*import* java.io.File;  
*import* java.io.IOException;  
*import* java.io.PrintStream;  
*import* java.nio.file.Files;  
*import* java.util.\*;  
  
*/\*\*  
 \*This FileHandler class handles all file handling tasks like  
 \* <p>1) Display files in ascending order.</p>  
 \* <p>2) Operations</p>  
 \* <p> 2.1)Add File / Folder.</p>  
 \* <p> 2.2)Delete File.</p>  
 \* <p> 2.3)Search for a file.</p>  
 \*At first it requires a valid path of folder/directory on which you want to perform operations. Make sure the path is not system secure.  
 \*It will repetitively ask for a path till Path is not valid or Folder is not present.  
 \** ***@author*** *Vishnu  
 \** ***@version*** *1.0  
 \** ***@since*** *28-April-2021  
 \*/  
  
public class* FileHandler{  
*/\*\*  
 Scanner object to take input from user  
 \*/  
public static* Scanner *sc*= *new* Scanner(System.*in*);  
*/\*\* To store the path variable.\*/  
public static* String *path*;  
*/\*\* Instance of File Object\*/  
public static* File *f*;  
*/\*\* While Developing the program to identify and solve errors.\*/  
protected static booleandeveloper* = *false*;  
  
*/\*\*  
 \* Main Method to Execute The Program.  
 \** ***@throws*** *IOExceptionHad to implement because of Main Menu Method  
 \** ***@throws*** *InputMismatchExceptionHad to implement because of Main Menu Method.  
 \** ***@param args****String[] args  
 \*/  
public static void* main(String[] args) *throws* IOException,InputMismatchException{  
System.*out*.printf("%-25s\*\*\*\*\*\*\*\*\*\*\*\*\* Welcome to \"LOCKERS PVT. LTD.\" \*\*\*\*\*\*\*\*\*\*\*\*\*"," ");  
System.*out*.printf("\n%-10s~~~~~~~~~~~~~~~ Project Name is \"LOCKEDME.COM\"It's help you to handle files ~~~~~~~~~~~~~~~"," ");  
System.*out*.printf("\n%15s<============== This Project is developed by \" Vishnu \" ==============>\n"," ");  
*changeFolder*();  
 }  
  
*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;  
System.*out*.printf("| %-5s | %-100s | %-10s | %-10s \n","No.","FileName","Type","Path");  
*for* (File i:file) {  
System.*out*.printf("| %-5s | %-100s | %-20s | %-10s",++j,i.getName(),*typeOfFile*(i), i.getAbsolutePath());  
 }  
System.*out*.println(" ");  
 }  
*static void* printFile(File[] file) *throws* IOException{  
*int* j=0;  
System.*out*.printf("| %-5s | %-100s | %-20s | %-10s \n","No.","FileName","Type","Path");  
System.*out*.println("---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------");  
*for* (File i:file) {  
System.*out*.printf("| %-5s | %-100s | %-20s | %-10s", ++j, i.getName(), *typeOfFile*(i),i.getAbsolutePath());  
System.*out*.println(" ");  
 }  
 }  
*static* String stringreader() {  
String str = "";  
*try* {  
str = *sc*.nextLine();  
 } *catch* (Exception var3) {  
System.*out*.println("Invalid Input!");  
 }  
*return* str;  
 }  
*static int* integerChoice() {  
*boolean*valid = *false*;  
String choice;  
*try* {  
choice = *sc*.nextLine();  
 } *catch* (Exception e) {  
*return* 0;  
 }  
*if* (choice.matches("[0*-*9]")) {  
*return* Integer.*valueOf*(choice);  
 } *else* {  
*return* 0;  
 }  
 }  
*/\*\*To Exit The Program\*/  
public static void* exit() {  
System.*out*.printf("\n\n%-50sThank you for using our application\tQuitting..."," ");  
System.*exit*(0);  
 }  
  
