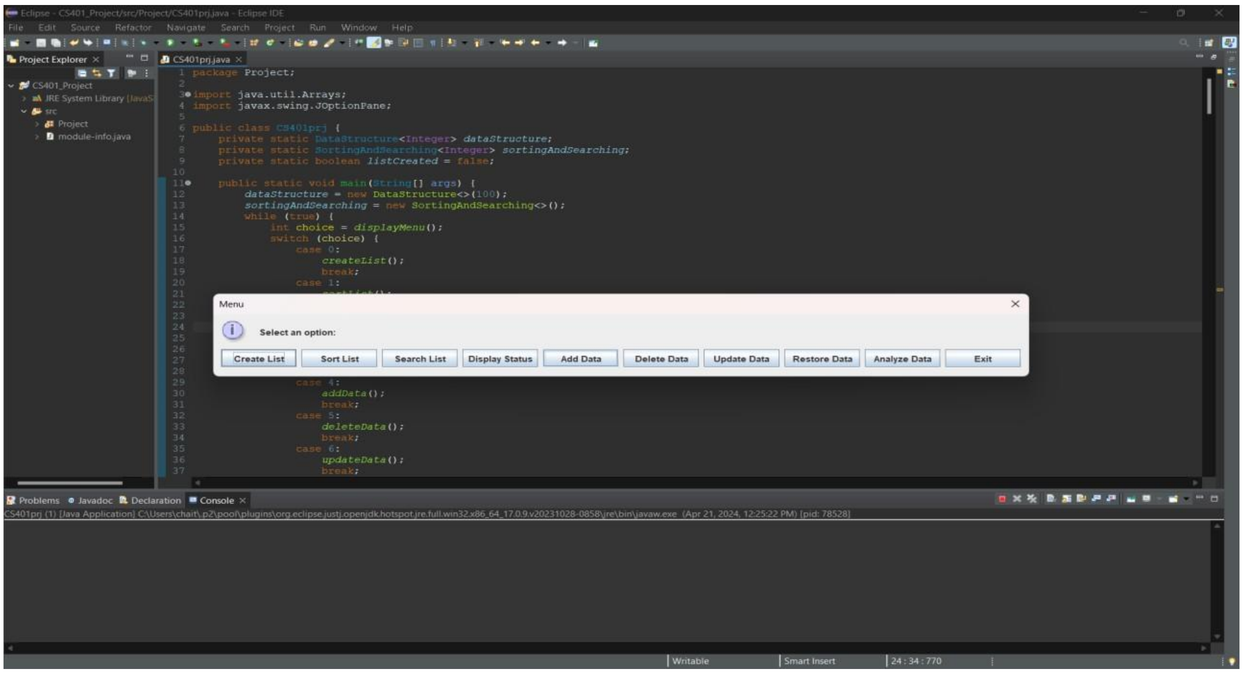
d) Operational document (user's manual: how to run your program, what is expected result or screen shots)

User Manual: List Management Program

This manual provides instructions on how to use the list management program. The program allows you to create a list of elements, add, delete, update, and search for elements within the list, and perform different sorting operations.



```java
package Project;

import java.util.Arrays;
import javax.swing.JOptionPane;

public class CS401prj {
    private static DataStructure<Integer> dataStructure;
    private static SortingAndSearching<Integer> sortingAndSearching;
    private static boolean listCreated = false;

    public static void main(String[] args) {
        dataStructure = new DataStructure<>(100);
        sortingAndSearching = new SortingAndSearching<>();
        while (true) {
            int choice = displayMenu();
            switch (choice) {
                case 0:
                    createList();
                    break;
                case 1:
```

```java
                case 4:
                    addData();
                    break;
                case 5:
                    deleteData();
                    break;
                case 6:
                    updateData();
                    break;
```

```
CS401prj (1) [Java Application] C:\Users\chait\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe  (Apr 21, 2024, 12:25:22 PM) [pid: 78528]
```

Using the Program

The program typically interacts with the user through a menu-driven interface. Here's a breakdown of the common functionalities:

Create a List:

- Select the option to create a new list.
- Predefined dataset with 100 values list will be taken as an input.

Adding Elements:

- Select the option to add an element.
- Enter the new element's value when prompted.

- The program will add the element to the existing list and provide a confirmation message.

Deleting Elements:

- Select the option to delete an element.
- Enter the value of the element you want to remove.
- The program will search for the element and remove it if found. It will display a message indicating success or failure.

Updating Elements:

- Select the option to update an element.
- Enter the value of the element you want to modify.
- Then, provide the new value for the element.
- The program will locate the element and update its value if found. It will display a message indicating success or failure.

Displaying List Status:

- Select the option to display the list status.
- The program will show the current number of elements and potentially the entire list contents.

Sorting the List:

- The program might offer options to sort the list using different algorithms (e.g., selection sort, merge sort).
- Select the desired sorting algorithm.
- The program will sort the list and potentially display the sorted elements.

Searching the List:

- The program might provide options for different search algorithms (e.g., linear search, binary search).
- Select the search algorithm you want to use.
- Enter the value of the element you want to find.
- The program will search for the element and indicate success or failure.

Exiting the Program:

- The program likely has an option to exit.
- Select the exit option to terminate the program.

Expected Results

- ➢ The program should respond to user input and perform the selected actions accordingly. Here are some examples of expected results:

- After creating a list, the program should confirm the number of elements created.
- Adding an element should result in a confirmation message upon successful addition.
- Deleting an element should display a message indicating success if found or failure if not found.
- Updating an element should provide a message confirming the update or indicate if the element wasn't found.
- Displaying the list status should show the current number of elements and potentially the entire list contents.
- Sorting the list should rearrange the elements based on the chosen algorithm.
- Searching the list should indicate success if the element is found and failure if not found.

Error Handling

- The program should ideally handle invalid user input gracefully. Here are some examples of error handling:
- If the user enters a non-numeric value when specifying the number of elements, the program should prompt the user to enter a valid number.
- If the user tries to delete or update a non-existent element, the program should display a message indicating the element was not found.

Note: The specific functionalities and error handling might vary depending on the actual program implementation.

Additional Information

The program might also offer features like comparing the performance of different sorting algorithms by displaying the number of comparisons made during each sort. Consult the program's specific documentation or help messages for more details on available functionalities.