UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module	: 4COSC010C.3: Software Development II
Module Leader	: Deshan Sumanathilaka[PROG]/ Sulari Fernando[DES]
Date of submission	: 08 August 2022
Student ID	: 20210046 / w1912792
Student First Name	: Rukshan
Student Surname	: Dias

Table of Contents

"I confirm that I understand what plagiarism / collusion / contract cheating is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."

Name : H.Rukshan Piyumadu Dias

Student ID : w1912792

1.Test Cases

1.1 Array version (Task 1)

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	Fuel Queue Initialized Correctly After the program starts, 100 or VFQ	Displays 'empty' for all queues.	Displays 'empty' for all Queues.	Pass
2	View Empty Queues, 101 or VEQ *Assume: Jane already has been added to Queue 2, 1st location*	Display, Every queue is empty except queue 2 1st location.	Display, Every queue is empty except queue 2 1st location.	Pass
3	Add passenger "Jane" to Queue 2 102 or ACQ Enter Queue: 2 Enter Name: Jane	Display 'Jane added to queue 2 successfully"	Display 'Jane added to queue 2 successfully"	Pass
4	Add passenger "Matt" to Queue 10 102 or ACQ Enter Queue: 10 Enter Name: Matt	Display, "Invalid queue number" And ask for the input of the queue number again.	Display, "Invalid queue number" And ask for the input of the queue number again.	Pass
5	Add passenger "Liam" to Queue 1. 102 or ACQ. Enter Queue: 1 Enter Name: Liam *Assume: queue 1 is already full.*	Display, "Queue 1 is full now. check another queue or try again later."	Display, "Queue 1 is full now. check another queue or try again later."	Pass
6	Remove customer from queue 2, location 1. 103 or RCQ. Enter queue number:2 Enter location:1 *Assume: Passenger in queue2 1st index is, Jane.*	Display, "Jane removed"	Display, "Jane removed"	Pass
7	Remove customer from queue 2, location 2. 103 or RCQ. Enter queue number:2 Enter location:2	Display," location 2 is Empty. pls try again with the correct location."	Display," location 2 is Empty. pls try again with the correct location."	Pass

	Assume: Queue 2, 2nd location is empty			
8	Remove a served customer “John” from queue 1. *Assume: Passenger called, John is there in queue 2.*	Display, ” John removed”	Display, ” John removed”	Pass
9	Remove a served customer “Harry” from queue 1. *Assume: there’s no customer called, Harry in queue 1.*	Display, “There's no passenger called, Harry in queue 1.”	Display, “There's no passenger called, Harry in queue 1.”	Pass
10	View Customers Sorted in alphabetical order 105 or VCS: *Assume: there are 3 customers in queue 1, 2 customers in queue 2 and no customers in queue 3.*	Queue 1 --> abbe, james, zayn Queue 2 --> ben, david Queue 3 -->	Queue 1 --> abbe, james, zayn Queue 2 --> ben, david Queue 3 -->	Pass
11	Store Program Data into file. 106 or SPD:	Display, “Data stored in data.txt file..”	Display, “Data stored in data.txt file..”	Pass
12	Load Program Data from file. 107 or LPD:	Get data from txt file and add those values to queue.	Get data from txt file and add those values to queue.	Pass
13	View Remaining Fuel Stock. 108 or STK:	Display remaining fuel stock in Fuel station. “Remaining fuel stock: 6540L”	Display remaining fuel stock in Fuel station. “Remaining fuel stock: 6540L”	Pass
14	Add Fuel Stock. 109 or AFS: fuel amount to add: 10	Display, “10L has been added to fuel stock. New fuel stock is, 6550L”	Display, “10L has been added to fuel stock. New fuel stock is, 6550L”	Pass
15	Add Fuel Stock. 109 or AFS: fuel amount to add: -80	Display, “invalid, fuel amount should be greater than 0.” And ask to enter the input again.	Display, “invalid, fuel amount should be greater than 0.” And ask to enter the input again.	Pass
16	Add Fuel Stock. 109 or AFS: fuel amount to add: Hello	Display, “Enter a valid amount.”	Display, “Enter a valid amount.”	Pass

		And ask to enter the input again.	And ask to enter the input again.	
17	Exit the Program. 999 or EXT:	Display, “Exit from the program. Thank you for using Fuel Management System!”	Display, “Exit from the program. Thank you for using Fuel Management System!”	Pass
18	Enter an invalid input in menu option. Enter a option: 10000	Display, “invalid input..pls try again.” And ask again to input. “Enter a option: ”	Display, “invalid input..pls try again.” And ask again to input. “Enter a option: ”	Pass

1.2 Class version (Task 2,3)

No	Test Case	Expected Result	Actual Result	Pass/Fail
19	Add customer to a Queue. 102 or ACQ: Enter first name: jason Enter second name: dias Enter vehicle number: we2312 Enter liters required: 20 *Assume: queue 1 is the array with the minimum length.*	Display, “jason added to queue number 1” Object has been added to array with a minimum length (queue 1)	Display, “jason added to queue number 1” Object has been added to array with a minimum length (queue 1)	Pass
20	Add customer to a Queue. 102 or ACQ: Enter first name: jason Enter second name: dias Enter vehicle number: we2312 Enter liters required: Hello	Display, “Invalid input. Enter a number..” And ask again to input. “Enter liters required:”	Display, “Invalid input. Enter a number..” And ask again to input. “Enter liters required:”	Pass
21	Add customer to a Queue. 102 or ACQ: Enter first name: Rukshan Enter second name: dias	Display: “Fuel Queues are full now.	Display: “Fuel Queues are full now.	Pass

