Programmation en Java : boucles et tableaux

Bernard Hugueney

May 17, 2017

Contents

1 Objectifs pédagogiques

- savoir utiliser les tableaux en Java
  - déclaration
  - instanciation, intialisation
  - accès en lecture ou écriture
- être capable de concevoir un algorithme traitant une séquence d'éléments.

2 Rappels

2.1 Boucles `while`

L'instruction `while` permet de répéter un jeu d'instructions *tant que* la condition de boucle est vraie. Deux syntaxes sont possibles:

```
1 while(/*condition*/){
2     /*instruction_1;
3       instruction_2;
4       ...*/
```

```
5  }
6
7  do{
8      /*instruction_1;
9        instruction_2;
10       ...*/
11 } while(/*condition*/);
```

2.2 Tableaux

Un tableau est une variable structurée composée d'un nombre entier N de variables du **même type**. Le nombre N est la **taille** du tableau. On peut accéder à chacun de ses éléments par un indice (numéro de la case du tableau) allant de 0 à $N-1$

2.2.1 Déclaration et instancation (création)

La **déclaration** d'un tableau se fait de la manière suivante (où `TypeElt` est le type des éléments du tableau, par exemple `int` pour un tableau d'entiers):

```
1  TypeElt[] nomTableau;
```

C'est donc une déclaration "normale" d'une variable (ici appellée `nomTableau`) dont le type (du tableau, donc) est `TypeElt []`.

Un tableau déclarée doit ensuite être **instancié** (c'est-à-dire effectivement créé): `nomTableau = new TypeElt[taille];` Là encore, il s'agit d'une affectation "normale" (=) de la valeur à droite du signe = à la variable dont le nom est à gauche du signe =. Comme pour toutes les déclarations de variables, il est possible d'affecter directement une valeur lors de la déclaration. Ainsi `int [] nbJoursParMois = new int[12];` déclare et instancie une variable nommée `nbJoursParMois` contenant 12 entiers.

2.2.2 Initialisation et accès

L'**initialisation** consiste à attribuer à chacune des cases du tableau une valeur.

L'initialisation peut être effectuée lors de la déclaration d'un tableau, en indiquant la liste des valeurs respectives entre accolades. Par exemple, un tableau de taille 12 donnant le nombre de jours par mois se définit de la manière suivante :

```
1  int[] nbJoursParMois = {31,28,31,30,31,30,31,31,30,31,30,31};
```

L'initialisation peut également être réalisée en accédant à chacune des cases du tableau, et en lui attribuant une valeur. L'accès à un élément du tableau se fait en donnant son **indice**: `nomTableau[indice]`

Par exemple `joursParMois[1]=29;` stocke la valeur 29 dans la case d'indice 1. À noter qu'en Java, les cases d'un tableau de taille N sont numérotées de 0 à $N-1$, `nbJoursParMois[1]` correspond donc au nombre de jours du mois de février.

Pour récupérer la valeur d'une case d'un tableau, la syntaxe est similaire. Par exemple `int nbJoursFevrier = nbJoursParMois[1];` récupère la valeur de la case d'indice 1 du tableau `nbJoursParMois`, et la stocke dans la variable `nbJourFevrier`.

2.2.3 Taille

Pour récupérer la taille d'un tableau vous pouvez utilisez l'*attribut* `length` qui lui est associé. L'exemple suivant récupère la taille du tableau `nbJoursParMois` et la stocke dans une variable nommée nbMois.

```java
int nbMois = joursParMois.length;
```

Cet attribut vous servira notamment dans des boucles `for` pour parcourir l'ensemble des éléments d'un tableau. Remarquez (cf. conditions d'arrêt ou non pour les boucles) qu'un indice dans un tableau `nomTableau` doit rester strictement inférieur (opérateur `<`) à `nomTableau.length`.

3 Exercices

3.1 Boucle `while`

Quelle différence faites-vous entre les deux syntaxes de la boucle `while`? Comparer les résultats des deux fragments de code suivants:

```java
int i = 1;
while(i < 1){
   System.out.print("Hello world!");
}
```

et

```java
int i = 1;
do{
    System.out.print("Hello world!");
} while(i < 1);
```

3.2 Filtre de saisie

Écrire le **pseudo-code** d'un programme qui demande à l'utilisateur d'entrer une note entre 0 et 20, tant que l'utilisateur n'a pas entré une note valide.

Écrire le code Java associé.

3.3 Remplissage de tableau

Écrire le **pseudo-code** d'un programme permettant de remplir un tableau de 30 cases avec les puissances successives de 2, dans l'ordre décroissant. Écrire le code Java associé, complété par l'affichage du tableau.

Quelle est la complexité (combien d'instructions sont exécutées par l'ordinateur) de votre solution (en fonction de la taille du tableau) ?

Bonus peut-on faire en sorte d'avoir une complexité linéaire suivant le nombre d'éléments du tableau?

3.4 Maximum d'un tableau

Écrire le pseudo-code d'un programme qui recherche le maximum des éléments d'un tableau et afficher le premier indice du tableau pour lequel l'élément est égal à ce maximum.

Modifier votre pseudo-code pour afficher tous les indices correspondant aux éléments maximaux.

Écrire le code java associé.

3.5 Histogrammes

Écrire le pseudo-code d'un programme qui fait l'histogramme des notes (supposées être des entiers compris entre 0 et 100) que l'utilisateur entre.

Écrire le code java associé, le programme devra :

- demander les notes par un filtre de saisie simple,
- calculer la moyenne de ces notes, leur maximum et leur minimum,
- afficher les notes par catégories (entre 0 et 9, 10 et 19, ...).

3.6 Bonus : Dichotomie

Écrire le pseudo-code d'une recherche dichotomique dans un tableau que l'on suppose trié.

On rappelle le principe de la recherche dichotomique dans un tableau trié par ordre croissant. En supposant que l'on dispose d'un tableau T dans lequel on cherche l'indice où est stocké l'entier n.

- On tient en permanence deux indices *gauche* et *droite* tels que $T[gauche] \leq n \leq T[droite]$.
- À chaque étape, on prend le milieu de l'intervalle $[gauche, droite]$ et on le compare à n.
- En fonction du résultat, soit on renvoie l'indice du milieu, soit on met à jour *gauche* ou *droite* et on continue à chercher n dans le sous-tableau correspondant.
- On s'arrête lorsque *gauche* est plus grand que *droite* : ce cas est un cas d'échec (n n'a été trouvé nulle part) et on affiche l'indice -1.