Virtual Key for Your Repositories

Phase 1 Project – Specification Document

Name: K V Sagar

Project and Developer details:

- **Lockers Pvt. Ltd.** is looking to digitalise their products and their pilot project is named **LockedMe.com** .
- This project will be able to create a virtual key for the repository.
- The user will be able to:
    1. Display the existing files in the repository in a sorted order (ascending).
    2. Add a new file to the directory.
    3. Delete a file from the directory.
    4. Search for a file from the directory.
- To manually add, delete or search a file from the directory is quite tedious and time consuming, this product will help reduce the time taken and simplify the user experience.
- This product will feature 2 separate menu options, the **Main Menu** and the **Action Menu**.
- The **Main Menu** proposes 3 options to the user, they are as follows:
    1. Display the existing files in the directory in a sorted order (ascending).
    2. Enter the Action Menu.
    3. Exit the application.
- The Main Menu also displays the application name i.e. LockedMe.com and displays the developer details.
- The sub-menu i.e., the **Action Menu** has 4 options, they are as follows:
    1. Add a new file to the directory.
    2. Delete a file from the directory.
    3. Search for a file in the directory.
    4. Go back to the Main Menu.
- Developer name is **K V Sagar**.

Sprints Planned and Task Achieved

- The total duration of this is 15 days i.e. 3 weeks.
- The has been divided into 3 Sprints each of 5 days.
- The goals of each of the sprints are mentioned below:

1. **Sprint 1:**
    - A well defined backlog of all the tasks is created to ensure on time delivery.
    - The first sprint is dedicated to the design development of the application.
    - The developer considers all the requirements of the client and decides the approach to be taken.
    - As this is a menu driven application the do-while loop has been used to program the menu.
    - A switch case is used in order to accept input from user and present the requested option.
    - Tasks Achieved : At the end of Sprint 1, the design was finalized.

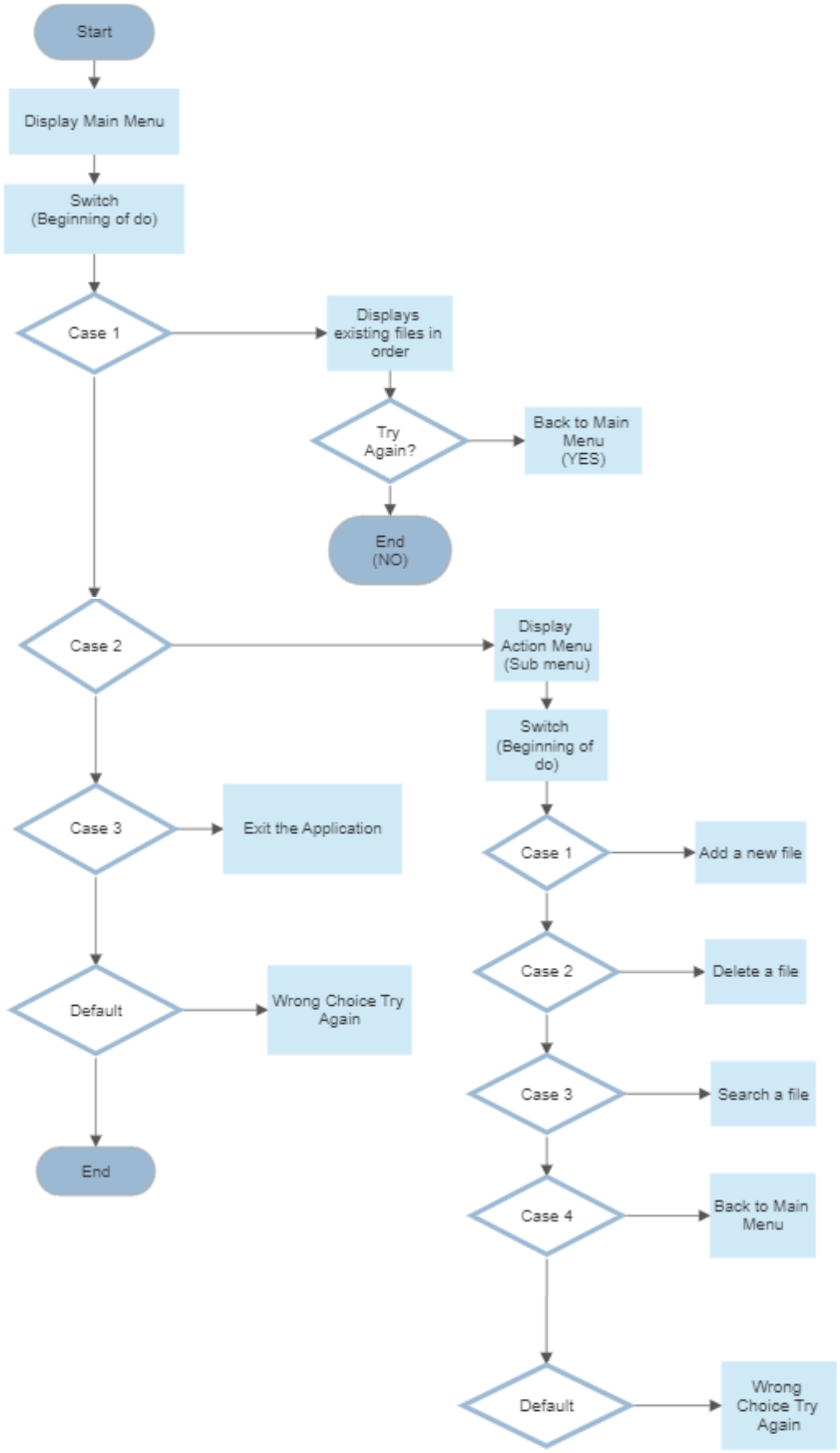
2. **Sprint 2:**
    - The Main Menu is first developed taking all requirements into consideration.
    - Then the sub-menu which is the Action Menu is developed according to the needs.
    - For each of the menus a separate do-while has been used.
    - The following are the options of the Main Menu:
        a. Display existing files in a sorted order (ascending).
        b. Enter the Action Menu.
        c. Exit the application.
    - The following are the options of the Action Menu:
        a. Add a new file to the directory.
        b. Delete a file from the directory.
        c. Search a file in the directory.
    - Tasks Achieved : At the end of Sprint 2 the code for the Main Menu and the Action Menu with all their functionalities.

