Company Lockers Private Ltd.

Product LockedMe.com
Prototype of the Application
Name : Amit Yadav
GitHub : https://github.com/amityadav872699/Project-Phase-1-JAVA-FSD.git

The prototype of the application is operated as a CLI (Command Line Program) without GUI. Its usage is to do file operations such as create new files along with content, delete a file or search a file from a specified directory and list them afterward in sorting order.

The implementation is done with the help of Java 8 and IDE IntelliJ.

Sprint Planning

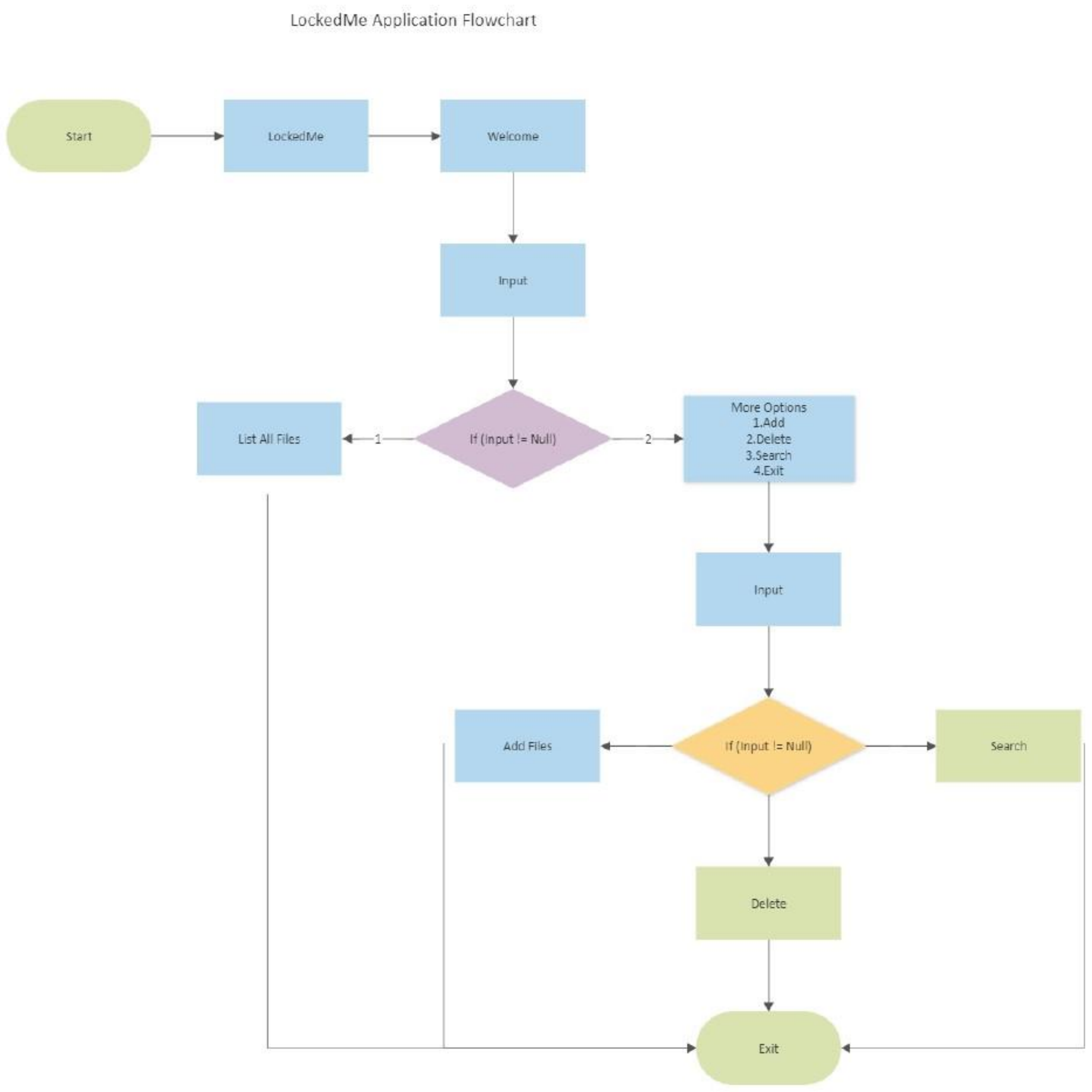
The Implementation is done in two sprints which are mentioned below:

Sprint 1:

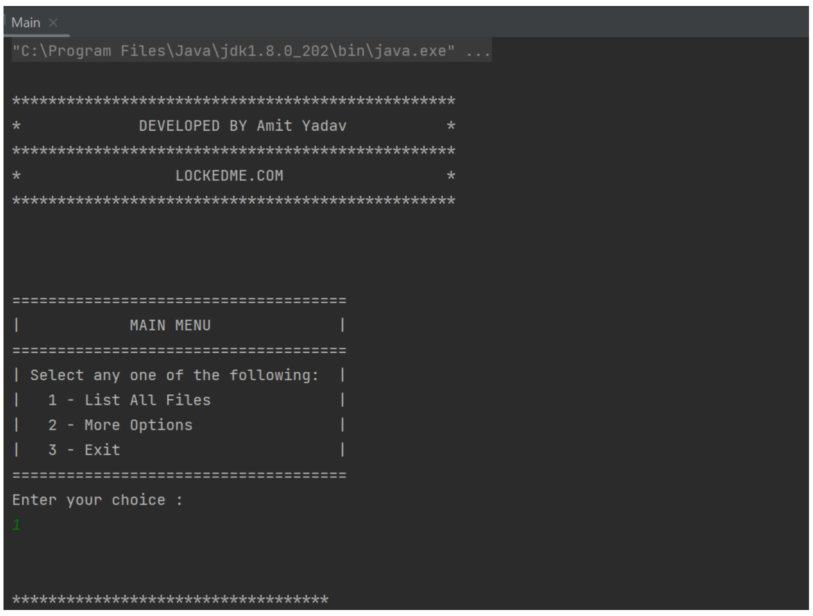
- Clarify the specification and requirements.
- Implement view content mechanism.
- Implement list of all files in sorted order.
- Implement functionality to close the program safely.

Sprint 2:

- Implement functionality to add create files along the content.
- Implement functionality to delete a file if it is present in that user specified directory.
- Implement functionality to search a file in the same directory.
- Documentation.

