package com.lockedme;

import java.io.File;

import java.io.FileNotFoundException;

import java.nio.file.InvalidPathException;

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;

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class FileOperations {

public static String folderName;

public static String path;

//folder related

//public static String getFolderName() {

// Scanner sc1 = new Scanner(System.in);

// String folderName = sc1.nextLine();

// sc1 = new Scanner(System.in);

// return folderName;

// }

public static String getFolderName() throws IOException{

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

String folderName = br.readLine();

return folderName;

}

public static void createFolderIfNotPresent(String folderName) {

File folder = new File(folderName);

try{// If the Folder doesn't exist

folder.mkdirs();

System.out.println("Folder created successfully " + folderName);

HandleFileOptions.handleWelcomeScreenInput(folderName);

} catch (Exception e) {

System.out.println("Failed to create folder " + folderName);

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

}

}

//file related

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());

//The Recursively indent and displaying all 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 folderName,String fileToAdd,Scanner sc) throws InvalidPathException, IOException {

Path pathToFile = Paths.get(folderName + fileToAdd);

try {

Files.createDirectories(pathToFile.getParent());

Files.createFile(pathToFile);

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

writeFile(folderName,fileToAdd,sc);

} catch(InvalidPathException e){

throw new InvalidPathException(folderName+fileToAdd,"Invalid path provided");

}catch(IOException e) {

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

}}

public static void writeFile(String folderName,String fileToAdd,Scanner sc) {

Path pathToFile = Paths.get(folderName + fileToAdd);

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

sc.nextLine();

String choice = sc.nextLine().toLowerCase();

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

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

try {

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

String content = br.readLine();

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

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

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

} catch (IOException e) {

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\*\*\*\*\* Could not find any file with the given file name \"" + fileName + "\" \*\*\*\*\*\n"+

"Note: Filename is case sensitive.");

} else {

System.out.println("\n\nFound file at the 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());

}

// the need to search in the directories separately to ensuring all the files of required file name is been searched.

if (file.isDirectory()) {

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

}

}

}

}

public static void displayAllFiles(String path) {

// All The required files and folders inside user specified folder

System.out.println("Display all the files with directory structure in the Ascending order\n");

System.out.println(path);

// List the Files in the Directory displays files along with the Folder structure

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

System.out.println("Display all the files in Ascending Order\n");

Collections.sort(filesListNames);

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

}

public static void deleteFileRecursively(String folderName,String fileToDelete) throws FileNotFoundException {

File currFile = new File(folderName + fileToDelete);

File[] files = currFile.listFiles();

try {

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

for (File file : files) {

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

if (file.isDirectory()) {

deleteFileRecursively(folderName,fileToDelete);

}

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);

}}catch(FileNotFoundException e){

throw new FileNotFoundException("Missing file-"+folderName+fileToDelete);

}

}

}

package com.lockedme;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.List;

import java.util.Scanner;

public class HandleFileOptions extends FileOperations{

public static void handleWelcomeScreenInput(String folderName) {

boolean running = true;

Scanner sc = new Scanner(System.in);

do {

try {

MenuOptions.displayFileMenu();

int input = sc.nextInt();

switch (input) {

case 1:

//display files in folder

FileOperations.displayAllFiles(folderName);

break;

case 2:

//go to add or delete or search file menu option

HandleFileOptions.handleFileMenuOptions(folderName);

break;

case 3:

//specify folder location again

return;

case 4:

//exit program

System.out.println("\nThe program is been exited successfully.\n");

running = false;

sc.close();

System.exit(0);

break;

default:

System.out.println("\nPlease select the appropriate option from the above.\n");

}

} catch (Exception e) {

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

handleWelcomeScreenInput(folderName);

}

} while (running == true);

}

public static void handleFileMenuOptions(String folderName) {

boolean running = true;

Scanner sc = new Scanner(System.in);

do {

try {

MenuOptions.displayFileMenuOptions();

int input = sc.nextInt();

switch (input) {

case 1:

// file add to the main folder.

