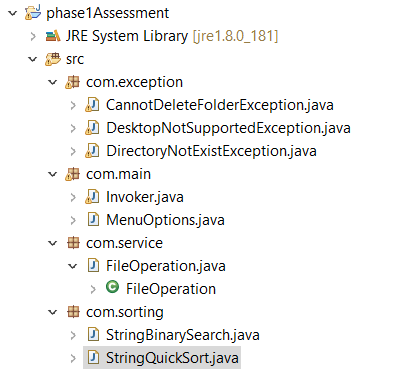
Package and File Structure of the project



***COM.Main.Invoker.java***

**package** com.main;

**import** java.io.IOException;

**import** java.nio.file.AccessDeniedException;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.nio.file.NoSuchFileException;

**import** java.nio.file.NotDirectoryException;

**import** java.util.InputMismatchException;

**import** java.util.LinkedList;

**import** java.util.Scanner;

**import** com.exception.DirectoryNotExistException;

**import** com.service.FileOperation;

**import** com.exception.CannotDeleteFolderException;

**import** com.service.FileOperation;

**public** **class** Invoker {

MenuOptions ovb = **new** MenuOptions();

FileOperation fileOpr = **new** FileOperation();

LinkedList<String> filesList;

String linebreak = "\n \*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\* \n";

Scanner getUserInput;

**public** Invoker() {

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

}

**public** **void** fileListSorting() {

System.***out***.println("Enter No.[1-3] to select options given below: ");

ovb.loopDisplay(ovb.getSecondLevelListOption());

**int** orderingOption = getOption(2);

**switch**(orderingOption) {

**case** 1:{

fileOpr.sortByAscending();

fileOpr.display();

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

fileListSorting();

**break**;

}

**case** 2:{

fileOpr.sortByDescending();

fileOpr.display();

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

fileListSorting();

**break**;

}

**case** 3:{

mainMenu();

**break**;

}

**default**:{

fileListSorting();

**break**;

}

}

}

**public** **void** openFileOption(String fileName) {

String userInput = getUserInput.next();

**while**(!userInput.matches("[yn]?")) {

System.***out***.println("Enter y for yes or n for no : ");

userInput = getUserInput.next();

}

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

**try** {

fileOpr.openFileInDesktop(fileName);

} **catch** (com.exception.DesktopNotSupportedException e) {

System.***err***.println(e.getMessage() );

fileOperationMenu();

}**catch**(IOException e) {

System.***err***.println(e.getMessage() );

fileOperationMenu();

}

}

fileOperationMenu();

}

**private** String getValidFileName() {

String fileNameToBeValidate = getUserInput.next();

**while**(fileNameToBeValidate.matches(".\*[:/\"?<>\\\\|\*]+.\*")) {

System.***out***.println("Kindly avoid the following characters [: / \" ? < > \\ | \* ] in the fileName. Give a valid FileName");

fileNameToBeValidate = getUserInput.next();

}

**return** fileNameToBeValidate;

}

**public** **void** addFile() {

System.***out***.println("Enter a FileName to be created: ");

String newFileName = getValidFileName() ;

**try** {

fileOpr.createFile( newFileName);

System.***out***.println("Do you want to open the newly created file? y/n");

openFileOption(newFileName);

} **catch** (AccessDeniedException e) {

System.***err***.println(e.getMessage());

System.***err***.println("Goto main menu, change the destination path and try it ");

fileOperationMenu();

}**catch**(FileAlreadyExistsException e) {

System.***err***.println(e.getMessage());

System.***err***.println(linebreak);

addFile();

}**catch**(IOException e) {

System.***err***.println(e.getMessage()+" Goto the previous menu, change the destination path and try it again ");

fileOperationMenu();

}

}

**public** **void** fileOperationMenu() {

System.***out***.println("Enter No.[1-4] to select options given below: ");

ovb.loopDisplay(ovb.getSecondLevelFileOprOption());

**int** orderingOption = getOption(3);

**switch**(orderingOption) {

**case** 1:{

addFile();

**break**;

}

**case** 2:

{

deleteFile();

**break**;

}

**case** 3:{

searchFile();

**break**;

}

**case** 4:

mainMenu();

**break**;

**default**:

{

fileOperationMenu();

