Rapport :
Projet JAVA

Sujet :

Trouver les doublons de fichiers dans un ou deux répertoires.

WOLFGER Killian
ESCALES Loïc

Explication :

Au lancement de l'application c'est la classe Application qui ouvre la fenêtre.

C'est la classe FindDuplicateFile qui s'occupe de la recherche de fichier en plusieurs exemplaires. Elle a deux méthodes : la première prend un seul dossier en paramètre et compare la totalité des fichiers entre eux ; la deuxième prend deux dossiers en paramètre et compare les fichiers de l'un par rapport à l'autre. Les fichiers identiques sont stockés dans une liste de fichier. Il y a donc au minimum deux fichiers dans cette liste. Le tout est stocké dans une liste ce qui revient à une liste d'une liste de fichier.

Les deux méthodes ont d'abords besoin d'indexer la totalité des fichiers contenus dans les répertoires. Pour ce faire on va utiliser Files.walkFileTree qui permet de naviguer dans la totalité de l'arborescence et d'appeler les méthodes d'une classe « visiteur » passée en paramètre. En l'occurence c'est IndexVisitor qui est passée en paramètre. Pour chaque fichier qui est rencontré on l'ajoute dans une table de hachage dont la clé est la taille du fichier et la valeur une liste de fichiers. On utilise une liste de fichiers car plusieurs fichiers ont la même taille.

En réalité ce n'est pas directement le fichier qui est stocké mais un FileComparator. Cette classe permet de comparer deux fichiers en comparant les empreintes obtenues par un algorithme de hachage. Nous avons choisi MD5 pour sa rapidité et la longueur de son empreinte bien qu'il ne soit pas infaillible. L'empreinte est mémorisée pour éviter de la recalculer. Le calcul de l'empreinte se fait à l'aide d'Apache Commons Codec qui simplifie l'utilisation des algorithmes de hachage.

C'est là que les deux méthodes divergent.

Dans le cas de l'analyse d'un seul dossier il suffit de comparer la liste des fichiers indexés à elle-même pour ressortir les fichiers dupliqués. On fait attention à ne pas comparer le fichier à lui-même. Comme les fichiers de même taille sont stockés dans la même liste on a juste à comparer cette liste à elle-même, ce qui est beaucoup plus rapide que s'il fallait comparer tout l'index avec lui-même.

Dans le cas de l'analyse de deux dossiers on va réutiliser Files.walkFileTree mais avec un autre « visiteur » : CompareFileVisitor. Il compare chaque fichier trouvé avec la liste des fichiers indexée. La aussi on a juste à comparer avec la liste des fichiers de même taille.

Pour constituer la liste de fichiers identiques on utilise une table de hachage avec

comme clé l'empreinte du fichier et comme valeur une liste de fichier. Chaque dois que deux fichiers sont identiques on les ajoute en faisant attention que l'un deux n'a pas déjà été ajouté. Une fois tous les fichiers comparés on a juste à récupérer toutes les valeurs de la table de hachage pour les mettre dans une liste.

FindDuplicateFiles a deux attributs : minFileLength pour ne comparer que les fichiers qui font au moins la taille spécifiée en octet. Et regexFileName pour ne comparer que les fichiers qui lui corresponde. Ces attributs sont aussi présents dans les classes « visiteurs » comme ce sont elles qui s'occupent de recenser les fichiers.

Utilisation :

Il faut veiller à ajouter la bibliothèque Apache.
Une fois l'interface lancé, il faut sélectionner 1 ou 2 dossiers puis lancé la comparaison en utilisant les boutons relatifs à ces fonctions. Après calcul, les doublons sont affichés dans la zone de texte en bas de la fenêtre.

Conclusion :

→ L'utilisation de threads et de calcul parallèle permettrait de réduire le temps de calcul en partageant les tâches. Le problème en calcul séquentiel c'est que les calculs vont figer l'interface pendant qu'ils s'exécutent. Pour éviter cela il faut paralléliser et séparer le thread gui et les threads de calcul.

En ce qui concerne l'interface, nous aurions aimé ajouté des fonctionnalités utilisateurs, et avoir un design plus propre et des boutons mieux agencés.
Améliorations :

1. -Supprimer le textArea et utiliser des widgets pour un meilleur affichage
2. -Réorganiser les boutons et rendre le tout ergonomique
3. -Mettre des options pour supprimer les doublons
4. -Permettre l'ouverture des fichiers depuis l'appli
5. -Pouvoir arrêter l'opération si elle prend trop de temps