Project
LockedMe.com-Virtual Key for Repositories

Contents of Document:

Developer details

The project is developed by **Himanshi Singh** from **FSD Java Developers (All engineers batch)** and the code for this project is hosted at himanshisinghh/LockedMEMain (github.com)

Sprints planning and Task completion

The project is planned to be completed in 3 sprints. The tasks are expected to be completed in the sprints are:

- **Sprint 1**

  1. Creating the flow of the application

  2. Initializing git repositories to track changes in task as development progresses.

- **Sprint 2**

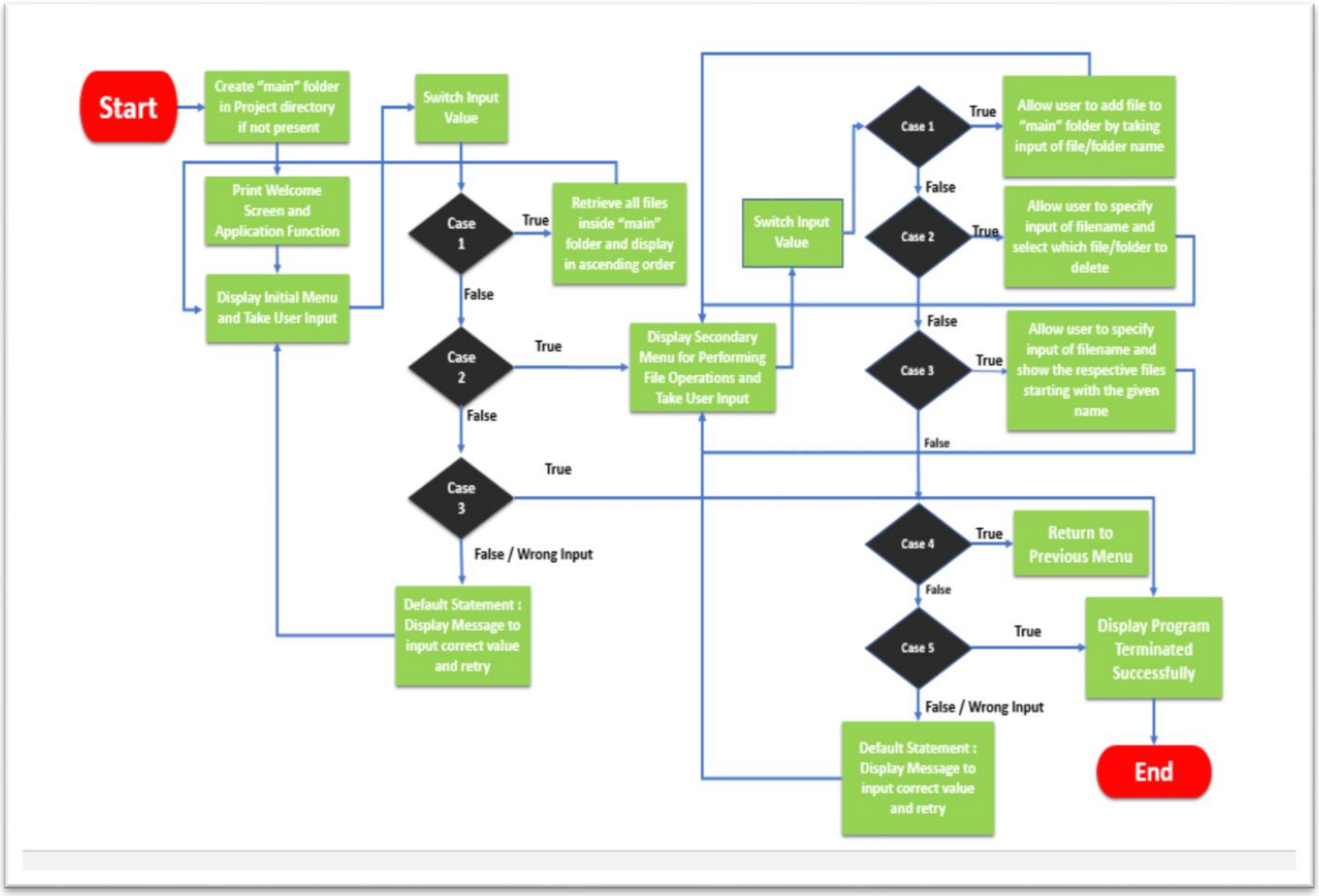
  1. Writing the Java program to achieve the project requirements.

  2. Testing the Java program with various User inputs

- **Sprint 3**
  1. Pushing code to GitHub
  2. Creating the Specification document highlighting the Application Capabilities, Appearance, and User interactions

Flow of the Application

This includes flowcharts & Algorithm:



Start
Create "main" folder in Project directory if not present
Print Welcome Screen and Application Function
Display Initial Menu and Take User Input
Switch Input Value
Case 1
True
Retrieve all files inside "main" folder and display in ascending order
False
Case 2
True
Display Secondary Menu for Performing File Operations and Take User Input
False
Case 3
True
False / Wrong Input
Default Statement : Display Message to input correct value and retry
Switch Input Value
Case 1
True
Allow user to add file to "main" folder by taking input of file/folder name
False
Case 2
True
Allow user to specify input of filename and select which file/folder to delete
False
Case 3
True
Allow user to specify input of filename and show the respective files starting with the given name
False
Case 4
True
Return to Previous Menu
False
Case 5
True
Display Program Terminated Successfully
False / Wrong Input
Default Statement : Display Message to input correct value and retry
End

Flow of
Application.LockeM

Core concepts used in this project

- File Handling
- Collections framework
- Searching & Sorting
- Exceptions Handling
- Flow Control statements & Recursions
- Data Structures: Arrays, Strings

Describing the product capabilities, appearance and user interactions

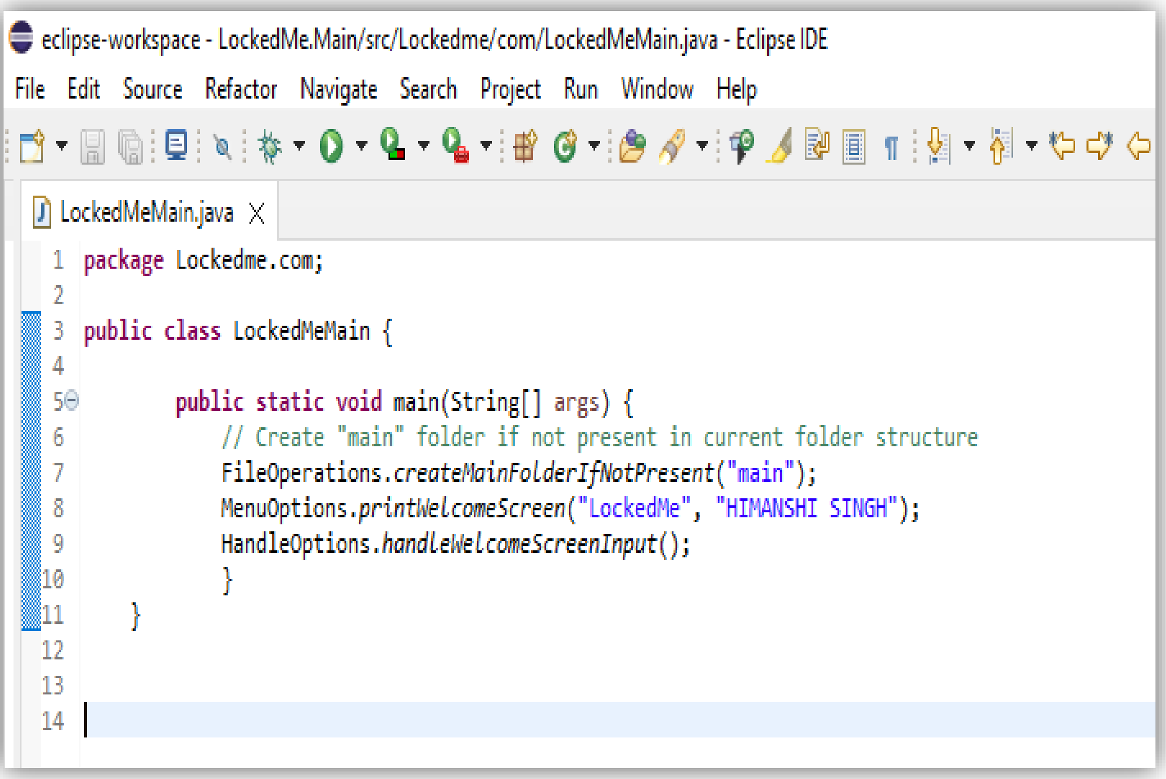
To demonstrate the products capabilities, appearance and user interactions, following are the sub-sections configured for the project:

1. **Creating the project code in Eclipse**
2. **Writing a program in Java for entry point the application (LockedMe.java)**
3. **Writing a program in Java to display list options available for the user (ListOptions.java)**
4. **Writing a program in Java to handle List options selected by user (HandleOptions.java)**
5. **Writing a program in Java to perform the File handling operations as specified by user (FileHandlingOps.java)**
6. **Pushing the code to GitHub repository**

Step 1: Creating a new project in Eclipse

- **Open Eclipse**
- **Go to file--> New--> Project--> Java Project--> Next.**
- **Type in any project name and click on "Finish".**
- **Select your project and go to file → New→ Class.**
- **Enter LockedMe as class name and do check the check box "public static void main (String [] args)", and click on "Finish**".

