Programmation Objet en Java - Feuille de TP 9

Exercice 1. On écrira le code de cet exercice dans la classe `EnhancedCopy`.

a) Écrire une méthode `copyWithLineNumbers` qui prend en paramètres les chemins `scr` et `dst` de deux fichiers et permet de recopier le premier fichier dans le deuxième en rajoutant les numéros des lignes (suivis d'une tabulation). Attention, on veut que la méthode fonctionne même sur de **très gros** fichiers.

b) Tester la méthode. Que se passe-t-il si le premier fichier n'existe pas ?

c) Modifier votre code pour que le texte soit recopié à la suite de ce qui se trouve dans le deuxième fichier (essayez en copiant deux fois dans le même fichier).

d) Que se passe-t-il si le deuxième fichier n'existe pas ? Modifies votre code pour faire en sorte qu'il soit créé.

e) Modifier le programme pour que tous les mots du texte soient recopiés en majuscules.

f) (optionnel) Sur le même modèle, écrire deux méthodes permettant d'encrypter puis décrypter un texte en décalant les lettres (mais pas la ponctuation et les espaces) d'une position dans l'alphabet. On pourra supposer que le texte ne contient pas de caractères accentués.

Exercice 2. On souhaite compter les occurrences des mots dans un texte.

a) Écrire un classe `Histogram` qui permet de fabriquer et d'afficher un histogramme de mots, c'est-à dire un objet qui associe à chaque mot sont nombre d'occurrences. L'affichage doit être en ordre alphabétique et également indiquer les nombre de mots (pas forcément distincts) au total.

b) Dans une classe de test, écrire méthode statique `histogramFromFile` qui, étant donné le chemin d'un fichier encodé en ISO-LATIN-1, extraie l'histogramme des mots qu'il contient. Testez-la sur les fichiers fournis.

Remarque : on a besoin de distinguer les caractères qui composent les mots des autres. Pour cela, on utilise la méthode `split` *de* `String` *et l'expression régulière* `"[^\\pIsLatin]+` *qui représente n'importe quelle séquence caractères qui ne sont pas des lettres.*

c) Écrire deux méthodes `saveHistogramBackUp` et `makeHistogramFormBackUp` qui prennent des `Path` en paramètre et permettent de sauvegarder l'histogramme et le récupérer sans utiliser la sérialisation.

d) Sauvegarder l'histogramme et le récupérer en utilisant la sérialisation.

Exercice 3.

a) On considère un fichier qui contient une matrice de mots : chaque ligne de la matrice est écrite sur une ligne du fichier ; sur une ligne, les mots sont séparé par des espaces. Écrire une méthode `matrixFromFile` qui prend en paramètre le chemin d'un fichier et renvoie la matrice (tableau à deux dimensions) lue à partir du fichier. Si le fichier est vide, la méthode renvoie `null` et si la matrice est mal formée (lignes de taille différentes), elle lance une exception. Attention, la méthode ne doit lire le fichier qu'une seule fois.

b) Écrire une méthode `mapFromMatrix` qui prend en paramètre une matrice de mots et renvoie une `Map` qui, à chaque mot présent dans la matrice, associe la liste des cases de la matrices dans lesquelles il se trouve. Une case de la matrice est représentée par un couple d'indices. Si la matrice est vide, la méthode renvoie `null`.