

```
SimpleLearn_projects - Final_Project/src/LockedMeApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
SimpleLearn_projects - Final_Project/src/LockedMeAppTest.java
1 import java.io.File;
2 import java.io.IOException;
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.List;
6 import java.util.Scanner;
7
8 public class LockedMeApp {
9     private static final String DIRECTORY_PATH = "C:\\Users\\santh\\Desktop\\SimpleLearn_projects\\Final_Project\\test"; // Update with the actual directory path
10     private static Scanner scanner;
11
12     public static void main(String[] args) {
13         displayWelcomeScreen();
14         showMainMenu();
15     }
16
17     private static void displayWelcomeScreen() {
18         System.out.println("=====");
19         System.out.println("Welcome to LockedMe.com");
20         System.out.println("Developed by Santhosh");
21         System.out.println("=====");
22     }
23
24     private static void showMainMenu() {
25         System.out.println("\nMAIN MENU");
26         System.out.println("1. Retrieve file names in ascending order");
27         System.out.println("2. Manage files");
28         System.out.println("3. Close the application");
29
30         int choice = getUserChoice(1, 3);
31
32         switch (choice) {
33             case 1:
34                 retrieveFileNames();
35                 break;
36             case 2:
37                 showFileManagementMenu();
38                 break;
39             case 3:
40                 closeApplication();
41                 break;
42         }
43     }
44
45     private static void retrieveFileNames() {
46         List<String> fileNames = getFileNamesInAscendingOrder();
47         System.out.println("\nFILE NAMES IN ASCENDING ORDER");
48         if (fileNames.isEmpty()) {
49             System.out.println("No files found in the directory.");
50         } else {
51             for (String fileName : fileNames) {
52                 System.out.println(fileName);
53             }
54         }
55         showMainMenu();
56     }
57
58     static List<String> getFileNamesInAscendingOrder() {
59         File directory = new File(DIRECTORY_PATH);
60         File[] files = directory.listFiles();
61         List<String> fileNames = new ArrayList<>();
62
63         if (files != null) {
64             for (File file : files) {
65                 if (file.isFile()) {
66                     fileNames.add(file.getName());
67                 }
68             }
69             Collections.sort(fileNames);
70         }
71
72         return fileNames;
73     }
74
75     private static void showFileManagementMenu() {
76         System.out.println("\nFILE MANAGEMENT MENU");
77         System.out.println("1. Add a file to the directory");
78         System.out.println("2. Delete a file from the directory");
79         System.out.println("3. Search for a file in the directory");
80         System.out.println("4. Navigate back to the main menu");
81
82         int choice = getUserChoice(1, 4);
83
84         switch (choice) {
85             case 1:
86                 addFile();
87                 break;
88             case 2:
89                 deleteFile();
90                 break;
91             case 3:
92                 searchFile();
93                 break;
94             case 4:
95                 showMainMenu();
96                 break;
97         }
98     }
99
100     private static void addFile() {
101         System.out.println("Enter file name:");
102         String fileName = scanner.nextLine();
103         File file = new File(DIRECTORY_PATH + "/" + fileName);
104         try {
105             file.createNewFile();
106             System.out.println("File created successfully.");
107         } catch (IOException e) {
108             System.out.println("Error creating file: " + e.getMessage());
109         }
110     }
111
112     private static void deleteFile() {
113         System.out.println("Enter file name to delete:");
114         String fileName = scanner.nextLine();
115         File file = new File(DIRECTORY_PATH + "/" + fileName);
116         if (file.exists()) {
117             if (file.delete()) {
118                 System.out.println("File deleted successfully.");
119             } else {
120                 System.out.println("Error deleting file: " + e.getMessage());
121             }
122         } else {
123             System.out.println("File does not exist in the directory.");
124         }
125     }
126
127     private static void searchFile() {
128         System.out.println("Enter file name to search:");
129         String fileName = scanner.nextLine();
130         List<String> fileNames = getFileNamesInAscendingOrder();
131         if (fileNames.contains(fileName)) {
132             System.out.println("File found in the directory.");
