

**DEV1 – Laboratoires Java I****TD 10 – Tableaux**

Dans ce TD vous trouverez une introduction aux tableaux.

Les codes sources et les solutions de ce TD se trouvent à l'adresse :

<https://git.esi-bru.be/dev1/labo-java/tree/master/td10-tableaux>

**Table des matières**

<b>1</b>	<b>Créer et manipuler des tableaux</b>	<b>2</b>
<b>2</b>	<b>Tableaux de différents types</b>	<b>3</b>
<b>3</b>	<b>Retourner un tableau</b>	<b>3</b>
<b>4</b>	<b>Tableaux et tests</b>	<b>4</b>
<b>5</b>	<b>Exercices récapitulatifs</b>	<b>5</b>

# 1 Créer et manipuler des tableaux

Un tableau est une collection indexée d'éléments de même type. Un tableau est créé avec une taille donnée et ne peut plus en changer par la suite. Dans l'exemple ci-dessous nous créons et manipulons un tableau de taille 5 contenant des entiers. Comme cela est illustré dans l'exemple, on accède aux éléments d'un tableau par son indice.

```
1 package esi.dev1.td10;
2
3 /**
4  * Exemple d'utilisations des tableaux
5  * @author dev1
6  */
7 public class Tableau {
8
9     /**
10     * Exemple d'utilisations de tableaux d'entier.
11     * @param args pas utilisé.
12     */
13     public static void main(String[] args) {
14         int[] tab = {1, 2, 3, 4, 5};
15         System.out.println("taille: "+tab.length);
16         System.out.println("1er élément: "+tab[0]);
17         System.out.println("dernier élément: "+tab[tab.length-1]);
18         System.out.println("autre élément: "+tab[2]);
19         System.out.println("affiche un drôle de truc: "+tab);
20
21         afficherTab(tab);
22         incrémenter(tab);
23         afficherTab(tab);
24
25         int[] tab2 = new int[10];
26         afficherTab(tab2);
27     }
28
29     /**
30     * Affiche un tableau.
31     * @param tab le tableau à afficher
32     */
33     static void afficherTab(int[] tab) {
34         for (int valeur : tab) { // pour chaque valeur dans le tableau 'tab'
35             System.out.print(valeur+" ");
36         }
37         System.out.println(); // on passe à la ligne
38     }
39
40     /**
41     * Incrémente chaque composante du tableau.
42     * @param tab le tableau à incrémenter.
43     */
44     static void incrémenter(int[] tab) {
45         for (int i = 0; i < tab.length; i++) {
46             tab[i]++;
47         }
48     }
49 }
```

Tableau.java

## Exercice 1 Doubler

Reprenez le code ci-dessus et ajoutez-y une méthode `void multiplier(int[] tab, int valeur)` qui multiplie chaque élément du tableau par la valeur donnée.

### Exercice 2 Afficher

Modifiez la méthode `afficherTab` afin que l’affichage soit par exemple pour un tableau contenant les valeurs 1, 2, 3, 4, 5 :

[1, 2, 3, 4, 5]

## 2 Tableaux de différents types

Les tableaux peuvent contenir des éléments de tous les types. L’exemple suivant illustre l’utilisation d’un tableau de `String`.

```
1 package esi.dev1.td10;
2
3 /**
4  * Exemple d'utilisation de tableaux de chaines
5  * @author dev1
6  */
7 public class TableauChaines {
8     public static void main(String[] args) {
9         String[] mots = {"the", "quick", "brown", "fox", "jumps", "over", "the",
10             "lazy", "dog"};
11
12         for (String mot : mots) { // pour chaque mot dans le tableau 'mots'
13             System.out.print(mot+" "); // affiche le mot suivi d'un espace.
14         }
15         System.out.println(); // passe à la ligne.
16         afficheTableau(mots); // la même chose mais dans une méthode.
17     }
18
19     static void afficheTableau(String[] mots) {
20         for (String mot : mots) {
21             System.out.print(mot+" ");
22         }
23         System.out.println();
24     }
25 }
```

TableauChaines.java

### Exercice 3 Les tailles

Complétez l’exemple ci-dessus en y ajoutant une méthode `void afficherTailles(String[] tab)` qui affiche la taille de chaque `String` d’un tableau. Par exemple si le tableau est celui de l’exemple la méthode affiche 3 5 5 3 5 4 3 4 3.

### Exercice 4 Le plus long mot

Toujours dans l’exemple, ajoutez une méthode `String plusLongMot(String[] tab)` qui reçoit un tableau de `String` et retourne le plus long mot. S’il y a plusieurs mots les plus longs, la méthode retournera n’importe lequel parmi ceux-ci. Dans l’exemple ci-dessus la méthode retournera `quick`, `brown` ou `jumps`.

