

LockedMe – Virtual Key for Repositories

(Project code Document)

Version History

Author	Rabin Kumar Pati
Doc purpose	Source Code Document
Date	13-Aug -2021
Version	01

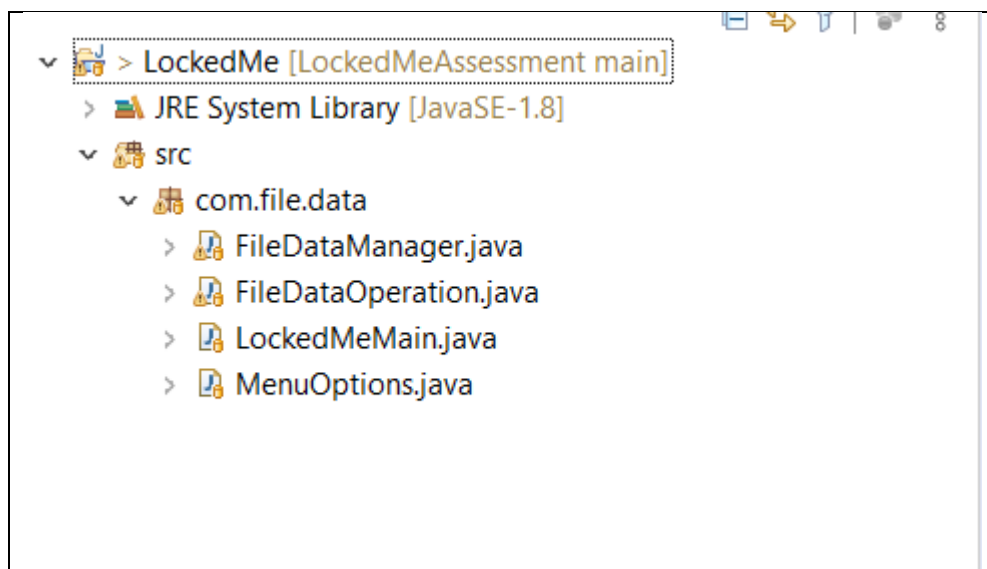
Table of Contents

1.	Document Contents:	3
1.1	Folder Structure	3
1.2	Project GitHub Details.....	3
1.3	Source Code	4
1.3.1	LockedMeMain.java.....	4
1.3.2	MenuOptions.java.....	4
1.3.3	FileDataOperation.java	6
1.3.4	FileDataManager.java	8

1. Document Contents:

- Folder Structure in Project Workspace.
- Project GitHub details.
- Source Code

1.1 Folder Structure



1.2 Project GitHub Details

Repositories Name	RABINPATI/LockedMeAssessment
GitHub Link	https://github.com/RABINPATI/LockedMeAssessment

1.3 Source Code

1.3.1 LockedMeMain.java

```
LockedMeMain.java
1 package com.file.data;
2
3 public class LockedMeMain {
4
5     // Folder Path for file to access data
6
7     static final String FOLDER_PATH = "D:\\Project\\FSD\\Phase 1\\LockedMeAssessment\\LockedMe";
8
9     public static void main(String[] args) {
10
11         // Display Application name
12
13         MenuOptions.displayWelcomeHeaderNote();
14
15         // Display file processing operations
16
17         MenuOptions.menuProcessOperations(FOLDER_PATH);
18
19     }
20
21 }
```

