Virtual Key for Your Repositories

Project details:

Console based java application that can list(in ascending order) create, delete, and search for files. It does not crash and provides options to exit.

Developer Details:

1. Name: Roshan Anil Shambharkar.
2. Organization: RTMNU
3. Learning Institute: Simplilearn
4. Course: Java Full Stack Developer
5. Project 1

Sprints Planned:

Sprint 1:
1. Decide Flow of the program
2. Design Classes and Methods
3. Design Exceptions
4. Create Project Structure
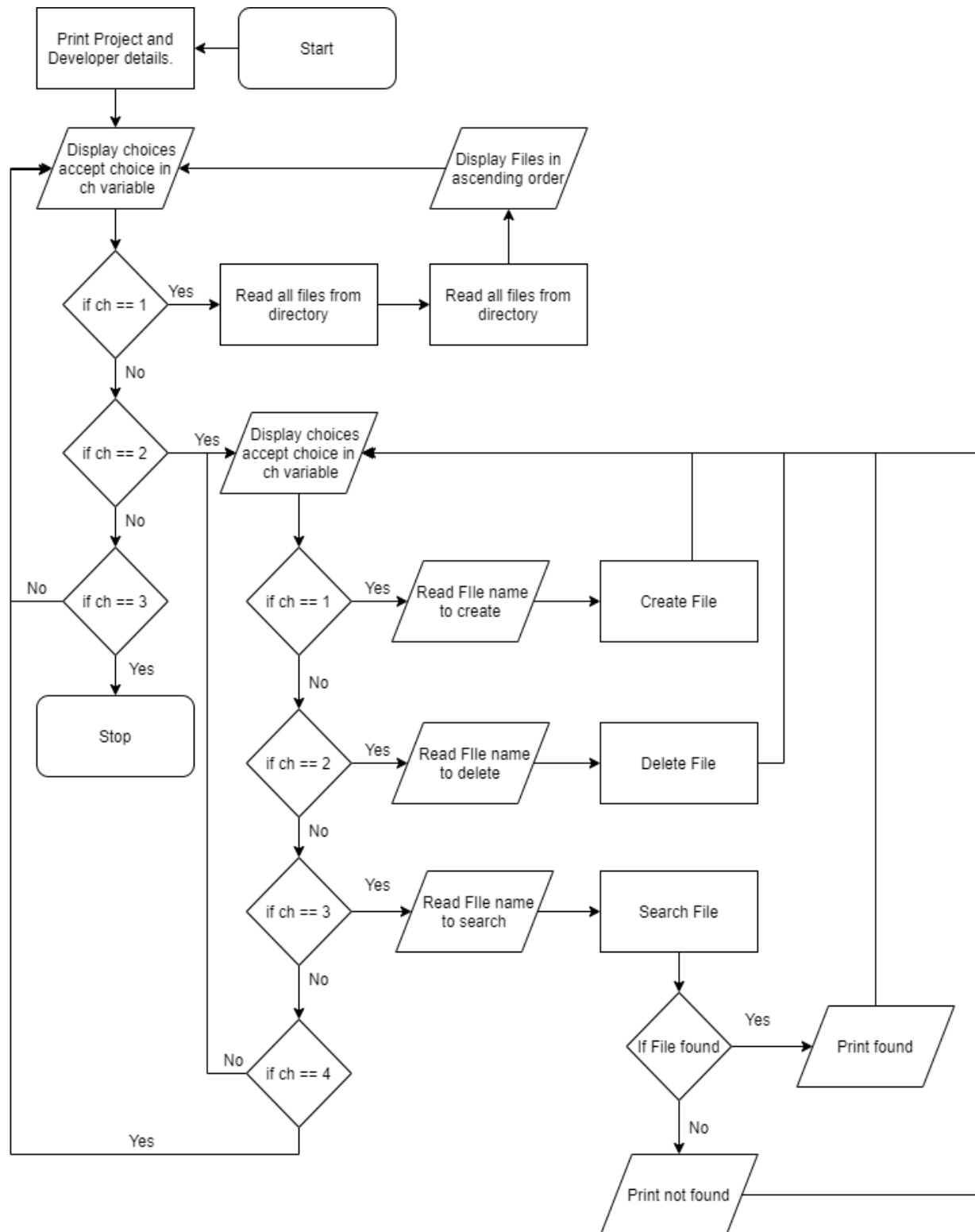5. Code Data tier
6. Code Business tier interface

