TP JAVA n°5
Fichiers

Introduction :

On considère une série d'enregistrements concernant des ventes réalisées par un exportateur de véhicules miniatures. Pour chaque vente, il entre dans son registre de nombreuses informations : - nom de la société cliente - nom et prénom du contact, adresse, téléphone - nombre d'unités vendues - prix de vente - etc... Ces informations sont stockées dans un fichier au format 'csv' (comma separated values) : ventes.csv
Dans un premier temps, regardez son contenu avec un editeur de texte (geany, gedit ou autre...). La première ligne contient les noms des attributs (NUM_COMMANDE, QUANTITE,...). Les ligne suivantes contiennent les valeurs d'attributs correspondant à une vente donnée. en tout plus de 2000 ventes sont répertoriées dans ce fichier.
Ouvrez-le maintenant à l'aide d'un tableur. Les données sont maintenant "rangées" en lignes et colonnes pour faciliter la lecture. Notez bien l'emplacement du fichier ventes.csv dans l'arborescence de fichiers.

Exercice 1 : Lecture de fichier texte et traitement des données

La lecture dans un fichier s'effectue avec la librairie java.io où sont définies les classes FileReader et BufferedReader : `import java.io.*;`
Pour pouvoir lire un fichier, il faut connaître son chemin d'accès, noté ici "/chemin/vers/le/fichier". En Java, ce chemin est nécessaire pour créer un flux de lecture de type FileReader. Pour accéder au contenu du fichier, il faut définir un objet de type BufferedReader servant à charger dans un "tampon" les données du fichier (le "buffer").

```
FileReader f = new FileReader ("/chemin/vers/le/fichier");
BufferedReader b = new BufferedReader(f);
```

Il est important de vérifier que cette opération d'ouverture s'effectue correctement avant de poursuivre le programme (nombreuses possibilités d'erreur : fichier effacé, erreur de nom, pas de droits de lecture,...). On utilise une instruction de test spécifique pour vérifier que l'ouverture du fichier s'est correctement effectuée, de type try...catch... (essaye … sinon …) permettant de prévoir une action de secours lorsqu'une une opération "risquée" échoue.

```
try{
        FileReader f = new FileReader ("/chemin/vers/le/fichier");
        BufferedReader g = new BufferedReader(f);
        System.out.println("L'ouverture s'est bien passée!");
        }
catch (Exception e){
```

```
        System.out.println("Problème d'ouverture!");
}
```

1) Créez un projet Java, recopiez les lignes ci-dessus en remplaçant "/chemin/vers/le/fichier" par l'emplacement réel de ventes.csv, et vérifiez que tout se passe bien.

Comme nous l'avons vu dans l'introduction, le fichier ventes.csv est organisé sous forme de tableau où chaque ligne contient une liste de valeurs. Cette liste de valeurs est couramment appelée un tuple. Chaque ligne du fichier contient un tuple à l'exception de la première qui contient la liste des attributs, c'est à dire le schéma des données.

La première ligne du fichier contient le schéma de données. On remarque que les différents attributs sont séparés par des virgules (la virgule est donc le caractère de séparation.) On souhaite afficher cette liste d'attributs. La commande: `s = g.readLine();` lit une ligne du fichier et la copie dans la variable s, de type "chaîne de caractères" (String). Comme précédemment, il convient d'effectuer un test d'erreur de lecture:

```
try{
s = g.readLine();  }
 catch (Exception e){
System.out.println("Problème de lecture!");
}
```

2) Appliquez la commande readline et affichez s.

s est une simple chaîne de caractères. On souhaite maintenant découper s pour construire un tableau de chaînes de caractères, appelé schema, où chaque case du tableau contient le nom d'un attribut. La commande `s.split(",")` retourne une liste de chaînes de caractères à partir d'une chaîne de caractères s et un caractère séparateur ",".

3) Définissez un tableau de chaînes de caractères schema. schema reçoit ensuite le résultat de la commande split appliquée sur la chaîne s.
4) Créez un entier m qui reçoit le nombre d'attributs (`schema.length`), puis affichez les attributs un par un.

L'objet g est un flux de données. Lorsqu'une donnée a été lue, l'objet positionne sa tête de lecture sur la donnée suivante. Ainsi chaque nouvel appel à la méthode `readLine()` permet de lire une nouvelle ligne (jusqu'à ce qu'on atteigne la fin du fichier).

