Lockers PVT LTD– Virtual Key for Repositories

This document contains sections for:

- Sprint planning and Task completion
- Core concepts used in project
- Flow of the Application.
- Demonstrating the product capabilities, appearance, and user interactions.
- Unique Selling Points of the Application
- Conclusions

The code for this project is uploaded in https://github.com/Monishak2/JAVA-PHASE-1-PROJECT-
The project is developed by Monisha K

Sprints planning and Task completion

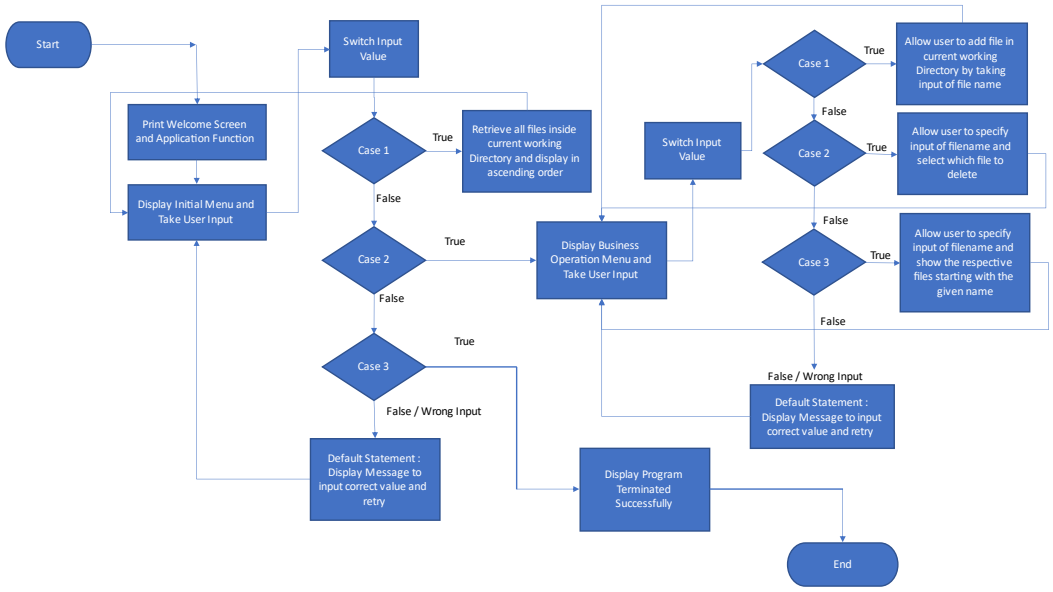
The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

- Creating the flow of the application
- Initializing git repository to track changes as development progresses.
- Writing the Java program to fulfill the requirements of the project.
- Testing the Java program with different kinds of User input
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Core concepts used in project

Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Streams API

Flow of the Application



Demonstrating the product capabilities, appearance, and user interactions

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1. Creating the project in Eclipse
2. Writing a program in Java for the entry point of the application (**Lockers.java**)
3. Writing a program in Java to perform the File operations as specified by user (**Business.java**)
4. Pushing the code to GitHub repository

Step 1: Creating a new project in Eclipse

- Open Eclipse
- Go to File -> New -> Project -> Java Project -> Next.
- Type in any project name and click on “Finish.”
- Select your project and go to File -> New -> Class.
- Enter **Lockers**in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

