Use Directory Explorer

Estimated time needed: 40 minutes

After completing this lab, you will know how to manage files and directories using Java packages. You will be creating a console application to handle files and directories.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully-online integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Use Java's file-handling classes
- List all the files and subdirectories in a directory
- Implement features to create new directories
- Delete existing directories
- Rename files or directories
- Navigate through the directory structure

File-handling classes

The `java.io.File` class is used to represent files and directories.

1. A File object represents a file or directory on the file system.

2. The file path is the sequence of directories and subdirectories that leads to the file. The File class uses the `path` attribute to store the file path.

3. The file name is the name of the file without the directory path. The File class provides the `getName()` method to retrieve the file name.

4. The File class that represents the directory provides the `list()` method to retrieve a list of files and subdirectories within a directory.

5. The File class provides the `mkdir()` method to create a new directory.

6. Create a project directory by running the following command.

   ```
   mkdir my_dir_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_dir_proj/src
   mkdir -p my_dir_proj/classes
   ```

```
mkdir -p my_dir_proj/test
cd my_dir_proj
```

3. Now create a file named `DirectoryExplorer.java` inside the src directory.

```
touch /home/project/my_dir_proj/src/DirectoryExplorer.java
```

4. Click the **Open DirectoryExplorer.java in IDE** button to open the file for editing.

Open **DirectoryExplorer.java** in IDE

5. Read each statement and the accompanying explanations in the following program to understand how to handle files and directories in Java. Paste the following content in `DirectoryExplorer.java`.

```
import java.io.File;
import java.util.Scanner;
public class DirectoryExplorer {
    public static void main(String s[]) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the name of the file or directory with the path");
        String fileName = scanner.nextLine();
        File file = new File(fileName);
        if(file.exists()) {
            if(file.isFile()) {
                System.out.println(fileName+" is a file");
            } else {
                System.out.println(fileName+" is a directory");
            }
        } else {
            System.out.println(fileName+" is a not a valid file or directory");
        }
    }
}
```

6. In the IDE, compile the Java program. specifying the destination directory as the `classes` directory that you created. See the following code example.

```
javac -d classes src/DirectoryExplorer.java
```

7. Set the `CLASSPATH` variable.

```
export CLASSPATH=$CLASSPATH:/home/project/my_dir_proj/classes
```

8. Now, when you run the Java program, the Java program will run seamlessly as expected.

```
java DirectoryExplorer
```

You will see the following output:

If the user input is a valid file name

```
Enter the name of the file or directory with the path
/home/project/my_dir_proj/src/DirectoryExplorer.java
/home/project/my_dir_proj/src/DirectoryExplorer.java is a file
```

If the user input is a valid directory name

```
Enter the name of the file or directory with the path
/home/project/my_dir_proj/src
/home/project/my_dir_proj/src is a directory
```

If the user input is an invalid name (neither directory nor file)

```
Enter the name of the file or directory with the path
/home/project/nonexistentname
/home/project/nonexistentname is a not a valid file or directory
```

Create a directory or file

1. Click the **Open DirectoryExplorer.java in IDE** button to open the file for editing if the IDE is not already open.

Open **DirectoryExplorer.java** in IDE

2. Read each statement in the following program to understand how to create a file or directory. Paste the following content in `DirectoryExplorer.java`.

```
// Import the File class for file and directory operations
import java.io.File;
// Import the Scanner class for user input
 import java.util.Scanner;
// Import the IOException class for handling file I/O exceptions
import java.io.IOException;
public class DirectoryExplorer {
    public static void main(String s[]) {
        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);
        // Loop indefinitely until the user chooses to exit
        while (true) {
            // Display the menu options to the user
            System.out.println("\nPress 1 for File Management," + "\nAny other key to exit");
            // Read the user's choice
            String userAction = scanner.nextLine();
            // Option 1: File Management
            if (userAction.equals("1")) {
                // Prompt the user to enter the name of the file or directory with the path
                System.out.println("Enter the name of the file or directory with the path");
                // Read the file name with path
                String fileName = scanner.nextLine();
                // Create a File object representing the file or directory
                File file = new File(fileName);
                // Check if the file or directory exists
                if (file.exists()) {
                    // Check if the file is a file or directory
                    if (file.isFile()) {
                        System.out.println(fileName + " is a file");
                    } else {
                        System.out.println(fileName + " is a directory");
                    }
                } else {
                    // If the file or directory does not exist, prompt the user to create it
                    System.out.println(fileName + " is a not a valid file or directory");
                    System.out.println("To create a file with given name press 1\n"
                            + "To create a directory with given name press 2\n"
                            + "To do nothing and continue, press any other key");
```

```
                    // Read the user's choice
                    String createChoice = scanner.nextLine();
                    // Create a file
                    if (createChoice.equals("1")) {
                        // Get the parent directory
                        String parentDirStr = file.getParent();
                        File parentDir = new File(parentDirStr);
                        // Create the parent directory if it does not exist
                        if (!parentDir.exists()) {
                            boolean created = parentDir.mkdirs();
                            if (!created) {
                                System.out.println("The parent directory could not be created");
                                continue;
                            }
                        }
                        // Create the file
                        try {
                            file.createNewFile();
                            System.out.println("File successfully created!");
                        } catch (IOException ioe) {
                            System.out.println("Unable to create file. " + ioe.getMessage());
                        }
                    }
                    // Create a directory
                    else if (createChoice.equals("2")) {
                        // Create the directory
                        boolean created = file.mkdirs();
                        if (created) {
                            System.out.println("The directory has been created");
                        } else {
                            System.out.println("The directory couldn't be created");
                        }
                    }
                }
            }
            // Exit the program
            else {
                System.out.println("Bye!");
                break;
            }
        }
    }
}
```