**break**;

}

}

}

**private** **void** searchFile() {

System.***out***.println("Enter the File Name to search : ");

String fileNm = getValidFileName();

**boolean** searchResult = fileOpr.searchFile(fileNm);

**if**(searchResult) {

System.***out***.println("Do you want to open the searched file? y/n");

openFileOption(fileNm);

}

fileOperationMenu();

}

**private** **void** jumpToMainMenu(){

System.***out***.println("If you want goto previous menu, Enter 1 or \n If you want to delete other file Enter 2");

**int** option = getOption(4);

**switch**(option) {

**case** 1:{

fileOperationMenu();

**break**;

}

**case** 2:{

deleteFile();

**break**;

}

**default**:{

jumpToMainMenu();

**break**;

}

}

}

**private** **void** deleteFile() {

System.***out***.println("Enter File Name to be deleted : ");

String fileNameToBeDeleted = getValidFileName();

**try** {

fileOpr.deleteFile(fileNameToBeDeleted);

} **catch** (NoSuchFileException e) {

System.***err***.println(e.getMessage());

jumpToMainMenu();

} **catch** (CannotDeleteFolderException e) {

System.***err***.println(e.getMessage());

deleteFile();

}**catch**(IOException e) {

System.***err***.println(e.getMessage());

jumpToMainMenu();

}

fileOperationMenu();

}

**private** **int** getOption(**int** methodCalled) {

**int** option = 0;

String inputValue = getUserInput.next();

**if**(inputValue.matches("[0-9]+")) {

option =Integer.*parseInt*(inputValue);

}**else** {

switchToMethods(methodCalled);

}

**return** option;

}

**private** **void** switchToMethods(**int** methodCalled) {

**switch**(methodCalled) {

**case** 1:

mainMenu();

**break**;

**case** 2:

fileListSorting();

**break**;

**case** 3:

fileOperationMenu();

**break**;

**case** 4:

jumpToMainMenu();

**break**;

}

}

**public** **void** mainMenu() {

System.***out***.println("Enter No.[1-4] to select the options");

ovb.loopDisplay(ovb.getFirstLevelOption());

**int** option = getOption(1);

**switch**(option) {

**case** 1:{

fileListSorting();

**break**;

}

**case** 2:{

fileOperationMenu();

**break**;

}

**case** 3 :{

getDestFolderPath();

mainMenu();

**break**;

}

**case** 4:{

endFunction();

**break**;

}

**default**:{

mainMenu();

**break**;

}

}

}

**private** **void** endFunction() {

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

System.***out***.println("Thanks for using this File Handling Application ");

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

System.*exit*(0);

}

**private** **void** getDestFolderPath() {

System.***out***.println("Enter the Destination Folder Path : ");

String folderPath = getUserInput.next();

**while**(!folderPath.matches("([A-Za-z]):\\\\.\*")) {

System.***out***.println("Enter valid Folder Path");

folderPath = getUserInput.next();

}

**try** {

fileOpr.setFolderPath(folderPath);

} **catch** (NotDirectoryException | DirectoryNotExistException e) {

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

getDestFolderPath();

}

filesList = fileOpr.getFileList();

}

**public** **static** **void** main(String[] args) **throws** NotDirectoryException, DirectoryNotExistException {

Invoker invoke = **new** Invoker();

String welcomeNote ="\t Welcome to our Company: \"Lockers Pvt. Ltd.\" "

+ "\n \t \t"+"Aim of our company is to digitize our products. As a initiative to our aim"+

"\n"+"We have launched our first prototype named \"LockedMe.com\" ."+

"\n \t"+"Developed by Raja Rajeswari Jayachandran.";

System.***out***.println(welcomeNote+"\n");

invoke.getDestFolderPath();

**try** {

invoke.mainMenu();

}**catch**(InputMismatchException e) {

System.***out***.println("Input MisMatch Exception try again with correct input data" );

}

}

***Com.Main.MenuOption.Java***

**package** com.main;

**public** **class** MenuOptions {

**public** String[] getFirstLevelOption() {

String[] firstLevelOption = {"Sorted Files List" ,"File Operation (Add,Delete&Search)","To Change the destination Folder path","Quit"};

