Document Specification

Project 1 (Virtual Key for your Repositories)

By

Mariam Awaji

Contents of this document:

Sprint Planning:

This project is planned to be completed in three sprints as following:

Sprint 1

Sprint Goal: Prepare required documents and framework and code welcome screen of the application

Sprint Duration: 1 Week.

Sprint Backlog include the following stories:

1. I want the application to have main menu and secondary menu
2. I want the main menu to display a welcome screen.
3. I want the main menu to display information about the application and the developer details.
4. I want the main menu to display user interface options and interaction information.
5. I want to see the flow of the application.

Tasks to be completed in the sprint are:

1. Creating specification document - Product's capabilities, appearance, and user interactions.
2. Decide Java concepts being used in the project.
3. Setting up Git and the GitHub account to store and track enhancements of the project.
4. Capturing the flow of the application and creating the flowchart of the application.
5. Writing Java code to create first menu (Main menu) of the application that displays:
    a. Application name and the developer details.
    b. The details of the user interface such as options displaying the user interaction information.

Sprint 2

Sprint Goal: Add features to the application to accept the user input to select one of the options listed in the welcome screen.

Sprint Duration: 1 Week.

Sprint Backlog include the following stories:

1. I want the application to show three options in the main menu:
    a. Retrieve all files in the directory and sort them ascendingly.
    b. Navigate to the second menu.
    c. Exit the application.
2. I want the application to accept user input to perform any of the previous options.

Tasks to be completed in the sprint are:

1. Write Java code to accept user input to select one of the options listed in the Welcome screen.
2. Write Java code to retrieve all files in the directory and sort them ascendingly.
3. Write Java code to enable users to navigate to the second menu (File Option Menu).
4. Write Java code to enable the user to exit the application.

Sprint 3

Sprint Goal: Write java code to handle user input selections to add a file, delete a file, search a file, back to the previous menu and exit the application.

Sprint Duration: 1 Week.

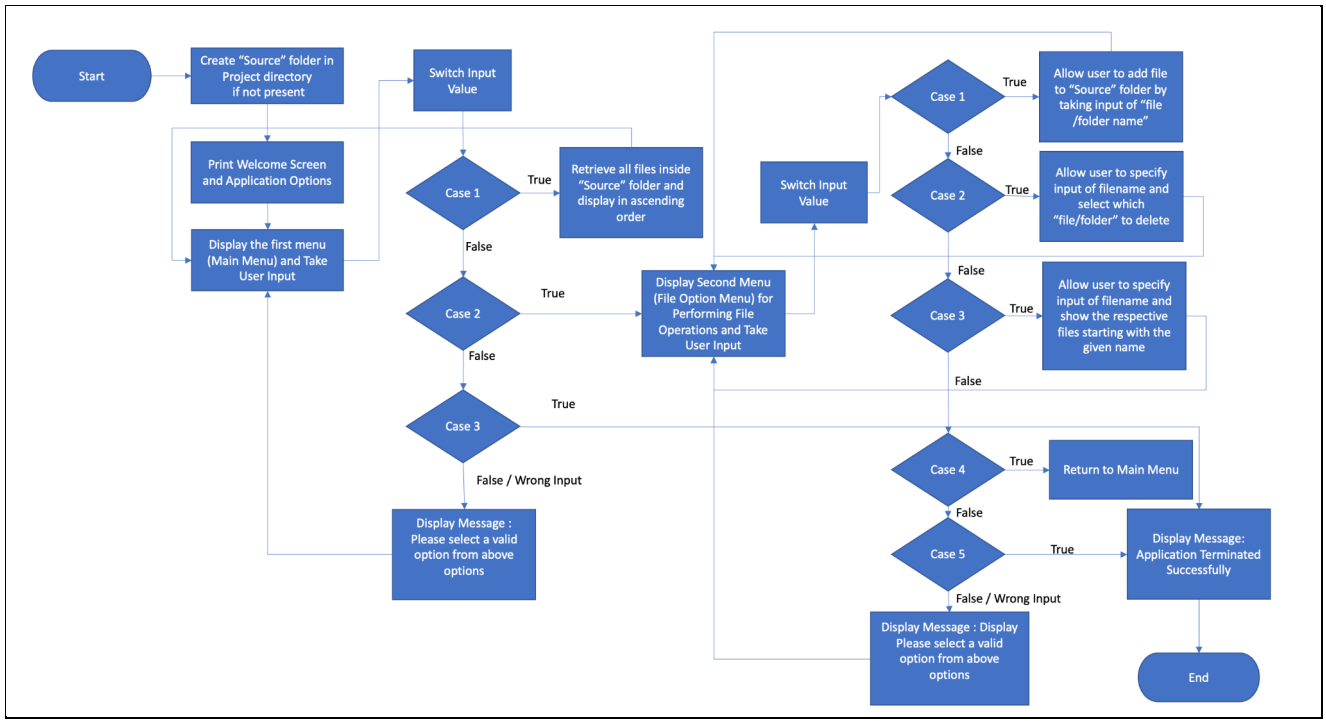
Sprint Backlog include the following stories:

1. I want to add some functionalities to handle files in the secondary menu.
2. I want to enable user to add a file to the existing directory.
3. I want the application to return a message (File already exists) when user is trying to add a file with a name that already exists.
4. I want to enable user to delete a file from the existing directory.
5. I want to enable user to search a file from the existing directory.
6. I want the application to return a (FNF) message when user is trying to delete or search a file that does not exist.
7. I want the application to return messages upon valid and invalid user entry.
8. I want to enable user to return to the main menu.
9. I want to enable user to exit the program at secondary menu.

Tasks to be completed in the sprint are:

1. Write Java code to add a file to the directory.
2. Write Java code to delete a file from the directory.
3. Write Java code to search a file in the directory.
4. Write Java code to enable the user to return to the main menu..
5. Write Java code to enable the user to exit the application from the second menu.
6. Test the program using different user inputs.
7. Push the program files to GitHub.

Flowchart of the application:



Core concepts used to program the application:

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

Link to GetHub Repository:

https://github.com/m-awaji/-Mariam-Awaji-jfs-sda.git

Steps to create the application

1. Planning the flow of the application and creating its flowchart.
2. Planning sprints and tasks to be completed in each sprint as shown in the “Sprint Planning” part in the “Document Specification” file.
3. Prepare project framework by creating a new Java project called “Project 1 Virtual Key for your Repositories” on Eclipse.
4. Creating a package called “Com” inside the project.
5. Creating a class called “LockedMe” and the main method to display Welcome Screen and the details of the application and the developer.

```
public class LockedMe {

                public static void main(String[] args) {

                        System.out.println("Welcome to LockedMe.com\n\n"
                                        + "This application was developed by Mariam Awaji
\n");
                        System.out.println("The application will enable you to :\n"
                                        + "1) Retrieve all file names in the \"Source\" folder.\n"
                                        + "2) Search, add, or delete files in \"Source\" folder."
                                        + " \nNote: When searching or deleting files, please
make sure that you type the correct filename.");
                }
```

Output:

```
Welcome to LockedMe.com

This application was developed by Mariam Awaji

The application will enable you to :
1) Retrieve all file names in the "Source" folder.
2) Search, add, or delete files in "Source" folder.
Note: When searching or deleting files, please make sure that you type the correct filename.
```

6. Write a method called “DisplayMainMenu” in the “LockedMe” class to display Main Menu options for the user.

```
public static void DisplayMainMenu() {
                                    String menu = "\n\nEnter (1, 2 or 3) to do one of the following
and press Enter:\n\n"
                                                + "1 -> Retrieve all files inside \"Source\"
folder\n"
                                                + "2 -> Display menu for File operations\n"
                                                + "3 -> Exit application\n";
                                    System.out.println(menu);

                            }
```

Output:

```
Welcome to LockedMe.com

This application was developed by Mariam Awaji

The application will enable you to :
1) Retrieve all file names in the "Source" folder.
2) Search, add, or delete files in "Source" folder.
Note: When searching or deleting files, please make sure that you type the correct filename.

Enter (1, 2 or 3) to do one of the following and press Enter:

1 -> Retrieve all files inside "Source" folder
2 -> Display menu for File operations
3 -> Exit application
```

7. Write a method called “DisplayFileMenuOptions ”to display Secondary Menu (File Operation Menu).

```
public static void DisplayFileMenuOptions() {
                            String fileMenuOptions = "\n\nEnter (1, 2, 3, 4 or 5) to perform one of
"
                                                + "the following file operations and press Enter\n\n"
                                                + "1) Add a file to \"Source\" folder\n"
                                                + "2) Delete a file from \"Source\" folder\n"
                                                + "3) Search for a file from \"Source\" folder\n"
                                                + "4) Show Previous Menu\n"
                                                + "5) Exit application\n";

                            System.out.println(fileMenuOptions);
```

```
            }
```

Output:

```
Enter (1, 2 or 3) to do one of the following and press Enter:

1 -> Retrieve all files inside "Source" folder
2 -> Display menu for File operations
3 -> Exit application

2


Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application
```

8. Write a method called “WelcomeScreen” to handle user input in the Main Menu

```
public static void WelcomeScreen() {
                        boolean running = true;
                        Scanner sc = new Scanner(System.in);

                        do {
                                try {
                                        DisplayMainMenu();
                                        int input = sc.nextInt();

                                        switch (input) {
                                        case 1:
                                                displayAllFiles("Source");
                                                break;
                                        case 2:
                                                handleFileMenuOptions();
                                                break;
                                        case 3:
```

```
                                        System.out.println("Application terminated
successfully.");
                                        running = false;
                                        sc.close();
                                        System.exit(0);
                                        break;
                                default:
                                        System.out.println("Please select a valid
option from above options.");
                                }
                        } catch (Exception e) {
                                System.out.println(e.getClass().getName());
                                WelcomeScreen();
                        }
                } while (running == true);
        }
```