Here's an explanation of the code:

- Creates a Scanner object to read user input.
- Reads the user's choice. `1` for file management. Any other key for exit.
- If the user chooses "1", proceed to File Management.
- If the user chooses any other key, exit the program.
- Prompt the user to enter the name of the file or directory with the path.
- Reads the file name with path.
- Creates a File object representing the file or directory.
- Checks if the file or directory exists.
- If it exists, checks if it is a file or directory and displays appropriate message.
- If it does not exist, prompts the user to create the file or directory.
- Displays options: "To create a file with given name press 1", "To create a directory with given name press 2", or "To do nothing and continue, press any other key".
- Reads the user's choice.
- If the user chooses "1", create a file.
    - Get the parent directory.

    - Creates the parent directory if it does not exist. Continues the file management loop if the parent directory cannot be created.
    - Creates the file.
- If the user chooses "2", create a directory.
    - Creates the directory. Displays appropriate message if the parent directory cannot be created.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/DirectoryExplorer.java
```

4. Now, run the Java program.

```
java DirectoryExplorer
```

Please see the following sample output.

```
Press 1 for File Management,
Any other key to exit
1
Enter the name of the file or directory with the path
/home/project/myTempDir
/home/project/myTempDir is a not a valid file or directory
To create a file with given name press 1
To create a directory with given name press 2
To do nothing and continue, press any other key
2
The directory has been created
Press 1 for File Management,
Any other key to exit
1
Enter the name of the file or directory with the path
/home/project/myTempDir
/home/project/myTempDir is a directory
Press 1 for File Management,
Any other key to exit
```

You can try running with various permutations and combinations.

Add logic for more operations

At this point, the code only can tell you whether the path provided is a file or directory. Now you will add the logic to list all the files and directories, to rename the directory or delete the directory, if the given path is that of a directory. If the give path is a file, you will allow the user to rename the file or delete the file.

In this program, for readability, you will split the logic into two methods `fileManagement(File file)` - to handle all the operations related to files and `directoryManagement(File dirObj)` - to handle all the operations related to directory.

1. Click the following button called **Open DirectoryExplorer.java in IDE** to open the file for editing if the IDE is not already open.

Open **DirectoryExplorer.java** in IDE

2. Read each statement in the following program to understand
    - If the given path is a directory, how to list all the files and directories in the directory, if the given path is that of a directory, rename the directory or delete the directory.
    - If the given path is a file, rename the file or delete the file.

Paste the following content in `DirectoryExplorer.java`.