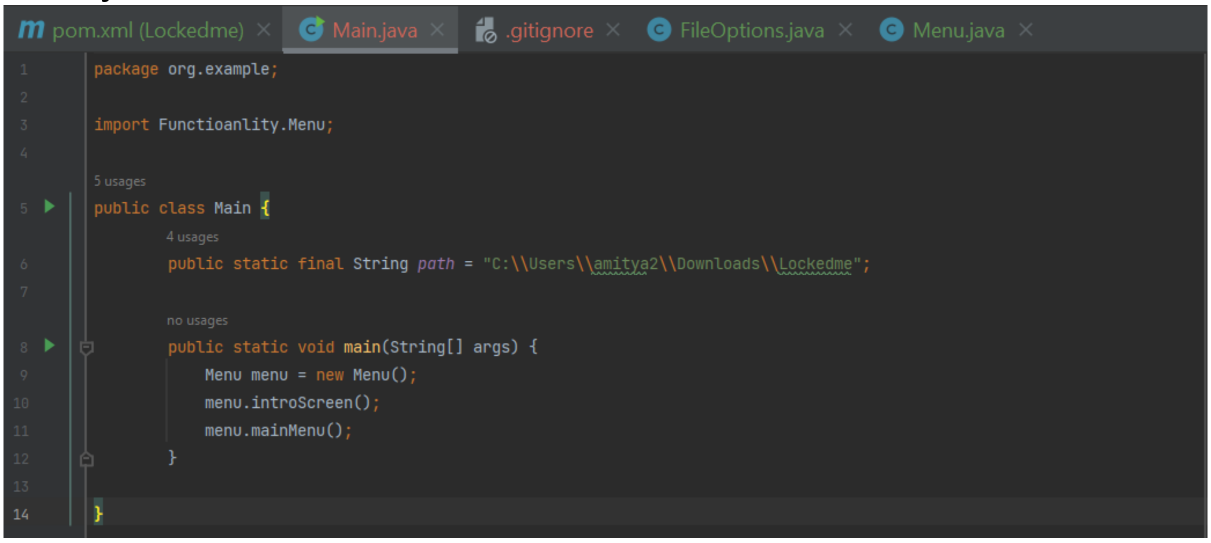


Output


```
Main
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

**************************************************
*             DEVELOPED BY Amit Yadav            *
**************************************************
*                 LOCKEDME.COM                   *
**************************************************




=====================================
|            MAIN MENU              |
=====================================
| Select any one of the following:  |
|   1 - List All Files              |
|   2 - More Options                |
|   3 - Exit                        |
=====================================
Enter your choice :
1


***********************************
```