Step 2: Writing a program in Java for the entry point of the application (**Lockers.java**)

```
package Solution;

import java.util.Scanner;
import java.io.IOException;

public class Lockers {

	public static void main(String[] args) throws IOException {
		int ch=0, choice=0;
		Scanner sc =new Scanner(System.in);

		System.out.println("\t Welcome to LOCKERS PVT LIMITED! ");


		while(true)
		{
		System.out.println("Choose one of the Following options     :");
		System.out.println("1. List  of Current Files");
		System.out.println("2. Business Operations Menu");
	    System.out.println("3. Close the  Application");
			try{
				ch = sc.nextInt();
			}
			catch(Exception e)
{
 System.out.println("NULL EXCEPTION");
}

			switch(ch)
			{
			case 1: //List of files in Ascending Order
				Business.listFiles();
				break;
			case 2:
```

```
        System.out.println("Please choose one of the   following options :");
    System.out.println("1.  To Add a File");
        System.out.println("2.  To  Delete a File");
        System.out.println("3.  To Search for a File");
    System.out.println("4.  Close the Application ");
                                        try{
                                                choice = sc.nextInt();
                                        }
                                        catch(Exception e)
                        {
                         System.out.println("Null Exception occurred");
                        }
                                        switch(choice)
                                        {
                                        case 1:
                                            // file creation
                                                System.out.println("Input the name of a file to be created: ");
                                                String fileCreate = sc.next();

                                                Business.createFile(fileCreate);
                                                break;

                                        case 2:
                                                //file deletion
                                                System.out.print("Input the name of a file to be deleted: ");
                                                String fileDelete = sc.next();
                                                Business.deleteFile(fileDelete);
                                                break;
                                        case 3:
                                           // Searching from file
                                                System.out.println("Input the name of a file to be searched: ");
                                                String fileSearch = sc.next();
                                                Business.searchFile(fileSearch);
                                                break;
                                        case 4:
                                                //Exit from the application
                                                 sc.close();
                                                System.out.println("Close the application ");
                                                System.exit(1);
                                                break;
                                default:

                                                System.out.println("\n Invalid Operations \n");
                                                break;
                                }

                                                break;
                        case 3:


                                                sc.close();
                                                System.out.println("\n Thank you for your visit");
                                                System.exit(2);
                                                break;

                        default:
```

```
   System.out.println("\n Sorry, Invalid Input, Please read the options\n");
                                        break;

                                }

                        }

                }

}
```

Step 1.1: Writing method to display Initial Menu

```
public static void main(String[] args) throws IOException {
                int ch=0, choice=0;
                Scanner sc =new Scanner(System.in);

                System.out.println("\t Welcome to LOCKERS PVT LIMITED! ");


                while(true)
                {
                System.out.println("Choose one of the Following options     :");
                System.out.println("1. List  of Current Files");
                System.out.println("2. Business Operations Menu");
            System.out.println("3. Close the  Application");
                        try{
                                ch = sc.nextInt();
                        }
                        catch(Exception e)
        {
         System.out.println("NULL EXCEPTION");
        }
```

Output:

```
          Welcome to LOCKERS PVT LIMITED!
Choose one of the Following options :
1. List  of Current Files
2. Business Operations Menu
3. Close the  Application
```

Step 3.2: Writing method to display Secondary Menu for File Operations

```
                    switch(ch)
                    {
                    case 1: //List of files in Ascending Order
                            Business.listFiles();
                            break;
                    case 2:

        System.out.println("Please choose one of the   following options :");
System.out.println("1.  To Add a File");
        System.out.println("2.  To  Delete a File");
        System.out.println("3.  To Search for a File");
System.out.println("4.  Close the Application ");
                                        try{
                                                choice = sc.nextInt();
                                        }
                                        catch(Exception e)
                {
                 System.out.println("Null Exception occurred");
                }
```

Output:

```
Please choose one of the following options :
1.   To Add a File
2.   To  Delete a File
3.   To Search for a File
4.   Close the Application
```

Step 3: Writing a program in Java to handle Menu options selected by user (**Business.java**)

- Select your project and go to File -> New -> Class.
- Enter **Business** in class name and click on “Finish.”
- **Business** consists methods for -:

3.1 In Business the List of Fles is displayed in Ascending order.
3.2 In Business to add a file is created.
3.3 In Business to delete a file is created.
3.4 In Business to search a file is also created.

Step 3.1: Writing a method to sort the files in ascending order

```
public class Business{

        //To arrange the files in ascending oder bubble sort is used
        protected static String[] sort_sub(String array[], int size){
                String temp = "";
                for(int i=0; i<size; i++){
                        for(int j=1; j<(size-i); j++){
                                if(array[j-1].compareToIgnoreCase(array[j])>0){
                                        temp = array[j-1];
                                        array[j-1]=array[j];
                                        array[j]=temp;
                                }
                        }
                }
                return array;
        }



        //Listing files
        protected static void listFiles() {

                int fileCount = 0;
           ArrayList<String> filenames = new ArrayList<String>();

File directoryPath = new      File(System.getProperty("user.dir"));
                File[] listOfFiles = directoryPath.listFiles();
                fileCount = listOfFiles.length;


                System.out.println("Files in ascending Oder: ");
                for (int i = 0; i < fileCount; i++) {
                  if (listOfFiles[i].isFile()) {
                    filenames.add(listOfFiles[i].getName());
                  }
                }

                String[] str = new String[filenames.size()];

              for (int i = 0; i < filenames.size(); i++) {
                  str[i] = filenames.get(i);
               }

               String[] sorted_filenames = sort_sub(str, str.length);

                for(String currentFile: sorted_filenames) {
                        System.out.println(currentFile);
                }

        }
```