133         } else {
134             System.out.println("File not found in the directory.");
135         }
136     }
137
138     private static void closeApplication() {
139         System.out.println("Application closed.");
140     }
141
142     private static int getUserChoice(int min, int max) {
143         while (true) {
144             System.out.println("Enter your choice:");
145             int choice = scanner.nextInt();
146             if (choice < min || choice > max) {
147                 System.out.println("Invalid choice. Please enter a valid number between " + min + " and " + max);
148             } else {
149                 return choice;
150             }
151         }
152     }
153 }
```

```
SimpleLearn_projects - Final_Project/src/LockedMeApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
SimpleLearn_projects - Final_Project/src/LockedMeAppTest.java
43
44
45     private static void retrieveFileNames() {
46         List<String> fileNames = getFileNamesInAscendingOrder();
47         System.out.println("\nFILE NAMES IN ASCENDING ORDER");
48         if (fileNames.isEmpty()) {
49             System.out.println("No files found in the directory.");
50         } else {
51             for (String fileName : fileNames) {
52                 System.out.println(fileName);
53             }
54         }
55         showMainMenu();
56     }
57
58     static List<String> getFileNamesInAscendingOrder() {
59         File directory = new File(DIRECTORY_PATH);
60         File[] files = directory.listFiles();
61         List<String> fileNames = new ArrayList<>();
62
63         if (files != null) {
64             for (File file : files) {
65                 if (file.isFile()) {
66                     fileNames.add(file.getName());
67                 }
68             }
69             Collections.sort(fileNames);
70         }
71
72         return fileNames;
73     }
74
75     private static void showFileManagementMenu() {
76         System.out.println("\nFILE MANAGEMENT MENU");
77         System.out.println("1. Add a file to the directory");
78         System.out.println("2. Delete a file from the directory");
79         System.out.println("3. Search for a file in the directory");
80         System.out.println("4. Navigate back to the main menu");
81
82         int choice = getUserChoice(1, 4);
83
84         switch (choice) {
85             case 1:
86                 addFile();
87                 break;
88             case 2:
89                 deleteFile();
90                 break;
91             case 3:
92                 searchFile();
93                 break;
94             case 4:
95                 showMainMenu();
96                 break;
97         }
98     }
99
100     private static void addFile() {
101         System.out.println("Enter file name:");
102         String fileName = scanner.nextLine();
103         File file = new File(DIRECTORY_PATH + "/" + fileName);
104         try {
105             file.createNewFile();
106             System.out.println("File created successfully.");
107         } catch (IOException e) {
108             System.out.println("Error creating file: " + e.getMessage());
109         }
110     }
111
112     private static void deleteFile() {
113         System.out.println("Enter file name to delete:");
114         String fileName = scanner.nextLine();
115         File file = new File(DIRECTORY_PATH + "/" + fileName);
116         if (file.exists()) {
117             if (file.delete()) {
118                 System.out.println("File deleted successfully.");
119             } else {
120                 System.out.println("Error deleting file: " + e.getMessage());
121             }
122         } else {
123             System.out.println("File does not exist in the directory.");
124         }
125     }
126
127     private static void searchFile() {
128         System.out.println("Enter file name to search:");
129         String fileName = scanner.nextLine();
130         List<String> fileNames = getFileNamesInAscendingOrder();
131         if (fileNames.contains(fileName)) {
132             System.out.println("File found in the directory.");
133         } else {
134             System.out.println("File not found in the directory.");
135         }
136     }
137
138     private static void closeApplication() {
139         System.out.println("Application closed.");
140     }
141
142     private static int getUserChoice(int min, int max) {
143         while (true) {
144             System.out.println("Enter your choice:");
145             int choice = scanner.nextInt();
146             if (choice < min || choice > max) {
147                 System.out.println("Invalid choice. Please enter a valid number between " + min + " and " + max);
148             } else {