Ici, les lignes suivantes contiennent des tuples. Commençons par lire le premier tuple.

5) Lisez une nouvelle ligne et affichez-la.

Cette ligne doit être "découpée" pour extraire les différentes valeurs.

6) Extrayez les différentes valeurs dans un tableau nommé tuple. Affichez ce tableau élément par élément sous la forme : ATTRIBUT : valeur.
7) Affichez de même le troisième tuple (sans afficher le deuxième).
8) Une fois ces manipulations effectuées, fermez le fichier avec la commande `f.close()`.

Exercice 3 : La classe Tuple

Nous allons maintenant créer une classe pour gérer dans un même objet **Tuple** les données de type schéma et valeurs.

1) Définissez dans votre projet cette nouvelle **classe**.
2) Le **constructeur** initialise le tableau schema avec le tableau fourni en paramètre, et initialise valeurs avec un tableau vide de même dimension.
3) **setValeurs** permet de remplir le tableau valeurs avec le tableau fourni en paramètre. Pensez à vérifier la compatibilité de dimension...
4) **getSchema** et **getValeurs** permettent de regarder le contenu de schema et valeurs
5) toString affiche le contenu du tuple sous la forme d'une liste : ATTRIBUT1 : valeur1 ATTRIBUT2 : valeur2 etc...
6) Reprenez l'exercice précédent en utilisant cette classe pour stocker les données lues dans le fichier. Si t est un objet de type **Tuple**, `System.out.println(t)`; doit produire le même affichage que dans la question précédente.

Exercice 4 : Encapsulation des opérations d'accès aux données

Comme nous l'avons vu, les opérations d'accès aux données nécessitent d'effectuer des tests try, catch, qui garantissent un traitement efficace des erreurs d'accès. Pour "alléger" le programme principal, nous allons créer une classe **LecteurDeTuples** qui sert à "cacher" ces opérations compliquées...
Pour pouvoir fournir ces services, l'objet contient un attribut descripteur de type **FileReader** et un attribut buffer de type **BufferedReader**. Le tuple courant est stocké dans l'attribut tuple.
Cette classe propose deux opérations principales:

1) Le constructeur initialise le **FileReader** et le **BufferedReader** à partir de l'emplacement du fichier fourni en paramètre, charge le schéma à partir de la première ligne du fichier, et initialise tuple avec le schéma. Si le fichier ne s'ouvre pas correctement, on génère un message d'erreur (grâce à try, catch)
2) La méthode **litTupleSuivant()** déplace le **BufferedReader** sur l'élément suivant et met à jour le tuple courant. Si la lecture ne s'effectue pas correctement, on génère un message d'erreur (grâce à try, catch) La méthode retourne true si l'élément existe, false si on atteint la fin du fichier.
3) Ajouter **getTuple()** qui retourne le tuple courant et **toString()** qui affiche le tuple courant.

Ajoutez la classe LecteurDeTuples à votre projet.

4) Dans le programme principal, remplacez le descripteur `f` et le buffer `g` par un objet unique `h` de type **LecteurDeTuples** (et n'oubliez pas de supprimer également tous les try, catch).

L'accès à l'élément suivant se fait maintenant par `h.litTupleSuivant()`. L'affichage s'effectue maintenant par `System.out.println(h)`.

5) Reprenez la question précédente et vérifiez que le tout fonctionne correctement

Exercice 5 : Extraction de la totalité du fichier

Il est possible de lire le fichier dans sa totalité en séparant les lignes en plaçant h.litTupleSuivant() dans une boucle. Dans le programme principal, ouvrez de nouveau le fichier (en initialisant le **lecteurDeTuples** h) :

```
LecteurDeTuples h = new LecteurDeTuples("/chemin/vers/le/fichier");
```

et exécutez les lignes suivantes :

```
while (h.litTupleSuivant()){
        System.out.println("**************************");
        System.out.println(h);
        }
```

Si vous avez correctement programmé la méthode setValeurs, Il doit y avoir un souci.

1) Essayez de comprendre le problème. Le problème vient-il du fichier ou du programme? Calculez (et affichez) le nombre de tuples incorrects.
2) Résoudre ce problème.