

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:

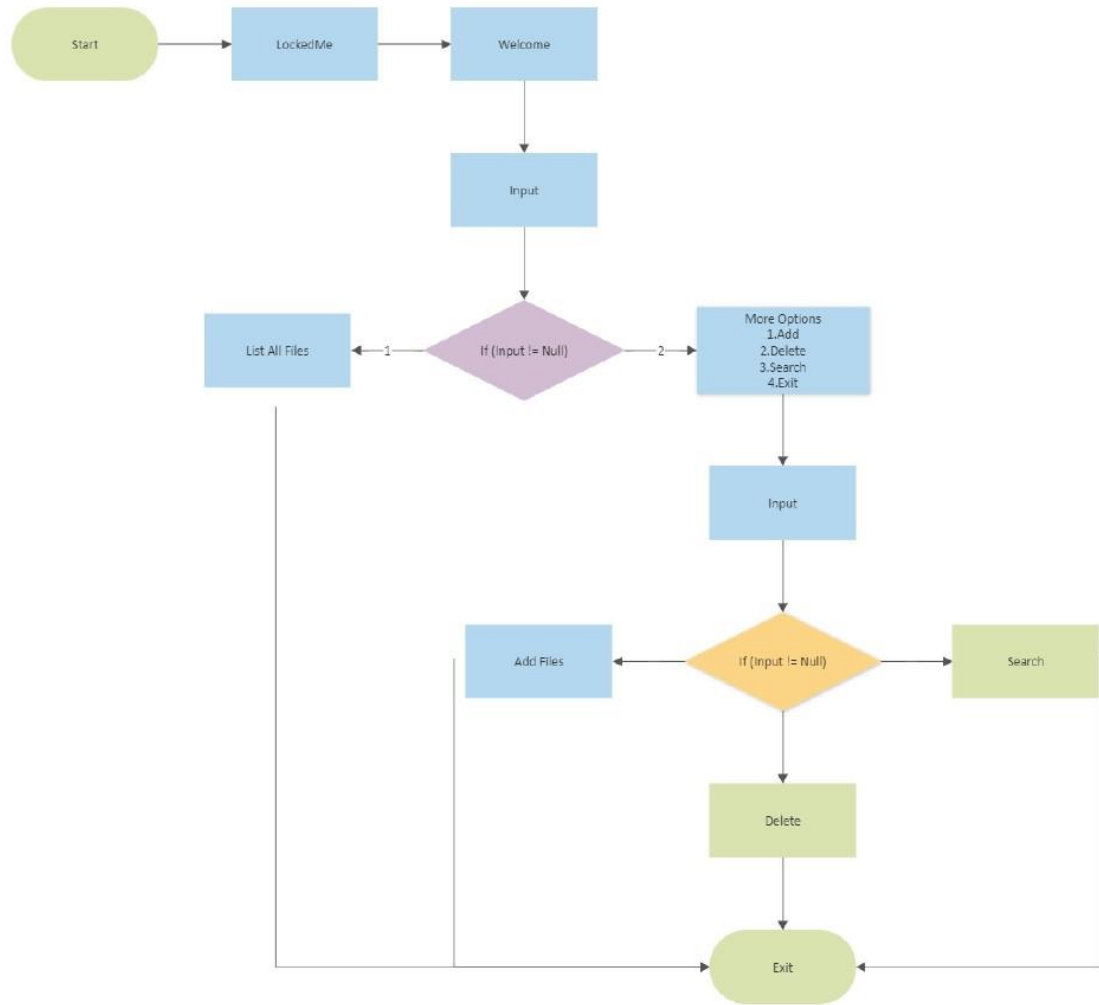
Sprint 1:

- 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.

LockedMe Application Flowchart



Output

```

Main x
"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

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
1 package org.example;
2
3 import Functionanlity.Menu;
4
5 5 usages
6 public class Main {
7     4 usages
8     public static final String path = "C:\\Users\\amitva2\\Downloads\\Lockedme";
9
10    no usages
11    public static void main(String[] args) {
12        Menu menu = new Menu();
13        menu.introScreen();
14        menu.mainMenu();
15    }
16 }
```

