LockedMe.com

Developed by Shruti Sharma.
This project is hosted at https://github.com/Shru03/LockedME .

CONTENTS

SPRINTS PLAN AND TASK COMPLETION

The project will be finished in two sprints.
The following tasks are expected to be performed during the first sprint:

- Creating the application's flow
- Setting up a git repository to track changes as the project develops.
- Writing a Java program to meet the project's requirements.

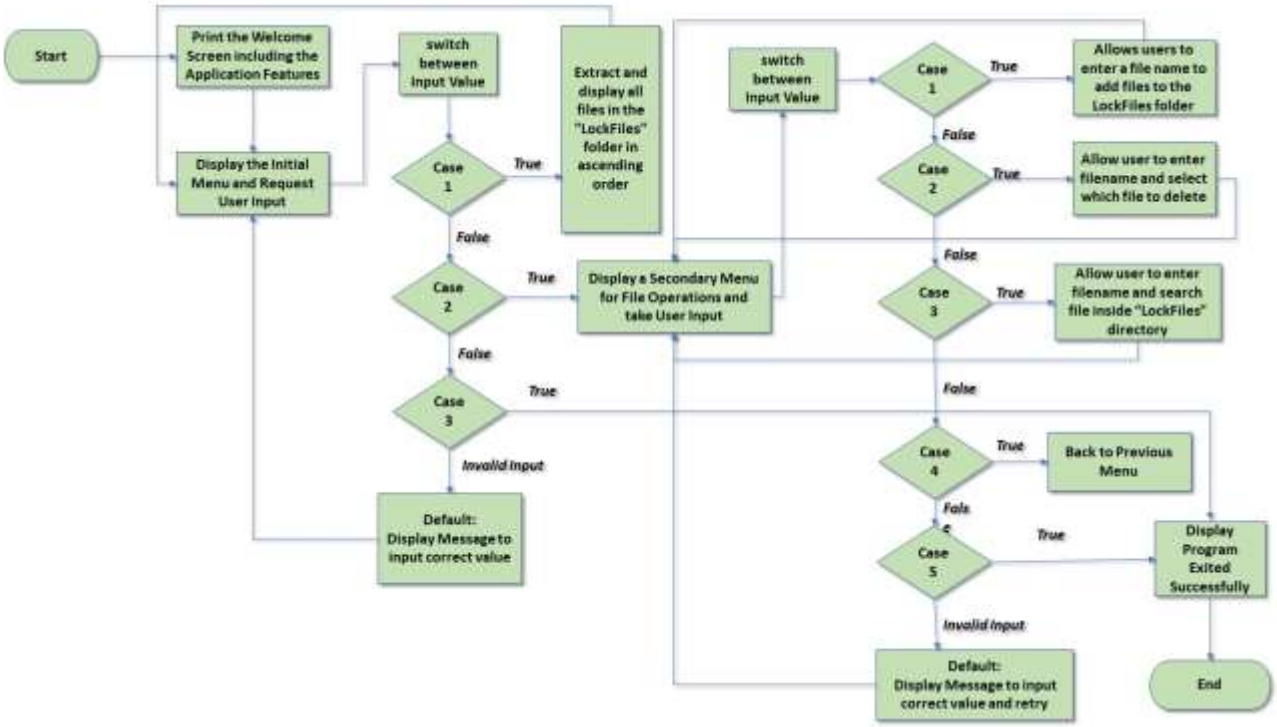
The following tasks are expected to be done in sprint two:

- Testing the Java software with various types of user input
- Using GitHub to push code.
- Creating a specification document that highlights the features, appearance, and user interactions of the program.

BASIC CONCEPTS USED IN PROJECT

File Handling, Sorting, Flow Control, Exception Handling and Collections Framework.

FLOW OF THE APPLICATION



PRODUCT CAPABILITIES, APPEARANCE, AND USER INTERACTIONS

The following sub-sections have been adjusted to showcase the project's appearance and user interactions in order to demonstrate the product's capabilities:

1. The first step is to create the project in Eclipse
2. Create a Java program for the application's entry point (LockedMeMain.java)
3. Create a Java software that displays menu options to the user (Menu.java)
4. Writing a Java program to handle user-selected menu options/choices (ManageChoices.java)
5. Writing a Java Program that display files in Ascending Order (DisplayFilesAO.java)
6. Write Java programs to perform File operations as defined by the user (AddingFile.java, DeletingFile.java, SearchingFile.java)
7. Uploading the source code to the GitHub repository

Step 1: Creating a new project in Eclipse

- ✓ Open Eclipse
- ✓ Go to File -> New -> Project -> Java Project -> Next.
- ✓ Type in project name (“LockedME”) and click on “Finish.”

