



Les tableaux en Java

Algorithmique et langage de programmation

Julien Romero, Gaël Thomas

Qu'est ce qu'un tableau en Java

- Un tableau est une structure de données qui contient plusieurs éléments **du même type**

Un tableau de 6 entiers

1	17	4	7	2	13
---	----	---	---	---	----

Allocation d'un tableau

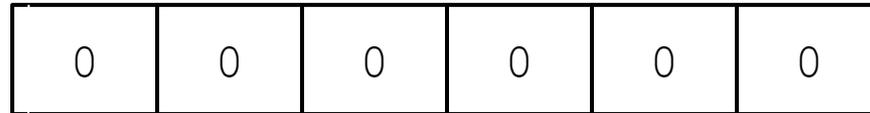
- Un tableau doit être alloué dans la mémoire avec `new type[n]`



Allocation d'un tableau de n éléments
ayant pour type `type`

- Par exemple : `new int[6]`

Alloue un tableau de 6 entiers



Allocation d'un tableau

- L'opérateur `new` renvoie une **référence** vers un tableau (une référence est un identifiant unique d'une structure de données)
- Par exemple, `new int[6]` renvoie une référence vers ce tableau :

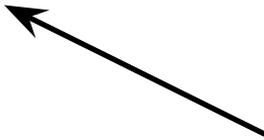


Note : Java met à 0 chaque élément lors d'une allocation

Déclaration

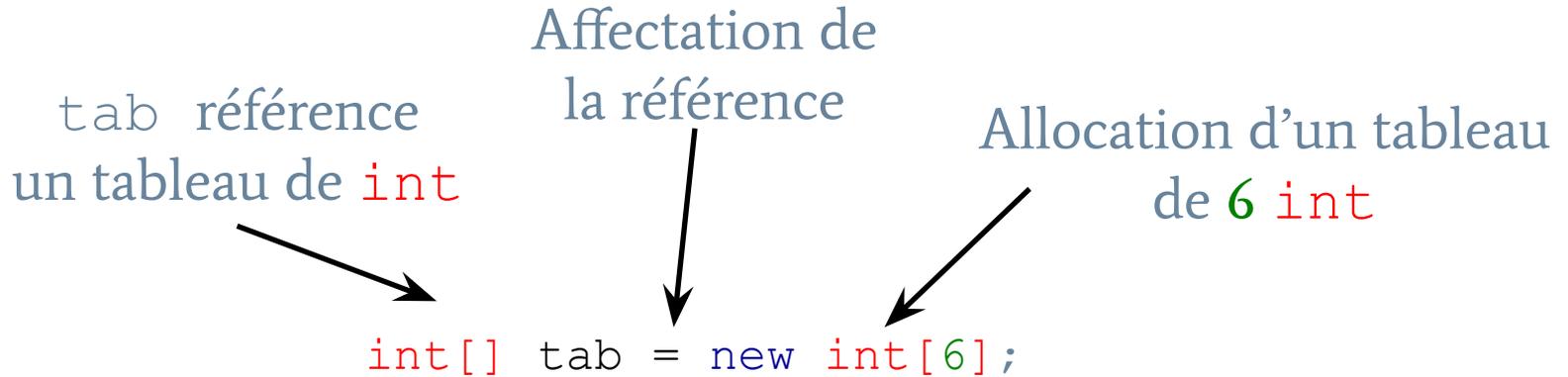
- Il n'existe pas de variable de type tableau en Java !
- En revanche, on peut déclarer une variable de type référence vers un tableau :

```
type [] var;
```



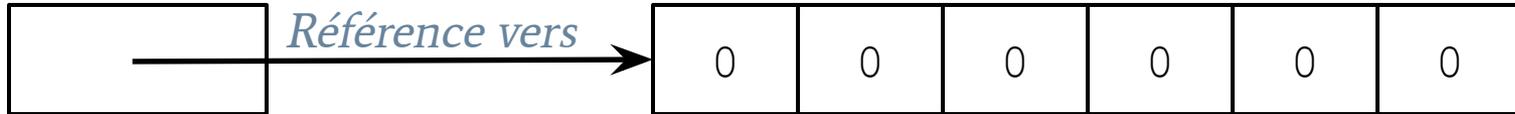
var est une variable
de type référence vers un tableau
contenant des éléments de type type