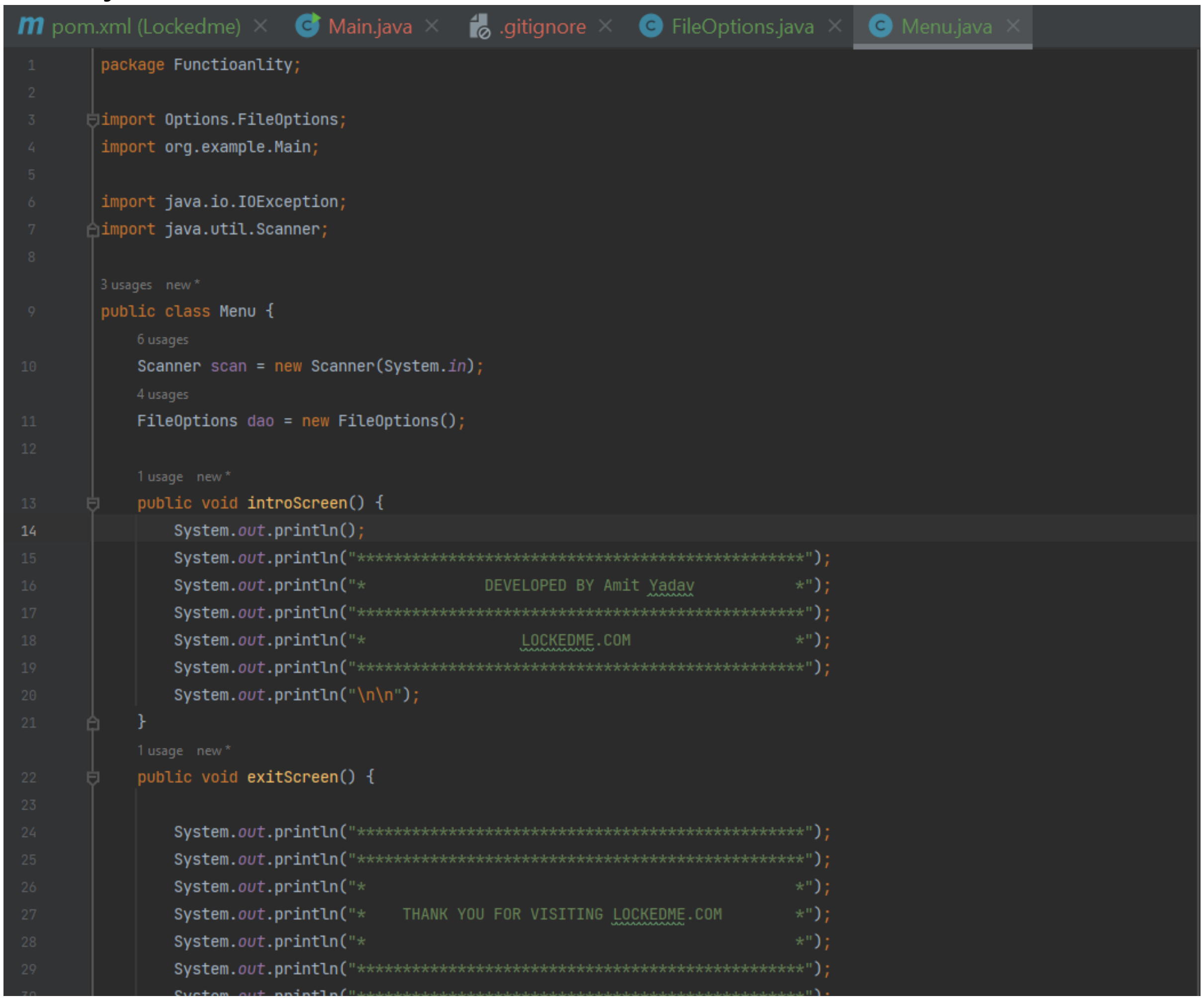
Source Code

Main. java

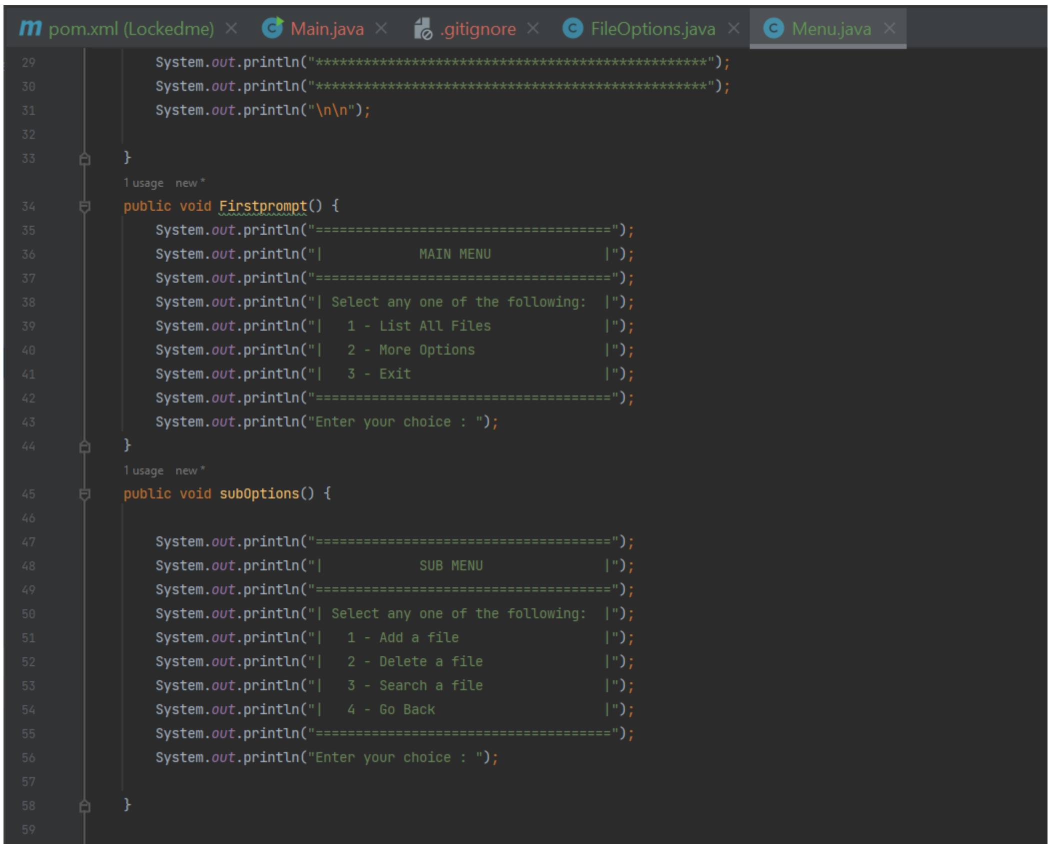


```java
package org.example;

import Functioanlity.Menu;

5 usages
public class Main {
        4 usages
        public static final String path = "C:\\Users\\amitya2\\Downloads\\Lockedme";

        no usages
        public static void main(String[] args) {
            Menu menu = new Menu();
            menu.introScreen();
            menu.mainMenu();
        }

}
```

