**SOURCE CODE**

//VirtualRepository

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

import java.io.File;

import java.io.IOException;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

/\*\*

\* this file handling class handles the files by performing operations.

\* 1. Display files in ascending order.

\* 2. Operations

\* 2.1 Add file/folder.

\* 2.2 Delete file

\* 2.3 Search for a file.

\* At first we require a valid path of folder/directory on which you want to perform operations.

\* It repetitively ask for a path till path is not valid or folder is not present.

\*/

/\*\*

\* @author Nisha Rajeshwar Popshetwar

\* @version 1.0

\* @since30-05-2021

\*

\*/

public class VirtualRepository {

/\*

\* Scanner object to take inputs from user

\*/

public static Scanner s= new Scanner(System.in);

/\*

\* to store the path

\*/

public static String path;

/\*

\* instance of a file

\*/

public static File f;

public static void main(String[] args) throws IOException {

System.out.println(" "+"\*\*\*\*\*\*\*\*\*Welcome to LOCKERS PVT LTD\*\*\*\*\*\*\*\*\*\*");

System.out.println("");

System.out.println(" "+"------------project name is: LOCKEDME.COM------------");

System.out.println("");

System.out.println(" "+"=========Developed By: NISHA RAJESHWAR POPSHETWAR=========");

System.out.println("");

System.out.println("");

/\*

\* following set code displays current date and time

\*/

LocalDateTime myDateobj=LocalDateTime.now();

DateTimeFormatter myformat = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

String formattedDate = myDateobj.format(myformat);

System.out.println("Date: "+formattedDate);

System.out.println("------------------------------------------------------------------");

System.out.println("");

/\*

\* calls folderPath method.

\*/

folderPath();

}

static String typeOfFile(File i)

{

if(i.isDirectory())

{return "Folder";}

else if(i.isFile())

{

String[] name =i.getName().split("\\.");

return name[name.length-1];

}return "none";

}

static void printFile(List<File> file) throws IOException {

int j=0;

for (File i:file) {

System.out.println(++j+" "+i.getName()+" "+"at:"+" "+ i.getAbsolutePath());

}

System.out.println(" ");

}

static void printFile(File[] file) throws IOException {

int j=0;

for (File i:file) {

System.out.println(++j+" "+i.getName()+" "+"at:"+" "+ i.getAbsolutePath());

}

System.out.println(" ");

}

static String readStr()

{

String str = null;

try {

str=s.nextLine();

}

catch(Exception e)

{

System.out.println("invalid input");

}

return str;

}

static int options()

{ String option;

try

{

option=s.nextLine();

}

catch(Exception e)

{

return 0;

}

if(option.matches("[0-9]"))

{

return Integer.valueOf(option);

}

else

return 0;

}

/\*

\* to exit the program

\*/

public static void exit()

{

System.out.println("thank you for using our application..... ");

System.exit(0);

}

public static void folderPath() throws IOException

{

System.out.println("\tPlease Give Path of Folder to perform operations : ");

path=readStr();

f =new File(path);

while(!f.isDirectory())

{

System.out.println("\t!!! Please Give Path of valid Folder/Directory : ");

path=readStr();

f =new File(path);

}mainMenu();

}

/\*

\* displays following operations

\* 1.To display files in ascending order

\* 2.perform various operations on file

\* 3.to exit the program

\*/

public static void mainMenu() throws IOException

{

int option;

System.out.println(" "+"\*\*\*\*\*\*\*\*\*\*\*\*\*\*MainMenu\*\*\*\*\*\*\*\*\*\*\*\*\*");

do{

System.out.println("\t\t 1.Display files in ascending order");

System.out.println("\t\t 2.preform different oprations on file");

System.out.println("\t\t 3.Exit the program");

option = options();

try

{

if(option<1|| option>3)

{

System.out.println("invalid option. please choose valid option");

}

}

catch(Exception e)

{

System.out.println("please give valid input");

}

} while(option<1 || option>3);

switch(option) {

case 1:display();

case 2:fileOperations();

case 3:exit();

}

}

/\*

\* This method is used to perform following operations on file

\* 1.to add file/folder.

\* 2.to remove file/folder.

\* 3.search for a file/folder.

\*/

public static void fileOperations() throws IOException

{

int option;

System.out.println("--------File Opertions--------");

System.out.println("\t 1.Add a file");

System.out.println("\t 2.Delete a file");

System.out.println("\t 3.search for a file");

System.out.println("\t 4.to return to Main Menu");

System.out.println("");

System.out.println("\t please select any one option:");

option = options();

try

{

if (option<1 || option>4)

{

System.out.println("please select valid option");

fileOperations();

}

}

catch(Exception e)

{

System.out.println("please give valid input!!");

}

switch(option)

{

case 1:addFileDir();

case 2:deleteFile();

case 3:searchFile();

case 4:mainMenu();

}

}

/\*

\* for option 1 from fileOperations ,this method is executed. which performs following operations

\* 1. to add a new file

\* 2. to add a new folder

\*/

public static void addFileDir() throws IOException

{

int option;

do {

System.out.println("enter 1. to add a file");

System.out.println("enter 2. to add a folder");

option =options();

try

{

if(option<1||option>2) {

System.out.println("please choose valid options");

fileOperations();

}

}

catch(Exception e)

{

System.out.println("please enter valid option");

}

}while(option<1 || option>2);

if(option==1)

{

boolean b=addFile();

if(b)System.out.println("file created successfully");

else

System.out.println("file not created");

}

else if(option ==2)

{

boolean b1=addFolder();

if(b1)System.out.println("folder created successfully");

else System.out.println("unable create folder...");

}

fileOperations();

}

/\*

\* this method is used to create a new file.

