Phase1 – 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

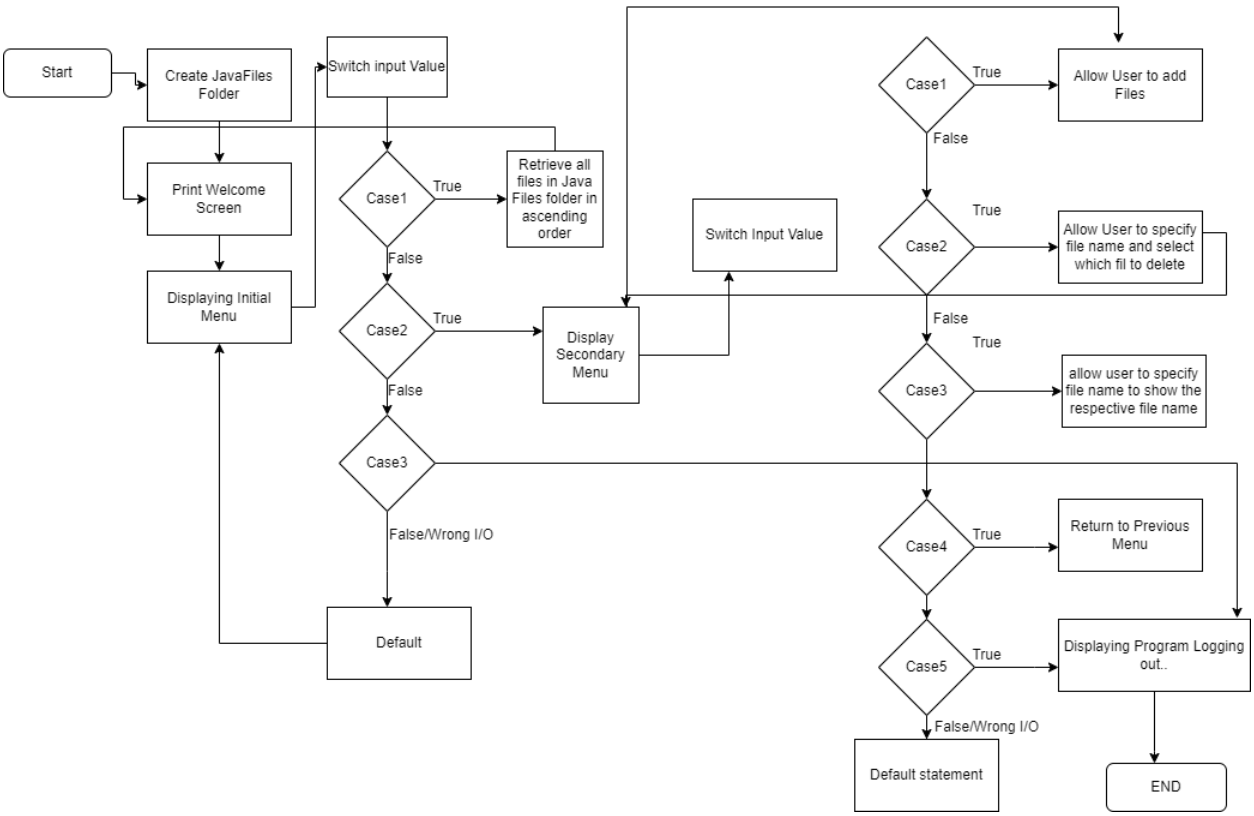
The code for this project is hosted at
https://github.com/NidamanuriRahulKumar/Phase1CoreJavaMainProject.git
The project is developed by **Nidamanuri Rahul Kumar**.

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.

Flow Chart:



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 (**MainDef.java**)
3. Writing a program in Java to display Menu options available for the user (**MenuOptions.java**)
4. Writing a program in Java to handle Menu options selected by user (**Controls.java**)
5. Writing a program in Java to perform the File operations as specified by user (**FileOperations.java**)
6. 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 **MainDef** 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 (**MainDef.java**)

```
package SimpliLearn.Phase1;

public class MainDef{

public static void main(String[] args) {

FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");

MenuOptions.printWelcomeScreen("javaProject", "Rahul Kumar N");

Controls.handleWelcomeScreenInput();

}

}
```

Step 3: Writing a program in Java to display Menu options available for the user (**MenuOptions.java**)

- Select your project and go to File -> New -> Class.
- Enter **MenuOptions** in class name and click on “Finish.”

```
package SimpliLearn.Phase1;
public class MenuOptions {
public static void printWelcomeScreen(String appName, String developerName) {
}
public static void displayMenu() {
String menu = "\n\n>>>>----Select any option number from below and press Enter----<<<<\n\n"
                + "1) Retrieve all files inside \"JavaFiles\" folder\n"
                + "2) Display menu for File operations\n"
                + "3) Exit program\n";
```

```
System.out.println(menu);
}
public static void displayFileMenuOptions() {
String fileMenu = "\n\n>>>>---- Select an Operation ----<<<<\n\n"
                + "1) Add a file to \"JavaFiles\" folder\n"
                + "2) Delete a file from \"JavaFiles\" folder\n"
                + "3) Search file from \"JavaFiles\" folder\n"
                + "4) Show Previous Menu\n"
                + "5) Exit program\n";
System.out.println(fileMenu);
}
}
```