```
import java.io.File;
import java.util.Scanner;
import java.io.IOException;
public class DirectoryExplorer {
    // Method to handle file management operations
    private static void fileManagement(File file) {
        // Display options to the user
        System.out.println("\nPress 1 to rename the file," +
                "\nPress 2 to delete the file," +
                "\nAny other key to exit");
        Scanner scanner = new Scanner(System.in);
        String userChoice = scanner.nextLine();
        // Handle user choice
        if (userChoice.equals("1")) {
            // Get new name for the file
            System.out.println("Enter the new name for the file " + file.getName());
            String newfileName = scanner.nextLine();
            // Attempt to rename the file
            boolean changed = file.renameTo(new File(file.getParent(), newfileName));
            //Print message about success or failure of file renaming
            if (changed) {
                System.out.println("Filename successfully changed");
            } else {
                System.out.println("Filename couldn't be changed!");
            }
        } else if (userChoice.equals("2")) {
            // Delete the file
            boolean deleted = file.delete();
            //Print message about success or failure of file deletion
            if (deleted) {
                System.out.println("Filename successfully deleted");
            } else {
                System.out.println("Filename couldn't be deleted!");
            }
        }
    }
    // Method to handle directory management operations (list, rename, or delete)
    private static void directoryManagement(File dirObj) {
        // Display options to the user
        System.out.println("\nPress 1 to list the directory," +
                "\nPress 2 to rename the directory," +
                "\nPress 3 to delete the directory," +
                "\nAny other key to exit");
        Scanner scanner = new Scanner(System.in);
        String userChoice = scanner.nextLine();
        // Handle user choice
        if (userChoice.equals("1")) {
            // List the contents of the directory
            String fileNames[] = dirObj.list();
            if (fileNames.length == 0) {
                System.out.println("The directory is empty!");
```

```
            } else {
                for (int i = 0; i < fileNames.length; i++) {
                    System.out.println(fileNames[i]);
                }
            }
        } else if (userChoice.equals("2")) {
            // Rename the directory
            System.out.println("Enter the new name for the directory " + dirObj.getName());
            String newDirName = scanner.nextLine();
            // Attempt to rename the directory
            boolean changed = dirObj.renameTo(new File(dirObj.getParent(), newDirName));
            //Print message about success or failure of dir renaming
            if (changed) {
                System.out.println("Directory name successfully changed");
            } else {
                System.out.println("Directory name couldn't be changed!");
            }
        } else if (userChoice.equals("3")) {
            // Delete the directory
            boolean deleted = dirObj.delete();
            //Print message about success or failure of dir deletion
            if (deleted) {
                System.out.println("Directory successfully deleted");
            } else {
                System.out.println("Directory couldn't be deleted! It might not be empty");
            }
        }
    }
    public static void main(String s[]) {
        Scanner scanner = new Scanner(System.in);
        // Loop indefinitely until the user chooses to exit
        while (true) {
            // Display the menu options to the user
            System.out.println("\nPress 1 for File Management," + "\nAny other key to exit");
            // Read the user's choice
            String userAction = scanner.nextLine();
            // Option 1: File Management
            if (userAction.equals("1")) {
                // Prompt the user to enter the name of the file or directory with the path
                System.out.println("Enter the name of the file or directory with the path");
                // Read the file name with path
                String fileName = scanner.nextLine();
                // Create a File object representing the file or directory
                File file = new File(fileName);
                // Check if the file or directory exists
                if (file.exists()) {
                    // Check if the file is a file or directory
                    if (file.isFile()) {
                        System.out.println(fileName + " is a file");
                        // Call the fileManagement method for file operations
                        fileManagement(file);
                    } else {
                        System.out.println(fileName + " is a directory");
                        // Call the directoryManagement method for directory operations
                        directoryManagement(file);
                    }
                } else {
                    // If the file or directory does not exist, prompt the user to create it
                    System.out.println(fileName + " is not a valid file or directory");
                    System.out.println("To create a file with given name press 1\n"
                            + "To create a directory with given name press 2\n"
                            + "To do nothing and continue, press any other key");
                    // Read the user's choice
                    String createChoice = scanner.nextLine();
                    // Create a file
                    if (createChoice.equals("1")) {
                        // Get the parent directory
                        String parentDirStr = file.getParent();
                        File parentDir = new File(parentDirStr);
                        // Create the parent directory if it does not exist
                        if (!parentDir.exists()) {
                            boolean created = parentDir.mkdirs();
                            if (!created) {
                                System.out.println("The parent directory could not be created");
                                continue;
                            }
                        }
```

