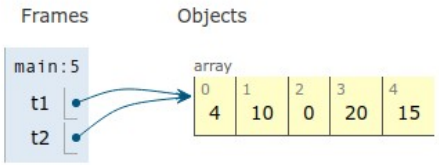
Les tableaux Java

Un tableau, plusieurs références

Un tableau Java n'est pas un type primitif du langage. Il est manipulé à travers une ou plusieurs références :

Java

```
1  public class YourClassNameHere {
2      public static void main(String[] args) {
3        int[] t1 = {4, 10, 0, 20, 15};
4        int[] t2 = t1;
5      }
6  }
```



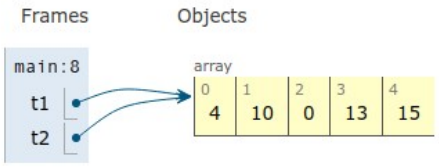
Sur cet exemple `t1` et `t2` désignent le même tableau contenant `{4, 10, 0, 20, 15}`.

Java

```
1  public class YourClassNameHere {
2      public static void main(String[] args) {
3        int[] t1 = {4, 10, 0, 20, 15};
4        int[] t2 = t1;
5
6        t2[3] = 13;
7        System.out.println(t1[3]);
8      }
9  }
```

Print output (drag lower right corner to resize)

```
13
```



Évidemment, dans ces conditions si `t2[3]` est modifié, `t1[3]` l'est aussi car il s'agit du **même** tableau.

Égalité de tableaux

Déterminer que deux tableaux sont identiques n'est pas évident. Il ne suffit pas de comparer leurs références.

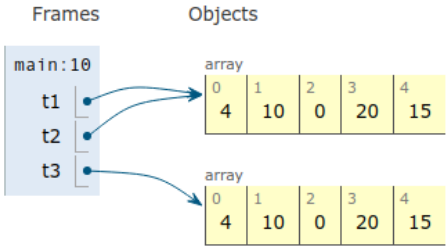
Java

```
1  public class YourClassNameHere {
2      public static void main(String[] args) {
3        int[] t1 = {4, 10, 0, 20, 15};
4        int[] t2 = t1;
5        int[] t3 = {4, 10, 0, 20, 15};
6
7
8        System.out.println("t1 == t2 : " + (t1 == t2));
9        System.out.println("t1 == t3 : " + (t1 == t3));
10     }
11 }
```

Edit this code

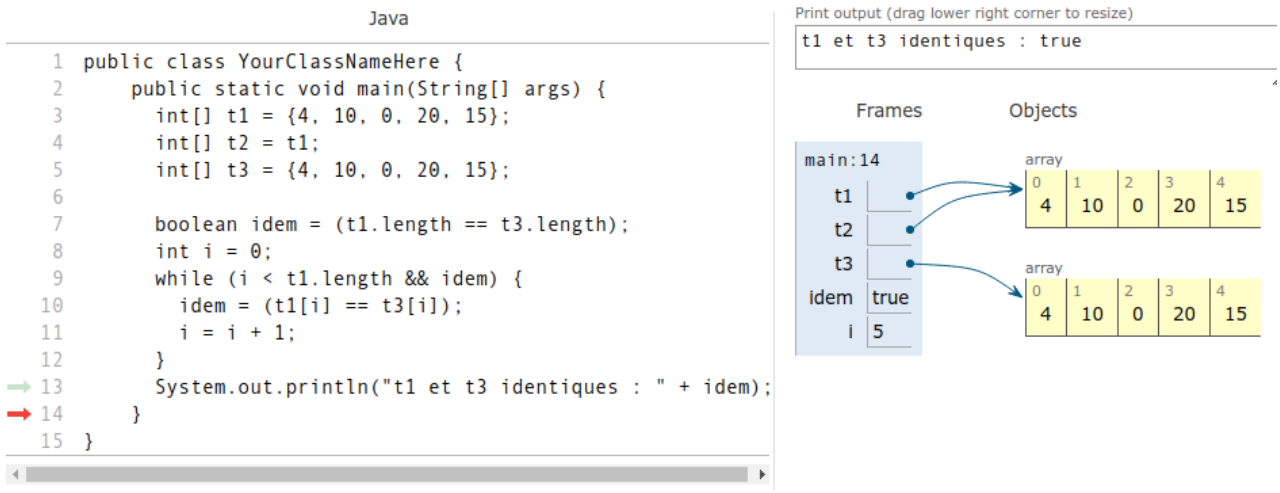
Print output (drag lower right corner to resize)

```
t1 == t2 : true
t1 == t3 : false
```



