

Les tableaux en Java

- Tableau de types primitifs et dim 1
- Tableau d'objets et dim 1
- Tableau de dimension 2 de types primitifs, rectangulaire et non-rectangulaires
- Travail à réaliser

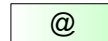
1

1

Tableau de types primitifs

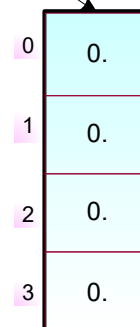
- `double[] monTab = new double[4] ;`

monTabPrimitifsNum



lien

Le tableau est un objet



XH

2

2

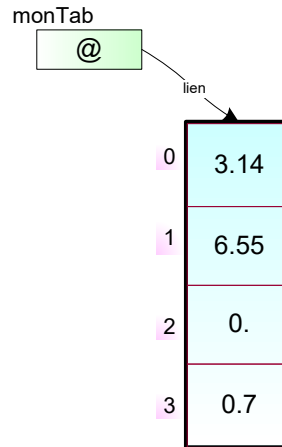


Tableau de types primitifs

- `monTab[0] = 3.14 ;`
- `monTab[1] = 6.55 ;`
- `monTab[3] = 0.7;`
- `//Utilisation`
- `double result= monTab[1] * 3;`

`//Itération sur le tableau`

```
for( int i=0; i<monTab.length; i++){  
    sysout.println(monTab[i]);  
}
```



XH

3

3



foreach Java5

`//foreach Java5`

`double cumul = 0;`

`for(double var : monTab) {`

`cumul = cumul + var;`

`}`

- `sysout.println(« résultat du cumul
: » + cumul);`

XH

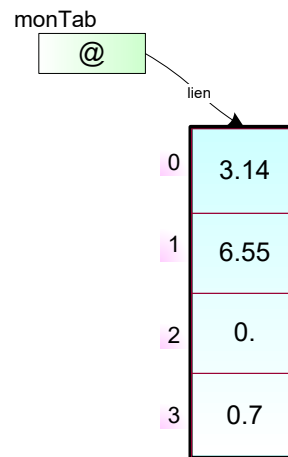
4

4



Autre écriture: l'initialisateur

- Des accolades pour l'initialisateur
- `double[] monTab = {3.14, 6.55, 0., 0.7} ;`
- Les 3 étapes précédentes en une ligne



XH

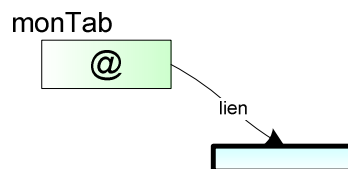
5

5



Cas particulier: le tableau de longueur nulle

- `double[] monTab = {} ;`



- `Sysout.println (« longueur du tableau : » + monTab.length);`
- Exemples: Pas d'argument passé à un programme, une chaîne vide

XH

6

6



Synthèse

Le tableau de type primitif

- Déclarations de tableaux
 - `int[] mois; // mois est affecté à null`
 - ou `int [] mois = new int[12];`
 - ou `int [] mois={31,28,31,30,31,30,31,31,30,31,30,31};`
 - deux syntaxes autorisées : `int mois[]` ou `int [] mois;` la seconde est préférée !
- Accès aux éléments
 - le premier élément est indexé en 0
 - Au runtime, vérification des bornes → levée d'exception
- Les tableaux sont des objets
- A voir plus loin
- Déclarations de tableaux de primitif de dimension 2 (matrice) (rectangulaire)
 - `double [][] m= new double [4][3];`

XH

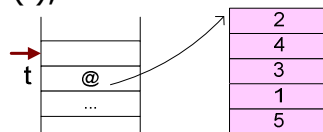
7

7

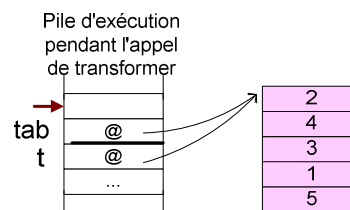


Passage de tableau en paramètres

- rappel: en java, passage de paramètre par valeur uniquement (et par recopie dans la pile)
- tableau en paramètre
 - la variable de type tableau est une référence
 - le passage par valeur et recopie de Java ne peut que transmettre la référence sur le tableau
- `static void transformer(int [] tab){`
 - `tab[0] = 8; // --> t[0] == 8;`
 - ...
- `}`
- `int[] t = {2,4,3,1,5};`
- `transformer(t);`