Step 2: Create a Java program for the application's entry point **(LockedMeMain.java)**

- ✓ Select the project and go to File -> New -> Class.
- ✓ Enter **LockedMeMain** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish”.

```
public class LockedMeMain {
    public static void main(String[] args) {

        //Prints welcome screen
        System.out.println("------------------------");
        System.out.println("|\tLockedMe.com\t|");
        System.out.println("------------------------");
        System.out.println("Developed By - Shruti Sharma @Company Lockers Pvt. Ltd.\n");

        String features = "This programme can be used to:\n"
                + " • Get the List of all the files in the \"LockFiles\" folder\n"
                + " • For searching, adding, or deleting files in the \"LockFiles\" folder.\n" ;

        //Prints features of program
        System.out.println(features);

        //Prints Main menu
        Menu.displayMainMenu();

        //Handles Main menu options by calling mainMenuOptions method of class ManageChoices.
        ManageChoices manageChoices = new ManageChoices();
        manageChoices.mainMenuOptions();
    }

}
```

Output:

```
------------------------
|       LockedMe.com    |
------------------------
Developed By - Shruti Sharma @Company Lockers Pvt. Ltd.

This programme can be used to:
 • Get the List of all the files in the "LockFiles" folder
 • For searching, adding, or deleting files in the "LockFiles" folder.
```

Step 3: Create a Java Program that displays menu options to the user **(Menu.java)**

- ✓ Select your project and go to File -> New -> Class.
- ✓ Enter **Menu** in class name and click on “Finish.”
- ✓ **Menu** consists of two methods for -:
  - Displaying Main Menu (displayMainMenu( ))
  - Displaying File Menu for File Operations (displayFileMenu( ))

```
public class Menu {

    //Options for Main menu
    public static void displayMainMenu() {
        String mainMenu = "\n------MAIN MENU ------\n\n"
                + "1. Display files inside \"LockFiles\" folder\n"
                + "2. Go for File operations: Searching, Adding or Deleting\n"
                + "3. Exit program\n"
                + "\nType option number and press Enter.\n";
        System.out.println(mainMenu);
        }

    //Options for File menu
    public static void displayFileMenu() {
        String fileMenu = "\n------ FILE OPERATIONS MENU ------\n\n"
                + "1. Add a file to \"LockFiles\" folder\n"
                + "2. Delete a file from \"LockFiles\" folder\n"
                + "3. Search for a file from \"LockFiles\" folder\n"
                + "4. Show Previous Menu\n"
                + "5. Exit program\n"
                + "\nType option number and press Enter.";
        System.out.println(fileMenu);
        }

}
```

Output:

```
------MAIN MENU ------

1. Display files inside "LockFiles" folder
2. Go for File operations: Searching, Adding or Deleting
3. Exit program

Type option number and press Enter.

2

------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
```

Step 4: Writing a Java program to handle user-selected menu options/choices **(ManageChoices.java)**

- Select your project and go to File -> New -> Class.
- Enter **ManageChoices** in class name and click on "Finish."
- **ManageChoices** consists of two methods -:

    1. Handling input selected by user in Main Menu **mainMenuOptions( ))**
    2. Handling input selected by user in File Menu for File Operations (**fileMenuOptions( ))**

1. **mainMenuOptions( )**

```
public void mainMenuOptions() {
          boolean running = true;
          Scanner sc = new Scanner(System.in);
    do {
          try {
                int choice = sc.nextInt();

                switch (choice) {
                case 1:
                      DisplayFilesAO ao = new DisplayFilesAO();
                      ao.ascendOrder();
                      break;
                case 2:
                      Menu.displayFileMenu();
                      ManageChoices manageChoices = new ManageChoices();
                      manageChoices.fileMenuOptions();
                      break;
                case 3:
                      System.out.println("Program exited successfully");
                      running =false;
                      sc.close();
                      System.exit(0);
                      break;
                default:
                      System.out.println("Please enter a valid option");
                }
          } catch (Exception e) {
                e.printStackTrace();
                mainMenuOptions();
          } finally {
                System.out.println("\nTry more operations from the menu. \n");
                Menu.displayMainMenu();
          }
    }while (running ==true);
```

