Boucles et Conditions : projet *NamesLoop*

Introduction, les conditions *if…else*

Les conditions utilisent des données booléennes, *true* ou *false*. Elles vérifient si une condition est remplie. Si oui, elles effectuent un bout de code prédéfinit, sinon, elles exécutent au autre bout de code prédéfinit ou passent directement au reste du programme. Elles ont la forme *if … else …*

```
if (BeaucoupDeFric == true)
    Restaurant = "Jardin des sens";
else
    Restaurant = "MacDo" ;
```

Si le personne est riche elle ira manger dans un certain restaurant, sinon, elle ira ailleurs…
Note : on peut en faire à plusieurs possibilités : *if… else if … else if… else if… else…*

```
if (Fric == 100)
    Resto = McDo;
else if (Fric == 200)
    Resto = Quick;
else if (Fric == 300)
    Resto = Flunch;
else
    Resto = chezMoi
```

Switch…Case

Les conditions *switch…case* remplacent les conditions *if…else if…else* quand le nombre de *else if* est trop important. Elles sont en effet plus rapides à écrire :

```
switch (note) {
    case 'A' :
        System.out.println ("Très Bien") break ;
    case 'B' :
        System.out.println ("Bien") break ;
    case 'Z' :
        System.out.println ("Lobotomisé !") break;
    default : System.out.println ("Tricheur !");
}
```

Note: pour sortir de la condition, on utilise le mot *break*, regardez bien l'utilisation du *;* pour terminer les instruction (et pas les lignes !)

Boucles *for*

Les boucles *for* testent une condition et exécutent le bout de code attaché à la boucle *tant que la condition n'est pas remplie*. Leur syntaxe officielle est la suivante :

```
for (initialization ; test ; increment) { statement ; }
```

Ceci ne vous évoque certainement rien, alors examinez le code suivant:

```
String [ ] salutation = new String[10] ;
int i;
for (i = 0 ; i < salutation.length ; i++)
    salutation[I] = “M.”;
```

Le programme crée une matrice du nom de *salutation* qui peut contenir 10 éléments.
Il crée ensuite une variable i de type Integer.
La boucle *for* commence : la valeur de i est 0, tant que la valeur de i est inférieure à la taille de la matrice (taille obtenue par la commande *salutation.length*), le code de la dernière ligne est exécuté. i est incrémenté de 1, puis la condition est re-vérifiée, ainsi de suite jusqu’à i = 10.

Boucles *while*

Les boucles *while* sont très semblables aux boucles for. Elles fonctionnent ainsi :

```
while (i < 10) {
    // corps de la boucle
i++ ;
}
```

Tant que i<10, le corps de la boucle est exécuté, puis i est incrémenté et la condition est re-testée.

Boucles *do…while*

Les boucles *do…while* sont des variantes des boucles *while*. Leur particularité réside dans le fait que la condition est testée *après* la première exécution de la boucle. Le code à exécuté tant que la condition n’est pas satisfaite est exécuté *au moins* une fois.

```
do {
    // corps de la boucle
} while (i < 10)
```

Voici un exemple plus concret sur l’utilisation des boucles :

```
class BoucleDo {
    public static void main(String args [ ]) {
        int x = 1;
        do {
        System.out.println(“La boucle est bouclée; passage” +x);
        x++ ;
        } while (x <= 100) ;
    }
}
```

Vous pouvez compiler ce programme, il fonctionne… ! Nommez le fichier source « BoucleDo.java ».

Break et *Continue*

Break et *Continue* sont deux mots clés qui servent à interrompre une boucle durant son exécution (break) et à reprendre l’exécution de cette boucle ensuite (continue).

Application : Projet *NamesLoop*

Ce programme, une fois de plus de ligne de commande crée une matrice contenant des prénoms de programmeurs célèbres et les associe à leur nom de famille. On utilise donc les matrices, les boucles *for* la création d'une méthode, l'affichage de texte. Ce programme est extrait de l'excellent ouvrage *Java2* (Roger Cadenhead & Laura Lemay) Ed. CampusPress.

```
class NamesLoop { // Création de la classe

     String[] firstNames = { "Dennis", "Grace", "Bjarne", "James" }; // matrice des
                                                                     //prénoms
     String[] lastNames = new String[firstNames.length]; //matrice des noms (vide)

     void printNames() { // méthode pour afficher
         for (int i = 0; i < firstNames.length; i++) // boucle for
             System.out.println(firstNames[i] + " " + lastNames[i]); //instruction
         }

     public static void main (String arguments[]) {
             // remplissage de la seconde matrice avec les prénoms
             // appel de la méthode d'affichage
         NamesLoop a = new NamesLoop();
         a.printNames(); // méthode d'affichage
         System.out.println("-----");
         a.lastNames[0] = "Ritchie";
         a.lastNames[1] = "Hopper";
         a.lastNames[2] = "Stroustrup";
         a.lastNames[3] = "Gosling";

         a.printNames(); // méthode d'affichage
     }
 }
```

Note : Ici la boucle *for* sert à déterminer la taille de la matrice des noms de famille. Elle compte le nombre d'éléments dans la matrice des prénoms, ce nombre d'éléments *i* est associé à la matrice des noms de famille (car chaque personne a un prénom et un nom de famille)