|  |
| --- |
| LockedMe.com  2022 |
|  |
| September 22  Company Lockers Pvt. Ltd.  Authored by: Ashish Kumar Pathak |

# Developer Details

|  |
| --- |
| Application Name: LockedMe.com Name: Ashish Kumar Pathak  Designation: Java Developer |
| *GITHUB LINK TO PROJECT:*  https://github.com/ashu143k/LockedMe.git |
| *Algorithm and Flow chart of program*  Step 1: Start the program with Creation of class name LockedMe.  Step 2: Creating a static object of Class Scanner as scan.  Step 3: A print statement for Welcome Screen and Developer details.  Step 4: In Main method we create a Loop of type while(true).  Step 5: Print Statement as -----Main Menu-----.  Step 6: Print Statement to give choice for user as  Enter 1: To get file names in ascending order  Enter 2: For Business Level Operations  Enter 3: To Close the program.  Step 7: Scanner object scan the input in integer variable - choice.  Step 8: We create a switch (choice) where for each given in input choice from user calls  The relevant method.  **Pictorials Representation of Class Creation and Main method:**  Step 9: User Input 1 in Main Menu Options i.e., Printing files in Ascending order,  Case: 1 is invoking ascendingOrderFile().  Step 10: Inside as ascendingOrderFile() Method we create an Object of File class to get the directory.  Step 11: Condition fileDir.isDirectory() if condition pass, Creation of a List for directory, Step 12: The list gets sorted using Collections.sort() method then list is printed using Iterator object with help of for loop.  If Condition fails, Print statement that given directory is empty or not a directory.  **Pictorials Representation flow of ascendingOrderFile() Method:** |

Step 12: User Input 2 in Main Menu Options i.e., Business Level Operation, Case: 2 is invoke businessLevelOperation().

Step 13: Inside businessLevelOperation() method there are multiple print statement as

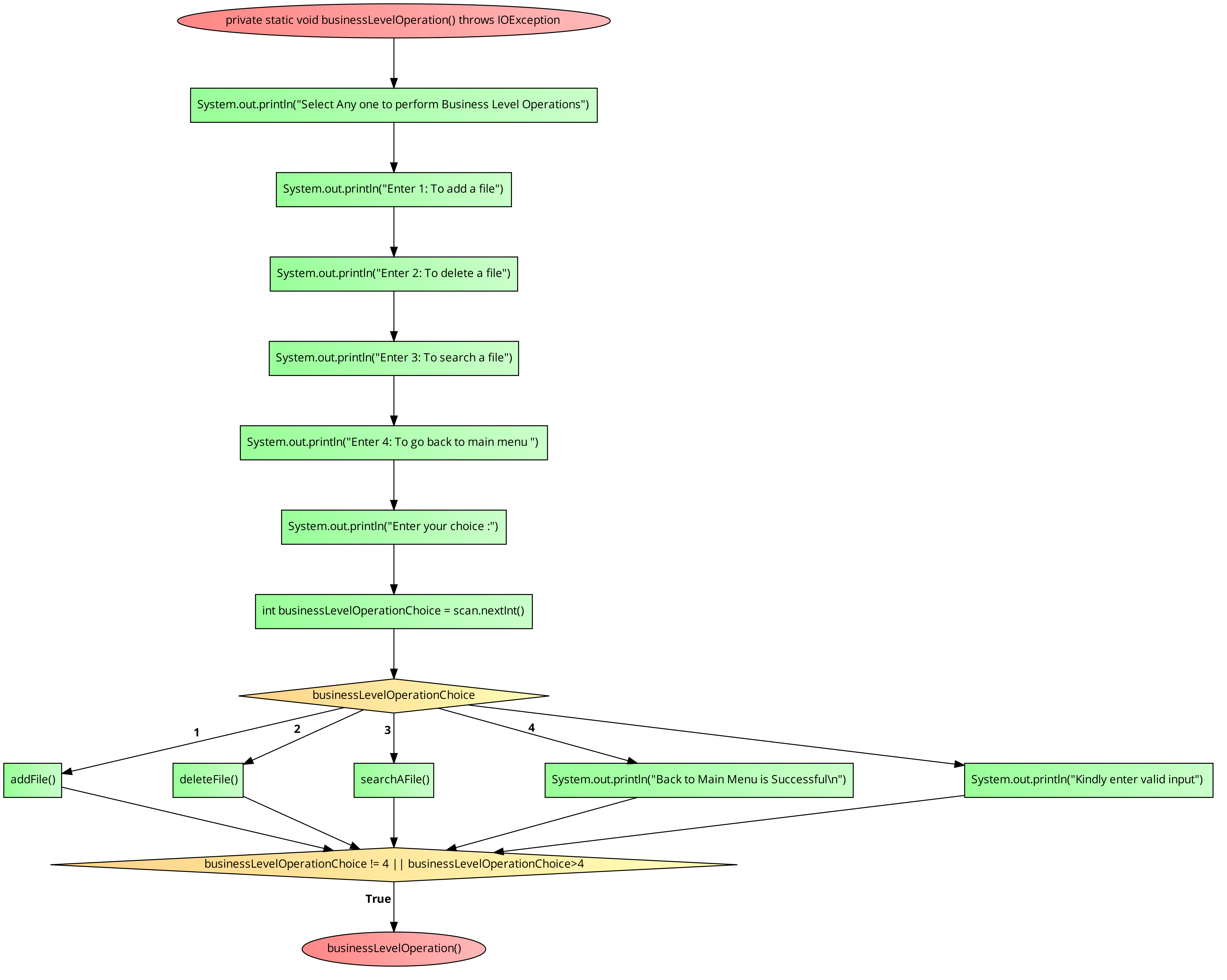
Select any one to perform Business Level Operations i.e.,1. To Add a file, 2. To delete a file, 3. To Search a file, 4. To go back to Main menu.