Output:

```
------MAIN MENU ------

1. Display files inside "LockFiles" folder
2. Go for File operations: Searching, Adding or Deleting
3. Exit program

Type option number and press Enter.

1
-------------Sorting by filename in ascending order------------

aa.txt
ab.txt
ba.txt
bb.txt
ca.txt
cb.txt
data.txt
djk.txt



Try more operations from the menu.


------MAIN MENU ------

1. Display files inside "LockFiles" folder
2. Go for File operations: Searching, Adding or Deleting
3. Exit program

Type option number and press Enter.

2

------ FILE OPERATIONS MENU ------

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

```
------MAIN MENU ------

1. Display files inside "LockFiles" folder
2. Go for File operations: Searching, Adding or Deleting
3. Exit program

Type option number and press Enter.

3
Program exited successfully
```

2 fileMenuOptions( )

```
//handles file menu operations
  public void fileMenuOptions() {
        boolean running = true;
        Scanner sc = new Scanner(System.in);
        do {
            try {
                  int choice = sc.nextInt();
                  switch (choice) {
                  case 1:
                        AddingFile ad = new AddingFile();
                        ad.addFile();
                        break;
                  case 2:
                        DeletingFile dl = new DeletingFile();
                        dl.delFile();
                        break;
                  case 3:
                      SearchingFile sf = new SearchingFile();
                      sf.searchFile();
                        break;
                  case 4:
                        Menu.displayMainMenu();
                        ManageChoices manageChoices = new ManageChoices();
                        manageChoices.mainMenuOptions();
                        break;
                  case 5:
                        System.out.println("Program exited successfully");
                        running =false;
                        sc.close();
                        System.exit(0);
                        break;
                  default:
                        System.out.println("Please enter valid Input.");
                        break;
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }  finally {
                  System.out.println("\nTry more operations from the menu. \n");
                  Menu.displayFileMenu();
            }
      } while (running == true);
  }

}
```

Output

Input option 1:

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
1
Enter file name : happy.txt
Created a File: happy.txt
```

Input option 2:

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
2

Choose File to delete :
aa.txt
ab.txt
ba.txt
bb.txt
ca.txt
cb.txt
data.txt
djk.txt
happy.txt
Enter file name to be deleted :
djk.txt
djk.txt file is Deleted
```

Input option 3:

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
3
Enter filename to be searched : happy.txt
File is Found at location : C:\Users\siddh\eclipse-workspace\LockedME\LockFiles
```

Input option 4:

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
4

------MAIN MENU ------

1. Display files inside "LockFiles" folder
2. Go for File operations: Searching, Adding or Deleting
3. Exit program

Type option number and press Enter.
```

Input option 5:

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
5
Program exited successfully
```

Step 5 Writing a Java Program that display files in Ascending Order **(DisplayFilesAO.java)**

- ✓ Select the project and go to File -> New -> Class.
- ✓ Enter **DisplayFilesAO** in any class name and click on "Finish."
- ✓ **DisplayFilesAO** class consist of one method:
    - **ascendOrder( ) –** To Display files in AO in LockFiles directory

```
import java.io.File;

//displays the file's contents in ascending order.
public class DisplayFilesAO {


    public void ascendOrder() {
    File fileDir = new File("C:\\Users\\siddh\\eclipse-workspace\\LockedME\\LockFiles");

    //Arrange contents of the directory in ascending order
    if(fileDir.isDirectory()){
        List<String> listFile = Arrays.asList(fileDir.list());
        Collections.sort(listFile);
        System.out.println("-------------Sorting by filename in ascending order------------\n");
        for(String s:listFile){
            System.out.println(s);
        }
        System.out.println("\n");
    }
    else{
        System.out.println(fileDir.getAbsolutePath() + " is not a directory");
    }
  }
}
```

Output

```
-------------Sorting by filename in ascending order-------

aa.txt
ab.txt
ba.txt
bb.txt
ca.txt
cb.txt
data.txt
happy.txt
```

Step 6 Write Java programs to perform File operations as defined by the user **(AddingFile.java, DeletingFile.java, SearchingFile.java)**