Output:

```
Enter (1, 2 or 3) to do one of the following and press Enter:

1 -> Retrieve all files inside "Source" folder
2 -> Display menu for File operations
3 -> Exit application

1
Displaying all files with directory structure in ascending order

111.txt
123.txt
abc.txt

Displaying all files in ascending order

111.txt
123.txt
abc.txt
```

9. Write a method called “WelcomeScreen” to handle user input in the secondary menu (File Option Menu)

```
public static void handleFileMenuOptions() {
                boolean running = true;
                Scanner sc = new Scanner(System.in);
                do {
                        try {
                                DisplayFileMenuOptions();
                                CreateSourceFolder("Source");
```

```
                                        int input = sc.nextInt();
                                        switch (input) {
                                        case 1:
                                                System.out.println("Enter the name of the file
to be added to the \"Source\" folder");
                                                String fileToAdd = sc.next();

                                                createFile(fileToAdd, sc);

                                                break;
                                        case 2:
                                                System.out.println("Enter the name of the file
to be deleted from \"Source\" folder");
                                                String fileToDelete = sc.next();
                                                CreateSourceFolder("Source");
                                                List<String> filesToDelete =
displayFileLocations(fileToDelete, "Source");

                                                {String deletionPrompt = "\n(Enter 0 to
confirm deletion or to exit to the prevoius menu)";
                                                System.out.println(deletionPrompt);
                                                int i = sc.nextInt();
                                                if (i != 0) {
                                                        DeleteFiles(filesToDelete.get(i - 1));
                                                } else {
                                                        for (String path : filesToDelete) {
                                                                DeleteFiles(path);
                                                        }
                                                }
                                                }

                                                break;
                                        case 3:
                                                System.out.println("Enter the name of the file
to be searched from \"Source\" folder");
                                                String fileName = sc.next();

                                                CreateSourceFolder("Source");
                                                displayFileLocations(fileName, "Source");


                                                break;
```

```
                                        case 4:
                                                return;
                                        case 5:

                                                System.out.println("Application terminated
successfully.");
                                                running = false;
                                                sc.close();

                                                System.exit(0);
                                        default:
                                                System.out.println("Please select a valid
option from above options.");
                                        }
                                } catch (Exception e) {
                                        System.out.println(e.getClass().getName());
                                        handleFileMenuOptions();
                                }
                        } while (running == true);
                }
```

Output:

```
Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

1
Enter the name of the file to be added to the "Source" folder
222.txt

222.txt created successfully


Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

4


Enter (1, 2 or 3) to do one of the following and press Enter:

1 -> Retrieve all files inside "Source" folder
2 -> Display menu for File operations
3 -> Exit application
```

10. Write a method called “CreateSourceFolder ”to create “Source” folder in the project if it’s not present.

```
public static void CreateSourceFolder(String folderName) {
                                File file = new File(folderName);

                                if (!file.exists()) {
                                        file.mkdirs();
                                }
                        }
```

11. Write a method to display all files in “Source” folder in ascending order.

```
public static void displayAllFiles(String path) {
                                        CreateSourceFolder("Source");

                                    System.out.println("Displaying all files with directory structure in
ascending order\n");

                                        List<String> filesListNames = listFilesInDirectory(path,0, new
ArrayList<String>());

                                        System.out.println("Displaying all files in ascending order\n");
                                        Collections.sort(filesListNames);

                                        filesListNames.stream().forEach(System.out::println);
                                }

                                public static List<String> listFilesInDirectory(String path, int
indentationCount, List<String> fileListNames) {
                                        File dir = new File(path);
                                        File[] files = dir.listFiles();
                                        List<File> filesList = Arrays.asList(files);

                                        Collections.sort(filesList);

                                        if (files != null && files.length > 0) {
                                                for (File file : filesList) {

                                                        System.out.print(" ".repeat(indentationCount
* 2));

                                                        if (file.isDirectory()) {
                                                                System.out.println(file.getName());

                                                                fileListNames.add(file.getName());

listFilesInDirectory(file.getAbsolutePath(), indentationCount + 1, fileListNames);
                                                        } else {
                                                                System.out.println(file.getName());
                                                                fileListNames.add(file.getName());
                                                        }
                                                }
```

```
                } else {
                        System.out.print(" ".repeat(indentationCount * 2));
                        System.out.println("Empty Directory");
                }
                System.out.println();
                return fileListNames;
        }
```

