TD1 Entrées/sorties

Développement Logiciel (L2-S4)

Vendredi 8 février 2013

Exercice 1 Entrées/sorties clavier.
Le but de cet exercice est de calculer la moyenne de notes données au clavier. Le système demande des notes jusqu'à ce que le mot clé "fini" soit donné. A ce moment là le système s'arrête et affiche la moyenne. Il faudra faire attention à ce que chaque note soit comprise entre 0 et 20 et soit bien numérique. Pour ceci nous allons utiliser `System.in` qui est un `InputStream`. Il est alors possible d'utiliser la classe `Scanner` permettant la lecture de l'entrée avec `Scanner scanner = new Scanner(System.in)`. La lecture du clavier se fait alors avec `String line = scanner.nextLine()`.

Exercice 2 File.
Le but de cet exercice est d'utiliser la classe `File`. Cette classe permet de traiter des fichiers et des répertoires. Elle contient notamment les méthodes `boolean isFile()`, `boolean isDirectory()` et `boolean exists()` qui indiquent respectivement si l'instance correspond à un fichier, un dossier ou un élément existant dans le système de fichiers. Quand l'instance de `File` correspond à un répertoire, il est possible de lister les fichiers qu'il contient avec `String[] list()`.

a. Créez une méthode `void testFile (String path)` (*path* signifie *chemin* en français) permettant de tester si le fichier existe, si c'est un fichier ou un répertoire.

b. Si le fichier n'existe pas, alors on le crée en utilisant la méthode `File.createNewFile()`.

c. Modifiez votre fonction pour les cas où il s'agit d'un répertoire et dans ce cas énumérer l'ensemble des fichiers s'y trouvant en donnant leur chemin absolu avec la méthode `File.getAbsolutePath()`.

d. On souhaite à présent afficher que les fichiers dont l'extension est `.java`. Nous allons pour cela utiliser la fonction `String[] list(FilenameFilter filter)` qui permet de filtrer les listes de fichiers comment on peut le faire avec `ls`. La Javadoc nous apprend que l'interface `FilenameFilter` décrit une méthode `boolean accept(File dir, String name)` qui doit retourner `true` (donc *vrai*) si le fichier doit être ajouté à la liste et `false` sinon.

Exercice 3 Copie de fichier.
Le but de cet exercice est de copier le contenu d'un fichier dans un autre fichier. Pour cela nous allons créer la fonction `void copie(String filenameFrom, String filenameTo)`.

a. Dans un premier temps nous allons écrire le contenu du fichier au clavier. Pour cela nous pouvons utiliser un `BufferedReader` qui permettra de mettre le texte dans une mémoire tampon et un `FileReader` qui ouvre le texte en lecture. Pour créer le `BufferedReader` nous pouvons faire `BufferedReader buffer = new BufferedReader(new FileReader(filename))`. Pour lire une ligne du fichier, il suffit d'exécuter `buffer.readLine()`.

b. Ensuite, modifiez votre code pour écrire les lignes dans le fichier `filenameTo`. Pour cela, vous aurez besoin de `BufferedWriter` et `FileWriter`, qui s'utilisent comme ceci : `BufferedWriter buffer = new BufferedWriter(new FileWriter(filename))` en appelant `buffer.write(String s)` pour écrire dedans.

c. Modifiez votre code pour que le fichier de sortie ne contienne pas les commentaires *Java*, c'est-à-dire les lignes commençant par `//` et les expressions entre `/*` et `*/`.

Exercice 4 Classe `Student`.
Le but de cet exercice est de de créer la classe `Student` (*Etudiant* en français). Un étudiant peut se représenter par un *nom*, un *prénom*, un *mot de passe* et un *ensemble de notes*.

a. Créez la classe `Student` avec son constructeur et sa méthode `String toString()` qui affiche son nom et son prénom.

b. On veut pouvoir connaître la moyenne d'un étudiant. Pour cela, nous allons parcourir l'ensemble des notes avec l'aide d'un itérateur.

c. On veut aussi pouvoir sauvegarder l'ensemble des éléments sauf le mot de passe qui doit rester confidentiel (il sera récupéré par un autre moyen que nous verrons peut-être plus tard dans le semestre).

d. Créez la méthode `static void main(String[] args)` qui devra :
   - demander le nom et le prénom de l'étudiant,
   - demander la liste de ses notes (réutilisez ce que vous avez fait dans l'exercice 1),
   - afficher l'instance de l'étudiant et sa moyenne,
   - sauvegarder cette instance dans un fichier ayant pour nom : `<nom de l'étudiant>.student`.