Exemple de déclaration



Variable `tab`

Le tableau alloué avec `new`



Allocation et initialisation

- On peut aussi allouer un tableau et l'initialiser avec

```
type[] tab = { x1, ... xn };
```

- Par exemple :

```
double[] tab = { 2.3, 17.0, 3.14, 8.83, 7.26 };
```

- En détail, le programme va
 - Allouer le tableau (comme avec `new`) puis initialiser les éléments
 - Renvoyer une référence vers le tableau

Accès à un tableau

- Accès à la taille du tableau avec `tab.length`
- Accès à un élément avec `tab[indice]`
 - Exemple : `tab[i] = tab[j] * 2;`
 - Attention : les éléments sont indexés à partir de 0
⇒ un tableau possède les éléments allant de 0 à `tab.length-1`
- Un accès en dehors des bornes du tableau provoque une erreur à l'exécution
(`ArrayOutOfBoundsException`)

Parenthèse: La mémoire simplifiée

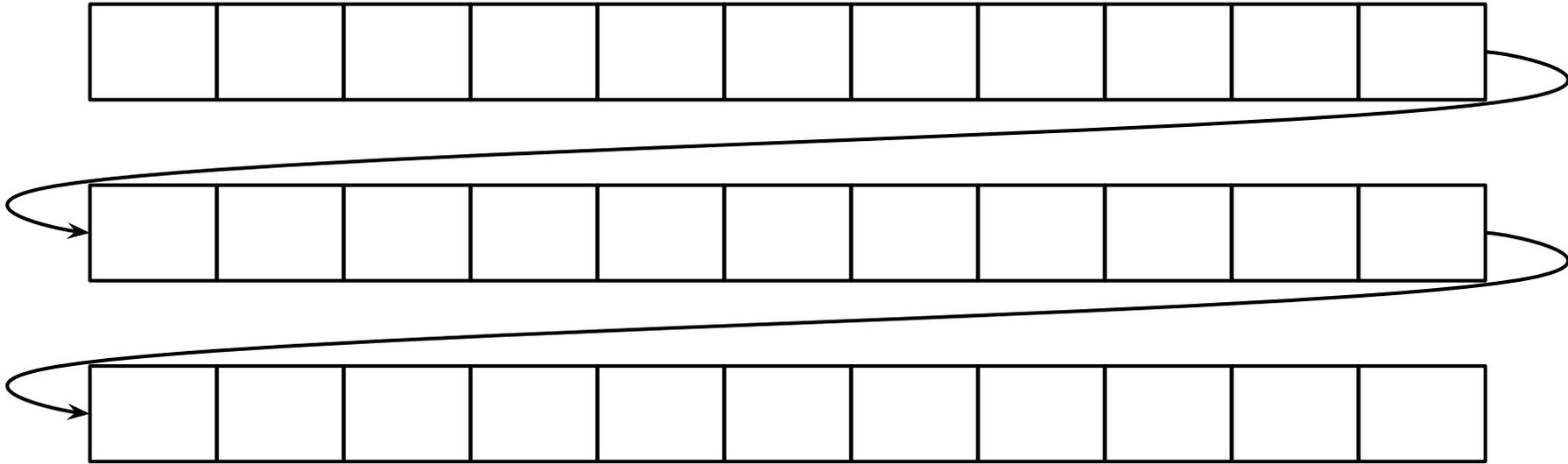
- Quelle différence entre un *octet* et un *byte* ?

Parenthèse: La mémoire simplifiée

- Quelle différence entre un *octet* et un *byte* ?
 - *Octet* = 8 bits (0/1)
 - *Byte* = plus petite unité de mémoire allouable, ce qui dépend des ordinateurs (8 bits en général, mais ça peut varier)
- En général, un *byte* est trop petit. On utilise donc un *int* comme unité de base.
 - Souvent, les calculs sont plus rapides avec un *int*
 - La taille d'un *int* dépend de l'ordinateur
 - En Java, on a une machine virtuelle qui fixe la taille des types pour tous les ordinateurs
 - *byte* = 1 octet
 - *char* = *short* = 2 octets
 - *int* = 4 octets
 - *long* = 8 octets
 - ...

