

NFA032 : TP 2

Création et utilisation d'objets

19 février 2018

Le but de ce TP noté est d'apprendre à utiliser des objets de classes java prédéfinies. Il conviendra de créer des objets de ces classes prédéfinies (notamment `Date` et `String`) et de les manipuler en utilisant les méthodes publiques de ces classes, afin de réaliser un traitement. À la fin du TP, vous devrez remettre une archive zip sur votre espace numérique de formation (ENF, sur le site www.lecnam.net, comme vous l'avez fait lors du premier TP).

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.

2 Le traitement à réaliser

On désire écrire un programme qui lit au clavier une ligne au format CSV contenant trois champs (trois colonnes) séparées par des points-virgules. Le premier est le nom d'une personne, le second son prénom et le troisième contient la date de naissance. Le nom et le prénom sont entourés de guillemets (signe "). La date est donnée au format jj/mm/aaaa (jour avec deux chiffres, mois avec deux chiffres, année avec 4 chiffres, séparés par une barre oblique). Le programme devra afficher les trois données sur trois lignes différentes. Le nom et le prénom doivent être affichés sans les guillemets. La date doit être affichée en créant un objet de type `Date` représentant cette date et en appelant la méthode `toString()` sur cet objet.

— Exemple de ligne au format CSV :

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

— Exemple d'exécution du programme :

```
> java CSV
Entrez la ligne CSV: "Hugo";"Victor";26/02/1802
Hugo
Victor
Fri Mar 26 00:00:00 CET 1802
```

— Exemple avec une ligne incorrecte :

```
> java CSV
Entrez la ligne CSV: Victor Hugo 26 février 1802
Erreur de format
```

3 Méthodes utiles

Pour réaliser le TP, vous devez utiliser des méthodes des classes `String` et `Date`. Nous mentionnons ici quelques méthodes qui peuvent vous être utiles (ce qui ne signifie pas que votre programme doive nécessairement toutes les utiliser).

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.

3.4 Classe Date

- Pour utiliser la classe `Date`, vous devez importer le paquetage suivant : `import java.util.Date;`
- Il existe un constructeur prenant en paramètre trois entiers, le premier contenant l'année moins 1900, le second le mois, le troisième le jour. Par exemple, pour créer un objet pour le 14 juillet 1789, il faut appeler le constructeur comme suit : `new Date(1789-1900,7,14)` ou si vous préférez `new Date(-111,7,14)`. Pourquoi faut-il enlever 1900 à l'année? C'est très étrange et il n'y a pas de justification possible, mais c'est comme cela.
- `toString()` : appliquée à un objet de la classe `Date`, cette méthode affiche la date au format de type `Fri Aug 14 00:00:00 CET 1789`.

4 Déroulement du TP

1. Créez un projet java dans Eclipse nommé `Tp2`.
2. Créez dans ce projet une classe `CSV`.
3. Dans la méthode `main()`, demandez à l'utilisateur d'entrer une ligne au format CSV et stockez là dans une variable nommée `ligne`.
4. Décomposez la ligne en ses 3 champs (nom, prénom, date de naissance) et stockez les dans 3 chaînes de caractères.
5. Décomposez la chaîne de caractère contenant la date de naissance afin d'isoler le jour, le mois et l'année que vous convertirez en entiers. Créez alors un objet de la classe `Date` en utilisant le constructeur à 3 paramètres décrit plus haut.
6. Affichez le résultat.
7. Modifiez votre programme pour tester les cas d'erreur (format CSV non respecté : nombre de champs différents de 3, date au format incorrect).
8. Créez une archive zip et déposez la sur votre ENF en utilisant le formulaire de remise du `Tp2`.

5 Pour aller plus loin...

Nous vous proposons 3 versions améliorées du TP, à n'envisager que si vous avez déjà remis votre projet. Vous pourrez éventuellement modifier votre archive zip et redéposer sur l'ENF une version améliorée.

1. Écrire une version améliorée dans une classe `CSV1.java` qui extrait les informations même de certains formats incorrects (par exemple avec un autre séparateur pour les champs ou les nombres de la date). Vous pourrez demander à l'utilisateur d'indiquer les séparateurs utilisés ou bien créer une liste de séparateurs courants (barre verticale |, virgule, espace).
2. Écrire une version dans une classe `CSV2.java` qui fonctionne même s'il manque certains guillemets.
3. Écrire une version dans une classe `CSV3.java` qui lit un fichier CSV contenant plusieurs lignes au format CSV, stocke dans une `ArrayList` les dates de naissance de chacune des lignes (sans stocker les noms et prénoms), puis affiche les éléments de l'`ArrayList` créée. Voici un exemple de classe `LireLigne` qui utilise un `BufferedReader` pour lire un fichier texte sur le disque et l'afficher à l'écran ligne par ligne. Vous pourrez vous en inspirer pour cette dernière version améliorée.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;

class LireLigne {
    public static void main(String[] argv) throws IOException
    {
        BufferedReader lecteurAvecBuffer=null;
        String ligne;
        String exempleCheminWindows = "C:\\Users\\Eric Soutil\\Desktop\\essai.csv";
        String exempleCheminLinux = "/home/jean/essai.csv";

        try {
            lecteurAvecBuffer = new BufferedReader(new FileReader(exempleCheminWindows));
            while ((ligne = lecteurAvecBuffer.readLine())!=null) {
                System.out.println(ligne);
                // ou traitement de la ligne
            }
            lecteurAvecBuffer.close();
        }
        catch(FileNotFoundException exc) {
            System.out.println("Erreur d'ouverture");
        }
    }
}
```