Là encore le résultat est celui attendu. l'évaluation de l'expression `(t1 == t2)` rend vrai. Il s'agit de la comparaison du contenu de `t1` et de `t2`. Comme les deux variables désignent le **même** tableau. La réponse est donc logiquement vraie (`true`). Par contre, l'évaluation de l'expression `(t1 == t3)` rend faux. En effet, `t1` désigne un tableau, quand `t3` désigne un autre tableau. La

comparaison des deux références ne peut donc que rendre faux (`false`).



Comparer deux tableaux consiste à vérifier qu'ils ont la même longueur :

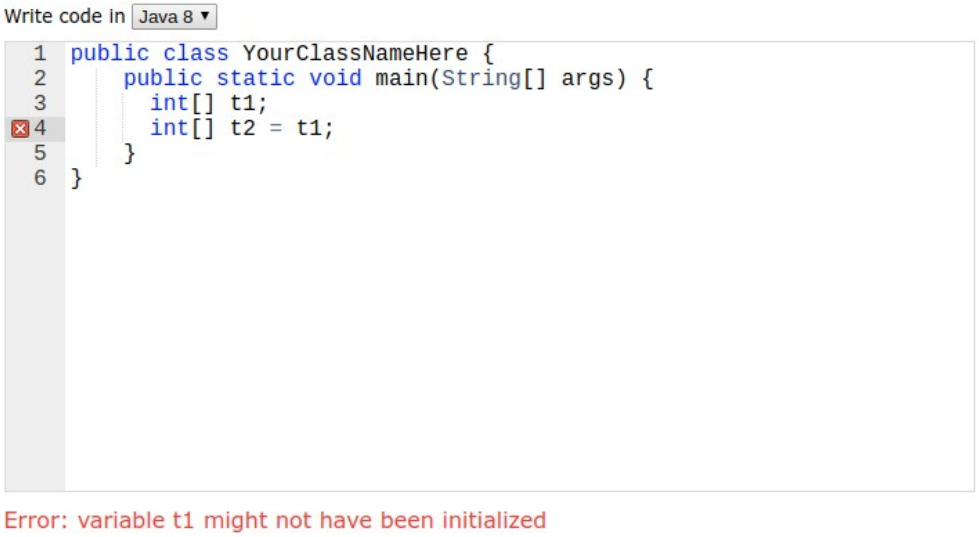
```
boolean idem = (t1.length == t3.length);
```

et le même contenu :

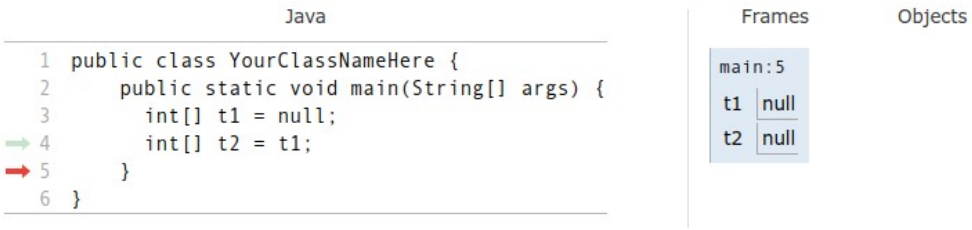
```
int i = 0;
while (i < t1.length && idem) {
  idem = (t1[i] == t3[i]);
  i = i + 1;
}
System.out.println("t1 et t3 identiques : " + idem);
```

Un tableau vide

Une référence sur un tableau peut être `null`. C'est à dire ne désigne aucun tableau.



