TP Utilisation des arbres comme hiérarchie de fichiers

```
import java.io.File;

public class Folder{
    /* Construire l'arbre des dossiers à partir de l'adresse indiquée */
    public static Tree<String> buildTree(File dir){

        File[] fileTab = dir.listFiles();

        Tree<String> arbre = new Tree<>(dir.getName());

        for (File file : fileTab){
            if(file.isDirectory()){
                arbre.addChildren(buildTree(file));
            }
        }

        return arbre;
    }
    /* Un algorithme qui modifie l'arbre parmi ceux proposés.
    * Vous pouvez changer le nom en fonction */
    public static void delete (Tree<String> hierarchy, String name, String
path){

        //System.out.println(hierarchy.data()+" - " + name);
        //System.out.println(hierarchy.data().compareTo(name));
        if(hierarchy.data().compareTo(name) == 0){
            String filename;
            if (path == "") {
                System.out.println(hierarchy.data());
                filename = hierarchy.data();
            }else{
                System.out.println(hierarchy.data());
                filename = path + hierarchy.data();
            }
            File f = new File(filename);
            System.out.println(f.isDirectory());
            System.out.println(f.isFile());

            File[] filelist = f.listFiles();
            int i = 0;
            for(File d: filelist){
                System.out.println(d.getName());
                if(!d.delete()){
                    delete(hierarchy.child(i), hierarchy.child(i).data(),
filename+"/");
                }
                i++;
            }
```

```
            //for (int i = 0; i<hierarchy.nbChildren(); i++){
            //    delete(hierarchy.child(i), hierarchy.child(i).data(),
filename+"/");
            //}

            if(f.delete()){
                System.out.println(filename + "OK");
            }else{
                System.out.println(filename + "ERREUR");
            }
        }else{
            for(int i = 0; i<hierarchy.nbChildren(); i++){
                delete(hierarchy.child(i), name, path + hierarchy.data() +
"/");
            }
        }

        return;
    }

    /* Créer les répertoires à partir de l'arbre */
    public static void createFolders(File dir, Tree<String> hierarchy){
        //hierarchy.display();
        /* Pour créer un nouveau répertoire, créer un nouvel objet File
        * puis utiliser file.mkdirs (ou mkdir)
        * A noter : on peut contruire un fichier dans un repertoire
existant
        * avec new File(File rep, String name)
        */
        if (!dir.isDirectory()){
            dir.mkdir();
        }

        for (int i = 0; i<hierarchy.nbChildren(); i++){
            if(hierarchy.child(i) != null){
                createFolders(new
File((dir+"/"+hierarchy.child(i).data())), hierarchy.child(i));
            }
        }
        return;
    }


    public static void main(String... args){
        File currentDir = new File("TP3/test");

        /* Un objet File de java. Le point de départ est le repertoire
contenant le fichier source.
        * Pour avoir son nom, dir.getName().
        * On teste que c'est un dossier avec f.isDirectory().
        */
        //System.out.println(currentDir.getAbsolute());
```

```
        Tree<String> hierarchy = buildTree(currentDir);
        hierarchy.display();
        //update(hierarchy, "name");
        //hierarchy.display();
        currentDir = new File("racine");
        //currentDir.mkdir();
        createFolders(currentDir, hierarchy);
        hierarchy = buildTree(currentDir);
        hierarchy.display();

        delete(hierarchy, "name", "");
        /*File test = new File("TP3/unautretest");
        //test.mkdir();
        File[] fichierdetest = test.listFiles();
        for(File f: fichierdetest){
            f.delete();
        }
        test.delete();*/

    }
}
```