Project Details

Project Name : Virtual Key for Repositories

Project Version : 1.0

Project Abstract :

A console application developed using J2SE concepts as a prototype of LockedMe.com . This application will provide a CUI (Command User Interface) to users .Where user can see menus with different options to handle files and directories. By using this application user can do following things,

1. Creates a root directory by creating a bin folder in the file application.
2. It gives user the capability to list files/folder present in this root directory as well as add and remove files from this root directory.
3. It also gives the user a capability to search all the files with a particular name in the directory.

Developer Details:

Developer Name: Bibhu Ranjan Mohanty

Email : bib.mhnt@gmail.com

Outlines:

Github Link : https://github.com/bibhu-otz/VirtualKey

1. Sprint Planning

No. of Sprints Planned: 2

Sprint 1:

In the first sprint a basic structure to the application was developed, with the simple capability of listing the files present in the root directory.

The application has a three tier structure.

The presentation layer, the business validation layer and the data access layer.

1) The presentation layer is the console, and the void main of the application corresponds to the presentation layer. It is responsible for taking the inputs from the user as well as showing the outcome/output to the user. Thus it is the layer which communicates with the user.

2) The Business Validation layer is responsible for validating the input from the user.

3) The data access layer process this user input and performs the operation as per the user input.

This basic structure was created using various classes and interfaces. Also the logic to list the files present in the root directory was developed.

Sprint 2:

The above application was enhanced to incorporate additional features like addition/ removal/ search of a file.

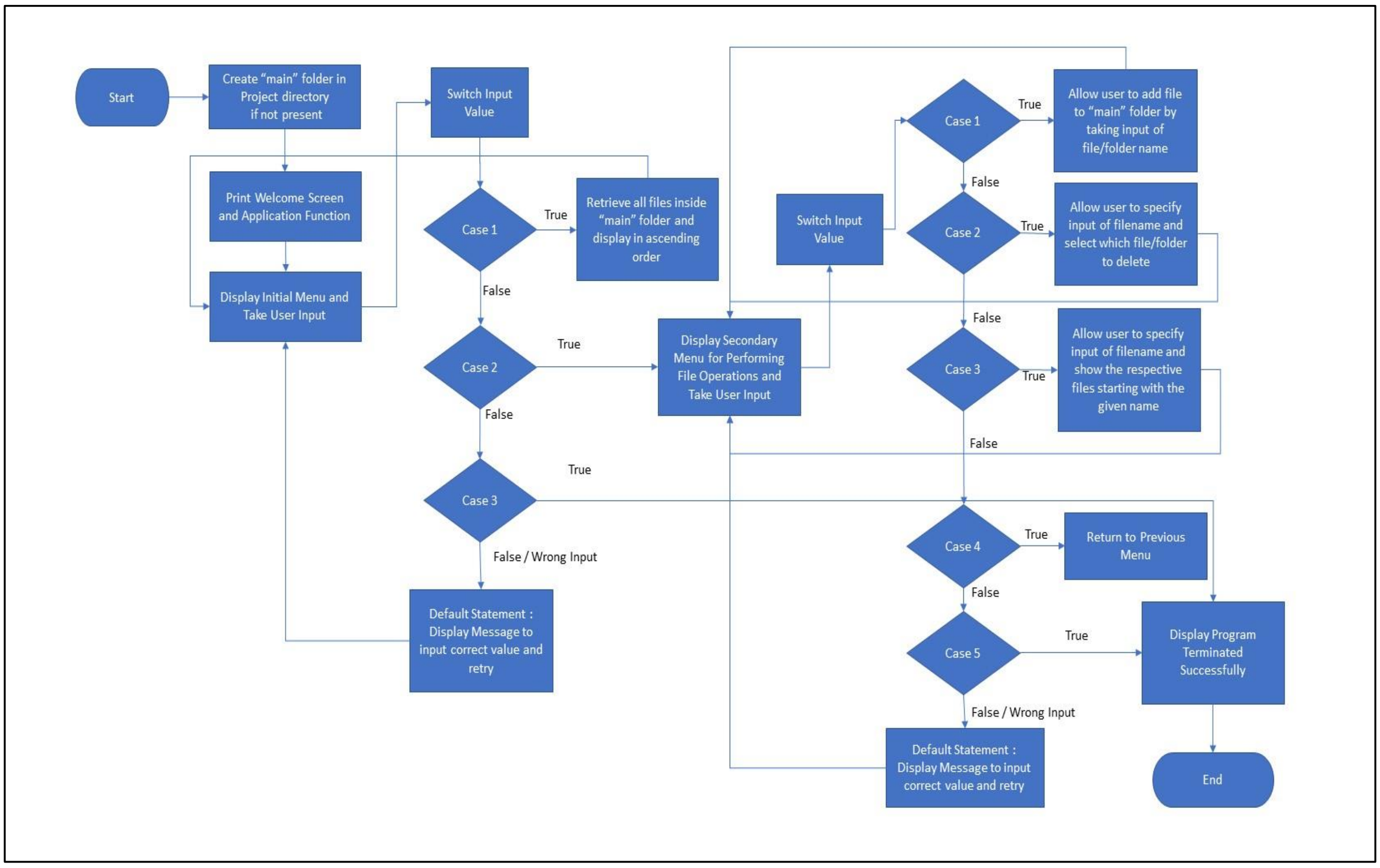
This included creating a submenu. And adding the above options. New methods had to be created for this purpose.