Step 2: Writing a program in Java for the entry point of application (start)


eclipse-workspace - LockedMe.Main/src/Lockedme/com/LockedMeMain.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LockedMeMain.java

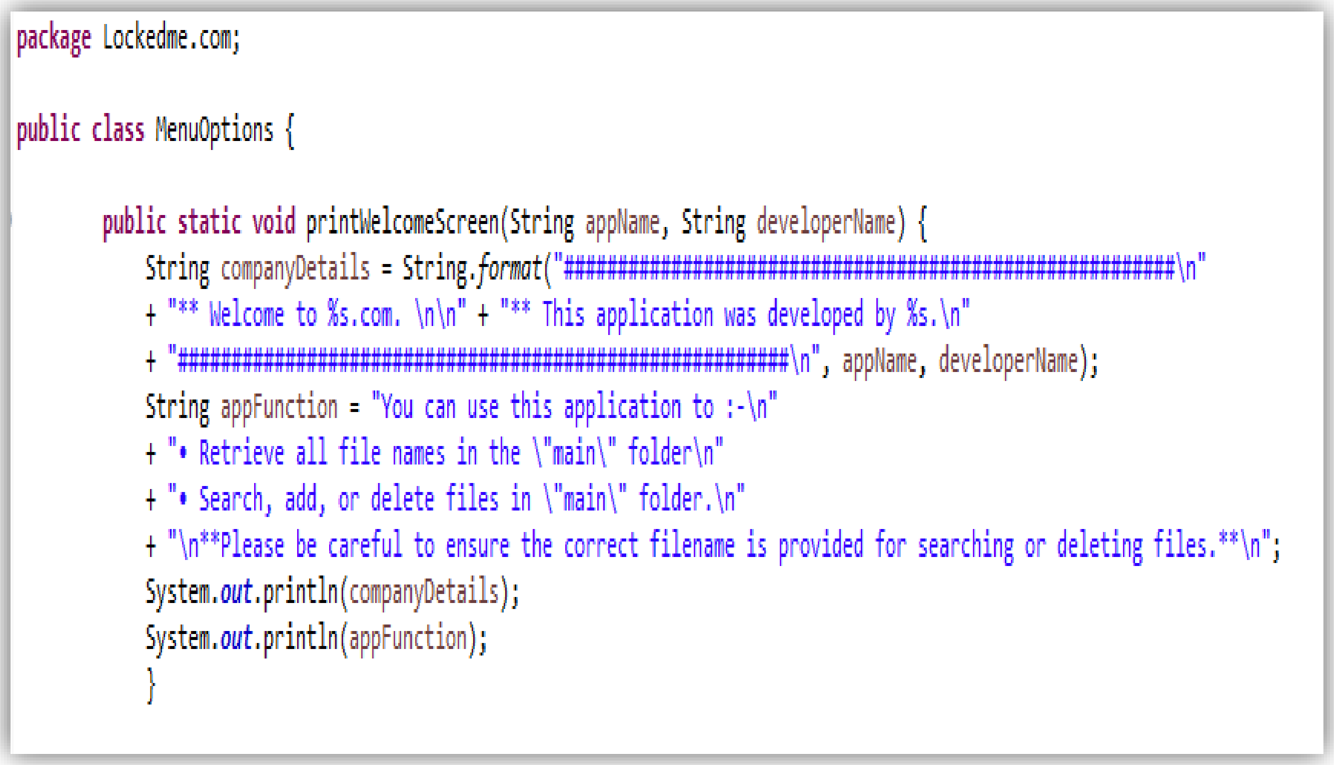
```java
package Lockedme.com;

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", "HIMANSHI SINGH");
            HandleOptions.handleWelcomeScreenInput();
            }
    }



```

Step 3: Writing a program in Java to display List options available for the user (MenuOptions.java)

- Select your project and go to File→ New → Class
- Enter MenuOptions as class name and click “Finish.”
- MenuOptions include methods for:

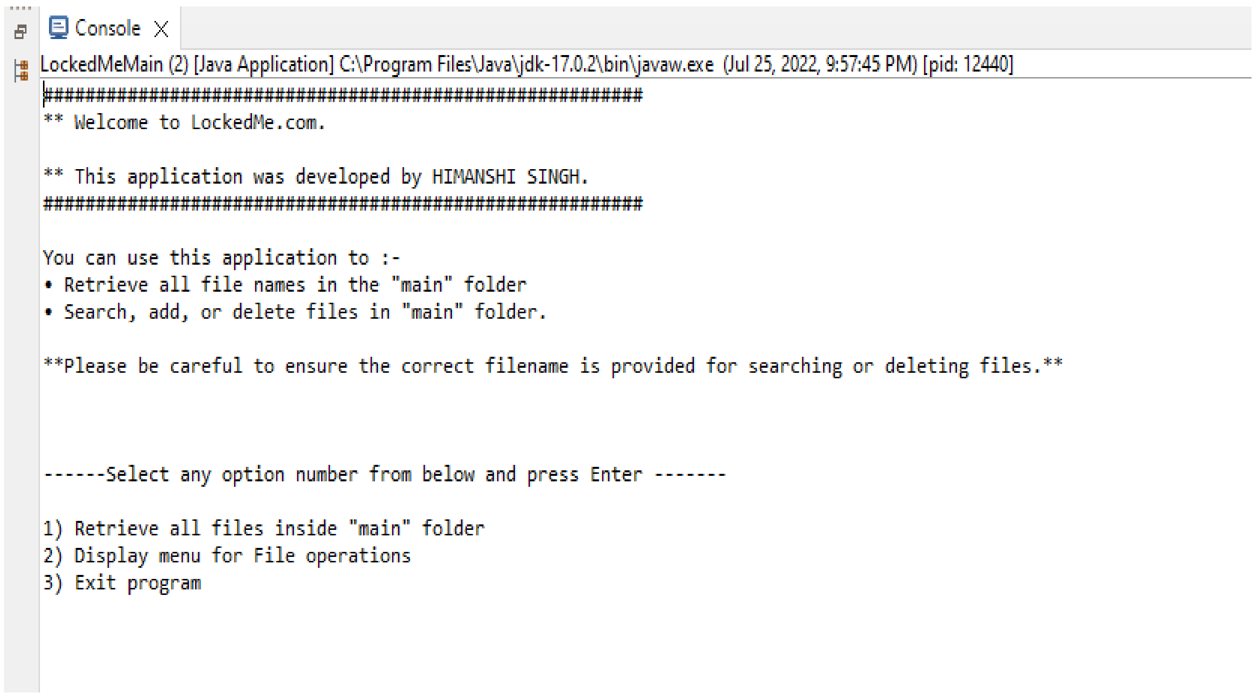
  i. Displaying Welcome Screen (Welcome to LockedMe.com)
  ii. Displaying Initial Menu
  iii. Displaying Secondary Menu for File Operations available

Step 3(i): Writing method to display Welcome Screen.



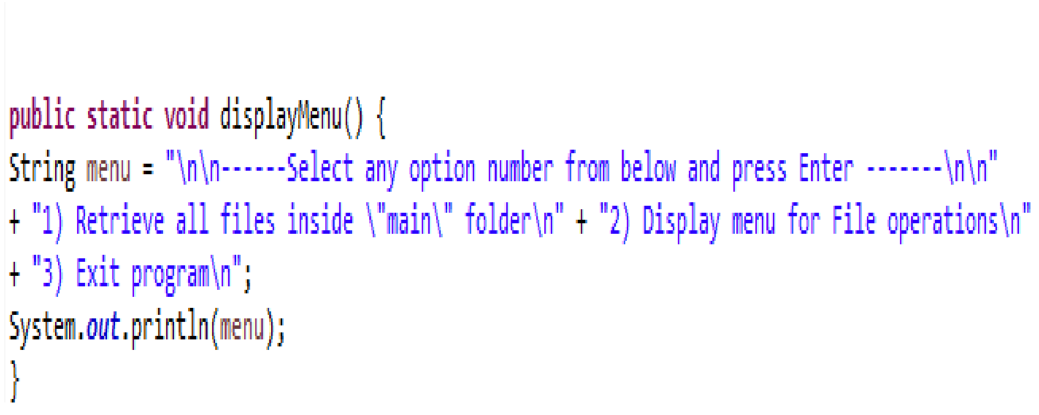
```
package Lockedme.com;

public class MenuOptions {

        public static void printWelcomeScreen(String appName, String developerName) {
            String companyDetails = String.format("#########################################################\n"
            + "** Welcome to %s.com. \n\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:



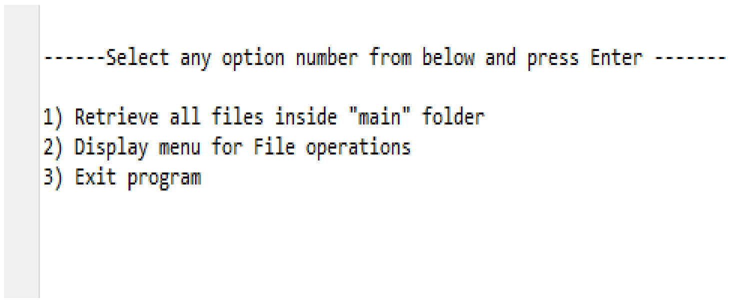
```
LockedMeMain (2) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  (Jul 25, 2022, 9:57:45 PM) [pid: 12440]
#########################################################
** Welcome to LockedMe.com.