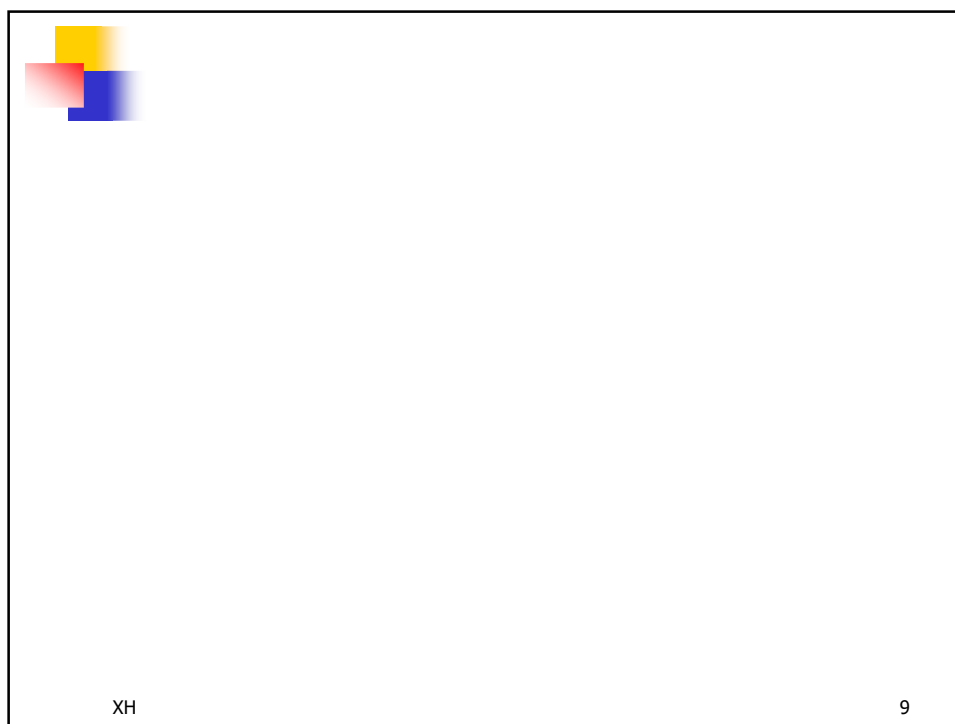


XH

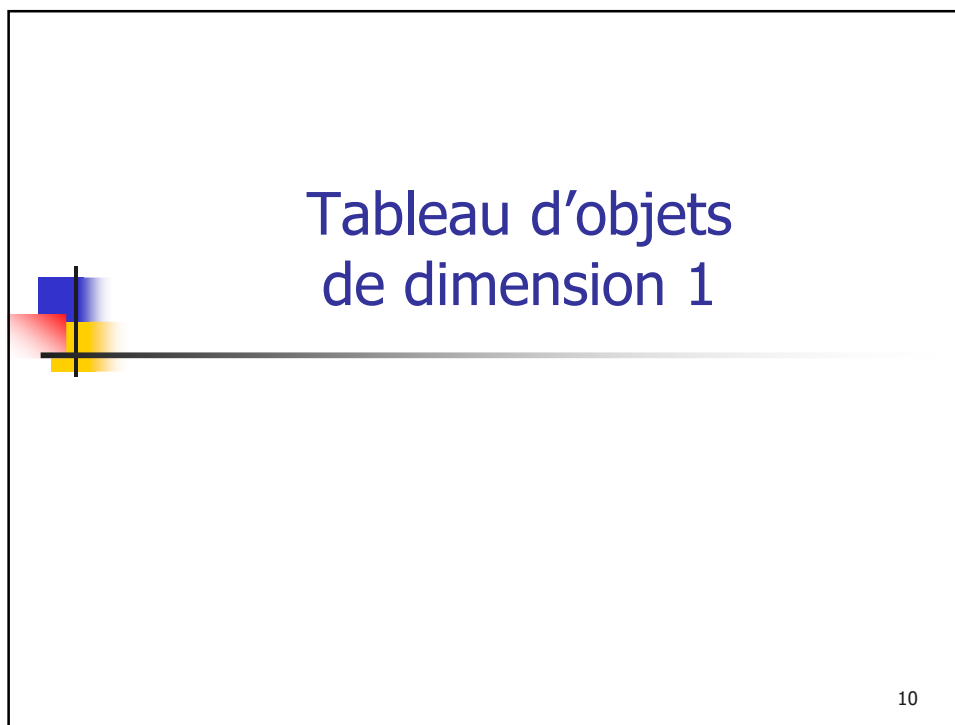


8

8



9

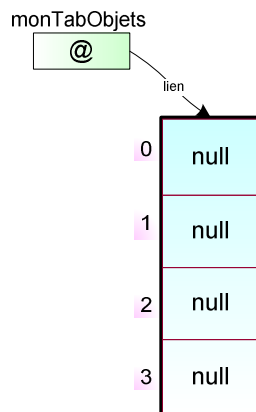


10



Tableau d'objets de dimension 1

- `Livre[] monTab;`
- `monTab = new Livre[4];`



XH

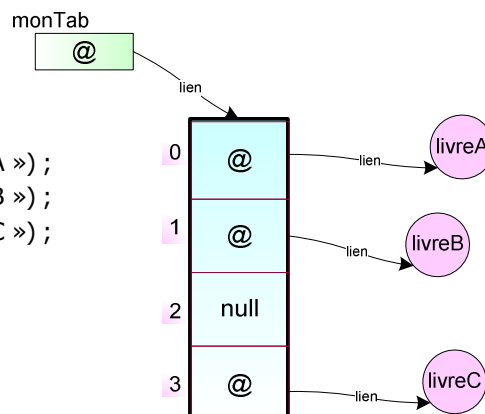
11

11



remplissage

- `monTab[0] = new Livre(« livreA »);`
- `monTab[1] = new Livre(« livreB »);`
- `monTab[3] = new Livre(« livreC »);`



XH

12

12



Utilisation/itération sur le tableau d'objets

```
//Itération sur le tableau
for( int i=0; i<monTab.length; i++){
    if( monTab[i] != null )
        sysout.println(monTab[i]);
}

//foreach Java5
for( Livre var : monTab) {