	Enter vehicle number: cb2312 Enter liters required: 30 *Assume: All queue are full.*	You will be added to a waiting list.”	You will be added to a waiting list.”	
22	Remove a served customer. 104 or PCQ: Enter name: jason *Assume: there’s a passenger called Jason in Fuel queue.*	Display: “jason removed”	Display: “jason removed”	Pass
23	Remove a served customer. 104 or PCQ: Enter name: Hello *Assume: there’s no passenger called Hello in Fuel queue.*	Display: “Entered name can't be found in fuel queue..”	Display: “Entered name can't be found in fuel queue..”	Pass
24	Remove a served customer. 104 or PCQ: Enter name: Sam *Assume: Sam was in queue 1. There is a passenger called, “Smith” in waiting queue.*	Display: “Sam removed Smith has been added to Fuel queue 1, from waiting queue”	Display: “Sam removed Smith has been added to Fuel queue 1, from waiting queue”	Pass

1.3 Javafx (Task 4)

No	Test Case	Expected Result	Actual Result	Pass/Fail
25	View GUI from menu option. "111 or GUI: Enter a option: 111	Display: “Opening Fuel Station GUI” And GUI will open in background	Display: “Opening Fuel Station GUI” And GUI will open in background	Pass
26	GUI : Click on “View queus” button	Redirect to ViewAllQueues.fxml file	Redirect to ViewAllQueues.fxml file	Pass
27	GUI : Click on “Search Passenger” button	Redirect to SearchPassengers.fxml file	Redirect to SearchPassengers.fxml file	Pass

28	GUI : Click on “Fuel Income” button	Redirect to FuelIncome.fxml file	Redirect to FuelIncome.fxml file	Pass
29	GUI: On click view button on Fuel Queue tab	Details about Fuel Queue will display.	Details about Fuel Queue will display.	Pass
30	GUI: On click view button on waiting Queue tab	Details about waiting Queue will display.	Details about waiting Queue will display.	Pass
31	GUI: On click Go back button	Return to the Home page (HomePage.fxml)	Return to the Home page (HomePage.fxml)	Pass
32	GUI: On click search Name: Jason *Assume: there is a passenger called Jason in either fuel queue or waiting queue.*	Display the details of Jason	Display the details of Jason	Pass
33	GUI: On click search Name: Malik *Assume: there’s no passenger called Malik in either fuel queue or waiting queue.*	Display,” Entered name can't be found in fuel queue”	Display,” Entered name can't be found in fuel queue”	Pass
34	GUI: On click view button on Fuel income page.	Display total income of each fuel pump and fuel station.	Display total income of each fuel pump and fuel station.	Pass

2. Discussion

The test cases that I mentioned above, covers the all parts of the programme. Test case 01 use to check queue is initialised properly. It will display empty in all queues because nothing is added yet. Test case 02 will check what are the empty slots. If customer is added to a slot, that slot won't be displayed in this. In Test case 3,4 and 5. It covers the add customer to queue part of the programme. In 3rd case, customer will be added to queue. In 4th case, customer won't be added to queue because enter input for queue number is not between 1 to 3. In 5th case, customer wont be added because queue is already full. In Test case 6 and 7 customer will be removed from queue index. If enter index is null, a message will be display to user. And in Test case 8 and 9. Served customer will be removed. If entered name is not in the queue, it will print, "Name can not be found". Queue sorting has been done in Test case 10, by using the Bubble sort method. Which will display names in alphabetical order. Test case 11 12 will store the entered data in a file. And load the data to the queues. File handling has been used for those test cases. I have chosen Test cases 13 – 16 to show how fuel stock has been used in this programme. The programme can display the remaining fuel stock. And if you want to add Fuel stock, the programme will display "invalid input " until you enter the correct input. Input validation has been done. I choose Test cases 17 and 18 to show that the menu is working correctly. When you enter an option which is not in the menu, "invalid input" will be printed and ask for input again. Input validation has been done.

From 19 – 21, it mentioned about adding customers to the queue in the Class version. Input validation has been done for this part. It will display a message if the fuel queue is full. I have used Test cases 22 – 24 to show how the waiting queue is used. when adding a passenger, if the fuel queue is full passenger will be added to the waiting queue. And when removing a served customer, the first passenger in the waiting queue will be added to the fuel queue. The circular queue has been used.

Test case 25 is the starting test case of the Javafx part. When entering 111 or GUI the JavaFX application will open. From 26-28 when the user clicks on a button, the user will redirect to its relevant page. New scenes have been used. Test cases 29 and 30 will display the fuel queues and waiting for queues and their details. In test cases 32 and 33, it will display the details of a searched passenger. If the search passenger is not in the queue, it will display, "Entered name can't be found in fuel queue". And for test case number 34, it will show the fuel income of each pump and the total income of the Station.

The chosen Test cases ensure that this programme covered all aspects that can be faced.