\* user have to give a file name as a input to create a file.

\* if a file with a same name and extension as of given input then file already present and not created and returns false

\* if a file is created successfully the returns true.

\*/

public static boolean addFile()

{

System.out.println("please enter a name of a file with or without .extension");

String fileName = readStr();

File file = new File(path+"/"+fileName);

boolean created=false;

if(file.exists())System.out.println(file+" is already exists at:"+file.getAbsolutePath());

else

{

try

{

if(!file.exists()) { created= file.createNewFile(); }

}

catch(Exception e)

{

System.out.println("unable to create a file");

created=false;

}

}

return created;

}

/\*

\* this method is used to create a new folder.

\* user have to give a folder name as a input to create a folder.

\* if a folder with a same name as of given input then folder already present and not created and returns false

\* if a folder is created successfully the returns true.

\*/

public static boolean addFolder()

{

System.out.println("please enter a name of a folder you want to create");

String folderName = readStr();

File folder = new File(path+"/"+folderName);

boolean created=false;

try

{

if(!folder.exists())

{

created=folder.mkdir();

}

else if(folder.exists())

{

System.out.println(folder+"folder is already exists at."+folder.getAbsolutePath());

}

}catch(Exception e)

{

System.out.println(e);

System.out.println("unable to create a folder");

created=false;

}

return created;

}

/\*

\* To delete a file or folder it will ask for confirmation to delete file/folder

\* if a file/folder is not present or the folder is empty deletion operation can not perform.

\*/

public static void deleteFile()throws IOException

{

System.out.println("please enetr a file name to delete with .extension");

String file= readStr();

File del=new File(path+"/"+file);

boolean deleted = false;

try

{

if(del.exists())

{

System.out.println("Are you sure you want to delete "+file);

System.out.println("press Y for yes,Any letter for cancel");

String confirmation=s.nextLine();

if(confirmation.equalsIgnoreCase("Y")) {

deleted = del.delete();

}

else

{

System.out.println("canceling....");

}

}

}

catch(Exception e)

{

System.out.println("file can not deleted .some exception occure");

}

if (deleted)System.out.println("successfully deleted");

else if (!del.exists())System.out.println("There no such a file");

fileOperations();

}

/\*

\* This method is used to search a file/folder in the current folder

\* if the file is present , it will print path of the file.

\* this method is case sensitive

\*/

public static void searchFile() throws IOException

{

int j=0;

System.out.println("Enter File name to search");

String file = readStr();

String fileName= file;

File[] list= f.listFiles();

if(list.length==0 && f.isDirectory())

{

System.out.println("dierctory is Empty");

}

else if(!f.isDirectory())

{

System.out.println(f.getName()+"not a directory");

}

else if(f.isDirectory() && list.length>0)

{

List<File> foundFile = new ArrayList<>();

boolean found = false;

File searchFile= new File(path+"/"+fileName);

for(File i:list)

{

if(i.getName().matches(fileName+"[.][0-9|a-z|A-Z]\*")||i.getName().equals(fileName))

{

foundFile.add(i);

++j;

found= true;

}

}

if(found && j>0)

{

System.out.println("file "+fileName+" is found");

printFile(foundFile);}

else if(!found||j==0)

{

System.out.println("file not found");}

}

fileOperations();

}

/\*

\* Display the files in ascending order of current folder

\*/

public static void display()throws IOException

{

File[] list = f.listFiles();

if(!f.isDirectory()) System.out.println("it is not a folder");

else if(list.length==0 || list==null) System.out.println("directory is empty");

else {

sort1(list);

printFile(list);

}

mainMenu();

}

/\*

\* Selection sort algorithm

\* To sort the array by repeatedly finding minimum element from unsorted part and

\* put it at the beginning.

\* In every iteration of selection sort, the minimum element from unsorted sub array is picked ,

\* and moved to the sorted sub array.

\*/

/\*

\* @return list String[]

\*/

public static File[] sort1(File[]list)

{

for(int i=0;i<list.length-1;i++)

{

int index=i;

for(int j=i+1;j<list.length;j++)

{

if (list[j].getName().compareTo(list[index].getName()) <= 0)

{

index=j;

}

File smallNumber=list[index];

list[index]=list[i];

list[i]=smallNumber;

}

}

return list;

}

}