Corrigé des exercices sur *comment programmer*

Exercice 4.1.1 *Montant d'une facture*

Le but de l'exercice est d'écrire un programme qui calcule le montant hors taxes et le montant TTC d'une facture comportant plusieurs produits. Pour chaque produit, l'utilisateur rentrera le prix, la quantité et le taux de TVA parmi les trois taux : normal (20%), intermédiaire (10%) et réduit (5,5%).

Pour écrire ce programme, vous suivrez les 7 étapes décrites dans le cours, en détaillant par écrit le résultat de chacune des étapes.

Une des questions qui se posent à la lecture de l'énoncé est comment savoir le nombre d'articles de la facture. Il y a deux solutions : demander en début de programme le nombre d'articles ; demander après chaque article s'il y a d'autres à ajouter. Les deux options sont correctes. Pour ce corrigé, nous allons choisir la première.

4.0.1 Étape 1 : entrées et sorties

Entrées : un nombre entier n d'articles, puis n séries de trois informations : un prix, une quantité et un taux de TVA. Le prix sera un double, la quantité un nombre entier et le taux de TVA un caractère (n pour normal, i pour intermédiaire et r pour réduit).

Sorties : le prix total hors taxe et le prix total TTC.

4.0.2 Étape 2 : méthode de calcul

Le prix total hors taxe est la somme des produits prix par quantité de chaque article. Le prix total TTC est la somme des produits prix par quantité et par un plus le taux de TVA de chaque article.

4.0.3 Étape 3 : cas particuliers

Que faire si le nombre d'articles entré est négatif ou nul ?

Dans ce cas, nous choisissons d'afficher des totaux nuls.

Que faire si le taux de TVA entré est un caractère différent de n, i et r ?

Dans ce cas, nous choisissons d'appliquer le taux normal de TVA.

4.0.4 Étape 4 : quelles instructions

Pour choisir le taux de TVA, un if à trois branches pourra permettre de choisir le bon taux à appliquer pour le total TTC. Pour la saisie de n séries de trois informations, une boucle for s'impose,

car on connaît ce nombre n avant d'entrer dans la boucle. On sait dès le départ combien de tours de boucle doivent être effectués.

4.0.5 Étape 5 : agencement des instructions

Le choix du taux de TVA doit être fait pour chaque article ; par ailleurs chaque article est traité au cours d'un tour de boucle. Il faut donc nécessairement que le if soit à l'intérieur de la boucle.

4.0.6 Étape 6 : formalisation de l'algorithme

```
Entrées:
  nbarticles un entier: nombre d'articles
␣␣nbarticles␣fois:
␣␣␣␣␣prix␣(reel)
␣␣␣␣␣quantite␣(entier)
␣␣␣␣␣tauxTVA␣(caractère,␣i,␣r␣ou␣n)
Sorties:
␣␣totalHT␣(réel)
␣␣totalTTC␣(réel)
Début
␣␣totalHT␣<-␣0
␣␣totalTTC␣<-␣0
␣␣écrire␣"Nombre␣d'articles: "
␣␣lire␣nbarticles
␣␣pour␣n␣allant␣de␣1␣à␣nbarticles
␣␣␣␣␣écrire␣"Prix? ``
    lire prix
    écrire "Quantité?␣"
    lire quantite
    écrire "Taux␣de␣TVA␣(n,i,r)?␣"
    lire tauxTVA
    totalHT = totalHT + (prix *quantité)
    si tauxTVA == 'i' alors
      totalTTC = totalTTC + (prix *quantité *1.1)
    sinon si tauxTVA == 'r' alors
      totalTTC = totalTTC + (prix *quantité *1.055)
    sinon
      totalTTC = totalTTC + (prix *quantité *1.2)
  fin pour
  écrire "Total␣hors␣taxe:␣"
  écrire totalHT
  écrire "Total␣TTC:␣"
  écrire totalTTC
fin
```