3. Code

3.1 Array Version (Task 1)

```
import java.io.*;
import java.util.*;

public class fuelManagementSystem {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        //initialising variables
        char replay='y';
        String[][] queue=new String[3][6];
        int fuel_stock=6600;

        System.out.println("Hello Welcome to Fuel Queue Management System.");

        while (replay=='y') {
            //Printing menu
            System.out.println("100 or VFQ: View all Fuel Queues.\n" +
                    "101 or VEQ: View all Empty Queues.\n" +
                    "102 or ACQ: Add customer to a Queue.\n" +
                    "103 or RCQ: Remove a customer from a Queue.\n" +
                    "104 or PCQ: Remove a served customer.\n" +
                    "105 or VCS: View Customers Sorted in alphabetical order\n" +
                    "106 or SPD: Store Program Data into file.\n" +
                    "107 or LPD: Load Program Data from file.\n" +
                    "108 or STK: View Remaining Fuel Stock.\n" +
                    "109 or AFS: Add Fuel Stock.\n" +
                    "999 or EXT: Exit the Program.\n");

            while (true) {
                System.out.print("Enter a option:");
                String option = sc.next().toUpperCase(); //getting user input

                switch (option) {
                    case "100", "VFQ":
                        view_queues("1", queue[0]);
                        view_queues("2", queue[1]);
                        view_queues("3", queue[2]);
                        System.out.println("----------------------------------------
--------\n");
                        break;

                    case "101", "VEQ":
                        System.out.println("View empty queues");
                        view_empty_queues(queue);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "102", "ACQ":
                        fuel_stock=add_customer(queue,fuel_stock);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "103", "RCQ":
                        fuel_stock=remove_customer(queue,fuel_stock);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "104", "PCQ":
```

```
                        System.out.println("Remove a served customer.");
                        remove_served_customer(queue);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "105", "VCS":
                        sort_queue(queue);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "106", "SPD":
                        //store_data_toFile(queue);
                        store_data_toFile(queue,fuel_stock);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "107", "LPD":
                        show_data_fromFile(queue,fuel_stock);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "108", "STK":
                        System.out.println("Remaining fuel stock: "+fuel_stock+"L");
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "109", "AFS":
                        fuel_stock+=add_fuel_stock(fuel_stock);
                        System.out.println("New fuel stock is, "+fuel_stock+"L");
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "999", "EXT":
                        replay='n';
                        System.out.println("Exit from the program.\nThank you for
using Fuel Management System!");
                        break;
                    default:
                        System.out.println("invalid input..pls try again.");
                        continue;

                }
                break;
            }
        }
    }
    //-----------------------------Methods---------------------------------

    //for 100 or VFQ: View all Fuel Queues.
    private static void view_queues(String type,String [] queue){
        for (int i=0;i<queue.length;i++){
            if ((queue[i])==null){
                System.out.println("queue "+type+": place "+(i+1)+" -> Empty");
            }else {
                System.out.println("queue "+type+": place "+(i+1)+" -> "+queue[i]);
            }
        }
        System.out.println("\n");
    }

    // for 102 or ACQ: Add customer to a Queue.
    private static int add_customer(String[][]queue,int fuel){
        Scanner sc=new Scanner(System.in);
```

```
        System.out.print("Enter name:");
        String name = sc.next().toLowerCase();
        int queue_no;
        while (true){
            try {
                System.out.print("Enter queue: ");
                queue_no = sc.nextInt();
                if (queue_no>3 || queue_no<1){
                    System.out.println("Invalid queue number.");
                    continue;
                }
                break;
            }catch (Exception e){
                System.out.println("Invalid input. enter a number.");
                sc.nextLine(); //clear input
                continue;
            }
        }
        int num=queue_no-1;
        for (int col=0;col<queue[num].length;col++) {
            int lastVal = queue[num].length - 1;
            if (queue[num][lastVal]==null){
            if (queue[num][col] == null) {
                queue[num][col] = name;
                System.out.println(name+" added to queue "+queue_no);
                fuel -= 10;
                break;
            }
            }else {
                System.out.println("Queue "+queue_no+" is full now. check another
queue or try again later.");
                break;
            }
        }
        return fuel;
    }

    // for 103 or RCQ: Remove a customer from a Queue.
    private static int remove_customer(String [][] queue,int fuel){
        Scanner sc=new Scanner(System.in);
        int queue_no;
        int location;
        while (true){
            try {
                System.out.print("Enter queue number:");
                queue_no=sc.nextInt();
                if (queue_no<1 || queue_no>3){
                    System.out.println("Enter a valid queue number.");
                    continue;
                }
                System.out.print("Enter location:");
                location=sc.nextInt();
                if (location<1 || location>6){
                    System.out.println("Enter a valid queue number.");
                    sc.nextLine();//clear input
                    continue;
                }
                break;
            }catch (Exception e){
                System.out.println("Enter a number not a letter.");
                sc.nextLine();//clear input
                continue;
            }
```

```
        }
        if (queue[queue_no-1][location-1]!=null){
            System.out.println(queue[queue_no-1][location-1]+" removed.");
            queue[queue_no-1][location-1]=null;
            fuel+=10;
            queue=customers_moveFront(queue,queue_no); //moving customers to front
        }else {
            System.out.println("location "+location+" is Empty. pls try again with
correct location.");
        }

        return fuel;
    }

    private static String[][] customers_moveFront(String[][]queue,int queue_no){