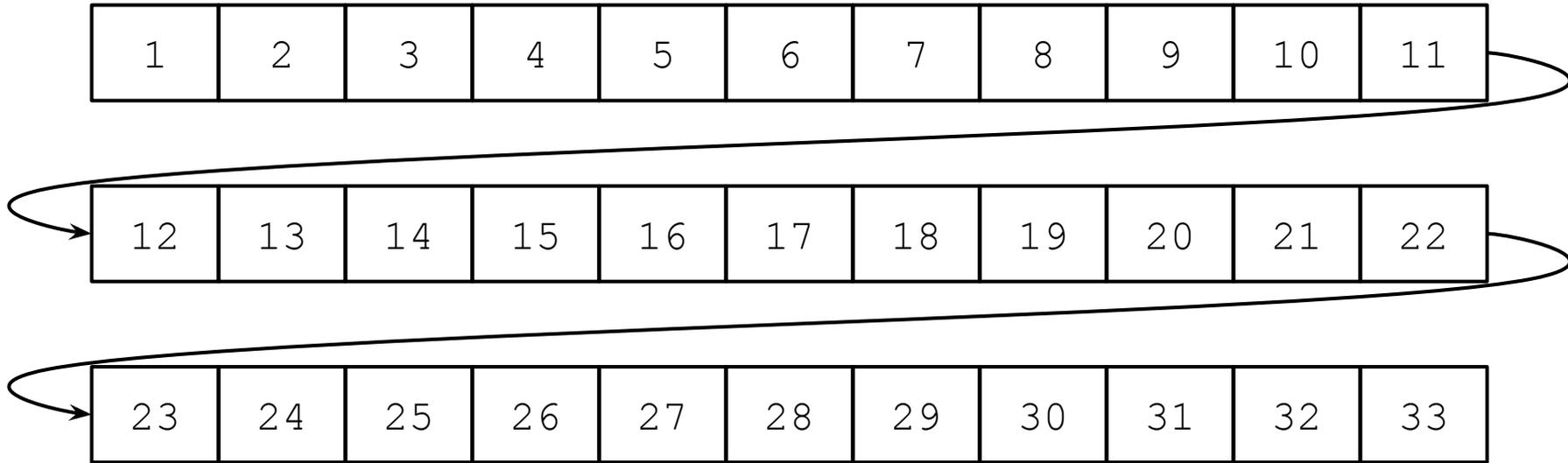
Parenthèse: La mémoire simplifiée

La mémoire est une séquence de “cases” allouables.