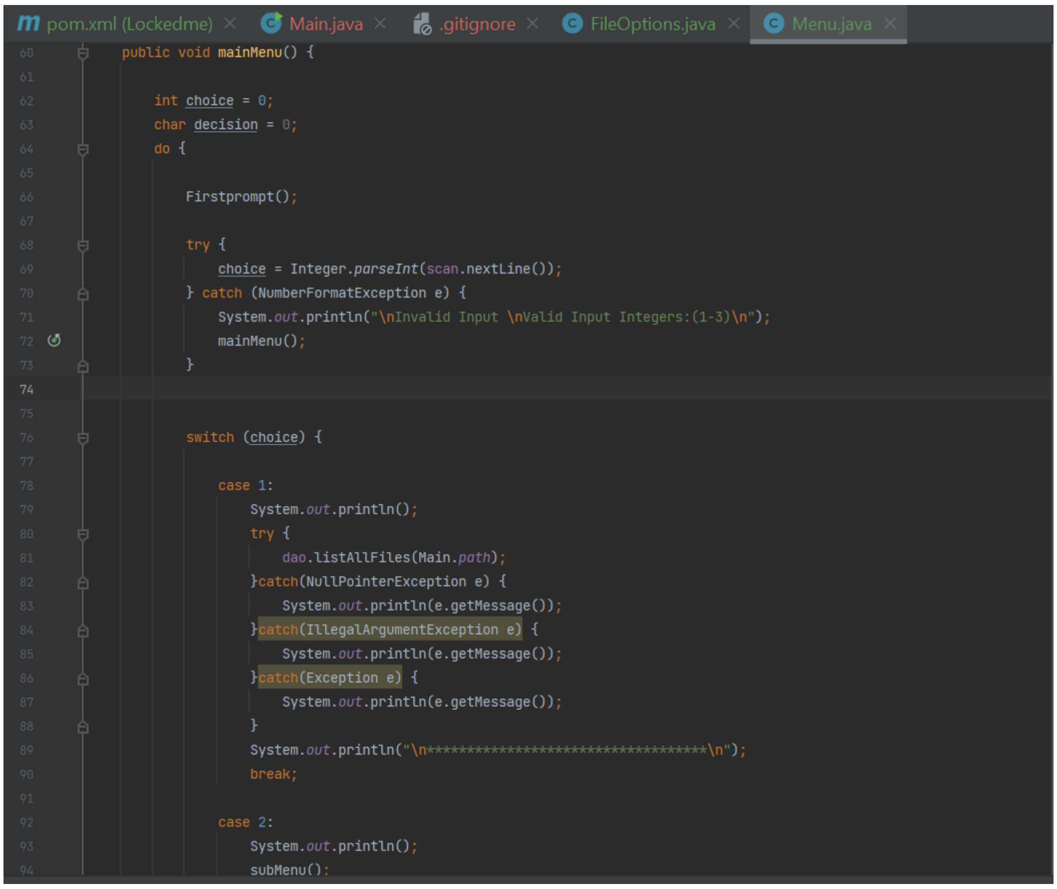
Menu.java



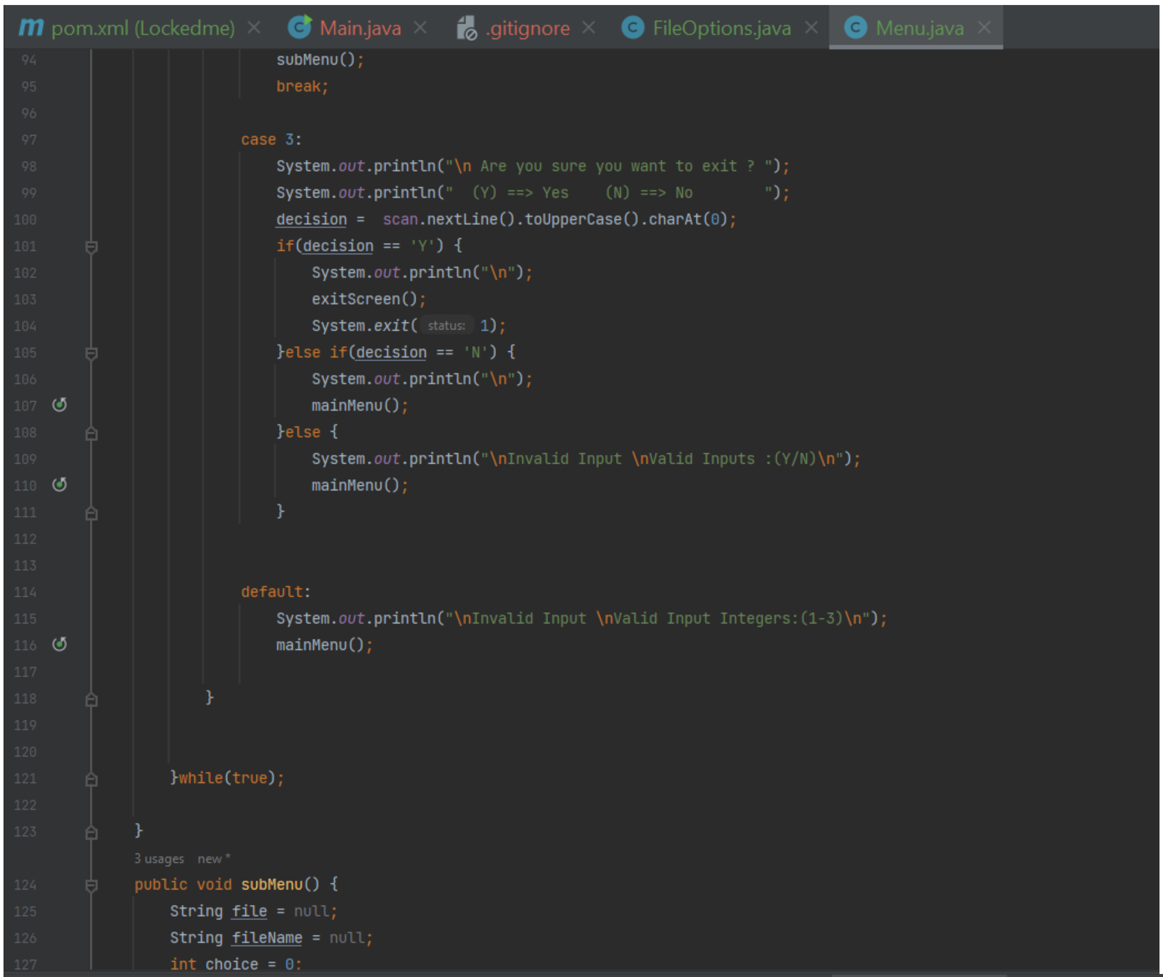
```java
package Functioanlity;

import Options.FileOptions;
import org.example.Main;

import java.io.IOException;
import java.util.Scanner;

3 usages   new *
public class Menu {
    6 usages
    Scanner scan = new Scanner(System.in);
    4 usages
    FileOptions dao = new FileOptions();

    1 usage   new *
    public void introScreen() {
        System.out.println();
        System.out.println("**************************************************");
        System.out.println("*             DEVELOPED BY Amit Yadav            *");
        System.out.println("**************************************************");
        System.out.println("*                 LOCKEDME.COM                   *");
        System.out.println("**************************************************");
        System.out.println("\n\n");
    }
    1 usage   new *
    public void exitScreen() {

        System.out.println("**************************************************");
        System.out.println("**************************************************");
        System.out.println("*                                                *");
        System.out.println("*    THANK YOU FOR VISITING LOCKEDME.COM         *");
        System.out.println("*                                                *");
        System.out.println("**************************************************");
```



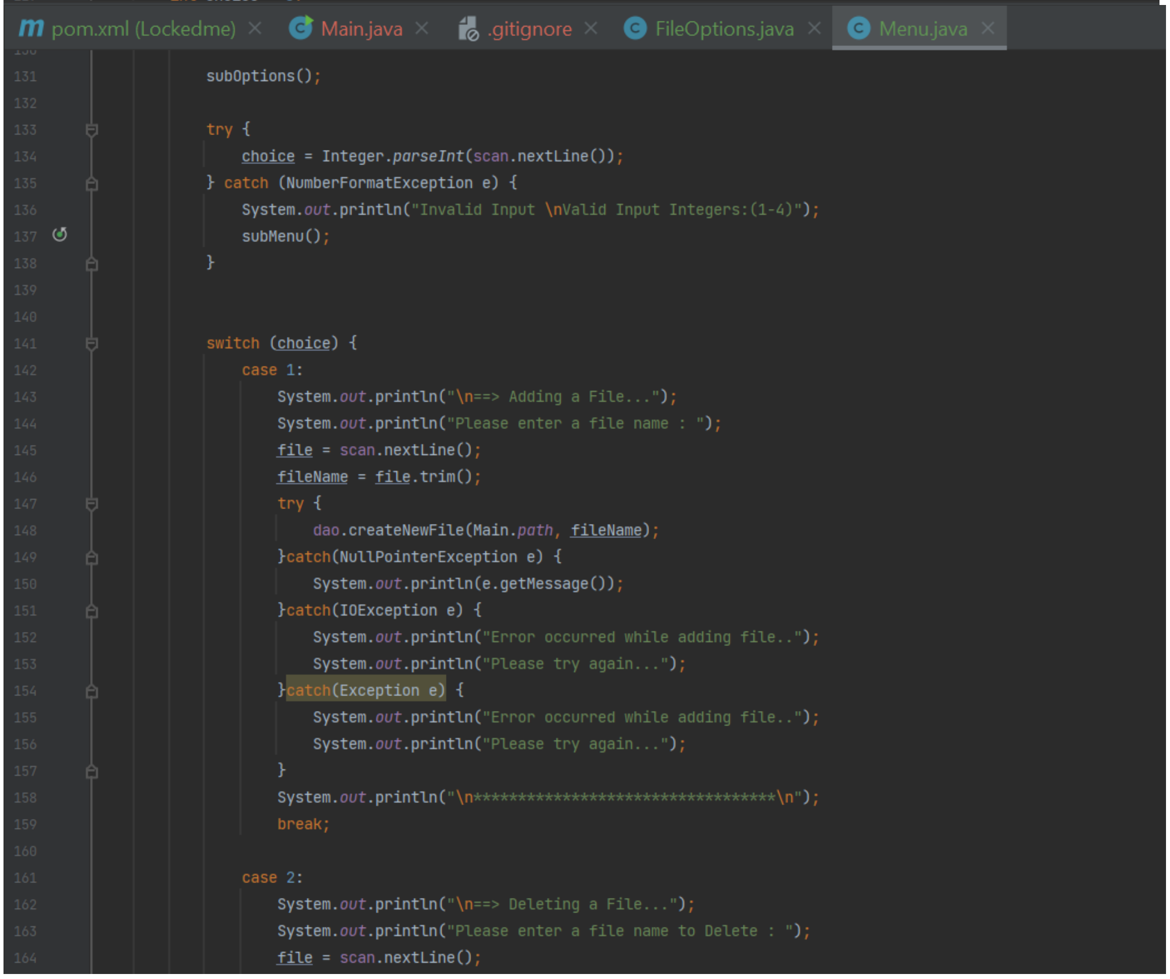
```java
        System.out.println("*************************************************");
        System.out.println("*************************************************");
        System.out.println("\n\n");

    }
    public void Firstprompt() {
        System.out.println("=====================================");
        System.out.println("|            MAIN MENU              |");
        System.out.println("=====================================");
        System.out.println("| Select any one of the following:  |");
        System.out.println("|   1 - List All Files              |");
        System.out.println("|   2 - More Options                |");
        System.out.println("|   3 - Exit                        |");
        System.out.println("=====================================");
        System.out.println("Enter your choice : ");
    }
    public void subOptions() {

        System.out.println("=====================================");
        System.out.println("|            SUB MENU               |");
        System.out.println("=====================================");
        System.out.println("| Select any one of the following:  |");
        System.out.println("|   1 - Add a file                  |");
        System.out.println("|   2 - Delete a file               |");
        System.out.println("|   3 - Search a file               |");
        System.out.println("|   4 - Go Back                     |");
        System.out.println("=====================================");
        System.out.println("Enter your choice : ");

    }