2. Core concepts used in project

- Collections framework
- File Handling
- Sorting
- Exception Handling
- Interface & Inheritance

3. Flow of the application

Flow of Application.pptx


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
False
Case 3
True
False / Wrong Input
Default Statement : Display Message to input correct value and retry
Display Secondary Menu for Performing File Operations and Take User Input
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

4. Demonstrating the product capabilities, appearance, and user interactions

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

I	Creating the project in Eclipse
II	Writing a program in Java for the entry point of the application (**LockedMe.java**) under org.virtualkey package
III	Create a sub package org.virtualkey.entities, then create two class named as Directory , WelcomeScreen and create an Interface named as Screen
	Class Name — **Purpose**
	Directory — This class is responsible for all type of operation realted to directory like listing of all file etc.
	WelcomeScreen — This class is responsible for displaying Welcome Screen
	Interface Name — **Purpose**
	Screen — This interface having some common method which are implemented by other classes for performing file handling operations.
IV	Create a sub package org.virtualkey.operations, then create three class named as DirectoryService , FileOption and ScreenService
VI	Pushing the code to GitHub repository

Step 1: Creating a new project in Eclipse

- Open Eclipse
- Goto File -> New -> Project -> java Project -> Next
- Type the name of the project and click on “Finish”
- Select your project create a package org.virtualkey
- Under the package org.virtualkey create a New Class Named as LockedMe.java
- Create a Main Method in side this class.
- This class will be the main method class where application Execution starts.

Step 2: Writing a program in Java for the entry point of the application (LockedMe.java) under org.virtualkey package

```java
package org.virtualkey;

import org.virtualkey.entities.WelcomeScreen;

public class LockedMe {

	public static void main(String[] args) {

		 WelcomeScreen welcome = new WelcomeScreen();
		welcome.welcomeScreen();;
		welcome.GetUserInput();
	}

}
```

Step 3: Create a sub package org.virtualkey.entities, then create two class named as Directory , WelcomeScreen and create an Interface named as Screen

Directory.java

```java
package org.virtualkey.entities;
import java.util.ArrayList;
import java.util.Collections;
import java.io.File;
import java.nio.file.FileSystems;
import java.nio.file.Path;


public class Directory {

   public static final String name = "src/main/";

    private ArrayList<File> files = new ArrayList<File>();

    Path path = FileSystems.getDefault().getPath(name).toAbsolutePath();

    File Dfiles = path.toFile();

    public String getName() {
        return name;
    }
    public void print() {
       System.out.println("Existing Files: ");
       files.forEach(f -> System.out.println(f));
    }

    public ArrayList<File> fillFiles() {

        File[] directoryFiles = Dfiles.listFiles();
        files.clear();
        if(directoryFiles != null) {
       for (int i = 0; i < directoryFiles.length; i++) {
            if (directoryFiles[i].isFile()) {
                files.add(directoryFiles[i]);
            }
       }
       Collections.sort(files);
        }
       return files;
    }
```

WelcomeScreen.java

```
package org.virtualkey.entities;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

import org.virtualkey.operations.DirectoryService;
import org.virtualkey.operations.ScreenService;


public class WelcomeScreen implements Screen {
         private String welcomeText = "Welcome to LockedMe:A Virtual Key";
         private String developerText = "Developed By: BIBHU RANJAN MOHANTY";
         private ArrayList<String> options = new ArrayList<>();

         public WelcomeScreen() {
             options.add ("1. Display the current file names in ascending 
order.");

             options.add ("2. Manage Files/Folders");
             options.add ("3. Quit");

         }

         public void welcomeScreen() {
            System.out.println(welcomeText);
             System.out.println(developerText);
             System.out.println("\n");
             Show();
         }

         public void Show() {
            System.out.println("Main Menu");
             for (String s : options)  {
                 System.out.println(s);
             }

         }
```

```java
        public void GetUserInput() {
            int selectedOption  = 0;

            while (true) {
                  selectedOption = this.getOption();
                  if(selectedOption == 3) {
                          System.out.println ("Program Terminated Successfully.");
                  break;
                  }
                  this.NavigateOption(selectedOption);
            }
        }
        public void NavigateOption(int option) {
            switch(option) {
                case 1: //Show Files in the current Directory
                    this.ShowFiles();
                    this.Show();
                    break;

                case 2: // Show File Options menu

    ScreenService.setCurrentScreen(ScreenService.FileOptionsScreen);
                    ScreenService.getCurrentScreen().Show();
                    ScreenService.getCurrentScreen().GetUserInput();
                    this.Show();
                    break;
                default:
                    System.out.println("Invalid Option");
                    break;
            }

        }

        public void ShowFiles() {

            // Get the files from the Directory
           System.out.println("List of Files: ");
           DirectoryService.PrintFiles();

        }

        private int getOption() {
            Scanner in = new Scanner(System.in);

            int returnOption = 0;
            try {
                returnOption = in.nextInt();
            }
            catch (InputMismatchException ex) {

            }
            return returnOption;

        }
}
```

Screen.java

```java
package org.virtualkey.entities;

public interface Screen {
	public void Show();
	
    public void NavigateOption(int option);
    
    public void GetUserInput();
}
```

