1. Create and Write to a File: Write a Java program that prompts the user for a diary entry, then creates a file named "diary.txt" and writes the current date followed by the user's entry into this file. Ensure the program checks if the file already exists and informs the user, to avoid overwriting any previous content.

2. Read from a File: Write a Java application that opens the "diary.txt" file created in the previous question and displays its content on the console. The program should handle cases where the file does not exist by displaying an appropriate error message.

3. Append Content to an Existing File: Write a Java program that adds a new diary entry to the "diary.txt" file without overwriting its existing content. The program should ask the user for the new entry and append it to the file along with a timestamp.

4. List Files and Directories: Write a program in Java that asks the user for a directory path and then lists all files and subdirectories in that directory. If the directory does not exist, the program should inform the user.

5. Filter and List Specific File Types: Create a Java application that lists all the ".txt" files in a given directory. The program should prompt the user for the directory path and then display a list of all text files found in that directory.

6. Delete a Specific File: Write a Java program where the user can enter the name of a file to be deleted from the system. The program should check if the file exists and delete it, providing a confirmation message upon successful deletion or an error message if the file does not exist.

7. Copy File Content: Write a Java program that copies the content from one file (source) to another (destination). The program should prompt the user for both source and destination file paths and perform the copy operation, ensuring that it doesn't overwrite an existing file without user confirmation.

8. Rename a File: Develop a Java application that renames a specified file. The program should request the current file name and the new file name from the user, renaming the file accordingly and confirming the action upon completion.

9. Display File Metadata: Create a Java program that displays metadata of a specified file. The user should be able to input the file name, and the program should output the file size, last modified date, and other available attributes.

10. Recursive Directory Listing: Write a Java program that recursively lists all files and subdirectories within a given directory. The program should prompt the user for the directory path and then display a structured list of all contents, including files and directories nested within any subdirectories.