```



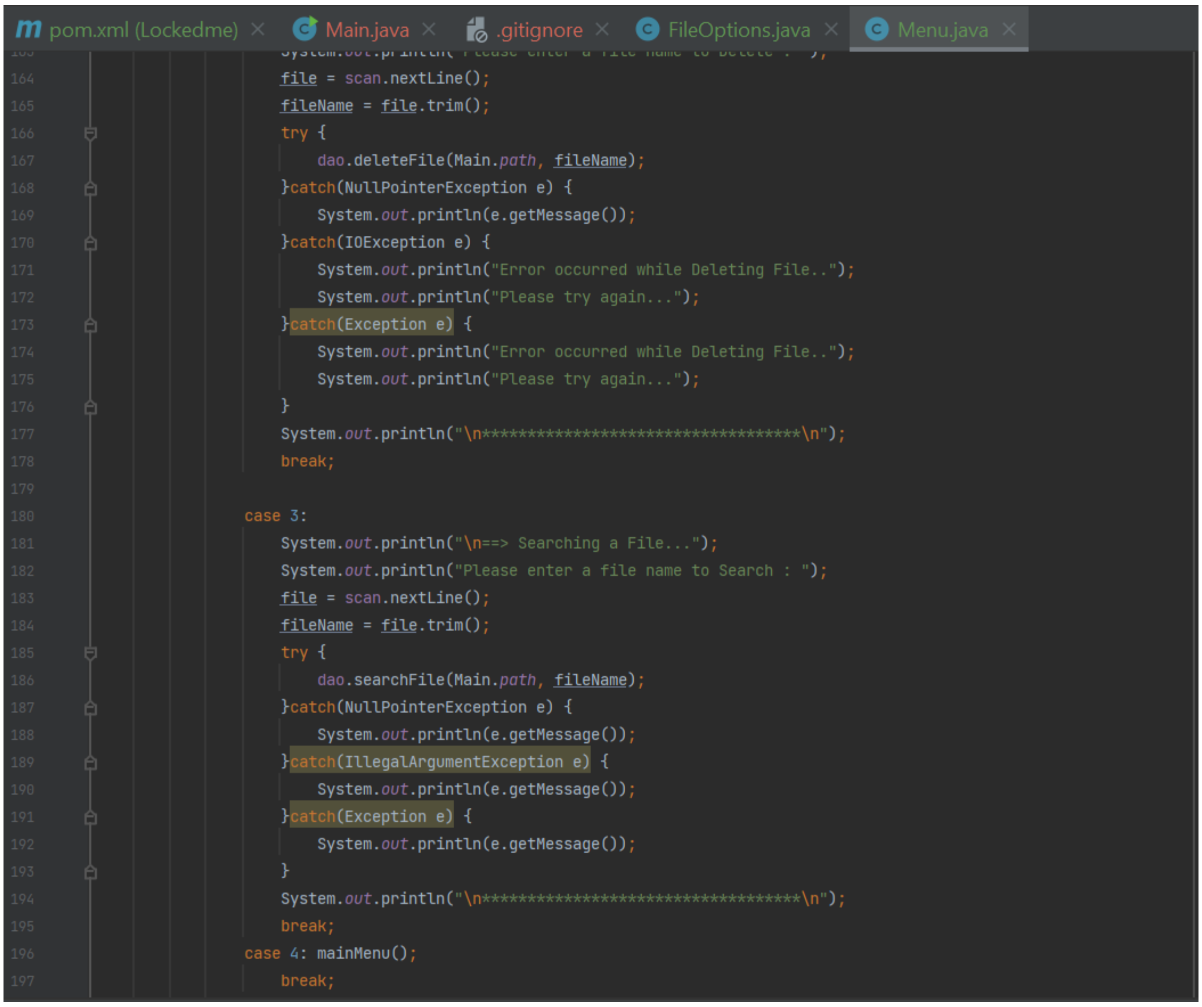
```java
    public void mainMenu() {

        int choice = 0;
        char decision = 0;
        do {

            Firstprompt();

            try {
                choice = Integer.parseInt(scan.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("\nInvalid Input \nValid Input Integers:(1-3)\n");
                mainMenu();
            }


            switch (choice) {

                case 1:
                    System.out.println();
                    try {
                        dao.listAllFiles(Main.path);
                    }catch(NullPointerException e) {
                        System.out.println(e.getMessage());
                    }catch(IllegalArgumentException e) {
                        System.out.println(e.getMessage());
                    }catch(Exception e) {
                        System.out.println(e.getMessage());
                    }
                    System.out.println("\n***********************************\n");
                    break;

                case 2:
                    System.out.println();
                    subMenu();
```



