TP CSV et SVG

NFA032

1 Préliminaires : le format CSV

Le format CSV (Comma Separated Values ou Valeurs Séparées par des Virgules) est un format textuel d'échange de données tabulaires utilisée notamment par les tableurs et les bases de données relationnelles.
Une ligne de tableau est représentée par une ligne CSV et les valeurs des différentes colonnes sont séparées par des points-virgules. Les champs textuels peuvent être entourés de guillemets. Exemple de ligne au format CSV :

```
"Hugo";"Victor";26/02/1802
```

2 Lecture dans un fichier CSV

Pour lire dans un fichier contenant du texte, et en particulier pour lire le fichier CSV dans ce Tp, nous vous proposons d'utiliser un scanner (classe des bibliothèques Java) exactement comme pour les lectures au clavier. On utilisera donc les méthodes nextLine, nextInt, ect. Si vous n'êtes pas familier avec cette classe nous vous invitons à explorer sa documentation sur le site d'Oracle. Si par ailleurs vous n'êtes pas encore familiarisé avec les entrées/sorties à partir ou vers des fichiers nous vous conseillons vivement de **lire les transparents à ce sujet de l'UE NFA035 et de travailler les Tps qui s'y rapportent**.

Lors de la lecture via la classe `Scanner` on doit passer au constructeur de cette classe un objet de de type `File` à partir duquel va s'effectuer la lecture. On lui passera l'objet `System.in`, on utilise un autre constructeur qui prend en paramètre un objet représentantsi l'on veut que la lecture se au clavier. Dans cette exercice, on veut lire à partir d'un fichier : on passera au constructeur un objet qui représente ce fichier. Il faudra donc commencer par créer cet objet, par exemple en utilisant le constructeur de la classe `File`.

Lors d'une lecture au clavier, on peut lire à tout moment, 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.

```
import java.io.*;
import java.util.Scanner;
import java.util.NoSuchElementException;
```

```
public class LectureFichier{
   public static void main(String[] args){
      String line;
      int i=0;
      try{
            File file = new File("test.txt");
            Scanner scan = new java.util.Scanner(file);
            while(scan.hasNextLine()){
                i++;
                line = scan.nextLine();
                System.out.println("ligne " + i + ": " + line);
            }
            scan.close();
            System.out.println("Le fichier est composé de "+i+" lignes.");
      }catch(FileNotFoundException ioex){
             System.out.println("le fichier n'existe pas");
      }catch(NoSuchElementException nsex){
            System.out.println("fini");
      }
  }
}
```

Dans la ligne : `java.io.File file` new java.io.File("test.txt");= 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.

S'il n'y a plus rien à lire dans le fichier, la méthode `nextLine` lève l'exception `NoSuchElementException`. C'est le cas également de toutes les autres méthodes de lecture (nextInt, nextDouble, etc). 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 pouvez utiliser des méthodes des classes `String`, `Character` et `Integer` qui peuvent vous être utiles.

3.1 Classe `String`

- `length()` : renvoie la longueur de la chaîne
- `charAt(int i)` : prend en paramètre un nombre et renvoie le caractère (type `char`) qui a pour rang ce nombre. Par exemple `"abc".charAt(0)` renvoie le `char 'a'`.
- `s1.toLowerCase()` et `s1.toUpperCase()` renvoient une nouvelle chaîne égale à `s1` mais avec toutes les lettres en minuscule et en majuscule respectivement.

— `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"`.
— `indexOf(String s)` : renvoie l'indice de la première occurrence de la chaîne `s` dans la chaîne. Par exemple `"un deux trois".indexOf("deux")` renvoie `3`, car la chaîne `"deux"` commence à l'indice `3` de la chaîne `"un deux trois"`.
— `substring(int debut, int fin)` : renvoie la sous-chaîne de la chaîne sur laquelle la méthode est appelée comprise entre les indices `debut` et `fin`. Le caractère d'indice `debut` est inclus dans le résultat, mais pas celui d'indice `fin`. Par exemple `"bonjour".substring(2,4)` renvoie la sous-chaîne "nj". Autrement dit, elle renvoie la sous-chaîne comprenant les caractères d'indice `2` et `3`.

3.2 Classe `Integer`

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

3.3 Classe `Character`

— `Character.isDigit(char c)` : prend un char en paramètre et renvoie un booléen disant si oui ou non ce caractère est un chiffre.
— `Character.isLetter(char c)` : prend un char en paramètre et renvoie un booléen disant si oui ou non ce caractère est une lettre.

Question 1

Écrivez un programme qui lit un fichier (de plusieurs lignes) au format CSV. Chaque ligne du fichier contiendra les données d'une personne définies sur trois champs : nom, prénom et date de naissance au format XX/XX/YYYY. Votre programme doit vérifier 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 cellules d'une ligne qui sont séparées par un point-virgule. Le programme doit signaler une erreur si toutes les lignes n'ont pas le même nombre de colonnes, et afficher le nombre de lignes lues à la fin. Attention, votre programme doit fonctionner quelque soit le nombre de colonnes du fichier CSV lu et non seulement pour trois colonnes comme ici.