```
                        // Create the file
                        try {
                            file.createNewFile();
                            System.out.println("File successfully created!");
                        } catch (IOException ioe) {
                            System.out.println("Unable to create file. " + ioe.getMessage());
                        }
                    }
                    // Create a directory
                    else if (createChoice.equals("2")) {
                        // Create the directory
                        boolean created = file.mkdirs();
                        if (created) {
                            System.out.println("The directory has been created");
                        } else {
                            System.out.println("The directory couldn't be created");
                        }
                    }
                }
            }
            // Exit the program
            else {
                System.out.println("Bye!");
                break;
            }
        }
    }
}
```

File Management - `private static void fileManagement(File file)`

This code is responsible for handling operations related to files. Specifically, it allows the user to perform two main actions on a file:

- Rename the file.
- Delete the file.

The method first displays a menu of options to the user:

- Press 1 to rename the file
- Press 2 to delete the file
- Any other key to exit

A Scanner object is used to read the user's choice from the console.

If the user presses 1, the method allows the user to rename the file. The user is prompted to enter a new name for the file. The renameTo method is used to attempt to rename the file. If the renaming is successful, a success message is displayed. Otherwise, an error message is shown.

If the user presses 2, the method allows the user to delete the file. The delete method is used to attempt to delete the file. If the deletion is successful, a success message is displayed. Otherwise, an error message is shown.

If the user enters any other key, the method simply exits without performing any action.

Directory Management - `private static void directoryManagement(File dirObj)`

This code is responsible for handling operations related to directories (folders). It allows the user to perform three main actions on a directory:

- List the contents of the directory.

- Rename the directory.
- Delete the directory.

The method first displays a menu of options to the user:

- Press 1 to list the directory
- Press 2 to rename the directory
- Press 3 to delete the directory
- Any other key to exit

This allows the user to choose what they want to do with the directory.

If the user presses 1, the method lists the contents of the directory. The list() method is used to retrieve an array of file and subdirectory names within the directory. If the directory is empty, a message is displayed. Otherwise, the contents of the directory are printed to the console.

If the user presses 2, the method allows the user to rename the directory. The user is prompted to enter a new name for the directory. The renameTo method is used to attempt to rename the directory. If the renaming is successful, a success message is displayed. Otherwise, an error message is shown.

If the user presses 3, the method allows the user to delete the directory. The delete method is used to attempt to delete the directory. If the deletion is successful, a success message is displayed. Otherwise, an error message is shown (for example, if the directory is not empty).

If the user enters any other key, the method simply exits without performing any action.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/DirectoryExplorer.java
   ```

4. Now, run the Java program.

   ```
   java DirectoryExplorer
   ```

Please see the following sample output.

```
Press 1 for File Management,
Any other key to exit
1
Enter the name of the file or directory with the path
/home/project/myTempDir
/home/project/myTempDir is not a valid file or directory
To create a file with given name press 1
To create a directory with given name press 2
To do nothing and continue, press any other key
2
```

```
The directory has been created
Press 1 for File Management,
Any other key to exit
1
Enter the name of the file or directory with the path
/home/project/myTempDir
/home/project/myTempDir is a directory
Press 1 to list the directory,
Press 2 to rename the directory,
Press 3 to delete the directory,
Any other key to exit
1
The directory is empty!
Press 1 for File Management,
Any other key to exit
2
Bye!
```

You can try running with various permutations and combinations.

Recursively printing the list of all directories (Optional)

In the `DirectoryExplorer` if the user input is a directory on the that directory is listed. If the directory has another directory inside, that is not recursively listed. Now you will write a new program to recursively list the inner directories. This uses a mechanism called `recursion` wherein a method calls itself.

1. Create a file named `DirectoryRecursiveList.java` inside the src directory.

```
touch /home/project/my_dir_proj/src/DirectoryRecursiveList.java
```

