Cours de Java : Les tableaux Chapitre 2

Un tableau est une variable destinée à contenir **plusieurs éléments de même type**.
On peut se représenter un tableau comme un ensemble de cases adjacentes dans lesquelles se trouvent des données : les éléments du tableau. Chaque élément du tableau est repéré par un **indice** précisant sa position au sein de l'ensemble.
En java :
- les tableaux sont des **objets**;
- quand on crée un objet tableau, il faut obligatoirement préciser le **type** et le **nombre** des éléments contenus dans le tableau.

1. Création d'un tableau initialisé explicitement (sans utiliser l'opérateur new)

1.1 Exemples

```
int [] tab1 = {4, 8 , 10};                  //(*1)
double tab2 [] = {1.3, 5.6, 7.8, 4.75};     //(*2)
String [] tab3 = {"toto", "tata"};          //(*3)
char tab4 [] = {'a', 'b', 'c', 'd'};        //(*4)
```

(*1) : déclaration et création **d'un objet tableau nommé tab1 qui contient 3 éléments de type int initialisés à 4, 8 et 10.** Quand on déclare un tableau, il faut commencer par indiquer le type des éléments du tableau. Les **crochets []** indiquent qu'il s'agit d'un tableau : ils peuvent être placés avant ou après le nom du tableau (il est **préférable de placer les crochets avant le nom**). Après l'opérateur =, les valeurs initiales des éléments du tableau sont placées entre accolades {} et séparées par des virgules. Le **nombre de valeurs initiales** correspond **au nombre d'éléments du tableau**; le nombre d'éléments du tableau correspond à sa **taille,** ou à sa **dimension**. Les **valeurs initiales** doivent correspondre à des **constantes du type des éléments** du tableau.

Cette instruction **réserve l'emplacement mémoire** pour **3 éléments** de type int. Chaque élément est repéré par sa « position » dans le tableau, nommée **indice**. Conventionnellement, **la 1ère position porte le numéro 0.** Ici, les indices des éléments du tableau vont donc de 0 à 2.

L'expression tab1[i] désigne un élément dont la position dans le tableau est fournie par la valeur de i et **joue le même rôle qu'une variable de type int.**
Ici, les 3 éléments du tableau seront donc désignés par tab1[0], tab1[1] et tab1[2].

Représentation du tableau tab1 :

Indice	Valeur
0	4
1	8
2	10

(*2) : déclaration et création d'un objet tableau nommé tab2 qui contient 4 éléments de type double initialisés à 1.3, 5.6, 7.8 et 4.75. Cette nstruction **réserve l'emplacement mémoire** pour **4 éléments** de type double. L'expression tab2[0] est **de type double** (type simple ou primitif) et permet **d'accéder en consultation et en modification** à l'élément d'indice 0 du tableau tab2. tab2[0] est initialisé avec la constante réelle 1.3.

(*3) : déclaration et création d'un objet tableau nommé tab3 qui contient 2 éléments de type String initialisés à "toto" et "tata". Cette instruction **réserve l'emplacement mémoire** pour **2 objets** de type

String. tab3[1] est un "**objet" de type String**, plus exactement une variable référence qui pointe ou contient l'adresse mémoire d'un objet contenant la chaine de caractères "tata".

(*4) : déclaration et création d'un objet tableau nommé tab4 qui contient 4 éléments de type char initialisés à 'a', 'b', 'c' et 'd'. Cette instruction **réserve l'emplacement mémoire** pour **4 éléments** de type char. L'expression tab4[3] est **de type char** (type simple ou primitif) et permet **d'accéder en consultation et en modification** à l'élément d'indice 3 du tableau tab4. tab4[3] est le dernier élément du tableau tab4 et est initialisé avec la constante caractère 'd'.

Rappel : les constantes caractère s'écrivent entre simples quotes ou apostrophes (ex 'r') alors que les constantes chaînes de caractères s'écrivent entre guillemets doubles (ex : "hello").