3. **Sprint 3:**
    - This involves checking for any logical loopholes, fixing bugs and exception handling.
    - For example, the user may enter a file name that already exists in the directory, in such a the application will not accept the input and lets the user know that file already exists. The user is allowed to try again.
    - To choose an option from a menu the user must enter the appropriate number associated to that action.
    - Consider this scenario where a user may enter a different character such as an alphabet or a special character deliberately or by mistake. In such a case the application will not terminate instead it will let the user know that entered value is an invalid input and allows the user to try again.
    - This is done using the exception handling concept.
    - Tasks Achieved : At the end of Sprint 3 all the exceptions were handled and all the logical loopholes were taken care of.

Flow of the application

- The flow is as follows:
    1. Welcome screen with application name and also Main Menu.
    2. The 3 options of the main.
    3. User is asked to enter their choice.
    4. If option 1 is chosen all the existing files of the directory are displayed in a sorted order (ascending).
    5. If option 2 is chosen user is taken to Action Menu.
        a. It displays the various possible actions that can be done by the user.
        b. The user is asked to enter their choice.
        c. If option 1 is chosen then a new file can be added to the existing directory where the user is asked to enter the file name.
        d. If option 2 is chosen a file from the existing directory is deleted where the file to be deleted can be chosen by the user by entering the file's name.
        e. If option 3 is chosen then the file name entered by the user is searched in the existing directory.
        f. If option 4 is chosen the user is taken back to the Main Menu.
    6. If option 3 is chosen then the application is closed.

