Algorithmique et Programmation

Tp no. 9 : Exercices sur sous-programmes et tableaux à deux dimensions

V. Aponte

27 novembre 2017

Exercice 1 : réécriture exercice « ventes »

Dans cet exercice nous reprenons une correction partielle de l'exercice sur les ventes dans june concession de voitures. Pour mémoire, il s'agit de modéliser le nombre ventes dans une matrice, où chaque ligne correspond à un vendeur, et chaque colonne à un modèle. Les vendeurs et les modèles sont identifiés par leur nom. Dans cet exercice nous allons nous pencher sur la réécriture de cet exercice avec des sous-programmes.

Question 1

Le fichier `ExoVentesQ1.java` est une réponse à question 1 de cet exercice. On utilise des tableaux de chaînes pour représenter les noms de vendeur, et les noms de modèles. Comment sont ils employés au cours de la lecture des ventes ? Testez ce programme.

Question 2

Le fichier `ExoVentesQ1_M.java` contient deux méthodes permettant de réaliser une partie des tâches du programme de la question précédente.

```
public class ExoVentesQ1_M {
    /**
     * Affiche les ventes par vendeur.
     * @param v tableau de ventes
     * @param vend tableau de vendeurs
     * @param m tableau de modeles
     */
    public static void afficherVentes(int[][]v, String[]vend, String[] m){
        Terminal.ecrireStringln("Tableau de ventes:");
        Terminal.ecrireStringln("******************");
        for (int i = 0; i < v.length; i++) {
            System.out.print("  " + vend[i] + " : ");
            for (int j = 0; j < v[i].length; j++) {
                System.out.print(m[j] + " : ");
                Terminal.ecrireString(v[i][j] + ", ");
            }
```

```
            Terminal.sautDeLigne();
        }
    }
    /**
     * Lit au clavier les ventes des vendeurs dans @param nomv pour
     * les modeles dans @param nomm et retourne une matrice avec les
     * ventes lues.
     * @param nomv tableau noms vendeurs
     * @param nomm tableau noms modèles
     * @return
     */
    public static int[][] lireVentes(String[] nomv, String[] nomm) {
        int[][] v = new int[nomv.length][nomm.length];
          // Lecture + initialisation des ventes
        for (int i = 0; i < v.length; i++) {
            System.out.println("Ventes␣de␣␣" + nomv[i] + "␣:␣");
            for (int j = 0; j < v[i].length; j++) {
                System.out.print("␣␣nb␣de␣" + nomm[j] + "␣:␣");
                v[i][j] = Terminal.lireInt();
            }
        }
        return v;
    }
    public static void main(String[] args) {
        // Vendeurs et modèles considérés.
        String[] vendeurs = {"andre", "ingemar", "jean_jerome", "cindy", "joey"};
        String[] modeles = {"twingo", "clio", "megane", "velsatis"};

        // Ajouter un appel pour initialiser le tableau de ventes.
        int[][] ventes; // corriger ici

        // Appel pour affichage
        afficherVentes(ventes, vendeurs, modeles);
        }
}
```

1. Ces deux méthodes prennent plusieurs paramètre. Assurez vous de comprendre quel est leur rôle, et comment ses paramètres sont employés dans le corps des méthodes.
2. La méthode main doit invoquer ces sous-programmes de manière pertinente, mais elle ne le fait que partiellement. Compilez et testez ce programme. Quel est le problème ? Complétez la méthode main pour que le programme se comporte comme souhaité.

Question 3

Le fichier `ExoVentesQ2.java` contient une solution à la question 2 de cet exercice, mais sans utiliser des méthodes. Etudiez ce programme et testez le. Il faut bien comprendre encore une fois le lien avec les tableaux de noms de modèles. On veut maintenant écrire une nouvelle version de ce programme qui utilise un sous-programme afin de calculer le nombre total d'exemplaires vendus pour un modèle donné. Dans le fichier `ExoVentesQ2_M_squel.java` on vous propose un squelette de programme à compléter avec le corps de la méthode qui calcule le total de ventes par modèle.

```
public class ExoVentesQ2_M_squel {
    /**
     * Retourne le nombre d'exemplaires vendus de modèle dans la colonne @posModele
     * @param posModele  colonne du modèle
     * @param v  tableau de ventes
     * @return
     */
    public static int nbVendusModele(int posModele, int[][]v){
      // à compléter...
    }

    public static void main(String [] args){

    String [] vendeurs = {"andre","ingemar", "jean_jerome","cindy","joey"};
    String [] modeles =  {"twingo", "clio", "megane", "velsatis"};
    // on donne la liste de valeurs pour gagner du temps lors des tests.
    int [][] ventes = {
      {0,3,2,0},
      {2,3,0,1},
      {1,1,1,1},
      {5,1,0,0},
      {1,1,2,0}};

      // Afficher le total vendu par modèle
      // Pour chaque modèle
      for (int j=0; j< modeles.length; j++){
         Terminal.ecrireString("Nombre␣vendus␣modele␣"+modeles[j]+"␣:␣");
         System.out.println(nbVendusModele(j,ventes));
      }
  }
}
```

Question 3

Modifiez ce programme afin de faire réaliser la boucle d'affichage par un sous-programme. Attention, votre sous-programme doit reprendre le plus fidèlement possible la boucle d'affichage du `main`.

Question 4

Inspirez vous de ces solutions afin de répondre à la question 3 de l'exercice en utilisant des sous-programmes.