2. Utilisation d'un tableau

Un élément d'un tableau **peut être utilisé comme une n'importe quelle variable du même type.**

Un **indice** peut prendre la forme de n'importe quelle expression arithmétique de **type entier.**
Par exemple, si `n`, `p`, `k` et `j` sont de type `int`, ces notations sont correctes :

```
tab2[n-3]
tab4[3*p-2*k+j%l]
```

Exemple :

```
int [] tab1 = {4, 8 , 10};
double [] tab2  = {1.3, 5.6, 7.8, 4.75};
String [] tab3 = {"toto", "tata"};
char [] tab4  = {'a', 'b', 'c', 'd'};

int j = 5, k = 3;

System.out.println(tab1[2]);    //affichage de la valeur du dernier élément de tab1 : tab1[2]
System.out.println(tab4[0]);    //affichage de la valeur de tab4[0]
tab2[0] = 79.6;                 // on affecte 79.6 dans tab2[0]
tab1[1]--;                      // on décrémente la valeur de tab1[1]
System.out.println(tab1[0]);    //affichage de la valeur de tab1[0]
System.out.println(tab1[1]);    //affichage de la valeur de tab1[1]
tab4[2] += 6;                   // on ajoute 6 à tab4[2], donc tab4[2] contient le code ASCII de
                                // la lettre 'i'

tab3[0] = "hello";
System.out.println(tab4[j - k]);    //affichage de la valeur de tab4[2]
System.out.println(tab3[0]);    //affichage de la valeur de tab3[0]
System.out.println(tab4[4]);     //plantage lors l'exécution (*)
```

(*) : lorsque l'on accède à un élément d'un tableau, l'interpréteur de commandes java contrôle la validité de l'indice. En cas de débordement d'indice (indice < 0 ou >= taille du tableau), l'erreur ou l'exception ArrayIndexOutOfBoundsException est levée.

Q1 Tester le programme ci-dessus.

2.1 La propriété length

Tout tableau possède une propriété (i.e. un attribut ou une variable membre) nommée **length** qui stocke la **taille** du tableau (dans un entier).

Exemple (qui utilise les tableaux créés précédemment) :

```
int lg;                     // déclaration d'une variable de type int nommée lg
```

```
lg = tab1.length;          // on accéde à la propriété length du tableau tab1 en utilisant l'opérateur "."
                           // la valeur de cette propriété est affectée dans la variable lg (lg vaut 3)
System.out.println(lg);                  // affichage de la valeur contenue dans lg
System.out.println(tab2.length);         // ou directement : affichage de la taille de tab2
```

Q2 Tester le code ci-dessus.

Remarque : **l'opérateur + permet de concaténer (i.e. mettre bout à bout) des chaînes de caractères.**

Exemples :

```
String str = "bonjour " + "tout le monde";
System.out.println(str);

str += "!";
System.out.println(str);
System.out.println(str + "!");  // le système "ajoute" la chaîne "!" à str et affiche le résultat
                                // str n'est pas modifiée !

int j = 5;
System.out.println(str + j);    // le système "ajoute" j (après l'avoir converti en chaine de caractères)
                                // à str et affiche le résultat
                                // str n'est pas modifiée !
System.out.println("Valeur de j : "+ j);
```

Q3 Tester le code ci-dessus.

2.2 Parcours d'un tableau

Bien souvent, on souhaite appliquer le même traitement à tous les éléments, ou à une partie des éléments d'un tableau. Dans ce cas, on utilise une **boucle for**.