System.out.println("\nEnter the name of the file to be added to:\n"+folderName+

"\nEg.abc.txt\n");

String fileToAdd = sc.next();

Path tempPath=Paths.get(folderName+fileToAdd);

if(Files.exists(tempPath)) {

System.out.println("File already exist.");

FileOperations.writeFile(folderName,fileToAdd,sc);

}else {

FileOperations.createFile(folderName,fileToAdd,sc);

}

break;

case 2:

// File to be deleted from folder.

FileOperations.displayAllFiles(folderName);

System.out.println("\nAbove is a list of available files to delete from:\n"+folderName

+ "\nWith reference to above list, Enter the name of the file to deleted\n" );

String fileToDelete= sc.next();

FileOperations.deleteFileRecursively(folderName,fileToDelete);

break;

case 3:

// File to be searched from folder.

System.out.println("\nEnter the name of the file to be searched from \n"+folderName+"\n");

String fileName = sc.next();

FileOperations.displayFileLocations(fileName, folderName);

break;

case 4:

// Go to Previous menu

return;

case 5:

// Exit

System.out.println("\nThe program is been exited successfully.\n");

running = false;

sc.close();

System.exit(0);

default:

System.out.println("\nPlease select a valid option from above.\n");

}

} catch (Exception e) {

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

handleFileMenuOptions(folderName);

}

} while (running == true);

}}

package com.lockedme;

import java.io.BufferedReader;

import java.io.File;

import java.util.List;

import java.util.Scanner;

public class HandleFolderOptions extends FileOperations{

public static void handleFolderInput() {

boolean running = true;

do {

try {

MenuOptions.displayFolderMenu();

// user specified folder.

System.out.println("\nEnter a valid folder path:"+

"\nEg. D\\User\\Documents\\");

String folderName = getFolderName();

if(folderName.endsWith("\\")) {

File file = new File(folderName);

// If folder doesn't exist, ask user if should create

if (!file.exists()) {

HandleFolderOptions.handleFolderMenuOptions(folderName);

} else {

HandleFileOptions.handleWelcomeScreenInput(folderName);

}

}else {System.out.println("\nFilepath should end with \\. Please try again");

handleFolderInput();

}

}

catch (Exception e) {

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

handleFolderInput();

}

} while (running == true);

}

public static void handleFolderMenuOptions(String folderName) {

boolean running = true;

Scanner sc = new Scanner(System.in);

do {

try {

MenuOptions.displayFolderMenuOptions();

int input = sc.nextInt();

switch (input) {

case 1:

// Specify folder path again

MenuOptions.displayFolderMenu();

break;

case 2:

// create folder

FileOperations.createFolderIfNotPresent(folderName);

break;

case 3:

// Exit

System.out.println("\nThe program is been exited successfully.");

running = false;

sc.close();

System.exit(0);

default:

System.out.println("\nPlease select a valid option from above.");

}

} catch (Exception e) {

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

handleFolderMenuOptions(folderName);

}

} while (running == true);

}

}

**package** com.lockedme;

**public** **class** LockedMeMain {

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

// Create "main" folder if it is not been present in current folder structure.

MenuOptions.*printWelcomeScreen*("LockedMe", "Lim Si Yi");

HandleFolderOptions.*handleFolderInput*();

}

}

**package** com.lockedme;

**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 user specified folder\n"

+ "• Search, add, or delete files in user specified folder.\n";

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

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

}

**public** **static** **void** displayFolderMenu() {

String foldermenu ="\n\*\*\*\*\*\* Please provide the path of folder to work on \*\*\*\*\*\*"

+"\n\*\*\*\*\*\*Reminder:Input \\ at the end of folder path\*\*\*\*\*\*";

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

}

**public** **static** **void** displayFolderMenuOptions() {

String folderOptions ="\n\*\*\*\*\*\* Folder specified does not exist. Select any option number from below and press Enter \*\*\*\*\*\*\n\n"

+ "\n1) Specify a different folder path" +"\n2) Create folder path"+"\n3) Exit program\n";

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

}

**public** **static** **void** displayFileMenu() {

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

+ "\n1) Retrieve all files inside folder" + "\n2) Display menu for File operations"

+ "\n3) Show Previous Menu"+"\n4) 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"

+ "\n1) Add a file to folder" + "\n2) Delete file in folder"

+ "\n3) Search for a file from folder" + "\n4) Show Previous Menu" + "\n5) Exit program\n";

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

}

}