Step 4: Create a sub package org.virtualkey.operations, then create three class named as DirectoryService , FileOption and ScreenService

DirectoryService.java

```java
package org.virtualkey.operations;
import java.io.File;

import org.virtualkey.entities.Directory;
public class DirectoryService {
	private static Directory fileDirectory = new Directory();
	
	    public static void PrintFiles() {
	    	
	      fileDirectory.fillFiles();

	        for (File file : DirectoryService.getFileDirectory().getFiles())
	        {
	            System.out.println(file.getName());
	        }
	    }
	    public static Directory getFileDirectory() {
	        return fileDirectory;
	    }

	    public static void setFileDirectory(Directory fileDirectory) {
	        DirectoryService.fileDirectory = fileDirectory;
	    }

}
```

FileOption.java

```java
package org.virtualkey.operations;

import java.io.File;
import java.io.IOException;
import java.nio.file.FileSystems;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

import org.virtualkey.entities.Directory;
import org.virtualkey.entities.Screen;

public class FileOptions implements Screen {
private Directory dir = new Directory();

      private ArrayList<String> options = new ArrayList<>();

    public FileOptions() {
         options.add("1. Add a File");
        options.add("2. Delete A File");
        options.add("3. Search A File");
        options.add("4. Return to Menu");
    }

    @Override
    public void Show() {
      System.out.println("File Options Menu");
        for (String s : options) {
            System.out.println(s);
        }

    }

    public void GetUserInput() {
        int selectedOption;
        while ((selectedOption = this.getOption()) != 4) {
            this.NavigateOption(selectedOption);
        }
    }

    @Override
    public void NavigateOption(int option) {

      switch(option) {

            case 1: // Add File
                this.AddFile();
                this.Show();
                break;
            case 2: // Delete File
                this.DeleteFile();
                this.Show();
                break;
            case 3: // Search File
                this.SearchFile();
                this.Show();
                break;
            default:
                System.out.println("Invalid Option");
                break;

          }
        }

        public void AddFile() {
            System.out.println("Please Enter the Filename:");

            String fileName = this.getInputString();

            System.out.println("You are adding a file named: " + fileName);

             try {
                    Path path = FileSystems.getDefault().getPath(Directory.name +
fileName).toAbsolutePath();
                    File file = new File(dir.getName() + fileName);

                    if (file.createNewFile()) {
                      System.out.println("File created: " + file.getName());
                      dir.getFiles().add(file);

                    } else {
                      System.out.println("This File Already Exits, no need to add
another");
                    }
             }catch (IOException e){
                    System.out.println(e);
             }
       }

        public void DeleteFile() {

             System.out.println("Please Enter the Filename:");

            String fileName = this.getInputString();

            System.out.println("You are deleting a file named: " + fileName);
            Path path = FileSystems.getDefault().getPath(Directory.name +
fileName).toAbsolutePath();
             File file = path.toFile();
            if (file.delete()) {
               System.out.println("Deleted File: " + file.getName());
               dir.getFiles().remove(file);
            } else {
              System.out.println("Failed to delete file:" + fileName + ", file was
not found.");
            }
        }

        public void SearchFile() {

             Boolean found = false;

             System.out.println("Please Enter the Filename:");

            String fileName = this.getInputString();

            System.out.println("You are searching for a file named: " + fileName);
```

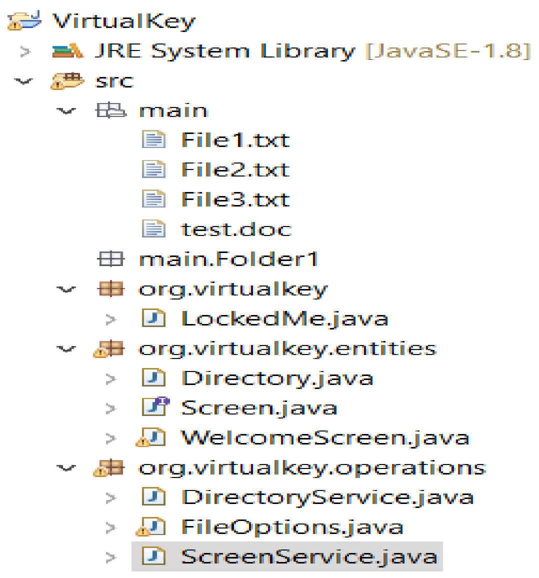
```java
            //TODO Fix it so ArrayList obtains files
            //Finished TODO

            ArrayList<File> files = dir.getFiles();


            for(int i = 0; i < files.size(); i++) {
                    if(files.get(i).getName().startsWith(fileName)) {
                          System.out.println("Found " + files.get(i).getName());
                          found = true;
                    }
            }
            if (found == false) {
             System.out.println("File not found");
            }
        }

        private String getInputString() {

            Scanner in = new Scanner(System.in);
            return(in.nextLine());

        }

        private int getOption() {
            Scanner in = new Scanner(System.in);

            int returnOption = 0;
            try {
                returnOption = in.nextInt();
            }
            catch (InputMismatchException ex) {
             System.out.println("Invalid input");
            }
            return returnOption;

        }

}
```

