Project: **LockedMe.com**

Developer: Siddharth Basu

- Project details:

Company Lockers Pvt Ltd is aiming to digitise a project LockedMe.com which is a file handling system capable of performing operations such as create, read, get and update file attributes from the console on a particular directory folder based on user input. It is a menu based application specifying the user to input certain numbers from the displayed menu to perform their corresponding file operations.

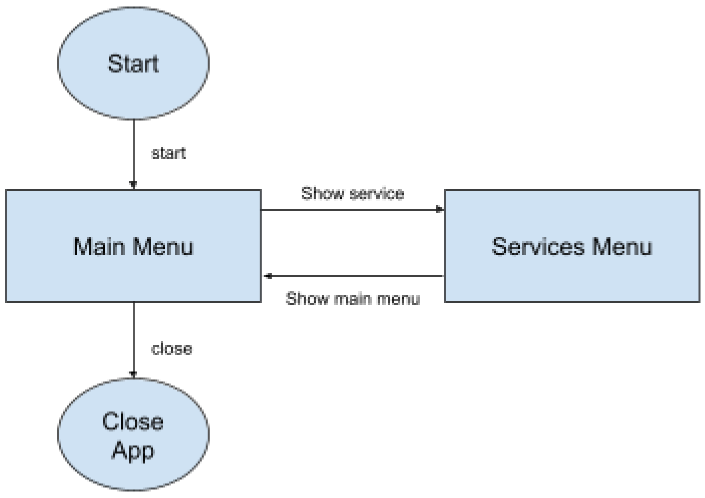
For the prototype the directory where the file operations will be done is ~/Files folder in the working directory of the Java application.

- Github Link:

https://github.com/siddharth-basu98/LockedMe.com_console_app

- Services offered in the Application:

The application is divided into two tiers, namely, the main menu and the services menu.



Main Menu: Displays the name of the application and the developer details and the directory where file operations are going to be done.

The main menu displays two options to the user, namely

1) Show the services menu
2) Close the application

Services Menu: This is the main menu of the application where the services offered are mentioned to the user and he/she inputs numbers to perform them.

The service menu displays 11 options to the user, namely,

1) Create and add the file in the directory
2) Delete a file from the directory
3) Get a sorted list of all the files
4) Get files created on a particular date
5) Get files created before a particular date
6) Get files created after a particular date
7) Get files by file extensions
8) Get files by creator name
9) Get files by file name
10) Update the creator name of the file
11) Return back to the main menu

AGILE SCRUM ACTIVITIES:

1) **Sprint Planning:**
   During this stage a list of user stories are prioritised to be implemented during the sprint. This is attended by the entire Scrum team: the project owner, the scrum master and the development team.
   Product backlog → Sprint Backlog

2) **Daily Scrum:**
   This is a 15 minute meting at the start of the day conducted among the development team to discuss the work done a day prior, work to be done today and impediments if any.

3) **Sprint Review Meeting:**
   This is the meeting scheduled for the entire scrum team and even other relevant stakeholders at the end of a single sprint for feedback and collaboration. The team demonstrates the work done The project owner either accepts or rejects the work. Finally if required the product backlog is revised based on the feedback.

4) **Sprint Retrospective:**
The is part where the scrum team focuses on the process and discusses what went well in the sprint, what didn't go well and what needs to be done differently the next time.

USER STORIES:

– As a user of the system
I would like to have the functionality to **add a file to the directory**
So that when I want to add a file I can do it from the system.

Acceptance:
- Program should ask file details adding the file.
- If a file with the same name exists, return "file already exists".

– As a user of the system
I would like the functionality of **deleting a file**
So that when I deem a file to be unnecessary, I can do it from the system.

Acceptance:
- Program should return a "File not found" message if the file is not found.
- Case sensitivity should be considered while deleting a file.

– As a user of the system
I would like get a **sorted list of all the files** present in the current directory on name
So that I can view the files present in the system.