        //reordering queue
        for (int i=0;i<queue[queue_no-1].length;i++){
            if (queue[queue_no-1][i]==null){
                for (int k=i+1; k<queue[queue_no-1].length; k++){
                    queue[queue_no-1][k-1] = queue[queue_no-1][k];
                }
                int lastVal=queue[queue_no-1].length-1;
                queue[queue_no-1][lastVal]=null;
                break;
            }
        }return queue;
    }

    // for 101 or VEQ: View all Empty Queues.
    private static void view_empty_queues(String [][] queue){
        for (int row=0;row<queue.length;row++){
            System.out.println("\nQueue No: "+(row+1));
            for (int col=0;col<queue[row].length;col++){
                if (queue[row][col]==null){
                    System.out.println("Queue no, "+(row+1)+" location no, "+(col+1)+"
--> Empty");
                }
            }

        }
        System.out.println("-------------------------------------------------");
    }

    //for 104 or PCQ: Remove a served customer.
    private static void remove_served_customer(String [][] queue){
        Scanner sc=new Scanner(System.in);
        int queue_no;
        String customer_name;
        System.out.print("Enter name: ");
        customer_name=sc.nextLine();
        while (true){
            try {
                System.out.print("Enter his/her queue no: ");
                queue_no=sc.nextInt();
                if (queue_no<1 || queue_no>3) {
                    System.out.println("Enter a valid queue number.");
                    continue;
                }
                break;
            }catch (Exception e){
                System.out.println("Enter a number not a letter.");
                sc.nextLine();
```

```
                continue;
            }
        }
        //remove part
        int index=0;
        boolean isNameEqual=false;
        for (String i : queue[queue_no-1]){
            if (i!=null && i.equalsIgnoreCase(customer_name)){
                System.out.println(i+" removed");
                queue[queue_no-1][index]=null;
                isNameEqual=true;
                break;
            }index++;
        }
        if (!isNameEqual){//if isNameEqual is false
            System.out.println("There's no passenger called, "+customer_name+" in
queue "+queue_no);
        }
        queue=customers_moveFront(queue,queue_no); //moving customers to front
    }

    //for 109 or AFS: Add Fuel Stock.
    private static int add_fuel_stock(int fuel) {
        Scanner sc = new Scanner(System.in);
        int new_fuel_count;
        while (true) {
            try {
                System.out.print("Enter fuel amount to add: ");
                new_fuel_count = sc.nextInt();//get input
                if (new_fuel_count < 0) {
                    System.out.println("invalid, fuel amount should be greater than
0.");
                    //sc.nextLine();//clear input
                    continue;
                }
                System.out.println(new_fuel_count+"L has been added to fuel stock.");
                break;
            } catch (Exception e) {
                System.out.println("Enter a valid amount.");
                sc.nextLine();//clear input
                continue;
            }
        }
        return new_fuel_count;
    }
    //for 107 or LPD: Load Program Data from file.

    //for 106 or SPD: Store Program Data into file.
    private static void store_data_toFile(String[][] queue_list,int fuelStock){
        try {
            FileWriter dataFile=new FileWriter("Task01_data.txt");
            for (int row=0;row< queue_list.length;row++){
                for (int col=0;col<queue_list[row].length;col++){
                    dataFile.write(queue_list[row][col]+"\n");
                }
            }
            dataFile.write(String.valueOf(fuelStock));
            dataFile.close();
            System.out.println("Data stored in Task01_data.txt file..");
        }catch (IOException e){
            System.out.println("error when writing file..");
        }
    }
```

```
//107 or LPD: Load Program Data from file.
private static void show_data_fromFile(String[][] queue_list,int fuelStock){
    try{
        File loadedFile=new File("Task01_data.txt");
        Scanner readFile=new Scanner(loadedFile);
        while (readFile.hasNext()){
            for (int row=0;row< queue_list.length;row++){
                for (int col=0;col<queue_list[row].length;col++){
                    queue_list[row][col]=readFile.nextLine();
                }
            }
            fuelStock=readFile.nextInt();
        }
        readFile.close();
        System.out.println("Data has been loaded");
    }catch (IOException e){
        System.out.println("error occur");
    }
}


//for 105 or VCS: View Customers Sorted in alphabetical order
//https://www.geeksforgeeks.org/row-wise-sorting-2d-array/
private static void sort_queue(String [][]queue){
    String[][] copyArr=new String[3][6]; //copying queue to a new array
    //copying queue to new array
    for (int row=0;row< queue.length;row++){
        for (int col=0;col< queue[row].length;col++){
            copyArr[row][col]=queue[row][col];
        }
    }

    // loop for rows of matrix
    for (int i = 0; i < copyArr.length; i++) {

        // loop for column of matrix
        for (int j = 0; j < copyArr[i].length; j++) {

            // loop for comparison and swapping
            for (int k = 0; k < copyArr[i].length - j - 1; k++) {
                try {
                    if (Str_compare(copyArr[i][k],copyArr[i][k + 1])  >0) {

                        // swapping elements
                        String t = copyArr[i][k];
                        copyArr[i][k] = copyArr[i][k + 1];
                        copyArr[i][k + 1] = t;
                    }
                }catch (NullPointerException n){
                    continue;
                }

            }
        }
    }

    // printing the sorted queue
    for (int i = 0; i < copyArr.length; i++) {
        System.out.print("Queue "+(i+1)+" --> ");
        for (int j = 0; j < copyArr[i].length; j++)
            if (copyArr[i][j]!=null){
                System.out.print(copyArr[i][j] + ", ");
```

```
                }

            System.out.println();
        }
    }

    //String comparison
    private static int Str_compare(String s1, String s2){
        int len1=s1.length();
        int len2=s2.length();

        int min;
        //finding min value
        if (len1>len2){
            min =len2;
        }else {
            min=len1;
        }
         char[] ch1=s1.toCharArray(); //convert string to letter by letter array
         char[] ch2=s2.toCharArray();

         for (int i=0;i<min;i++){
             char letter1=ch1[i];
             char letter2=ch2[i];
             if (letter1!=letter2){
                 int val=letter1-letter2;
                 return val;
             }
         }
    return 0;
    }
}
```