ScreenService.java

```java
package org.virtualkey.operations;

import org.virtualkey.entities.Screen;
import org.virtualkey.entities.WelcomeScreen;

public class ScreenService {

           public static WelcomeScreen WelcomeScreen = new WelcomeScreen();
           public static FileOptions FileOptionsScreen = new FileOptions();

           public static Screen CurrentScreen = WelcomeScreen;

           public static Screen getCurrentScreen() {
               return CurrentScreen;
           }
           public static void setCurrentScreen(Screen currentScreen) {
               CurrentScreen = currentScreen;
           }

}
```

Step 5 : Screenshot

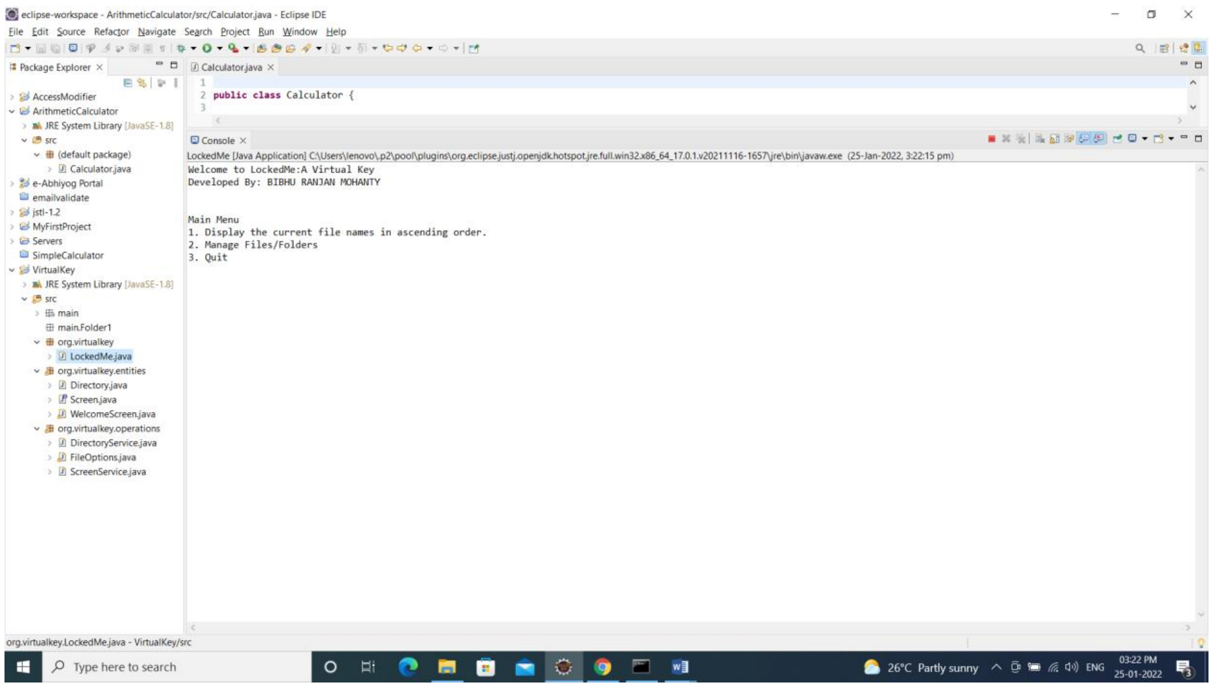
Project Structure :



```
VirtualKey
  > JRE System Library [JavaSE-1.8]
  ˅ src
    ˅ main
        File1.txt
        File2.txt
        File3.txt
        test.doc
      main.Folder1
    ˅ org.virtualkey
      > LockedMe.java
    ˅ org.virtualkey.entities
      > Directory.java
      > Screen.java
      > WelcomeScreen.java
    ˅ org.virtualkey.operations
      > DirectoryService.java
      > FileOptions.java
      > ScreenService.java
```

Output Screens :