Sprint 2:
1. Code Business Tier Methods
   a. List File Method
   b. Create File Method
   c. Search File Method
   d. Delete File Method
2. Code the level one menu
3. Code the level two menu
4. Call the Business Tier Methods in the Main File
5. Use String Formatting to display the output in a standard manner

Algorithm:

1) Start
2) Print Options to list, manage files and exit
3) If choice is 1
   a) Read all files in the directory
   b) Sort files in ascending order using Collections.sort()
   c) Print all the files
   d) Goto 3
4) If the choice is 2
   a) Print Options to add, search and delete files
   b) If the choice is 1
      i) Accept file name from the user
      ii) Create a new file with that name
      iii) Goto b
   c) If the choice is 2
      i) Accept file name from the user
      ii) Delete the file with that name
      iii) Goto b
   d) If the choice is 3
      i) Accept file name from the user
      ii) Search for file with that name
      iii) If file found print found
      iv) Else print Not found
      v) Goto b
   e) If the choice is 4 goto a
   f) Goto 2
5) If the choice is 3 goto 7
6) Goto 2
7) Stop

Flowchart of the System:

Start → Print Project and Developer details.

Print Project and Developer details. → Display choices accept choice in ch variable

Display Files in ascending order → Display choices accept choice in ch variable

Display choices accept choice in ch variable → if ch == 1

if ch == 1 —Yes→ Read all files from directory → Read all files from directory → Display Files in ascending order

if ch == 1 —No→ if ch == 2

if ch == 2 —Yes→ Display choices accept choice in ch variable

if ch == 2 —No→ if ch == 3

if ch == 3 —No→ (loop back to Display choices accept choice in ch variable)

if ch == 3 —Yes→ Stop

Display choices accept choice in ch variable → if ch == 1

if ch == 1 —Yes→ Read FIle name to create → Create File

if ch == 1 —No→ if ch == 2

if ch == 2 —Yes→ Read FIle name to delete → Delete File

if ch == 2 —No→ if ch == 3

if ch == 3 —Yes→ Read FIle name to search → Search File

if ch == 3 —No→ if ch == 4

if ch == 4 —No→ (back to Display choices)

if ch == 4 —Yes→ (back to main Display choices)

Search File → If File found

If File found —Yes→ Print found

If File found —No→ Print not found

Core Concepts Implemented:

1. Encapsulation: FileDetail class in the data layer encapsulates all the file data such as file name, file parent, file length, file path by making them private and providing public methods to access them
2. Abstraction: The class FileManagerBOImpl abstracts all the implementation details from the presentation layer class (VitualKeyMain) and the interface (FileManagerBO)
3. Inheritance: The AlphaNumericOnly and the FileNameNotFoundException.java class extend/ inherit from the RuntimeException class to provide user-defined exception messages if the file name provided by the user is not a valid name and if the file name provided to delete by the user is not available.
4. Polymorphism: The FileDetail class overrides the toString method to provide custom string output. It also overrides the compareTo method of the comparator method to provide the sorting capability. It also has overloaded constructors.
5. java.util.FIle was used to create delete, search and list files from the directory
6. Linked List was used to store the list of files in the directory and Collections.sort was used to sort them.

Link to the Repository: https://github.com/roshan100kar/Virtual-Key-Repositories

Conclusion:

The file manager was created using the three-tier architecture. Special attention was paid to implement the pillars of the OOPM. Best practices were followed as and when possible. Rigorous testing was done to ensure that there are no spontaneous exits and all exceptions are handled. Some Exceptions are handled using Custom exception classes. The throws and throw keywords were used to handle exceptions. The Comparable interface was used to provide the compareTo method to help in sorting

Unique Selling Points:

● The application displays filename by aligning widths of columns
● The application does not exit without giving proper messages
● The list of filenames is displayed in ascending order
● The file to be deleted is checked if it exists so that IOException is not raised

Enhancements:

● Providing a GUI.
● Classifying files by their extension.
● Tracking changes in files.