TD1 (application): un correcteur orthographique

Avant de commencer, récupérez l'archive du TD sur moodle. Prenez votre classe "TableauGenerique" et mettez-là dans le même répertoire que les autres fichiers de l'archive.

Classe LecteurDico

Cette classe permet de lire un fichier texte qui contient des mots (un par ligne) et de créer un tableau de String qui est utilisé ensuite pour travailler. Il vous suffit pour cela d'utiliser son constructeur en donnant en paramètre le chemin vers le fichier, comme illustré dans le main. Il vous faut donc changer le main en mettant le chemin vers votre dictionnaire puis compiler (javac *.java). Ignorez le message suivant qui est seulement un warning.

```
Note: CorrecteurOrthographique.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Vous pouvez ensuite lancer la classe `LecteurDico` (java LecteurDico) pour vérifier que tout est en place. Elle affiche un mot du dictionnaire, vous pouvez en afficher d'autres.

Correcteur orthographique

Il s'agit d'écrire un correcteur orthographique qui utilise un dictionnaire afin de corriger des textes. Pour ne pas avoir de problème avec l'encodage des accents en français, il s'agit d'un correcteur orthographique en anglais. Afin de corriger les textes, il faudra chercher les mots dans le dictionnaire anglais en utilisant les classes `LecteurDico` pour charger le dictionnaire et `TableauxGenerique` pour utiliser les méthodes de recherche dans le tableau.

1. **Classe CorrecteurOrthographique**. Cette classe contient un dictionnaire qui est un tableau générique. Son constructeur prend en paramètre le chemin vers le fichier contenant le dictionnaire, et appelle la méthode *getDico()* de la classe `LecteurDico` pour ensuite créer le tableau générique. On vous a déjà donné la méthode `boolean estCorrect(String mot)` qui renvoie vrai si l'appel de la méthode "recherche" de la classe "TableauGenerique" ne renvoie pas -1 (i.e. l'élément est bien dans le tableau). On vous demande de compléter les autres méthodes de cette classe. Lisez tout le code pour comprendre son organisaton.
2. **Nombre de mots corrects**. Ecrire la méthode `private int compteCorrect(String[] mots)` qui prend un tableaux de mots anglais et compte combien de mots sont corrects dans ce texte.

Testez avec une phrase suffisamment longue, comme 'Yes we cann! Do itt. For sure my deaar. I live in a sweet home. Thi is viry long inded.", et comparer l'efficacité de la recherche simple et de la recherche dichotomique.

Pour mesurer les temps, on a utilisé le code suivant qui met la date actuelle en millisecondes dans t (i.e. le nombre de millisecondes écoulées depuis le January 1, 1970).

```
long t = System.currentTimeMillis();
```

La différence entre le temps après le décompte des mots et le temps avant le décompte des mots vous donne le temps écoulé.

3. **Correction orthographique**. Les fautes d'orthographe les plus fréquentes sont:

   - oublier une lettre (exemple: Thi),
   - mettre une lettre en trop (exemple: deaar),
   - mettre une lettre à la place d'une autre (exemple: viry),
   - permuter des lettres (exemple: hoem).

   Complétez les méthodes prévues pour chacun de ces cas, qui s'applique quand le mot est incorrect et qui renvoie une liste de corrections possibles. Pour cela, il vous faut prendre le mot, et pour chaque position dans le mot, essayer d'ajouter une lettre à la position donnée, ou bien l'enlever, ... Et pour chaque tentative de correction, on vérifie si le mot créé est correct en le cherchant dans le dictionnaire. Si c'est le cas, on l'ajoute à la liste des corrections possibles.

   Des exemples vous sont donnés dans la classe `TestCorrecteur`, vous pouvez en ajouter d'autres. Et il est là aussi très intéressant de comparer le temps total pour la correction orthographique dans le cas où la methode "estCorrect" utilise la recherche simple ou la recherche dichotomique. Exemple obtenu pour la phrase "Yes we cann! Do itt. For sure my deaar. I live in a sweet home. Thi is viry long inded." : 2112ms avec la recherche simple et 9ms avec la recherche dichotomique. C'est 234 fois plus rapide, imaginez pour corriger un texte de 500 lignes ...