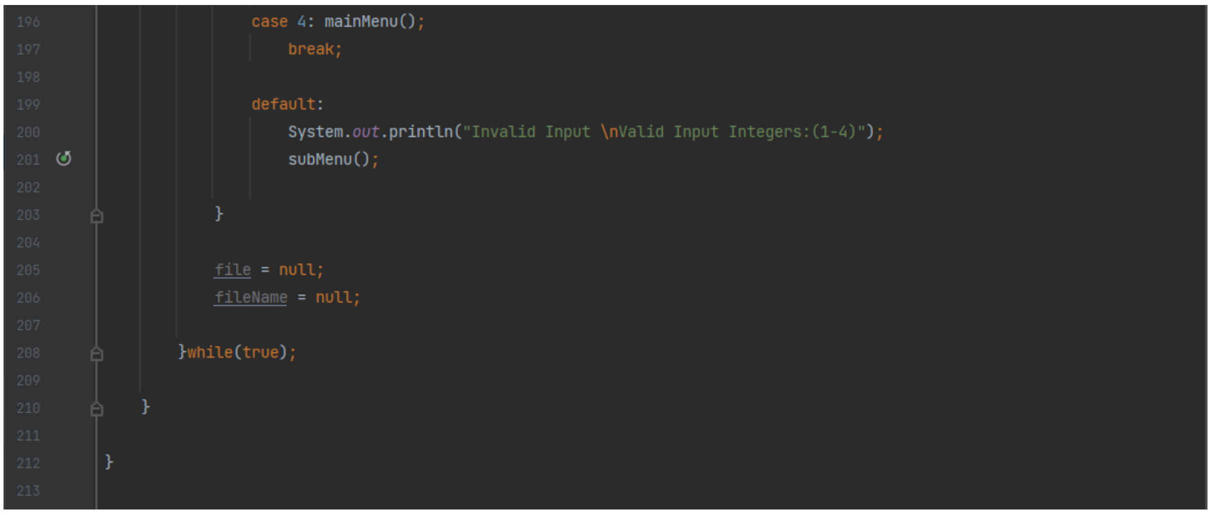
```java
                    subMenu();
                    break;

                case 3:
                    System.out.println("\n Are you sure you want to exit ? ");
                    System.out.println("  (Y) ==> Yes    (N) ==> No        ");
                    decision =  scan.nextLine().toUpperCase().charAt(0);
                    if(decision == 'Y') {
                        System.out.println("\n");
                        exitScreen();
                        System.exit( status: 1);
                    }else if(decision == 'N') {
                        System.out.println("\n");
                        mainMenu();
                    }else {
                        System.out.println("\nInvalid Input \nValid Inputs :(Y/N)\n");
                        mainMenu();
                    }


                default:
                    System.out.println("\nInvalid Input \nValid Input Integers:(1-3)\n");
                    mainMenu();

            }


        }while(true);

    }
    3 usages   new *
    public void subMenu() {
        String file = null;
        String fileName = null;
        int choice = 0;
```



```java
            subOptions();

            try {
                choice = Integer.parseInt(scan.nextLine());
            } catch (NumberFormatException e) {
                System.out.println("Invalid Input \nValid Input Integers:(1-4)");
                subMenu();
            }


            switch (choice) {
                case 1:
                    System.out.println("\n==> Adding a File...");
                    System.out.println("Please enter a file name : ");
                    file = scan.nextLine();
                    fileName = file.trim();
                    try {
                        dao.createNewFile(Main.path, fileName);
                    }catch(NullPointerException e) {
                        System.out.println(e.getMessage());
                    }catch(IOException e) {
                        System.out.println("Error occurred while adding file..");
                        System.out.println("Please try again...");
                    }catch(Exception e) {
                        System.out.println("Error occurred while adding file..");
                        System.out.println("Please try again...");
                    }
                    System.out.println("\n***********************************\n");
                    break;

                case 2:
                    System.out.println("\n==> Deleting a File...");
                    System.out.println("Please enter a file name to Delete : ");
                    file = scan.nextLine();
```

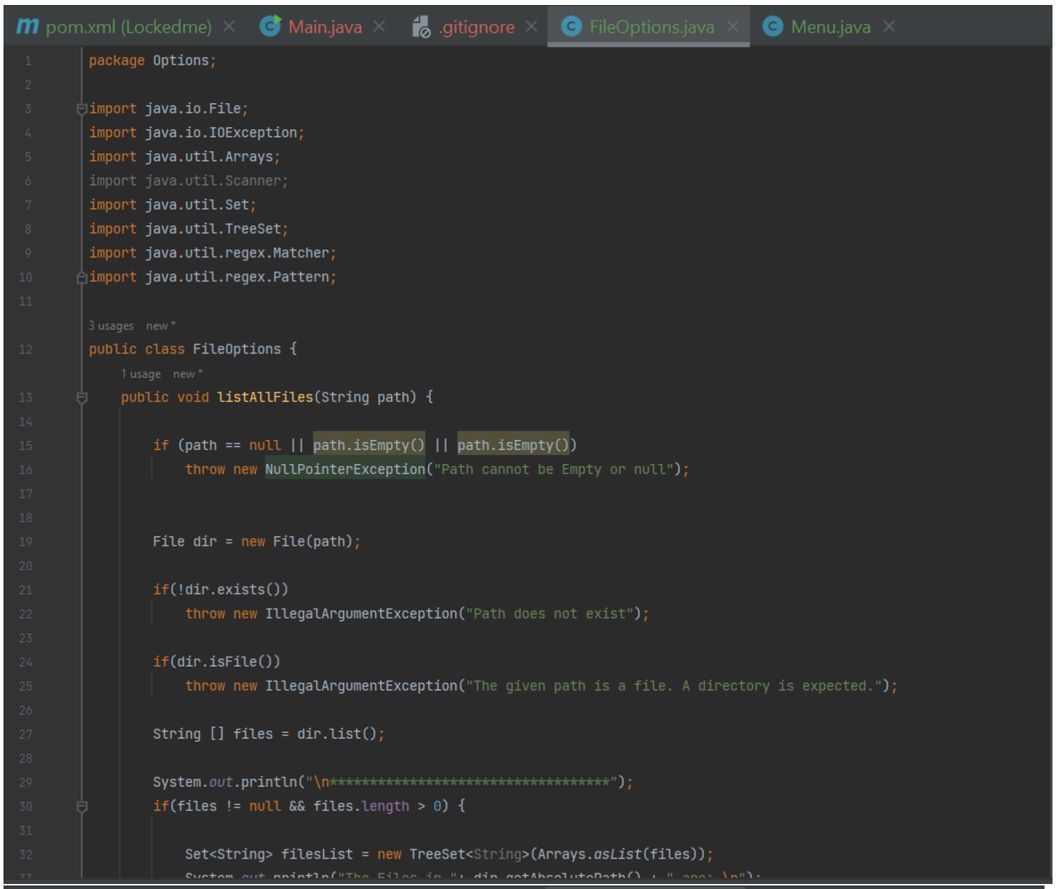


```java
                    file = scan.nextLine();
                    fileName = file.trim();
                    try {
                        dao.deleteFile(Main.path, fileName);
                    }catch(NullPointerException e) {
                        System.out.println(e.getMessage());
                    }catch(IOException e) {
                        System.out.println("Error occurred while Deleting File..");
                        System.out.println("Please try again...");
                    }catch(Exception e) {
                        System.out.println("Error occurred while Deleting File..");
                        System.out.println("Please try again...");
                    }
                    System.out.println("\n*************************************\n");
                    break;

                case 3:
                    System.out.println("\n==> Searching a File...");
                    System.out.println("Please enter a file name to Search : ");
                    file = scan.nextLine();
                    fileName = file.trim();
                    try {
                        dao.searchFile(Main.path, fileName);
                    }catch(NullPointerException e) {
                        System.out.println(e.getMessage());
                    }catch(IllegalArgumentException e) {
                        System.out.println(e.getMessage());
                    }catch(Exception e) {
                        System.out.println(e.getMessage());
                    }
                    System.out.println("\n*************************************\n");
                    break;
                case 4: mainMenu();
                    break;
```

