Apprendre à programmer en langage objet (Java)

Auprès d'Ingesup Paris

TP 5 : Les tableaux

1. Découverte et pratique

Il est parfois nécessaire de conserver en mémoire plusieurs données de même type, par exemple une valeur calculée à chaque itération d'une boucle. Cela nécessite de pouvoir écrire un programme dans lequel le nombre de variables disponibles est lui-même un paramètre – une suite finie (mais de longueur variable) de valeurs du même type. Une telle construction s'appelle un tableau.
En Java, le type "tableau d'éléments de type T" se note T[]. Les éléments d'un tableau t de taille n sont numérotes de 0 à n-1, et sont désignés par t[0], t[1], . . . , t[n-1].

Remarque. On a déjà rencontré un exemple de tableau dans l'en-tête de la fonction main d'un programme dont le paramètre est un tableau sur le type String (il contient la liste des paramètres donnés au lancement du programme).

```
public static void main(String[] args) { ... }
```

Le fragment de programme suivant déclare une variable tab de type "tableau d'entiers", puis il alloue un tableau de 5 entiers (via l'opérateur new) et l'affecte à la variable tab. Les cinq éléments du tableau tab se comporteront ensuite comme cinq variables de type int, désignées par tab[0], tab[1], . . ., tab[4].

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

On peut obtenir la *longueur* du tableau correspondant à une variable t (c'est-à-dire son nombre de cases) à l'aide de l'expression t.length. Les éléments d'un tableau t sont ainsi désignés par t[0], t[1], . . ., t[t.length - 1].

Remarque. Contrairement à une chaîne de caractères w de type String (autre type référence), pour la- quelle on récupère la longueur via l'expression w.length() (méthode sans paramètre), un tableau t voit sa longueur fixée au moment de l'allocation, longueur qui devient alors un *attribut* du tableau, auquel on accède via l'expression t.length (pas de parenthèses dans ce cas).

Exemple 1

Supposons que l'on veuille lire un entier n, puis lire n entiers et les afficher dans l'ordre inverse de celui dans lequel ils ont été lus. La séquence de code Java correspondante est :

```
int n;
Scanner sc = new Scanner(System.in);
System.out.print("Taille du tableau a renverser : ");
n = sc.nextInt();
int[] t = new int[n] ;
```

Exercice 2

Écrire un programme qui affiche les paramètres avec lesquels il a été lancé à raison d'un paramètre par ligne.

Dans l'instruction tab = new int[5]; l'opérateur new renvoie l'adresse mémoire de l'emplacement des éléments du tableau nouvellement créé ; cette adresse est alors affectée à la variable tab, ce qui permet ensuite d'accéder à chacun des éléments du tableau. Ainsi, la *valeur* d'une variable de type tableau est une indication de l'emplacement mémoire de ce tableau et est sans rapport avec le *contenu* du tableau

(c'est-à-dire les valeurs de ses cases).

Exemple 3. Après la séquence de code suivante :

```
int[] tab = new int[5];
tab[0] = 4; tab[1] = 12; tab[2] = -3; tab[3] = 0; tab[4] = 5;
System.out.println(tab);

int[] tac = tab;  tac[0] = 3;
System.out.println(tab[0]);

int[] tad = {3,12,-3,0,5};
System.out.println(tab==tac);
System.out.println(tab==tad);
```

Tout d'abord, l'instruction 3 provoque l'affichage de la valeur de tab (quelque chose comme [I@735cda3f) et en aucun cas l'affichage de son contenu.
Ensuite, l'instruction 5 fait que la variable tac désigne le *même* tableau que tab. En particulier, l'instruction 7 produit l'affichage 3. L'affectation tac = tab ne recopie donc pas le contenu du tableau. Pour recopier le contenu d'un tableau, il faut recopier les valeurs des cases une à une (en utilisant une boucle) par exemple :

```
public static int[] copieTableau(int[] t){
    int[] s = new int[t.length];
    for (int i=0; i<t.length; i++) {
        s[i] = t[i]; return s;
    }
}
```

Enfin, après l'instruction 9, la variable tad désigne un nouveau tableau ayant le même contenu que celui désigné par tab : l'évaluation du test **tab == tad** cependant donne false.

Exercice 4 : Écrire une ***fonction sontEgaux*** qui teste si deux tableaux d'entiers sont égaux.

2. Exercices complets

Exercice 5 :
On s'intéresse ici à des tableaux de nombres entiers à une seule dimension, possédant un nombre impair d'éléments et dont tous les éléments sont différents. On appelle médiane d'un tel tableau t l'élément m de t tel que t contienne autant d'éléments strictement inférieurs à m que d'éléments strictement supérieurs à m.

1. Écrire une ***fonction nbInf*** qui, étant donné un tableau d'entiers t et un entier v, renvoie le nombre d'éléments du tableau t strictement inférieurs à v.
2. Écrire une ***fonction mediane*** qui, étant donné un tableau t satisfaisant les conditions énoncées, renvoie la position de la médiane dans le tableau t.
3. Écrire une ***fonction verifTableau*** qui, étant donné un tableau t de nombres entiers, renvoie-la valeur booléenne ***true*** si le tableau t satisfait effectivement les conditions énoncées et la valeur false si ce n'est pas le cas.
4. Écrire une ***fonction tabInf*** qui, étant donné un tableau t, renvoie ***null*** si t ne satisfait pas les conditions énoncées et un tableau contenant tous les éléments de t inférieurs à sa médiane sinon.

Exercice 6 :
On se propose de construire une méthode de tri, appelé *tri par sélection*.

1. Écrire une ***fonction maximum*** qui prend en paramètres deux entiers a et b de type int et qui renvoie le maximum des deux.
2. Écrire une **fonction maximumTab** qui prend en paramètres un tableau t d'entiers de type int et deux indices i et j et qui, en supposant i≤j valides et t non vide, renvoie la plus grande valeur

contenue dans t à un indice compris entre i et j.

3. Écrire une ***fonction indice*** qui prend en paramètres un tableau t d'entiers de type int et une valeur c de type int et qui renvoie le plus petit indice d'une occurrence de c dans t si une telle occurrence existe et renvoie -1 sinon.
4. Écrire une ***fonction échanger*** qui prend en paramètres un tableau t d'entiers de type int et deux indices i et j et qui échange dans t les éléments d'indice i et j.
5. Des trois fonctions précédentes, déduire une ***fonction trier*** qui prend en paramètre un tableau t d'entiers de type int et le trie : sélection de l'élément maximum et déplacement en dernière position, sélection de l'élément maximum sur la partie du tableau restant à trier et déplacement en avant dernière position, etc.