Acceptance:
- Case sensitivity should be maintained while sorting according to capital letters considered to be smaller than small letters (as per ascii).
- Extensions should be considered to be a part of the file name while sorting.
- If no files exist, return a message "there are no files in the directory".

– As a user of the system
I would like to **get files on the basis of the date** on which they were created
So that I can get a list of files based on the date of creation.

Acceptance:

- If no files were created on that particular date, display the message “no files found on this date”
- If the input date is not of the form “YYYY-MM-DD” display invalid date input.

- As a user of the system
  I would like to **get list of files created before/after a user input date**
  So that I can get a list of files hinting on the date of creation.

  Acceptance:

  - If no files were created on that particular date, display the message “no files found on this date”
  - If the input date is not of the form “YYYY-MM-DD” display invalid date input.

- As a user of the system
  I would like to **get a list of files having the same extension**
  So that I can view similar files in one go.

  Acceptance:

  - The file name must have a valid extension of length greater than 0, otherwise display invalid extension name.
  - If no files have the entered extension, display “no files belonging to the given extension”

- As a user of the system
  I would like to have the functionality of **retrieving files created by a specific creator**
  So that I can get a list of files created by a user to check for only his/her work.

  Acceptance:

  - The entered creator name should have length greater than 0, otherwise display “The entered user must be valid”
  - If no files exist for the creator, then display “No files found belonging to this creator”

- As a user of the system
  I would like to have the functionality of **retrieving details of a file based on its name**
  So that I can see other metadata of the specified file.

  Acceptance:
  - The file name entered should have proper extensions and minimum length 3
  - If no file exists with the given name then just return the message “no file with that name exists in the directory”

- As a user of the system
  I would like the functionality of **renaming/updating the creator of the file**
  So that when a file has a wrong creator name, I can rename it from the system.

  Acceptance:
  - If the file whose name needs to be changed doesn’t exists, return a “wrong file mentioned” message.
  - The new creator name should be valid having length > 0.

- As a user of the system
  I would like to have the the **feature of a rollback to main menu**.
  So that I can navigate back to the main menu when I want to.

  Acceptance:
  - Any change in the contents of the directory should be committed to the data store and be reflected in the directory. If rollback is selected the main menu should be displayed.

- As a user of the system
  I would like to have the **feature of exit the application**
  So that when I am done using the system, I can close the application safely.

  Acceptance:
  - When the program needs to be closed all changes done should reflect in the directory.

SPRINTS PLANNED:

Sprint 1: Activities planned,

- Develop a main menu of the application
- Develop the service menu of the application
- Add functionality to add files at the given directory
- Add functionality to delete files from a given directory
- Get a list of all files in the directory sorted by name of the file
- Get the list of files created on a particular date

Sprint 2: Activities planned,

- Get the list of files created before a particular date
- Get the list of files created after a particular date
- Get files by file extension name
- Get file by the creator name
- Get file by the file name
- Update the creator of the file

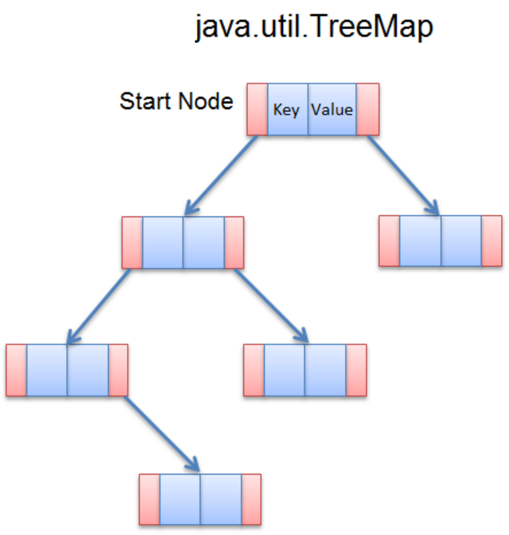
MODEL LAYER : **File Object:**

File Object Attributes
……………………….