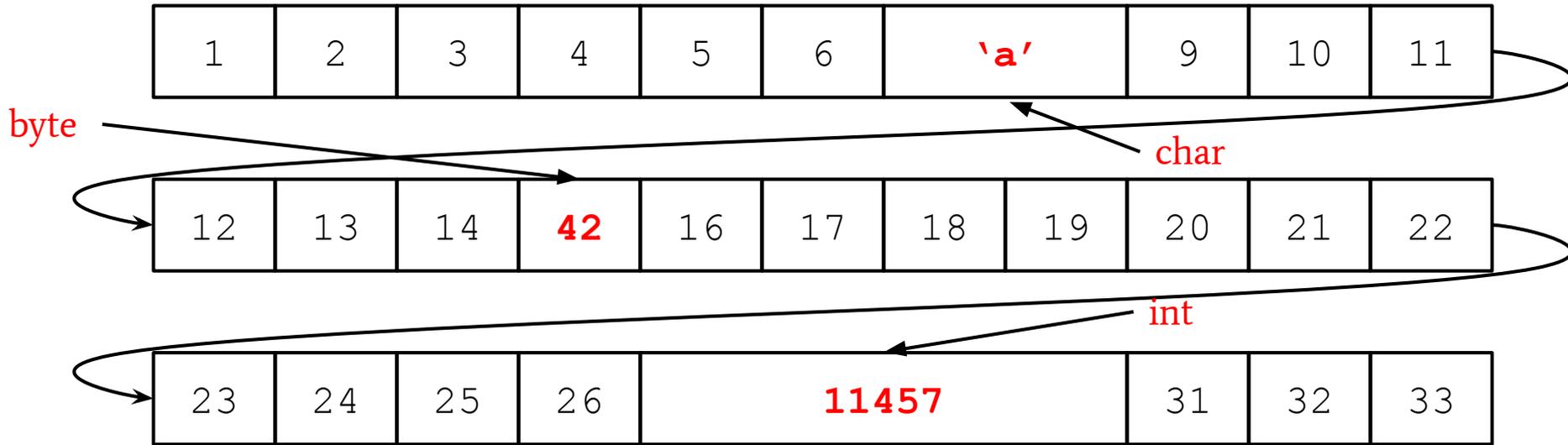
Parenthèse: La mémoire simplifiée

Pour se repérer dans la mémoire, chaque case a une **adresse**



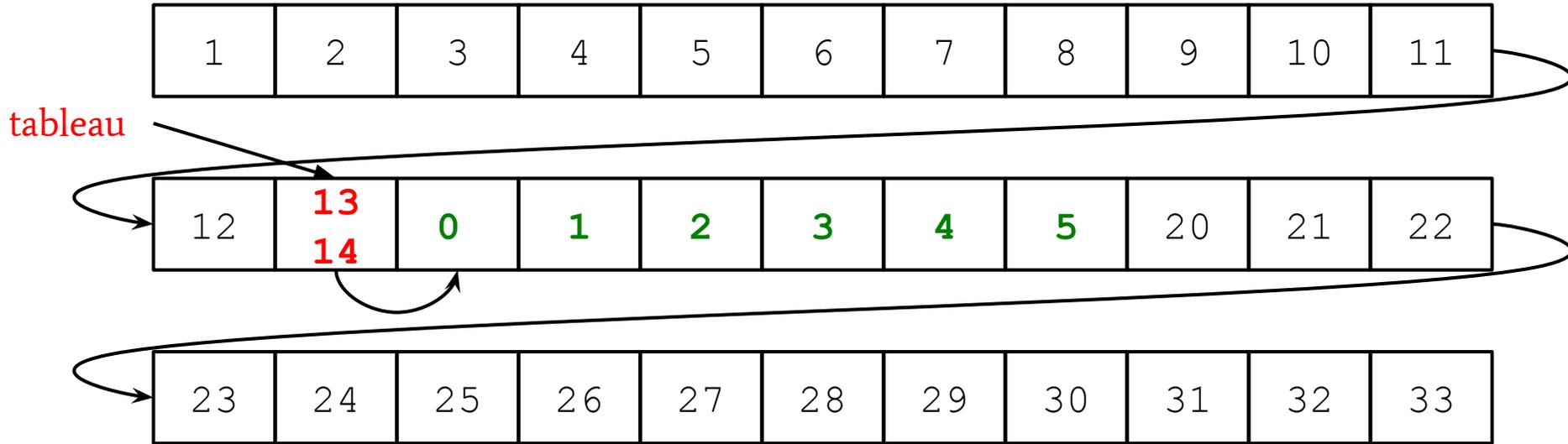
Parenthèse: La mémoire simplifiée

Les **types primitifs** (boolean, char, byte, short, int, long, float, double) ont une taille fixe et représentent **une seule valeur**, les adresses sont cachées