3.2 Class Version (Task 2 & 3)

3.2.1 FuelStation.java (Main class)

```
import java.util.Scanner;

public class PetrolStation {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        //initialising variables
        char replay='y';

        System.out.println("Hello Welcome to Fuel Queue Management System.");

        while (replay=='y') {
            //Printing menu
            System.out.println("100 or VFQ: View all Fuel Queues.\n" +
                    "101 or VEQ: View all Empty Queues.\n" +
                    "102 or ACQ: Add customer to a Queue.\n" +
                    "103 or RCQ: Remove a customer from a Queue.\n" +
                    "104 or PCQ: Remove a served customer.\n" +
                    "105 or VCS: View Customers Sorted in alphabetical order\n" +
```

```
                    "106 or SPD: Store Program Data into file.\n" +
                    "107 or LPD: Load Program Data from file.\n" +
                    "108 or STK: View Remaining Fuel Stock.\n" +
                    "109 or AFS: Add Fuel Stock.\n" +
                    "110 or IFQ: income of each Fuel queue.\n"+
                    "999 or EXT: Exit the Program.\n");

            while (true) {
                System.out.print("Enter a option:");
                String option = sc.next().toUpperCase(); //getting user input

                switch (option) {
                    case "100", "VFQ":
                        //fuelQueue[0][1]=new FuelQueue("Shan","Dias","we1232",10);
                        //fuelQueue[0][5]=new FuelQueue("Rukshan","Dias","we1232",10);

                        FuelQueue.view_queues();

                        System.out.println("----------------------------------------
--------\n");
                        break;

                    case "101", "VEQ":
                        System.out.println("View empty queues");
                        FuelQueue.view_empty_queues();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "102", "ACQ":
                        FuelQueue.add_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "103", "RCQ":
                        FuelQueue.remove_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "104", "PCQ":
                        System.out.println("Remove a served customer.");
                        FuelQueue.remove_served_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "105", "VCS":
                        FuelQueue.sort_queue();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "106", "SPD":
                        FuelQueue.store_data_toFile();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "107", "LPD":
                        FuelQueue.show_data_fromFile();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "108", "STK":
                        System.out.println("Remaining fuel stock:
"+FuelQueue.getFuelStock()+"L");
                        System.out.println("----------------------------------------
```

```
--------\n");
                        break;
                    case "109", "AFS":
                        FuelQueue.add_fuel_stock();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "110", "IFQ":
                        FuelQueue.DisplayFuelIncome();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "999", "EXT":
                        replay='n';
                        System.out.println("Exit from the program.");
                        break;
                    default:
                        System.out.println("invalid input..pls try again.");
                        continue;

                }
                break;
            }
        }

    }
}
```

3.2.2 FuelQueue.java

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class FuelQueue extends passenger{
    //class attributes
    private static int fuelStock=6600;
    private static int [] FuelIncome=new int[5];
    private static FuelQueue fuelQueue[][]=new FuelQueue[5][6];

    //class methods

    //constructor
    public FuelQueue(String fName, String sName, String vehicalNo, int liters){
        super(fName,sName,vehicalNo,liters);
    }

    //FuelQueue array getter
    public static FuelQueue[][] getFuelQueue() {
        return fuelQueue;
    }

    //100 or VFQ: View all Fuel Queues
    public static void view_queues(){
        FuelQueue queue[][]=getFuelQueue();
        FuelQueue PassesngerQueue;
        for (int i=0;i<queue.length;i++){
            for (int j=0;j< queue[i].length;j++){
```

```
                PassesngerQueue=queue[i][j];
                if ((queue[i][j])==null){
                    System.out.println("queue "+(i+1)+": place "+(j+1)+" -> Empty");
                }else {
                    System.out.println("queue "+(i+1)+": place "+(j+1)+" ->
"+PassesngerQueue.getFirstName());
                }
            }
            System.out.println("\n");
        }

    }

    //101 or VEQ: View all Empty Queues
    public static void view_empty_queues(){
        FuelQueue queue[][]=getFuelQueue();
        for (int row=0;row<queue.length;row++){
            System.out.println("\nQueue No: "+(row+1));
            for (int col=0;col<queue[row].length;col++){
                if (queue[row][col]==null){
                    System.out.println("Queue no, "+(row+1)+" location no, "+(col+1)+"
--> Empty");
                }
            }

        }
    }

    //102 or ACQ: Add customer to a Queue.
    public static void add_customer(){
        FuelQueue queue[][]=getFuelQueue();
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter first name:");
        String Fname = sc.next();
        System.out.print("Enter second name:");
        String Sname = sc.next();
        System.out.print("Enter vehicle number:");
        String VehicleNo = sc.next();
        int liters;
        while (true){
            try {
                System.out.print("Enter liters required:");
                liters = sc.nextInt();
                break;
            }catch (Exception e){
                System.out.println("Invalid input. Enter a number..");
                sc.nextLine(); //clear input
                continue;
            }