Step 4: Writing a program in Java to handle Menu options selected by user (**Controls.java**)

- Select your project and go to File -> New -> Class.
- Enter **Controls** in class name and click on “Finish.”

```
package SimpliLearn.Phase1;
import java.util.List;
import java.util.Scanner;
public class Controls {
        public static void handleWelcomeScreenInput() {
                boolean running = true;
                Scanner sc = new Scanner(System.in);
                do {
                        try {
                                MenuOptions.displayMenu();
                                int input = sc.nextInt();
                                switch (input) {
case 1:
FileOperations.displayAllFiles("JavaFiles");
break;
case 2:
```

```
Controls.handleFileMenuOptions();
break;
case 3:
System.out.println("Program exited successfully.");
running = false;
sc.close();
System.exit(0);
break;
default:
System.out.println("Select a valid option");
                                        }
} catch (Exception e) {
System.out.println(e.getClass().getName());
handleWelcomeScreenInput();
}
} while (running == true);
}
public static void handleFileMenuOptions() {
boolean running = true;
Scanner sc = new Scanner(System.in);
do {
try {
MenuOptions.displayFileMenuOptions();
        FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");
        int input = sc.nextInt();
switch (input) {
case 1:   // File Add
        System.out.println("Enter the name of the file to add\"JavaFiles\" folder");
                                        String fileToAdd = sc.next();
                                        FileOperations.createFile(fileToAdd, sc);
                                        break;
case 2:   // File/Folder delete
```

```
System.out.println("Enter the name of the file to delete \"JavaFiles\" folder");
String fileToDelete = sc.next();
FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");
List<String> filesToDelete = FileOperations.displayFileLocations(fileToDelete, "JavaFiles");
String deletionPrompt = "\nSelect file to delete" + "\n(Enter 0 if you want to delete all elements)";
System.out.println(deletionPrompt);
int idx = sc.nextInt();
if (idx != 0) {
FileOperations.deleteFileRecursively(filesToDelete.get(idx - 1));
} else {
for (String path : filesToDelete) {
FileOperations.deleteFileRecursively(path);
}
}
break;
case 3:
System.out.println("Enter file name to be searched from \"JavaFiles\" folder");
                                                    String fileName = sc.next();
FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");
FileOperations.displayFileLocations(fileName, "JavaFiles");
break;
case 4:
            return;
case 5:
System.out.println(" Logging Out..successfully.");
            running = false;
            sc.close();
            System.exit(0);
default:
            System.out.println("select a valid option ");
            }
            } catch (Exception e) {
```

```
            System.out.println(e.getClass().getName());
            handleFileMenuOptions();
                        }
                } while (running == true);
            }
}
```

Step 5: Writing a program in Java to perform the File operations as specified by user (**FileOperations.java**)