149                 return choice;
150             }
151         }
152     }
153 }
```

```
Simplelearn_projects - Final_Project/src/LockedMeApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Simplelearn_projects - Final_Project/src/LockedMeAppTest.java
100= static void addFile() {
101     scanner = new Scanner(System.in);
102     System.out.print("\nEnter the name of the file to add: ");
103     String fileName = scanner.nextLine();
104
105     File file = new File(DIRECTORY_PATH + File.separator + fileName);
106     try {
107         if (file.createNewFile()) {
108             System.out.println("File added successfully.");
109         } else {
110             System.out.println("File already exists in the directory.");
111         }
112     } catch (IOException e) {
113         System.out.println("An error occurred while adding the file: " + e.getMessage());
114     }
115
116     showFileManagementMenu();
117 }
118
119= static void deleteFile() {
120     scanner = new Scanner(System.in);
121     System.out.print("\nEnter the name of the file to delete: ");
122     String fileName = scanner.nextLine();
123
124     File directory = new File(DIRECTORY_PATH);
125     File[] files = directory.listFiles();
126     boolean fileDeleted = false;
127
128     if (files != null) {
129         for (File file : files) {
130             if (file.isFile() && file.getName().equals(fileName)) {
131                 if (file.delete()) {
132                     System.out.println("File deleted successfully.");
133                     fileDeleted = true;
134                 } else {
135                     System.out.println("Failed to delete the file.");
136                 }
137                 break;
138             }
139         }
140     }
141
142     if (!fileDeleted) {
143         System.out.println("File not found.");
144     }
145
146     showFileManagementMenu();
147 }
148
149= static void searchFile() {
150     scanner = new Scanner(System.in);
151     System.out.print("\nEnter the name of the file to search: ");
152     String fileName = scanner.nextLine();
153
154     File directory = new File(DIRECTORY_PATH);
155
156     Writeable Smart Insert 20:59:756
```

```
Simplelearn_projects - Final_Project/src/LockedMeAppTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Simplelearn_projects - Final_Project/src/LockedMeAppTest.java
1 import org.junit.jupiter.api.Assertions;
2 import org.junit.jupiter.api.BeforeEach;
3 import org.junit.jupiter.api.Test;
4 import java.io.ByteArrayInputStream;
5 import java.io.ByteArrayOutputStream;
6 import java.io.File;
7 import java.io.IOException;
8 import java.io.PrintStream;
9 import java.nio.file.Files;
10 import java.nio.file.Path;
11 import java.util.Arrays;
12 import java.util.List;
13
14 public class LockedMeAppTest {
15     private static final String TEST_DIRECTORY_PATH = "C:\\Users\\santh\\Desktop\\Simplelearn_projects\\Final_Project\\test"; // Update with the actual test directory path
16     private static final String TEST_FILE_NAME = "testFile.txt";
17
18     private File testDirectory;
19     private ByteArrayOutputStream outputStream;
20
21     @BeforeEach
22     void setUp() {
23         testDirectory = new File(TEST_DIRECTORY_PATH);
24         testDirectory.mkdir();
25         outputStream = new ByteArrayOutputStream();
26         System.setOut(new PrintStream(outputStream));
27         deleteTestDirectory();
28         createTestDirectory();
29     }
30
31     private void createTestDirectory() {
32         try {
33             Files.createDirectory(Path.of(TEST_DIRECTORY_PATH));
34         } catch (IOException e) {
35             e.printStackTrace();
36         }
37     }
38
39     private void deleteTestDirectory() {
40         File directory = new File(TEST_DIRECTORY_PATH);
41         if (directory.exists()) {
42             File[] files = directory.listFiles();
43             if (files != null) {
44                 for (File file : files) {
45                     file.delete();
46                 }
47             }
48             directory.delete();
49         }
50     }
51
52     @Test
53     void testRetrieveFileNames_EmptyDirectory() {
54         List<String> expectedFileNames = Arrays.asList();
55         List<String> actualFileNames = LockedMeApp.getFileNamesInAscendingOrder();
56
57     Writeable Smart Insert 152:27:4906
```

```
SimpleLearn_projects - Final_Project/src/LockedMeAppTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LockedMeAppTest.java
58 }
59
60 @Test
61 void testRetrieveFileNames_NonEmptyDirectory() {
62     createTestFile(TEST_FILE_NAME);
63
64     List<String> expectedFileNames = Arrays.asList(TEST_FILE_NAME);
65     List<String> actualFileNames = LockedMeApp.getFileNamesInAscendingOrder();
66
67     Assertions.assertEquals(expectedFileNames, actualFileNames);