        }

        int num=ShortestQueue(queue);//check the shortest queue

        for (int col=0;col<queue[num].length;col++) {
            int lastVal = queue[num].length - 1;
            if (queue[num][lastVal]==null){
                if (queue[num][col] == null) {
                    queue[num][col] =new FuelQueue(Fname,Sname,VehicleNo,liters);
                    System.out.println(Fname+" added to queue number "+(num+1));
                    fuelStock -= liters;
                    setFuelIncome(num,liters);
                    break;
```

```
                }
            }else {
                System.out.println("Fuel Queues are full now.");
                System.out.println("You will be added to a waiting list.");
                WaitingList.enqueue(Fname,Sname,VehicleNo,liters);
                break;
            }
        }
    }

    //Fuel income array
    public static void setFuelIncome(int index,int liters) {
        int income=liters*430;
        FuelIncome[index]+=income;
    }

    //108 STK - fuel stock Getter
    public static int getFuelStock() {
        return fuelStock;
    }

    //find the shortest queue
    public static int ShortestQueue(FuelQueue [][] queue){
        int shortestIndex=0;
        int NullCount=0;
        int temp=0;
        for (int row=0;row<queue.length;row++){
            for (int col=0;col<queue[row].length;col++){
                if (queue[row][col]==null){
                    NullCount+=1;
                }
            }
            if (NullCount>temp){
                shortestIndex=row;
            }
            temp=NullCount;
            NullCount=0;

        }
        return shortestIndex;
    }

    //103 or RCQ: Remove a customer from a Queue
    public static void remove_customer(){
        FuelQueue queue[][]=getFuelQueue();
        Scanner sc=new Scanner(System.in);
        int queue_no;
        int location;
        while (true){
            try {
                System.out.print("Enter queue number:");
                queue_no=sc.nextInt();
                if (queue_no<1 || queue_no>5){
                    System.out.println("Enter a valid queue number.");
                    continue;
                }
                System.out.print("Enter location:");
                location=sc.nextInt();
                if (location<1 || location>6){
                    System.out.println("Enter a valid queue number.");
                    sc.nextLine();//clear input
                    continue;
                }
```

```
                break;
            }catch (Exception e){
                System.out.println("Enter a number not a letter.");
                sc.nextLine();//clear input
                continue;
            }

        }
        passenger Passenger;
        Passenger=queue[queue_no-1][location-1];
        if (queue[queue_no-1][location-1]!=null){
            System.out.println(Passenger.getFirstName()+" removed.");
            FuelQueue.setFuelStock(Passenger.getLitersRequired());

            queue[queue_no-1][location-1]=null;

            queue=customers_moveFront(queue,queue_no); //moving customers to front
        }else {
            System.out.println("location "+location+" is Empty. pls try again with
correct location.");
        }

    }

    //move customer front
    private static FuelQueue[][] customers_moveFront(FuelQueue[][]queue,int queue_no){
        //reordering queue
        for (int i=0;i<queue[queue_no-1].length;i++){
            if (queue[queue_no-1][i]==null){
                for (int k=i+1; k<queue[queue_no-1].length; k++){
                    queue[queue_no-1][k-1] = queue[queue_no-1][k];
                }
                int lastVal=queue[queue_no-1].length-1;
                queue[queue_no-1][lastVal]=null;
                break;
            }
        }return queue;
    }

    //setter for fuel stock
    public static void setFuelStock(int liters) {
        FuelQueue.fuelStock += liters;
    }

    //104 or PCQ: Remove a served customer.
    public static void remove_served_customer(){
        FuelQueue queue[][]=getFuelQueue();
        boolean isNameEqual=false;
        int col;
        int row;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter name: ");
        String customer_name=sc.nextLine();
        try {
            for (row=0;row< queue.length;row++){
                for ( col=0;col< queue[row].length;col++){
                    if (queue[row][col]!=null &&
((queue[row][col].getFirstName()).equalsIgnoreCase(customer_name))){
                        System.out.println(queue[row][col].getFirstName()+" removed");
                        queue[row][col]=null;
                        isNameEqual=true;
                        break;
                    }
```

```
                }
                if (isNameEqual==true){
                    break;
                }
            }customers_moveFront(queue,row+1);
            int lastVal=queue[row].length-1;
            queue[row][lastVal]=WaitingList.dequeue();
            System.out.println(queue[row][lastVal].getFirstName()+" has been added to
Fuel queue "+(row+1)+", from waiting queue");

        }catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Entered name can't be found in fuel queue..");
        }

    }

    //105 or VCS: View Customers Sorted in alphabetical order
    public static void sort_queue(){
        FuelQueue queue[][]=getFuelQueue();
        FuelQueue[][] copyArr=new FuelQueue[5][6]; //copying queue to a new array
        //copying queue to new array
        for (int row=0;row< queue.length;row++){
            for (int col=0;col< queue[row].length;col++){
                copyArr[row][col]=queue[row][col];
            }
        }

        // loop for rows of matrix
        for (int i = 0; i < copyArr.length; i++) {

            // loop for column of matrix
            for (int j = 0; j < copyArr[i].length; j++) {