4. Click the following button called **Open DirectoryRecursiveList.java in IDE** to open the file for editing.

Open **DirectoryRecursiveList.java** in IDE

5. Read each statement and the accompanying explanations in the following program to understand how to list directories in Java, recursively. Paste the following content in `DirectoryRecursiveList.java`.

```
// Import the File class for file and directory operations
import java.io.File;
// Import the Scanner class for user input
import java.util.Scanner;
// Import the IOException class for handling file I/O exceptions
import java.io.IOException;
public class DirectoryRecursiveList {
    /**
     * Method to recursively list the contents of a directory
     *
     * @param dirObj the directory object
     */
```

```
    private static void directoryList(File dirObj) {
        // Print a message indicating that the directory is being listed recursively
        System.out.println("Printing " + dirObj + " recursively");
        // Get a list of files and subdirectories in the directory
        File files[] = dirObj.listFiles();
        // Iterate over the list of files and subdirectories
        for (int i = 0; i < files.length; i++) {
            // Check if the current item is a file
            if (files[i].isFile()) {
                // Print the absolute path of the file
                System.out.println(files[i].getAbsolutePath());
            } else {
                // If the current item is a subdirectory, recursively list its contents
                directoryList(files[i]);
            }
        }
    }
    public static void main(String s[]) {
        Scanner scanner = new Scanner(System.in);
        // Loop indefinitely until the user chooses to exit
        while (true) {
            // Display the menu options to the user
            System.out.println("\nPress 1 to recursively list a directory," + "\nAny other key to exit");
            // Read the user's choice
            String userAction = scanner.nextLine();
            // Option 1: Recursively list a directory
            if (userAction.equals("1")) {
                // Prompt the user to enter the name of the directory with the path
                System.out.println("Enter the name of the directory with the path");
                // Read the directory name with path
                String dirName = scanner.nextLine();
                // Create a File object representing the directory
                File dirObj = new File(dirName);
                // Check if the directory exists and is a valid directory
                if (dirObj.exists() && dirObj.isDirectory()) {
                    // Recursively list the contents of the directory
                    directoryList(dirObj);
                } else {
                    // If the directory does not exist, or is not a valid directory
                    System.out.println(dirName + " is not a valid directory");
                    continue;
                }
            }
            // Exit the program
            else {
                System.out.println("Bye!");
                break;
            }
        }
    }
}
```

Here's a breakdown of the what the code does:

1. If the user chooses option 1, the program prompts the user to enter the name of the directory with the path.
2. The program reads the directory name with path using a Scanner object.
3. The program creates a File object representing the directory using the entered path.
4. The program checks if the directory exists and is a valid directory using the exists() and isDirectory() methods. If it's not a valid directory, the program prints an error message and continues to the next iteration of the loop.
5. If the directory is valid, the program calls the directoryList() method to recursively list the contents of the directory.

6. Compile the Java program, specifying the destination directory using the `classes` directory that you previously created.

```
javac -d classes src/DirectoryRecursiveList.java
```

4. Now, run the Java program.

```
java DirectoryRecursiveList
```

Here is a sample output of the program.

```
Press 1 to recursively list a directory,
Any other key to exit
1
Enter the name of the directory with the path
/home/project/my_dir_proj
Printing /home/project/my_dir_proj recursively
Printing /home/project/my_dir_proj/test recursively
Printing /home/project/my_dir_proj/src recursively
/home/project/my_dir_proj/src/DirectoryExplorer.java
/home/project/my_dir_proj/src/DirectoryRecursiveList.java
Printing /home/project/my_dir_proj/classes recursively
/home/project/my_dir_proj/classes/DirectoryRecursiveList.class
Press 1 to recursively list a directory,
Any other key to exit
1
Enter the name of the directory with the path
/home/project/my_dir_proj/lav
/home/project/my_dir_proj/lav is not a valid directory
Press 1 to recursively list a directory,
Any other key to exit
```