*/\*\*  
 \* The Method takes input of Path From User. <b>If The Path is Invalid it will warn and repetitively ask till the Path is valid.</b>  
 \** ***@throws*** *IOExceptionFor invalid Path or invalid Input  
 \*/  
public static void* changeFolder() *throws* IOException,InputMismatchException{  
*if* (*developer*) System.*out*.println("Change Folder is Running...");  
System.*out*.print("\tPlease Give Path of Folder to perform operations : ");  
*path* = *stringreader*();  
*f* = *new* File(*path*);  
*while* (!*f*.isDirectory()) {  
System.*out*.print("\t!!! Please Give Path of valid Folder/Directory : ");  
*path* = *stringreader*();  
*f* = *new* File(*path*);  
 }*MainMenu*();  
 }  
  
*/\*\*  
 \* This Method is for Displaying Main Menu where user have 4 options  
 \* <p>1 )To Display Files in ascending order.</p>  
 \* <p>2 )Perform various operations on File.</p>  
 \* <p>3 )To change the current folder.</p>  
 \* <p>4 )to exit the Program.</p>  
 \** ***@throws*** *IOExceptionFor Invalid Input  
 \** ***@throws*** *InputMismatchExceptionFor alphabetic input.  
 \*/  
public static void* MainMenu() *throws* IOException, InputMismatchException{  
*int* choice;  
System.*out*.println("\n<<<<<<<<<<<<<<<< Main Menu >>>>>>>>>>>>>>>>>>\n");  
*do* {  
System.*out*.println("\t\t1. Display files in ascending order.");  
System.*out*.println("\t\t2. Perform operation on files.");  
System.*out*.println("\t\t3. Change Folder/Directory");  
System.*out*.println("\t\t4. Exit The Program.");  
System.*out*.print("\t\tPlease Select any option from above choices : ");  
choice = *integerChoice*();  
*try* {  
*if* (choice <1 || choice >4) System.*out*.println("\t!!! Invalid choice. Please Enter valid choice between 1 and 4.!!!");  
 }*catch* (InputMismatchExceptione) {  
System.*out*.println("\t!!! Please Give Valid Input!!!");  
 }  
 } *while* (choice <1 || choice >4);  
*switch* (choice) {  
*case* 1 ->*displayFiles*();  
*case* 2 ->*operationMenu*();  
*case* 3 ->*changeFolder*();  
*case* 4 ->*exit*();  
 }  
 }  
  
*/\*\*  
 \* This Function display Operation Menu where user would have following three operations to perform on a File.  
 \* <p>1 )To Add a File/Folder.</p>  
 \* <p>2 )To Remove a File/Folder.</p>  
 \* <p>3 )Search For a File.</p>  
 \** ***@throws*** *IOExceptionIf User give invalid Input or the File which does not exists.  
 \*/  
public static void* operationMenu() *throws* IOException,InputMismatchException{  
*int* choice;  
System.*out*.println("<<<<<<====== Operation Menu ======>>>>>>");  
System.*out*.println("\t1. Add a File.");  
System.*out*.println("\t2. Remove a File.");  
System.*out*.println("\t3. Search for a File.");  
System.*out*.println("\t4. To return to Main Menu");  
System.*out*.print("\t\tPlease select any choice from above options : ");  
choice = *integerChoice*();  
*try* {  
*if* (choice <1 || choice >4) {  
System.*out*.println("\t!!! Please Give valid Choice between 1 and 4 !!!");  
*operationMenu*();}  
 }*catch* (Exception e){  
System.*out*.println("\t!!! Please Give Valid Input !!!");  
 }  
*switch* (choice) {  
*case* 1 ->*addFileOrDir*();  
*case* 2 ->*deleteFile*();  
*case* 3 ->*searchFile*();  
*case* 4 ->*MainMenu*();  
 }  
 }  
  