## 3 Retourner un tableau

Il est bien sûr possible qu’une méthode retourne un tableau. Dans l’exemple suivant la méthode `copie` effectue une copie d’un tableau reçu en paramètre. La méthode `pasUneCopie` illustre une erreur fréquente lorsque l’on débute dans l’apprentissage des tableaux en Java.

```

1 package esi.dev1.td10;
2
3 /**
4  * Exemples de méthodes retournant un tableau.
5  * @author dev1
6  */
7 public class TableauCopie {
8
9     public static void main(String[] args) {
10         int[] tableau = {1, 2, 3, 4, 5};
11         afficherTab(tableau);
12         int[] copie1 = copie(tableau);
13         int[] copie2 = pasUneCopie(tableau);
14         tableau[0] = 9;
15         System.out.print("tableau: ");
16         afficherTab(tableau);
17         System.out.print("copie1: ");
18         afficherTab(copie1);
19         System.out.print("copie2: ");
20         afficherTab(copie2);
21     }
22
23     static int[] copie(int[] tab) {
24         int[] copie = new int[tab.length];
25         for (int i = 0; i < tab.length; i++) {
26             copie[i] = tab[i];
27         }
28         return copie;
29     }
30     static int[] pasUneCopie(int[] tab) {
31         int[] copie = tab;
32         return copie;
33     }
34     static void afficherTab(int[] tab) {
35         for (int element : tab) {
36             System.out.print(element + " ");
37         }
38         System.out.println();
39     }
40 }

```

TableauCopie.java

### Exercice 5 Miroir

Complétez l'exemple ci-dessus en y ajoutant une méthode `int miroir(int[] tab)` qui retourne un nouveau tableau dont les éléments sont les éléments du tableau reçu en paramètre mais en ordre inverse. Par exemple si le tableau est celui de l'exemple la méthode retourne un tableau contenant les entiers 5, 4, 3, 2, 1.

## 4 Tableaux et tests

La manipulation des tableaux n'est pas aisée, les tests JUnit sont un outil essentiel afin de s'assurer de la justesse de nos méthodes.

La méthode `assertArrayEquals` de la classe `org.junit.Assert` permet de s'assurer que deux tableaux sont strictement égaux : de même taille, de même type et que tous leurs éléments sont égaux et apparaissent dans le même ordre.

```

1 package esi.dev1.td10;
2
3 import org.junit.Test;

```

```

4 import static org.junit.Assert.*;
5
6 public class TableauTest {
7
8     @Test
9     public void testIncrémenter() {
10         int[] tab = {1, 2, 3};
11         Tableau.incrémenter(tab);
12         int[] tabAttendu = {2, 3, 4};
13         assertEquals(tabAttendu, tab);
14     }
15
16     @Test
17     public void testIncrémenterTableauVide() {
18         int[] tab = {};
19         Tableau.incrémenter(tab);
20         int[] tabAttendu = {};
21         assertEquals(tabAttendu, tab);
22     }
23 }
24

```

TableauTest.java

### Exercice 6 Tests

Créer des tests JUnit pour les méthodes des exercices 1, 4 et 5. Pour chacune de ces méthodes testez un cas général ainsi que les cas limites (tableau vide, maximum en début/fin de tableau, valeurs négatives, etc).

## 5 Exercices récapitulatifs

### Exercice 7 TableauUtil

Dans une classe `TableauUtil` écrivez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- ▷ `static double min(double[] tab)` qui retourne le minimum du tableau passé en paramètre ;
- ▷ `static double max(double[] tab)` qui retourne le maximum du tableau ;
- ▷ `static double somme(double[] tab)` qui retourne la somme des éléments du tableau ;
- ▷ `static double moyenne(double[] tab)` qui retourne la moyenne des éléments du tableau ;
- ▷ `static double[] copie(double[] tab)` qui retourne une copie du tableau passé en paramètre ;

### Exercice 8 TableauUtil (suite)

Dans la classe `TableauUtil` ajoutez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- ▷ `int indiceMax(double[] tab)` : retourne l'indice du maximum du tableau passé en paramètre ;
- ▷ `boolean estTrié(double[] tab)` : retourne vrai si le tableau est trié par ordre croissant et faux sinon ;
- ▷ `int indice(int[] tab, int valeur)` : retourne l'indice de cette valeur dans le tableau. Que retournez-vous si la valeur n'apparaît pas ? Indiquez-le dans la javadoc.

- ▷ `boolean contient(String[] tab, String mot)` : retourne vrai si un tableau de `String` contient un mot passé en paramètre ;  
Astuce : utiliser la méthode `equals` afin de tester l'égalité entre deux `String` et non pas l'opérateur `'=='`.
- ▷ `boolean contient(double[] tab, double valeur, double erreur)` : retourne vrai si un élément du tableau est proche de la `valeur` à l'`erreur` près, c'est-à-dire si la distance entre l'élément et la valeur est inférieure à `erreur`.

### Exercice 9 TableauUtil (suite de la suite)

Dans la classe `TableauUtil` ajoutez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- ▷ `static void inverser(double[] tab)` qui inverse l'ordre des éléments du tableau passé en paramètre ;
- ▷ `static boolean doublons(int[] tab)` qui retourne vrai si le tableau possède des doublons (2 fois la même valeur) et faux sinon ;
- ▷ `static int nbNégatifs(double[] tab)` qui retourne le nombre d'éléments négatifs du tableau ;
- ▷ `static void échange(double[] tab, indice1, indice2)` qui échange la valeur se trouvant à l'indice `indice1` avec la valeur se trouvant à l'indice `indice2` ;
- ▷ `static int[] indicesMin(double[] tab)` qui retourne un tableau contenant les indices des minimums.

### Exercice 10 Initialisation par défaut

Dans la méthode principale d'une classe `TestInit` :

- ▷ créez un tableau d'entier de taille 10 et afficher-le (comme à la ligne 18 de la classe `Tableau` ci-dessus).  
Quelle valeur par défaut initialise chacune des cases du tableau ?
- ▷ créez un tableau de double de taille 10 et afficher-le ;
- ▷ créez un tableau de 10 booléens et afficher-le ;
- ▷ créez un tableau de 10 `String` et afficher-le.