→ Welcome Screen



eclipse-workspace - ArithmeticCalculator/src/Calculator.java - Eclipse IDE

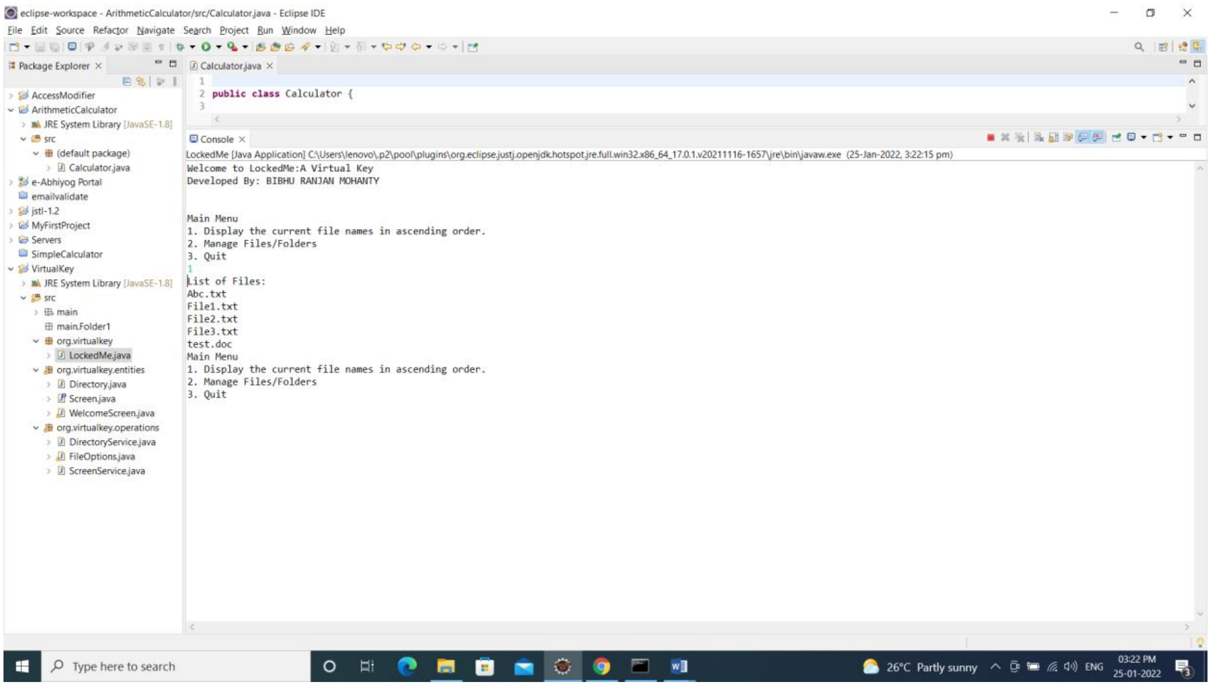
```
public class Calculator {
```

LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)

```
Welcome to LockedMe:A Virtual Key
Developed By: BIBHU RANJAN MOHANTY


Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Calculator.java
Console
org.virtualkey.LockedMe.java - VirtualKey/src
Type here to search
26°C Partly sunny
ENG
03:22 PM
25-01-2022

→ Display Current File names

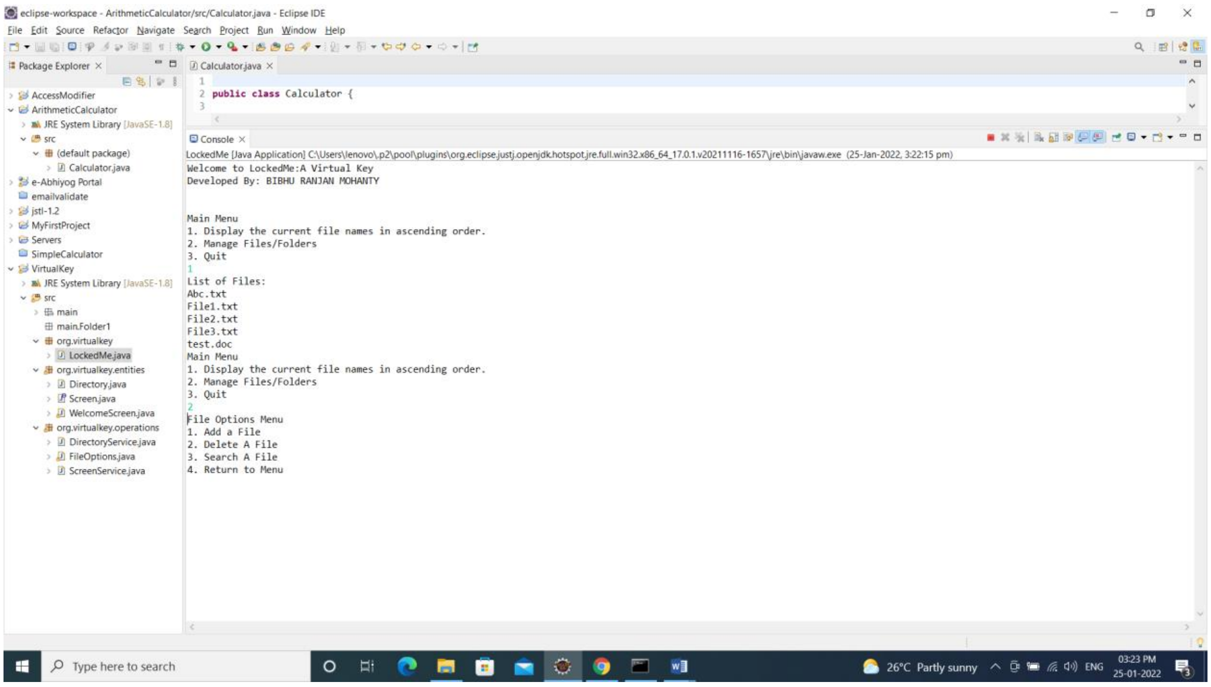


```
public class Calculator {
```

LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)

