Basic File Manager

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING)

By

Lalmalsawm Guite

Registration number: 12222076

Supervisor

Mr. Waseem Ud DinWani

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation/dissertation proposal entitled **Basic File Manager** in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried

out under supervision of my project supervisor **Mr. Waseem Ud Din Wani.**

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Lalmalsawm Guite

R. No: 12222076

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B. Tech Dissertation/dissertation proposal entitled, **Basic File Manager** submitted by **Lalmalsawm Guite** at **Lovely Professional University, Phagwara, India** is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Mr. Waseem Ud Din Wani

Date:

Acknowledgment:

I would like to take this opportunity to express my sincere gratitude to my Java teacher, Sir Waseem ud Din Wani. His guidance and support throughout the course have been invaluable to me.

Sir Waseem has a deep understanding of Java and a passion for teaching. He is always willing to go the extra mile to help his students succeed. His lectures are clear and concise, and his assignments are well-designed to challenge us and help us learn.

I am particularly grateful for Sir Waseem's patience and encouragement. He never gives up on his students, even when we struggle. He is always there to offer help and support, and he always believes in us.

I am confident that the skills and knowledge I have gained in Sir Waseem's Java class will be invaluable to me in my future career. I am truly grateful for his guidance and support.

Thank you, Sir Waseem!

Table of Contents

Abstract*:*

The "Basic File Manager" project aims to create a user-friendly Java application that facilitates file and directory management. This project allows users to perform operations such as creating, renaming, copying, moving, and deleting files and directories. By implementing file and directory structures using Java's File class, users can interact with the application through a command-line interface.

Introduction*:*

The rapid growth of digital data necessitates efficient file management tools. A basic file manager serves as a foundational tool for organizing and manipulating files and directories. In this project, we introduce a user-friendly and versatile file manager application developed in Java.

The management of files and data is an essential skill that underpins nearly every aspect of modern life and work. Whether you're a student organizing research materials, a professional managing critical documents, or simply an individual curating your personal media collection, the ability to efficiently and effectively manage files is paramount. This college project on basic file management aims to explore the fundamental principles and practices of organizing, storing, and manipulating digital files.

As we delve into this topic, we will uncover the significance of file management in maintaining order and accessibility in our digital lives. We will learn how proper file organization can save time, reduce frustration, and enhance productivity. Understanding file management basics is not only practical but also empowers us to be more responsible digital citizens, ensuring data security and preventing the loss of valuable information.

Throughout this project, we will examine various aspects of file management, including file structure, naming conventions, folder organization, file types, backup strategies, and the use of software tools to streamline these processes. By the project's end, you will have gained a comprehensive understanding of how to effectively manage your digital files, helping you navigate the digital world with confidence and efficiency.

Objective of the Project:

The primary objectives of the "Basic File Manager" project are as follows:

- Understand File Structure: Comprehend the hierarchical structure of files and folders, and recognize how this structure impacts the organization and retrieval of data.

- Implement Effective File Naming Conventions: Develop and apply systematic file naming conventions that enhance searchability and reduce ambiguity.

- Create Logical Folder Structures: Design and maintain well-structured folders to categorize and store files efficiently.

- Identify and Handle Different File Types: Recognize various file formats and learn how to work with them appropriately.

- Implement Data Backup Strategies: Explore the importance of data backup, and learn how to create and maintain backup copies of critical files to prevent data loss.

- Utilize File Management Software: Familiarize oneself with file management software tools and understand how they can streamline and simplify file-related tasks.

- Enhance Digital Organization: Gain practical experience in improving digital organization, leading to increased productivity and reduced stress in managing digital content.

- Promote Data Security: Develop an awareness of data security best practices to protect digital files from loss or unauthorized access.

- Troubleshoot File Management Issues: Acquire problem-solving skills to address common file management challenges.

Description of the Project:

The "Basic File Manager" project employs Java to develop a command-line file manager application. Users interact with the application by entering commands to perform file operations. The application utilizes the Java File class to manage file and directory structures efficiently.