Solution: I have combined the above ten functionalities into one comprehensive program. This program has a menu-driven interface that allows the user to perform various file operations.

```
import java.io.*;
import java.nio.file.*;
import java.util.Scanner;
import java.util.Date;

public class FileHandlingDemo {
    private static final Scanner scanner = new Scanner(System.in);
    private static final String DIARY_FILE = "diary.txt";

    public static void main(String[] args) {
        while (true) {
            System.out.println("\nFile Handling Operations:");
            System.out.println("1. Create and Write to a File");
            System.out.println("2. Read from a File");
            System.out.println("3. Append Content to an Existing File");
            System.out.println("4. List Files and Directories");
            System.out.println("5. Filter and List Specific File Types");
            System.out.println("6. Delete a Specific File");
            System.out.println("7. Copy File Content");
            System.out.println("8. Rename a File");
            System.out.println("9. Display File Metadata");
            System.out.println("10. Recursive Directory Listing");
            System.out.println("11. Exit");
            System.out.print("Choose an operation: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume the newline

            switch (choice) {
                case 1:
                    createAndWriteFile();
                    break;
                case 2:
                    readFile();
                    break;
                case 3:
                    appendToFile();
                    break;
                case 4:
                    listFilesAndDirectories();
                    break;
                case 5:
                    filterAndListTxtFiles();
                    break;
                case 6:
                    deleteSpecificFile();
                    break;
                case 7:
```

```
                copyFileContent();
                break;
            case 8:
                renameFile();
                break;
            case 9:
                displayFileMetadata();
                break;
            case 10:
                recursiveDirectoryListing();
                break;
            case 11:
                System.exit(0);
            default:
                System.out.println("Invalid choice, please choose again.");
        }
    }
}

// Method to create a new file and write to it
private static void createAndWriteFile() {
    try {
        Path path = Paths.get(DIARY_FILE);
        // Check if file already exists
        if (Files.exists(path)) {
            System.out.println("File already exists. Content will be overwritten.");
        } else {
            System.out.println("Creating a new file.");
        }

        System.out.print("Enter your diary entry: ");
        String entry = scanner.nextLine();
        String content = new Date() + "\n" + entry;

        // Write the content to the file
        Files.write(path, content.getBytes());
        System.out.println("Entry saved successfully.");
    } catch (IOException e) {
        System.out.println("Error writing to file: " + e.getMessage());
    }
}

// Method to read from a file and display its contents
private static void readFile() {
    try {
        Path path = Paths.get(DIARY_FILE);
        // Check if the file exists before reading
        if (!Files.exists(path)) {
```

```
            System.out.println("File does not exist.");
            return;
        }

        System.out.println("Diary entries:");
        // Read and print each line from the file
        Files.lines(path).forEach(System.out::println);
    } catch (IOException e) {
        System.out.println("Error reading file: " + e.getMessage());
    }
}

// Method to append content to an existing file
private static void appendToFile() {
    try {
        Path path = Paths.get(DIARY_FILE);
        // Check if the file exists to append, else create it
        if (!Files.exists(path)) {
            System.out.println("File does not exist, creating a new file.");
            createAndWriteFile();
            return;
        }

        System.out.print("Enter your diary entry to append: ");
        String entry = scanner.nextLine();
        String content = "\n" + new Date() + "\n" + entry;

        // Append the new content to the existing file
        Files.write(path, content.getBytes(), StandardOpenOption.APPEND);
        System.out.println("Entry appended successfully.");
    } catch (IOException e) {
        System.out.println("Error appending to file: " + e.getMessage());
    }
}

// Method to list all files and directories in a given path
private static void listFilesAndDirectories() {
    System.out.print("Enter the directory path: ");
    String directoryPath = scanner.nextLine();

    File directory = new File(directoryPath);
    // Validate the directory path
    if (!directory.exists() || !directory.isDirectory()) {
        System.out.println("Directory does not exist or is not a directory.");
        return;
    }

    System.out.println("Listing files and directories:");
```

```
    // List and print each entry in the directory
    String[] fileList = directory.list();
    if (fileList != null) {
        for (String file : fileList) {
            System.out.println(file);
        }
    }
}

// Method to filter and list .txt files in a directory
private static void filterAndListTxtFiles() {
    System.out.print("Enter the directory path to filter .txt files: ");
    String directoryPath = scanner.nextLine();

    File directory = new File(directoryPath);
    // Check if the directory exists
    if (!directory.exists() || !directory.isDirectory()) {
        System.out.println("Directory does not exist or is not a directory.");
        return;
    }

    System.out.println("Listing .txt files:");
    File[] files = directory.listFiles((dir, name) -> name.endsWith(".txt"));
    if (files != null) {
        for (File file : files) {
            System.out.println(file.getName());
        }
    }
}

// Method to delete a specific file given by the user
private static void deleteSpecificFile() {
    System.out.print("Enter the file name to delete: ");
    String fileName = scanner.nextLine();

    File file = new File(fileName);
    // Check if the file exists before deletion
    if (!file.exists()) {
        System.out.println("File does not exist.");
        return;
    }

    if (file.delete()) {
        System.out.println("File deleted successfully.");
    } else {
        System.out.println("File could not be deleted.");
    }
}
```

```
// Method to copy content from one file to another
private static void copyFileContent() {
    System.out.print("Enter source file path: ");
    String sourcePath = scanner.nextLine();
    System.out.print("Enter destination file path: ");
    String destPath = scanner.nextLine();

    Path source = Paths.get(sourcePath);
    Path destination = Paths.get(destPath);

    try {
        // Confirm before overwriting an existing file
        if (Files.exists(destination)) {
            System.out.print("Destination file already exists. Overwrite? (y/n): ");
            String response = scanner.nextLine();
            if (!response.equalsIgnoreCase("y")) {
                System.out.println("Copy operation cancelled.");
                return;
            }
        }

        // Perform the copy operation
        Files.copy(source, destination, StandardCopyOption.REPLACE_EXISTING);
        System.out.println("File copied successfully.");
    } catch (IOException e) {
        System.out.println("Error copying file: " + e.getMessage());
    }
}

// Method to rename a file specified by the user
private static void renameFile() {
    System.out.print("Enter current file name: ");
    String currentName = scanner.nextLine();
    System.out.print("Enter new file name: ");
    String newName = scanner.nextLine();

    File file = new File(currentName);
    File newFile = new File(newName);

    // Check if the file exists before renaming
    if (!file.exists()) {
        System.out.println("File does not exist.");
        return;
    }

    if (file.renameTo(newFile)) {
        System.out.println("File renamed successfully.");
```

```
        } else {
            System.out.println("File could not be renamed.");
        }
    }

    // Method to display metadata of a file
    private static void displayFileMetadata() {
        System.out.print("Enter file name: ");
        String fileName = scanner.nextLine();

        File file = new File(fileName);
        // Check if the file exists before displaying metadata
        if (!file.exists()) {
            System.out.println("File does not exist.");
            return;
        }

        // Display file metadata
        System.out.println("File Metadata:");
        System.out.println("Size: " + file.length() + " bytes");
        System.out.println("Last Modified: " + new Date(file.lastModified()));
    }

    // Method to recursively list all files and subdirectories in a given directory
    private static void recursiveDirectoryListing() {
        System.out.print("Enter the directory path: ");
        String directoryPath = scanner.nextLine();

        File directory = new File(directoryPath);
        if (!directory.exists() || !directory.isDirectory()) {
            System.out.println("Directory does not exist or is not a directory.");
            return;
        }

        System.out.println("Recursive Directory Listing:");
        listDirectory(directory, 0);
    }

    // Helper method to recursively list directory contents
    private static void listDirectory(File dir, int level) {
        if (!dir.isDirectory()) {
            return;
        }

        File[] files = dir.listFiles();
        if (files != null) {
            for (File file : files) {
                for (int i = 0; i < level; i++) {
```

```
                System.out.print("  ");
            }
            System.out.println(file.getName());
            if (file.isDirectory()) {
                listDirectory(file, level + 1);
            }
        }
    }
  }
}
```