Compteur de mots (partie 1)

L'objectif de ce travail pratique est d'implémenter un programme permettant de fournir un certain nombre de statistiques à partir d'un texte donné en paramètre, tel que son nombre de mots, le nombre d'occurrence(s) des mots les plus fréquents, etc.

Ce programme sera réalisé au cours du semestre en utilisant différentes structures de données, dans le but de comparer leur efficacité.

1 Création de la « classe mère »

Les différentes versions du programme que vous allez réaliser aurons une partie de leur code en commun. Dans le but d'éviter la duplication de code, nous allons utiliser une *classe abstraite* qui factorisera l'ensemble du code utile pour vos différentes classes.

Dans un premier temps, créer cette classe abstraite :

```
public abstract class Compteur {
    private String nomFichier;
    private int nbMots;
    private int nbMots5;
    public abstract void addOccurrence(String mot);
    public Compteur (String fichierTexte);
}
```

La méthode `addOccurrence(String)` sera implémentée plus tard, suivant les structures de données sous-jacentes considérées dans les différentes classes que vous écrirez au long du semestre.

Il est en revanche possible dès maintenant d'implémenter le constructeur de cette classe abstraite, qui prend un nom de fichier (`fichierTexte`) en paramètre, qui en extrait chacun des mots, les traite, et qui les ajoute à la structure de données via un appel à `addOccurrence(String)` (même si cette méthode est *abstraite*).

Les mots extraits doivent être traités avant l'appel à `addOccurrence`. Vous veillerez à respecter les critères suivants :

1. Votre programme doit ignorer la casse. Les mots radio, Radio, et rAdiO seront considérés comme un seul et même mot [1].
2. Afin d'éviter de « bruit » dû aux pronoms, articles (in)déterminés, possessifs, etc., nous ne considérerons que les mots contenant strictement plus de 4 caractères (donc au moins 5). Les autres ne seront pas considérés.
3. La ponctuation du texte doit être ôtée de chaque mot extrait. En effet, dans la typologie française, il n'y a pas d'espace avant les signes de ponctuation « simples » (point, virgule, parenthèse, etc.). N'étant pas séparés par un espace, ils sont par defaut considérés comme faisant parti des *tokens* (mots) extraits. Votre programme doit donc les ôter avant de les insérer dans la structure de données.

Voici un morceau de code permettant de lire, ligne par ligne, un fichier `nomFichier` donné en paramètre.

```
Scanner scanner=new Scanner(new File(nomFichier));
 // On boucle sur chaque ligne
while (scanner.hasNextLine()) {
    String line = scanner.nextLine();
    //faites ici votre traitement
}
scanner.close();
```

Utilisez ce morceau de code dans votre constructeur, et adaptez-le de manière à (i) gérer les exceptions dues à l'ouverture/la lecture d'une fichier (ii) traiter chaque ligne lue pour en extraire les différents mots [2], les traiter (filtrer les mots de taille inférieure à 5, ôter ponctuation, etc.) avant de les donner un-à-un en paramètre de `addOccurrence`.

2 Un premier test

Avant d'implémenter une version 100% fonctionnelle de notre programme, nous allons créer une première classe *concrète*, fille de `Compteur`, qui affichera l'ensemble des mots trouvés au sein du texte, et que vous aurez préalablement traités.

1. il vous est conseillé de convertir en minuscule chaque mot lu, avec la méthode `toLowerCase()` de la classe `String`
2. utiliser la méthode `split(String)` de `String`

Pour cela, il vous suffit de créer une classe fille à `Compteur` où `addOccurrence(String)` se contente d'afficher l'objet qui lui est donné en paramètre, de la façon suivante :

```
public class CompteurTest extends Compteur {
    public void addOccurrence(String mot){
        System.out.println(mot);
    }
    public static void main(String[] args){
        if (args.length<1) System.err.println("nom de fichier manquant");
        else CompteurTest c = new CompteurTest(args[0]);
    }
}
```

Voici le début de la sortie que doit produire votre programme avec le fichier `test.txt` disponible sur `Moodle`.

```
important
tester
régulièrement
programme
cours
développement
lorsque
celui
comportera
beaucoup
difficile
...
```