Menu.java

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
1 package Functionanlity;
2
3 import Options.FileOptions;
4 import org.example.Main;
5
6 import java.io.IOException;
7 import java.util.Scanner;
8
9 3 usages new *
10 public class Menu {
11     6 usages
12     Scanner scan = new Scanner(System.in);
13     4 usages
14     FileOptions dao = new FileOptions();
15
16     1 usage new *
17     public void introScreen() {
18         System.out.println();
19         System.out.println("*****");
20         System.out.println("*          DEVELOPED BY Amit Yadav          *");
21         System.out.println("*****");
22         System.out.println("*          LOCKEDME.COM          *");
23         System.out.println("*****");
24         System.out.println("\n\n");
25     }
26
27     1 usage new *
28     public void exitScreen() {
29
30         System.out.println("*****");
31         System.out.println("*****");
32         System.out.println("*          *");
33         System.out.println("*    THANK YOU FOR VISITING LOCKEDME.COM    *");
34         System.out.println("*          *");
35         System.out.println("*****");
36         System.out.println("*****");
37     }
```

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
29     System.out.println("*****");
30     System.out.println("*****");
31     System.out.println("\n\n");
32
33 }
34
35 1 usage: new *
36 public void Firstprompt() {
37     System.out.println("*****");
38     System.out.println("|          MAIN MENU          |");
39     System.out.println("*****");
40     System.out.println("| Select any one of the following: |");
41     System.out.println("| 1 - List All Files             |");
42     System.out.println("| 2 - More Options              |");
43     System.out.println("| 3 - Exit                      |");
44     System.out.println("*****");
45     System.out.println("Enter your choice : ");
46
47 }
48
49 1 usage: new *
50 public void subOptions() {
51
52     System.out.println("*****");
53     System.out.println("|          SUB MENU          |");
54     System.out.println("*****");
55     System.out.println("| Select any one of the following: |");
56     System.out.println("| 1 - Add a file                |");
57     System.out.println("| 2 - Delete a file            |");
58     System.out.println("| 3 - Search a file            |");
59     System.out.println("| 4 - Go Back                  |");
60     System.out.println("*****");
61     System.out.println("Enter your choice : ");
62
63 }
```

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
60 public void mainMenu() {
61
62     int choice = 0;
63     char decision = 0;
64     do {
65
66         Firstprompt();
67
68         try {
69             choice = Integer.parseInt(scan.nextLine());
70         } catch (NumberFormatException e) {
71             System.out.println("\nInvalid Input \nValid Input Integers:(1-3)\n");
72             mainMenu();
73         }
74
75
76         switch (choice) {
77
78             case 1:
79                 System.out.println();
80                 try {
81                     dao.listAllFiles(Main.path);
82                 } catch (NullPointerException e) {
83                     System.out.println(e.getMessage());
84                 } catch (IllegalArgumentException e) {
85                     System.out.println(e.getMessage());
86                 } catch (Exception e) {
87                     System.out.println(e.getMessage());
88                 }
89                 System.out.println("\n*****\n");
90                 break;
91
92             case 2:
93                 System.out.println();
94                 subMenu();
95         }
96     } while (decision != 'q');
```

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
94         subMenu();
95         break;
96
97         case 3:
98             System.out.println("\n Are you sure you want to exit ? ");
99             System.out.println(" (Y) ==> Yes (N) ==> No ");
100             decision = scan.nextLine().toUpperCase().charAt(0);
101             if(decision == 'Y') {
102                 System.out.println("\n");
103                 exitScreen();
104                 System.exit( status: 1);
105             }else if(decision == 'N') {
106                 System.out.println("\n");
107                 mainMenu();
108             }else {
109                 System.out.println("\nInvalid Input \nValid Inputs :(Y/N)\n");
110                 mainMenu();
111             }
112
113
114         default:
115             System.out.println("\nInvalid Input \nValid Input Integers:(1-3)\n");
116             mainMenu();
117
118     }
119
120
121 }while(true);
122
123 }
124 3 usages new *
125 public void subMenu() {
126     String file = null;
127     String fileName = null;
128     int choice = 0;
129
130     subOptions();
131
132
133     try {
134         choice = Integer.parseInt(scan.nextLine());
135     } catch (NumberFormatException e) {
136         System.out.println("Invalid Input \nValid Input Integers:(1-4)");
137         subMenu();
138     }
139
140
141     switch (choice) {
142     case 1:
143         System.out.println("\n==> Adding a File...");
144         System.out.println("Please enter a file name : ");
145         file = scan.nextLine();
146         fileName = file.trim();
147         try {
148             dao.createNewFile(Main.path, fileName);
149         }catch(NullPointerException e) {
150             System.out.println(e.getMessage());
151         }catch(IOException e) {
152             System.out.println("Error occurred while adding file..");
153             System.out.println("Please try again..");
154         }catch(Exception e) {
155             System.out.println("Error occurred while adding file..");
156             System.out.println("Please try again..");
157         }
158         System.out.println("\n*****\n");
159         break;
160
161     case 2:
162         System.out.println("\n==> Deleting a File...");
163         System.out.println("Please enter a file name to Delete : ");
164         file = scan.nextLine();
```

```
m pom.xml (Lockedme) x Main.java x .gitignore x FileOptions.java x Menu.java x
163 System.out.println("Please enter a file name to delete : ");
164 file = scan.nextLine();
165 fileName = file.trim();
166 try {
167     dao.deleteFile(Main.path, fileName);
168 } catch (NullPointerException e) {
169     System.out.println(e.getMessage());
170 } catch (IOException e) {
171     System.out.println("Error occurred while Deleting File..");
172     System.out.println("Please try again...");
173 } catch (Exception e) {
174     System.out.println("Error occurred while Deleting File..");
175     System.out.println("Please try again...");
176 }
177 System.out.println("\n*****\n");
178 break;
179
180 case 3:
181     System.out.println("\n==> Searching a File..");
182     System.out.println("Please enter a file name to Search : ");
183     file = scan.nextLine();
184     fileName = file.trim();
185     try {
186         dao.searchFile(Main.path, fileName);
187     } catch (NullPointerException e) {
188         System.out.println(e.getMessage());
189     } catch (IllegalArgumentException e) {
190         System.out.println(e.getMessage());
191     } catch (Exception e) {
192         System.out.println(e.getMessage());
193     }
194     System.out.println("\n*****\n");
195     break;
196 case 4: mainMenu();
197     break;
```

```
196 case 4: mainMenu();
197     break;
198
199 default:
200     System.out.println("Invalid Input \nValid Input Integers:(1-4)");
201     subMenu();
202
203 }
204
205 file = null;
206 fileName = null;
207
208 }while(true);
209
210 }
211
212 }
213
```