    if( var != null )
        sysout.println( var );
}
```

XH

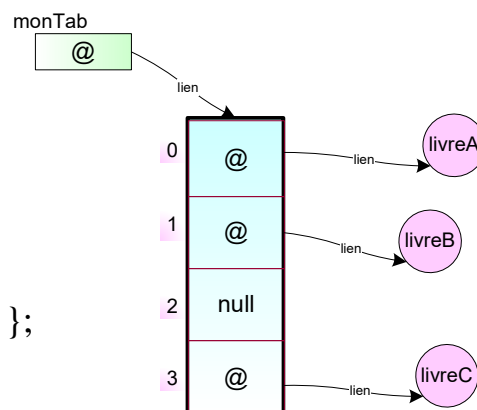
13

13



L'initialisateur avec les accolades

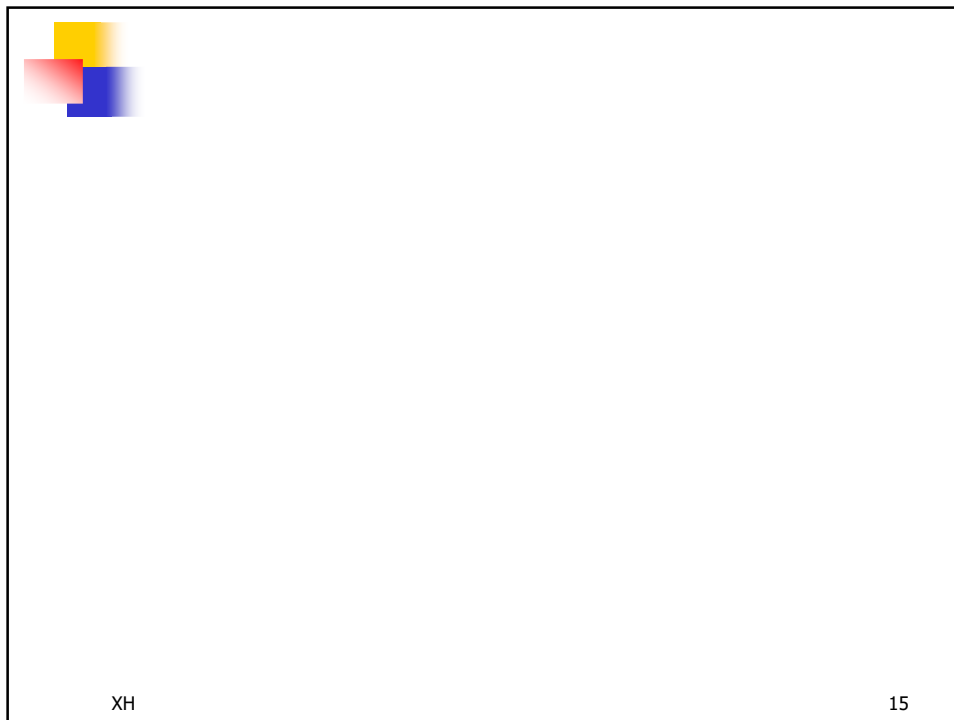
```
Livre [] monTab = {
    new Livre( "livreA"),
    new Livre( "livreB"),
    null,
    new Livre( "livreC") };;
```



XH

14

14



15




Tableau à deux dimensions de type primitif

Rectangulaire (matrice)
Non-rectangulaire

16

16



Tableau à deux dimensions de type primitif rectangulaire

- //matrice de 3 lignes et 4 colonnes
- **int[][] tab2 = new int[3][4];**
- System.***out.println("remplissage du tableau avec qq valeurs");***
- tab2[1][2] = 67;
- tab2[2][3] = 12;