```
Welcome to LockedMe:A Virtual Key
Developed By: BIBHU RANJAN MOHANTY


Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
1
List of Files:
Abc.txt
File1.txt
File2.txt
File3.txt
test.doc
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
```

→Manage Files

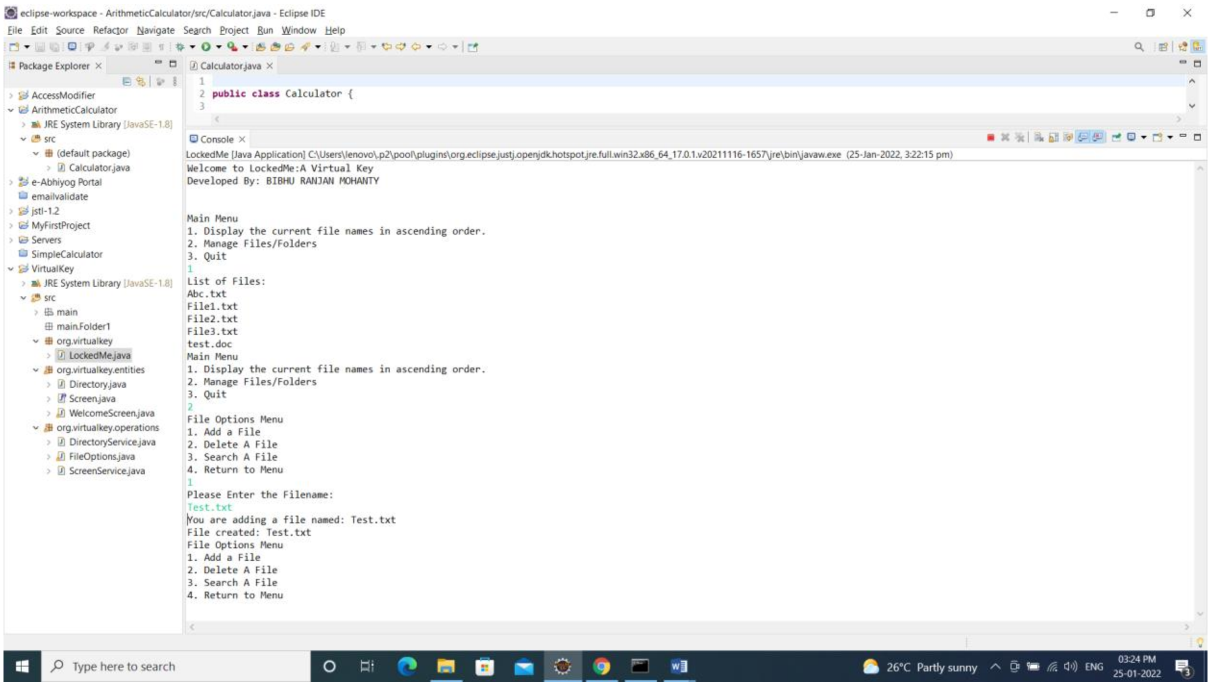


```
public class Calculator {
```

LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)

```
Welcome to LockedMe:A Virtual Key
Developed By: BIBHU RANJAN MOHANTY


Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
1
List of Files:
Abc.txt
File1.txt
File2.txt
File3.txt
test.doc
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
2
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
```

→ Add a New File

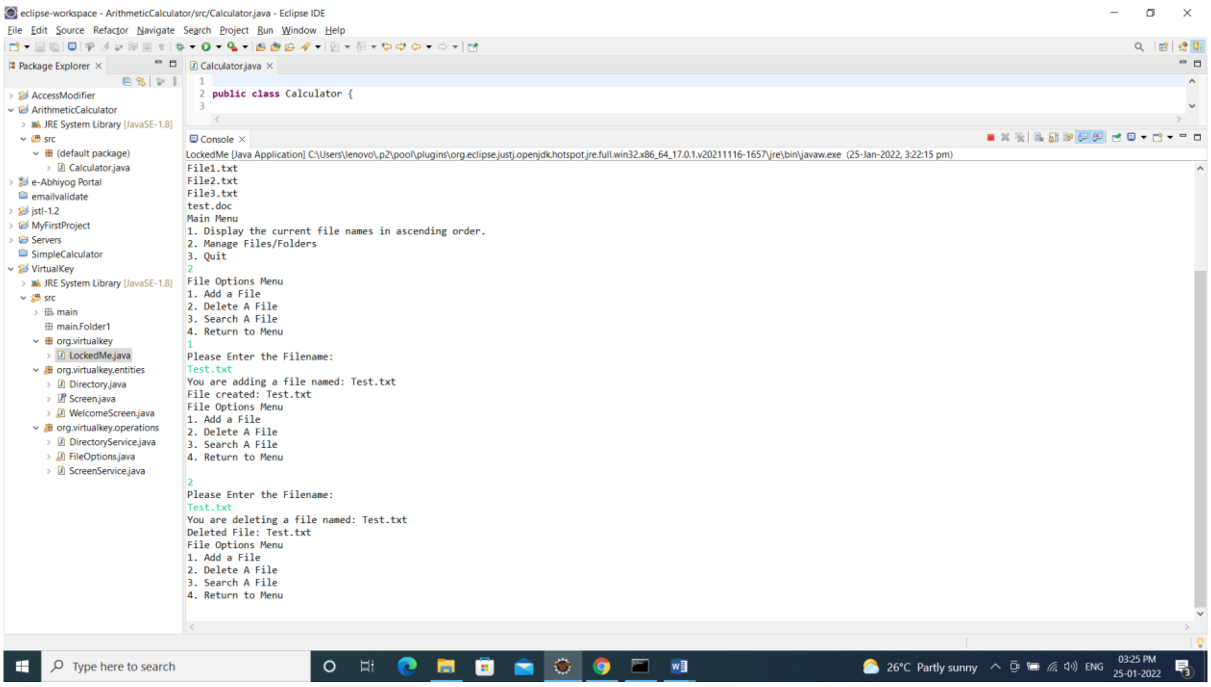