** This application was developed by HIMANSHI SINGH.
#########################################################

You can use this application to :-
• Retrieve all file names in the "main" folder
• Search, add, or delete files in "main" folder.

**Please be careful to ensure the correct filename is provided for searching or deleting files.**



------Select any option number from below and press Enter -------

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program
```

Step 3(ii): **Writing method to display Initial Menu.**



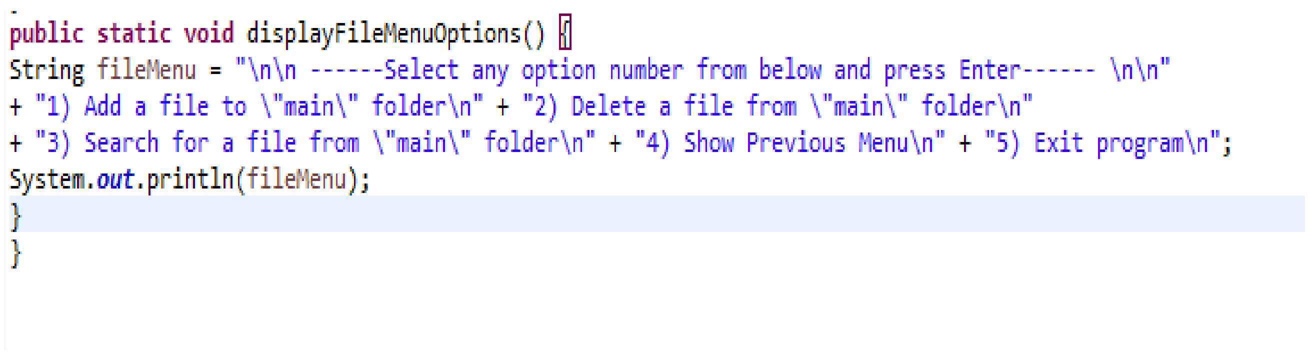
```
public static void displayMenu() {
String menu = "\n\n------Select any option number from below and press Enter -------\n\n"
+ "1) Retrieve all files inside \"main\" folder\n" + "2) Display menu for File operations\n"
+ "3) Exit program\n";
System.out.println(menu);
}
```

Output:



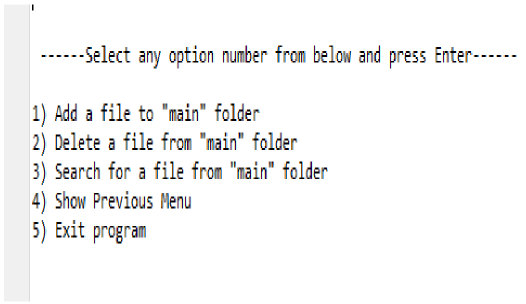
```
------Select any option number from below and press Enter -------

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program
```

Step 3(iii) writing method to display Initial Menu



```
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:



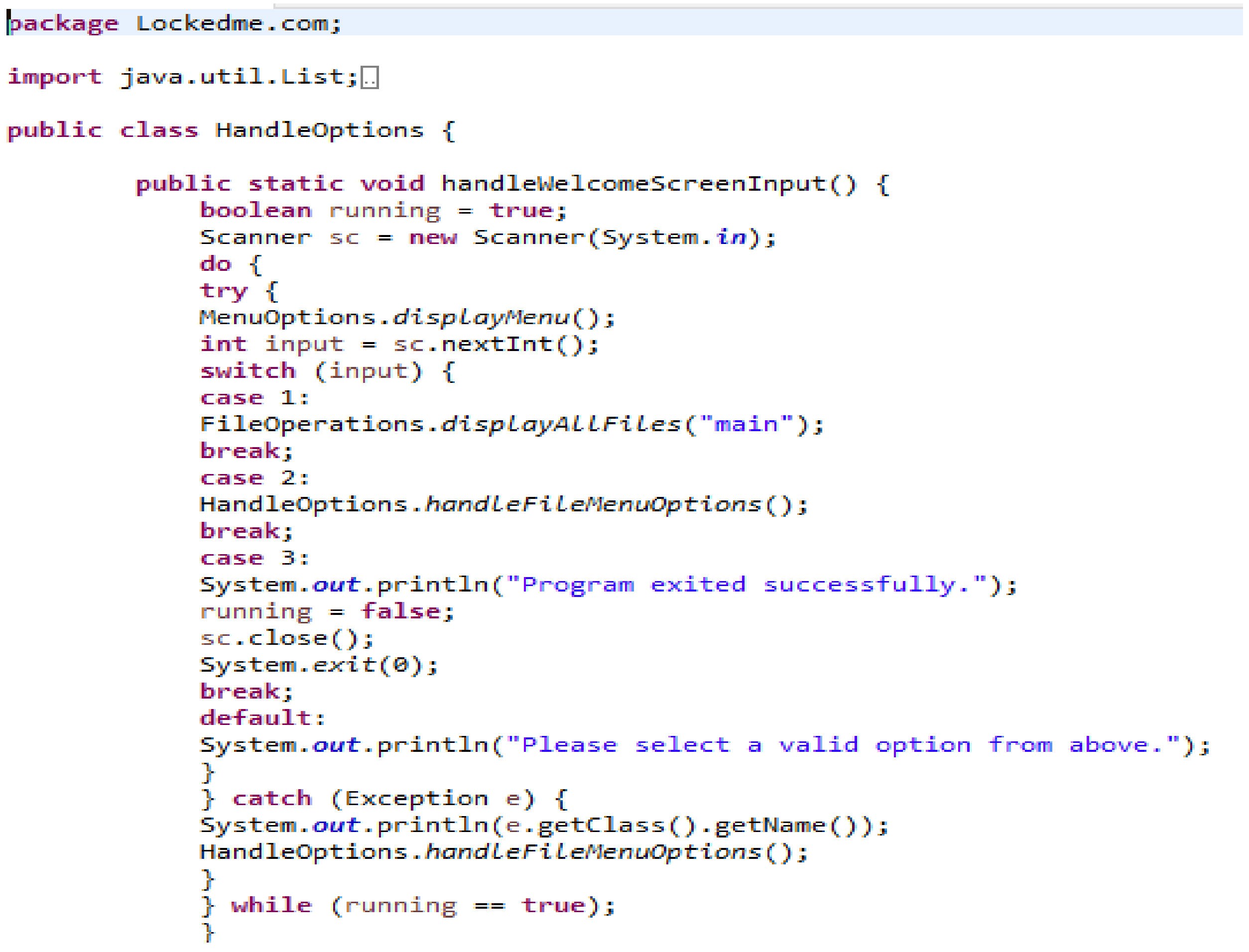
```
 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program
```

Step 4: Writing a program in Java to handle List options selected by user (HandleOptions.java)

- **Select your project and go to File -> New -> Class.**
- **Enter HandleOptions in class name and click on "Finish."**
- **HandleOptions consists methods for -:**
  i. Handling input selected by user in initial Menu

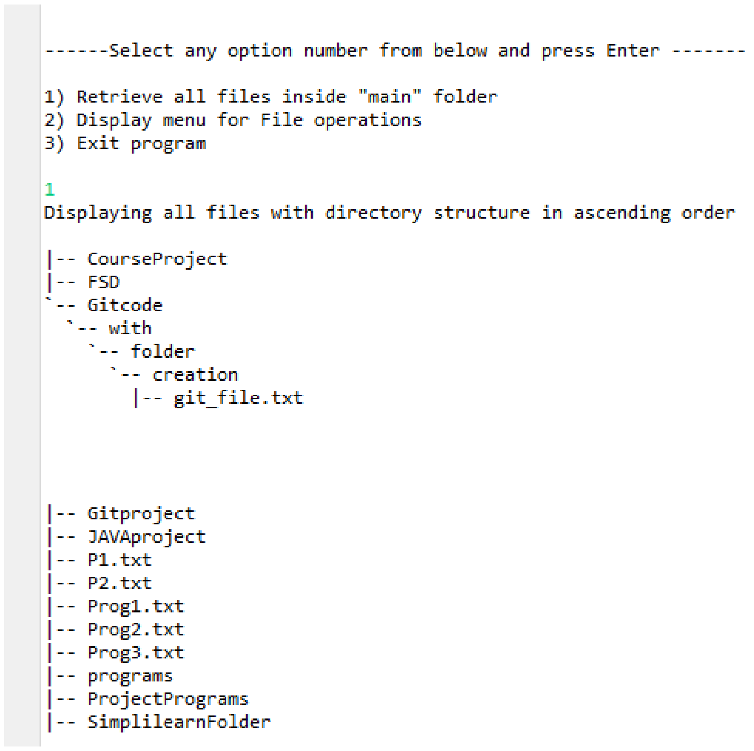
  ii. Handling input selected by user in secondary Menu for File Operations

Step 4 (i): Writing method to handle user input in initial Menu

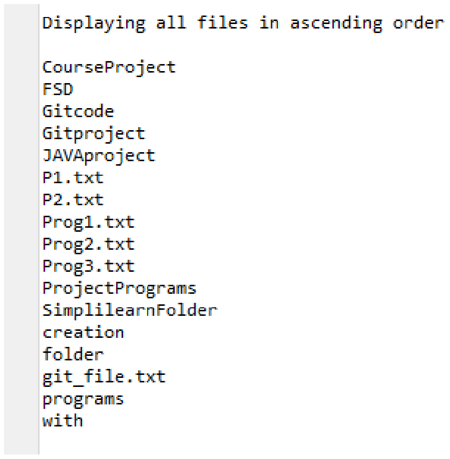


```
package Lockedme.com;

import java.util.List;

public class HandleOptions {

        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());
            HandleOptions.handleFileMenuOptions();
            }
            } while (running == true);
            }
```