Parenthèse: La mémoire simplifiée

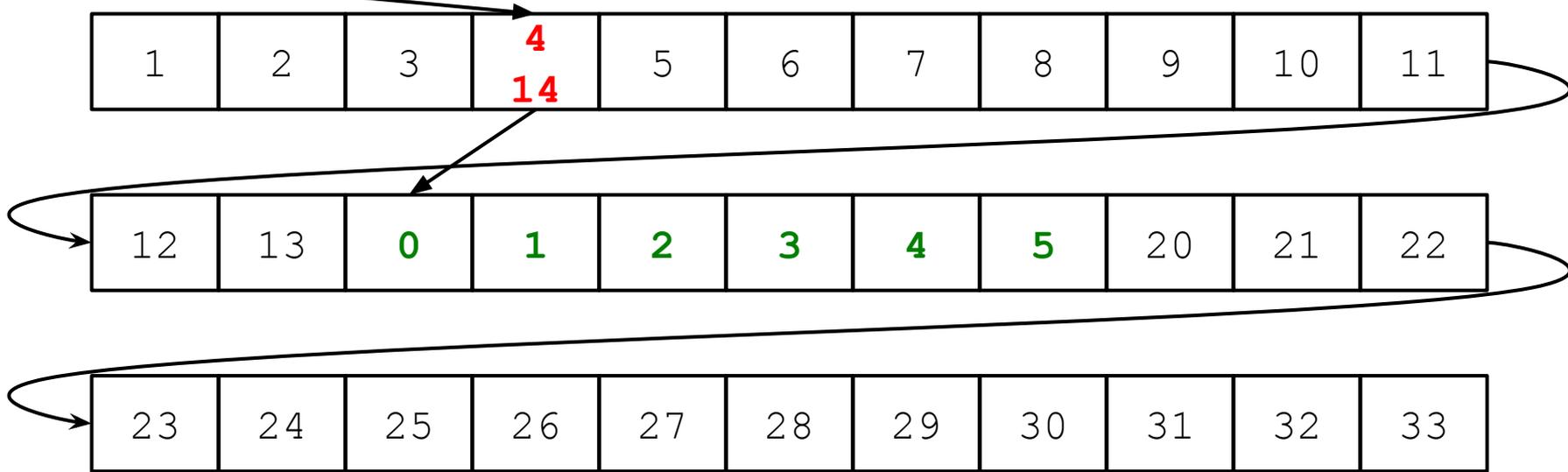
Pour les **types par référence** (ex: les tableaux), la taille des données peut varier. On ne donne donc que l'adresse de début des valeurs, Java se débrouille pour trouver le reste.



Parenthèse: La mémoire simplifiée

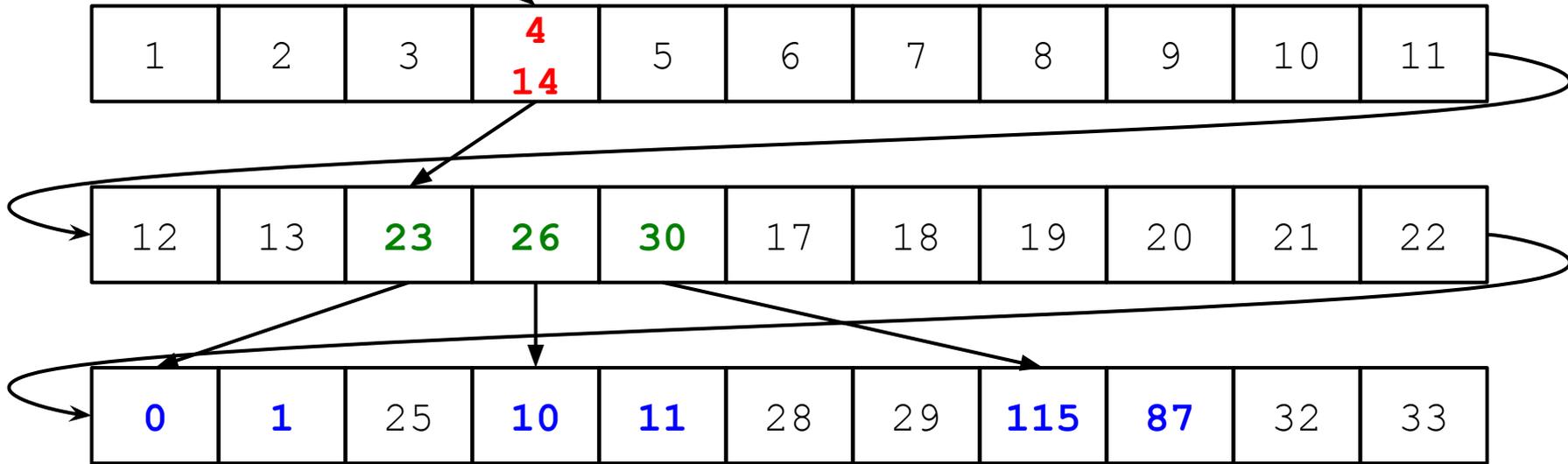
Attention! Les références (= les adresses) sont elles même des valeurs !