2.2.1 La boucle for "classique"

Exemple (qui utilise les tableaux créés précédemment) :

```
int i;                          // déclaration d'une variable de boucle nommée i

System.out.println("Affichage de tous les éléments du tableau tab1:");
for (i = 0; i < tab1.length ; i++)
{
        /*traitement effectué pour i  = 0, puis i = 1, et enfin pour i = 2*/
        System.out.println("\tValeur à l'indice " + i + " : " + tab1[i]);
}
System.out.println();   // saute une ligne
/*A ce point de d'exécution du programme i vaut 3.*/
System.out.println("Valeur de i  : "  + i);                  // vérification

System.out.println("Modification des 3 derniers éléments du tableau tab2:");
for (i = tab2.length -3; i < tab2.length ; i++)
        tab2[i] = 2*i + 0.5;

System.out.println("Affichage de tous les éléments du tableau tab2:");
for (i = 0; i < tab2.length ; i++)
        System.out.println("\tValeur à l'indice " + i + " : " + tab2[i] );
System.out.println();   // saute une ligne
```

Q4 Tester le code ci-dessus.

2.2.2 Une autre syntaxe de la boucle for disponible en java

Exemple (qui utilise les tableaux créés précédemment) :

```
System.out.println("Affichage de tous les éléments du tableau tab4:");
for (char c : tab4)        //(*)
{
        /*la variable c prend successivement toutes les valeurs stockées dans tab4 : tab4[0], tab4[1],
        tab4[2], et tab4[3] : le traitement suivant est réalisé pour toutes les valeurs du tableau*/
        System.out.print(c);     // affichage de la valeur stockée dans c
        System.out.print(" ");
}
System.out.println();   // saute une ligne
```

(*) : entre les parenthèses qui suivent le mot clé "for", on déclare une variable qui **doit être du type des éléments du tableau** que l'on souhaite "parcourir". Cette déclaration doit être suivi du symbole":", et du nom du tableau à parcourir.

Q5 Tester le code ci-dessus.

Q6 Utiliser cette syntaxe de la boucle "for" afin d'afficher tous les éléments du tableau tab3.

Remarque : cette syntaxe de la boucle "for" ne permet que d'**accéder en consultation** aux éléments d'un tableau : elle ne permet pas de les modifier.

Illustration :

```
/*Boucle qui ne sert à rien !*/
for (char c : tab4)
{
        c = 't';
        /*lors de la 1ère itération, la valeur de tab4[0] est affectée dans c, puis 't' est affecté dans
        c, mais tab4[0] lui n'est pas modifié !*/
}


System.out.println("Affichage de tous les éléments du tableau tab4:");
for (char c : tab4)
{
        System.out.print(c);     // affichage de la valeur stockée dans c
        System.out.print(" ");
}
System.out.println();   // saute une ligne
```

Q7 Tester le code ci-dessus.

3. Création d'un tableau avec l'opérateur new

Dans ce cas, **par défaut**, tous les éléments du tableau sont initialisés à :

- 0 quand il s'agit d'un tableau d'entiers, de réels ou de char;
- false quand il s'agit d'un tableau de booléens;
- null quand il s'agit d'un tableau d'objets.

3.0 Le mot clé "null"

Le mot « null » est un mot clé du langage java. Il signifie "référence nulle", "adresse nulle" ou "pointeur nul". Quand une variable référence contient la valeur "null", cela signifie qu'elle contient l'adresse nulle. Dans ce cas, comme elle ne contient pas la référence d'un objet "valide", on ne peut pas l'utiliser pour accéder à un membre de l'objet (une méthode par exemple).

Exemple :

```
String str;                 //déclaration de l'objet nommé str (*)
str = null;                 //on affecte null dans str;
char c;                     // déclaration d'une variable, de type char, nommée c
c = str.charAt(0);          // provoque une erreur "NullPointerException" à l'exécution
```

(*) : la déclaration de l'objet correspond plus exactement à la **déclaration d'une variable référence** nommée str destinée à contenir l'adresse mémoire (ou la référence) d'un objet de type classe String.

Q8 Tester le code ci-dessus.

Si str contient la référence d'un String valide au moment où on appelle la méthode charAt() (pour l'objet str), l'erreur "NullPointerException" disparait. Cf. exemple ci-dessous :

```
String str;                 //déclaration de l'objet nommé str
str = "hello";              //on affecte "hello" dans str;
char c;                     // déclaration d'une variable, de type char, nommée c
c = str.charAt(0);          // le caractère 'h' est affecté dans la variable c
```

Q9 Tester le code ci-dessus.

