



## **Virtual File System Organiser**

Nana Kwaku Amoako, Valerie Ackon. Ewura Abena Opare-Ansah, Ben Charles Abdul

Department of Computer Science, Ashesi University

CS222: Data Structures and Algorithms

Dr. Govindha Yeluripati

August 3, 2024

## Abstract

This paper presents the design and implementation of a file management or explorer system in Java through the implicit use of a tree data structure to represent hierarchical directory structures. The system comprises two primary classes: **File** and **Folder**, encapsulated within a **FileExplorerElement** superclass to ensure polymorphic behaviour. The **Folder** class serves as a tree node, containing a list of **FileExplorerElement** objects as its children, thus enabling the representation of directories containing both files and subdirectories.

Key operations of the file management system include file and directory creation, deletion, searching, sorting, and displaying the directory structure. The searching mechanism leverages recursive traversal (depth-first searching) to locate files and folders, while sorting is implemented using a custom algorithm that classifies based on set attributes to maintain efficient organization of directory contents. This custom algorithm borrows from concepts in different sort algorithm types.

**Keywords:** *file, folder, data structure, algorithms, search, sort, recursion*

# Contents

<b>1. Project Overview</b>	4
<b>2. Objectives</b>	4
<b>3. Design and Implementation Details</b>	4
Code Structure	4
Description of Classes:	5
<b>4. Data Structures and Algorithms Used</b>	5
Data Structures:	5
Algorithms:	5
<b>5. Performance Analysis and Optimization Techniques</b>	6
Time Complexity, Worst Case Scenario Analysis:	6
Optimisation Techniques:	6
<b>6. Challenges Faced and Solutions Implemented</b>	6
Challenge: Accurate Indentation for Directory Display:	6
Challenge: File Creation in Nested Directories:	6
Challenge: Ensuring Robust Error Handling:	7
GUI Integration:	7
<b>7. Instructions for Using the Application</b>	7
Running the GUI Application:	7
Running the CLI:	7
Sample Screenshot:	8
<b>8. Using the CLI</b>	8
<b>Appendix (GitHub Link)</b>	9

# 1. Project Overview

The project aimed to develop a comprehensive file explorer with a graphical user interface (GUI) and a command-line interface (CLI) for enhanced usability. The application provides functionalities similar to standard file explorers, seen within different operating systems, allowing users to create, delete, move, search, and sort files and directories.

## 2. Objectives

1. Develop a functional file explorer with a user-friendly GUI.
2. Implement essential file operations such as create, delete, and move.
3. Ensure accurate display of directory structures.
4. Provide a CLI for advanced users.
5. Optimise the application for performance.

## 3. Design and Implementation Details

### Code Structure

**Directory**: Manages the directory structure and file operations.

**Folder**: Represents a storage container for files and other folders. Folders make up the directory and are the reason for the implementation of a tree data structure.

**File**: Represents files.

**FileExplorerElement**: Base class for files and folders.

**FileExplorer**: Main GUI application class.

**Main**: CLI interface for the file explorer.

## Description of Classes:

- **Directory**: Contains methods for creating, deleting, moving files and directories, and displaying the directory structure.
- **Folder**: Extends **FileExplorerElement**, representing directories.
- **File**: Extends **FileExplorerElement**, representing files.
- **FileExplorer**: Implements the GUI using Java Swing, providing an interface for users to interact with the file system.
- **Main**: Provides a CLI interface for file operations.

## 4. Data Structures and Algorithms Used

### Data Structures:

- **Tree Structure**: Used to represent the file and directory hierarchy, enabling efficient traversal and management. It was chosen because of the hierarchical nature of file management systems.
- **ArrayList**: Used to store the contents of directories, allowing dynamic resizing and easy access.

### Algorithms:

- **Depth-First Search (DFS)**: Used for displaying the directory structure and searching files.
- **Sorting Algorithm**: Custom sorting is implemented to sort files and directories based on various attributes (e.g. name, item type, size, date modified)

## 5. Performance Analysis and Optimization Techniques

Time Complexity, Worst Case Scenario Analysis:

- **File and Directory Creation:**  $O(n)$ , where  $n$  is the number of directories in the path.
- **File and Directory Deletion:**  $O(n)$ , where  $n$  is the number of elements in the directory.
- **Directory Display:**  $O(n)$ , where  $n$  is the total number of files and directories.
- **Sort:**  $O(n \log n)$ , where  $n$  is the total number of files and directories.
- **Search:**  $O(n)$ , where  $n$  is the total number of files and directories.

Optimisation Techniques:

- Efficient use of data structures like *ArrayList* and *Trees* for managing file and directory operations.
- Recursive algorithms for displaying and traversing directories to minimise code complexity.

## 6. Challenges Faced and Solutions Implemented

Challenge: Accurate Indentation for Directory Display:

- **Solution:** Modified the recursive method to properly indent directory levels using a multiplier for spaces.

Challenge: File Creation in Nested Directories:

- **Solution:** Enhanced the `createFile` method to navigate through the directory path and create the file at the correct location.

## Challenge: Ensuring Robust Error Handling:

- **Solution:** Introduced try-catch blocks across all methods to handle exceptions and provide meaningful error messages.

## GUI Integration:

- **Solution:** Seamlessly integrated file operations with the GUI, ensuring user inputs are correctly handled and reflected in the file system.