                // loop for comparison and swapping
                for (int k = 0; k < copyArr[i].length - j - 1; k++) {
                    try {
                        if (copyArr[i][k].getFirstName().compareTo(copyArr[i][k +
1].getFirstName()) >0) {

                            // swapping elements
                            FuelQueue t = copyArr[i][k];
                            copyArr[i][k] = copyArr[i][k + 1];
                            copyArr[i][k + 1] = t;
                        }
                    }catch (NullPointerException n){
                        continue;
                    }

                }
            }
        }

        // printing the sorted queue
        for (int i = 0; i < copyArr.length; i++) {
            System.out.print("Queue "+(i+1)+" --> ");
            for (int j = 0; j < copyArr[i].length; j++)
                if (copyArr[i][j]!=null){
                    System.out.print(copyArr[i][j].getFirstName() + ", ");
                }

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

```
    //106 or SPD: Store Program Data into file.
    public static void store_data_toFile(){
        FuelQueue queue[][]=getFuelQueue();
        try {
            FileWriter data_file= new FileWriter("data.txt");
            for (int row=0;row<queue.length;row++){
                data_file.write("Queue "+(row+1)+"\n");
                for (int col=0;col<queue[row].length;col++){
                    data_file.write("Queue no, "+(row+1)+" location "+(col+1)+":\n");
                    if (queue[row][col]!=null){
                        data_file.write("\tFirst Name:
"+queue[row][col].getFirstName()+"\n\tSecond Name:
"+queue[row][col].getSecondName()+"\n\tVehicle No:
"+queue[row][col].getVehicleNo()+"\n\tLiters requied:
"+queue[row][col].getLitersRequired()+"\n");
                    }else {
                        data_file.write("\tEmpty space\n");
                    }
                }
                data_file.write("\n");
            }
            data_file.close();
            System.out.println("wrote to file..");
        }catch (IOException e){
            System.out.println("error occurred..");
        }
    }

    //107 or LPD: Load Program Data from file.
    public static void show_data_fromFile(){
        try {
            File dataFile=new File("data.txt");
            Scanner readFile=new Scanner(dataFile);
            String fileLine;
            while (readFile.hasNext()){
                fileLine=readFile.nextLine();
                System.out.println(fileLine);
            }
        }catch (IOException e){
            System.out.println("Error");
        }
    }

    //for 109 or AFS: Add Fuel Stock.
    public static void add_fuel_stock() {
        Scanner sc = new Scanner(System.in);
        int new_fuel_count;
        while (true) {
            try {
                System.out.print("Enter fuel amount to add: ");
                new_fuel_count = sc.nextInt();
                if (new_fuel_count < 0) {
                    System.out.println("invalid, should be grater than 0.");
                    continue;
                }
                break;
            } catch (Exception e) {
                System.out.println("Enter a valid amount.");
                sc.nextLine();//clear input
                continue;
            }
        }
```

```
        setFuelStock(new_fuel_count);
    }

    //110 or IFQ: display fuel income
    public static void DisplayFuelIncome(){
        for(int i=0;i<FuelIncome.length;i++){
            System.out.println("Fuel queue "+(i+1)+" income: "+FuelIncome[i]);
        }
    }

    //fuel income array Getter -for GUI
    public static int[] getFuelIncome() {
        return FuelIncome;
    }
}
```

3.2.3 passenger.java

```
public class passenger {
    //class attributes
    private String FirstName;
    private String SecondName;
    private String VehicleNo;
    private int LitersRequired;

    //class methods
    public passenger(String fName, String sName, String vehicleNo, int liters){
        this.FirstName=fName;
        this.SecondName=sName;
        this.VehicleNo =vehicleNo;
        this.LitersRequired=liters;
    }

    //getters
    public String getFirstName() {
        return FirstName;
    }

    public String getSecondName() {
        return SecondName;
    }

    public String getVehicleNo() {
        return VehicleNo;
    }

    public int getLitersRequired() {
        return LitersRequired;
    }

}
```

3.2.4 WaitingList.java

```
/*
referred from below sources:
https://www.programiz.com/dsa/circular-queue
```

```
https://github.com/im-mani-teckieshare/data-structure-algorithm-java/blob/main/data-structure/src/circularqueue/CircularQueue.java
*/

public class WaitingList {
    //class attributes
    private static int capacity=6;
    private static int front = 0;
    private static int rear = -1;
    private static int size = 0;
    private static FuelQueue waiting_list[] = new FuelQueue[capacity];

    //class methods


    public static void enqueue(String Fname,String Sname, String VehicleNo,int liters)
{
        if(isFull()) {
            System.out.println("Queue is full can't insert");
            return;
        }
        rear = (rear + 1) % capacity;
        waiting_list[rear] = new FuelQueue(Fname,Sname,VehicleNo,liters);
        size++;

    }
    public static FuelQueue dequeue() {
        if(isEmpty()) {
            System.out.println("Queue is empty");
            return null;
        }
        FuelQueue data = waiting_list[front];
        front = (front + 1) % capacity;
        size--;
        return data;
    }


    public static boolean isFull() {
        return size == capacity;
    }

    public static boolean isEmpty() {
        return size == 0;
    }
}
```