→ File Name (String)
→ File ID (int)
→ File creator name (String)
→ Date of creation (Date)
→ File extension (String)
→ File description (String)

Data Access Object (DAO) Layer

Data Structure: Treemap <String, FileObject>



- **Reason:**

  Since a directory can not contain duplicate files having the same name, the TreeMap key can be served with the file name. Further since we need to give the sorted list of all files in the directory according to the name, therefore, since a TreeMap always stores values in the sorted order of the keys, this will reduce the complexity of sorting and searching based on the file name.

- **Sorting Algorithm:**

  The implementation of the TreeMap is that of a self-balancing Binary Search Tree. An inorder traversal of the tree returns the list of values correspond to the sorted list of keys defined in the algorithm. While the insertion takes O(log n) complexity, getting the sorted list of file is a O(n) inorder traversal over the data structure.

- **Searching Algorithm:**

  The searching algorithm is a O(log n) search from the root node following a path down and on each step either going to the left child or the right child till either we find the node or reach a leaf node. Since it follows a Binary Search Tree principle, all nodes to the right subtree are bigger than the node, and the left subtree is smaller. So it takes a decision to go left or right by comparing the key value of the node and the target node.

JAVA CONCEPTS USED:

- **Three layered Architecture:** Constructed the application following the three layer architecture by dividing the app into a presentation layer (console app) in the Main directory, A model layer containing the structure of the File Object we need to store and a service or business layer in between.

- **Exception Handling**: Created the application using checked exceptions by developing an exception layer by extending the Exception class and throwing exceptions throughout the application to prevent the abnormal abortion of the program.

- **Interfaces:** Constructed the interfaces for the DAO and Service layers and instantiated model objects through the interfaces as a design principle.

- **Classes and Objects:** The class is a template consisting of data members and methods against which many objects of its type are instantiated. In our case a single file is saved as an object and has many data members.

- **Collections Interface:** According to the requirements of the application, a TreeMap<> was adjudged as the best candidate for the data structure. It falls under the Sorted Map interface.

- **Lambda functions:** The comparators in the application were designed using Lamda expressions to make the code more functional. The predicates inside the filter method were also written in lambda notation.

- **Stream :** The stream function was used while checking the values of the TreeMap one by one to check for a certain predicate and collect the satisfying objects onto a result List interface object.

- **Data Hiding:** Most of the data members defined in the classes were defined as private with appropriate setters and getters so that any alien function isn’t able to change its values from outside without proper access.

- **Throws and throw:** These keywords were apty used in method definitions and inside functions to indicate the possibility of throwing an exception and actually throwing one if certain conditions are met.

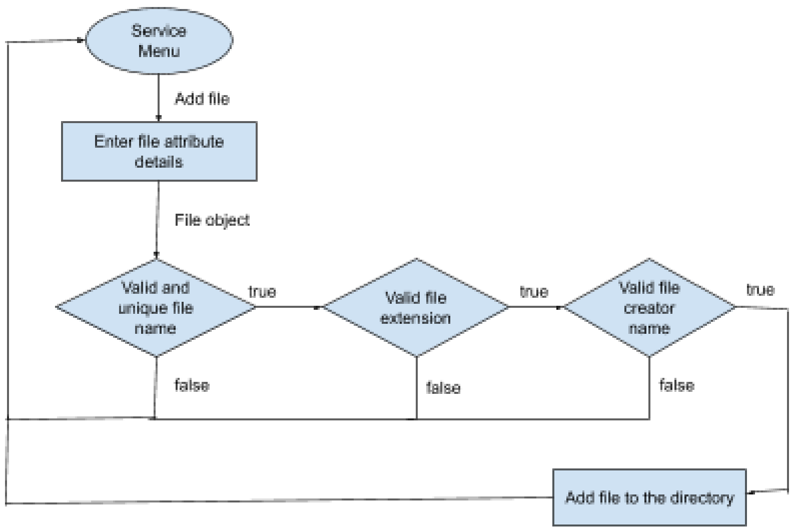
- **Agile:** The agile framework was followed while doing the project by dividing the work to be done in sprints and implementing user stories in a phase wise manner.