System Architecture:

The file manager application follows a modular architecture, including the following components:

- User Interface: A command-line interface for user input and output.
- File Manager Logic: Java classes for handling file operations and directory management.
- File and Directory Structures: Utilizing Java's File class to represent file and directory structure

Source Code:

The complete source code for the "Basic File Manager" can be found at [GitHub repository link]. The code is organized into various classes and methods, each with a specific purpose related to file management. Detailed code documentation is available within the repository.

```
import java.io.File;
import java.util.Scanner;

public class BasicFileManager {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        File currentDirectory = new File(System.getProperty("user.dir"));
```

```
while (true) {
    System.out.println("Current Directory: " + currentDirectory.getAbsolutePath());
    System.out.println("Select an operation:");
    System.out.println("1. List files and directories");
    System.out.println("2. Create file");
    System.out.println("3. Create directory");
    System.out.println("4. Rename file/directory");
    System.out.println("5. Copy file/directory");
    System.out.println("6. Move file/directory");
    System.out.println("7. Delete file/directory");
    System.out.println("8. Change directory");
    System.out.println("9. Exit");

    int choice = scanner.nextInt();
    scanner.nextLine();

    switch (choice) {
        case 1:
            listFilesAndDirectories(currentDirectory);
            break;
        case 2:
            createFile(currentDirectory, scanner);
            break;
        case 3:
            createDirectory(currentDirectory, scanner);
            break;
        case 4:
            renameFileOrDirectory(currentDirectory, scanner);
            break;
            case 5:
                copyFileOrDirectory(currentDirectory, scanner);
                break;
            case 6:
                moveFileOrDirectory(currentDirectory, scanner);
                break;
            case 7:
                deleteFileOrDirectory(currentDirectory, scanner);
                break;
            case 8:
                changeDirectory(currentDirectory, scanner);
                break;
            case 9:
                System.out.println("Exiting the file manager. Goodbye!");
                scanner.close();
                System.exit(0);
            default:
                System.out.println("Invalid choice. Please select a valid option.");
        }
    }
}

private static void listFilesAndDirectories(File directory) {
    File[] files = directory.listFiles();
    for (File file : files) {
        System.out.println(file.getName());
    }
}

private static void createFile(File directory, Scanner scanner) {
    System.out.print("Enter the name of the file to create: ");
    String fileName = scanner.nextLine();
    File newFile = new File(directory, fileName);

    try {
        if (newFile.createNewFile()) {
            System.out.println("File created successfully.");
        } else {
            System.out.println("File already exists.");
        }
    } catch (Exception e) {
        System.out.println("Error creating file: " + e.getMessage());
    }
}

private static void createDirectory(File directory, Scanner scanner) {
    System.out.print("Enter the name of the directory to create: ");
    String dirName = scanner.nextLine();
    File newDirectory = new File(directory, dirName);

    if (newDirectory.mkdir()) {
        System.out.println("Directory created successfully.");
    } else {
        System.out.println("Directory already exists.");
    }
}

private static void renameFileOrDirectory(File directory, Scanner scanner) {
    System.out.print("Enter the name of the file/directory to rename: ");
    String oldName = scanner.nextLine();
    File oldFile = new File(directory, oldName);

    System.out.print("Enter the new name: ");
    String newName = scanner.nextLine();
    File newFile = new File(directory, newName);

    if (oldFile.renameTo(newFile)) {
        System.out.println("Renamed successfully.");
    } else {
        System.out.println("Error renaming file/directory.");
    }
}

private static void copyFileOrDirectory(File directory, Scanner scanner) {
    System.out.print("Enter the name of the file/directory to copy: ");
    String sourceName = scanner.nextLine();
    File sourceFile = new File(directory, sourceName);

    System.out.print("Enter the destination directory: ");
    String destinationDir = scanner.nextLine();
    File destinationDirectory = new File(directory, destinationDir);

    if (sourceFile.isDirectory()) {
        copyDirectory(sourceFile, new File(destinationDirectory, sourceFile.getName()));
    } else {
        copyFile(sourceFile, new File(destinationDirectory, sourceFile.getName()));
    }
}

private static void copyFile(File source, File destination) {
}

private static void copyDirectory(File source, File destination) {

}

private static void moveFileOrDirectory(File directory, Scanner scanner) {
    System.out.print("Enter the name of the file/directory to move: ");
    String sourceName = scanner.nextLine();
    File sourceFile = new File(directory, sourceName);

    System.out.print("Enter the destination directory: ");
    String destinationDir = scanner.nextLine();
    File destinationDirectory = new File(directory, destinationDir);

    if (sourceFile.renameTo(new File(destinationDirectory, sourceFile.getName()))) {
        System.out.println("Moved successfully.");
    } else {
        System.out.println("Error moving file/directory.");
    }
}

private static void deleteFileOrDirectory(File directory, Scanner scanner) {
    System.out.print("Enter the name of the file/directory to delete: ");
    String name = scanner.nextLine();
    File fileToDelete = new File(directory, name);

    if (fileToDelete.isDirectory()) {
        deleteDirectory(fileToDelete);
        } else {
            if (fileToDelete.delete()) {
                System.out.println("Deleted successfully.");
            } else {
                System.out.println("Error deleting file/directory.");
            }
        }
    }

    private static void deleteDirectory(File directory) {

    }

    private static void changeDirectory(File currentDirectory, Scanner scanner) {
        System.out.print("Enter the path to the new directory: ");
        String newPath = scanner.nextLine();
        File newDirectory = new File(currentDirectory, newPath);

        if (newDirectory.isDirectory()) {
            currentDirectory = newDirectory;
        } else {
            System.out.println("Invalid directory path.");
        }
    }
}
```