Output:

```
Choose one of the Following options :
1. List  of Current Files
2. Business Operations Menu
3. Close the  Application
1
Files in ascending Oder:
.classpath
.demo
.hello
.project
```

Step 3.2: Writing method to create a file.

```
//Create a file
        protected static void createFile (String fileToBeCreated) {
                File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeCreated );

                try {
                        if (file.createNewFile() ) {
                                System.out.println("File Created Sucessfully");
                        } else {
                                System.out.println("File already exists");
                        }
                } catch (IOException e) {

                        e.printStackTrace();
                }
        }
```

Output:

```
Please choose one of the following options :
1.   To Add a File
2.   To  Delete a File
3.   To Search for a File
4.   Close the Application
1
Input the name of a file to be created:
zara.txt
File Created Sucessfully
```

Step 3.3: Writing method to delete the files

```
protected static void deleteFile(String fileToBeDeleted) {
        File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeDeleted );

                if(file.exists()) {
                        if ( file.delete() ) {
        System.out.println("File deleted successfully!");
                        }
                } else {
```

```
                System.out.println("File not deleted it is not found");
                        }
                }
```

Output:

```
File Created Sucessfully
Choose one of the Following options :
1. List  of Current Files
2. Business Operations Menu
3. Close the  Application
2
Please choose one of the following options :
1.  To Add a File
2.  To  Delete a File
3.  To Search for a File
4.  Close the Application
2
Input the name of a file to be deleted:
hello
File deleted successfully!
Choose one of the Following options :
1. List  of Current Files
2. Business Operations Menu
3. Close the  Application
```

Step 3.4: Writing method to search the files

```
//Searching of File
                protected static void searchFile(String fileToBeSearched) {
                      File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeSearched );

                          //Check whether file whether file exists or not.
                          //If yes then display associated message
                          if(file.exists()) {
                                  System.out.println(" File found");
                          } else {
                System.out.println(" File is not found,Please re-check");
                          }          PrintWriter pw;
              try {
               pw = new PrintWriter(fileToBeSearched); //may throw exception
                 pw.println("saved");
              }
              // providing the checked exception handler
              catch (FileNotFoundException e) {

                 System.out.println(e);
              }
                  }
```

OUTPUT:

```
Please choose one of the following options :
1.  To Add a File
2.  To  Delete a File
3.  To Search for a File
4.  Close the Application
3
Input the name of a file to be searched:
zara.txt
 File found
```

Step 4: Pushing the code to GitHub repository

- Open your command prompt and navigate to the folder where you have created your files.

  cd <folder path>

- Initialize repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m <commit message>

- Push the files to the folder you initially created using the following command:

  git push -u origin master

Unique Selling Points of the Application

1. The application is designed to keep on running and taking user inputs even after exceptions occur. To terminate the application, appropriate option needs to be selected.

2. The application can take any file/folder name as input. Even if the user wants to create nested folder structure, user can specify the relative path, and the application takes care of creating the required folder structure.

3. User is also provided the option to write content if they want into the newly created file.

4. The application doesn't restrict user to specify the exact filename to search/delete file/folder. They can specify the starting input, and the program searches all files/folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

5. The application also allows user to delete folders which are not empty.

6. The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

7. When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.

    7.1. Ascending order of folders first which have files sorted in them,
    7.2. Ascending order of all files and folders inside the "main" folder.

8. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code. Application has been developed keeping in mind that there should be very less "hardcoding" of data.

Conclusions

Further enhancements to the application can be made which may include:

- Conditions to check if user is allowed to delete the file or add the file at the specific locations.
- Asking user to verify if they really want to delete the selected directory if it's not empty.
- Retrieving files/folders by different criteria like Last Modified, Type, etc.
- Allowing user to append data to the file.