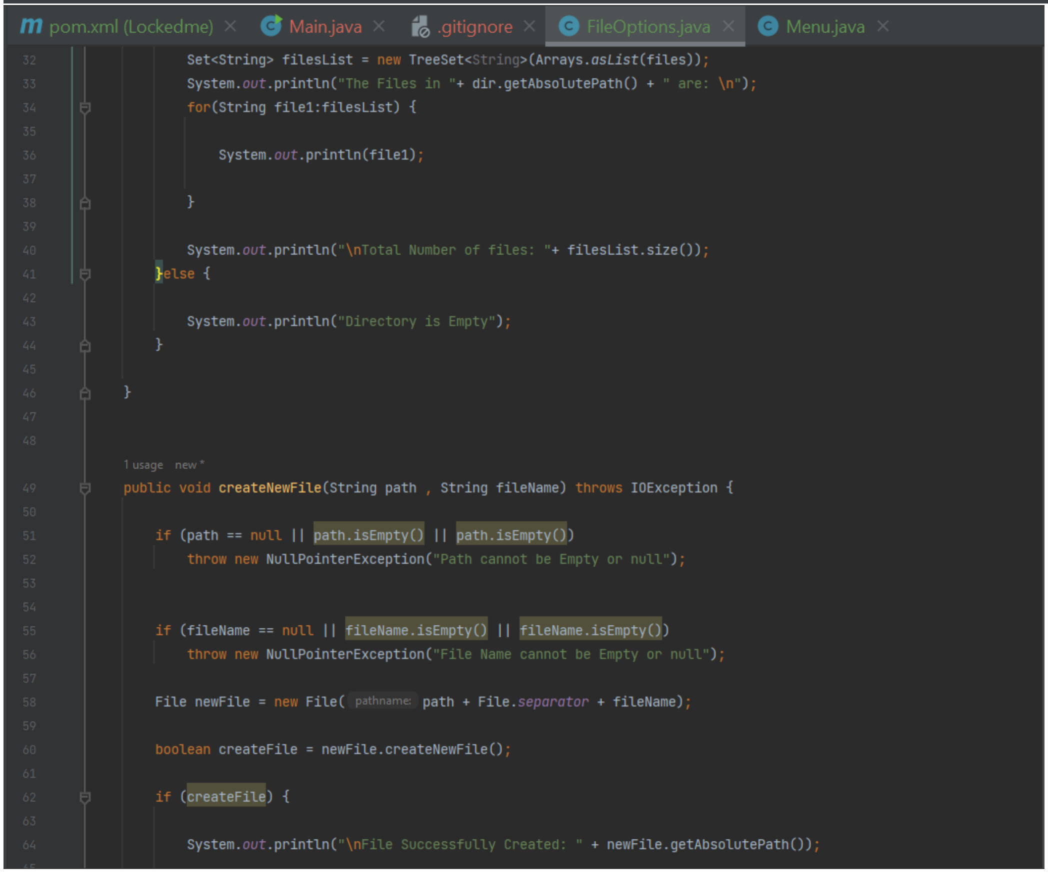


```java
                case 4: mainMenu();
                    break;

                default:
                    System.out.println("Invalid Input \nValid Input Integers:(1-4)");
                    subMenu();

            }

            file = null;
            fileName = null;

        }while(true);

    }

}
```