Step 14: We ask user to give a choice and store in integer Variable as businessLevelOperationChoice.

Step 15: We use switch conditional statement in which for each businessLevelOperationChoice input it calls to particular case and a method is executed.

Step 16: If statement for businessLevelOperationChoice Variable is not equal or greater than 4 we calls the businessLevelOperation() again.

**Pictorials Representation flow of businessLevelOperation() Method:**



Step 17: User Input 1 in businessLevelOperation() method i.e., Adding a file,

Case: 1 is invoke addFile().

Step 18: Inside addFile() method print “Enter your file name”.

Step 19: Variable String type filename takes the input from user.

Step 20: Using File object as newFile we get the directory.

Step 21: If newFile.createNewFile() returns true the files get create with message “File is created”.

Step 22: If Condition fails i.e., False we gets message as “File is Already Exist, Please try again with different name for file”. And addFile() called again.

**Pictorials Representation flow of addFile() Method:**



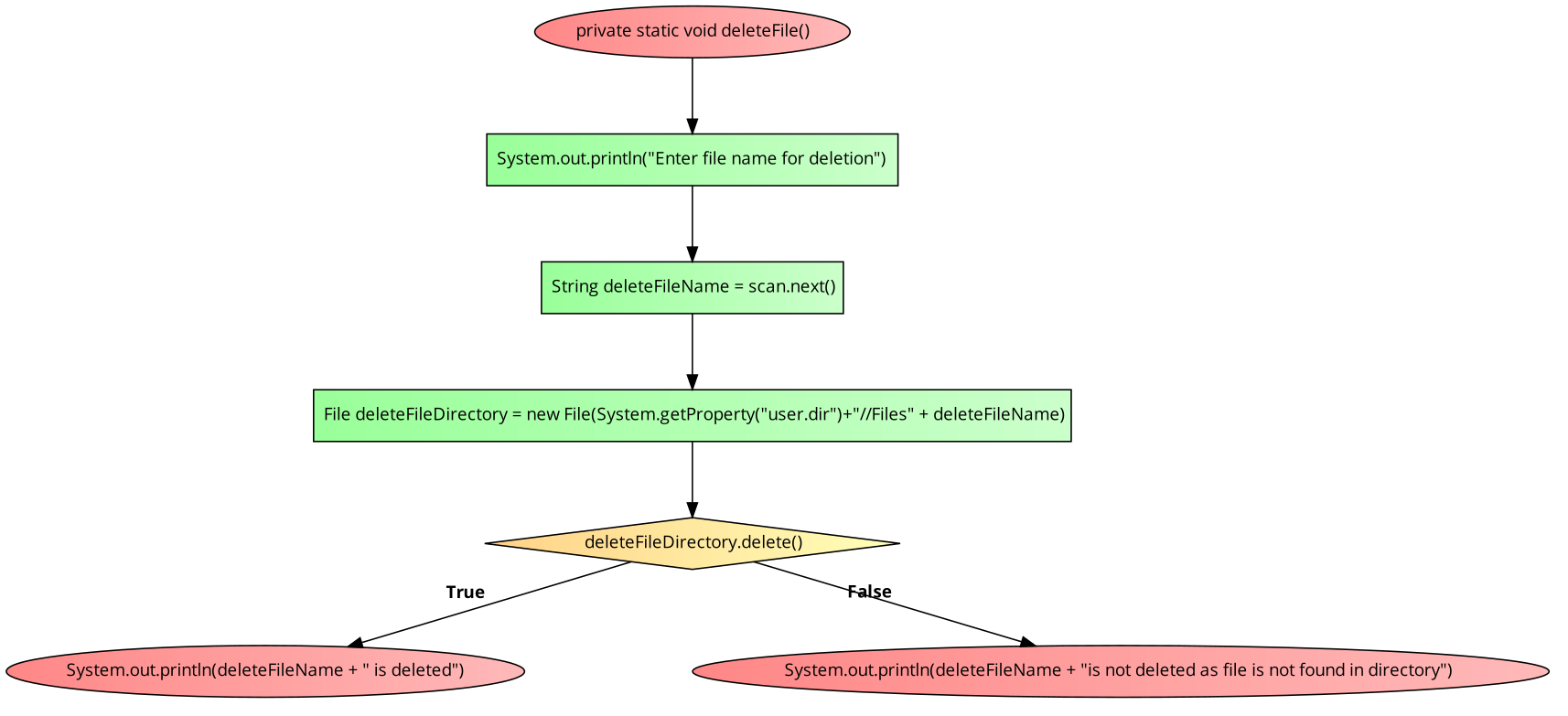
Step 23: User Input 2 in businessLevelOperation () method i.e., deletion of a file, Case: 2 is invoke deleteFile().

Step 24: Inside the method Print statement “Enter file name for deletion”, String Variable deleteFileName takes input from user.

Step 25: Using File object as deleteFileDirectory we get the directory where the file exists by given deleteFileName variable.

Step 26: If deleteFileName.delete() returns true print “File is deleted” else “file cannot be deleted as its not found in directory.

Pictorials Representation flow of deleteFile() Method:



Step 27: User input 3 in businessLevelOperation() method i.e., To search a file, Case: 3 is invoke searchAFile().

Step 28: Using File class object is created as searchFileDirectory, in which file is searched.

Step 29: If searchFileDirectory.isDirectory returns true. Print statement “Enter file name to search”, String searchFileName takes input from user.

Step 30: List object listFile is created and stores the directory files in ArrayList format.

Step 31: Iterator of type String is created as searchIterate for listFile and Boolean flag initialized with false.