Flow Chart


Start
Display Main Menu
Switch
(Beginning of do)
Case 1
Displays existing files in order
Try Again?
Back to Main Menu (YES)
End (NO)
Case 2
Display Action Menu (Sub menu)
Switch (Beginning of do)
Case 3
Exit the Application
Case 1
Add a new file
Case 2
Delete a file
Default
Wrong Choice Try Again
Case 3
Search a file
End
Case 4
Back to Main Menu
Default
Wrong Choice Try Again

Core Concepts and Algorithms

- **Packages:**
  - The following are the various packages that have been imported:
    1. **java.util.Scanner -** to accept inputs from the user.
    2. **java.util.Set -** to use the set interface which extends the Collection interface.
    3. **java.util.TreeSet -** the TreeSet class implements the Set interface and uses a tree for storage.
    4. **java.util.InputMismatchException -** used when inputs are taken from the user using the Scanner class and the type of input does not match the variable type.
- **do-while loop:**
  - This application is menu driven i.e. inputs entered by the user drives the direction of flow.
  - To be able to provide the user to perform tasks multiple times without having to start from scratch a loop must be used.
  - The do-while loop is preferred choice because it provides at least one iteration even if the condition is not satisfied.
  - Here, 2 separate do-while loops are used for the two menus of the application i.e. the Main Menu and the Action Menu.
- **switch case:**
  - The user's input drives the direction of flow of the application and the user has multiple options to choose from.
  - An if-else ladder also serves the same purpose as this but will be tedious to implement any changes.
  - The advantage of using a switch case is the presence of default case.
  - Each of the 2 menus of the application have a separate switch case.
  - The user is asked enter their choice and this input is used as the conditional statement for the switch case. This is done separately for both the menus.
- **Sets :**
  - The Set interface extends the Collection interface present in the java.util package.
  - The TreeSet which implements the SortedSet interface is used for storing the names of the files.

- The reasons for using the TreeSet are as follows:
    a. A TreeSet does not allow duplicate entries i.e. here, two files with the same name are not allowed.
    b. The TreeSet by default maintains its elements in their natural order i.e. here, the files would be stores in sorted order (ascending) by default.
    c. It also provides multiple built in methods which eases the process of achieving the end goal.
    d. A few of its methods that have been used here are add(), remove() and contains().

- **Static context:**
    - The TreeSet and the methods used are made static.
    - This is because a static method will be allotted memory only once and not for each instance of the class (object) hence, they are also called class methods.
    - A static method can access only static objects hence the TreeSet is made static too.
    - This saves memory as each method will be allocated memory only once.
    - The only downside to making a method static is that it cannot be overridden but for the scope of this project it does not cause any hinderance.
- **Exception Handling:**
    - The try{} and catch(){} blocks are used to avoid the program to stop running.
    - Consider the scenario where a user is asked to enter a number to choose an option and the user enter an alphabet, special character or even an alphanumeric value. This may be done deliberately or by mistake.
    - Since, the flow of this application is driver by the user's input it is important to make sure correct inputs are accepted.
    - The InputMisMatchException is handled here.

- The algorithm is shown below:

```
//START
// Main menu body
do{
        try{
                input user choice;
                switch(choice){
                case 1:  Display existing files in sorted order.
                         Try Again?
                case 2:  //Action Menu body
                        do{
                                try{
                                input user choice;
                                case 1: Add a new file.
                                        Try again?
                                case 2: Delete a file. break;
                                        Try again?
                                case 3: Search a file. break;
                                        Try again?
                                case 4: Go to Main Menu. break;
                                        Try again?
                                default: Wrong choice sleected.
                                        Try again?
                                }
                                catch(InputMisMatchException e){
                                //body
                                }
                        }
                        while( condition);
                case 3: Exit Application.

                default: Wrong choice selected.
                        Try again?
```

```
                    }
                    catch(InputMisMatchException e){
                        //body
                    }
        }
        while(condition);
        //END
```

Conclusion and USP

- The project LockedMe.com has been split into 3 sprints to meet the end goals efficiently within thin the deadline.
- It uses a menu driven approach where the inputs are given by the user which define the flow of the application.
- All the required specifications have been met using various concepts like exception handling, looping, etc.
- All logical loopholes have been taken care of. For example, user may try to add a file that already exists but a directory cannot contain the two files with the same name.
- In such a scenario the application will not stop running instead it will ask the user to try again.
- The application uses a TreeSet to store data which means additional sorting mechanism is not required as a TreeSet by default will store elements in their natiral order.

GitHub repository link

simpliLearn/Phase-1-Project-VirtualKeyForRepo at main · K-V-Sagar/simpliLearn (github.com)