

---

## Les variables Java

Nicolas Baudru

mél : [nicolas.baudru@esil.univmed.fr](mailto:nicolas.baudru@esil.univmed.fr)

page web : [nicolas.baudru.perso.esil.univmed.fr](http://nicolas.baudru.perso.esil.univmed.fr)

---

Java est un langage à typage fort...



Lors de la déclaration d'une variable, il faut spécifier le **type** de cette variable. Toute variable de votre programme doit avoir un **type** et un **nom**.

```
int nombre;  
String maChaine;  
Chien fritz;  
Lapin sauteur = new Girafe();
```

Il existe deux sortes de variables :

- ▶ les **types primitifs** qui contiennent des valeurs élémentaires (i.e. de simples suites de bits dans la mémoire). **Exemple** : int, float , boolean.
- ▶ les **références aux objets** qui contiennent ... des références aux objets.  
**Exemple** : String, Chien, Joueur,...

Il existe 8 types primitifs en Java de taille différente. Quand vous déclarez une variable, vous devez spécifier son type (= sa taille = un nombre fixe de bits). Chaque variable contient une valeur et porte un nom.



Une variable peut être vue comme un contenant (un verre par exemple). qui a un nom et une taille.

**Exemple :** Je voudrais un petit verre contenant du café, par analogie devient : Je voudrais une variable de type `int` contenant la valeur 5. Et cette variable se nomme nombre.

| Type                          | Nombre de bits | Intervalle                        |
|-------------------------------|----------------|-----------------------------------|
| <b>booléens et caractères</b> |                |                                   |
| boolean                       | selon la JVM   | true/false                        |
| char                          | 16 bits        | 0 à 65 535                        |
| <b>les entiers</b>            |                |                                   |
| byte                          | 8 bits         | -128 à 127                        |
| short                         | 16 bits        | -32 768 à 32 767                  |
| int                           | 32 bits        | -2 147 483 648<br>à 2 147 483 647 |
| long                          | 64 bits        | trop long à écrire                |
| <b>les décimaux</b>           |                |                                   |
| float                         | 32 bits        | variable                          |
| double                        | 64 bits        | variable                          |

Une affectation, c'est mettre une valeur dans une variable.

Comment affecter une valeur à une variable?

- ▶ taper une valeur littérale après le signe égal : `x = 12` ; `test = true` ;
- ▶ affecter la valeur d'une variable à une autre : `y = x` ;
- ▶ employer une expression combinant les deux : `x = y + 3` ;

Exemple :

```
int taille = 32;
char initiale = 'j'; // ne pas oublier les ' '
double d = 431.123; // le point joue le role de la virgule
boolean test;
test = true; // les booléens et les entiers sont différents
int y = taille + 3;
float = 32.4f; // le f différencie les floats des doubles
```

- ▶ On ne peut pas mettre une grande valeur dans une petite variable :

```
int x = 32;  
byte b = x;
```

Le compilateur n'acceptera pas la seconde affectation, même si 32 est « assez petit » pour entrer dans un byte. En fait, le compilateur ne connaît pas la valeur de  $x$ . Il sait seulement que  $x$  est un entier. Et un entier peut ne pas entrer dans un byte...

- ▶ Il est possible de mettre une petite valeur dans une grande variable :

```
int x = 32;  
long l = x;
```

Intuitivement on peut toujours verser le contenu d'un petit verre dans un grand verre...

Que ce soit pour nommer une variable (de type primitif ou de référence), une méthode ou une classe, vous devez respecter les règles suivantes :

- ▶ Le nom doit commencer par une lettre, « \_ » ou « \$ ».
- ▶ Après le premier caractère, il est aussi possible d'utiliser des chiffres.
- ▶ Le nom ne doit pas être un mot réservé du langage.

Quelques mots-clés Java réservés :

|         |          |           |          |         |            |           |          |
|---------|----------|-----------|----------|---------|------------|-----------|----------|
| boolean | char     | byte      | short    | int     | long       | float     | double   |
| public  | private  | protected | abstract | final   | static     | native    | volatile |
| if      | else     | do        | while    | switch  | case       | default   | for      |
| break   | continue | assert    | class    | extends | implements | interface | import   |
| new     | package  | super     | this     | catch   | finally    | try       | throw    |
| return  | void     | const     | ...      |         |            |           |          |

- ▶ Il n'existe pas de variable objet
- ▶ Il n'y a que des variables références (ou références) à des objets
- ▶ Une variable référence décrit comment accéder à un objet. Elle ressemble à un pointeur ou une adresse mais ce n'est pas un pointeur, ni une adresse.