Output:

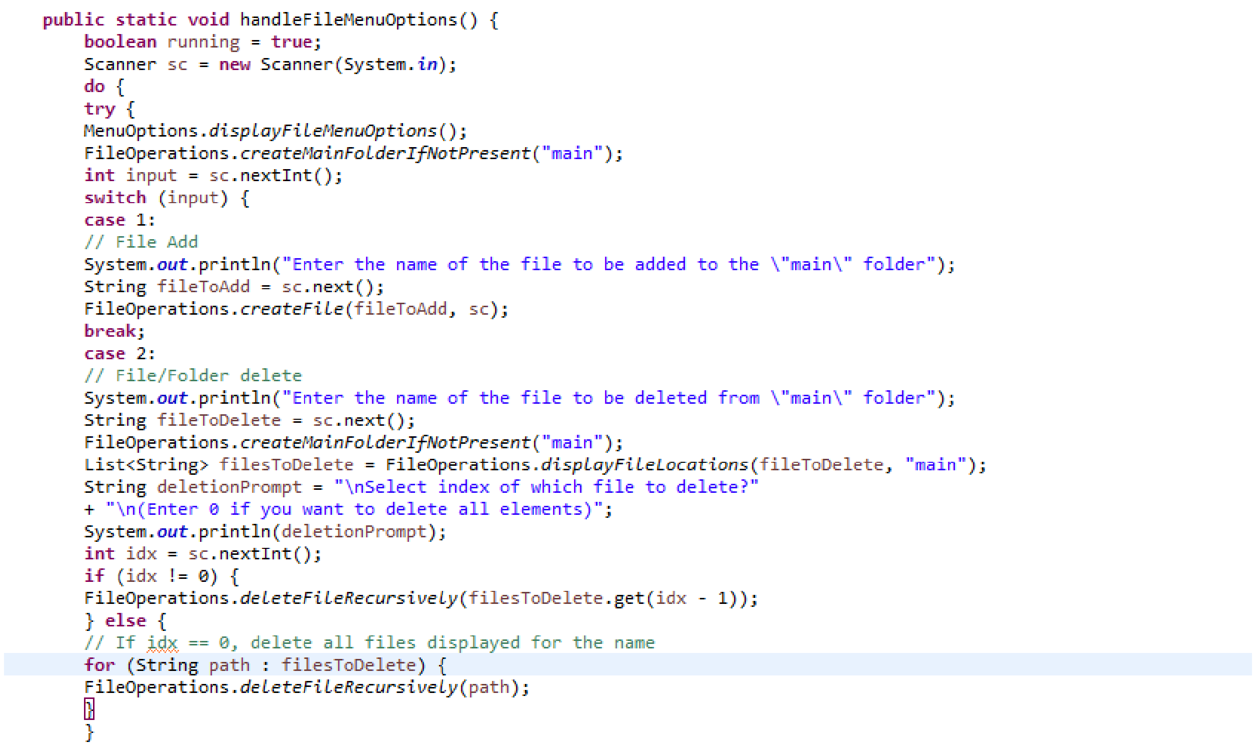


```
------Select any option number from below and press Enter -------

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- CourseProject
|-- FSD
`-- Gitcode
  `-- with
    `-- folder
      `-- creation
        |-- git_file.txt




|-- Gitproject
|-- JAVAproject
|-- P1.txt
|-- P2.txt
|-- Prog1.txt
|-- Prog2.txt
|-- Prog3.txt
|-- programs
|-- ProjectPrograms
|-- SimplilearnFolder
```

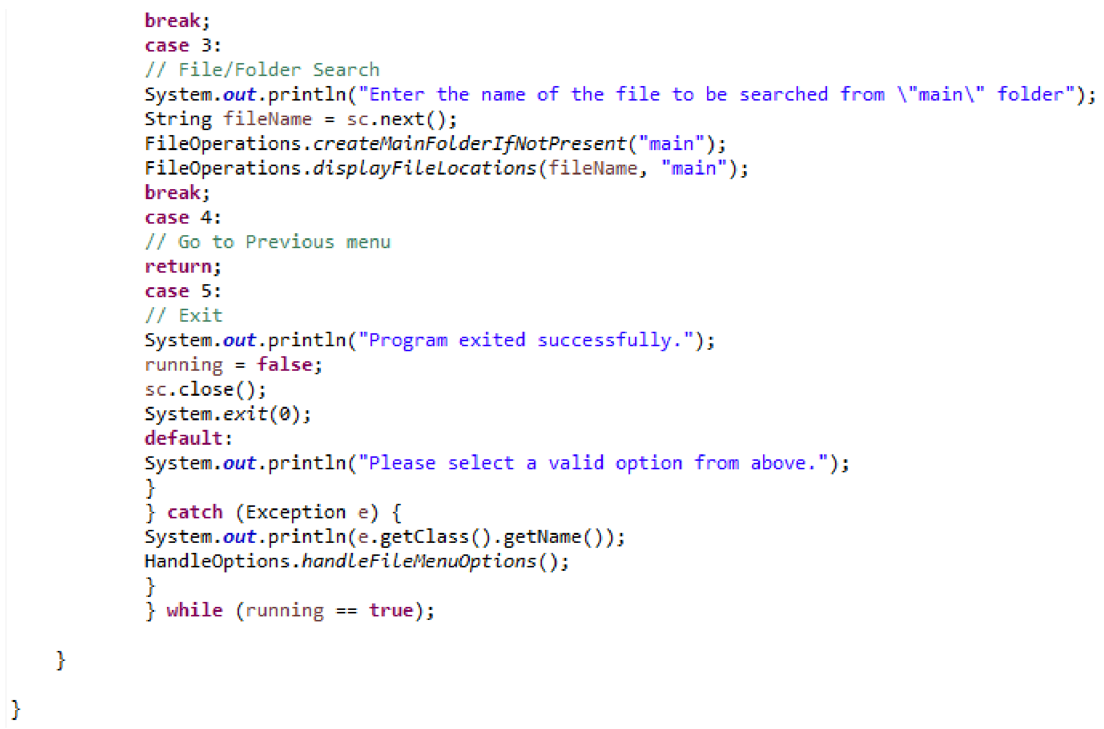


```
Displaying all files in ascending order

CourseProject
FSD
Gitcode
Gitproject
JAVAproject
P1.txt
P2.txt
Prog1.txt
Prog2.txt
Prog3.txt
ProjectPrograms
SimplilearnFolder
creation
folder
git_file.txt
programs
with
```

Step 4(ii): Writing method to handle user input in Secondary Menu for File Operations

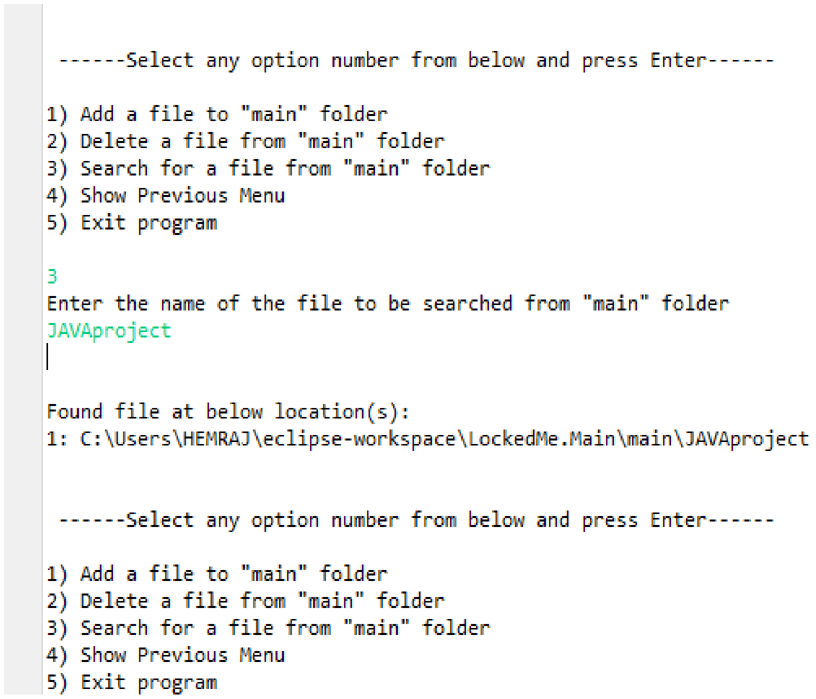


```
public static void handleFileMenuOptions() {
    boolean running = true;
    Scanner sc = new Scanner(System.in);
    do {
    try {
    MenuOptions.displayFileMenuOptions();
    FileOperations.createMainFolderIfNotPresent("main");
    int input = sc.nextInt();
    switch (input) {
    case 1:
    // 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());
            HandleOptions.handleFileMenuOptions();
            }
            } while (running == true);

    }

}
```

Output:

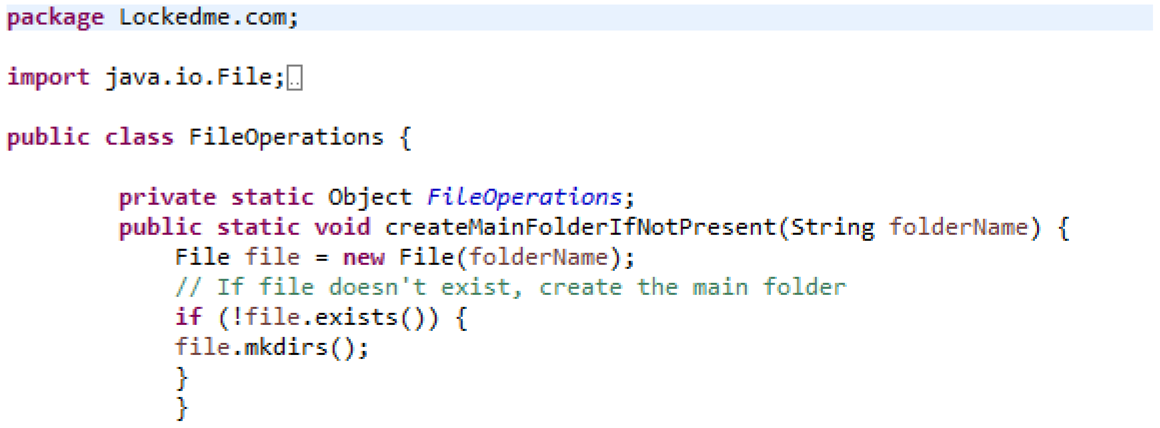


```
 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
JAVAproject


Found file at below location(s):
1: C:\Users\HEMRAJ\eclipse-workspace\LockedMe.Main\main\JAVAproject


 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program
```

Step 5: Writing a program in Java to perform the File Handling operations as specified by user ()

- Select your project and go to File -> New -> Class.
- Enter **FileOperations** in class name and click on “Finish.”
- **FileOperations** consists methods for -:
    i. Creating “main” folder in project if it’s not already present
    ii. Displaying all files in “main” folder in ascending order and also with directory structure.
    iii. Creating a file/folder as specified by user input.
    iv. Search files as specified by user input in “main” folder and it’s subfolders.
    v. Deleting a file/folder from “main” folder

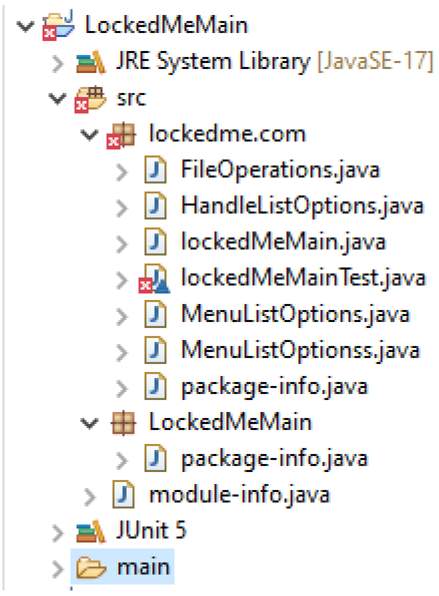
Step 5(i): Writing method to create "main" folder in project if it's not present



```
package Lockedme.com;

import java.io.File;

public class FileOperations {

        private static Object FileOperations;
        public static void createMainFolderIfNotPresent(String folderName) {
            File file = new File(folderName);
            // If file doesn't exist, create the main folder
            if (!file.exists()) {
            file.mkdirs();
            }
            }
```

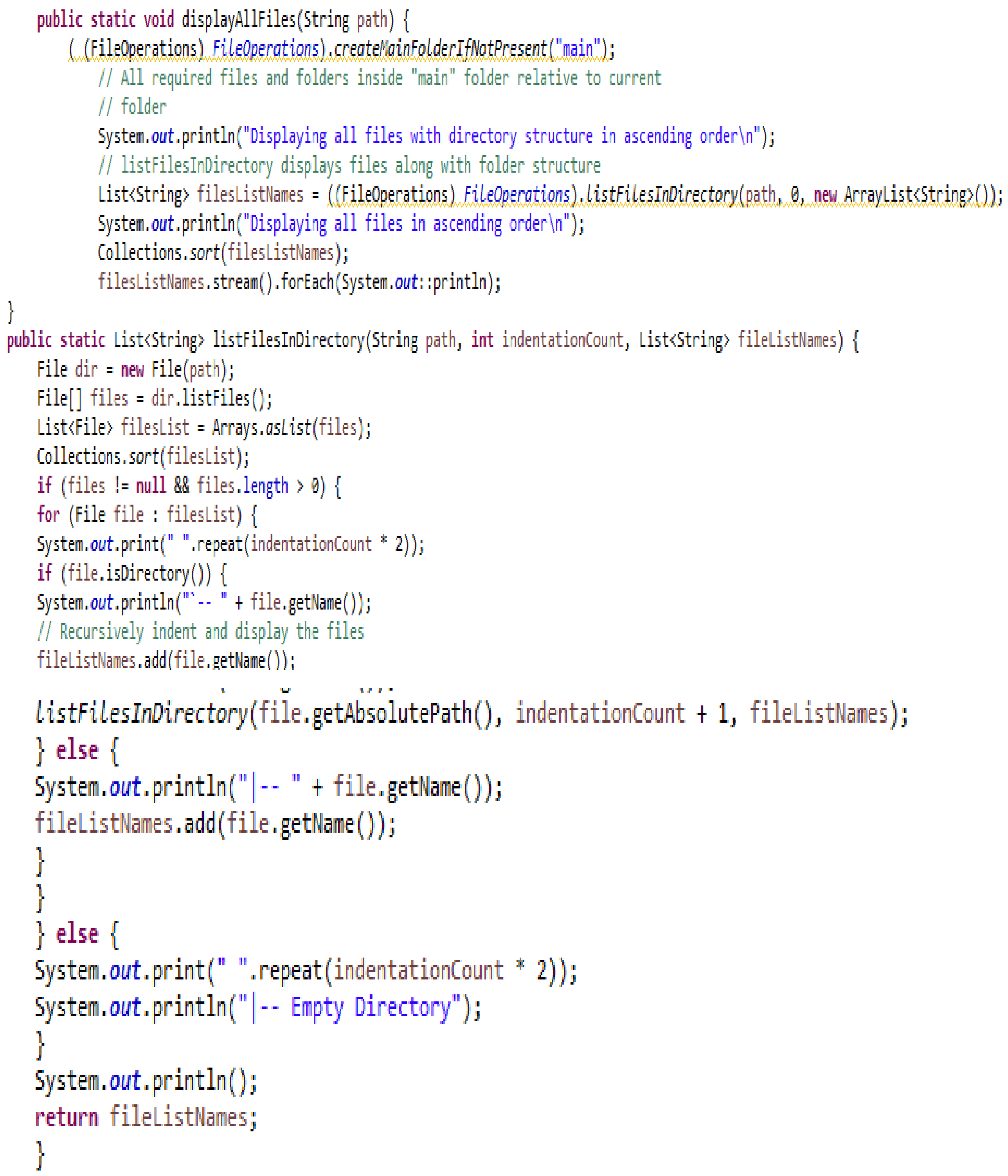
Output:

-



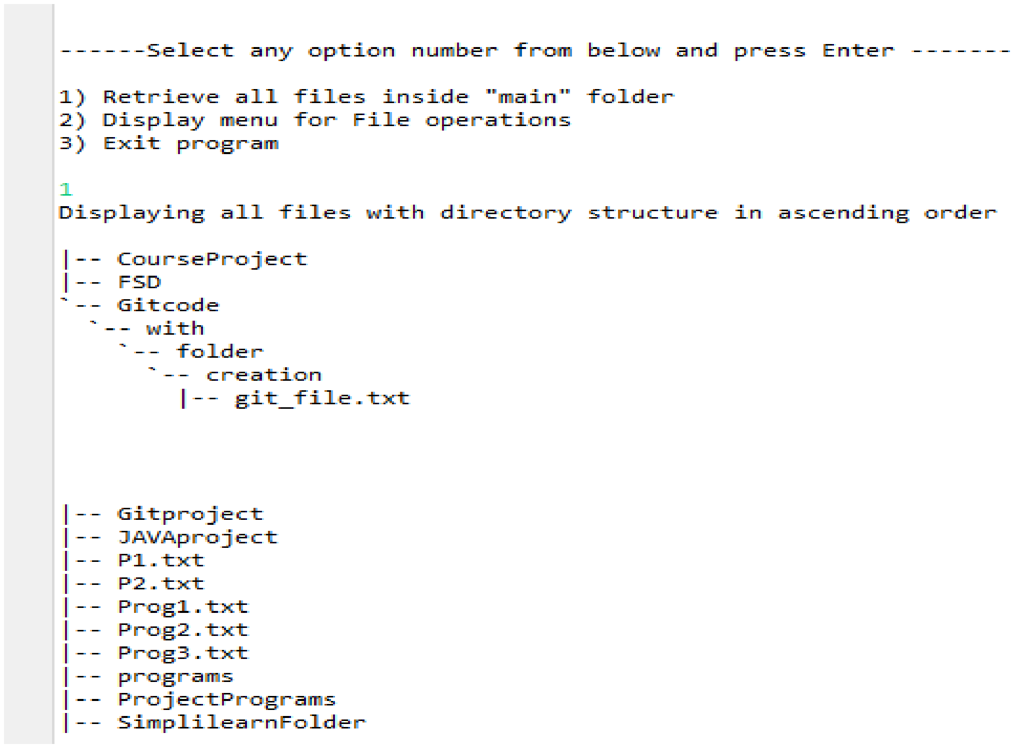
```
LockedMeMain
  JRE System Library [JavaSE-17]
  src
    lockedme.com
      FileOperations.java
      HandleListOptions.java
      lockedMeMain.java
      lockedMeMainTest.java
      MenuListOptions.java
      MenuListOptionss.java
      package-info.java
    LockedMeMain
      package-info.java
    module-info.java
  JUnit 5
  main
```

Step 5(ii): Writing method to display all files in "main" folder in ascending order and also with directory structure. ("`--" represents a directory. "|--" represents a file.)

