## Les tableaux en Java

En Java, on utilise :

| | | taille initiale connue *a priori* ? non | taille initiale connue *a priori* ? oui |
|---|---|---|---|
| taille pouvant varier lors de l'utilisation du tableau ? | oui | `ArrayList` | `ArrayList` |
| | non | tableaux de taille fixe | tableaux de taille fixe |

## Les `ArrayList`

Un **tableau dynamique**, est une *collection* de données homogènes, dont *le nombre peut changer* au cours du déroulement du programme, par exemple lorsqu'on ajoute ou retire des éléments au/du tableau.

Les tableaux dynamiques sont définis en Java par le biais du type

`ArrayList`

Pour les utiliser, il faut tout d'abord importer les définitions associées à l'aide de la directive suivante :

```
import java.util.ArrayList;
```

à placer en tout début de fichier

## Déclaration d'un tableau dynamique

Une variable correspondant à un tableau dynamique se déclare de la façon suivante :

```
ArrayList<type> identificateur;
```

où *identificateur* est le nom du tableau et *type* correspond au type des éléments du tableau.

Le type des éléments doit nécessairement correspondre à un **type évolué**.

Exemple :

```
ArrayList<String> tableau;
```

## Initialisation d'un tableau dynamique

Un tableau dynamique initialement vide (sans aucun élément) s'initialise comme suit :

```
ArrayList<type> identificateur = new ArrayList<type>();
```

où *identificateur* est le nom du tableau et *type* correspond au type des éléments du tableau.

Exemple :

```
ArrayList<String> tableau = new ArrayList<String>();
```

## Méthodes spécifiques

Un certain nombre d'opérations sont **directement attachées** au type `ArrayList`.

L'utilisation de ces opérations spécifiques se fait avec la syntaxe suivante :

*nomDeTableau.nomDeMethode(arg1, arg2, ...);*

Exemple :

```
ArrayList<String> prenoms = new ArrayList<String>();

System.out.println(prenoms.size()); // affiche 0
```

## Méthodes spécifiques

Quelques fonctions disponibles pour un tableau dynamique nommé `tableau`, de type `ArrayList<type>` :

`tableau.size()` : renvoie la taille de `tableau` (un entier)

`tableau.get(i)` : renvoie l'élément à l'indice `i` dans le tableau (`i` est un entier compris entre 0 et `tableau.size() - 1`)

`tableau.set(i, valeur)` : affecte `valeur` à la case `i` du tableau (cette case doit avoir été créée au préalable)

## Méthodes spécifiques

Quelques fonctions disponibles pour un tableau dynamique nommé `tableau`, de type `ArrayList<type>` :

`tableau.isEmpty()` : détermine si `tableau` est vide ou non (`boolean`).

`tableau.clear()` : supprime tous les éléments de `tableau` (et le transforme donc en un tableau vide). Pas de (type de) retour.

## Méthodes spécifiques

Quelques fonctions disponibles pour un tableau dynamique nommé `tableau`, de type `ArrayList<type>` :

`tableau.remove(i)` : supprime l'élément d'indice `i`

`tableau.add(valeur)` : ajoute le nouvel élément `valeur` à la fin de `tableau`. Pas de retour.

## Exemple de quelques manipulations de base

```
import java.util.ArrayList;

class ArrayListExemple {
 public static void main(String[] args){
   ArrayList<String> liste = new ArrayList<String>();

   liste.add("un");
   liste.add("deux");

   for(String v : liste) {
    System.out.print(v + " ");
   }

   System.out.println(liste.get(1));

   liste.set(0, "premier");

   }
  }
}
```

## Que faire pour des types de base ?

En Java, à chaque type de base correspond un type évolué prédéfini :

- `Integer` est le type évolué correspondant à `int`
- `Double` est le type évolué correspondant à `double`
- etc.

☞ Utiles dans certains contextes (typiquement les `ArrayList`)

☞ La conversion du type de base au type évolué **se fait automatiquement**

## Exemple

Ecrivons un programme qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```

## Exemple

Ecrivons un programme qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
ArrayList<Integer> vect = new ArrayList<Integer>();

System.out.println("Donnez la taille voulue : ");
int taille = scanner.nextInt();
System.out.println("Saisie de " + taille + " valeurs :");
while (vect.size() < taille) {
  System.out.println("Entrez la valeur " + vect.size() + " : ");
  int val = scanner.nextInt();
  if ((val < 0) && (!vect.isEmpty())) { vect.remove(vect.size() - 1); }
  else if (val == 0)                  { vect.clear();                 }
  else if (val > 0)                   { vect.add(val);                }
}
```

## Comparaison d'éléments

Attention : les éléments d'un tableau dynamique sont toujours des **références**

- ☞ Comparaison au moyen de `equals`

```
ArrayList<Integer> tab = new ArrayList<Integer>();

tab.add(2000);
tab.add(2000);

System.out.println(tab.get(0) == tab.get(1)); // false

System.out.println((tab.get(0)).equals(tab.get(1))); // true
```