```java
public class Calculator {
```

LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)

```
Welcome to LockedMe:A Virtual Key
Developed By: BIBHU RANJAN MOHANTY


Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
1
List of Files:
Abc.txt
File1.txt
File2.txt
File3.txt
test.doc
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
2
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
1
Please Enter the Filename:
Test.txt
You are adding a file named: Test.txt
File created: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
```

→ Delete a file

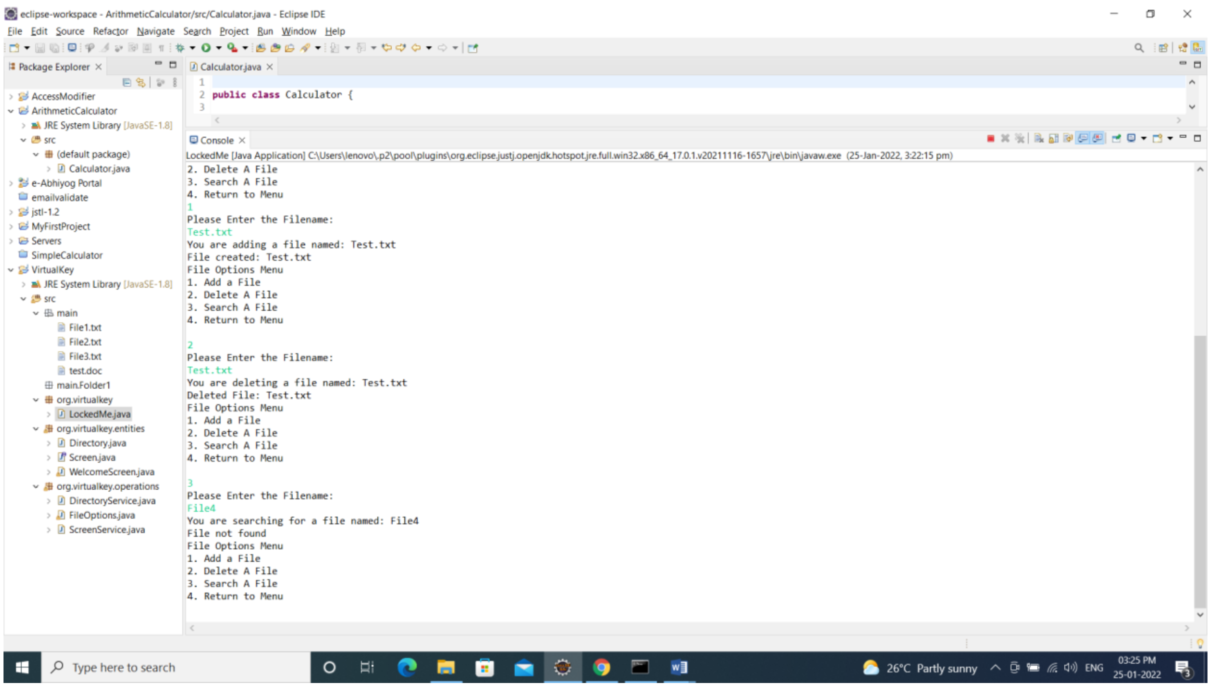


```java
public class Calculator {
```

LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)

```
File1.txt
File2.txt
File3.txt
test.doc
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
2
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
1
Please Enter the Filename:
Test.txt
You are adding a file named: Test.txt
File created: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

2
Please Enter the Filename:
Test.txt
You are deleting a file named: Test.txt
Deleted File: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
```

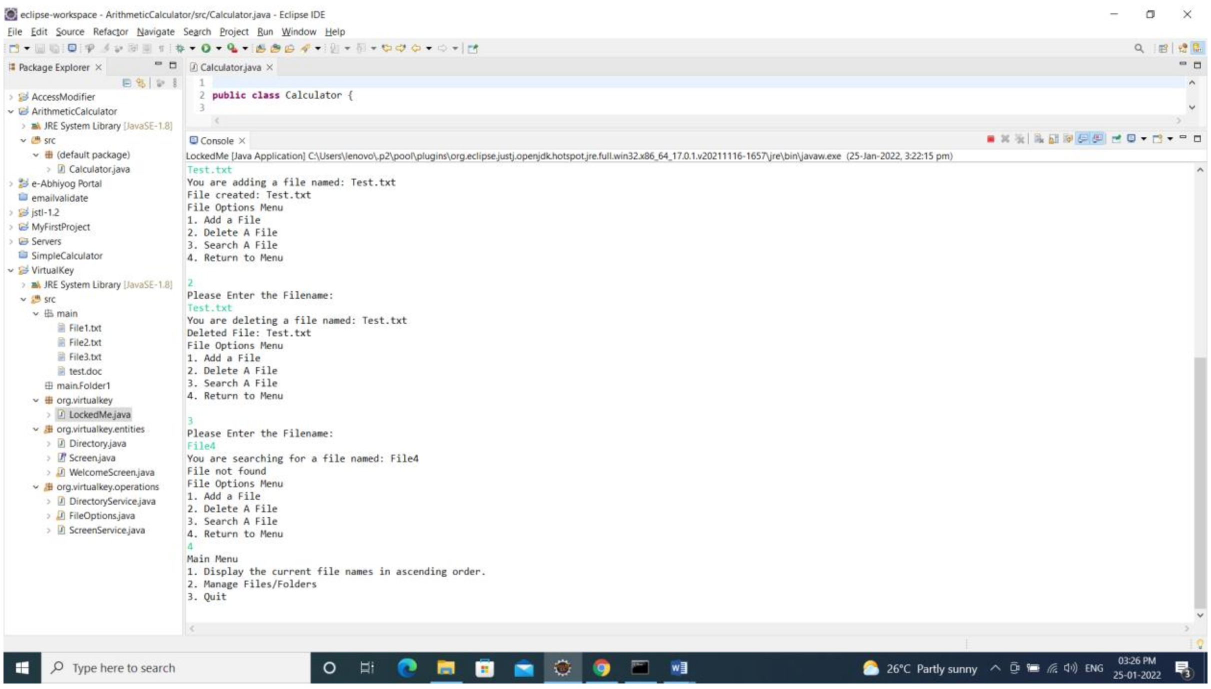
→ Search a file