1.3.2 MenuOptions.java

```
1 package com.file.data;
2
3 import java.text.SimpleDateFormat;
4
5
6 public class MenuOptions {
7
8     /**
9     * Add the details for application Name in header part.
10    */
11
12    public static void displayWelcomeHeaderNote() {
13
14        System.out.println("_____ " + "\n");
15        System.out.println("_____ Application Name: LockedMe.com");
16        System.out.println("_____");
17    }
18
19    /**
20    * Add the details about developer and date in below part.
21    */
22    public static void displayWelcomeFooterNote() {
23
24        System.out.println("_____");
25        System.out.println("_____ Developed By RABIN");
26        System.out.println("_____ Date:");
27        System.out.println("_____ + new SimpleDateFormat("dd-MM-yyyy").format(new Date());");
28        System.out.println("_____");
29    }
30
31
32 }
33
```

```

34=  /**
35   * Display the menu options
36   */
37
38=  public static void displayFileMenuOptions() {
39      String fileMenu = "\n\n***** Select any number from below for file operation and press Enter
40      + "1) Retrieve all files \n"
41      + "2) Add a file \n"
42      + "3) Search a file \n"
43      + "4) Delete a file \n"
44      + "5) Exit program\n";
45
46      System.out.println(fileMenu);|
47
48  }
49
50=  /**
51   *
52   * @param folderPath
53   * File processing operation
54   * Retrieve files
55   * Add files,Search file, Delete file
56   */
57
58=  public static void menuProcessOperations(String folderPath) {
59
60      boolean running = true;
61      Scanner sc = new Scanner(System.in);
62      do {
63          try {
64              MenuOptions.displayFileMenuOptions();
65              System.out.println("Enter an option to proceed");
66              int input = sc.nextInt();
67
68              switch (input) {
69                  case 1:
70                      // Display all files from LockedMeFileData folder.
71
72                      FileDataOperation.displayFileNames(folderPath);
73                      break;
74                  case 2:
75                      // Add a file into LockedMeFileData folder.
76
77                      FileDataOperation.createFiles(folderPath);
78                      break;
79                  case 3:
80                      // Search a file from LockedMeFileData folder.
81
82                      FileDataOperation.searchFile(folderPath);
83                      break;
84                  case 4:
85                      //Delete a file from LockedMeFileData folder.
86
87                      FileDataOperation.deleteFile(folderPath);
88                      break;
89
90                  case 5:
91                      System.out.println("Program exited successfully.");
92                      running = false;
93                      sc.close();
94                      displayWelcomeFooterNote();
95                      System.exit(0);
96
97                      break;
98                  default:
99                      System.out.println("Please select a valid option from above.");
100              }
101          } catch (Exception e) {
102              System.out.println("Please select a valid option from above.");|
103              menuProcessOperations(folderPath);
104          }
105      } while (running == true);
106  }
107
108 }
109

```

1.3.3 FileDataOperation.java

```
1 package com.file.data;
2
3 import java.util.ArrayList;
4
5
6
7
8 public class FileDataOperation {
9
10     /**
11      * @param folderPath
12      * Add file
13      */
14
15     public static void createFiles(String folderPath) {
16
17         Scanner scanObj = new Scanner(System.in);
18
19         String file_Name;
20         int noOfLines;
21         List<String> dataContentList = new ArrayList<String>();
22
23         try {
24             System.out.println("Enter File Name to create:");
25             file_Name = scanObj.nextLine();
26
27             // Read number of Lines from User
28
29             System.out.println("Enter how many lines in the file");
30             noOfLines = Integer.parseInt(scanObj.nextLine());
31
32             if (noOfLines > 0) {
33                 // Read data contents from User
34                 for (int i = 1; i <= noOfLines; i++) {
35
36                     System.out.println("Enter line" + i + ": ");
37                     dataContentList.add(scanObj.nextLine());
38                 }
39
40                 // Save the data content into file
41
42                 boolean isFileSaved = FileDataManager.createFiles(folderPath, file_Name, dataContentList);
43                 if (isFileSaved)
44                     System.out.println("File : " + file_Name + " saved successfully");
45                 else {
46                     System.out.println("File not saved due to error occurred. Please check log for more details");
47                 }
48             } catch (Exception e) {
49                 System.out.println("Please insert file name to create");
50             }
51         }
52     }
53
54     /**
55      * @param folderPath
56      * Display Allfiles
57      */
58
59     public static void displayFileNames(String folderPath) {
60
61         List<String> fileNamesList = FileDataManager.getAllFiles(folderPath);
62
63         // shorting the file in ascending order
64
65         Collections.sort(fileNamesList);
66
67         if (!fileNamesList.isEmpty()) {
68             fileNamesList.forEach(fileName -> {
69                 System.out.println(fileName);
70             });
71         } else {
72             System.out.println("Files not found in directory");
73         }
74     }
75 }
```

```

74  /**
75   * @param folderPath
76   * Search File
77   */
78  public static void searchFile(String folderPath) {
79
80      Scanner scanObj = new Scanner(System.in);
81
82      String file_Name;
83
84      try {
85          System.out.println("Enter File Name to be search:");
86          file_Name = scanObj.nextLine();
87
88          // Search the file from folder
89
90          boolean isFileAvailable = FileDataManager.searchFile(folderPath, file_Name)
91
92          if (isFileAvailable)
93              System.out.println(file_Name + " : found in directory");
94          else {
95              System.out.println(file_Name + " : not found in directory");
96          }
97      } catch (Exception e) {
98          System.out.println("Please insert file name to search");
99      }
100  }
101
102  }
103  /**
104   * @param folderPath
105   * Delete File
106   */
107  public static void deleteFile(String folderPath) {
108
109      Scanner scanObj = new Scanner(System.in);
110
111      String file_Name;
112
113      try {
114          System.out.println("Enter File Name to delete:");
115          file_Name = scanObj.nextLine();
116
117          // Delete the file from folder
118
119          boolean isFileDeleted = FileDataManager.deleteFile(folderPath, file_Name)
120
121          if (isFileDeleted)
122              System.out.println(file_Name + " : deleted successfully");
123          else {
124              System.out.println(file_Name + " : not found in directory");
125          }
126          // Close the scanner object
127      } catch (Exception e) {
128          System.out.println("Please insert file name to delete.");
129      }
130  }
131
132  }
133 }
134