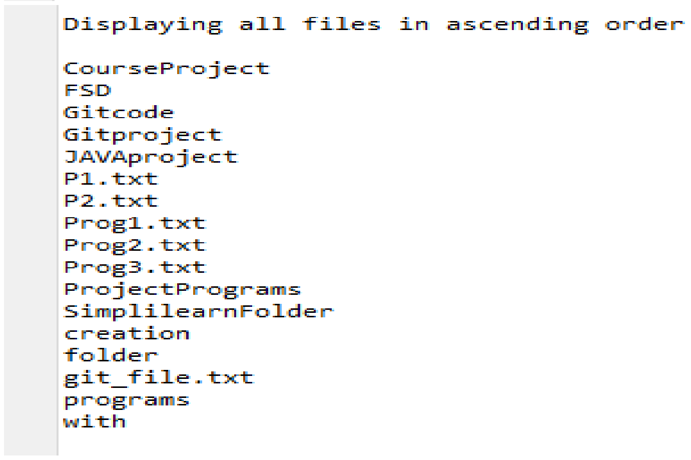


```java
    public static void displayAllFiles(String path) {
        ( (FileOperations) FileOperations).createMainFolderIfNotPresent("main");
            // All required files and folders inside "main" folder relative to current
            // folder
            System.out.println("Displaying all files with directory structure in ascending order\n");
            // listFilesInDirectory displays files along with folder structure
            List<String> filesListNames = ((FileOperations) FileOperations).listFilesInDirectory(path, 0, new ArrayList<String>());
            System.out.println("Displaying all files in ascending order\n");
            Collections.sort(filesListNames);
            filesListNames.stream().forEach(System.out::println);
}
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());
    // Recursively indent and display 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;
    }
```

Output:

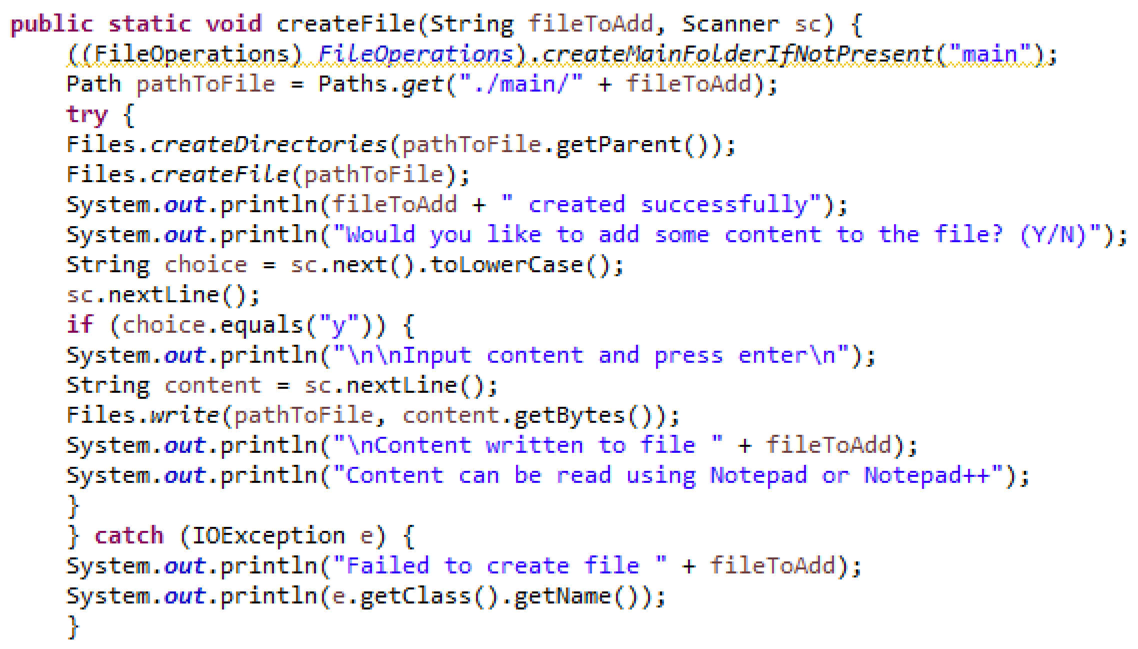


```
------Select any option number from below and press Enter -------

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- CourseProject
|-- FSD
`-- Gitcode
  `-- with
    `-- folder
      `-- creation
        |-- git_file.txt




|-- Gitproject
|-- JAVAproject
|-- P1.txt
|-- P2.txt
|-- Prog1.txt
|-- Prog2.txt
|-- Prog3.txt
|-- programs
|-- ProjectPrograms
|-- SimplilearnFolder
```



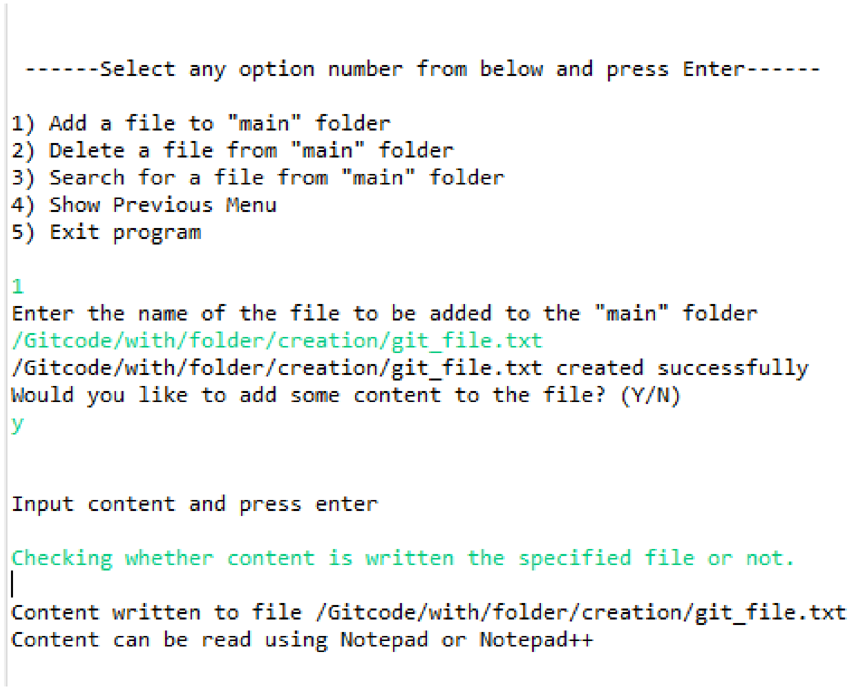
```
Displaying all files in ascending order

CourseProject
FSD
Gitcode
Gitproject
JAVAproject
P1.txt
P2.txt
Prog1.txt
Prog2.txt
Prog3.txt
ProjectPrograms
SimplilearnFolder
creation
folder
git_file.txt
programs
with
```

Step 5(iii): Writing method to create a file/folder as specified by user input.

