab1 == tab2) car ceux-ci comparent les références et non les contenus.
Il faut utiliser la méthode: Arrays.equals(tab1, tab2) qui retourne un booléen
Ex:

```
if(Arrays.equals(tab1, tab2) ==true){
……
```

Les Tableaux
Utilisation

Recopie du contenu

Le contenu (total ou partiel) d'un tableau peut être copié dans un autre avec la méthode:

System.arraycopy(Object src , int srcPos , Object dest , int destPos, int length)

Ex: pour copier les trois premières valeurs du tableau tab1 au début du tableau tab2)

```
System.arraycopy(tab1 , 0, tab2 , 0 , 3);
```

Les Tableaux
Tableaux d'objet

```
StringBuilder [ ] monTableau;       //déclaration d'un tableau d'objets

monTableau= new StringBuilder [ 3 ] ;   /*création du tableau pour
                                          contenir 3 références d'objet de
                                            type StringBuilder*/

for(int i = 0 ; i < monTableau.length ; i++){
     monTableau [ i ] = new StringBuilder();
     }              //création des objets
```