Rapport de projet Java

NEONAKIS Ionas ROUSSET Maxime SAKAR Nurullah

Gestionnaire De Fichiers

Etape 1 :

Pour la création d'arbre fichiers nous avons créé une Class abstraite AbstractArbreFichiers contenant tous les attributs de cet arbre. Nous avons ensuite créé deux classes, ArbreFichierDossier et ArbreFichierFichier, qui héritent de la classe abstraite et qui simulent respectivement le comportement de dossiers et de fichiers. Dans ces classes nous avons implémenté la méthode décrite dans le sujet ainsi que certaines méthodes supplémentaires pour utiliser les commandes dans la partie 3. Dans ArbreFichierDossier on retrouve les méthodes :

ajouterFils(AbstractArbreFichier a) : qui ajoute un fils au dossier courant (mkfile, mkdir)

supprimerFils(AbstractArbreFichier a) : qui supprime un fils au dossier courant (rm)

infoNoeud() : donne les fils du dossier (ls)

cheminAbsolu() : donne le chemin jusqu'à la racine (pwd)

seDirigerVers(String s) : retourne un fils du dossier ou son père (cd)

peutSeDirigerVers(String s) : vérifie que le string est un fils du dossier

trouverUnFichier(AbstractArbreFichier a, String f), trouve le chemin vers un fichier (Find avec paramètre)

trouver(AbstractArbreFichier a) : méthode qui retourne un string de tous les fichiers et dossiers du dossier

(Find sans paramètre)

getString(AbstractArbreFichier a) : retourne le chemin du fichier ou dossier jusqu'au dossier en paramètre.

ArbreFichierFichier contient :

lignesMatch(String pattern) : retourne les lignes qui matchent le pattern (Grep)

Toutes les méthodes non-utilisées par ces classes renvoient IllegalCallerException.

Etape 2 :

Dans cette partie on s'intéresse à la lecture de fichier.txt

Nous avons créé une interface IlecteurArbreFichier qui contient la classe lireFichier.

De plus, il y a la classe abstraite LecteurArbreFichier qui est utilisée pour les documents contenants des champs spécifiques qui se trouvent dans la classe Champs.

Celle-ci est utilisée par la classe LecteurArbreFichier1 qui instancie les mots clés suivants :

MotDebut = « racine »

motFin= « fin »

charCommentaire= « % »

charDebut = « * »

positionCommentaire=3

positionEtoiles=0

positionNom=1

positionType=2

Dans la classe abstraite, la méthode lireFichier(String nomFichier) lit un fichier texte en vérifiant que le fichier ne contienne aucune erreur. Si une erreur apparaît l'exception FichierCorrompuException est levée.

Partie 3 : Dans cette partie nous nous sommes intéressés à l'interaction avec l'utilisateur.

Nous avons utilisé le design pattern Strategy pour implémenter les commandes. Ainsi nous avons créé l'interface CommandeEffectuer qui contient l'unique méthode

effectuerOperation(AbstractArbreFichiers a, String [] s), cette méthode prend en paramètre un AbstractArbreFichiers qui est utilisé pour effectuer des opérations sur les fils et un tableau de strings qui contient les arguments des commandes et retourne un AbstractArbreFichiers qui est utilisé uniquement dans la commande cd. Nous avons également créé la classe Commande qui a un champ de type CommandeEffectuer et qui utilise principalement la méthode setCommandeEffectuer(String ls) qui définie le type de commande à effectuer en fonction de ce que l'utilisateur entre.

Dans la classe Main nous avons la méthode main qui lance le programme en fonction des arguments entrés pour savoir si on doit lire un fichier ou pas. Il y a donc deux méthodes ProgrammePrincipal avec et sans arguments pour différencier la lecture de texte ou non. De plus, il y a la méthode saisieInteractive qui possède la boucle infinie et qui permet de communiquer avec l'utilisateur.