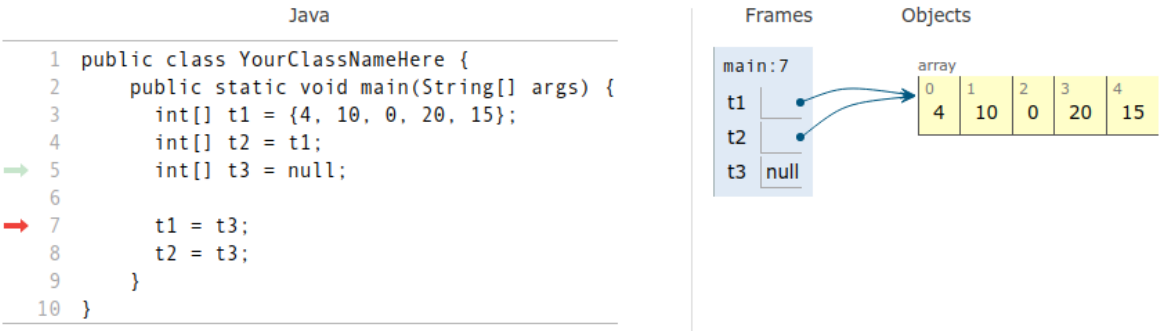
La variable `t1` n'est pas initialisée. Son contenu ne peut donc donc être affecté à une autre variable : « `Error: variable t1 might not have been initialized` »



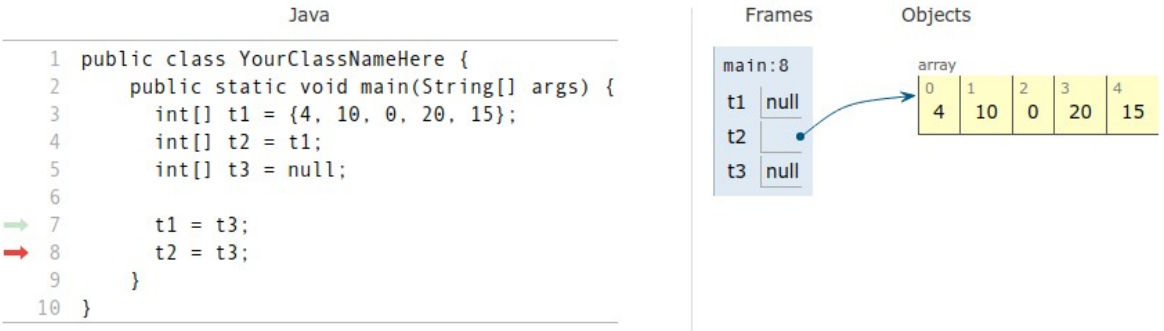
Dans ce cas, `t1` contient `null` et ne réfère donc aucun tableau. Pas contre, il est possible d'affecter le contenu de `t1` à une autre variable, `t2` en l'occurrence.

Un tableau non référencé

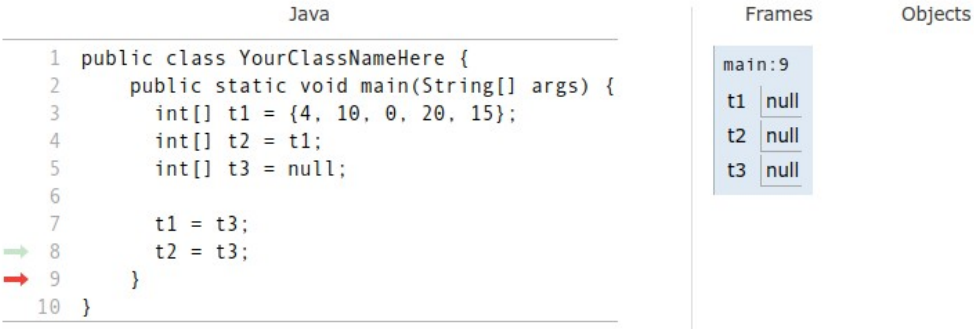
Un tableau peut ne plus être référencé par aucune variable. Que se passe-t-il alors ?



La situation de départ est un tableau `{4, 10, 0, 20, 15}` référencé par deux variables `t1` et `t2`.



Après que `t1` soit mis à `null`, via `t3`, le tableau n'est plus référencé que par `t2`. La situation ne doit étonner personne.



Maintenant que `t2` est `null`, le tableau n'est plus référencé par aucune variable. Il n'est plus accessible. Il est perdu pour le programme. En fait un système dit de ramasse-miettes va récupérer la place mémoire.