# 7. Instructions for Using the Application

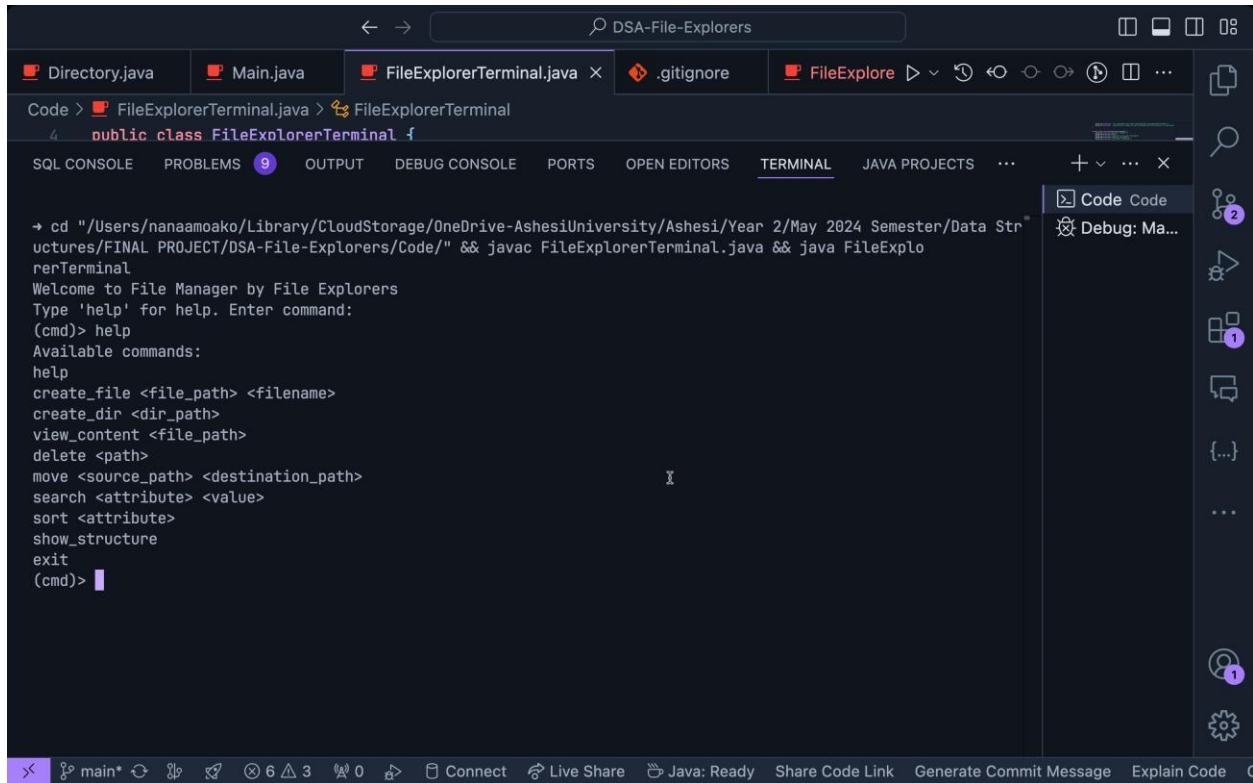
## Running the GUI Application:

1. Compile and run the `FileExplorer.java` file.
2. Use the provided GUI to create, delete, move files and directories, and view the directory structure.

## Running the CLI:

1. Compile and run the `Main.java` file.
2. Enter commands as per the CLI instructions to perform file operations.

## Sample Screenshot:



The screenshot shows an IDE window titled "DSA-File-Explorers". The editor displays the `FileExplorerTerminal.java` file. The code defines a `FileExplorerTerminal` class with a `main` method. The `main` method runs a shell command to compile and run the program. The output of the program is displayed in the terminal pane, showing a welcome message and a list of available commands: `help`, `create_file`, `create_dir`, `view_content`, `delete`, `move`, `search`, `sort`, `show_structure`, and `exit`.

```
Code > FileExplorerTerminal.java > FileExplorerTerminal
4 public class FileExplorerTerminal {

SQL CONSOLE PROBLEMS 9 OUTPUT DEBUG CONSOLE PORTS OPEN EDITORS TERMINAL JAVA PROJECTS ... + v ... x
+ cd "/Users/nanaamoako/Library/CloudStorage/OneDrive-AshesiUniversity/Ashesi/Year 2/May 2024 Semester/Data Str
uctures/FINAL PROJECT/DSA-File-Explorers/Code/" && javac FileExplorerTerminal.java && java FileExplo
rerTerminal
Welcome to File Manager by File Explorers
Type 'help' for help. Enter command:
(cmd)> help
Available commands:
help
create_file <file_path> <filename>
create_dir <dir_path>
view_content <file_path>
delete <path>
move <source_path> <destination_path>
search <attribute> <value>
sort <attribute>
show_structure
exit
(cmd)>
```

## 8. Using the CLI

- **create\_file <file\_path>**: Create a new file at the specified path.
- **create\_dir <dir\_path>**: Create a new directory at the specified path.
- **delete <path>**: Delete the file or directory at the specified path.
- **move <source\_path> <destination\_path>**: Move a file or directory from the source path to the destination path.
- **search <attribute> <value>**: Search for files or directories based on a specified attribute and value.
- **sort <attribute>**: Sort files and directories by the specified attribute.
- **show\_structure**: Display the current directory and file structure in a tree format.



## Appendix (GitHub Link)

[GitHub Repository](#)