PSEUDO-CODE ET LANGAGE JAVA : - LES TABLEAUX -

INTÉRÊT DES TABLEAUX PAR L'EXEMPLE

Comment conserver un ensemble de notes pour effectuer des calculs de moyenne, de maximum, de minimum pour une classe de 20 élèves. La seule solution algorithmique dont nous disposons à l'heure actuelle :

- Déclarer 20 variables représentant les 20 notes nommées : n1, n2, n3,…, n20
- Saisir les 20 notes dans 20 SAISIR distinctes : saisir n1, saisir n2,…, saisir n20
- Effectuer le calcul de la moyenne avec l'instruction : (n1 + n2 +… + n20)/20

Imaginons que nous soyons dans un programme avec quelques centaines ou milliers de valeurs à traiter ➔**ingérable**

Imaginons que le nombre d'élèves augmente ou diminue ➔**ingérable**

Conclusion : il est difficile de gérer ce type de problème en utilisant de simples variables.

⇨ Pour gérer ce problème, on utilisera un TABLEAU : permets de rassembler toutes ces variables en une seule.

STRUCTURE D'UN TABLEAU EN PSEUDO-CODE

Un tableau est une structure de données permettant un accès direct aux éléments au travers d'un indice (ou numéro de case) dont voici les caractéristiques :

⇨ Chaque case est numérotée (ou indicé) de 0 (la première case) à N-1 (la dernière).
⇨ N entier et >1
⇨ Toutes les cases contiennent une valeur d'un type donné.

Représentation d'un tableau A de N cases contenant des entiers.

	0	1	2	3	4	…	N-2	N-1
Tableau A :	15	25	0	22	-10	…	56	33

DÉCLARATION ET CRÉATION D'UN TABLEAU EN JAVA

Nous faisons ici la différence entre déclaration et création. Les tableaux sont caractérisés par une paire de crochets.

DÉCLARATION

```
int [] A;          // A est un tableau d'entier
float B[];           // B est un tableau de float
int [] C,D;        // C et D sont des tableaux d'entier
double E[],F[],i;    // E et F sont des tableaux  de double, i est un double
```

Dans ses 4 exemples, les tableaux sont déclarés, mais pas encore initialisés. D'ailleurs si l'on cherche à afficher le contenu du tableau en case 0, comme illustré par le code suivant, celui-ci provoque une erreur d'exécution :

```
System.out.println("" + A[0]);    //ERREUR : variable A might not have been initialized
```

Dans ces exemples, vous remarquez bien que la taille des tableaux n'est pas spécifiée et c'est normal !

La déclaration d'un tableau ne lui alloue pas de mémoire.

CRÉATION OU INITIALISATION

Après la déclaration vient l'étape de la création. C'est ici que l'on initialise le tableau en précisant le nombre de cases utilisées par ce dernier. Cette opération s'effectue à l'aide de l'opérateur **NEW**.

C'est l'opérateur new qui se charge de cette opération. Lorsque new est invoqué le tableau est créé.

Exemple :

```
A=new int[100];            //A est maintenant créé. A possède 100 cases.
B=new float[200];
```

DÉCLARATION ET CRÉATION

Dans les exemples précédents, nous avons séparé les 2 étapes. Le langage JAVA permet de faire cela sur une seule ligne.

```
int G[]=new int [50];      //G est déclaré puis créé dans la foulé. G contient 50 cases.
```

Notons qu'une fois créer le tableau est rempli avec la valeur 0 ou équivalent.

Dans l'exemple précédent, la taille du tableau (son nombre de cases) est définie en dure dans le programme (valeur statique). Il est possible de définir dynamiquement celle-ci à partir d'une saisie utilisateur ou bien un calcul dans le programme.

Exemple :

```
int nbCases;
Scanner sc=new Scanner(System.in);
nbCases = sc.nextInt();
int A[]= new int[nbCases];
```

Une autre façon de créer un tableau est de fixer les valeurs pour chaque élément dès sa déclaration dans une {liste de valeurs}. Dans ce cas-là, le tableau est déclaré et créé dans le même temps. On note aussi l'absence de l'opérateur new.

```
int A [] = {0,1,2,3,4,5,6,7,8,9};       //A est un tableau de 10 cases
double B [] = {0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};   //B est un tableau de 10 cases
char C [] = {'a','b','c','d','e','f','g'};    //C est un tableau de 7 cases.
String S [] = {"chaine1", "chaine2 ", "chaine3" , "chaine4"}; //S est un tableau de 4 cases.
```

On note dans tous les exemples que le tableau est en Majuscule. C'est une convention propre à ce cours qui a pour unique but de distinguer une variable classique commençant par une minuscule d'un tableau.

L'ACCÈS AUX ÉLÉMENTS DU TABLEAU

L'accès aux éléments du tableau se fait de façon directe en donnant le numéro de case entre crochets **[ ]**. Ce numéro est obligatoirement une valeur entière. On utilise souvent une variable appelée « indice » (index en anglais) et par convention on utilise très souvent les lettres i, j ,k , etc .. pour nommer ces indices.

