ESIAL 1A – Structure de Données
Manipulation de flots de données

Vous pourrez vous aider des slides fournies dans le répertoire.

Exercice 1 : Générateur de fichier de données

L'objectif de cet exercice est d'écrire la classe flots.GenerateurDonnees permettant de générer un fichier d'entiers. Les deux paramètres de la ligne de commande désignent respectivement le nom du fichier à créer et le type de fichier (trié, construit au hasard ou trié à l'envers).

Exemples d'appel :

```
java flots.GenerateurDonnees donnees1 Trie
java flots.GenerateurDonnees donnees2 Envers
java flots.GenerateurDonnees donnees1 Hasard
```

Le nombre de données générées est pris au hasard, ainsi que les données proprement dites (la classe java.util.Random fournit un générateur de nombres aléatoires).

Exercice 2 : Commande grep

La commande grep sous Unix permet de détecter la présence d'une chaîne dans un fichier. Elle affiche le texte et le numéro des lignes du fichier qui contiennent la chaîne cherchée.

L'objectif de cet exercice est d'écrire la classe flots.Grep qui simule une partie du fonctionnement la commande grep : la chaîne recherchée doit être intégralement présente. On fait ici abstraction des expressions régulières acceptées par grep. Les deux paramètres de la ligne de commande désignent respectivement la chaîne recherchée et le fichier à consulter. Le résultat est affiché sur la sortie standard.

Exemple d'appel :

```
java flots.Grep Madame donnees.txt
Ligne 1 : Madame E. Paterot 03 83 27 65 11
Ligne 6 : Madame P. Bilas 03 29 12 78 30
```

Exercice 3 : Commande sort

La commande sort sous Unix permet de trier ou d'interclasser des fichiers de données.

L'objectif de cet exercice est d'écrire la classe flots.Sort qui simule en partie le fonctionnement de la commande sort, pour interclasser deux fichiers triés, ligne par ligne. Le résultat est rangé dans un troisième fichier. Les noms des trois fichiers sont donnés en paramètre.

Exemple d'appel :

```
java flots.Sort fichTrie1 fichTrie2 fichRes
```

Exercice 4 : Analyse d'un fichier de données

Ecrire une application flots.NomsDate qui analyse un fichier de données, constitué d'un nombre quelconque de lignes, chacune d'elles contenant un nom, une année de naissance et éventuellement un commentaire encadré par /* et */. L'exécution affiche sur la sortie standard les noms et années de naissance du plus jeune et du plus âgé. Si le fichier est mal construit, l'application s'arrête en erreur.

Exemple d'appel :

```
java flots.NomsDate donnees.txt
Hermoli est le plus agé, né en 1906
Moissac est le plus jeune, né en 1994
```

Indication : Utiliser un analyseur lexical StreamTokenizer.

Exercice 5 : Analyse d'un texte de classe

Construire une classe flots.AnalyseurClasse utilisée pour analyser le texte d'une classe Java :

– Le constructeur admet en paramètre le nom du fichier contenant la classe à analyser.
– La méthode public String identificationPackage() permet de consulter le nom du package contenant la classe.
– La méthode public String [ ] identificationInterfaces() permet de consulter les noms des interfaces implantées par la classe.
– La méthode public boolean estAbstraite() permet de savoir si la classe est abstraite.
– La méthode public int nbreConstructeurs() permet de consulter le nombre de constructeurs de la classe.
– La méthode public int nbreClasses() permet de consulter le nombre de classes déclarées dans le fichier.

Ecrire aussi une classe de test de flots.AnalyseurClasse, que vous pouvez exécuter avec le fichier AnalyseurClasse.java par exemple.