```java
package Phase1project;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Filehandle
{

         public void listAllFiles(String path)
         {

        if (path == null || path.isEmpty() || path.isBlank())
            throw new NullPointerException("Path cannot be Empty or null");


        File dir = new File(path);

        if(!dir.exists())
            throw new IllegalArgumentException("Path does not exist");

        if(dir.isFile())
            throw new IllegalArgumentException("The given path is a file. A directory is expected.");



        String files[] = dir.list();
        System.out.println("\n***********************************");

        if(files != null && files.length > 0) {

            Set<String> filesList = new TreeSet<String>(Arrays.asList(files));
            System.out.println("The Files in "+ dir.getAbsolutePath() + " are: \n");

            for(String file1:files) {

                System.out.println(file1);

            }

            System.out.println("\nTotal Number of files: "+ filesList.size());
         else {

            System.out.println("Directory is Empty");
        }

    }


```

```
57        public static void createNewFile(String path , String fileName) throws IOException
58
59
60        {
61
62
63        if (path == null || path.isEmpty() || path.isBlank())
64            throw new NullPointerException("Path cannot be Empty or null");
65
66
67        if (fileName == null || fileName.isEmpty() || fileName.isBlank())
68            throw new NullPointerException("File Name cannot be Empty or null");
69
70        File newFile = new File(path + File.separator + fileName);
71
72        boolean createFile = newFile.createNewFile();
73
74        if (createFile) {
75
76            System.out.println("\nFile Successfully Created: " + newFile.getAbsolutePath());
77
78         else if(!createFile) {
79
80            System.out.println("\nFile Already Exist.. Please try again." );
81
82        }
83
84
85    }
86
87
88
89        public static void deleteFile(String path , String fileName) throws IOException
90        {
91
92        if (path == null || path.isEmpty() || path.isBlank())
93            throw new NullPointerException("Path cannot be Empty or null");
94
95
96        if (fileName == null || fileName.isEmpty() || fileName.isBlank())
97            throw new NullPointerException("File Name cannot be Empty or null");
98
99        File newFile = new File(path + File.separator + fileName);
100
101       boolean deleteFile = newFile.delete();
102
103       if (deleteFile) {
104
105           System.out.println("\nFile deleted Successfully");
106
107        else {
108
109           System.out.println("\nFile Not Found.. Please try again." );
110
111       }
112
113   }
```

```
114
115
116
117         public static void searchFile(String path , String fileName)
118         {
119
120         if (path == null || path.isEmpty() || path.isBlank())
121             throw new NullPointerException("Path cannot be Empty or null");
122
123
124         if (fileName == null || fileName.isEmpty() || fileName.isBlank())
125             throw new NullPointerException("File Name cannot be Empty or null");
126
127         File dir = new File(path);
128
129         if(!dir.exists())
130             throw new IllegalArgumentException("Path does not exist");
131
132         if(dir.isFile())
133             throw new IllegalArgumentException("The given path is a file. A directory is
    expected.");
134
135
136         String [] fileList = dir.list();
137         boolean flag = false;
138
139         Pattern pat = Pattern.compile(fileName);
140
141         if(fileList != null && fileList.length > 0) {
142             for(String file:fileList) {
143                 Matcher mat = pat.matcher(file);
144                 if(mat.matches()) {
145                     System.out.println("File Found at location: " + dir.getAbsolutePath());
146                     flag = true;
147                     break;
148                 }
149             }
150         }
151
152         if(flag == false)
153             System.out.println("File Not Found.. Please try again.");
154
155
156     }
157
158         public static void readFile(String filename,String path)
159         {
160             File myFile = new File(path + File.separator + filename);
161             System.out.println("-------------------------------------");
162             System.out.println("---------------CONTENTS :------------");
163             try {
164                 Scanner sc = new Scanner(myFile);
165                 while(sc.hasNextLine()){
166                     String line = sc.nextLine();
167                     System.out.println(line);
168                 }
169                 System.out.println("-------------------------------------");
```

```
170                 System.out.println("READING :-> SUCCESSFUL");
171                 System.out.println("-------------------------------------");
172                 sc.close();
173             } catch (FileNotFoundException e) {
174                 e.printStackTrace();
175                 System.out.println("-------------------------------------");
176                 System.out.println("READING :-> FAILED");
177                 System.out.println("-------------------------------------");
178 
179             }
180 
181         }
182 
183         public static void writeFile String filename,String path)
184         {
185             File myFile = new File(path + File.separator + filename);
186 
187          try
188             {
189 
190                 Scanner in = new Scanner(System.in);
191                 System.out.println("------------------------------");
192                 System.out.println("***** START WRITING *****");
193                 String Str = in.nextLine();
194                 FileOutputStream fileout = new FileOutputStream(myFile);
195                 byte b[] = Str.getBytes();
196                 fileout.write(b);
197                 System.out.println("------------------------------");
198                 fileout.close();
199                 System.out.println("************************************");
200                 System.out.println("WRITING :-> SUCCESSFUL");
201                 System.out.println("************************************");
202             }
203          catch(Exception e)
204          {
205 
206              System.out.println(e);
207              System.out.println("************************************");
208              System.out.println("WRITING :-> FAILED");
209              System.out.println("************************************");
210          }
211 
212     }
213 }
214 
215 
216 
217 
218 
```