6.1 AddingFile.java

```java
import java.io.File;

public class AddingFile {
    Scanner sc = new Scanner(System.in);
    String fname;

    public void addFile() {
        //Accepting the user's file name as input
        System.out.print("Enter file name : ");
        fname = sc.next();

        //creating file by calling createFile method
        AddingFile a = new AddingFile();
        a.createFile(fname);
    }

    public void createFile(String fname) {
        File dir = new File ("C:\\Users\\siddh\\eclipse-workspace\\LockedME\\LockFiles");
        File file = new File(dir, fname);
        OutputStream out = null;
        try {
            out = new FileOutputStream(file);
            out.close();
            System.out.println("Created a File: " + file.getName());
        } catch (IOException e) {
            e.printStackTrace();
        }


    }

}
```

Output

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
1
Enter file name : happy.txt
Created a File: happy.txt
```

6.2 DeletingFile.java

```
public class DeletingFile {
    Scanner sc = new Scanner(System.in);
    String fname;

    File dir = new File ("C:\\Users\\siddh\\eclipse-workspace\\LockedME\\LockFiles");

    //Takes the filename of the file to be removed and calls the file deletion function.
    public void delFile() {
        DeletingFile d = new DeletingFile();
        System.out.println("\nChoose File to delete : ");
        d.printFile();

        System.out.println("Enter file name to be deleted : ");
        fname = sc.next();

        d.deleteFile(fname);
    }

    //Deleting file from directory
    public void deleteFile(String fname) {
        File file = new File(dir, fname);

        if(file.delete())
            System.out.println(file.getName() +" file is Deleted");
        else
            System.out.println("No such file present");
    }

    //Display the files of directory
    public void printFile() {
        String[] st = dir.list();

        if (st ==null) {
            System.out.println("No files in Directory");
        }else {
            for (int i=0; i< st.length; i++) {
                String filename = st[i];
                System.out.println(filename);
            }
        }
    }

}
```

Output

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
2

Choose File to delete :
aa.txt
ab.txt
ba.txt
bb.txt
ca.txt
cb.txt
data.txt
djk.txt
happy.txt
Enter file name to be deleted :
djk.txt
djk.txt file is Deleted
```

6.3 SearchingFile.java

```
public class SearchingFile {
Scanner sc = new Scanner(System.in);

    File dir = new File ("C:\\Users\\siddh\\eclipse-workspace\\LockedME\\LockFiles");
    ArrayList<String> listfname = new ArrayList<>();
    String fname;

    //Takes the name of the file to be searched and calls the method that will do the searching.
    public void searchFile() {

        System.out.print("Enter filename to be searched : ");
        fname = sc.next();

        SearchingFile a = new SearchingFile();
        a.getList();
        a.find(fname);
    }

    //Gets the filenames from the directory and adds them to an array list for searching the files.
    public void getList() {
      String[] str = dir.list();

      if (str == null) {
         System.out.println( "Directory Empty.");
      } else {
         for (int i = 0; i< str.length; i++) {
            String filename = str[i];
            listfname.add(filename);
         }
      }

    }

    //Checking if the file is present in the array list
    public void find(String fname) {
        boolean present = listfname.contains(fname);

        if (present)
            System.out.println("File is Found at location : " + dir);
        else
            System.out.println("File Not Found in the directory.");
    }
```

Output

```
------ FILE OPERATIONS MENU ------

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

Type option number and press Enter.
3
Enter filename to be searched : happy.txt
File is Found at location : C:\Users\siddh\eclipse-workspace\LockedME\LockFiles
```

Step 6: Uploading the source code to the GitHub repository

Commands used in Git bash:

```
cd <folder path>

git init

git add .

git commit .  -m  <commit message>

git push -u origin master
```

UNIQUE SELLING POINTS OF THE APPLICATION

- ✓ Even if an exception occurs, the program is meant to continue running and accepting user input. The relevant option must be selected to terminate the application.
- ✓ Any file name can be added into the directory using this application.
- ✓ The user can specify a filename, and the application will look for the value in all files inside folder and show it.
- ✓ The user can also remove files using filename.
- ✓ After performing any file operation such as adding, searching, removing, or retrieving files, the user can effortlessly transition between options or return to the previous menu.
- ✓ Inside the directory, the user can get files in ascending order.
- ✓ The application has been built to be modular. Even if the path needs to be updated, it can be done using the source code.

CONCLUSIONS

Additional enhancements to the application can be made, such as:

- ✓ Allowing users to add content to the newly formed file if desired.
- ✓ Conditions to determine whether the user is permitted to delete or add files to specific locations.
- ✓ Viewing content of file before deleting, the user is asked to confirm that they definitely wish to delete it.
- ✓ Retrieving files/folders based on various criteria such as Last Modified, Type, and so forth.
- ✓ Allowing the user to add information to the file.