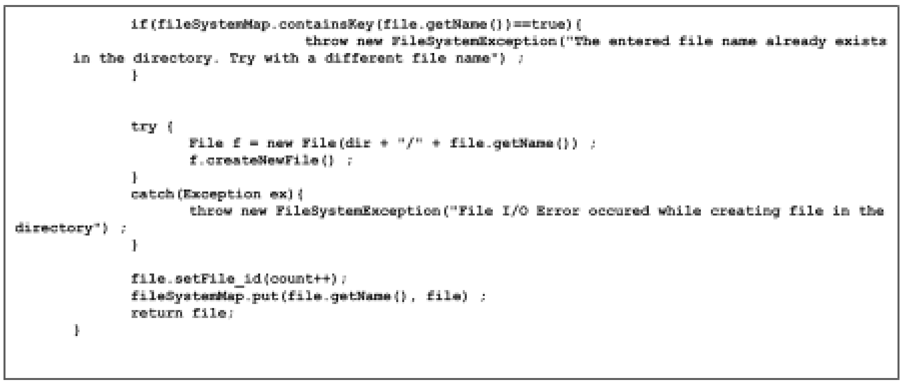
- **File I/O:** While creating files in the directory, file IO was used to add and delete files in the directory.

- **Git and Github:** To have version control in our system, git and github was used.

FLOWCHARTS AND DAO IMPL SNIPPETS (Validations handled in service layer not shown in the snippet. Refer to code for that):

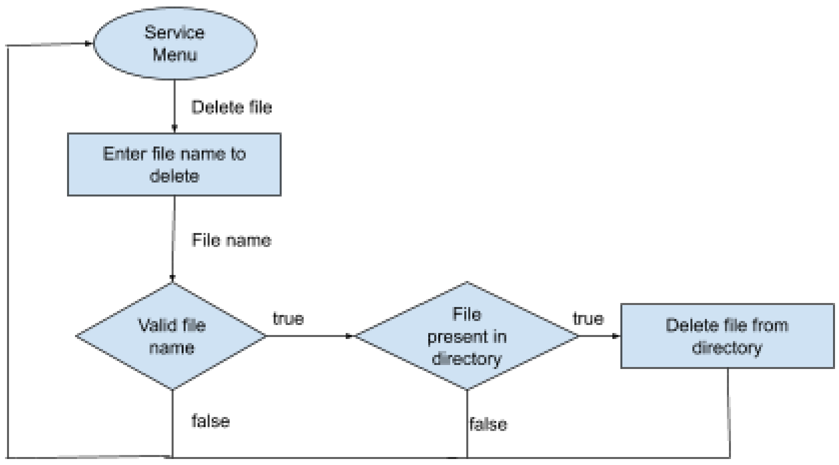
1) Add file to directory:





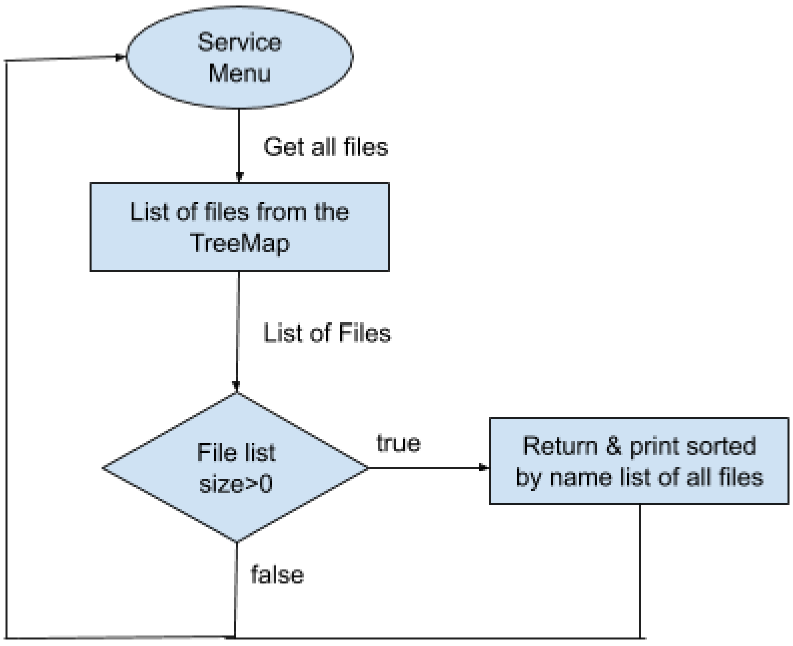
```
            if(fileSystemMap.containsKey(file.getName())==true){
                            throw new FileSystemException("The entered file name already exists
        in the directory. Try with a different file name") ;
            }


            try {
                    File f = new File(dir + "/" + file.getName()) ;
                    f.createNewFile() ;
            }
            catch(Exception ex){
                    throw new FileSystemException("File I/O Error occured while creating file in the
directory") ;
            }

            file.setFile_id(count++);
            fileSystemMap.put(file.getName(), file) ;
            return file;
        }
```

2) Delete file from directory


Service Menu
Delete file
Enter file name to delete
File name
Valid file name
true
File present in directory
true
Delete file from directory
false
false

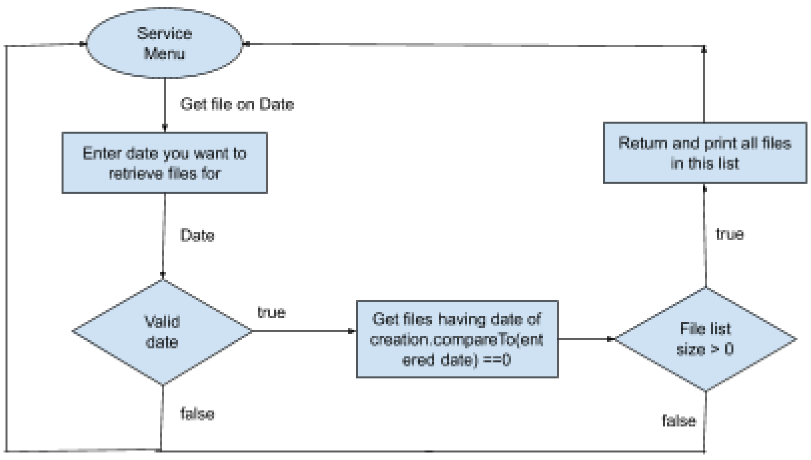
```
                if(fileSystemMap.containsKey(fileName)==false) {
                        throw new FileSystemException("The entered file name
doesn't exist in the directory") ;
                }

                String file_name = dir+"/" + fileName ;
                File toDeleteObj = new File(file_name);
                if (toDeleteObj.delete()) {
               System.out.println("Deleted the file: " + fileName);
             } else {
               throw new FileSystemException("The entered file name doesn't
exist in the directory") ;
             }
                fileSystemMap.remove(fileName) ;
```

3) Get sorted list of all files in the directory


Service Menu
Get all files
List of files from the TreeMap
List of Files
File list size>0
true
Return & print sorted by name list of all files
false

```
                List<FileObject> file_list= new
                ArrayList<FileObject>(fileSystemMap.values()) ;

                if(file_list.size()==0) {
                        throw new FileSystemException("There are currently no
files in the directory") ;
                }
                else {
                        return file_list ;
                }
```