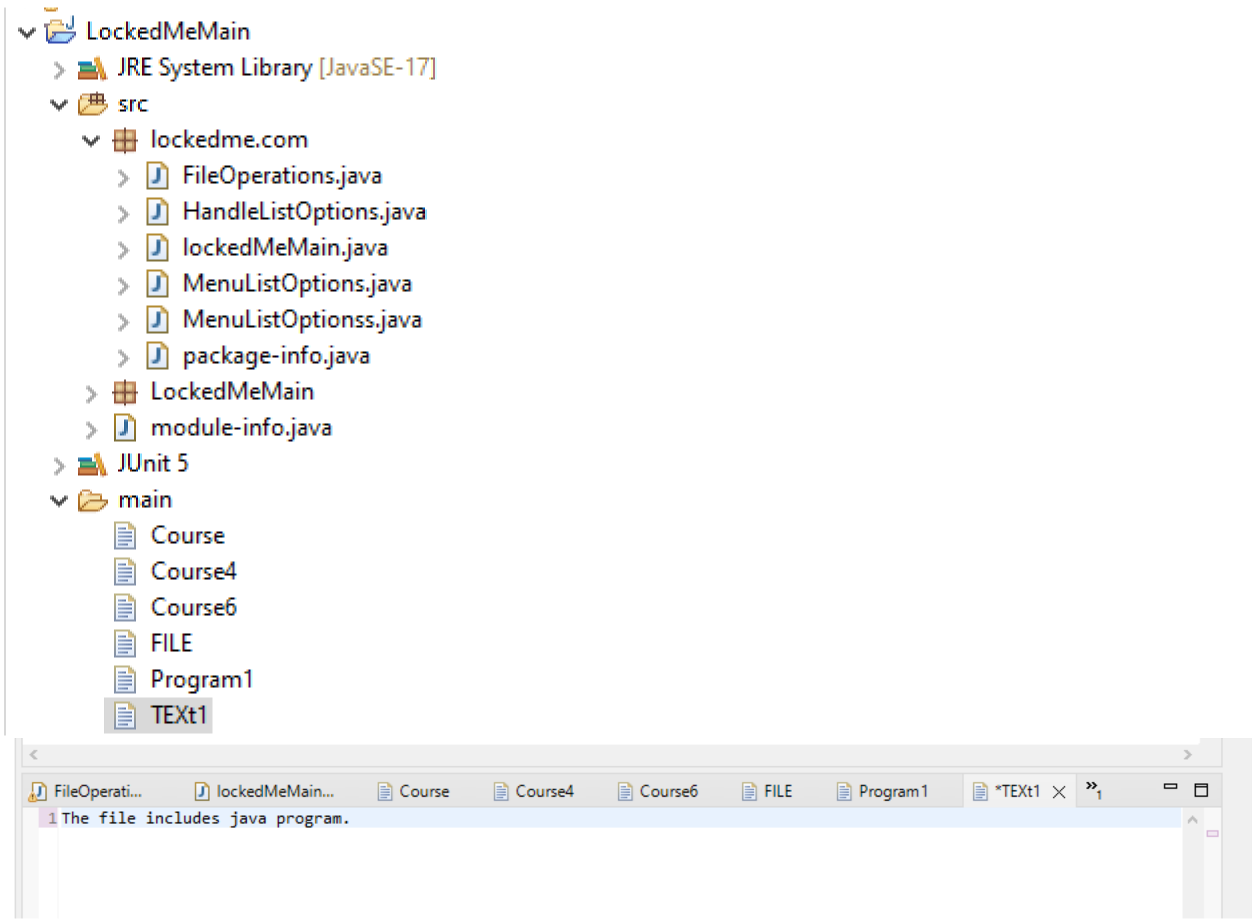


```java
public static void createFile(String fileToAdd, Scanner sc) {
    ((FileOperations) FileOperations).createMainFolderIfNotPresent("main");
    Path pathToFile = Paths.get("./main/" + fileToAdd);
    try {
    Files.createDirectories(pathToFile.getParent());
    Files.createFile(pathToFile);
    System.out.println(fileToAdd + " created successfully");
    System.out.println("Would you like to add some content to the file? (Y/N)");
    String choice = sc.next().toLowerCase();
    sc.nextLine();
    if (choice.equals("y")) {
    System.out.println("\n\nInput content and press enter\n");
    String content = sc.nextLine();
    Files.write(pathToFile, content.getBytes());
    System.out.println("\nContent written to file " + fileToAdd);
    System.out.println("Content can be read using Notepad or Notepad++");
    }
    } catch (IOException e) {
    System.out.println("Failed to create file " + fileToAdd);
    System.out.println(e.getClass().getName());
    }
```

Output:



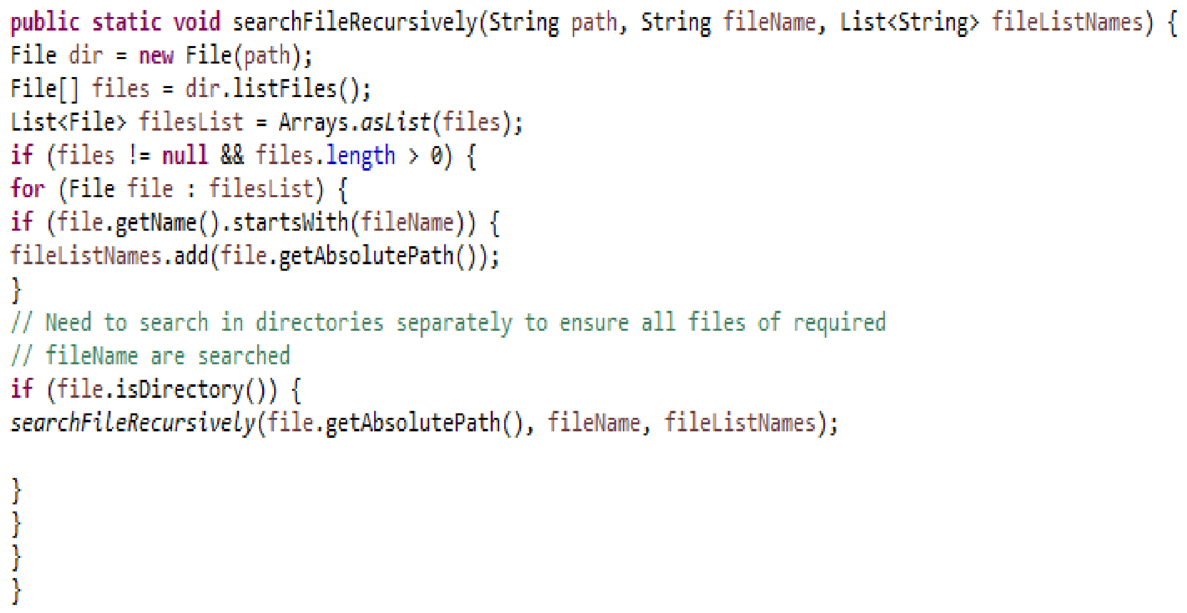
```
 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

1
Enter the name of the file to be added to the "main" folder
/Gitcode/with/folder/creation/git_file.txt
/Gitcode/with/folder/creation/git_file.txt created successfully
Would you like to add some content to the file? (Y/N)
y


Input content and press enter

Checking whether content is written the specified file or not.
|
Content written to file /Gitcode/with/folder/creation/git_file.txt
Content can be read using Notepad or Notepad++
```



```
The file includes java program.
```

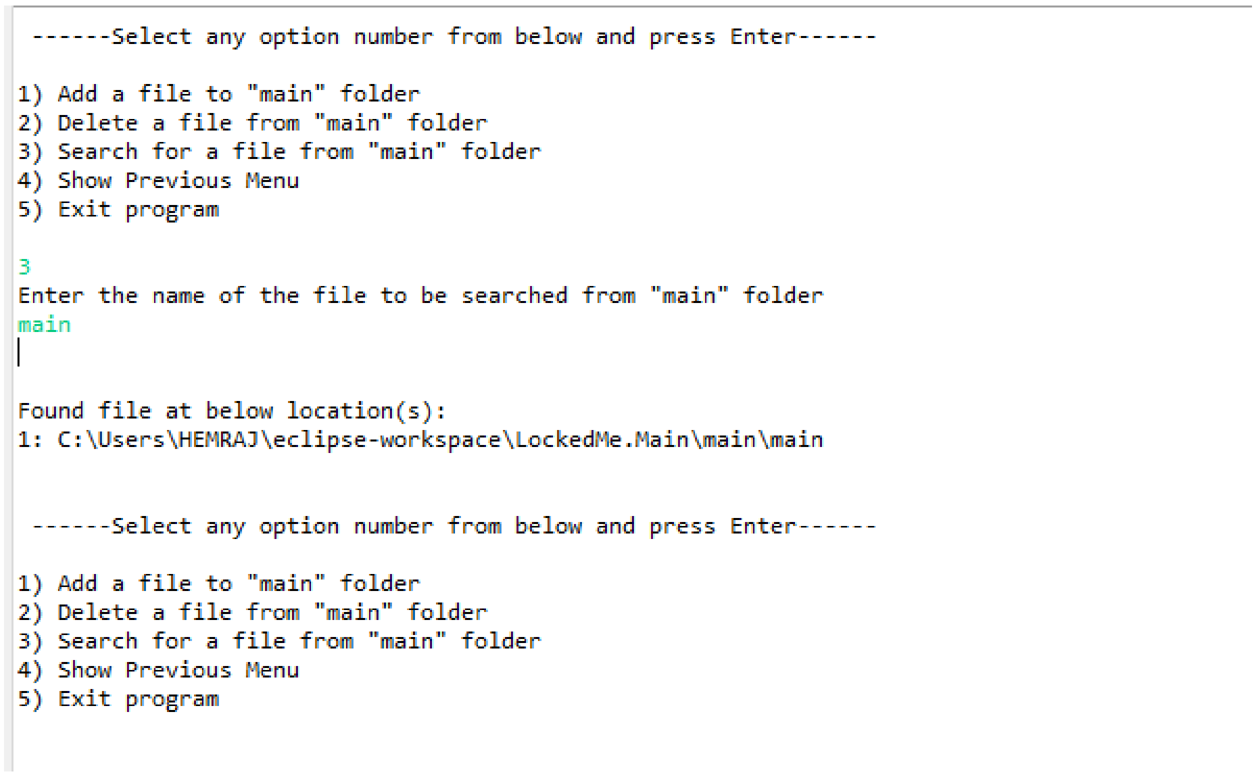
Step 5(iv): Writing method to search for all files as specified by user input in "main" folder and its subfolders.

```java
public static List<String> displayFileLocations(String fileName, String path) {
   List<String> fileListNames = new ArrayList<>();
   ((FileOperations) FileOperations).searchFileRecursively(path, fileName, fileListNames);
   if (fileListNames.isEmpty()) {
   System.out.println("\n\n***** Couldn't find any file with given file name \"" + fileName + "\" *****\n\n");
   } else {
   System.out.println("\n\nFound file at 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;
   }
```