FileOptions.java

```
m pom.xml (Locked) x Main.java x .gitignore x FileOptions.java x Menu.java x
1 package Options;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.util.Arrays;
6 import java.util.Scanner;
7 import java.util.Set;
8 import java.util.TreeSet;
9 import java.util.regex.Matcher;
10 import java.util.regex.Pattern;
11
12 3 usages new "
13 public class FileOptions {
14     1 usage new "
15     public void listAllFiles(String path) {
16         if (path == null || path.isEmpty() || path.isEmpty())
17             throw new NullPointerException("Path cannot be Empty or null");
18
19         File dir = new File(path);
20
21         if(!dir.exists())
22             throw new IllegalArgumentException("Path does not exist");
23
24         if(dir.isFile())
25             throw new IllegalArgumentException("The given path is a file. A directory is expected.");
26
27         String [] files = dir.list();
28
29         System.out.println("\n*****");
30         if(files != null && files.length > 0) {
31
32             Set<String> fileList = new TreeSet<String>(Arrays.asList(files));
33             System.out.println("The Files in " + dir.getAbsolutePath() + " are: \n");
34             for(String file1:fileList) {
35
36                 System.out.println(file1);
37
38             }
39
40             System.out.println("\nTotal Number of files: " + fileList.size());
41         } else {
42
43             System.out.println("Directory is Empty");
44         }
45     }
46 }
47
48 1 usage new "
49 public void createNewFile(String path , String fileName) throws IOException {
50
51     if (path == null || path.isEmpty() || path.isEmpty())
52         throw new NullPointerException("Path cannot be Empty or null");
53
54     if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
55         throw new NullPointerException("File Name cannot be Empty or null");
56
57     File newFile = new File( pathname: path + File.separator + fileName);
58
59     boolean createFile = newFile.createNewFile();
60
61     if (createFile) {
62
63         System.out.println("\nFile Successfully Created: " + newFile.getAbsolutePath());
64     }
65 }
```

```
m pom.xml (Lockedme) × Main.java × .gitignore × FileOptions.java × Menu.java ×
64      System.out.println("\nFile Successfully Created: " + newFile.getAbsolutePath());
65
66      }else if(!createFile) {
67
68          System.out.println("\nFile Already Exist.. Please try again." );
69
70      }
71
72  }
73
74
75
76  1 usage new *
77  public void deleteFile(String path , String fileName) throws IOException {
78
79      if (path == null || path.isEmpty() || path.isEmpty())
80          throw new NullPointerException("Path cannot be Empty or null");
81
82      if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
83          throw new NullPointerException("File Name cannot be Empty or null");
84
85      File newFile = new File( pathname: path + File.separator + fileName);
86
87      boolean deleteFile = newFile.delete();
88
89      if (deleteFile) {
90
91          System.out.println("\nFile deleted Successfully");
92
93      }else {
94
95          System.out.println("\nFile Not Found.. Please try again." );
96
97      }
```

```
m pom.xml (Lockedme) × Main.java × .gitignore × FileOptions.java × Menu.java ×
103  public void searchFile(String path , String fileName){
104
105      if (path == null || path.isEmpty() || path.isEmpty())
106          throw new NullPointerException("Path cannot be Empty or null");
107
108
109      if (fileName == null || fileName.isEmpty() || fileName.isEmpty())
110          throw new NullPointerException("File Name cannot be Empty or null");
111
112      File dir = new File(path);
113
114      if(!dir.exists())
115          throw new IllegalArgumentException("Path does not exist");
116
117      if(dir.isFile())
118          throw new IllegalArgumentException("The given path is a file. A directory is expected.");
119
120
121      String [] fileList = dir.list();
122      boolean flag = false;
123
124      Pattern pat = Pattern.compile(fileName);
125
126      if(fileList != null && fileList.length > 0) {
127          for(String file:fileList) {
128              Matcher mat = pat.matcher(file);
129              if(mat.matches()) {
130                  System.out.println("File Found at location: " + dir.getAbsolutePath());
131                  flag = true;
132                  break;
133              }
134          }
135      }
```


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.