XH

17

17



XH

18

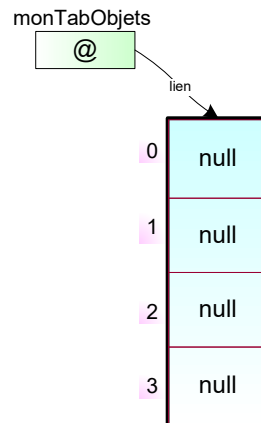
18



Tableau à deux dimensions de type primitif non-rectangulaire

- On ne traitera pas les tableaux à 2 dimension d'objets

■ `int [][] monTab = new int [4][] ;`



XH

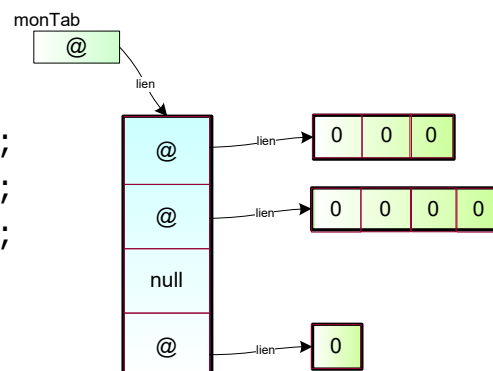
19

19



Tableau à deux dimensions de type primitif

- `monTab[0] = new int[3] ;`
- `monTab[1] = new int[4] ;`
- `monTab[3] = new int[1] ;`



XH

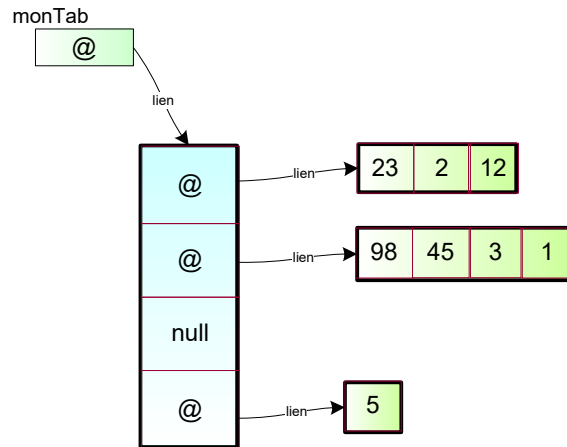
20

20



Tableau à deux dimensions de type primitif

- `monTab[0][0] = 23;`
- `monTab[0][1] = 2;`
- `monTab[0][2] = 12;`
- `monTab[1][0] = 98;`
- `monTab[1][1] = 45;`
- `monTab[1][2] = 3;`
- `monTab[1][3] = 1;`
- `monTab[3][0] = 5;`



XH

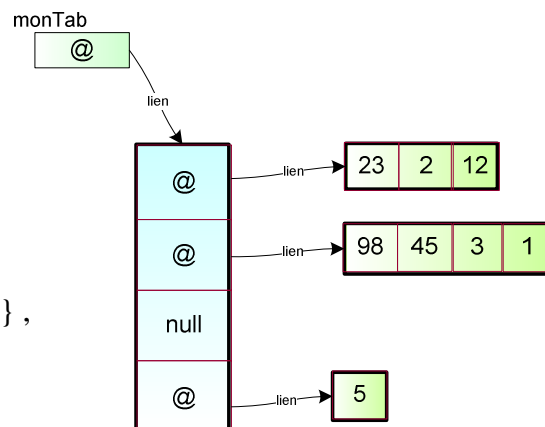
21

21



Autre écriture pour faire la même chose

```
int [][] monTab =
    { { 23, 2, 12 } ,
      { 98, 45, 3, 1 } ,
      null ,
      { 5 }
    };
```



XH

22

22



Itération sur le tableau dim2

```
//Itération sur le tableau
for( int L=0; L<monTab.length; L++){

    for( int c=0; c<monTab[L].length; c++) {
        sysout.println(monTab[L][c]);
    }
}
```

XH

23

23



Et le foreach?

```
System.out.println("Voici le contenu du tableau :");
for( int[] L : tab2){
    if ( L != null ){
        for( int c : L)
            System.out.print(" " + c);
    }
    System.out.println();
}
```

- //Voir aussi la démo fournie sur le site ftp habituel

XH

24

24



Remarque sur la méthode Arrays.sort()

- sort() vérifie si les éléments de la collection sont Comparable (instance of)
- D'où, on est sûr qu'il y a une méthode compareTo() qui sera utilisée (le tri naturel)

XH

25

25



Travail à réaliser

- Essayez les exemples des slides

XH

26

26