**LockedMeMain**

**package com;**

**public class LockedMeMain {**

**public static void main(String[] args) {**

**// Create "main" folder if not present in current folder structure**

**FileOperations.createMainFolderIfNotPresent("main");**

**MenuOptions.printWelcomeScreen("LockedMe", "Pertunia Mosana");**

**HandleOptions.handleWelcomeScreenInput();**

}

}

**MenuOptions**

**package com;**

**public class MenuOptions {**

**public static void printWelcomeScreen(String appName, String developerName) {**

**String companyDetails = String.format("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**

**+ "\*\* Welcome to %s.com. \n" + "\*\* This application was developed by %s.\n"**

**+ "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n", appName, developerName);**

**String appFunction = "You can use this application to :-\n"**

**+ "� Retrieve all file names in the \"main\" folder\n"**

**+ "� Search, add, or delete files in \"main\" folder.\n"**

**+ "\n\*\*Please be careful to ensure the correct filename is provided for searching or deleting files.\*\*\n";**

**System.out.println(companyDetails);**

**System.out.println(appFunction);**

**}**

**public static void displayMenu() {**

**String menu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"**

**+ "1) Retrieve all files inside \"main\" folder\n"**

**+ "2) Display menu for File operations\n"**

**+ "3) Exit program\n";**

**System.out.println(menu);**

**}**

**public static void displayFileMenuOptions() {**

**String fileMenu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"**

**+ "1) Add a file to \"main\" folder\n"**

**+ "2) Delete a file from \"main\" folder\n"**

**+ "3) Search for a file from \"main\" folder\n" + "4) Show Previous Menu\n" + "5) Exit program\n";**

**System.out.println(fileMenu);**

**}**

**}**

**HandleOptions**

**package com;**

**import java.util.List;**

**import java.util.Scanner;**

**public class HandleOptions {**

**public static void handleWelcomeScreenInput() {**

**boolean running = true;**

**Scanner sc = new Scanner(System.in);**

**do {**

**try {**

**MenuOptions.displayMenu();**

**int input = sc.nextInt();**

**switch (input) {**

**case 1:**

**FileOperations.displayAllFiles("main");**

**break;**

**case 2:**

**HandleOptions.handleFileMenuOptions();**

**break;**

**case 3:**

**System.out.println("Program exited successfully.");**

**running = false;**

**sc.close();**

**System.exit(0);**

**break;**

**default:**

**System.out.println("Please select a valid option from above.");**

**}**

**} catch (Exception e) {**

**System.out.println(e.getClass().getName());**

**handleWelcomeScreenInput();**

**}**

**} while (running == true);**

**}**

**public static void handleFileMenuOptions() {**

**boolean running = true;**

**Scanner sc = new Scanner(System.in);**

**do {**

**try {**

**MenuOptions.displayFileMenuOptions();**

**FileOperations.createMainFolderIfNotPresent("main");**

**int input = sc.nextInt();**

**switch (input) {**

**case 1:**

**// File Add**

**System.out.println("Enter the name of the file to be added to the \"main\" folder");**

**String fileToAdd = sc.next();**

**FileOperations.createFile(fileToAdd, sc);**

**break;**

**case 2:**

**// File/Folder delete**

**System.out.println("Enter the name of the file to be deleted from \"main\" folder");**

**String fileToDelete = sc.next();**

**FileOperations.createMainFolderIfNotPresent("main");**

**List<String> filesToDelete = FileOperations.displayFileLocations(fileToDelete, "main");**

**String deletionPrompt = "\nSelect index of which file to delete?"**

**+ "\n(Enter 0 if you want to delete all elements)";**

**System.out.println(deletionPrompt);**

**int idx = sc.nextInt();**

**if (idx != 0) {**

**FileOperations.deleteFileRecursively(filesToDelete.get(idx - 1));**

**} else {**

**// If idx == 0, delete all files displayed for the name**

**for (String path : filesToDelete) {**

**FileOperations.deleteFileRecursively(path);**

**}**

**}**

**break;**

**case 3:**

**// File/Folder Search**

**System.out.println("Enter the name of the file to be searched from \"main\" folder");**

**String fileName = sc.next();**

**FileOperations.createMainFolderIfNotPresent("main");**

**FileOperations.displayFileLocations(fileName, "main");**

**break;**

**case 4:**

**// Go to Previous menu**

**return;**

**case 5:**

**// Exit**

**System.out.println("Program exited successfully.");**

**running = false;**

**sc.close();**

**System.exit(0);**

**default:**

**System.out.println("Please select a valid option from above.");**

**}**

**} catch (Exception e) {**

**System.out.println(e.getClass().getName());**

**handleFileMenuOptions();**

**}**

**} while (running == true);**

**}**

**}**

**FileOperation**

**package com;**

**import java.io.File;**

**import java.io.IOException;**

**import java.nio.file.Files;**

**import java.nio.file.Path;**

**import java.nio.file.Paths;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.List;**

**import java.util.Scanner;**

**import java.util.stream.Collectors;**