*/\*\*  
 \* When user choose option 1 from Operation Menu, this method is executed. In this Menu User have 2 option  
 \* <p>1 )To make new File.</p>  
 \* <p>2 )To make new Folder.</p>  
 \** ***@throws*** *IOException<p>1 )For invalid Path or Invalid Input.</p><p>2 )If the Access to path is denied due to security reasons.</p>  
 \*/  
public static void* addFileOrDir() *throws* IOException,InputMismatchException{  
*int* choice = 0;  
*if* (*developer*) System.*out*.println("Add File or Dir running...");  
*do* {  
System.*out*.println("\t1. Press 1 to make a file\n\t2. Press 2 to make a folder/directory\n\tOr any other option to got o operation menu.");  
System.*out*.print("\tPlease specify do you want to make Directory or File : ");  
choice = *integerChoice*();  
*try*{  
*if* (choice <1||choice >2) {  
System.*out*.println("Redirecting to Operation Menu...");  
*operationMenu*();  
 }  
 }*catch* (Exception e){  
System.*out*.print("\t!!! Please Give Valid option as number !!!");  
 }  
 } *while* (choice <1 || choice >2);  
*if* (choice == 1) {  
*boolean*done = *addFile*();  
*if* (done) System.*out*.println("\t>>>> File has been created. >>>>\n");  
*else*System.*out*.println("\t !!! Unable to create file.");  
 }*else if* (choice == 2) {  
*boolean*done = *addFolder*();  
*if* (done) System.*out*.println("\t>>>> Folder has been created successfully >>>>>\n");  
*else*System.*out*.println("\t !!! Unable to create folder. Sorry for the inconvenience but please first check if any security issues are there.");  
 }  
*operationMenu*();  
 }  
  
*/\*\*  
 \* To create a new File. User have to give input as File name which he wants to create.<p><b><i>The Method is Case-Sensitive.</i></b></p>  
 \* If the Folder/File (with same extension) with given input is already present in folder, file would not be created and it will return False on unsuccessful.  
 \** ***@return*** *<b><i>Return True if the File is created SuccessFully else False.</i></b>  
 \*/  
public static boolean*addFile() {  
*if* (*developer*) System.*out*.println("Add File Running...");  
System.*out*.print("\t\tPlease Enter Name of the file with/without . dot extension. -> ");  
*//sc.nextLine();*String fileName= *stringreader*();  
File newFile= *new* File(*path* + "/" + fileName);  
*boolean*taskHappen= *false*;  
*if* (newFile.exists()) System.*out*.println("\t\t"+newFile+ " --> Already Exists at "+newFile.getAbsolutePath());  
*else*{  
*try* {  
*if* (!newFile.exists()) { taskHappen= newFile.createNewFile(); }  
 } *catch* (Exception e) {  
System.*out*.println("\t!!! Unable to create file due to some exceptions\n");  
taskHappen= *false*;  
 }  
 }  
*return* taskHappen;  
 }  
  
  
*/\*\*  
 \* To create a new Folder. User have to give input as Folder name which he wants to create.<p><b><i>The Method is Case-Sensitive.</i></b></p>  
 \* If the Folder/File (without extension) with same name is already present in folder, folder would not be created and it will return False on unsuccessful.  
 \** ***@return*** *<b><i>True if the Folder is created SuccessFully else False.</i></b>  
 \*/  
  
public static boolean*addFolder() {  
*if* (*developer*) System.*out*.println("Add Folder Running...");  
System.*out*.print("\tPlease Enter Name of Folder You Want ot create -> ");  
*//sc.nextLine();*String folderName= *stringreader*();  
File newFolder= *new* File(*path* + "/" + folderName);  
*boolean*taskHappen= *false*;  
*try* {  
*if* (!newFolder.exists()) {  
taskHappen= newFolder.mkdir();  
System.*out*.println(taskHappen);  
 } *else if* (newFolder.exists()) System.*out*.println("\t\t"+newFolder+ " Already Exists at --> "+newFolder.getAbsolutePath());  
 } *catch* (Exception e) {  
System.*out*.println("\tFollowing Exception Occurs while Making the folder\n");  
System.*out*.println(e);  
System.*out*.println("\tUnable to create Folder due to above exception\n");  
taskHappen= *false*;  
 }  
*return* taskHappen;  
 }  
  