Practice Exercise

Exercise 1

1. Add the recursive directory listing to the original menu options in `DirectoryExplorer.java`, in the `directoryManagement` method as option 4.
   ▼ Click here for solution

```
import java.io.File;
import java.util.Scanner;
import java.io.IOException;
public class DirectoryExplorer {
    // Method to handle file management operations
    private static void fileManagement(File file) {
        // Display options to the user
        System.out.println("\nPress 1 to rename the file," +
                "\nPress 2 to delete the file," +
                "\nAny other key to exit");
        Scanner scanner = new Scanner(System.in);
        String userChoice = scanner.nextLine();
        // Handle user choice
        if (userChoice.equals("1")) {
            // Get new name for the file
            System.out.println("Enter the new name for the file " + file.getName());
            String newfileName = scanner.nextLine();
            // Attempt to rename the file
            boolean changed = file.renameTo(new File(file.getParent(), newfileName));
            //Print message about success or failur of file renaming
            if (changed) {
                System.out.println("Filename successfully changed");
            } else {
                System.out.println("Filename couldn't be changed!");
            }
        } else if (userChoice.equals("2")) {
            // Delete the file
            boolean deleted = file.delete();
            //Print message about success or failur of file deletion
            if (deleted) {
                System.out.println("Filename successfully deleted");
            } else {
                System.out.println("Filename couldn't be deleted!");
            }
        }
    }
    private static void directoryList(File dirObj) {
        // Get a list of files and subdirectories in the directory
        File files[] = dirObj.listFiles();
        // Iterate over the list of files and subdirectories
        for (int i = 0; i < files.length; i++) {
            // Check if the current item is a file
            if (files[i].isFile()) {
                // Print the absolute path of the file
                System.out.println(files[i].getAbsolutePath());
            } else {
                // If the current item is a subdirectory, recursively list its contents
                directoryList(files[i]);
            }
        }
    }
    // Method to handle directory management operations (list, rename, or delete)
    private static void directoryManagement(File dirObj) {
        // Display options to the user
        System.out.println("\nPress 1 to list the directory," +
                "\nPress 2 to rename the directory," +
                "\nPress 3 to delete the directory," +
                "\nPress 4 to recursively list the directory," +
                "\nAny other key to exit");
        Scanner scanner = new Scanner(System.in);
        String userChoice = scanner.nextLine();
        // Handle user choice
        if (userChoice.equals("1")) {
            // List the contents of the directory
            String fileNames[] = dirObj.list();
            if (fileNames.length == 0) {
                System.out.println("The directory is empty!");
            } else {
                for (int i = 0; i < fileNames.length; i++) {
                    System.out.println(fileNames[i]);
                }
            }
```

```
    } else if (userChoice.equals("2")) {
        // Rename the directory
        System.out.println("Enter the new name for the directory " + dirObj.getName());
        String newDirName = scanner.nextLine();
        // Attempt to rename the directory
        boolean changed = dirObj.renameTo(new File(dirObj.getParent(), newDirName));
        //Print message about success or failure of dir renaming
        if (changed) {
            System.out.println("Directory name successfully changed");
        } else {
            System.out.println("Directory name couldn't be changed!");
        }
    } else if (userChoice.equals("3")) {
        // Delete the directory
        boolean deleted = dirObj.delete();
        //Print message about success or failure of dir deletion
        if (deleted) {
            System.out.println("Directory successfully deleted");
        } else {
            System.out.println("Directory couldn't be deleted! It might not be empty");
        }
    } else if (userChoice.equals("4")) {
            // Recursively list the contents of the directory
            directoryList(dirObj);
    }
}
public static void main(String s[]) {
    Scanner scanner = new Scanner(System.in);
    // Loop indefinitely until the user chooses to exit
    while (true) {
        // Display the menu options to the user
        System.out.println("\nPress 1 for File Management," + "\nAny other key to exit");
        // Read the user's choice
        String userAction = scanner.nextLine();
        // Option 1: File Management
        if (userAction.equals("1")) {
            // Prompt the user to enter the name of the file or directory with the path
            System.out.println("Enter the name of the file or directory with the path");
            // Read the file name with path
            String fileName = scanner.nextLine();
            // Create a File object representing the file or directory
            File file = new File(fileName);
            // Check if the file or directory exists
            if (file.exists()) {
                // Check if the file is a file or directory
                if (file.isFile()) {
                    System.out.println(fileName + " is a file");
                    // Call the fileManagement method for file operations
                    fileManagement(file);
                } else {
                    System.out.println(fileName + " is a directory");
                    // Call the directoryManagement method for directory operations
                    directoryManagement(file);
                }
            } else {
                // If the file or directory does not exist, prompt the user to create it
                System.out.println(fileName + " is not a valid file or directory");
                System.out.println("To create a file with given name press 1\n"
                        + "To create a directory with given name press 2\n"
                        + "To do nothing and continue, press any other key");
                // Read the user's choice
                String createChoice = scanner.nextLine();
                // Create a file
                if (createChoice.equals("1")) {
                    // Get the parent directory
                    String parentDirStr = file.getParent();
                    File parentDir = new File(parentDirStr);
                    // Create the parent directory if it does not exist
                    if (!parentDir.exists()) {
                        boolean created = parentDir.mkdirs();
                        if (!created) {
                            System.out.println("The parent directory could not be created");
                            continue;
                        }
                    }
                    // Create the file
                    try {
```

```
                        file.createNewFile();
                        System.out.println("File successfully created!");
                    } catch (IOException ioe) {
                        System.out.println("Unable to create file. " + ioe.getMessage());
                    }
                }
                // Create a directory
                else if (createChoice.equals("2")) {
                    // Create the directory
                    boolean created = file.mkdirs();
                    if (created) {
                        System.out.println("The directory has been created");
                    } else {
                        System.out.println("The directory couldn't be created");
                    }
                }
            }
            // Exit the program
            else {
                System.out.println("Bye!");
                break;
            }
        }
    }
}
```

Conclusion

In this lab, you learned:

- how to use Java's file-handling classes, `java.io.File` for files and directories.
- hot to list all the files and subdirectories in a directory
- how to create new directories and files
- how to delete existing directories and files
- how to rename files or directories
- how to navigate through a directory structure

Author(s)

Lavanya