tableau



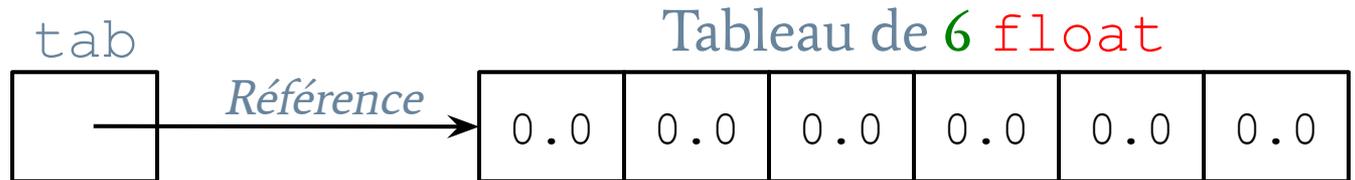
Parenthèse: La mémoire simplifiée

On peut avoir des tableaux de tableaux, i.e. des tableaux de références vers d'autres tableaux (ex: `int[][] tab = new int[3][2];`)



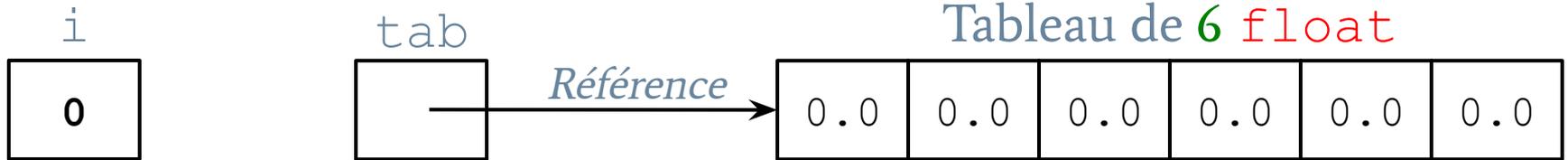
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



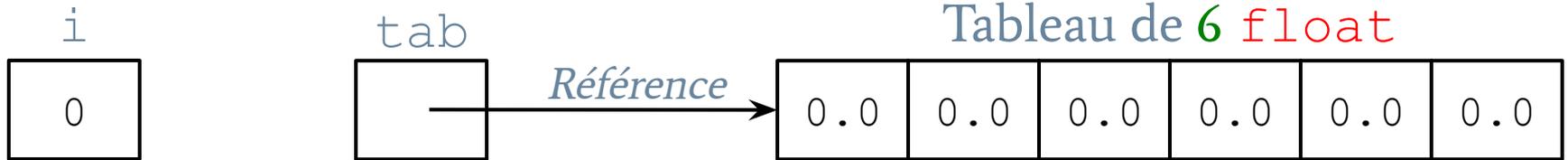
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for (int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



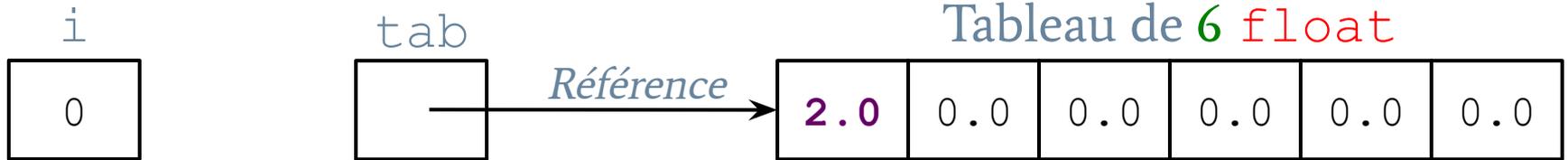
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



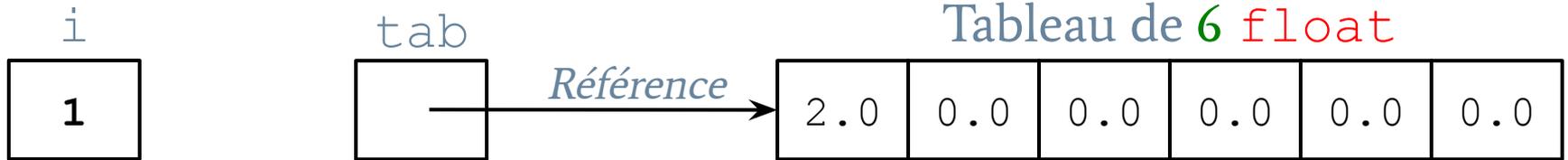
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



