Virtual Key for Repositories

Abhijeet Bhatnagar

This document contains the following sections:

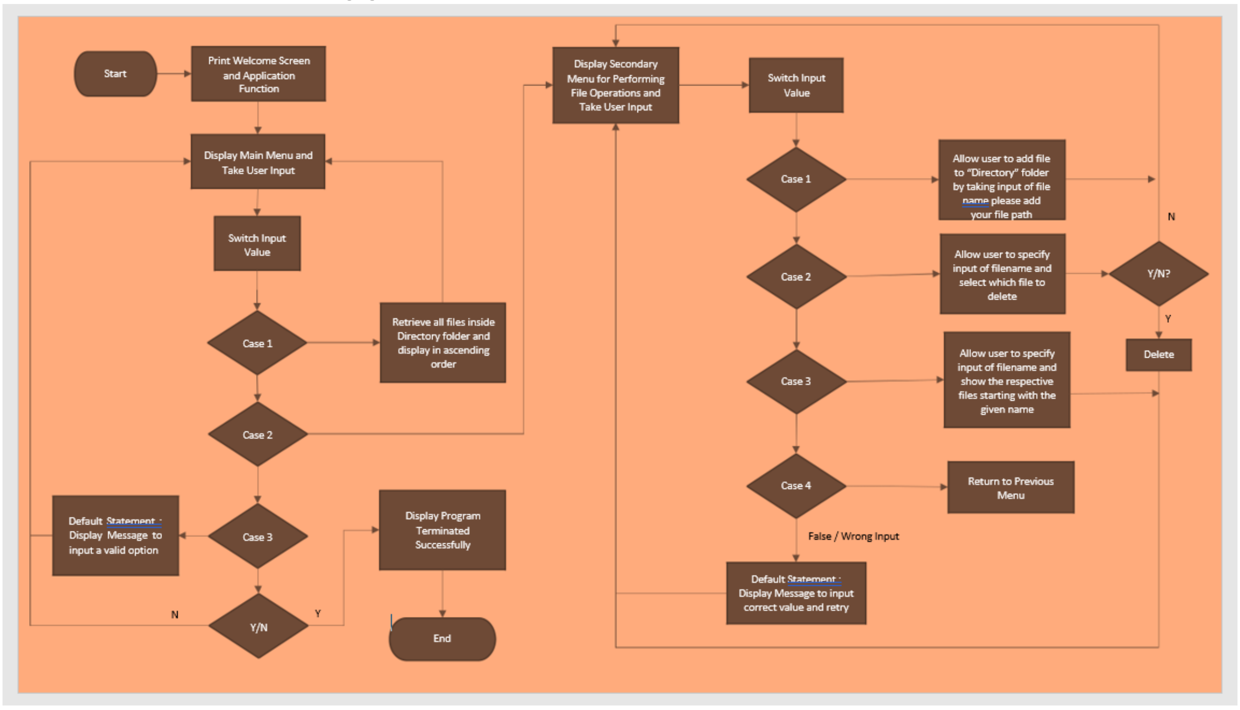
Sprint Planning and Task Completion

The task assumed to be completed are:

Core Concepts Used in the Project

- Collection Framework
- Flow control
- Recursion
- File Handling
- Exception Handling
- Sorting Algorithm Merge Sort time Complexity nlog(n)

Flow Chart of the Application


Start
Print Welcome Screen and Application Function
Display Main Menu and Take User Input
Switch Input Value
Case 1
Retrieve all files inside Directory folder and display in ascending order
Case 2
Case 3
Default Statement : Display Message to input a valid option
Display Program Terminated Successfully
Y/N
N
Y
End
Display Secondary Menu for Performing File Operations and Take User Input
Switch Input Value
Case 1
Allow user to add file to "Directory" folder by taking input of file name please add your file path
Case 2
Allow user to specify input of filename and select which file to delete
Y/N?
N
Y
Delete
Case 3
Allow user to specify input of filename and show the respective files starting with the given name
Case 4
Return to Previous Menu
False / Wrong Input
Default Statement : Display Message to input correct value and retry

Product Capabilities

Display Screen contain these options.

1. Show files – shows files from the directory in ascending order (Sorted).
2. Show files menu – shows the following file operations:
   2.1. Add a file – adds files to the directory.
   2.2. Delete a file – deletes files from the directory.
   2.3. Search for a file – searches for file in the directory.
   2.4. Return to main menu -Returns to main menu.
3. Exit the application.