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.

Question 3

On considère que chaque ligne du fichier CVS correspond à un objet `Personne` que votre programme doit créer et initialiser avec les données lues. Modifiez votre programme de manière à fabriquer la liste de toutes les personnes dont les données sont lues à partir d'un nom de fichier lu au clavier. Bien entendu il faudra auparavant créer les classes nécessaires : Personne et Date. Comment pour la question 2 vous utiliserez deux scanner. Votre programme doit afficher les données de la personne la plus jeune et la plus vieille de la liste.

4 Ecriture d'un fichier au format SVG

Pour ce qui est d'écrire dans un fichier, nous allons procéder comme pour les affichages à l'écran avec les méthodes `print` et `prinln`, mais au lieu de les appeler sur `System.out`, on va les appeler sur un objet représentant un fichier en écriture. Pour ce faire, il faut commencer par créer l'objet en question. Puis on écrit dans le fichier avec des `print` et `prinln`. Enfin, on ferme le fichier. Cette opération de fermeture est celle qui finalise le canal de communication avec le fichier. Cette opération finale est nécessaire pour assurer que le fichier est correctetment écrit.

Voici un programme qui écrit un fichier au format SVG.

```
import java.io.*;
import java.util.ArrayList;
public class EcritureSVG{
  public static void main(String[] args){
     try{
          PrintWriter pw = new PrintWriter("exemple.svg");
          pw.println("<?xml version=\"1.0\" encoding=\"utf-8\"?>");
          pw.println("<svg xmlns=\"http://www.w3.org/2000/svg\"");
          pw.println("     version=\"1.1\" width=\"300\" height=\"200\">");
          pw.println("<circle cx=\"90\" cy=\"80\" r=\"50\" fill=\"blue\" />");
          pw.println("</svg>");
          pw.close();
        }catch(FileNotFoundException ex){
          System.out.println("Nom de fichier incorrect " + ex.getMessage());
        }
  }
}
```

À noter dans ce programme : si l'on veut mettre un guillemet à l'intérieur d'une chaîne de caractère il faut la faire précéder d'une barre oblique. En ce qui concerne le nom de fichier, il peut être incorrect s'il spécifie un chemin d'accès qui n'existe pas.

Attention, méfiance : Java ne va pas vérifier si le fichier existe déjà et vous demander de confirmer que vous voulez le réécrire : il va écraser le fichier existant avec les informations écrites par la programme.

Question 1

Modifiez le programme pour que le nom de fichier soit entré au clavier.

Question 2

Modifiez le programme pour que la couleur du cercle soit entrée au clavier.

4.1 Création interactive d'une image

On vous donne le code d'une méthode qui permet de choisir deux opérations : ajouter un cercle ou ajouter un rectangle dans l'image et le squelette d'une méthode main qui permet d'ajouter des cercles et des rectangles à différents endroits et de différentes couleurs. Complétez le code pour que le programme crée une image comportant les formes demandées dans un fichier SVG.

```
import java.io.*;
import java.util.*;
public class InteractiveSVG{
    private Scanner scan;
    public int[]  saisirCoordonnees(){
        int[] res = new int[2];
        System.out.print("Entrez la coordonnée X: ");
        res[0] = scan.nextInt();
        System.out.print("Entrez la coordonnée Y: ");
        res[1] = scan.nextInt();
        scan.nextLine();
        return res;
    }
    public String saisirCouleur(){
        String res;
        System.out.print("Entrez la couleur que vous voulez: ");
        res = scan.nextLine();
        return res;
    }
    public int menu(){
        int res;
        System.out.println("1- Ajouter un rectangle");
        System.out.println("2- Ajouter un cercle");
        System.out.println("3- Créer l'image");
```

```
        do{
            System.out.print("Votre choix: ");
            res = scan.nextInt();
            scan.nextLine();
            if (res<1 || res>3)
                System.out.println("Choix incorrect: " + res);
        }while(res<1 || res>3);
        return res;
    }
    public void creeImage(){
        scan = new Scanner(System.in);
        ... A compléter
        do{
            choix = menu();
            if (choix == 1){
              ... A compléter

            }else if (choix == 2){
              ... A compléter
            }else if (choix == 3){
              ... A compléter
            }
        }while(choix !=3);
    }
    public  static void main(String[] args){
        InteractiveSVG isvg = new InteractiveSVG();
        isvg.creeImage();
    }
}
```

5 Utilisation d'objets

Écrivez maintenant une classe permettant de représenter une forme (cercle ou rectangle). Cette classe comprendra l'information sur la nature de la forme et sur les paramètres de la forme. Elle aura un constructeur et une méthode toString qui renverra le code SVG correspondant à la forme.

Modifiez la classe InteractiveSVG pour qu'elle construise d'abord un ArrayList de formes (ou une tableau de formes), puis utilise cet ArrayList ou ce tableau pour créer le fichier SVG.

6 Héritage

Créez deux ou trois classes utilisant l'héritage de telle sorte qu'il ya ait une classe pour les cercles et une autre pour les rectangles.