Enseignement de spécialité
Informatique et sciences du numérique
Formation des IA-IPR et chargés de mission
Atelier de programmation 4

David Pichardie, Luc Bougé

Jeudi 17 mars

Dans cette activité, nous allons manipuler des grands textes sous forme de listes de mots. Nous utiliserons des structures de données plus avancées que les tableaux : les collections qui vous seront présentées en séance.

La fonction

```
List<String> lireGrandTexte(String nomDefichier)
```

vous permet de charger un texte sous la forme d'une liste de chaînes de caractères. Nous travaillerons plus spécialement sur deux textes: *La Marseillaise* et *Les Misérables*. Le premier est obtenu par la commande

```
List<String> la_marseillaise = lireGrandTexte("La_Marseillaise.txt") ;
```

et le deuxième par

```
List<String> les_miserables = lireGrandTexte("Les_Miserables.txt");
```

Ces deux textes sont libres de droit et consultable en ligne. Pour simplifier, nous enlevons toutes les majuscules et supprimons la ponctuation de ces textes.

Exercice .1. Écrire un programme

```
void afficheLaMarseillaise()
```

qui charge le texte de la *Marseillaise* et affiche tous les mots qui le compose a l'aide d'une itération.

Remarque. *Attention, ne tenter pas la même opération avec* Les Misérables *car le document est beaucoup trop gros pour les capacités d'affichage de Javascool (plus de trois millions de caractères) !*

Exercice .2. Écrire un programme

```
int nombreDeMots(List<String> liste)
```

qui compte le nombre de mots dans une liste de mots. Tester sur le texte des *Misérables*. Remarquer que cette fonctionnalité existe déjà avec l'opération l.size(); sur une liste l mais nous vous demandons de réaliser vous-même ce calcul à l'aide d'une itération.

Exercice .3. La liste calculée par la fonction lireGrandTexte contient une chaîne de caractères spéciale "<NOUVELLE PAGE>" qui est insérée dans la liste à chaque changement de page. Utiliser ce mot pour calculer le nombre de page avec une fonction

```
int nombreDePages(List<String> liste)
```

Compter ainsi le nombre de pages dans *Les Misérables*.

Exercice .4. Écrire un programme

```
Map<String,Integer> tableDesFrequences(List<String> liste)
```

qui calcule la table d'association reliant chaque mot à sa fréquence d'apparition (nombre d'occurrence) dans la liste. Tester sur le texte de *La Marseillaise*.

Exercice .5. Écrire un programme

```
int nombreDeMotsDistincts(List<String> liste)
```

qui compte le nombre de mots distincts dans une liste de mots. Tester sur le texte des *Misérables*.

Exercice .6. Écrire un programme

```
void portraitDeCosette()
```

qui affiche les 100 premiers mots rencontrés après la première apparition du mot "cosette" (sans majuscules).

Exercice .7. Nous vous fournissons la fonction

```
Map<String,Integer> triMapParValeursDecroissantes(Map<String,Integer> table)
```

qui renvoie une version triée d'un table en triant les fréquences par ordre croissant. Utiliser cette fonction pour écrire un programme

```
void lesPlusFrequents(List<String> liste, int N)
```

qui affiche les N mots les plus fréquents de la liste liste en ignorant les mots de moins de 4 lettres. Utiliser cette opération avec N égal à 30 pour voir citer les personnages principaux des *Misérables*.

Exercice .8. Utiliser le mot spécial "<NOUVELLE PAGE>" pour calculer une table d'index pour chaque mot d'au moins 4 lettres du texte avec une fonction

```
Map<String,List<String>> tableIndex(List<String> liste)
```

Afficher la table d'index des 30 mots les plus fréquents.