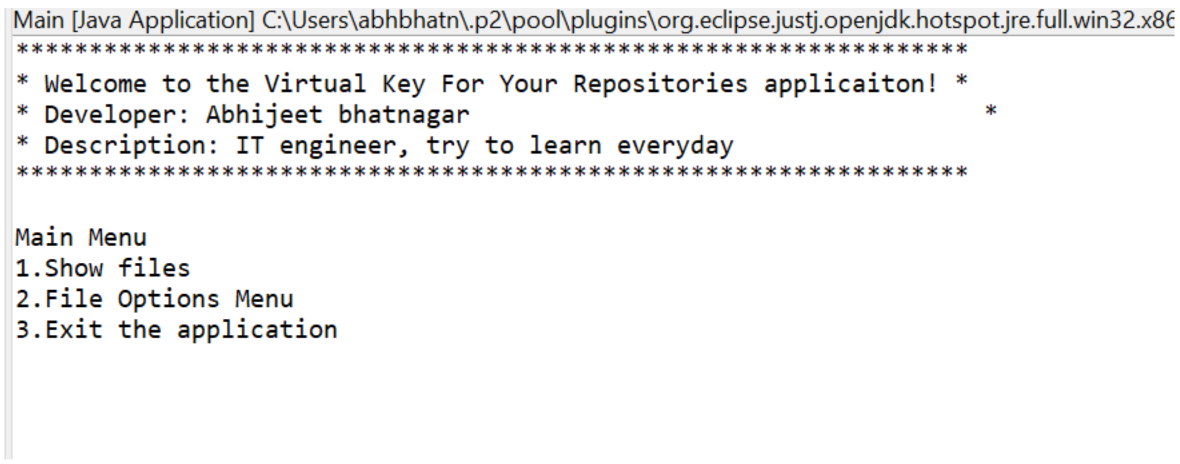
Source code for each of the function with output is shown below:

Welcome Screen of the Application:

```java
package VirtualKeyForRepositories;

public class Main {
    public static void main(String[] args) {
        WelcomeScreen welcome = new WelcomeScreen();
        welcome.Intro();
        welcome.MainMenu();
    }

}
```



```
Main [Java Application] C:\Users\abhbhatn\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
*****************************************************************
* Welcome to the Virtual Key For Your Repositories applicaiton! *
* Developer: Abhijeet bhatnagar                                   *
* Description: IT engineer, try to learn everyday
*****************************************************************


Main Menu
1.Show files
2.File Options Menu
3.Exit the application
```

Show files:

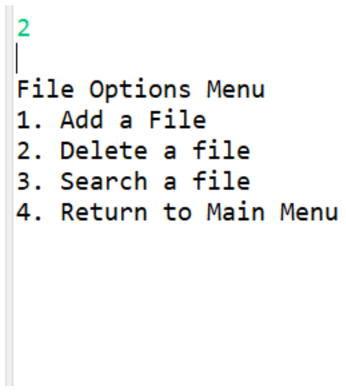
```java
System.out.println("\nMain Menu");
       System.out.println("1.Show files\n2.File Options Menu\n3.Exit the 
application\n");
       boolean running = true;
       Scanner option = new Scanner(System.in);
       do {
           try {
               int input = option.nextInt();
               switch (input) {
                   case 1:
                       this.ShowFiles();
                       this.MainMenu();
                       break;
                   case 2: //pending
                       FileOptionsMenu FileMenu = new FileOptionsMenu();
                       FileMenu.show();
                       break;
                   case 3:
                       System.out.println("Quit the application....");
                       System.out.println("Are you sure? Type 'Y' for yes and 'N' 
for no");

                       Scanner sure = new Scanner(System.in);
                       String s = sure.nextLine();
                       if(s.equals("y") || s.equals("Y")) {
                           System.out.println("Applicaiton terminated");
                       running = false;
                       System.exit(0);}
                       else {
                           MainMenu();
                       }
                   default:
                       System.out.println("Invalid option,please check the 
Input");
                       break;
               }
           } catch (Exception e) {
               System.out.println(e.getClass().getName() + ": Please enter a valid 
option");
               MainMenu();

           }
       }
       while (running == true) ;
   }

    private void ShowFiles() {
        Directory obj = new Directory(); //Retrieve files from directory
        obj.getFiles();
    }
    }
```

Output:



```
2

File Options Menu
1. Add a File
2. Delete a file
3. Search a file
4. Return to Main Menu
```