```

1.3.4 FileDataManager.java

```
LockedMeMain.java MenuOptions.java FileDataOperation.java FileDataManager.java
1 package com.file.data;
2
3 import java.io.File;
4
5
6
7
8
9 public class FileDataManager {
10
11     /**
12      *
13      * @param folderPath
14      * @param fileName
15      * @param dataContent
16      * @return file in directory
17      */
18
19     public static boolean createFiles(String folderPath, String fileName, List<String> dataContent) {
20
21         try {
22             File folderName = new File(folderPath, fileName);
23             FileWriter fw = new FileWriter(folderName);
24
25             dataContent.forEach(data -> {
26                 try {
27                     fw.write(data + "\n");
28                 } catch (IOException e) {
29                     // TODO Auto-generated catch block
30                     e.printStackTrace();
31                 }
32             });
33             fw.close();
34             return true;
35
36         } catch (Exception ex) {
37             ex.getMessage();
38             return false;
39         }
40     }
41
42
43     /**
44      *
45      * @param folderPath
46      * @return all files
47      */
48     public static List<String> getAllFiles(String folderPath) {
49
50         File folder = new File(folderPath);
51         File[] listOfFiles = folder.listFiles();
52
53         List<String> filesInFolder = new ArrayList<String>();
54         for (File file : listOfFiles) {
55             if (file.isFile()) {
56                 String fileName = file.getName();
57                 filesInFolder.add(fileName);
58             }
59         }
60         return filesInFolder;
61     }
62
63     /**
64      *
65      * @param folderPath
66      * @param file_Name
67      * @return true/false based on search results.
68      */
69     public static boolean searchFile(String folderPath, String file_Name) {
70
71         List<String> filesInFolderList = getAllFiles(folderPath);
72
73         boolean isSearched = false;
74     }
75 }
```



```

75         for (String fileName : filesInFolderList)
76         {
77             // Check user input file name should match with directory file.
78
79             if (fileName.equals(file_Name)) {
80                 File fileDetails = new File(folderPath + "\\ " + file_Name);
81
82                 if (fileDetails.exists())
83                     isSerached = true;
84                 break;
85             } else {
86                 isSerached = false;
87             }
88         }
89         return isSerached;
90     }
91     /**
92     *
93     * @param folderPath
94     * @param file_Name
95     * @return true/false based on delete results.
96     */
97
98     public static boolean deleteFile(String folderPath, String file_Name) {
99
100         // Check all files in directory for case sensitivity.
101
102         List<String> filesInFolderList = getAllFiles(folderPath);
103
104         boolean isDeleted = false;
105
106         for (String fileName : filesInFolderList)
107         {
108             // Check user input file name should match with directory file.
109             if (fileName.equals(file_Name)) {
110
111                 File fileDetails = new File(folderPath + "\\ " + file_Name);
112
113                 if (fileDetails.exists()) {
114                     if (fileDetails.delete())
115                         isDeleted = true;
116                     break;
117                 }
118             } else {
119                 isDeleted = false;
120             }
121         }
122         return isDeleted;
123     }
124 }

```