Input/Output:

Users input file operations via the command-line interface. The application processes these inputs, performs the requested operations, and displays the file and directory structures as needed. For instance, the application allows users to create files or directories and view the contents of a directory.

File Operations: The "Basic File Manager" supports the following file operations:

- **Create:** Users can create new files and directories.
- **Rename:** Files and directories can be renamed.
- **Copy:** Files and directories can be duplicated.
- **Move:** Files and directories can be relocated.
- **Delete:** Users can remove files and directories.

Scope of the Project:

While the "Basic File Manager" provides essential file management capabilities, it has certain limitations. It does not support advanced features such as file encryption, advanced search, or integration with cloud services. The project focuses on providing core file management functionalities.

Development Process:

Requirements gathering: Identify the features and functionalities that the file manager should provide. Consider the needs of the target users and the specific tasks that they need to perform.

System design: Develop a high-level overview of the system architecture, including the different components and how they will interact with each other.

Implementation: Write the code for the different components of the system, following the system design.

Testing: Test the system thoroughly to ensure that it meets all the requirements and functions as expected.

Deployment: Release the system to the target users.

Here is a more detailed breakdown of each step:

Requirements gathering:

Identify the target users: Who will be using the file manager? What are their needs and requirements?

Brainstorm features: What features and functionalities should the file manager provide?

Prioritize features: Not all features will be equally important to all users. Prioritize the features based on the needs of the target users.

System design:

Choose a programming language: The file manager can be developed in any programming language, such as Java, Python, or C++. Choose a language that you are familiar with and that is well-suited for developing desktop applications.

Design the user interface: The user interface should be easy to use and navigate. Consider using a graphical user interface (GUI) or a command-line interface (CLI).

Design the file system model: The file system model should represent the hierarchical structure of files and directories.

Design the file operations: Implement the necessary file operations, such as creating, deleting, moving, and copying files and directories.

Implementation:

Implement the different components of the system, following the system design.

Use a modular design approach to make the code reusable and maintainable.

Write unit tests to test the individual components of the system.

Testing:

Test the system thoroughly to ensure that it meets all the requirements and functions as expected.

Conduct functional testing to verify that the system performs all the intended features and functionalities.

Conduct non-functional testing to assess the performance, security, and usability of the system.

Deployment:

Release the system to the target users.

Provide documentation and support to help users get started with the system.

This development process can be adapted to fit the specific needs of the project. For example, if the file manager is being developed for a specific organization, the requirements gathering step may involve meeting with stakeholders to understand their needs.

Conclusion:

The "Basic File Manager" project has successfully demonstrated the implementation of fundamental file management principles in Java. The application provides a user-friendly command-line interface for performing various file and directory operations. The application utilizes the Java File class to efficiently manage file and directory structures.

This project has provided valuable insights into the design and development of a basic file manager application. It has also highlighted the importance of effective file organization and management in maintaining order and accessibility in our digital lives.

The project can be extended and enhanced in several ways, such as:

Developing a graphical user interface (GUI) for the application to provide a more user-friendly experience.

Implementing additional file operations, such as searching, filtering, and sorting.

Adding support for network drives and cloud storage services.

Developing mechanisms for automating file management tasks.

Implementing security features to protect the application and its contents from unauthorized access.

Overall, the "Basic File Manager" project has been a successful endeavor, providing valuable insights into the design and development of a basic file management application in Java.