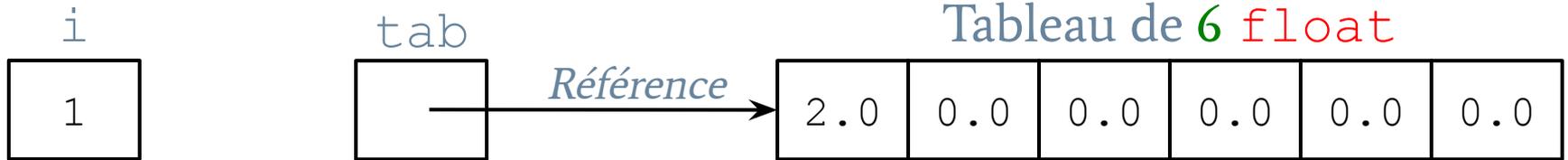
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



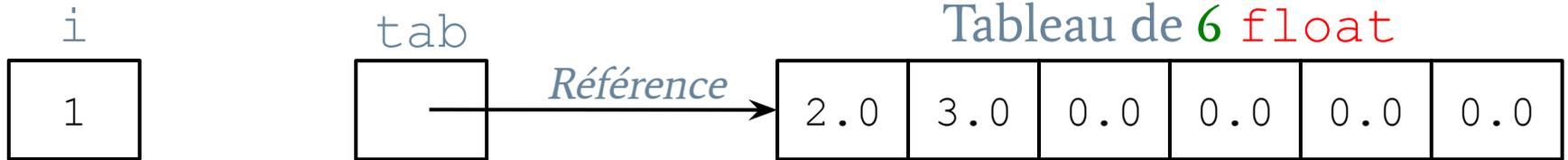
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



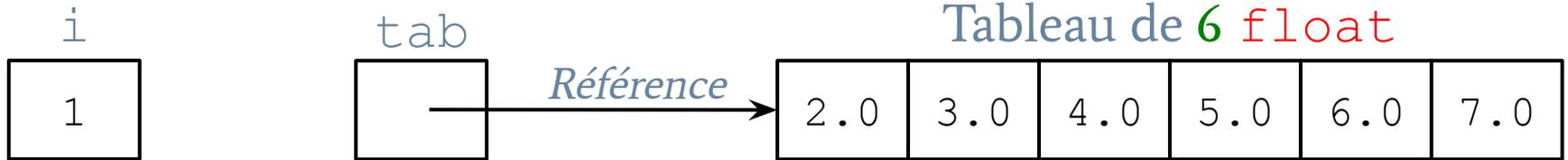
Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```



Exemple d'accès à un tableau

```
public static void main(String[] args) {  
    float[] tab = new float[6];  
    for(int i=0; i<tab.length; i++) {  
        tab[i] = i + 2;  
    }  
}
```

