Utilisation de fichiers

L'utilisation de fichiers en Java est un sujet complexe si on rentre dans le détail des différents types de fichiers et de leur utilisation ce que nous ne ferons pas. Notre but dans ce document est de donner les premiers rudiment qui suffisent pour lire et écrire des fichiers séquentiels.

Fichier texte

Un fichier texte est une suite de caractères. Dans cette suite, il y a des caractères spéciaux pour marquer les fins de lignes. Sous unix, c'est le caractère `'\n'` qui note la fin de ligne et sous windows, c'est la séquence de deux caractère `'\r', '\n'`. Ces deux caractères sont des caractères spéciaux, que l'on peut taper au clavier avec la touche `entree`.

Un fichier séquentiel est un fichier dans lequel les informations sont mise les unes à la suite des autres. Nous voudrions disposer de fichiers textes, dans lesquels on écrit des caractère, mais ce n'est pas très facile parce que Java utilise des caractères codés sur deux octets (unicode), alors que nos systèmes d'exploitation et les fichiers PPM utilisent le système de codage sur un octet appelé ASCII.

Faute de pouvoir stocker directement des caractères, nous allons stocker des octets (type `byte` en Java).

Lecture de fichier

Pour manipuler des fichiers, il faut utiliser une librairie prédéfinie appelée `java.io`. Pour l'utiliser, il faut commencer le fichier source par la ligne :

```
import java.io.*;
```

Ensuite, il faut ouvrir le fichier en créant un objet qui le représentera. Cet objet sera conservé dans une variable tout à fait ordinaire. Si on veut lire un fichier, il faut créer un objet instance de `FileInputStream`. On donne le nom du fichier en paramètre.

Pour lire un octet, on utilise la méthode `read()` de l'objet créé. Cette méthode renvoie un octet lu dans le fichier ou -1 si on est en fin du fichier.

Lorsqu'on n'utilise plus un fichier, il faut le fermer avec la méthode close.

```
import java.io.*;
class LireFichier{
    public static void main(String[] args){
        String nomFichier;
        FileInputStream fichier;
        int c;
        Terminal.ecrireString("Entrez le nom du fichier à afficher : ");
```

```
        nomFichier = Terminal.lireString();
        try{
            fichier = new FileInputStream(nomFichier);
            c = fichier.read();
            while (c != -1){
                Terminal.ecrireChar((char) c);
                c = fichier.read();
            }
            fichier.close();
        }catch(FileNotFoundException ex){
            Terminal.ecrireStringln("Ce fichier n'existe pas");
        }catch(IOException exc){
            Terminal.ecrireStringln("Erreur d'´eentre-sortie");
        }
    }
}
```

Remarquez que dans cet exemple, on lit un octet que l'on met dans une variable `c` de type `int`, puis on la convertit en `char` (conversion : `(char) c`). On est obligé de procéder comme cela.

Création d'un fichier

Si on veut écrire un nouveau fichier (ou réécrire un fichier en supprimant la version précédente), il faut créer un objet de la classe `FileOutputStream`, utiliser la méthode `write(x)` qui écrit un octet `x`. Ici encore, on ferme le fichier lorsqu'on a fini de l'écrire.

```
import java.io.*;
class EcrireFichier{
    public static void main(String[] args){
        String nomFichier;
        FileOutputStream fichier;
        String aEcrire;
        Terminal.ecrireString("Entrez le nom du fichier `a ´ecrer: ");
        nomFichier = Terminal.lireString();
        Terminal.ecrireStringln("Entrez des lignes `a enregistrer.");
        Terminal.ecrireStringln("Tapez FIN lorsque vous avez fini.");
        try{
            fichier = new FileOutputStream(nomFichier);
            aEcrire = Terminal.lireString();
            while(!aEcrire.equals("FIN")){
                for (int i = 0; i<aEcrire.length();i++){
                    fichier.write(aEcrire.charAt(i));
                }
                fichier.write('\r'); fichier.write('\n');
                aEcrire = Terminal.lireString();
            }
            fichier.close();
        }catch(IOException exc){
            Terminal.ecrireStringln("Erreur d'´eentre-sortie");
        }
    }
}
```

Traitement d’erreurs

L’utilisation de fichier peut lever des exceptions prédéfinies, notamment `FileNotFoundException` qui signifie que le fichier qu’on voulait ouvrir n’existe pas et `IOException` qui signifie qu’un problème d’entrées-sorties s’est produit (par exemple si l’on veut écrire alors que le disque est plein).