*/\*\*  
 \* To Delete the user specified File/Folder. It will again ask for confirmation of Deleting the File/Folder by displaying the details of same.<p><b><i>The Method is Case-Sensitive.</i></b></p>  
 \** ***@throws*** *IOException<p>1 )If the specified File/Folder is not Present</p><p>2 )The Folder Have some security issues. </p><p>3) the specified Folder is not Empty.</p><p>4)Access is Denied.</p>  
 \*/  
public static void* deleteFile() *throws* IOException{  
*if* (*developer*) System.*out*.println("Delete File Running...");  
System.*out*.print("\tSpecify The File Name to delete with its .dot extension else program will cause problem --> ");  
String FileName= *stringreader*();  
File df = *new* File(*path* + "/" + FileName);  
*boolean*taskHappen= *false*;  
*try* {  
*if* (df.exists()) {  
System.*out*.println("\tAre you sure You want to delete " + FileName+ " file.\nFile Details -->");  
System.*out*.println("\t\t File Name : " + df.getName() + "\n\t\tSize of file : " + Files.*size*(df.toPath()));  
System.*out*.print("Press Y for yes, any other key to cancel : ");  
String sure = *sc*.nextLine();  
*if* (sure.equals("Y") || sure.equals("y")) {  
taskHappen= df.delete();  
 } *else* System.*out*.println("Cancelling");  
 }  
 } *catch* (Exception e) {  
System.*out*.println("\t!! Exception has occur while deleting " + FileName+ "\n");  
 }  
*if* (taskHappen) System.*out*.println("\t>>>>>> Successfully deleted the file. <<<<<<<");  
*else* {  
*if* (df.isDirectory()) System.*out*.println(" \t!!! Directory must be empty to delete");  
*else if* (!df.exists()) System.*out*.println("\t\t!!!There is no such file " + FileName+" in "+*f*.getAbsolutePath()+" Folder");  
 }  
*operationMenu*();  
 }  
  
*/\*\*  
 \* For Searching the user specified File/Folder in the current Folder/Directory.  
 \* If the File is present, it will print the path of file with file name.  
 \* <p>If the searched path is folder, it asks user whether he want to change his current directory to searched one.</p>  
 \* <p><b><i>The Method is Case-Sensitive.</i></b></p>  
 \** ***@throws*** *IOExceptionFor Invalid Input From User  
 \*/  
public static void* searchFile() *throws* IOException{  
*if* (*developer*) System.*out*.println("Search File Running...");  
*int* j = 0;  
System.*out*.print("Enter File Name to Search : ");  
*//sc.nextLine();*String file = *stringreader*();  
String fileName= file;*//.toLowerCase();*File[] list = *f*.listFiles();  
*if* (list.length== 0 &&*f*.isDirectory()) {  
System.*out*.println("Directory is Empty");  
 } *else if* (!*f*.isDirectory()) {  
System.*out*.println(*f*.getName() + " not a Directory");  
 } *else if* (*f*.isDirectory() &&list.length>0) {  
*List*<File>foundList= *new* ArrayList<>();  
*boolean*found = *false*;  
*//int i = list.length;*File searchedFile= *new* File(*path* + "/" + fileName);  
*for* (File i: list) {  
*if* (i.getName().matches(fileName+ "[.][0*-*9|a*-*z|A*-*Z]\*") || i.getName().equals(fileName)) {  
foundList.add(i);  
 ++j;  
found = *true*;  
 }  
 }  
*if* (found &&j >0) {  
System.*out*.println("The " + foundList.size() + " files have been founded of name " + fileName);  
*printFile*(foundList);  
 } *else if* (!found || j == 0) { System.*out*.println("File Not Found"); }  
 }*operationMenu*();  
 }  
  
