

Table de hachage, generics, implantations de liste

Exercice 1 - Traitement de texte

On souhaite créer un petit traitement de texte permettant de faire des transformations sur des lignes de texte.

Ecrire un petit programme qui prend un nom de fichier contenant un texte et affiche celui-ci ligne à ligne sur la sortie standard.

Vous utiliserez pour cela la classe `java.util.Scanner`.

Transformer votre programme pour qu'il affiche les lignes en minuscule ou en majuscule suivant l'option de la ligne de commande.

```
java Text -upper input.txt
```

-upper met la ligne en majuscule

-lower met la ligne en minuscule

si la ligne de commande ne contient ni -upper ni -lower affiche chaque ligne de façon identique

Eviter d'écrire des `if ... else` pour associer à une option l'action à effectuer.

Que doit-on faire si l'option demandée n'existe pas, par exemple -toto.

Modifier votre programme pour que si l'utilisateur ne spécifie pas de nom de fichier, l'entrée standard (`System.in`) soit utilisée

Modifier votre programme pour qu'il puisse prendre une liste d'options suivi du nom du fichier, les options sont alors appliquées pour chaque ligne dans l'ordre d'apparition sur la ligne de commande :

Ajouter l'option -reverse qui met une ligne à l'envers.

Ajouter l'option -replace qui remplace les "!" par des "*".

Exercice 2 - Performance sur les listes

Le but de cet exercice est de tester les différences de performance entre les classes `ArrayList` et `LinkedList` sur différents algorithmes.

1 Nous allons dans un premier temps chronométrer le temps d'un parcours d'une `ArrayList` contenant un million (1 000 000) d'entiers en utilisant un `Iterator` (pour le parcours).

Utilisez la méthode `System.nanoTime()` pour effectuer une mesure de temps.

2 Modifier le code pour pouvoir facilement chronométrer le parcours dans le cas d'une `ArrayList` ou d'une `LinkedList`.

Effectuer les tests suivants sur les deux implémentations de `List` :

- en insérant un million d'entiers en première position dans une liste vide au départ (comme pour une file).
- parcours de la liste d'un million d'entiers par un itérateur
- parcours de la liste d'un million d'entiers par un itérateur en sens inverse
- parcours de la liste d'un million d'entiers par un index

Comparer les différents résultats et expliquer les différences.

Exercice 3 - Générification

Le but de cet exercice est de généraliser les classes `fr.uml.v.datas.LinkedList` et `fr.uml.v.datas.Link`

- 1 Paramétriser la classe `fr.uml.v.datas.LinkedList` pour que celle-ci soit générique.
- 2 Modifier la classe `fr.uml.v.datas.main.Main` en conséquence.