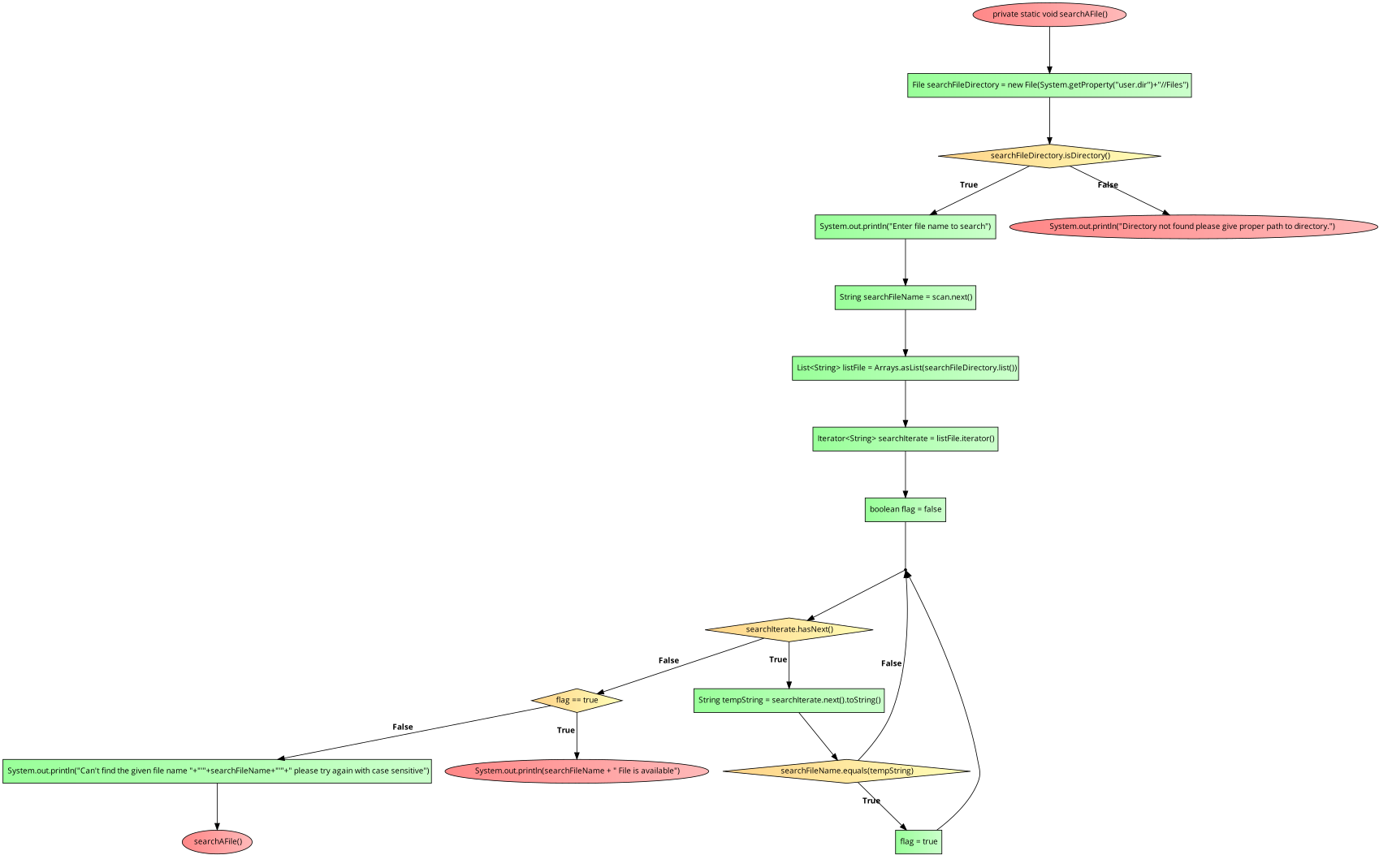
Step 32: For searchIterate.hasNext() if true, searchIterate.next().toString() is stored in String tempString.

Step 33: If searchFileName.equals(tempString) is true then flag set to True. Else it iterates for rest of list.

Step 34: If flag is true print “Given file is found” else “Can’t find the given file name, please try again with case sensitive”. searchAFile() is revoked.

Step 35: If searchFileDirectory.isDirectory returns false, Print Statement “Directory not found please give proper path to directory.

Pictorials Representation flow of searchAFile() Method:



Step 36: User input 4 in businessLevelOperation() method i.e., To go back to Main Menu, Case: 4 which print “Back to Main Menu is successful”. Back to Main Menu.

Step 37: User Input 3 in Main Menu Options i.e., To Close the Application, Case: 3 is executed with display message “Application is closed”. System.exist(choice);

Step 38: Stop the program.