68 }
69
70 @Test
71 void testAddFile_FileAddedSuccessfully() {
72     provideUserInput(TEST_FILE_NAME + System.lineSeparator());
73
74     LockedMeApp.addFile();
75
76     String expectedOutput = "File added successfully.";
77     String actualOutput = outputStream.toString().trim();
78
79     Assertions.assertEquals(expectedOutput, actualOutput);
80     Assertions.assertTrue(fileExists(TEST_FILE_NAME));
81 }
82
83 @Test
84 void testAddFile_FileAlreadyExists() {
85     createTestFile(TEST_FILE_NAME);
86     provideUserInput(TEST_FILE_NAME + System.lineSeparator());
87
88     LockedMeApp.addFile();
89
90     String expectedOutput = "File already exists in the directory.";
91     String actualOutput = outputStream.toString().trim();
92
93     Assertions.assertEquals(expectedOutput, actualOutput);
94 }
95
96 @Test
97 void testDeleteFile_FileDeletedSuccessfully() {
98     createTestFile(TEST_FILE_NAME);
99     provideUserInput(TEST_FILE_NAME + System.lineSeparator());
100
101     LockedMeApp.deleteFile();
102
103     String expectedOutput = "File deleted successfully.";
104     String actualOutput = outputStream.toString().trim();
105
106     Assertions.assertEquals(expectedOutput, actualOutput);
107     Assertions.assertFalse(fileExists(TEST_FILE_NAME));
108 }
109
110 @Test
111 void testDeleteFile_FileNotFound() {
112     provideUserInput(TEST_FILE_NAME + System.lineSeparator());
113
114     LockedMeApp.deleteFile();
115
116     String expectedOutput = "File not found.";
117     String actualOutput = outputStream.toString().trim();
118
119     Assertions.assertEquals(expectedOutput, actualOutput);
120 }
121
122 @Test
123 void testGetUserChoice_ValidChoice() {
124     provideUserInput("2" + System.lineSeparator());
125
126     int minChoice = 1;
127     int maxChoice = 3;
128     int expectedChoice = 2;
129     int actualChoice = LockedMeApp.getUserChoice(minChoice, maxChoice);
130
131     Assertions.assertEquals(expectedChoice, actualChoice);
132 }
133
134 @Test
135 void testGetUserChoice_InvalidChoiceThenValidChoice() {
136     provideUserInput("Invalid" + System.lineSeparator() + "2" + System.lineSeparator());
137
138     int minChoice = 1;
139     int maxChoice = 3;
140     int expectedChoice = 2;
141     int actualChoice = LockedMeApp.getUserChoice(minChoice, maxChoice);
142
143     Assertions.assertEquals(expectedChoice, actualChoice);
144 }
145
146 private void createTestFile(String fileName) {
147     File file = new File(testDirectory, fileName);
148     try {
149         file.createNewFile();
150     } catch (Exception e) {
151         e.printStackTrace();
152     }
153 }
154
155 private boolean fileExists(String fileName) {
156     File file = new File(testDirectory, fileName);
157     return file.exists();
158 }
159
160 private void provideUserInput(String userInput) {
161     ByteArrayInputStream inputStream = new ByteArrayInputStream(userInput.getBytes());
162     System.setIn(inputStream);
163 }
164 }
```

```
SimpleLearn_projects - Final_Project/src/LockedMeAppTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
LockedMeAppTest.java
136 void testSearchFile_FileNotFound() {
137     provideUserInput(TEST_FILE_NAME + System.lineSeparator());
138
139     LockedMeApp.searchFile();
140
141     String expectedOutput = "File not found.";
142     String actualOutput = outputStream.toString().trim();
143
144     Assertions.assertEquals(expectedOutput, actualOutput);
145 }
146
147 @Test
148 void testGetUserChoice_ValidChoice() {
149     provideUserInput("2" + System.lineSeparator());
150
151     int minChoice = 1;
152     int maxChoice = 3;
153     int expectedChoice = 2;
154     int actualChoice = LockedMeApp.getUserChoice(minChoice, maxChoice);
155
156     Assertions.assertEquals(expectedChoice, actualChoice);
157 }
158
159 @Test
160 void testGetUserChoice_InvalidChoiceThenValidChoice() {
161     provideUserInput("Invalid" + System.lineSeparator() + "2" + System.lineSeparator());
162
163     int minChoice = 1;
164     int maxChoice = 3;
165     int expectedChoice = 2;
166     int actualChoice = LockedMeApp.getUserChoice(minChoice, maxChoice);
167
168     Assertions.assertEquals(expectedChoice, actualChoice);
169 }
170
171 private void createTestFile(String fileName) {
172     File file = new File(testDirectory, fileName);
173     try {
174         file.createNewFile();
175     } catch (Exception e) {
176         e.printStackTrace();
177     }
178 }
179
180 private boolean fileExists(String fileName) {
181     File file = new File(testDirectory, fileName);
182     return file.exists();
183 }
184
185 private void provideUserInput(String userInput) {
186     ByteArrayInputStream inputStream = new ByteArrayInputStream(userInput.getBytes());
187     System.setIn(inputStream);
188 }
189 }
190
```