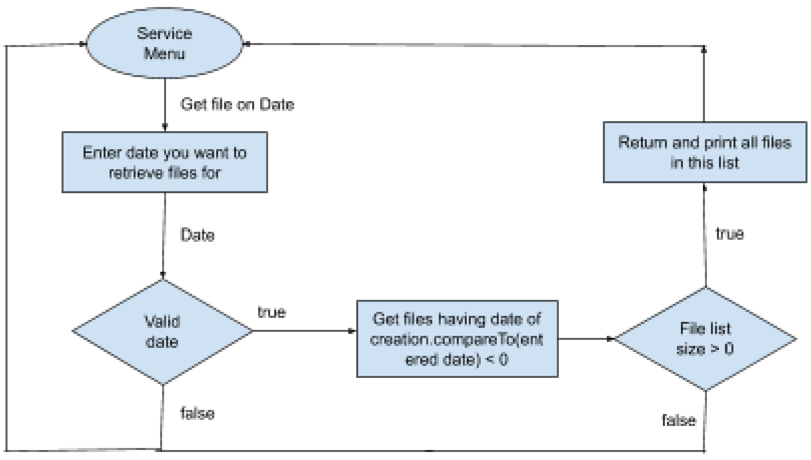
4) Get files created on a particular date



Service Menu
Get file on Date
Enter date you want to retrieve files for
Date
Valid date
true
false
Get files having date of creation.compareTo(entered date) ==0
File list size > 0
true
false
Return and print all files in this list

```
            List<FileObject> fileList = fileSystemMap.values().stream()
                                            .filter(s ->
            s.getDateCreated().compareTo(date)==0)
                              .collect(Collectors.toList());

            if(fileList.size()==0) {
                   throw new FileSystemException("There are no files
 created on this date") ;
            }
            else {
                   return fileList ;
            }
```

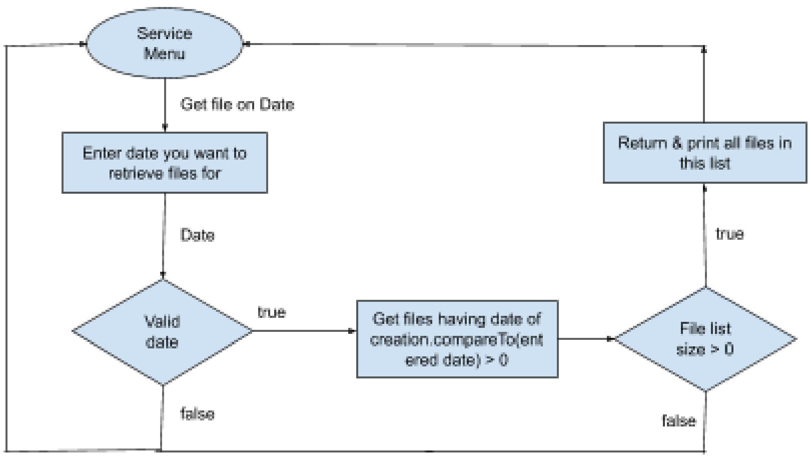
5) Get files created before a particular date



Service Menu
Get file on Date
Enter date you want to retrieve files for
Date
Valid date
true
false
Get files having date of creation.compareTo(entered date) < 0
File list size > 0
true
false
Return and print all files in this list

