**LockedMe – Virtual Key for Repositories**his project is hosted at https://github.com/KuldipShendge/LockedMe---Virtual-Key-for-Repositories

The project is developed by Kuldip Shendge  
  
Core concepts used in project

Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Streams API   
  
1.Source Code  
**1.1** Writing a program in Java for the entry point of the application (**LockedMeMain.java**)  
  
package com.lockedme;

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", "Kuldip Shendge");

HandleOptions.*handleWelcomeScreenInput*();

}

}

**1.2** Writing a program in Java to display Menu options available for the user (**MenuOptions.java**)  
  
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"

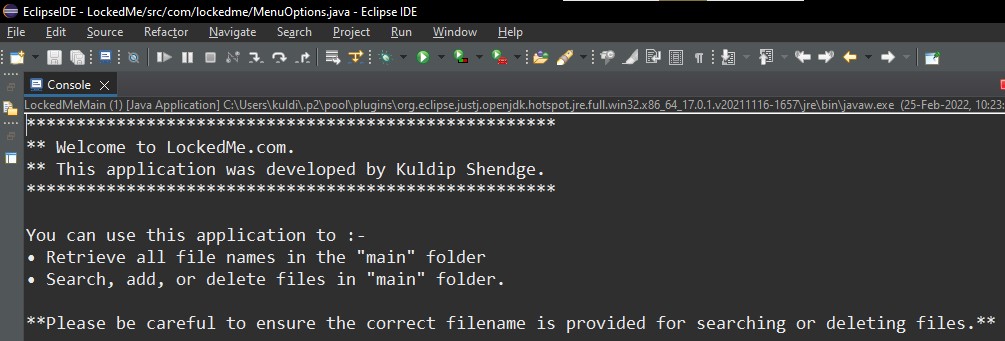
+ "• Retrieve all file names in the \"main\" folder\n"

+ "• 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);

}  
**OUTPUT:  
**

1.3 Writing method to display Initial Menu

public static void displayMenu() {

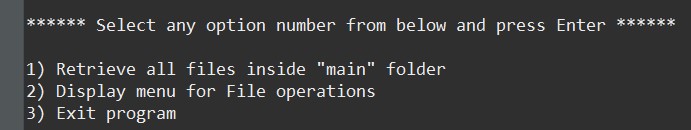
String menu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"

+ "1) Retrieve all files inside \"main\" folder\n" + "2) Display menu for File operations\n"

+ "3) Exit program\n";

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

}

**OUTPUT:  
  
  
  
  
1.4** Writing method to display Secondary Menu for File Operations  
  
public static void displayFileMenuOptions() {

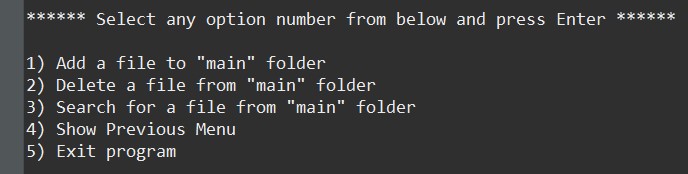
String fileMenu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"

+ "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);

}

**OUTPUT:**  
  
**1.5** Writing method to handle user input in initial Menu  
  
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.");

}

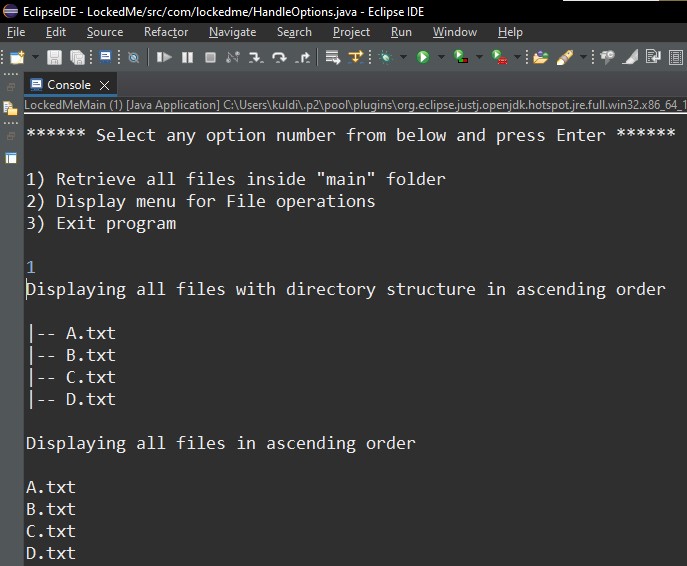
} catch (Exception e) {

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

*handleWelcomeScreenInput*();

}

} while (running == true);

}  
  
**OUTPUT:**  
  
**1.6** Writing method to handle user input in Secondary Menu for File Operations

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);

}

**OUTPUT:**