3.1 Exemples

Exemple 1 :

```
int [] tabI;                //déclaration d'un objet tableau de int nommé tabI (*1)
tabI = new int [6];         //création d'un tableau contenant 6 int initialisés à 0 (*2)

System.out.println("Affichage de tous les éléments du tableau tabI:");
for (int i=0; i<tabI.length; i++)
{
        System.out.print(tabI[i] );    // affichage de la valeur stockée dans tabI[i]
        System.out.print(" ");
}
System.out.println();// saute une ligne
```

(*1) : la **déclaration** de l'objet correspond plus exactement à la **déclaration d'une variable référence** nommée tabI destinée à contenir l'adresse mémoire (ou la référence) d'un tableau de int.

(*2) : l'opérateur **new permet de créer un objet, ce qui implique la réservation de l'espace mémoire occupé par cet objet.** Ici, l'objet créé est un tableau contenant 6 éléments de type int; ils sont tous initialisés à 0. Après avoir créer l'objet, **l'opérateur new renvoie l'adresse mémoire de l'objet créé et cette dernière est affectée dans tabI.**

Exemple 2 :

```
double [] tabD;             //déclaration d'un objet tableau de doubles nommé tabD (*1)
tabD = new double [6];      //création d'un tableau contenant 6 doubles initialisés à 0.0 (*2)
```

```
System.out.println("Affichage de tous les éléments du tableau tabD:");
for (double d : tabD)
{
        System.out.print(d);    // affichage de la valeur stockée dans d
        System.out.print(" ");
}
System.out.println();   // saute une ligne
```

(*1) : la **déclaration** de l'objet correspond plus exactement à la **déclaration d'une variable référence** nommée tabD destinée à contenir l'adresse mémoire (ou la référence) d'un tableau de doubles.

(*2) : l'opérateur **new permet de créer un objet, ce qui implique la réservation de l'espace mémoire occupé par cet objet.** Ici, l'objet créé est un tableau contenant 6 éléments de type double; ils sont tous initialisés à 0.0. Après avoir créer l'objet, **l'opérateur new renvoie l'adresse mémoire de l'objet créé et cette dernière est affectée dans tabD.**

Remarque : il est possible d'utiliser une seule instruction pour déclarer et créer le tableau :

```
double [] tabD = new double [6];
```

Exemple 3 :

```
String [] tabS;                 //déclaration d'un objet tableau de String nommé tabS
tabS = new String [3];          //création d'un tableau contenant 3 String initialisés à null

tabS[2] = "hello";
System.out.println("Affichage de tous les éléments du tableau tabS:");
for (String str : tabS)
{
        System.out.print(str);  // affichage de la valeur de str
        System.out.print(" ");
}
System.out.println();   // saute une ligne
```

Remarque : il est possible d'initialiser explicitement les éléments d'un tableau en utilisant l'opérateur new.
Exemple 3 :

```
String [] tabS2 = new String []{"toto", "tata"};
```

L'instruction précédente est équivalente à :

```
String [] tabS2 = {"toto", "tata"};
```

Q10 Tester les codes ci-dessus.

4. Exercices

Exercice 1 :
Réaliser un programme qui :

- déclare un tableau (pas un String) de 10 caractères en l'initialisant lors de sa déclaration avec les 10 premières lettres de l'alphabet;
- affiche les 10 caractères contenus dans le tableau.

Rappel : Les constantes de type caractère doivent être notées entre **quotes (ou apostrophes)**.

Exemple : 'r'

Exercice 2 :
Réaliser un programme qui :
- demande à l'utilisateur de saisir 8 valeurs entières et les range dans un tableau ;
- demande à l'utilisateur de saisir un (autre) entier nommé *val;*
- affiche le nombre de fois où *val* est présent dans le tableau.

Exercice 3 :
Réaliser un programme qui :
- demande à l'utilisateur de saisir 8 valeurs entières et les range dans un tableau ;
- détermine et affiche les valeurs minimale et maximale.

