Licence Informatique 3 année

POO & Java

TP n°3 – Compression et Décompression avec Arbres Binaires

Contexte

Ce TP fait suite aux travaux sur les **arbres binaires** et la programmation orientée objet. L'objectif est de développer un programme simple capable de **compresser** et de **décompresser** des fichiers texte en utilisant un algorithme basé sur les arbres binaires (algorithme simplifié de Huffman).

Ce TP introduit également une interface en ligne de commande (CLI) permettant de manipuler les fichiers et d'interagir avec l'application via des arguments.

Le projet repose sur les notions fondamentales suivantes :
- L'utilisation des **interfaces** et des **classes abstraites**.
- L'héritage et le **polymorphisme**.
- La manipulation de collections Java (`Hashtable`, `PriorityQueue`, etc.).
- L'utilisation de **CLI** pour le parsing des arguments.
- La lecture et l'écriture de fichiers texte.

Objectifs Pédagogiques

1. Renforcer les compétences en **programmation orientée objet** en Java.
2. Mettre en œuvre un algorithme de compression utilisant des **arbres binaires**.
3. Découvrir les bases de la **compression** et de la **décompression**.
4. Apprendre à intégrer une interface en ligne de commande pour faciliter l'interaction avec l'application.

Énoncé du TP

Votre mission est de développer une application Java permettant de :
1. **Compresser un fichier texte** en générant des codes binaires pour les caractères en fonction de leur fréquence.
2. **Décompresser un fichier binaire** pour reconstituer le texte original.
3. Ajouter une interface CLI permettant de spécifier les fichiers d'entrée et de sortie, ainsi que de choisir l'action à réaliser (compression ou décompression).

Étapes du Projet

1. Interface `Compression`

Créez une interface qui définira les fonctionnalités principales de l'application :
- Méthode pour **compresser** un texte.

- Méthode pour **décompresser** un texte.

Exemple :

```
public interface Compression {
    byte[] compress(String data);   // Compression
    String decompress(byte[] data); // Décompression
}
```

---

2. Classe `HuffmanCompression`

Implémentez l'interface `Compression` en utilisant un arbre binaire de Huffman.

Cette classe doit :
- Construire un arbre de Huffman à partir des fréquences des caractères.
- Générer des codes binaires uniques pour chaque caractère.
- Fournir les méthodes `compress` et `decompress` pour encoder et décoder un texte.

Exemple simplifié :

```
public class HuffmanCompression implements Compression {
    private Hashtable<Character, String> charToCode; // Associe un caractère à un co
    private Hashtable<String, Character> codeToChar; // Associe un code binaire à un
    @Override
    public byte[] compress(String data) {
        // Implémentation de la compression
    }
    @Override
    public String decompress(byte[] data) {
        // Implémentation de la décompression
    }
    private void generateCodes(HuffmanNode root, String code) {
        // Récursion pour générer les codes binaires
    }
    private HuffmanNode buildHuffmanTree(Hashtable<Character, Integer> frequencyTabl
        // Construction de l'arbre de Huffman
    }
}
```

---

3. Classe `FileManager`

Créez une classe utilitaire pour la gestion des fichiers.

Elle doit inclure :
- Une méthode pour lire un fichier texte en chaîne de caractères.
- Une méthode pour écrire un fichier compressé sous forme binaire.
- Une méthode pour lire des données binaires d'un fichier.

Exemple :

```
import java.nio.file.*;
public class FileManager {
    public static String readFile(String filePath) throws IOException {
        return new String(Files.readAllBytes(Paths.get(filePath)));
    }
```

```
    public static void writeFile(String filePath, byte[] data) throws IOException {
        Files.write(Paths.get(filePath), data);
    }
}
```

---

4. Classe `Main` avec CLI

Ajoutez une interface en ligne de commande pour interagir avec l'application.

