Structures de base

Table des matières

I. Epargne et boucles

Vous avez déposé 10000F à la Caisse d'Epargne et vous souhaitez savoir comment cette somme va évoluer. Chaque année elle rapporte 4,5% d'intérêts, ce qui signifie qu'elle est multipliée par 1,045.

1. Boucle FOR

Essayons de connaitre le montant de votre capital après 5 ans. Il faut multiplier le capital initial par 1,045 cinq fois de suite. Nous pouvons utiliser une boucle FOR. La structure d'une telle boucle est :

FOR (initialisation; condition; transition) { instructions }

Pour notre problème, cela se traduit par :

```
double capital=10000;
for (int i=1; i<=5; i++) capital=capital*1.045;
```

L'initialisation consiste à mettre le compteur i à 1. La condition pour laquelle on exécute les instructions est i<=5. La méthode de transition d'un passage à l'autre dans la boucle consiste à ajouter une unité au compteur i, ce qui se traduit par i++.

2. Un programme avec FOR

Ecrivons un programme qui calcule la valeur atteinte par votre capital après un nombre n d'années qui sera passé en paramètre.

```
public class epargne1 {

 public static void main(String args[]) {
  double capital=10000;
  double taux=1.045;
  //traduire l'argument en entier
  int n=Integer.parseInt(args[0]);
  //boucle pour répéter n fois le calcul
  for (int i=1; i<=n; i++)
   capital=capital*taux;
  System.out.print("Après "+n+" années, le capital vaut ");
  System.out.println(capital+" F.");
 }

}
```

3. Boucle WHILE

Modifions légèrement notre problème. Nous avons déposé 10000F à la Caisse d'Epargne et nous souhaitons savoir combien d'années il faudra attendre pour que le capital obtenu dépasse 15000F. Nous allons utiliser une boucle WHILE pour indiquer que tant que le capital ne dépasse pas 15000F il faut attendre une année de plus. La structure d'une telle boucle est :

WHILE (condition) { instructions }

Pour notre problème cela se traduit par :

```
int n=0; //nombre d'années
double capital=10000;
while (capital<15000) {
 capital=capital*1.045;
 n++;
}
```

La condition à vérifier pour que la boucle s'exécute est simplement capital<15000 (on n'a pas encore atteint les 15000F attendus). A chaque passage dans la boucle on calcule la nouvelle valeur du capital et on augmente d'une unité le nombre d'années.

Il reste à insérer cette boucle dans un programme pour pouvoir l'exécuter.

II. Tableaux

1. Création d'un tableau

Un tableau permet de regrouper des variables de **même** type dans une structure unique. On le déclare simplement en indiquant le type commun aux variables et en faisant suivre le nom par des crochets. Par exemple, pour un tableau nommé tab devant contenir des nombres entiers :

```
int tab[];
```

Avant de pouvoir utiliser le tableau, il faut lui réserver un espace mémoire en indiquant le nombre d'éléments qu'il va contenir. Pour que le tableau tab puisse contenir 5 nombres, on écrira l'instruction :

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

C'est le mot-clé **new** qui permet d'effectuer la réservation de mémoire.

Il est possible de regrouper déclaration et réservation mémoire en une seule instruction qui sera :

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

2. Accès aux éléments d'un tableau

Pour écrire ou lire la valeur d'un élément du tableau tab on écrit simplement **tab[i]** où i est l'indice de l'élément considéré.
Attention : la numérotation des éléments du tableau commence à 0; les éléments du tableau tab de 5 nombres sont donc tab[0], tab[1], tab[2], tab[3] et tab[4].
Le nombre d'éléments d'un tableau est donné par sa propriété **length**. Les indices utilisables pour le tableau tab vont donc de 0 à tab.length.

Exemple : remplissons le tableau tab avec les nombres 1,2,3,4 et 5. On pourra utiliser une boucle FOR.

```
int tab[]=new int[5];
for (int i=0; i< tab.length; i++) tab[i]=i+1;
```

Ecrire un programme qui crée ce tableau et affiche ensuite ses éléments.

3. Autre méthode d'initialisation

Il est possible d'initialiser un tableau en donnant immédiatement la liste de ses éléments placés entre accolades et séparés par des virgules. Par exemple, le tableau contenant les jours de la semaine sera déclaré de la façon suivante :

```
public static String jour[]=
 {"lundi","mardi","mercredi","jeudi","vendredi",
  "samedi","dimanche"};
```

Ecrire un programme qui utilise ce tableau pour afficher les jours de la semaine.

III. Au cas par cas

1. L'instruction switch

L'instruction **switch** permet d'exécuter des instructions différentes selon la valeur que prend une variable. Elle a la forme générale suivante :

```
switch (var) {
 case val1 : instructions;
 case val2 : instructions;
 default : instructions;
}
```

La variable testée doit être de type entier ou de type caractère. Les valeurs val1, val2, ... doivent être des constantes. La clause **default** permet de donner des instructions à exécuter si la variable a une valeur différente de celles qui sont prévues. **Attention :** dès qu'un cas correspond à la valeur de la variable, toutes les instructions qui suivent sont exécutées; on utilise le mot-clé **BREAK** pour sortir de la boucle.

2. Application

Ecrivons un programme qui donne une appréciation pour une note sur 10 obtenue à un devoir : "Très insuffisant" de 0 à 2, "Insuffisant" de 3 à 4, "Moyen" de 5 à 6, "Bien" de 7 à 8 et "Très bien" de 9 à 10.

Si la note est contenue dans la variable note, on peut utiliser le code suivant :

```
switch (note) {
 case 0: case 1: case 2:
  System.out.println("Très insuffisant.");
  break;
 case 3: case 4:
  System.out.println("Insuffisant.");
  break;
 case 5: case 6:
  System.out.println("Moyen.");
  break;
 case 7: case 8:
  System.out.println("Bien.");
  break;
 case 9: case 10:
  System.out.println("Très bien.");
  break;
 default :
  System.out.println("Note impossible.");
}
```

Le mot-clé **break** permet d'éviter l'exécution des instructions correspondant aux cas suivant le cas traité.

Ecrire le programme complet qui lit la note passée en paramètre et affiche l'appréciation.

Exercices

1. **Compter de 5 en 5**
   Ecrire un programme qui compte de 5 en 5 de 0 jusqu'à 100 puis affiche la somme des nombres trouvés. On pourra écrire deux versions : l'une utilisant une boucle FOR et l'autre utilisant une boucle WHILE.

2. **Minimum, maximum et somme**
   Ecrire un programme qui place les nombres passés en paramètres dans un tableau, puis calcule et affiche le minimum, le maximum et la somme de ces nombres.

3. **Mois**
   Ecrire un programme qui lit le nombre entier entre 1 et 12 passé en paramètre et qui affiche le nom du mois correspondant.
   On pourra utiliser un tableau ou l'instruction switch.

4. **Calculette**
   Ecrire un programme qui lit trois paramètres : un nombre entier, un caractère représentant une opération (+,-,x ou /) et un second nombre entier, puis qui effectue le calcul ainsi indiqué et affiche le résultat. (attention à éviter les divisions par 0)
   Note : on ne peut pas utiliser le signe * comme paramètre, il a une signification particulière.