TP CSV

1 Le format csv

Le format CSV (Comma Separated Values ou Valeurs Séparées par des Virgules) est un format textuel d'échange de données tabulaires utilisées notamment par les tableurs et les bases de données relationnelles. Par exemple, des données qui s'affichent comme suit dans un tableur :

	A	B	C	D	E
1	Numero	Nom	Prenom	Date_Naissance	Lieu_naissance
2	ETU00230	DIOP	Moussa	10/12/1990	FATICK
3	ETU00231	FALL	Marieme	10/03/2000	DAKAR
4	ETU00232	NDIAYE	Assane	10/12/1991	ZIGUINCHOR
5	ETU00233	DIOP	Doudou	10/12/1992	THIES
6	ETU00234	FALL	Moussa	10/12/1993	FATICK
7	ETU00235	NDIAYE	Moussa	10/01/1990	DAKAR
8	ETU00236	DIOP	Marieme	10/12/1990	ZIGUINCHOR
9	ETU00237	FALL	Assane	10/03/2000	THIES
10	ETU00238	NDIAYE	Doudou	10/12/1991	FATICK
11	ETU00239	DIOP	Moussa	10/12/1992	DAKAR
12	ETU00240	FALL	Moussa	10/12/1993	ZIGUINCHOR
13	ETU00241	NDIAYE	Marieme	10/01/1990	THIES
14	ETU00242	DIOP	Assane	10/12/1990	FATICK
15	ETU00243	FALL	Doudou	10/03/2000	DAKAR

peuvent être sauvegardées dans un fichier CSV qui contient le texte suivant :

```
1  Numero,Nom,Prenom,Date_Naissance,Lieu_naissance
2  ETU00230,DIOP,Moussa,10/12/1990,FATICK
3  ETU00231,FALL,Marieme,10/03/2000,DAKAR
4  ETU00232,NDIAYE,Assane,10/12/1991,ZIGUINCHOR
5  ETU00233,DIOP,Doudou,10/12/1992,THIES
6  ETU00234,FALL,Moussa,10/12/1993,FATICK
7  ETU00235,NDIAYE,Moussa,10/01/1990,DAKAR
8  ETU00236,DIOP,Marieme,10/12/1990,ZIGUINCHOR
9  ETU00237,FALL,Assane,10/03/2000,THIES
10 ETU00238,NDIAYE,Doudou,10/12/1991,FATICK
11 ETU00239,DIOP,Moussa,10/12/1992,DAKAR
12 ETU00240,FALL,Moussa,10/12/1993,ZIGUINCHOR
13 ETU00241,NDIAYE,Marieme,10/01/1990,THIES
14 ETU00242,DIOP,Assane,10/12/1990,FATICK
15 ETU00243,FALL,Doudou,10/03/2000,DAKAR
```

On voit que dans ces fichiers, les données ne sont pas alignées visuellement comme dans un tableur, mais les informations des colonnes sont séparées par une virgule. La première ligne contient les entêtes de colonne et la première case de chaque ligne contient un entête de

ligne. Certaines variantes de fichier CSV utilisent un point-virgule ou une tabulation comme séparateur de colonne.

2 Lecture dans un fichier csv

Pour lire le fichier CSV et pour lire dans un fichier contenant du texte en général, nous vous proposons d'utiliser un scanner exactement comme pour les lectures au clavier. On utilisera donc les méthodes **nextLine**, **nextInt**, ect. Il y a quand même deux différences avec les lectures au clavier : à la création du scanner, au lieu de donner l'objet System.in en paramètre au constucteur, on utilise un autre constructeur qui prend en paramètre un objet représentant en Java le fichier à ouvrir. Il faut commencer par créer cet objet. L'autre différence est qu'on peut toujours lire du nouveau au clavier alors qu'un fichier contient un texte limité : une fois qu'on a lu tout le fichier, on ne peut plus rien lire dedans. L'exemple suivant vous montre comment lire dans un fichier ligne par ligne.

```
1  package tp_csv;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.util.Scanner;
6
7  public class question0 {
8
9      public static void main(String[] args) {
10         // TODO Auto-generated method stub
11
12         String ligne;
13         try {
14             File file = new File("G://Mes documents/COURS/Cours format de données/exo/CSV/etudiant.csv");
15             Scanner scan = new Scanner(file);
16             int i=1;
17             while (scan.hasNextLine()) {
18                 ligne = scan.nextLine();
19                 System.out.println("Ligne "+i+" : "+ligne);
20                 i++;
21             }
22             scan.close();
23         } catch (FileNotFoundException e) {
24             // TODO: handle exception
25             System.out.println("Le fichier n'existe pas");
26         }
27
28     }
29
30 }
```

Dans la ligne : **java.io.File file = new java.io.File("G://Mes documents/COURS/Cours format de données/exo/CSV/etudiant.csv");** il y a création d'un objet de type File. Le paramètre du constructeur est une chaîne de caractère contenant le nom du fichier à ouvrir. Si le fichier n'existe pas, l'exception **FileNotFoundException** est levée. La méthode **hasNextLine** de la classe Scanner permet de tester s'il reste ou non des lignes à lire. À la fin du fichier, elle renvoi **false**. A noter qu'il est très important de fermer le canal de lecture avec **scan.close();**

3 Quelques méthodes utiles

Pour réaliser le TP, vous aurez à utiliser des méthodes des classes **String** et **Integer**.

3.1 Classe String

- **length()** : renvoie la longueur de la chaîne
- **trim()** : renvoie une chaîne dans laquelle les espaces en début et en fin de chaîne ont été supprimés. Par exemple " truc chose ".**trim()** renvoie la chaîne "truc chose".
- **split(String s)** : découpe la chaîne en plusieurs morceaux en utilisant la chaîne **s** comme séparateur. Le résultat est un tableau de chaînes. Par exemple "un!deux!trois".split("!") renvoie le tableau "un","deux","trois".

3.2 Classe Integer

- **Integer.parseInt** : prend en paramètre une chaîne et renvoie un entier. Par exemple Integer.parseInt("-58") renvoie l'entier -58.

4 Exercice

Question 1 :

Écrivez un programme **java** qui lit un fichier au format CSV et qui vérifie que toutes les lignes ont bien le même nombre de colonnes. Pour cela, vous pourrez lire le fichier ligne par ligne et utiliser la méthode **split** pour séparer les différents champs d'une ligne qui sont séparées par une virgule.

Question 2 :

Modifiez le programme pour que le nom du fichier CSV soit lu au clavier. Pour ce faire, il faut utiliser deux **scanner** : un qui lit le clavier pour lire le nom du fichier, l'autre qui lit le fichier pour vérifier qu'il y a bien le même nombre de colonnes sur chaque ligne.