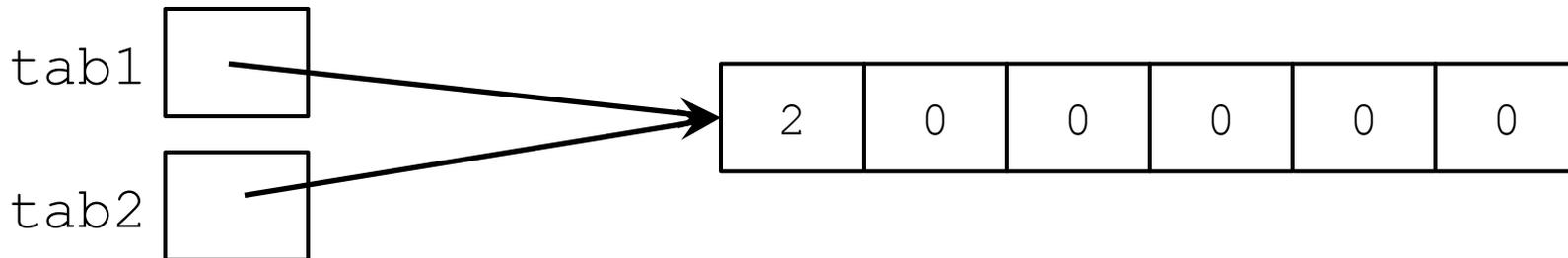


État à la fin de la boucle

Tableaux et aliasing

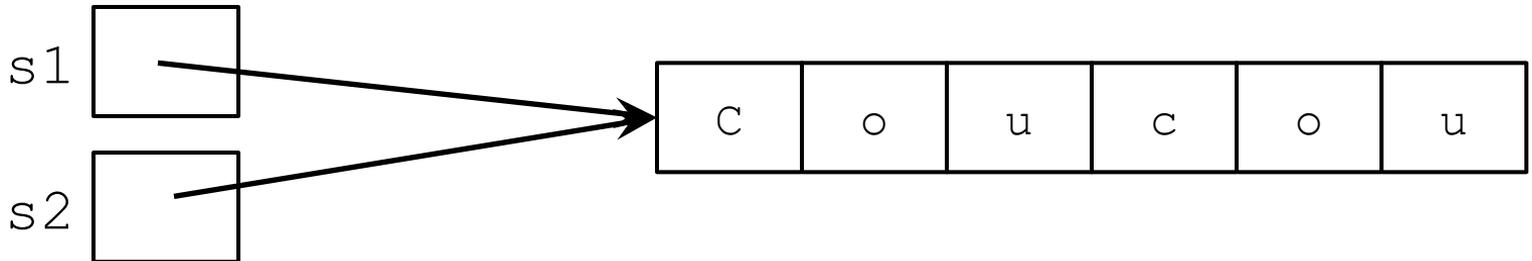
```
public static void main(String[] args) {  
    byte[] tab1 = new byte[6];  
    byte[] tab2 = tab1;  
    tab2[0] = 2;  
    System.out.println("tab1: " + tab1[0]);  
} /* affiche tab1: 2 */
```

- Dans l'exemple précédent, `tab1` et `tab2` sont deux variables différentes, mais elles référencent le même tableau



String et aliasing

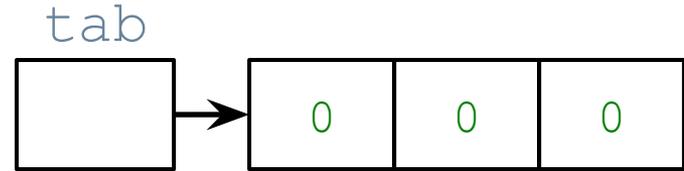
- Comme pour les tableaux, il n'existe en fait pas de variable de type `String` en Java !
- En revanche, `String` déclare une variable **référençant** une zone mémoire typée avec le type `String`
- Exemple : `String s1 = "Coucou"; String s2 = s1;`



Type référence versus type primitif

- Variable de type référence : contient une référence vers une structure de données
 - Tableaux et String
 - Très commun en Java !

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



- Variable de type primitif : contient une valeur
 - boolean, byte, short, char, int, long, float et double
 - On n'a pas de référence pour les types primitifs !

```
int x = 42;
```



Retour sur la méthode `main`

- Dans `public static void main(String[] args)`
 - `args` est une référence vers un tableau de chaînes de caractères correspondant aux arguments du programme
 - Si aucun argument : `args.length` vaut 0
 - Sinon, `args[0]` est le premier argument, `args[1]` le second, ...

Retour sur la méthode `main`

```
class CmdLine {  
    public static void main(String[] args) {  
        for(int i=0; i<args.length; i++) {  
            System.out.println(  
                "args[" + i + "]: " + args[i]);  
        }  
    }  
}
```

```
$ java CmdLine Bonjour monde  
args[0]: Bonjour  
args[1]: monde  
$
```

Notions clés

- Allocation d'un tableau avec
`new type[n]`
- Déclaration d'une variable référençant un tableau avec
`type[] var`
- Accès à un élément avec
`var[indice]`
- L'argument du `main` est le tableau des arguments du programme