Masters Langue et Informatique
Université Paris-Sorbonne

MASTER LANGUE ET INFORMATIQUE

Travaux Pratiques n° 10
Manipulation de fichiers
et lecture robuste de textes

Les exercices qui vous sont proposés ici ont pour but de comprendre la notion d'exception, d'apprendre à utiliser des fichiers binaires et textes.

1. MON PREMIER FICHIER TEXTE EN JAVA

Exercice 1 : Créer la classe exécutable MonPremierFichierTexte qui écrit dans le fichier texte « Nom.txt » les noms des membres du groupe.

Exercice 2 : Ajouter à la classe MonPremierFichierTexte la méthode Verifier qui vérifie si le fichier « Nom.txt » contient bien .les noms des membres du groupe.

2. MANIPULATIONS DE FICHIERS

Le but de cette partie est de manipuler les différents types de flots et de fichiers en se référençant aux exemples du cours et à la javadoc.

Exercice 1 : Créer la classe exécutable Copie qui permet de copier le contenu d'un fichier dans un autre.

Exercice 2 : Créer la classe exécutable Compresse qui permet de compresser le contenu d'un fichier.

Exercice 3 : Créer exécutable la classe Décompresse qui permet de décompresser le contenu d'un fichier.

Exercice 4 : Créer la classe exécutable Propriétés qui permet d'afficher toutes les propriétés d'un fichier.

3. LECTURE ROBUSTE D'UN TEXTE

Le but de cette partie est de construire une classe robuste de lecture de fichiers texte appelée LireTexte. Cette classe a le squelette suivant :

```
package tp5;

import java.io.*;
import java.util.*;

/** Lecture des mots d'un fichier-texte
 * @author montacie
 */
public class LireTexte {
    private String NomFichier;
    private BufferedReader entrée;
    private String ligne;
    private StringTokenizer tok;

    /** Création d'une nouvelle instance de LireTexte
     * @param ft nom du fichier texte
     */
    public LireTexte(String ft) {}

    /** Ouverture d'un fichier texte,
     * Lecture de la première ligne,
     * Teste si le fichier est vide
     * @throws IOException Erreur à l'ouverture
     * @throws FichierVide Fichier vide
     */
    protected void Ouvrir() throws IOException, FichierVide {}

    /** Fermeture du fichier
     * @throws IOException Erreur à la fermeture
     */
    protected void Fermer() throws IOException {}

    /** Lecture mot à mot du fichier
     * @return prochain mot
     * @throws IOException Erreur de lecture
     */
    public String readWord () throws IOException {}
}
```

Les caractéristiques de cette classe sont les suivantes :

Le constructeur a comme argument un nom de fichier.

L'interface publique est composée uniquement de la méthode readWord() retournant un mot à chaque appel et null en fin de fichier.

La méthode Ouvrir() déclenche une exception de type FichierVide (à définir) quand elle s'aperçoit que le fichier existe mais qu'il est vide.

Vous devrez utiliser les méthodes readLine(), hasMoreTokens(), nextToken() de la classe StringTokenizer.

Exercice 1 : Créer la classe FichierVide dérivée de la classe Exception.

Exercice 2 : Implémenter les méthodes de la classe LireTexte et tester cette classe sur un des textes du répertoire livre.

DEVOIR SPECIALISATION DE FLOTS

Le but de cette partie est d'enrichir les différentes classes de flots en créant la classe non exécutable FileInputStreamTP7 héritant de FileInputStream.

Exercice 1 : Ajouter la méthode Copie ayant comme argument une variable de type File dans laquelle s'effectuera la copie. Ecrire une classe de test.

Exercice 2 : Ajouter la méthode Compresse ayant comme argument une variable de type File dans laquelle s'effectuera la copie avec compression. Ecrire une classe de test.

Exercice 3 : Ajouter la méthode Propriétés permettant d'afficher toutes ses propriétés. Ecrire une classe de test.

Exercice 4 : Ajouter la méthode CompareTo ayant comme argument une variable de type FileInputStreamTP7 avec laquelle s'effectuera une comparaison (relation d'ordre lexicographique). Ecrire une classe de test.