L'accès se fait : soit en LECTURE, soit en ÉCRITURE.

ACCÈS EN LECTURE

Le tableau se trouve dans l'expression, à droite du signe d'affectation. Une opération de lecture consiste à récupérer la valeur d'une case.

Exemple :

```
int  B[]={10,20, 30,40, 50,10, 20,30, 40,50} ;      //B est un tableau prérempli de 10 cases
int i=5 ;              // i est appelé indice.
int val = B[3] + 5 ;   // on accède à la case d'indice 3 du tableau,
                       // le contenu de cette case + 5  est affecté à val
val = B[i] ;           // on accède à la i-ème case du tableau, le contenu de cette case est affecté à val
```

ACCÈS EN ÉCRITURE

On souhaite modifier le contenu d'une case. Exemple :

```
int N= 10 ;
int A[] = new int[N] ;
int i=5 ;      //i : indice.
int val = 8 ;
A[3] = A[2] + 10 ;     // on accède en écriture à la case 3 du tableau, le contenu de cette case est
                       // remplacé par la valeur 10 + A[2]
A [i] = val ;          // on accède à la ième case du tableau, son contenu est remplacé par val
```

ERREURS FRÉQUENTES

Dans de nombreux langages (en C, JAVA) un tableau est une suite de cases se suivant les unes à la suite des autres en mémoire.

Prenons le cas d'un tableau A d'entier (4 octets) de 5 cases . On a la représentation mémoire suivante :

		@1ère case	@2ème cases	@3ème cases	@4ème case	@5ème case			
		4 octets	4 octets	4 octets	4 octets	4 octets			
A[-2]	A[-1]	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

Notre tableau se trouve dans la mémoire. Cela signifie qu'il y a des données avant notre tableau et après. Une erreur fréquente consiste à accéder à un élément du tableau qui n'existe pas. Par exemple A[-1] ou A[5].

Ainsi il est possible d'accéder à des zones mémoires qui se trouvent avant le tableau ou après. Le problème c'est que cette zone mémoire est très certainement utilisée par une autre variable.

Dans ce cas, on vient modifier le contenu de cette variable. C'est ce qu'on appelle une opération à effet de bord ou plus connu sous le nom anglais de "stack overflow".

Le Java est particulier dans ce domaine. L'accès (en lecture ou écriture) à une case d'un tableau qui n'existe pas provoquera une erreur à l'exécution du programme. Attention ce mécanisme de protection n'existe pas forcément dans d'autres langages.

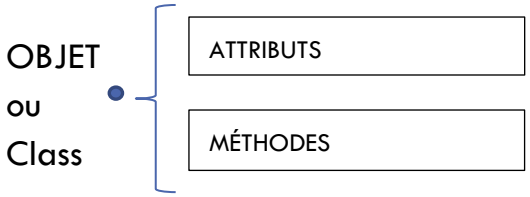
Retenons donc que Java ne vérifie pas les indices utilisés pour l'accès aux éléments du tableau. C'est donc au programmeur d'être vigilant.

```
int A [] = {0,1,2,3,4,5,6,7,8,9};
A[-5]=5;                          // erreur du type java.lang.ArrayIndexOutOfBoundsException
int a = A[50];                    // erreur du type java.lang.ArrayIndexOutOfBoundsException
System.out.println("" +  A[100]);  // ERREUR : la case 100 n'existe pas.
                                  // Message : Index 100 out of bounds for length 50
```

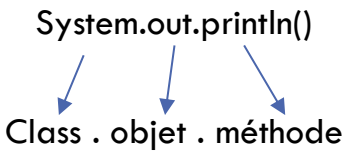
UN TABLEAU EST UN OBJET

En java, un tableau est un objet, même s'il contient des valeurs de type primitif. Par conséquent, il possède comme tout objet des attributs et méthodes. Il est notamment possible de récupérer la taille d'un tableau. Il suffit d'utiliser l'attribut **length**.

Un **Objet** rend des services au travers de ses méthodes et attributs. La figure ci-dessous montre comme accéder aux attributs et aux méthodes en utilisant l'opérateur (.) point appelé **accesseur de propriétés.**



Notons que nous utilisons cette notion d'objet depuis le début de l'année par exemple lorsque nous écrivons ceci :



Dans le cas d'un tableau, il y a un seul attribut accessible : length

Exemple :

```
int tab[] = new int[10] ;
System.out.println("Voici la taille du tableau : " + tab.length) ;  // affiche 10
```

LA CLASSE ARRAYS

Cette classe expose des méthodes utilitaires pour les tableaux. Cette classe ne peut-être instancié, toutes ses méthodes sont **static**. On parle de méthode de classe. D'une façon générale, il existe une version de chacune de ces méthodes pour tous les types primitifs Java.

Exemple :

```
int B[] = Arrays.copyOf(A, A.length);                //B est une copie de A
System.out.println("" + Arrays.toString(B));         //Conversion du tableau B en chaine de caractère
```

Cette classe, comme bien d'autres, rend des services au travers de ces méthodes. Il est impossible de les lister toutes ici. Cela fait partie des compétences d'un développeur : se plonger dans la documentation.