**return** firstLevelOption;

}

**public** String[] getSecondLevelListOption() {

String[] secondLevelListOption = {"Sorted File List in Ascending Order","Sorted File List in Descending Order","Return to Main Menu"};

**return** secondLevelListOption;

}

**public** String[] getSecondLevelFileOprOption() {

String[] secondLevelFileOprOption = { "Add File","Delete File","Search File","Return to Main Menu"};

**return** secondLevelFileOprOption;

}

**public** **void** loopDisplay(String[] optionMenu) {

**int** index = 1;

**for**(String option:optionMenu) {

System.***out***.println(index+".) "+option);

index++;

}

}

}

***Com.service.FileOperation***

package com.service;

import java.awt.Desktop;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.nio.file.FileAlreadyExistsException;

import java.nio.file.Files;

import java.nio.file.NoSuchFileException;

import java.nio.file.NotDirectoryException;

import java.nio.file.Paths;

import java.util.LinkedList;

/\*\*\*\* User Defined Exception Files\*\*\*\*/

import com.exception.DesktopNotSupportedException;

import com.exception.DirectoryNotExistException;

import com.exception.CannotDeleteFolderException;

/\*\*\*Sorting and Searching files \*\*\*\*/

import com.sorting.StringBinarySearch;

import com.sorting.StringQuickSort;

public class FileOperation {

private String[] fileNameArray;

private LinkedList<String> fileNameList;

private String folderPath;

private File folderName ;

public void setFileNameList(String[] fileNameArray) {

fileNameList = new LinkedList<>();

for(String fileName:fileNameArray) {

fileNameList.add(fileName);

}

}

public void setFolderPath(String folderPath) throws NotDirectoryException, DirectoryNotExistException {

folderName = new File(folderPath);

if(folderName.exists()) {

if(folderName.isDirectory()) {

this.folderPath = folderPath;

}else {

throw new NotDirectoryException("Enter Correct Folder Path");

}

}else{

throw new DirectoryNotExistException("Directory "+folderPath +" does not exist.");

}

}

public void createFile(String fileName) throws FileAlreadyExistsException, IOException {

File create= new File(folderPath+"\\"+fileName);

if(create.createNewFile()) {

fileNameList.add(create.getName());

sortByAscending(); //Sort the list after adding New File

System.out.println("File Created Successfully");

}else {

throw new FileAlreadyExistsException("File already exists.Kindly Enter another Name");

}

}

**public** **void** deleteFile(String fileNameToBeDeleted) **throws** NoSuchFileException, IOException, CannotDeleteFolderException {

File fileToDelete = **new** File(folderPath+"\\"+fileNameToBeDeleted);

**if**(fileToDelete.exists()) {

**if**(fileToDelete.isFile()) {

**if**( Files.*deleteIfExists*(Paths.*get*(folderPath+"\\"+fileNameToBeDeleted))) {

fileNameList.remove(fileNameToBeDeleted);

}**else** {

**throw** **new** NoSuchFileException("File does not Exist");

}

}**else** {

**throw** **new** CannotDeleteFolderException("Can't Delete Folder. ");

}

}**else** {

**throw** **new** NoSuchFileException("File does not Exist");

}

System.***out***.println("Deletion successful.");

}public LinkedList<String> getFileList(){

fileNameArray = folderName.list();

setFileNameList(fileNameArray);

LinkedList<String> fileNameLinkedList = fileNameList;

return fileNameLinkedList;

}

public void display() {

int index = 1;

if(fileNameArray.length == 0) {

System.out.println("\n \t Folder is Empty !!!!!!!!!");

}else {

for(String fileName:fileNameArray) {

System.out.println(index+".) "+fileName);

index++;

}

}

}

public boolean searchFile(String fileName) {

fileNameArray = getFilesNameArray(fileNameList);

boolean fileExist = StringBinarySearch.binarySearch(fileNameArray, fileName);

if(fileExist) {

System.out.println("File "+fileName +" exists. File-path : "+folderPath+"\\"+fileName);

}else {

System.out.println("File "+fileName +" does not exist. \n");

}

return fileExist;

}

public void openFileInDesktop(String fileName) throws IOException, DesktopNotSupportedException {