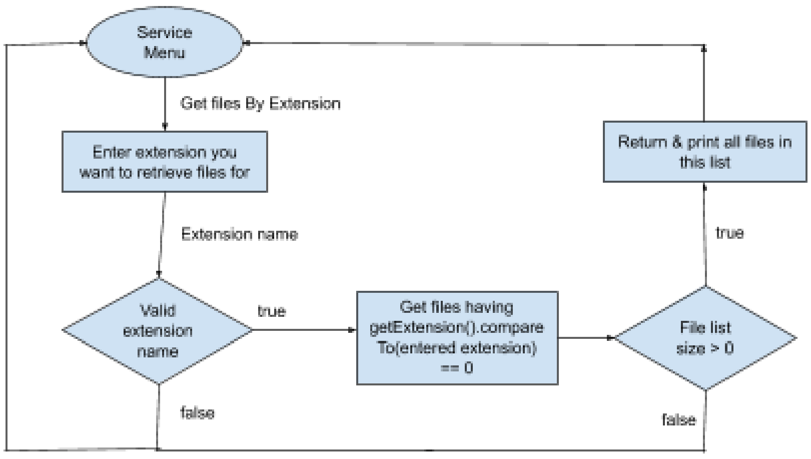
```
              List<FileObject> fileList = fileSystemMap.values().stream()
                                               .filter(s ->
              s.getDateCreated().compareTo(date)<0)
                           .collect(Collectors.toList());

              if(fileList.size()==0) {
                     throw new FileSystemException("There are no files
created on this date") ;
              }
              else {
                     return fileList ;
              }
```

6) <u>Get files created after a particular date</u>


Service Menu
Get file on Date
Enter date you want to retrieve files for
Date
Valid date
true
false
Get files having date of creation.compareTo(entered date) > 0
File list size > 0
true
false
Return & print all files in this list

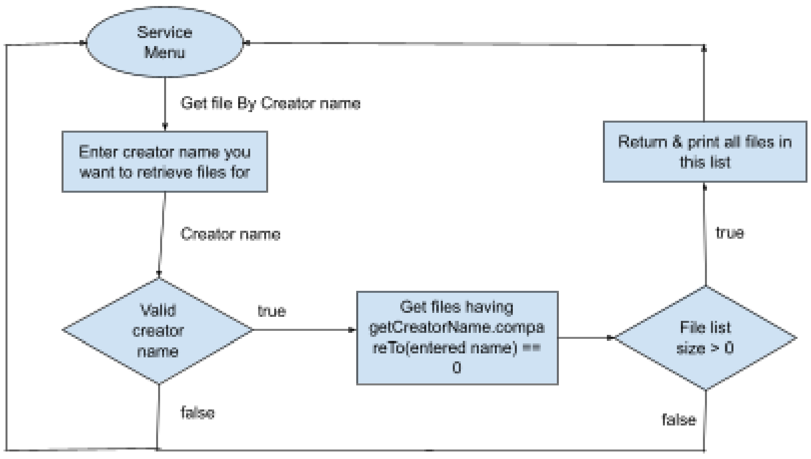
```
              List<FileObject> fileList = fileSystemMap.values().stream()
                                               .filter(s ->
              s.getDateCreated().compareTo(date)>0)
                           .collect(Collectors.toList());

              if(fileList.size()==0) {
                     throw new FileSystemException("There are no files
created on this date") ;
              }
              else {
                     return fileList ;
              }
```

7) Get files by extension name


Service Menu
Get files By Extension
Enter extension you want to retrieve files for
Extension name
Valid extension name
true
false
Get files having getExtension().compareTo(entered extension) == 0
File list size > 0
true
false
Return & print all files in this list

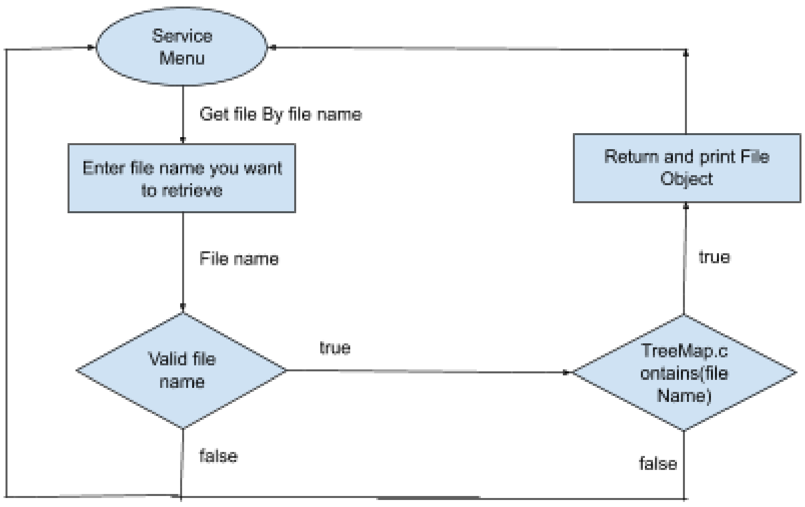
```
              List<FileObject> fileList = fileSystemMap.values().stream()
                                    .filter(s ->
s.getExtension().compareTo(extension)==0)
                             .collect(Collectors.toList());

              if(fileList.size()==0) {
                     throw new FileSystemException("There are currently no
files with this extension") ;
              }
              else {
                     return fileList ;
              }
```