```
public class Calculator {
```

```
LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)
2. Delete A File
3. Search A File
4. Return to Menu
1
Please Enter the Filename:
Test.txt
You are adding a file named: Test.txt
File created: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

2
Please Enter the Filename:
Test.txt
You are deleting a file named: Test.txt
Deleted File: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

3
Please Enter the Filename:
File4
You are searching for a file named: File4
File not found
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
```

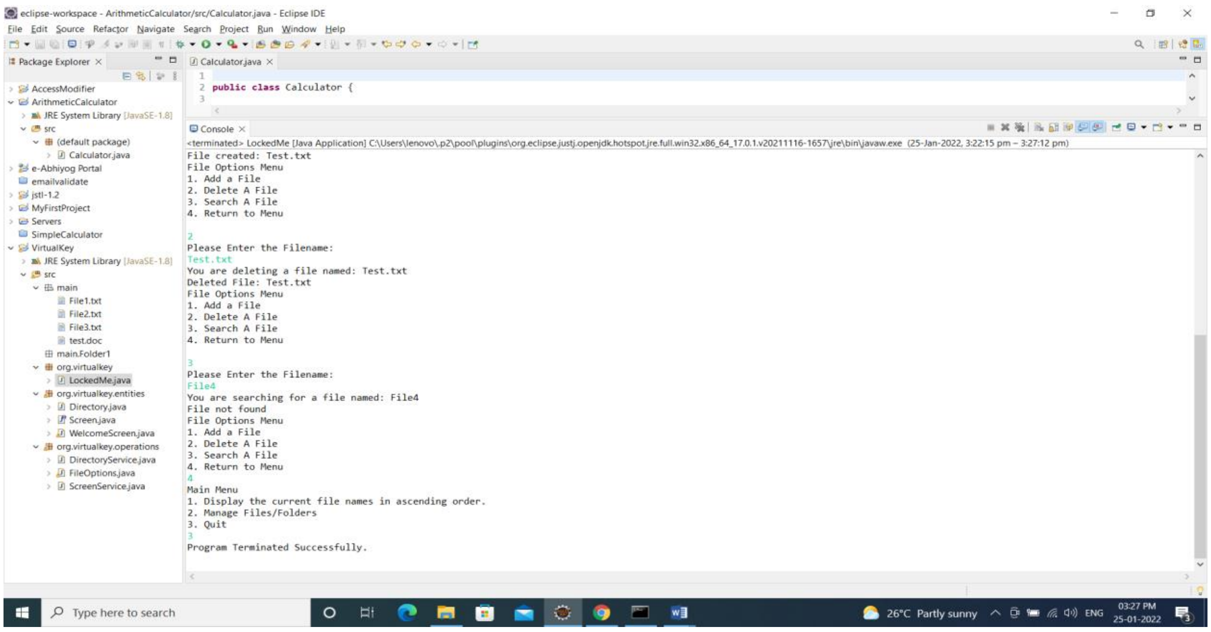
→ Goto Previous Menu



```
public class Calculator {
```

```
LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm)
Test.txt
You are adding a file named: Test.txt
File created: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

2
Please Enter the Filename:
Test.txt
You are deleting a file named: Test.txt
Deleted File: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

3
Please Enter the Filename:
File4
You are searching for a file named: File4
File not found
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
4
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
```

→ Quit the Application



```
public class Calculator {
```

```
<terminated> LockedMe [Java Application] C:\Users\lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (25-Jan-2022, 3:22:15 pm – 3:27:12 pm)
File created: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

2
Please Enter the Filename:
Test.txt
You are deleting a file named: Test.txt
Deleted File: Test.txt
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu

3
Please Enter the Filename:
File4
You are searching for a file named: File4
File not found
File Options Menu
1. Add a File
2. Delete A File
3. Search A File
4. Return to Menu
4
Main Menu
1. Display the current file names in ascending order.
2. Manage Files/Folders
3. Quit
3
Program Terminated Successfully.
```

Step 6: Pushing the code to GitHub repository

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

  cd <folder path>

- Initialize repository using the 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 the following 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, appropriate option needs to be selected.

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

3. User is also provided 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.

Conclusion

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.