File fileToOpen = new File(folderPath+"\\"+fileName);

if(Desktop.isDesktopSupported()) {

Desktop openApp = Desktop.getDesktop();

if(fileToOpen.exists()) {

openApp.open(fileToOpen);

}else {

throw new FileNotFoundException();

}

}else {

throw new DesktopNotSupportedException("Desktop Not supported");

}

}

public void sortByAscending() {

fileNameArray = getFilesNameArray(fileNameList);

int listLength = fileNameArray.length-1;

System.out.println("List Length"+listLength);

if(listLength == -1) {

StringQuickSort.quickSort(fileNameArray,0, listLength, "Ascending");

}

}

public void sortByDescending() {

fileNameArray = getFilesNameArray(fileNameList);

int listLength = fileNameArray.length-1;

if(listLength != -1) {

StringQuickSort.quickSort(fileNameArray,0, listLength, "Descending");

}

}

public String[] getFilesNameArray(LinkedList<String> fileNameList) {

int index = 0;

String[] fileNameArray = new String[fileNameList.size()];

for(String fileName:fileNameList) {

fileNameArray[index] = fileName;

index++;

}

return fileNameArray;

}

}

***Com.Sorting.StringBinarySearch.Java***

**package** com.sorting;

**public** **class** StringBinarySearch {

**public** **static** **boolean** binarySearch(String[] fileNameList,String fileName) {

**int** minIndex = 0;

**int** maxIndex = fileNameList.length-1;

**while**(minIndex <= maxIndex) {

**int** midIndex = minIndex+(maxIndex-minIndex)/2;

**int** comparisionResult = fileName.compareToIgnoreCase(fileNameList[midIndex]);

**if**(comparisionResult == 0) {

**return** **true**;

}**else** **if**(comparisionResult > 0) {

minIndex = midIndex +1;

}**else** **if**(comparisionResult < 0){

maxIndex = midIndex-1;

}

}

**return** **false**;

}

}

***Com.Sorting.StringQuickSort.Java***

**package** com.sorting;

**public** **class** StringQuickSort {

**static** **void** swap(String[] fileList,**int** low,**int** high) {

String temp = fileList[low];

fileList[low] = fileList[high];

fileList[high] = temp;

}

**public** **static** String[] quickSort(String[] filesList,**int** low,**int** high,String ordering) {

**if**(low >= high) {

**return** filesList;

}

**int** pivotIndex = 0;

**if**(ordering.equalsIgnoreCase("ascending")) {

pivotIndex = *listPartitionAscending*(filesList,low,high);

}**else** **if**(ordering.equalsIgnoreCase("descending")) {

pivotIndex = *listPartitionDescending*(filesList,low,high);

}

*quickSort*(filesList,low, pivotIndex-1,ordering);

*quickSort*(filesList,pivotIndex+1,high,ordering);

**return** filesList;

}

**static** **private** **int** listPartitionDescending(String[] filesList, **int** low, **int** high) {

String pivot = filesList[low];

**int** partitionLow = low;

**int** partitionHigh = high;

**while**(partitionLow < partitionHigh) {

**while**((pivot.compareToIgnoreCase(filesList[partitionLow]) <= 0) &&(partitionLow < partitionHigh) ) {

partitionLow++;

}

**while**(pivot.compareToIgnoreCase(filesList[partitionHigh]) > 0) {

partitionHigh--;

}

**if**(partitionLow < partitionHigh) {

*swap*(filesList, partitionLow, partitionHigh);

}

}

*swap*(filesList,low,partitionHigh);

**return** partitionHigh;

}

**static** **private** **int** listPartitionAscending(String[] filesList, **int** low, **int** high) {

String pivot = filesList[low];

**int** partitionLow = low;

**int** partitionHigh = high;

**while**(partitionLow < partitionHigh) {

**while**((pivot.compareToIgnoreCase(filesList[partitionLow]) >= 0) &&(partitionLow < partitionHigh) ) {

partitionLow++;

}

**while**(pivot.compareToIgnoreCase(filesList[partitionHigh]) < 0) {

partitionHigh--;

}

**if**(partitionLow < partitionHigh) {

*swap*(filesList, partitionLow, partitionHigh);

}

}

*swap*(filesList,low,partitionHigh);

**return** partitionHigh;

}

}