- Select your project and go to File -> New -> Class.
- Enter **FileOperations** in class name and click on “Finish.”

```
package SimpliLearn.Phase1;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
public class FileOperations {
        public static void createJavaFilesFolderIfNotPresent(String folderName) {
                File file = new File(folderName);

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

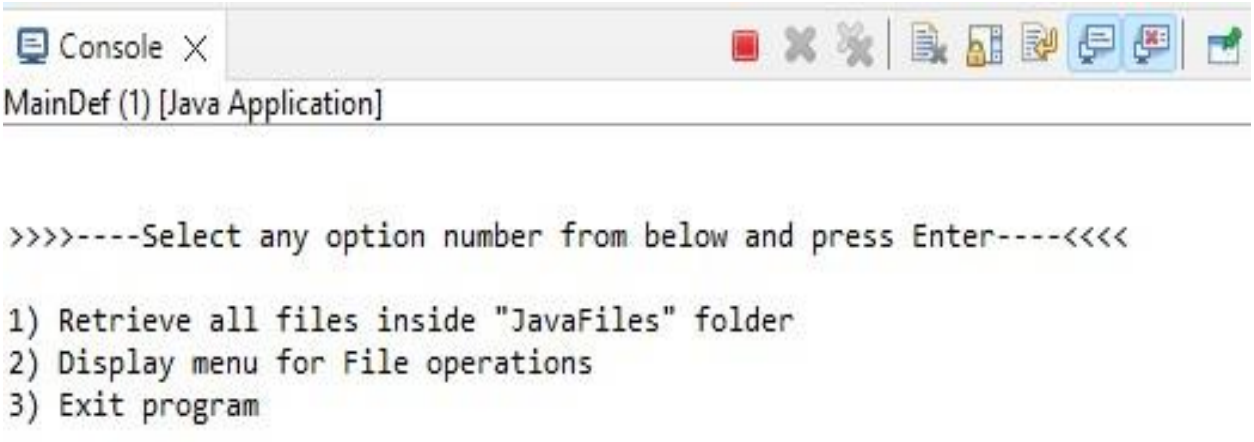
```
                    }
public static void displayAllFiles(String path) {
FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");
System.out.println("All files in directory path in ascending order\n");
List<String> filesListNames = FileOperations.listFilesInDirectory(path, 0, new ArrayList<String>());
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());
        // Recursively indent and display the files
        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("-->Directory is empty");
        }
        System.out.println();
```

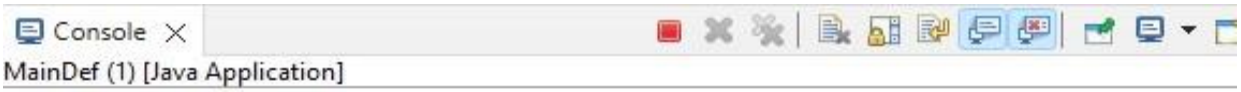
```
return fileListNames;
}
public static void createFile(String fileToAdd, Scanner sc) {
FileOperations.createJavaFilesFolderIfNotPresent("JavaFiles");
Path pathToFile = Paths.get("./JavaFiles/" + fileToAdd);
try {
Files.createDirectories(pathToFile.getParent());
Files.createFile(pathToFile);
System.out.println(fileToAdd + " created successfully");
System.out.println("Would you like to add some content to the file? (Y/N)");
String choice = sc.next().toLowerCase();
sc.nextLine();
if (choice.equals("y")) {
System.out.println("\n\nInput content and press enter\n");
String content = sc.nextLine();
Files.write(pathToFile, content.getBytes());
System.out.println("\nContent written to file " + fileToAdd);
System.out.println("Content can be read using Notepad or Notepad++");
}
} catch (IOException e) {
System.out.println("Failed to create file " + fileToAdd);
System.out.println(e.getClass().getName());
}
}
public static List<String> displayFileLocations(String fileName, String path) {
List<String> fileListNames = new ArrayList<>();
FileOperations.searchFileRecursively(path, fileName, fileListNames);
if (fileListNames.isEmpty()) {
System.out.println("\n\n>>>>----Couldn'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);
}
}
}
}
public static void deleteFileRecursively(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()) {
deleteFileRecursively(file.getAbsolutePath());
}
if (file.delete()) {
System.out.println(fileName + " deleted successfully");
```

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

Output's:






```
>>>>----Select any option number from below and press Enter----<<<<

1) Retrieve all files inside "JavaFiles" folder
2) Display menu for File operations
3) Exit program

1
All files in directory path in ascending order

--> Rahul1

Displaying all files in ascending order

Rahul1


>>>>----Select any option number from below and press Enter----<<<<

1) Retrieve all files inside "JavaFiles" folder
2) Display menu for File operations
3) Exit program



>>>>----Select any option number from below and press Enter----<<<<

1) Retrieve all files inside "JavaFiles" folder
2) Display menu for File operations
3) Exit program

2


>>>>---- Select an Operation ----<<<<

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

```
>>>>----Select any option number from below and press Enter----<<<<

1) Retrieve all files inside "JavaFiles" folder
2) Display menu for File operations
3) Exit program

2


>>>>---- Select an Operation ----<<<<

1) Add a file to "JavaFiles" folder
2) Delete a file from "JavaFiles" folder
3) Search file from "JavaFiles" folder
4) Show Previous Menu
5) Exit program

1
Enter the name of the file to add"JavaFiles" folder
Ram
Ram created successfully
Would you like to add some content to the file? (Y/N)
y


Input content and press enter

King of Mirzapur

Content written to file Ram
Content can be read using Notepad or Notepad++
```

```
>>>>---- Select an Operation ----<<<<

1) Add a file to "JavaFiles" folder
2) Delete a file from "JavaFiles" folder
3) Search file from "JavaFiles" folder
4) Show Previous Menu
5) Exit program

2
Enter the name of the file to delete "JavaFiles" folder
Ram


Found file at below location(s):
1: C:\Users\nrkum\git\Phase1CoreJavaMainProject\JavaMain_Project_Slearn\JavaFiles\Ram

Select file to delete
(Enter 0 if you want to delete all elements)
0
Ram at C:\Users\nrkum\git\Phase1CoreJavaMainProject\JavaMain_Project_Slearn\JavaFiles deleted successfully
```

```
>>>>---- Select an Operation ----<<<<

1) Add a file to "JavaFiles" folder
2) Delete a file from "JavaFiles" folder
3) Search file from "JavaFiles" folder
4) Show Previous Menu
5) Exit program

3
Enter file name to be searched from "JavaFiles" folder
Rahul


Found file at below location(s):
1: C:\Users\nrkum\git\Phase1CoreJavaMainProject\JavaMain_Project_Slearn\JavaFiles\Rahul1




>>>>---- Select an Operation ----<<<<

1) Add a file to "JavaFiles" folder
2) Delete a file from "JavaFiles" folder
3) Search file from "JavaFiles" folder
4) Show Previous Menu
5) Exit program

4


>>>>----Select any option number from below and press Enter----<<<<

1) Retrieve all files inside "JavaFiles" folder
2) Display menu for File operations
3) Exit program

3
Program exited successfully.
```

Step 6: 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

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.