4.0.7 Étape 7 : écriture du code

```
public class TauxTVA{
   public static void main(String[] args){
      int quantite, nbarticles;
      double prix, totalHT, totalTTC;
      char tauxTVA;
      totalHT = 0.0;
      totalTTC = 0.0;
      System.out.print("Combien␣d'articles?␣");
      nbarticles = Terminal.lireInt();
      for (int n=1; n<=nbarticles; n=n+1){
         System.out.print("Prix:␣");
         prix = Terminal.lireDouble();
         System.out.print("Quantité:␣");
         quantite = Terminal.lireInt();
         System.out.print("Taux␣de␣TVA␣(n,␣i,␣r):␣");
         tauxTVA = Terminal.lireChar();
         totalHT = totalHT + (prix *quantite);
         if (tauxTVA == 'i'){
            totalTTC = totalTTC + (prix *quantite *1.1);
         }else if (tauxTVA == 'r'){
            totalTTC = totalTTC + (prix *quantite *1.055);
         }else{
            totalTTC = totalTTC + (prix *quantite *1.2);
         }
      }
      System.out.print("Total␣HT:␣");
      System.out.println(totalHT);
      System.out.print("Total␣TTC:");
      System.out.println(totalTTC);
   }
}
```

Exercice 4.1.2 *Sapin trop laid*

Le but de l'exercice est d'écrire un programme qui affiche un soi-disant sapin constitué d'une suite d'étoiles. Le programme lira au clavier le nombre de triangles constituant le sapin.

```
> java SapinLaid
Combien de triangles? 3

      *
     ***
    *****
   *******
```

```
    *
   ***
  *****
 *******
    *
   ***
  *****
 *******
   ***
   ***
   ***
   ***
```

4.0.8 Étape 1 : entrées et sorties

Entrée : le nombre de triangles constituant le feuillage du sapin. Sortie : un dessin de sapin fait d'étoiles (cf. ci-dessus).

4.0.9 Étape 2 : méthode de calcul

Dessiner n triangles de 4 lignes, puis dessiner le tronc de 4 lignes aussi. Pour chaque triangle : initialiser le nombre d'espace à 5 et le nombre d'étoiles à 1. Faire 4 fois le dessin d'une ligne en dessinant le nombre d'espaces puis le nombre d'étoiles prévus, puis enlever 1 au nombre d'espaces et ajouter 2 au nombre d'étoiles. Pour le tronc : le nombre d'espace est de 4 et le nombre d'étoiles de 3. Ces nombres ne changent pas d'une ligne à l'autre.

4.0.10 Étape 3 : cas particuliers

Si le nombre de triangles est négatif ou nul : n'afficher que le tronc. La tempête a emporté toutes les feuilles !

4.0.11 Étape 4 : quelles instructions

Pour afficher les espaces en début de ligne : une boucle for, car on connaît le nombre de tours à faire. Pour afficher les étoiles : idem. Pour afficher les 4 lignes d'un triangle : une boucle for toujours. Pour les autres lignes du tronc : une boucle for. Pour afficher les n triangles : une boucle for !

4.0.12 Étape 5 : agencement des instructions

La boucle qui écrit une à une les lignes d'un triangle est nécessairement à l'intérieur de la boucle qui dessine les n triangles. De même les deux boucles qui écrivent un à un les espaces et une à une les étoiles contribuent à dessiner une ligne d'un triangle : elles sont donc toutes deux à l'intérieur de la boucle qui dessine les 4 lignes d'un triangle. Comme tous les espaces sont dessinés avant les étoiles, ces deux boucles sont l'une après l'autre.

Le dessin du tronc doit être fait en-dessous de tous les triangles : la boucle qui dessine ses 4 lignes est donc après la boucle des n triangles.