Commandes possibles :

```
java Main compress -i input.txt -o compressed.bin
java Main decompress -i compressed.bin -o output.txt
```

Parsing des arguments :
Utilisez une bibliothèque comme **Apache Commons CLI** pour gérer les options.

Exemple :

```
import org.apache.commons.cli.*;
public class Main {
    public static void main(String[] args) {
        Options options = new Options();
        options.addOption("c", "compress", false, "Compresser un fichier");
        options.addOption("d", "decompress", false, "Décompresser un fichier");
        options.addOption("i", "input", true, "Chemin du fichier d'entrée");
        options.addOption("o", "output", true, "Chemin du fichier de sortie");
        CommandLineParser parser = new DefaultParser();
        try {
            CommandLine cmd = parser.parse(options, args);
            String inputFile = cmd.getOptionValue("i");
            String outputFile = cmd.getOptionValue("o");
            if (cmd.hasOption("c")) {
                // Compression
                String data = FileManager.readFile(inputFile);
                HuffmanCompression compressor = new HuffmanCompression();
                byte[] compressedData = compressor.compress(data);
                FileManager.writeFile(outputFile, compressedData);
                System.out.println("Fichier compressé : " + outputFile);
            } else if (cmd.hasOption("d")) {
                // Décompression
                byte[] compressedData = FileManager.readFileAsBytes(inputFile);
                HuffmanCompression compressor = new HuffmanCompression();
                String decompressedData = compressor.decompress(compressedData);
                FileManager.writeFile(outputFile, decompressedData.getBytes());
                System.out.println("Fichier décompressé : " + outputFile);
            } else {
                System.out.println("Veuillez spécifier une option : --compress ou --
            }
        } catch (ParseException | IOException e) {
            System.err.println("Erreur : " + e.getMessage());
        }
    }
}
```

Livrables

À l'issue de ce TP, chaque binôme devra fournir les éléments suivants :

1. **Mise à jour du dépôt Git**
   - **Lien du dépôt Git** hébergé sur la plateforme de l'université (exemple : GitLab).
   - Le dépôt doit inclure :
     - Les fichiers `.java` nécessaires à l'exécution,
     - Les fichiers de test,
     - Le fichier **README.md**,
     - Tout autre fichier requis pour le projet (diagrammes, manuel utilisateur).
   - Chaque **push** doit être accompagné d'un message de commit clair.
   - Invitez l'encadrant de TP au projet Git pour permettre un suivi et des retours.
2. **Fichiers de test**
   - Un fichier texte d'entrée (`input.txt`) pour tester la compression.
   - Un fichier compressé (`compressed.bin`) généré par le programme.
   - Un fichier de sortie (`output.txt`) montrant la décompression réussie.
3. **Démo avec la CLI**
   - Une démonstration avec la ligne de commande (CLI) à réaliser en fin de séance.
   - Cette démonstration doit inclure les étapes suivantes :
     - Compression d'un fichier d'entrée et génération du fichier compressé.
     - Décompression du fichier compressé pour retrouver le contenu original.
   - La démo sera validée par l'encadrant.
4. **Mise à jour du fichier README**
   - Le fichier **README.md** doit inclure :
     - Une **description générale du projet**.
     - Les commandes pour exécuter le programme via la CLI.
     - Les étapes pour cloner, compiler et exécuter l'application.
     - Une explication des fichiers de test (fichier d'entrée, fichier compressé, fichier de sortie).
5. **Document de suivi de TP**
   - Actualisation du document type fourni avec :
     - La date de la séance,
     - Le plan de travail défini en début de TP,
     - Les tâches réalisées pendant la séance,
     - Les éventuelles difficultés rencontrées et les solutions envisagées.

Ressources Utiles

- Documentation officielle de Java : https://docs.oracle.com/en/java/
- Apache Commons CLI : https://commons.apache.org/proper/commons-cli/
- Tutoriel sur les arbres de Huffman : https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/
- Java PriorityQueue : https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html

---