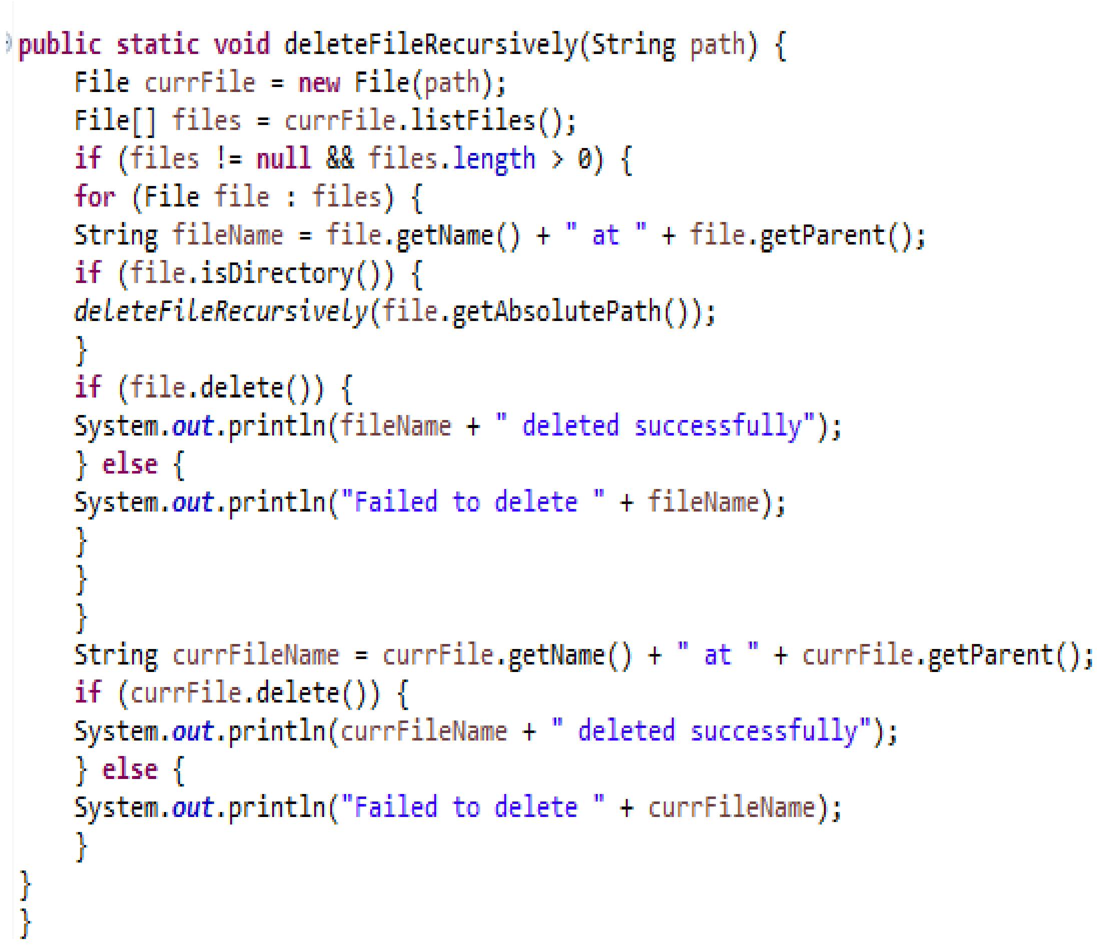
```java
public static void searchFileRecursively(String path, String fileName, List<String> fileListNames) {
File dir = new File(path);
File[] files = dir.listFiles();
List<File> filesList = Arrays.asList(files);
if (files != null && files.length > 0) {
for (File file : filesList) {
if (file.getName().startsWith(fileName)) {
fileListNames.add(file.getAbsolutePath());
}
// Need to search in directories separately to ensure all files of required
// fileName are searched
if (file.isDirectory()) {
searchFileRecursively(file.getAbsolutePath(), fileName, fileListNames);

}
}
}
}
```

Output:



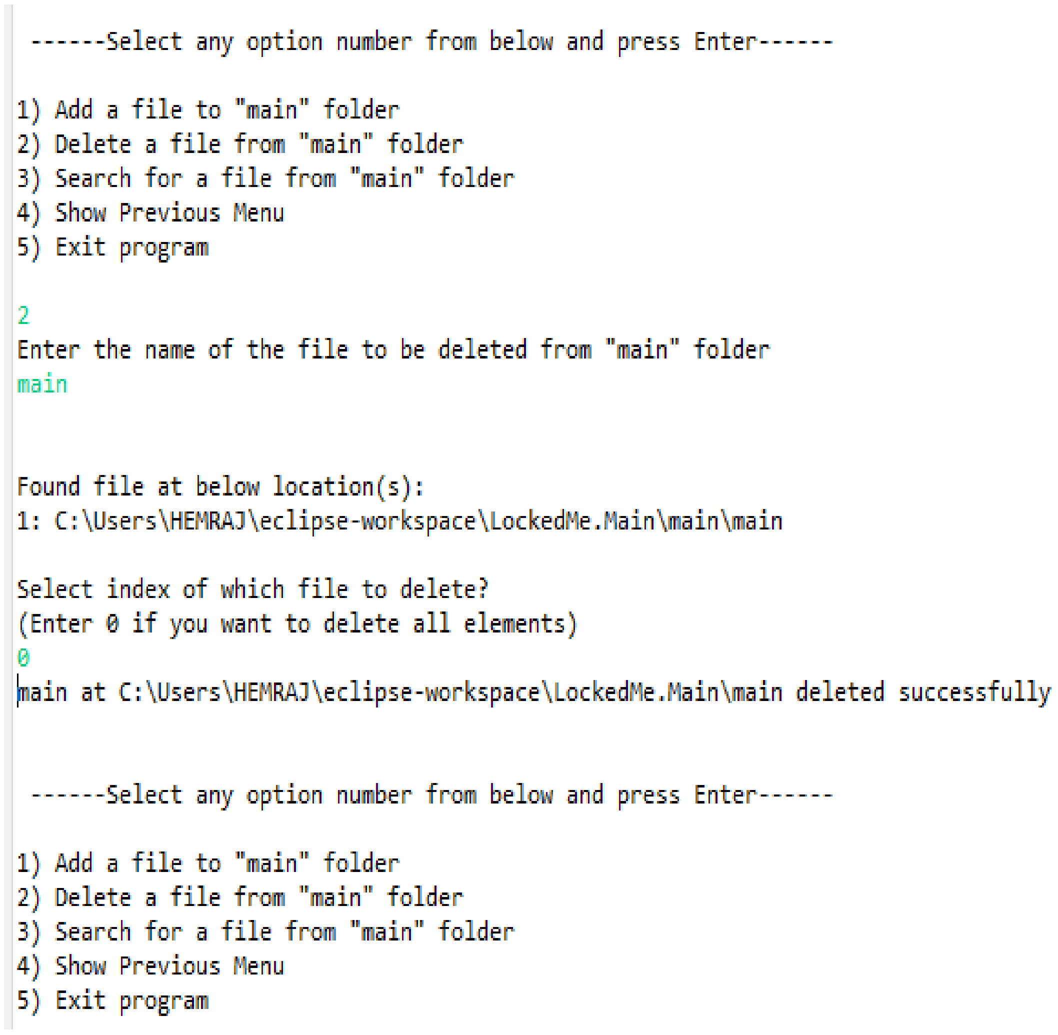
```
 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
main


Found file at below location(s):
1: C:\Users\HEMRAJ\eclipse-workspace\LockedMe.Main\main\main


 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program
```

Step 5(v): Writing method to delete file/folder specified by user input in "main" folder and its subfolders. It uses the searchFilesRecursively method and prompts user to specify which index to delete. If folder selected, all its child files and folder will be deleted recursively. If user wants to delete all the files specified after the search, they can input value 0.



```java
public static void deleteFileRecursively(String path) {
    File currFile = new File(path);
    File[] files = currFile.listFiles();
    if (files != null && files.length > 0) {
    for (File file : files) {
    String fileName = file.getName() + " at " + file.getParent();
    if (file.isDirectory()) {
    deleteFileRecursively(file.getAbsolutePath());
    }
    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);
    }
}
}
```

Output:



```
 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

2
Enter the name of the file to be deleted from "main" folder
main


Found file at below location(s):
1: C:\Users\HEMRAJ\eclipse-workspace\LockedMe.Main\main\main

Select index of which file to delete?
(Enter 0 if you want to delete all elements)
0
main at C:\Users\HEMRAJ\eclipse-workspace\LockedMe.Main\main deleted successfully


 ------Select any option number from below and press Enter------

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program
```

Step 6: Pushing the code to GitHub repository

- Open the command prompt and navigate to the folder where you have created your files.

  Cd< folder path>

- Initialize repository using following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . –m <commit message>

- Push the files to the folder you initially created using thefollowing command:

  git push –u origin master

Unique Selling Points of the Application

1. The application is designed to keep on running and taking user inputs even after exceptions occur. To terminate the application, must appropriate options needs to be selected.

2. The application can take any file/folder name as an input and even if the user wants create nested folder structure, user can just specify the relative path and the application takes care of creating the required folder structure.

3. User is also provide the option to write content if they want into the newly created file.

4. The application doesn't restrict user to specify the exact filename to search/delete file/folder. They can specify the starting input, and the program searches all files/folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

5. The application also allows user to delete folders which are not empty.

6. The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

7. When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.

7.1. Ascending order of folders first which have files sorted in them,

7.2. Ascending order of all files and folders inside the “main” folder.

8. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code. Application has been developed keeping in mind that there should be very less “hardcoding” of data.

Conclusions

Further enhancements to the application can be made which may include:

- Conditions to check if user is allowed to delete the file or add the file at the specific locations.
- Asking user to verify if they really want to delete the selected directory if it’s not empty.
- Retrieving files/folders by different criteria like Last Modified, Type, etc.
- Allowing user to append data to the file.