TP CSID – 11 sept 2012

Question 1

Créer un projet Java sous eclipse “tp-java-io”, et utiliser pour la suite le nom de package “fr.iut.tp.io”

Créer l'interface FileHandler, qui permet d'appeler N fois un callback en prenant à chaque fois un fichier en argument:

```
package fr.iut.tp.io;
import java.io.File;
public interface FileHandler {
    public void handleFile(File file);
}
```

Ecrire la classe “FileScanner“ qui fait un parcours récursif de fichiers, en appelant “fileHandler.handleFile(file)” pour chaque élément parcouru.

```
package fr.iut.tp.io;
import java.io.File;
public class FileScanner {
    public void handleRecursively(File dir, FileHandler handler) {
       // TODO … call handler.handleFile(...) for each sub-dir/file
in dir
    }
}
```

Question 2

Ecrire plusieurs sous-classes d'implémentations de FileHandler pour:

2 a)

PrintFileHandler = pour imprimer à l'écran les noms des fichiers

2 b)

ListFileHandler = pour remplir une liste de fichiers, et les retourner en fin de parcours. Un exemple d'utilisation pourra être:

```
public static void main(String[] args) {
   File dir = (args.length > 0)? new File(args[0]) : new File(“.”);
   ListFileHandler lh = new ListFileHandler();
   new FileScanner(). handleRecursively(dir, lh);
   List<File> files = lh.getResultList();
   System.out.println(“found files:” + files);
}
```

2 c)

MapByFullNameFileHandler = idem 2c) mais maintient une Map<String,List<File>> au cours du parcours, avec comme clef le shortName des fichiers

Un exemple typique de mise a jour d'une “Map<Key,List<Value>>” est le suivant:

```
   public void registerInList(Map<Key,List<Value>> map, Key key,
Value value) {
    List<Value> ls = map.get(key);
    if (ls == null) {
       ls = new ArrayList<Value>();
       map.put(key, ls);
    }
    ls.add(value);
  }
```

A la fin du parcours du répertoire, afficher tous les noms de fichiers correspondants à des fichiers en doublons, avec les chemins correspondants.

2 d)

TreeBySizeFileHandler = idem 2c) mais maintient une TreeMap<Long,List<File>> au cours du parcours, avec comme clef la taille des fichiers.

A la fin du parcours, afficher les 10 premiers noms de fichiers contenant les fichiers de plus grosses tailles, avec les tailles correspondantes.

Question 3

Pour un fichier donné, calculer son “SHA-1”, et l'afficher en hexadecimal

L'exemple a été vu dans le cours, on pourra le reprendre. Voici 3 étapes intermédiaires permettant de le

recoder de zéro:

a) écrire un premier bout de code ouvrant le fichier comme un InputStream (new BufferedInputStream(new FileInputStream(file)), et parcourir le contenu du fichier. N'oublier pas de wrapper le code correctement dans un try-catch-finally pour bien refermer le inputStream ouvert.

b) tous les octets lus du fichier doivent être envoyés dans la method “update()” d'un objet de type “MessageDigest”, obtenu par “MessageDigest md = MessageDigest.getInstance(“SHA1”)” et le resultat final est obtenu par “md.digest()”

c) le digest est du type “byte[]”, le convertir en “String”
pour cela, on peut par exemple utiliser dans la librairie commons-codec : “Hex.toHexa()”)... ou le recoder à la main:

```
private static final char[] DIGITS_LOWER = {'0', '1', '2', '3', '4',
'5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
public static String toHex(byte[]] data) {
        int l = data.length;
        char[] out = new char[l << 1];
        for (int i = 0, j = 0; i < l; i++) {
         out[j++] = DIGITS_LOWER[(0xF0 & data[i]) >>> 4];
         out[j++] = DIGITS_LOWER[0x0F & data[i]];
        }
        return new String(out);
}
```

d) Modifier le code précédent, pour préfixer les octets lus du fichier par la chaine de charactere “blob “ + file.length()
Cela permet d'avoir le même résultat que la command “git hash-object <<file>>” . Vérifier le résultat du code sur un fichier d'exemple.

Question 4

Refaire la question 2 c) mais en remplaçant le nom du fichier par le SHA1 de son contenu: MapBySHA1FileHandler =maintient une HashMap<String,List<File>> au cours du parcours, avec comme clef le SHA1 du fichier obtenu en question 3.

Afficher à la fin du parcours d'un répertoire tous les noms de fichiers dont le contenu sont des doublons.