




TP MELOG n°II: application des structures linéaires aux fichiers

L'objectif de ce TP est d'appliquer les principes de la structuration des données pour mettre en œuvre un programme de traitement de fichiers. La taille importante des fichiers considérés impose que la conception des algorithmes des programmes demandés conduise au choix des structures les plus adaptées pour éviter d'en arriver à des temps d'exécution excessifs, voire rédhibitoires.

0 Préliminaires

Utiliser les fichiers joints ( Tp.java et  Contenu.java et  texte.txt) pour se familiariser avec la lecture d'un fichier de texte, le découpage et un traitement sur ses mots, et l'écriture dans un nouveau fichier.

Question :

- ▶ Exécuter le programme Tp.

1 Mise au point d'un premier lexique

Nous cherchons dans un premier temps à créer un lexique de tous les mots contenus dans un texte donné.

1.1 Premier texte

Le texte en question est dans un fichier qui s'appelle  Textes/lesMiserables_A.txt et il contient le roman « Les Misérables » de Victor Hugo¹. Il est fourni (cliquer sur le nom du fichier dans ce paragraphe).

Question :

- ▶ Définissez et implantez en java les classes nécessaires pour effectuer au moins les opérations suivantes :
 - constituer, à partir d'un texte contenu dans un fichier d'entrée, un *lexique* des mots utilisés avec le nombre d'occurrences pour chaque mot ;
 - sauvegarder le contenu du lexique ainsi produit dans un fichier de sortie ;
 - afficher le nombre de mots différents présents dans le lexique ;
 - tester la présence d'un mot dans le lexique et retourner son nombre d'occurrences ;
 - supprimer un mot du lexique.

Les algorithmes non triviaux (à l'instar de l'algorithme permettant de construire le lexique) devront figurer dans votre rapport.

Le lexique sera constitué dans un ordre que vous choisirez et préciserez dans votre compte-rendu.

- ▶ Justifiez le choix de la structure de données que vous avez choisie pour stocker les éléments constitutifs du lexique.
- ▶ Illustrez le bon fonctionnement de votre programme sur des jeux d'essai basés sur la manipulation du lexique associé au texte `lesMiserables_A.txt`.

En ajoutant les instructions suivantes respectivement au début et à la fin du "main" :

```
long startTime = System.currentTimeMillis();  
long stopTime = System.currentTimeMillis();
```


évaluez les temps de traitement, éventuellement en comparant par rapport à l'utilisation d'une autre structure de données.

1. Traduction en anglais pour éviter les accents dont le codage peut poser problème.

2 Enrichissement du lexique

Nous cherchons maintenant à enrichir le premier lexique avec les mots contenus dans un deuxième texte.

2.1 Deuxième texte

Ce deuxième texte se trouve dans un nouveau fichier qui s'appelle  `notreDameDeParis.A.txt` et il contient le roman « Notre Dame de Paris » de Victor Hugo. Il est également fourni (cliquer sur le nom du fichier dans ce paragraphe).

Questions :

- ▶ Écrivez l'algorithme qui permette d'examiner un texte contenu dans un second fichier, d'enrichir le lexique avec les mots nouveaux et de mettre à jour le nombre d'occurrences de chacun des mots. Complétez les classes java précédemment définies pour que pareille opération puisse être assurée.
- ▶ Énumérez les nouveaux mots repérés dans cette seconde phase.
- ▶ Illustrez le bon fonctionnement de votre programme et indiquez les temps d'exécution.

3 Question bonus

Pour chaque mot, il faut maintenant enregistrer la liste de toutes les lignes où ce mot apparaît dans le texte. (remarque : les instructions de lecture en **Java** (voir ci-dessous) permettent de lire ligne par ligne ; c'est ceci qui permettra d'attribuer un numéro à chaque ligne)

Question :

- ▶ Revoyez le programme de création du lexique en tenant compte de cette nouvelle information à fournir.

4 Question super-bonus

Nous voulons maintenant connaître les mots les plus employés par Victor Hugo (dans les romans analysés).

Question :

- ▶ Constituez la liste des quinze mots de plus de trois lettres le plus fréquemment trouvés.

Quelques indications et rappels sur la lecture d'un fichier texte, la gestion par ligne, et par mot :

- ▶ Lecture d'un fichier texte (création d'un flot de caractères) :

```
Reader reader = new FileReader("xxx.txt");
```

- ▶ Prise en compte d'une ligne (création d'un flot de lignes) :

```
BufferedReader in = new BufferedReader(reader);
String ligne = in.readLine();
while (ligne != null) {
    // ... analyse de la ligne courante ...
    ligne = in.readLine(); // lecture de la ligne suivante }
}
```

- ▶ L'analyse d'une ligne se fait à l'aide d'un `StringTokenizer` qui permet de la découper en mots séparés les uns des autres par un ou plusieurs caractères délimiteurs. Ces caractères sont spécifiés dans une chaîne de caractères. Les mots sont obtenus successivement en invoquant la méthode `nextToken`, tandis que `hasMoreTokens` teste l'existence d'un mot suivant :

```
String delimites = " '.,;:?!(){}[] '<><>\"\\t";
StringTokenizer st = new StringTokenizer(ligne, delimites);
while (st.hasMoreTokens()) {
    String mot = st.nextToken();
    // ... traitement du mot courant ...
}
```