*/\*\*  
 \* Displays The Files in Current Folder in ascending order with respect to name of Files.  
 \** ***@throws*** *IOExceptionFor Invalid Input  
 \*/  
public static void* displayFiles() *throws* IOException{  
*// if (developer) System.out.println("Display Files Running...");  
// String[] list = f.list();  
// System.out.println("File Path " + f.getAbsolutePath());  
// System.out.println("Parent File " + f.getParent());  
// if (list.length == 0) System.out.println(f.getName() + " Directory is Empty");  
// String[] result = mergeSort(list);  
// for (String i : result) {  
// System.out.println("\t"+i);*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(" \t!!!! "+*f*.getAbsolutePath()+" Directory is empty");  
*else* {  
Arrays.*sort*(list);  
*printFile*(list);  
 }  
*MainMenu*();  
 }  
*//Merge Sort Algorithm  
  
 /\*\*  
 \* To Merge the left and right array to result array by comparing both of them.  
 \** ***@param left*** *String[]  
 \** ***@param right*** *String[]  
 \** ***@return*** *result String[]  
 \*/  
// public static String[] merge(File[]left,String[]right){  
// String[] result = new String[left.length+ right.length];  
// int leftPointer=0, rightPointer=0,resultPointer=0;  
// while (leftPointer<left.length||rightPointer<right.length) {  
// if (leftPointer<left.length&&rightPointer<right.length) {  
// if (left[leftPointer].compareTo(right[rightPointer]) <= 0) {  
// result[resultPointer++] = left[leftPointer++];  
// } else {  
// result[resultPointer++] = right[rightPointer++];  
// }  
// } else if (leftPointer<left.length) {  
// result[resultPointer++] = left[leftPointer++];  
// } else if (rightPointer<right.length) {  
// result[resultPointer++] = right[rightPointer++];  
// }  
// }  
// return result;  
// }  
public static* File[] merge(File[]left, File[]right){  
File[] s = *new* File[left.length+right.length];  
*int* leftPointer=0, rightPointer=0,resultPointer=0;  
*while* (leftPointer<left.length||rightPointer<right.length) {  
*if* (leftPointer<left.length&&rightPointer<right.length) {  
*if* (left[leftPointer].getName().compareTo(right[rightPointer].getName()) <= 0) {  
s[resultPointer++] = left[leftPointer++];  
 } *else* {  
s[resultPointer++] = right[rightPointer++];  
 }  
 } *else if* (leftPointer<left.length) {  
s[resultPointer++] = left[leftPointer++];  
 } *else if* (rightPointer<right.length) {  
s[resultPointer++] = right[rightPointer++];  
 }  
 }  
*return* s;  
}  
*/\*\*  
 \* This is Merge Sort algorithm to sort the files in ascending order. It is used in displayFiles() method.  
 \* It takes parameter as an array of String  
 \*  
 \** ***@param list*** *String[] Array to be sorted  
 \** ***@return*** *result String[] Sorted Array.  
 \*/  
public static* File[] mergeSort(File[] list) {  
File[] filesList= *new* File[list.length];  
*if* (list.length<= 1)*return* list;  
*int* l = list.length;  
*int* mid = list.length/ 2;  
File[] left = *new* File[mid];  
File[] right;  
*if* (l % 2 == 0) right = *new* File[mid];  
*else* right = *new* File[mid + 1];  
*for* (*int* i= 0; i<mid; i++) {  
left[i] = list[i];  
 }  
*for* (*int* i= 0; i<right.length; i++) {  
right[i] = list[mid + i];  
 }  
left = *mergeSort*(left);  
right = *mergeSort*(right);  
filesList= *merge*(left,right);  
*return* filesList;  
 }  
}