Une variable primitive contient des bits qui représentent sa valeur. Une variable référence contient des bits qui indiquent comment accéder à l'objet.



## Obtenir un objet

```
Chien monChien = new Chien();
```

- 1 Déclarer une variable référence : `Chien monChien = new Chien();`
- 2 Créer un objet : `Chien monChien = new Chien();`
- 3 Lier l'objet et la référence : `Chien monChien = new Chien();`

## Accéder à une instance/méthode de l'objet : l'opérateur « . »

```
monChien.taille;  
monChien.aboyer();
```

demande à la JVM d'accéder à la variable `taille` ou d'invoquer la méthode `aboyer()` de l'objet pointé par la variable référence `monChien`.

Quelle est la taille d'une référence ?

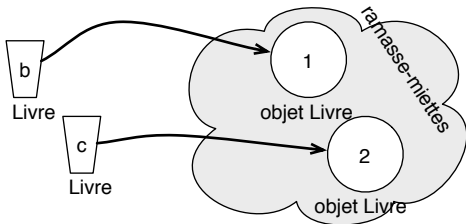
Toutes les références ont-elles la même taille ?

Peut-on effectuer des opérations arithmétiques « à la C » sur les références ?

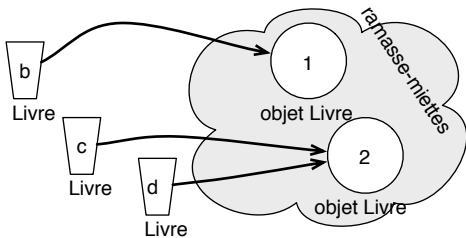
Une référence peut-elle contrôler plusieurs objets ?

Une référence peut-elle ne rien référencer ?

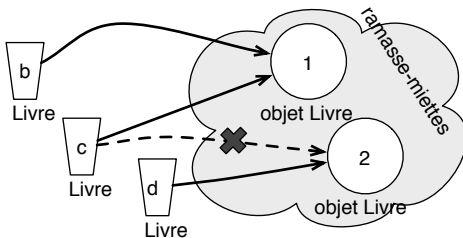
```
Livre b = new Livre();  
Livre c = new Livre();
```



```
Livre d = c;
```

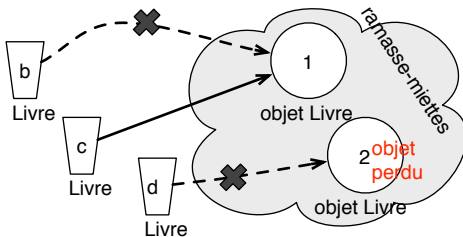


```
c = b;
```



```
b = null;
```

```
d = null;
```



Un tableau est une liste ordonnée d'éléments. Il permet entre autres d'accéder rapidement à un quelconque de ses éléments.

Un tableau est toujours typé et tous ses éléments doivent être du même type ( ou des sous-types ).

**Exemple :** Il est possible de mettre des bytes dans un tableau d'int (conversion implicite), mais il n'est pas possible de mettre des doubles dans un tableau d'int.

Les éléments du tableau sont en fait des variables, de types primitifs ou références. Un tableau ne contient jamais d'objets, mais il peut contenir des références à des objets (références de même type que le tableau).

Un tableau est un objet. Il réside donc dans le tas. Pour y accéder, il faut donc une référence pour ce tableau.

1 On déclare une variable référence pour un tableau de type int :

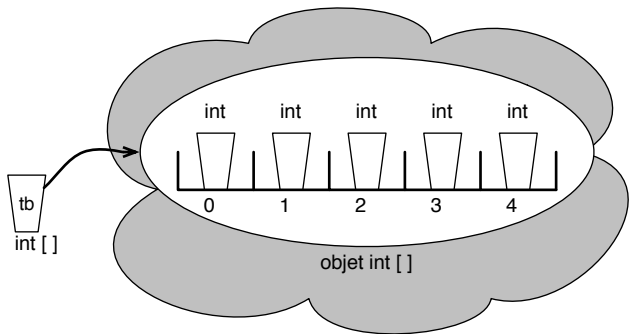
```
int [] tb;
```

2 On crée un tableau d'entiers de 5 éléments et on le lie à la référence tb ;

```
tb = new int [5];
```

3 On donne à chaque élément du tableau une valeur entière.

```
tb[0] = 6;  
tb[1] = 34;  
tb[2] = 12;  
tb[3] = 2;  
tb[4] = 6;
```