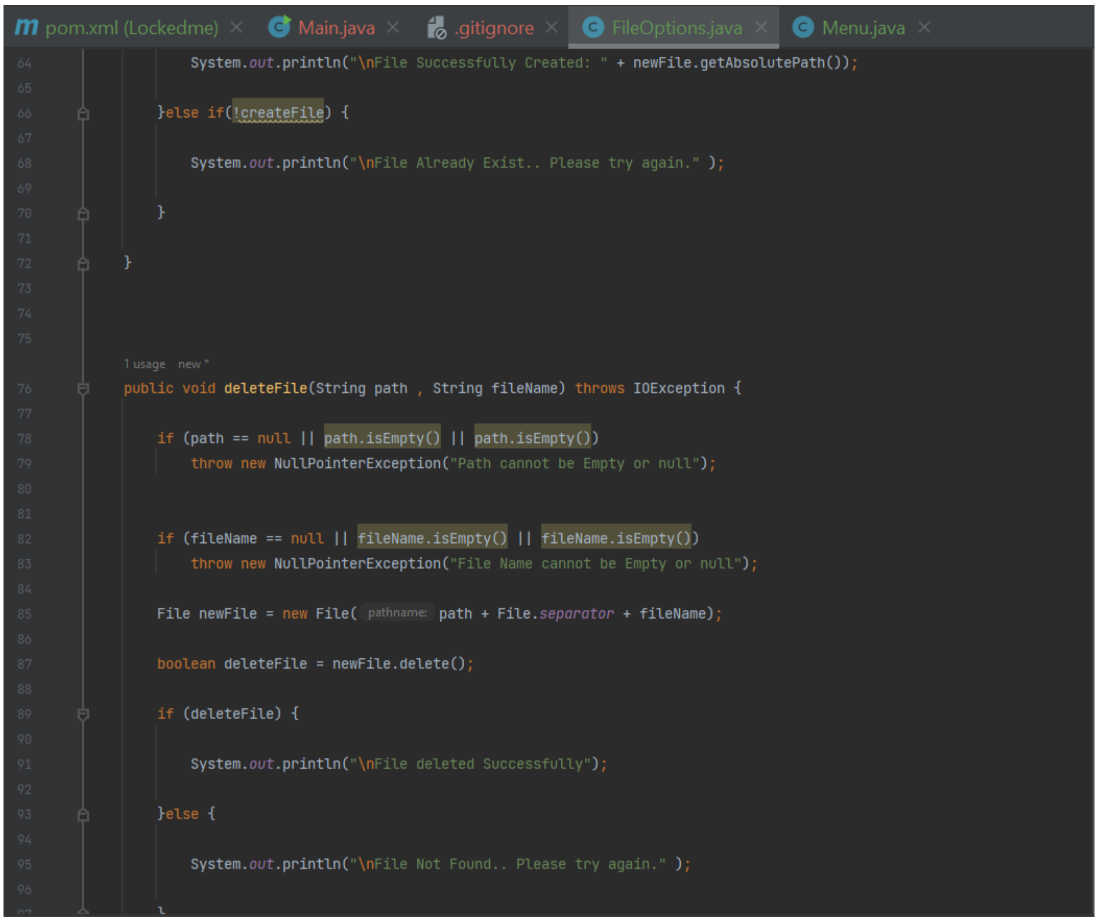
FileOptions.java



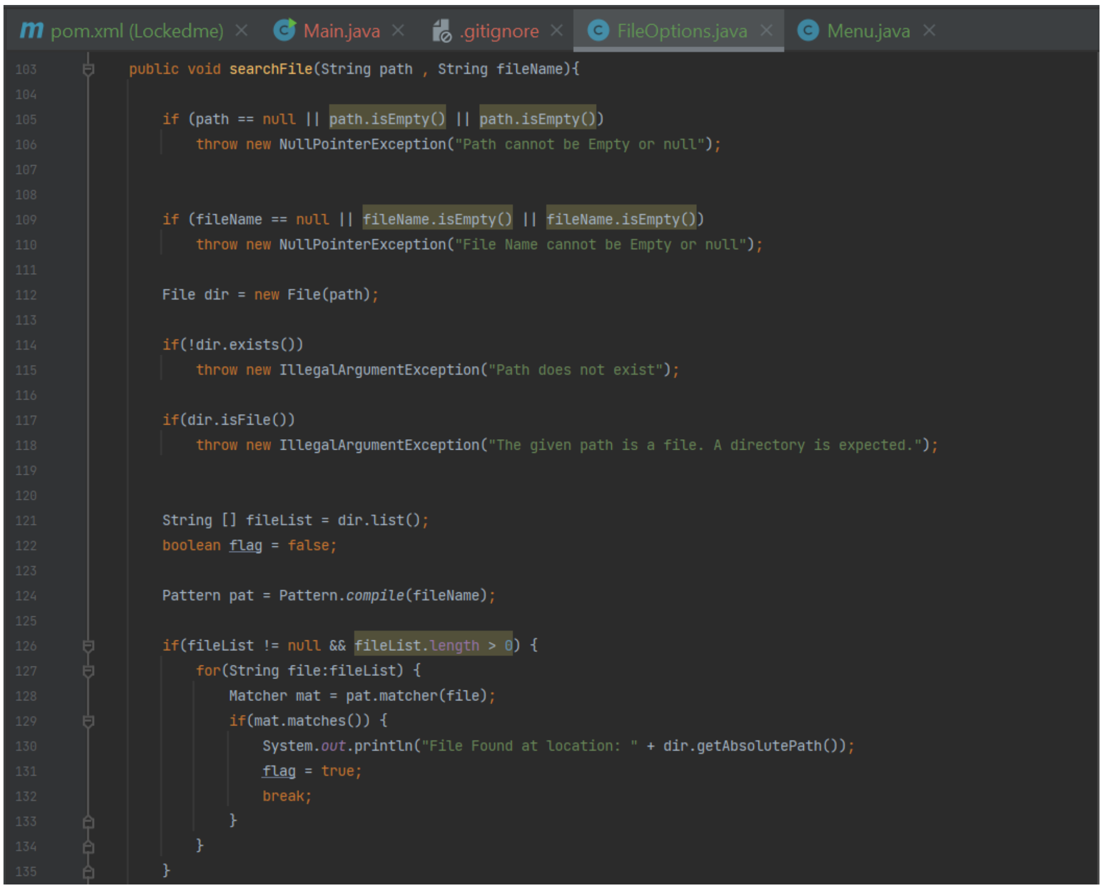
```java
package Options;

import java.io.File;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

3 usages   new *
public class FileOptions {
    1 usage   new *
    public void listAllFiles(String path) {

        if (path == null || path.isEmpty() || path.isEmpty())
            throw new NullPointerException("Path cannot be Empty or null");


        File dir = new File(path);

        if(!dir.exists())
            throw new IllegalArgumentException("Path does not exist");

        if(dir.isFile())
            throw new IllegalArgumentException("The given path is a file. A directory is expected.");

        String [] files = dir.list();

        System.out.println("\n***********************************");
        if(files != null && files.length > 0) {

            Set<String> filesList = new TreeSet<String>(Arrays.asList(files));
```



```java
            Set<String> filesList = new TreeSet<String>(Arrays.asList(files));
            System.out.println("The Files in "+ dir.getAbsolutePath() + " are: \n");
            for(String file1:filesList) {

                System.out.println(file1);

            }

            System.out.println("\nTotal Number of files: "+ filesList.size());
        }else {

            System.out.println("Directory is Empty");
        }

    }


    1 usage   new *
    public void createNewFile(String path , String fileName) throws IOException {

        if (path == null || path.isEmpty() || path.isEmpty())
            throw new NullPointerException("Path cannot be Empty or null");


        if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
            throw new NullPointerException("File Name cannot be Empty or null");

        File newFile = new File( pathname: path + File.separator + fileName);

        boolean createFile = newFile.createNewFile();

        if (createFile) {

            System.out.println("\nFile Successfully Created: " + newFile.getAbsolutePath());
```



```java
            System.out.println("\nFile Successfully Created: " + newFile.getAbsolutePath());

        }else if(!createFile) {

            System.out.println("\nFile Already Exist.. Please try again." );

        }

    }



    1 usage  new *
    public void deleteFile(String path , String fileName) throws IOException {

        if (path == null || path.isEmpty() || path.isEmpty())
            throw new NullPointerException("Path cannot be Empty or null");


        if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
            throw new NullPointerException("File Name cannot be Empty or null");

        File newFile = new File( pathname: path + File.separator + fileName);

        boolean deleteFile = newFile.delete();

        if (deleteFile) {

            System.out.println("\nFile deleted Successfully");

        }else {

            System.out.println("\nFile Not Found.. Please try again." );

        }
```



```java
    public void searchFile(String path , String fileName){

        if (path == null || path.isEmpty() || path.isEmpty())
            throw new NullPointerException("Path cannot be Empty or null");


        if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
            throw new NullPointerException("File Name cannot be Empty or null");

        File dir = new File(path);

        if(!dir.exists())
            throw new IllegalArgumentException("Path does not exist");

        if(dir.isFile())
            throw new IllegalArgumentException("The given path is a file. A directory is expected.");


        String [] fileList = dir.list();
        boolean flag = false;

        Pattern pat = Pattern.compile(fileName);

        if(fileList != null && fileList.length > 0) {
            for(String file:fileList) {
                Matcher mat = pat.matcher(file);
                if(mat.matches()) {
                    System.out.println("File Found at location: " + dir.getAbsolutePath());
                    flag = true;
                    break;
                }
            }
        }
```

Conclusion

1: The prototype is robust and platform independent.

2: User can easily use the prototype and safely exit out of it.

3: The prototype has a good interface with CLI (Command Line Interface).

4: As a developer, we can enhance it by introducing several new features such as appending in a file or overwriting a file and the file details for which user selected.

5: This prototype though is robust but user can only interact it with terminal or CLI so we can develop a good GUI interface for more better user-friendly.