FIP ING39 – Entrées/Sorties, Tp noté

S. Rosmorduc

25 septembre 2019

Le projet qui vous est fourni contient déjà des jeux de tests. Ceux que nous utiliserons sont un peu plus complet. Nous vous invitons à ne pas hésiter à ajouter des tests par vous-même.

On vous demande d'écrire un certain nombre de méthodes. Vous avez le droit, si cela vous semble utile, d'écrire des méthodes auxiliaires pour vous simplifier la vie.

L'usage de la classe `StreamTokenizer` n'est pas utile pour ces questions.

1 Le format CSV

Le format CSV (Comma separated values) est un format de fichier tableur (excel, gnumeric, etc) très simple : chaque ligne du fichier correspond à une ligne du tableur, et chaque cellule de la ligne (y compris les cellule vides) est séparée de la suivante par un *caractère de séparation* (la virgule à l'origine, mais nous utiliserons le point-virgule).

Par exemple, le fichier `notes.csv`, qui contient :

```
Turing;Alan;12
King;Augusta Ada;17
Babbage;Charles Henry;15
```

correspond au tableau :

Turing	Alan	12
King	Augusta Ada	17
Babbage	Charles Henry	15

Pour vous simplifier la vie, nous supposerons dans ce TP que la fin de ligne est toujours '`\n`' (sauf dans la dernière question, sur 2 points).

Le projet va vous amener à écrire plusieurs lecteurs et écrivains spécialisés pour des versions de plus en plus puissantes de CSV. Ces classes permettront de lire ou d'écrire le contenu d'un tableau, représenté en mémoire par un objet de classe `CSVTable`.

Question 1 : version simple des fichiers CSV

Un fichier CSV est un fichier texte qui représente un tableau. Les *lignes* du tableau correspondent aux lignes du fichier ; les cases du tableau sont séparées par des points virgules (" ;"). Notez que si deux " ;" se suivent, cela correspond à une case vide.

Dans cette représentation, une case ne peut contenir :

— ni le caractère « ;»

— ni un saut de ligne

Par ailleurs, on considère que :

— la fin de ligne (`\n`) *termine* une ligne. Donc, si un fichier contient une seule ligne de texte, terminée par un caractère de fin de ligne, il fait *une* ligne, et non deux.

— pour être cohérent avec le fait que deux " ;" qui se suivent correspondent à une case vide, on considère qu'une ligne vide (constituée simplement d'une fin de ligne, et rien d'autre), contient une case vide (et fait donc au moins une colonne).

Écrivez les classes `CSVWriter1` et `CSVReader1`.

On pourra éventuellement penser à utiliser la classe `BufferedReader`.

Exercice 2 : version plus complète des fichiers CSV

On souhaite proposer une nouvelle version des fichiers, dans lesquels il sera possible d'include des " ;" dans un texte. Pour cela, on décide que dans tout texte compris entre guillemets : `"...."`, les " ;" ne seront pas pris comme des séparateurs, mais comme des signes "normaux".

Le fichier :

```
Alfred;"1;2"
Toto;4
```

correspondra alors au tableau :

Alfred	1 ;2
Toto	4

On s'interdit toujours les passages à la ligne, et, de plus, on s'interdit les guillemets comme valeur normale dans une case.

Écrivez la classe`CSVReader2`.

Exercice 3 : version définitive des fichier CSV

La version 3 des fichiers CSV permet de stocker tout type de tableau. On prend les conventions suivantes :

— Si le contenu d'une case est écrit entre guillemets, celle-ci peut contenir des " ;" ou même des passages à la ligne ;

— Il est possible d'include des guillemets à l'intérieur de guillemets, en les doublant.

Ainsi, le texte :

```
un;"un␣essai
pour␣voir";deux
"on␣""protège""␣le␣texte";trois
```

Correspond au tableau

un	un essai\npour voir	deux
on "protège" le texte	trois	

Écrivez la classe `CSVReader3`.

Note : pour cette version, on vous fournit un squelette de classe (que vous pouvez suivre ou non). Si vous ne voulez pas le suivre, vous pouvez vous débarrasser d'à peu près tout ce qui est `private`.

Ici, on va lire le texte caractère par caractère, en prenant la convention suivante (très souvent pratique lors de la lecture de fichiers complexes) :

— le dernier caractère lu est facilement accessible (on en fait ici une variable d'instance) ;

— on suppose que (et on fait en sorte que !!!), quand on a terminé d'analyser le contenu d'une partie du texte, par exemple, une case du tableau, le caractère "lu" soit le caractère qui *suit* le texte qui vient d'être analysé. Dans notre cas, après une case, le caractère lu devrait être, soit " ;", soit une fin de ligne, soit une fin de fichier (-1).

Question 4 (non notée)

Créez pour les formats 2 et 3 les classes `CSVWriter2` et `CSVWriter3`.