**Lesson-13 – Files and Databases**

**Problem-1 [ 5 Point ]**

Create a Class FileSearch with the static method,

**public** **static** **boolean** searchForFile(String filename, String startDir){

}

Will search for a file with filename = filename( no path information is contained in the filename), starting from a directory whose full path is given by the input value startDir.

If the file is found, its contents are printed to the console, and the methos returns true. If the file is not found, the method simply return false.

In the probone package, Test class is provided that will create a directory structure and a simple file(in the setup method), and then will run your searchFor File method to verify that the simple file is found. A skeleton of the FileSearch class is also provided in this folder.

Note: For the Mac users use forward (\) in the file path, Windows users (//), if you found any issues.

// Pseudo code representation of File Search, which is easy to convert Java Code.

boolean searchForFile(Object file, Object startDir) {

Object[] fileSystemObjects = startDir.getContents();

for(Object o: fileSystemObjects) {

//base case

if(isFile(o) && isSameFile(o,file)) {

return true;

}

if(isDirectory(o)) {

searchForFile(file, o);

}

}

//file not found in startDir

return false;

}

**Problem-2 [ 10 Points ]**

You are tasked with implementing a **Word Frequency Counter** that reads text from a file named article.txt, processes the words, and displays the frequency of each unique word in the text. The program should follow these requirements:

1. **File Handling:**
   * Your program should **write** an article (provided in the problem description) into article.txt.
   * It should then **read** the file contents for further processing.
2. **Text Processing:**
   * Convert all words to **lowercase** to avoid case sensitivity issues (e.g., "Java" and "java" should be counted as the same word).
   * Remove **punctuation** such as commas, periods, and special characters.
   * Split the text into words and count their occurrences.
3. **Data Storage:**
   * Use a **HashMap<String, Integer>** to store words as keys and their frequencies as values.
4. **Output Display:**
   * Print the **word frequency table** in a readable format.

**Problem 3 – Working with Database [ 10 Points ]**

Perform Loading data from the book table to JTable, Insert, Delete, and Update on the book table as per the screenshot below with JTable and book table.

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |
| Click Add button to insert data into the Database and JTable  A screenshot of a computer  AI-generated content may be incorrect. | After Insertion  A screenshot of a computer  AI-generated content may be incorrect. |
| Click Update button to modify the data, Select the row from the Table and update in the JTable and table.  A screenshot of a computer  AI-generated content may be incorrect. | After Update  A screenshot of a computer  AI-generated content may be incorrect. |
| Before Deletion  A screenshot of a computer  AI-generated content may be incorrect. | After Clicking Deletion  A screenshot of a computer  AI-generated content may be incorrect. |

UI design with the startup codes given in the probthree package. Perform the Loading data, Insert, Delete and Update.

--------------------------------------------------------------------------------------------------------------