4.0.13 Étape 6 : formalisation de l'algorithme

```
Entrées:
  nbtriangles (entier)
Sorties:
   un dessin de sapin à l'écran
Variables␣locales:
␣␣nbespaces␣(entier):␣nombre␣d'espaces sur la prochaine ligne
  nbetoiles (entier): nombre d'étoiles␣sur␣la␣prochaine␣ligne
␣␣n␣(entier):␣compteur␣de␣boucle␣pour␣les␣triangles
␣␣lig␣(entier):␣compteur␣de␣boucle␣pour␣les␣lignes
␣␣i␣(entier):␣compteur␣de␣boucle␣pour␣les␣caractères
début
␣␣écrire␣"Combien␣de␣triangles"
␣␣lire␣nbtriangles
␣␣pour␣n␣de␣1␣à␣nbtriangles
␣␣␣␣nbespaces␣<-␣5
␣␣␣␣nbetoiles␣<-␣1
␣␣␣␣pour␣lig␣de␣1␣à␣4
␣␣␣␣␣␣pour␣i␣de␣1␣à␣nbespaces
␣␣␣␣␣␣␣␣ecrire␣' '
␣␣␣␣␣␣fin␣pour
␣␣␣␣␣␣pour␣i␣de␣1␣à␣nbetoiles
␣␣␣␣␣␣␣␣ecrire␣'*'
␣␣␣␣␣␣fin␣pour
␣␣␣␣␣␣passer␣à␣la␣ligne
␣␣␣␣␣␣nbespaces␣<-␣nbespaces␣-␣1
␣␣␣␣␣␣nbetoiles␣<-␣nbetoiles␣+␣2
␣␣␣␣fin␣pour
␣␣fin␣pour
␣␣nbespaces␣<-␣4
␣␣nbetoiles␣<-␣3
␣␣pour␣lig␣de␣1␣à␣4
␣␣␣␣pour␣i␣de␣1␣à␣nbespaces
␣␣␣␣␣␣ecrire␣' '
␣␣␣␣fin␣pour
␣␣␣␣pour␣i␣de␣1␣à␣nbetoiles
␣␣␣␣␣␣ecrire␣'*'
␣␣␣␣fin␣pour
␣␣␣␣passer␣à␣la␣ligne
␣␣fin␣pour
fin
```

4.0.14 Étape 7 : écriture du code

```
public class SapinLaid{
```

```
   public static void main(String[] args){
      int nbtriangles, nbespaces, nbetoiles;
      System.out.print("Combien␣de␣triangles?␣");
      nbtriangles = Terminal.lireInt();
      System.out.println();
      for (int n =1; n<= nbtriangles; n=n+1){
         nbespaces=5;
         nbetoiles = 1;
         for (int lig=0; lig<4; lig=lig+1){
            for (int i = 0; i < nbespaces; i=i+1){
               System.out.print('␣');
            }
            for (int i=0; i<nbetoiles; i=i+1){
               System.out.print('*');
            }
            System.out.println();
            nbetoiles = nbetoiles+2;
            nbespaces = nbespaces -1;
         }
      }
      nbespaces=4;
      nbetoiles = 3;
      for (int lig=0; lig<4; lig=lig+1){
         for (int i = 0; i < nbespaces; i=i+1){
            System.out.print('␣');
         }
         for (int i=0; i<nbetoiles; i=i+1){
            System.out.print('*');
         }
         System.out.println();
      }
   }
}
```

Exercice 4.1.3 *Sapin trop beau*

Même question que l'exercice précédent.

```
> java SapinBeau
Combien de triangles? 3

        *
       ***
      *****
        *
       ***
```

```
  *****
 *******
    *
   ***
  *****
 *******
*********
   ***
   ***
   ***
```

Quelle sont les différence entre ce sapin et celui dessiné à l'exercice précédent ? C'est que chaque triangle successif comporte une ligne de plus que le précédent. Le plus petit triangle ne compte que trois lignes alors que les triangles tous pareils du sapin laid ont 4 lignes.

La difficulté du dessin provient de ce que le dessin doit se *caler* sur le plus grand triangle, celui du bas. Les triangles situés au-dessus doivent s'adapter pour être centrés sur le milieu du plus grand triangle. De même, le tronc sera plus ou moins décalé selon la taille du triangle.

Comparons deux extraits (haut et bas) des dessins avec respectivement 2 triangles et 4 triangles.

```
---- haut du sapin à 2 triangles
   *
  ***
 *****
---- haut du sapin à 4 triangles
     *
    ***
   *****
---- tronc du sapin à 2 triangles
  ***
  ***
  ***
---- tronc du sapin à 4 triangles
    ***
    ***
    ***
```

Dans les deux cas (et ce serait également le cas des triangles intermédiaires), il y a deux espaces en plus pour décaler le plus grand des deux sapins. Et ces deux espaces correspondent à la différence des nombres de triangles (4-2).

Les étapes 1 à 4 du processus de développement de programme sont à peu près identiques au cas du sapin laid. Nous ne les répétons donc pas.