Exercice 4 :
Réaliser un programme qui :
- demande à l'utilisateur de saisir 10 valeurs entières et les range dans un tableau ;
- demande à l'utilisateur de saisir 2 entiers Val1 et Val2;
- affiche le nombre de valeurs appartenant à l'intervalle [*Val1,Val2*].

Exercice 5 :
Réaliser un programme qui :
- demande à l'utilisateur la saisie d'une date (**le jour et le mois uniquement**) ;
- calcule le nombre de jours écoulés depuis le début de l'année jusqu'à la date saisie.

On supposera que l'année en cours est l'année 2015.
Le programme doit vérifier la validité de la date saisie.
Exemples de dates non valides au format JJ/MM : 32/01 ; 20/13 ; 29/02 ; 31/11 ; …
Pour cela il faut utiliser un tableau comprenant le nombre de jours par mois de chaque mois de l'année 2015.

Exercice 6 :
Réaliser un programme qui :
- demande à l'utilisateur de saisir 10 valeurs entières et les range dans le tableau ;
- affiche le contenu du tableau ;
- « arrange » le tableau afin de placer les valeurs négatives avant les valeurs positives ou nulles ;
- affiche le nouveau contenu du tableau.

5. Les tableaux à 2 dimensions

Exemple :

```
double [][] tab = {{1.1, 2.2, 3.3, 4.4}, {0.5, 1, 1.5, 2}};
```

L'instruction précédente déclare et crée un tableau à 2 dimensions contenant des éléments de type double.
On peut le voir comme une matrice constituée de 2 lignes et de 4 colonnes :

Représentation du tableau tab :

	0	1	2	3
0	1.1	2.2	3.3	4.4
1	0.5	1	1.5	2

← indice de la colonne : j

↑ indice de la ligne : i

indice de la ligne : i

L'expression tab[i][j], de type double, permet d'accéder à l'élément situé sur la ligne d'indice i dans la colonne d'indice j. Par exemple, tab[1][2] est ici initialisé à 1.5.

Affichage de tab en utilisant 2 boucles for "classiques" imbriquées :

```
int i, j;  //déclaration de 2 variables de boucle

for (i = 0; i <= 1; i++)   //pour chaque ligne
{
        // afficher les éléments de la ligne d'indice i
        for (j = 0; j <= 3; j++)  //pour chaque élément de la ligne
        {
                System.out.print(tab[i][j] + " ");
        }
        System.out.println("");          //saut de ligne

}
```

On peut également voir le tableau tab comme étant constitué de 2 éléments, chaque élément étant une ligne donc un tableau à 1 dimension contenant 4 doubles :

- l'expression tab[0] permet d'accéder au 1er élément de tab et correspond à un tableau à 1 dimension contenant 4 valeurs réelles : 1.1, 2.2, 3.3 et 4.4;
- l'expression tab[1] permet d'accéder au second et dernier élément de tab.

Illustration :

```
System.out.println("Affichage des éléments de tab[0] : ");
for (double d : tab[0])
{
        System.out.print(d + " ");
}
System.out.println("");          //saut de ligne

System.out.println("Affichage des éléments de tab : ");
for (double [] soustab : tab)     //pour chaque élément de tab
{
        // tab est vu comme étant constitué de 2 éléments
        // qui sont eux-même des tableaux à 1 dimension : tab[0] et tab[1]
        // la variable soustab prend successivment les valeurs : tab[0] puis tab[1]
        for (double d : soustab)
        {
                System.out.print(d + " ");
        }
        System.out.println("");          //saut de ligne

}
```

Q11 Tester les codes ci-dessus.

Exercice 7 :

Réaliser un programme qui :

- déclare 2 matrices 3x4 d'entiers (3 lignes et 4 colonnes) (les 2 matrices seront initialisées lors de leurs déclarations) ;
- calcule la somme des 2 matrices ;
- affiche le résultat.