File Options Menu: (Add, delete, search, return)

```
private Directory dir = new Directory();
    public void MenuHandler() {
        System.out.println("""
                1. Add a File
                2. Delete a file
                3. Search a file
                4. Return to Main Menu""");
        boolean running = true;
        Scanner option = new Scanner(System.in);
        do {
            try {
                int input = option.nextInt();
                switch (input) {
                    case 1:  //Adds file to directory
```

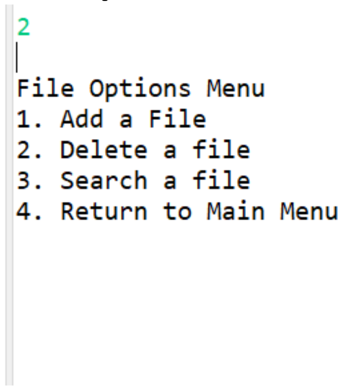
```java
                        this.AddFile();
                        this.show();
                        break;
                    case 2: // Deletes file from directory
                        this.DeleteFile();
                        this.show();
                        break;
                    case 3: // search for file in directory
                        this.SearchFile();
                        this.show();
                        break;
                    case 4: // return to main menu
                        WelcomeScreen obj = new WelcomeScreen();
                        obj.MainMenu();
                        break;
                    default:
                        System.out.println("Invalid Option");
                        break;
                }
            } catch (InputMismatchException e) {
                System.out.println(e + ": Please select a valid option");
                this.MenuHandler();
            }
        }
        while (running);

    }
 public void show() {
        System.out.println("\nFile Options Menu");
        MenuHandler();
        }
    private String getInputSting() {
        Scanner in = new Scanner(System.in);
        return (in.nextLine());
    }
    private void AddFile() {
        System.out.println("Enter file name:");
        String filename = this.getInputSting();
        System.out.println("Adding file:" + filename);
        try {
            Path path = FileSystems.getDefault().getPath(Directory.pathName +
filename).toAbsolutePath();
            File file = new File(dir.getPathName() + filename);
            if (file.createNewFile()) {
                System.out.println("File added: " + file.getName());
                dir.ListFiles().add(file);
            } else {
                System.out.println("File already Exists");
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }
    private void DeleteFile() {
        Directory obj = new Directory();
        obj.getFiles();
        System.out.println("Enter filename to delete:");
        String filename = this.getInputSting();
        System.out.println("Deleting file: " + filename + "\nAre you sure?
(Y/N)");
        String sure = this.getInputSting();
        if(sure.equals("y") || sure.equals("Y")) {  //asks sure?
            Path path = FileSystems.getDefault().getPath(Directory.pathName +
filename).toAbsolutePath();
            File file = path.toFile();
            if (file.delete()) {
                System.out.println("Deleted file: " + file.getName());
                dir.ListFiles().remove(file);
            } else {
                System.out.println("Failed, file not found");
            }
        }
        else
        {
            return;
        }
    }
    private void SearchFile() {
        boolean found = false;
        System.out.println("Enter file name:");
        String fileName = this.getInputSting();
        System.out.println("Searching for file: " + fileName);

        ArrayList<File> files = dir.ListFiles();
        for (File file : files) {
            if (file.getName().equals(fileName)) {
                System.out.println("Found " + fileName);
                found = true;
            }
        }
        if (!found) {
            System.out.println("File not found");
        }
    }
}
```

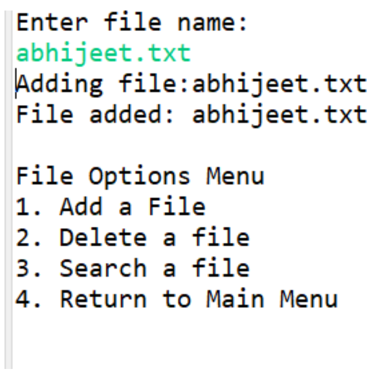
Output:

FileMenu output:



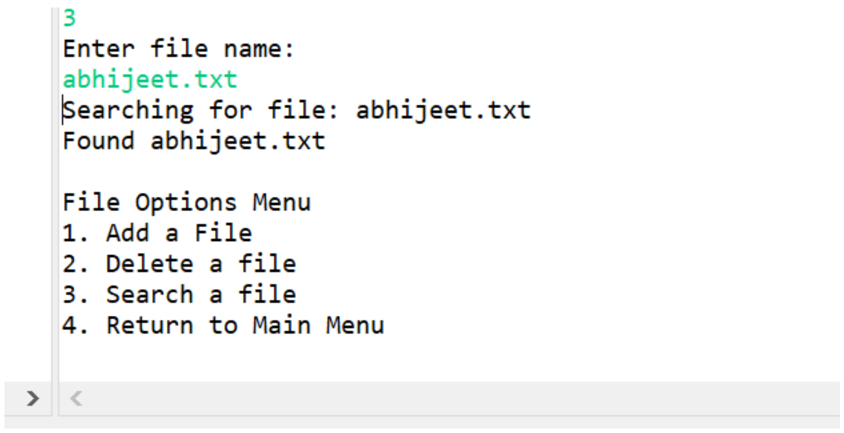
```
2

File Options Menu
1. Add a File
2. Delete a file
3. Search a file
4. Return to Main Menu
```



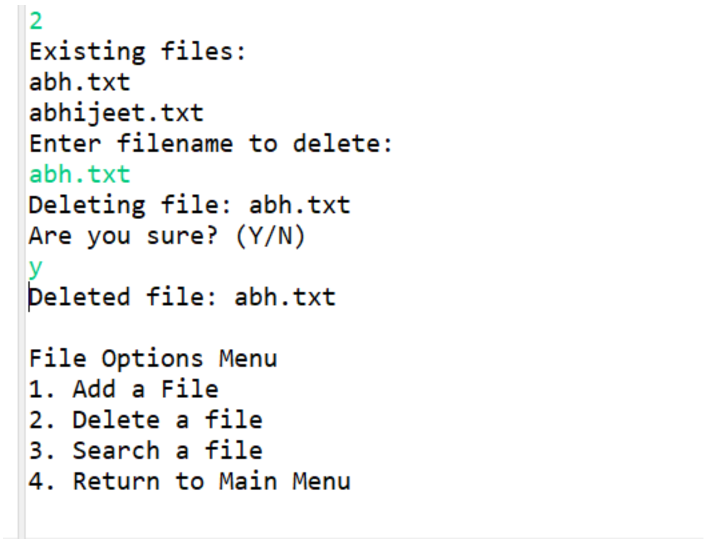
```
Enter file name:
abhijeet.txt
Adding file:abhijeet.txt
File added: abhijeet.txt

File Options Menu
1. Add a File
2. Delete a file
3. Search a file
4. Return to Main Menu
```



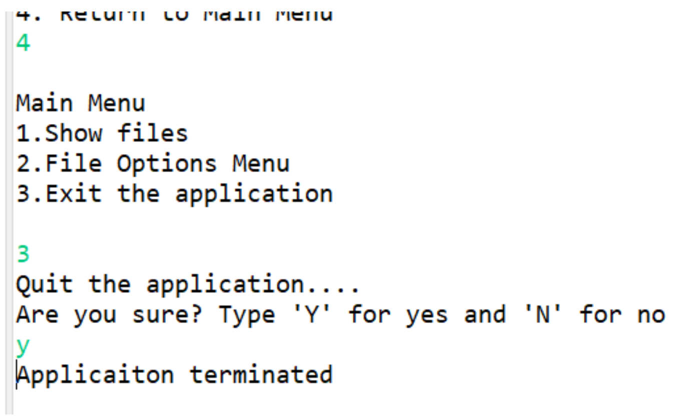
```
3
Enter file name:
abhijeet.txt
Searching for file: abhijeet.txt
Found abhijeet.txt

File Options Menu
1. Add a File
2. Delete a file
3. Search a file
4. Return to Main Menu
```



```
2
Existing files:
abh.txt
abhijeet.txt
Enter filename to delete:
abh.txt
Deleting file: abh.txt
Are you sure? (Y/N)
y
Deleted file: abh.txt

File Options Menu
1. Add a File
2. Delete a file
3. Search a file
4. Return to Main Menu
```



```
4

Main Menu
1.Show files
2.File Options Menu
3.Exit the application

3
Quit the application....
Are you sure? Type 'Y' for yes and 'N' for no
y
Applicaiton terminated
```

Unique Selling Points of the Application:

- The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it
- The second option should return the details of the user interface such as options displaying the following:
  - Add a file to the existing directory list
    - You can ignore the case sensitivity of the file names
  - Delete a user specified file from the existing directory list
    - You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
    - Return a message if FNF (File not found)
  - Search a user specified file from the main directory
    - You can add the case sensitivity on the file name to retrieve the correct file
    - Display the result upon successful operation
    - Display the result upon unsuccessful operation
  - Option to navigate back to the main context

- There should be a third option to close the application

Conclusion:

Further improvements to the application can be done According to the requirement