8) Get files by Creator name


Service Menu
Get file By Creator name
Enter creator name you want to retrieve files for
Creator name
Valid creator name
true
false
Get files having getCreatorName.compareTo(entered name) == 0
File list size > 0
true
false
Return & print all files in this list

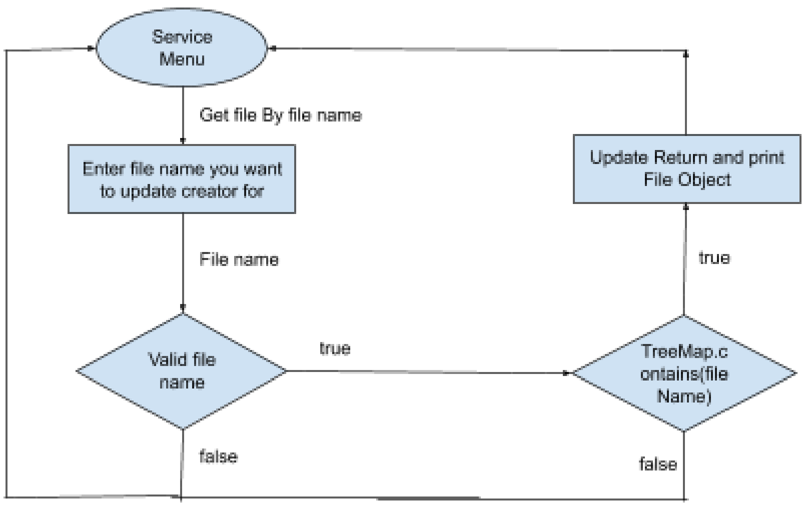
```
        List<FileObject> fileList = fileSystemMap.values().stream()
                                    .filter(s ->
s.getCreatorName().compareTo(name)==0)
                                    .collect(Collectors.toList());

                if(fileList.size()==0) {
                        throw new FileSystemException("There are no files
created by the entered user.") ;
                }
                else {
                        return fileList ;
                }
```

9) Get files by name


Service Menu
Get file By file name
Enter file name you want to retrieve
File name
Valid file name
true
false
TreeMap.contains(fileName)
true
false
Return and print File Object

```
                if(fileSystemMap.containsKey(name)==false) {
                        throw new FileSystemException("The file you requested
for doesn't exist in the directory") ;
                }
                else {
                        return fileSystemMap.get(name) ;
                }
```

10) <u>Update creator name of a file</u>


Service Menu
Get file By file name
Enter file name you want to update creator for
File name
Valid file name
true
false
TreeMap.containsKey(fileName)
true
false
Update Return and print File Object

```
            if(fileSystemMap.containsKey(fileName)==false) {
                throw new FileSystemException("There is no file in
the directory with the given name") ;
            }

            FileObject currentFile = fileSystemMap.get(fileName) ;
            currentFile.setCreatorName(newCreatorName);

            fileSystemMap.put(fileName, currentFile) ;

            return currentFile;
```

<u>**Future Scope:**</u>

- Ability to add content to the files and append any information from the terminal taking user input.
- A file system persistent database instead of a volatile data structure.
- More metadata and specialised searched based on file contents.

****