4.0.15 Étape 5 : agencement des instructions

L'imbrication des instructions est la même que pour le sapin laid. Mais il faudra adapter plusieurs éléments. Notamment, le nombre de lignes de chaque sapin doit être incrémenté d'une unité à chaque itération de la boucle des triangles. Cela tombe bien : le compteur de boucle est lui aussi incrémenté

d'une unité à chaque tour de boucle. On pourra donc faire intervenir ce compteur dans le nombre de lignes pour qu'au premier tour, le triangle ait trois lignes, au second tour de boucle, il en ait 4, etc.

Le nombre d'espaces à dessiner à gauche du tronc doit dépendre pour sa part du nombre de triangles.

En résumé, par rapport au sapin laid, la structure du programme est la même. Ce sont juste les données de calcul des boucles (nombre de tours, notamment) qui varient et induisent des dépendances entre le compteur d'une boucle englobante et les boucles englobées.

4.0.16 Étape 6 : formalisation de l'algorithme

```
Entrées:
  nbtriangles (entier)
Sorties:
  un dessin de sapin à l'écran
Variables␣locales:
␣␣nbespaces␣(entier):␣nombre␣d'espaces sur la prochaine ligne
  nbetoiles (entier): nombre d'étoiles␣sur␣la␣prochaine␣ligne
␣␣n␣(entier):␣compteur␣de␣boucle␣pour␣les␣triangles
␣␣lig␣(entier):␣compteur␣de␣boucle␣pour␣les␣lignes
␣␣i␣(entier):␣compteur␣de␣boucle␣pour␣les␣caractères
début
␣␣écrire␣"Combien␣de␣triangles"
␣␣lire␣nbtriangles
␣␣pour␣n␣de␣1␣à␣nbtriangles
␣␣␣␣nbespaces␣<-␣5␣+␣nbtriangles
␣␣␣␣nbetoiles␣<-␣1
␣␣␣␣pour␣lig␣de␣1␣à␣4
␣␣␣␣␣␣pour␣i␣de␣1␣à␣nbespaces
␣␣␣␣␣␣␣␣ecrire␣' '
␣␣␣␣␣␣fin␣pour
␣␣␣␣␣␣pour␣i␣de␣1␣à␣nbetoiles
␣␣␣␣␣␣␣␣ecrire␣'*'
␣␣␣␣␣␣fin␣pour
␣␣␣␣␣␣passer␣à␣la␣ligne
␣␣␣␣␣␣nbespaces␣<-␣nbespaces␣-␣1
␣␣␣␣␣␣nbetoiles␣<-␣nbetoiles␣+␣2
␣␣␣␣fin␣pour
␣␣fin␣pour
␣␣nbespaces␣<-␣4␣+␣nbtriangles
␣␣nbetoiles␣<-␣3
␣␣pour␣lig␣de␣1␣à␣4
␣␣␣␣pour␣i␣de␣1␣à␣nbespaces
␣␣␣␣␣␣ecrire␣' '
␣␣␣␣fin␣pour
␣␣␣␣pour␣i␣de␣1␣à␣nbetoiles
␣␣␣␣␣␣ecrire␣'*'
␣␣␣␣fin␣pour
```

```
␣␣␣␣passer␣à␣la␣ligne
␣␣fin␣pour
fin
```

4.0.17 Étape 7 : écriture du code

```
public class SapinBeau{
   public static void main(String[] args){
      int nbtriangles, nbespaces, nbetoiles;
      System.out.print("Combien␣de␣triangles?␣");
      nbtriangles = Terminal.lireInt();
      System.out.println();
      for (int n =1; n<= nbtriangles; n=n+1){
         nbespaces=5+nbtriangles;
         nbetoiles = 1;
         for (int lig=0; lig<2+n; lig=lig+1){
            for (int i = 0; i < nbespaces; i=i+1){
               System.out.print('␣');
            }
            for (int i=0; i<nbetoiles; i=i+1){
               System.out.print('*');
            }
            System.out.println();
            nbetoiles = nbetoiles+2;
            nbespaces = nbespaces -1;
         }
      }
      nbespaces=4+nbtriangles;
      nbetoiles = 3;
      for (int lig=0; lig<4; lig=lig+1){
         for (int i = 0; i < nbespaces; i=i+1){
            System.out.print('␣');
         }
         for (int i=0; i<nbetoiles; i=i+1){
            System.out.print('*');
         }
         System.out.println();
      }
   }
}
```