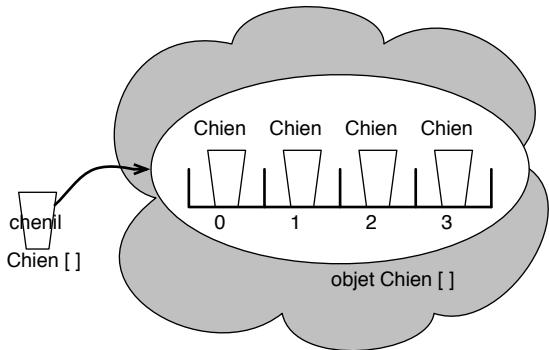


1 On déclare une variable référence pour un tableau de type Chien :

```
Chien [] chenil;
```

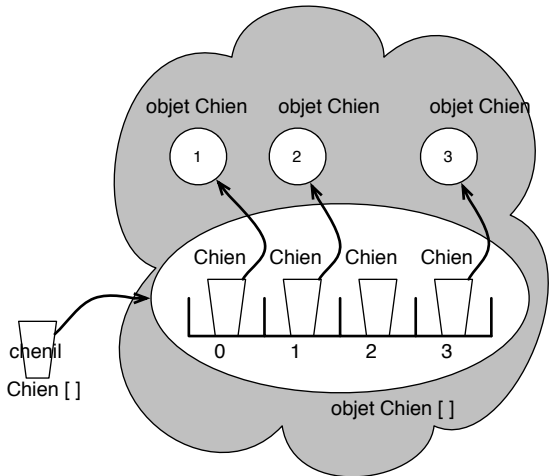
2 On crée un tableau de Chien de 4 éléments et on le lie à la référence chenil ;

```
chenil = new Chien [4];
```



3 On créer les objets Chien et on les lie aux éléments du tableau chenil :

```
chenil[0] = new Chien();  
chenil[1] = new Chien();  
chenil[3] = new Chien();
```



Quelle est la valeur de chenil[2] ?



Pour créer un objet et manipuler ses variables d'instances ou méthodes par l'intermédiaire d'une variable de référence :

```
Chien beethoven = new Chien();  
beethoven.nom = "Beethoven";  
beethoven.aboyer();
```

Et si le Chien est dans un tableau :

```
Chien[] meute = new Chien[6];  
meute[0] = new Chien();  
meute[0].nom = "Beethoven";  
meute[0].aboyer();
```

```
public class TestChien {  
  
    public static void main(String [] args){  
        Chien [] meute = new Chien [3];  
  
        meute [0] = new Chien ();  
        meute [1] = new Chien ();  
        meute [2] = new Chien ();  
  
        meute [0].nom = "Beethoven";  
        meute [1].nom = "Médor";  
        meute [2].nom = "Mirza";  
  
        int i = 0;  
        while ( i < meute.length) {  
            meute [i].aboyer ();  
            i = i + 1;  
        }  
    }  
}
```

```
public class Chien {  
    String nom;  
  
    void aboyer(){  
        System.out.println("Ouah!_Ouaf!");  
    }  
}
```

- ▶ Il existe deux sortes de variables : les types primitifs et les références.
- ▶ Les variables doivent toutes être déclarées avec un type et un nom.
- ▶ Une variable primitive contient des bits représentant sa valeur. Une variable référence contient des bits décrivant comment accéder à un objet sur le tas.
- ▶ Une référence qui a été déclarée pour référencer un type d'objet ne peut référencer que des objets de ce type.
- ▶ Une référence peut référencer plusieurs objets dans le temps, mais un seul objet à un instant donné. (sauf si le mot clé final est utilisé, auquel cas l'affectation est définitive).

- ▶ Une référence peut ne rien référencer du tout : on dit que sa valeur est à null.
- ▶ Un objet peut être référencé par zéro, une ou plusieurs références.
- ▶ Si un objet n'est pas référencé, alors il devient inutilisable (ramasse-miette)
- ▶ Seul un objet référencé est utilisable (via l'opérateur « . »). Si plusieurs références référencent cet objet, alors toutes peuvent être indifféremment utilisées pour manipuler l'objet.
- ▶ Un tableau est toujours un objet. Il peut contenir des types primitifs ou des références. Le type de ses éléments est le même que son propre type.