**import java.util.stream.IntStream;**

**public class FileOperations {**

**public static void createMainFolderIfNotPresent(String folderName) {**

**File file = new File(folderName);**

**// If file doesn't exist, create the main folder**

**if (!file.exists()) {**

**file.mkdirs();**

**}**

**}**

**public static void displayAllFiles(String path) {**

**FileOperations.createMainFolderIfNotPresent("main");**

**// All required files and folders inside "main" folder relative to current**

**// folder**

**System.out.println("Displaying all files with directory structure in ascending order\n");**

**// listFilesInDirectory displays files along with folder structure**

**List<String> filesListNames = FileOperations.listFilesInDirectory(path, 0, new ArrayList<String>());**

**System.out.println("Displaying all files in ascending order\n");**

**Collections.sort(filesListNames);**

**filesListNames.stream().forEach(System.out::println);**

**}**

**public static List<String> listFilesInDirectory(String path, int indentationCount, List<String> fileListNames) {**

**File dir = new File(path);**

**File[] files = dir.listFiles();**

**List<File> filesList = Arrays.asList(files);**

**Collections.sort(filesList);**

**if (files != null && files.length > 0) {**

**for (File file : filesList) {**

**System.out.print (" .repeat(indentationCount \* 2");**

**if (file.isDirectory()) {**

**System.out.println("`-- " + file.getName());**

**// Recursively indent and display the files**

**fileListNames.add(file.getName());**

**listFilesInDirectory(file.getAbsolutePath(), indentationCount + 1, fileListNames);**

**} else {**

**System.out.println("|-- " + file.getName());**

**fileListNames.add(file.getName());**

**}**

**}**

**} else {**

**System.out.print(" .repeat(indentationCount \* 2");**

**System.out.println("|-- Empty Directory");**

**}**

**System.out.println();**

**return fileListNames;**

**}**

**public static void createFile(String fileToAdd, Scanner sc) {**

**FileOperations.createMainFolderIfNotPresent("main");**

**Path pathToFile = Paths.get("./main/" + fileToAdd);**

**try {**

**Files.createDirectories(pathToFile.getParent());**

**Files.createFile(pathToFile);**

**System.out.println(fileToAdd + " created successfully");**

**System.out.println("Would you like to add some content to the file? (Y/N)");**

**String choice = sc.next().toLowerCase();**

**sc.nextLine();**

**if (choice.equals("y")) {**

**System.out.println("\n\nInput content and press enter\n");**

**String content = sc.nextLine();**

**Files.write(pathToFile, content.getBytes());**

**System.out.println("\nContent written to file " + fileToAdd);**

**System.out.println("Content can be read using Notepad or Notepad++");**

**}**

**} catch (IOException e) {**

**System.out.println("Failed to create file " + fileToAdd);**

**System.out.println(e.getClass().getName());**

**}**

**}**

**public static List<String> displayFileLocations(String fileName, String path) {**

**List<String> fileListNames = new ArrayList<>();**

**FileOperations.searchFileRecursively(path, fileName, fileListNames);**

**if (fileListNames.isEmpty()) {**

**System.out.println("\n\n\*\*\*\*\* Couldn't find any file with given file name \"" + fileName + "\" \*\*\*\*\*\n\n");**

**} else {**

**System.out.println("\n\nFound file at below location(s):");**

**List<String> files = IntStream.range(0, fileListNames.size())**

**.mapToObj(index -> (index + 1) + ": " + fileListNames.get(index)).collect(Collectors.toList());**

**files.forEach(System.out::println);**

**}**

**return fileListNames;**

**}**

**public static void searchFileRecursively(String path, String fileName, List<String> fileListNames) {**

**File dir = new File(path);**

**File[] files = dir.listFiles();**

**List<File> filesList = Arrays.asList(files);**

**if (files != null && files.length > 0) {**

**for (File file : filesList) {**

**if (file.getName().startsWith(fileName)) {**

**fileListNames.add(file.getAbsolutePath());**

**}**

**// Need to search in directories separately to ensure all files of required**

**// fileName are searched**

**if (file.isDirectory()) {**

**searchFileRecursively(file.getAbsolutePath(), fileName, fileListNames);**

**}**

**}**

**}**

**}**

**public static void deleteFileRecursively(String path) {**

**File currFile = new File(path);**

**File[] files = currFile.listFiles();**

**if (files != null && files.length > 0) {**

**for (File file : files) {**

**String fileName = file.getName() + " at " + file.getParent();**

**if (file.isDirectory()) {**

**deleteFileRecursively(file.getAbsolutePath());**

**}**

**if (file.delete()) {**

**System.out.println(fileName + " deleted successfully");**

**} else {**

**System.out.println("Failed to delete " + fileName);**

**}**

**}**

**}**

**String currFileName = currFile.getName() + " at " + currFile.getParent();**

**if (currFile.delete()) {**

**System.out.println(currFileName + " deleted successfully");**

**} else {**

**System.out.println("Failed to delete " + currFileName);**

**}**

**}**

**}**