Output:

```
Enter (1, 2 or 3) to do one of the following and press Enter:

1 -> Retrieve all files inside "Source" folder
2 -> Display menu for File operations
3 -> Exit application

1
Displaying all files with directory structure in ascending order

111.txt
123.txt
abc.txt

Displaying all files in ascending order

111.txt
123.txt
abc.txt
```

12. Write a method called “createFile” to create a file from user input.

```
public static void createFile(String fileToAdd, Scanner sc) {
                CreateSourceFolder("Source");
                Path pathToFile = Paths.get("./Source/" + fileToAdd);
                try {
                        Files.createDirectories(pathToFile.getParent());
                        Files.createFile(pathToFile);
                        System.out.println("\n"+fileToAdd + " created
successfully");

                } catch (IOException e) {
                        System.out.println("\nFailed to create file " +
fileToAdd+". File already exists.\n");
                }
```

```
                    }
```

Output:

```
Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

1
Enter the name of the file to be added to the "Source" folder
333.txt

333.txt created successfully


Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

1
Enter the name of the file to be added to the "Source" folder
333.txt

Failed to create file 333.txt. File already exists.
```

13. Write a method called to search for a file in the "Source" folder as specified by user input.

```
public static List<String> displayFileLocations(String fileName, String path) {
                        List<String> fileListNames = new ArrayList<>();
                        searchFileRecursively(path, fileName, fileListNames);

                        if (fileListNames.isEmpty()) {
                                System.out.println("\n\nCouldn't find any file with
given file name \"" + fileName + "\" \n\n");
                        } else {
                                System.out.println("\n\nFound file at below
location(s):");

                                List<String> files = IntStream.range(0,
fileListNames.size())
                                                .mapToObj(index -> (index + 1) + ": " +
```

```
fileListNames.get(index)).collect(Collectors.toList());

                                        files.forEach(System.out::println);
                                }

                                return fileListNames;
                        }

                        public static void searchFileRecursively(String path, String fileName,
List<String> fileListNames) {
                                File dir = new File(path);
                                File[] files = dir.listFiles();
                                List<File> filesList = Arrays.asList(files);

                                if (files != null && files.length > 0) {
                                        for (File file : filesList) {

                                                if (file.getName().startsWith(fileName)) {

fileListNames.add(file.getAbsolutePath());
                                                }

                                                if (file.isDirectory()) {

searchFileRecursively(file.getAbsolutePath(), fileName, fileListNames);
                                                }
                                        }
                                }
                        }
```

Output:

```
Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

3
Enter the name of the file to be searched from "Source" folder
123.txt


Found file at below location(s):
1: /Users/mariam/Desktop/JFS/Git/-Mariam-Awaji-jfs-sda/Phase1/Project1/Source/123.txt
```

14. Write a method called “DeleteFiles” to delete a file from the “Source” folder upon user input.

```
public static void DeleteFiles(String path) {

                                File currFile = new File(path);
                                File[] files = currFile.listFiles();

                                if (files != null && files.length > 0) {
                                        for (File file : files) {

                                                String fileName = file.getName() + " at " +
file.getParent();
                                                if (file.isDirectory()) {
                                                        DeleteFiles(file.getAbsolutePath());
                                                }

                                                if (file.delete()) {
                                                        System.out.println("\n"+fileName + "
deleted successfully");
                                                } else {
                                                        System.out.println("\n"+"Failed to
delete " + fileName);
                                                }
                                        }
                                }
```

```
                                String currFileName = currFile.getName() + " at " +
currFile.getParent();
                                if (currFile.delete()) {
                                        System.out.println("\n"+currFileName + " deleted
successfully");
                                } else {
                                        System.out.println("\n"+"Failed to delete " +
currFileName);
                                }


                }
```

Output:

```
Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application


2
Enter the name of the file to be deleted from "Source" folder
111


Found file at below location(s):
1: /Users/mariam/Desktop/JFS/Git/-Mariam-Awaji-jfs-sda/Phase1/Project1/Source/111.txt

(Enter 0 to confirm deletion or to exit to the prevoius menu)
0

111.txt at /Users/mariam/Desktop/JFS/Git/-Mariam-Awaji-jfs-sda/Phase1/Project1/Source deleted successfully


Enter (1, 2, 3, 4 or 5) to perform one of the following file operations and press Enter

1) Add a file to "Source" folder
2) Delete a file from "Source" folder
3) Search for a file from "Source" folder
4) Show Previous Menu
5) Exit application

2
Enter the name of the file to be deleted from "Source" folder
dfg


Couldn't find any file with given file name "dfg"
```

Conclusion in enhancing the application:

1. Add Java code to clear the console to enable user to follow new entered options without scrolling down.
2. Add a condition to check the validity of the format of the files being added from user input.
3. Enable user to add contents to files.