3.3 Javafx Version (Task 4)

3.3.1 FuelStation.java (Main class)

```
package com.example.w1912792_task04;

import java.util.Scanner;

public class FuelStation {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
```

```
        //initialising variables
        char replay='y';


        System.out.println("Hello Welcome to Fuel Queue Management System.");

        while (replay=='y') {
            //Printing menu
            System.out.println("100 or VFQ: View all Fuel Queues.\n" +
                    "101 or VEQ: View all Empty Queues.\n" +
                    "102 or ACQ: Add customer to a Queue.\n" +
                    "103 or RCQ: Remove a customer from a Queue.\n" +
                    "104 or PCQ: Remove a served customer.\n" +
                    "105 or VCS: View Customers Sorted in alphabetical order\n" +
                    "106 or SPD: Store Program Data into file.\n" +
                    "107 or LPD: Load Program Data from file.\n" +
                    "108 or STK: View Remaining Fuel Stock.\n" +
                    "109 or AFS: Add Fuel Stock.\n" +
                    "110 or IFQ: income of each Fuel queue.\n"+
                    "111 or GUI: Display GUI application of Fuel System\n"+
                    "999 or EXT: Exit the Program.\n");

            while (true) {
                System.out.print("Enter a option:");
                String option = sc.next().toUpperCase(); //getting user input

                switch (option) {
                    case "100", "VFQ":
                        FuelQueue.view_queues();
                        System.out.println("----------------------------------------
--------\n");
                        break;

                    case "101", "VEQ":
                        System.out.println("View empty queues");
                        FuelQueue.view_empty_queues();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "102", "ACQ":
                        FuelQueue.add_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "103", "RCQ":
                        FuelQueue.remove_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "104", "PCQ":
                        System.out.println("Remove a served customer.");
                        FuelQueue.remove_served_customer();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "105", "VCS":
                        FuelQueue.sort_queue();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "106", "SPD":
                        FuelQueue.store_data_toFile();
                        System.out.println("----------------------------------------
```

```
--------\n");
                        break;
                    case "107", "LPD":
                        FuelQueue.load_data_fromFile();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "108", "STK":
                        System.out.println("Remaining fuel stock:
"+FuelQueue.getFuelStock()+"L");
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "109", "AFS":
                        FuelQueue.add_fuel_stock();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "110", "IFQ":
                        FuelQueue.DisplayFuelIncome();
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "111", "GUI":
                        System.out.println("Opening Fuel Station GUI");
                        FuelStationApplication.main(args);
                        System.out.println("----------------------------------------
--------\n");
                        break;
                    case "999", "EXT":
                        replay='n';
                        System.out.println("Exit from the program.");
                        break;
                    default:
                        System.out.println("invalid input..pls try again.");
                        continue;

                }
                break;
            }
        }

    }
}
```

3.3.2 FuelQueue.java

```
package com.example.w1912792_task04;

import java.io.*;
import java.util.Scanner;

public class FuelQueue extends passenger implements Serializable{
    //class attributes
    private static int fuelStock=6600;
    private static int [] FuelIncome=new int[5];
    private static FuelQueue fuelQueue[][]=new FuelQueue[5][6];

    //class methods
```

```
    //constructor
    public FuelQueue(String fName, String sName, String vehicalNo, int liters){
        super(fName,sName,vehicalNo,liters);
    }

    //FuelQueue array getter
    public static FuelQueue[][] getFuelQueue() {
        return fuelQueue;
    }

    //100 or VFQ: View all Fuel Queues
    public static void view_queues(){
        FuelQueue queue[][]=getFuelQueue();
        FuelQueue PassesngerQueue;
        for (int i=0;i<queue.length;i++){
            for (int j=0;j< queue[i].length;j++){
                PassesngerQueue=queue[i][j];
                if ((queue[i][j])==null){
                    System.out.println("queue "+(i+1)+": place "+(j+1)+" -> Empty");
                }else {
                    System.out.println("queue "+(i+1)+": place "+(j+1)+" ->
"+PassesngerQueue.getFirstName());
                }
            }
            System.out.println("\n");
        }

    }

    //101 or VEQ: View all Empty Queues
    public static void view_empty_queues(){
        FuelQueue queue[][]=getFuelQueue();
        for (int row=0;row<queue.length;row++){
            System.out.println("\nQueue No: "+(row+1));
            for (int col=0;col<queue[row].length;col++){
                if (queue[row][col]==null){
                    System.out.println("Queue no, "+(row+1)+" location no, "+(col+1)+"
--> Empty");
                }
            }

        }
    }

    //102 or ACQ: Add customer to a Queue.
    public static void add_customer(){
        FuelQueue queue[][]=getFuelQueue();
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter first name:");
        String Fname = sc.next().toLowerCase();
        System.out.print("Enter second name:");
        String Sname = sc.next().toLowerCase();
        System.out.print("Enter vehicle number:");
        String VehicleNo = sc.next();
        int liters;
        while (true){
            try {
                System.out.print("Enter liters required:");
                liters = sc.nextInt();
                break;
            }catch (Exception e){
                System.out.println("Invalid input. Enter a number..");
```

```
                sc.nextLine(); //clear input
                continue;
            }

        }

        int num=ShortestQueue(queue);//check the shortest queue

        for (int col=0;col<queue[num].length;col++) {
            int lastVal = queue[num].length - 1;
            if (queue[num][lastVal]==null){
                if (queue[num][col] == null) {
                    queue[num][col] =new FuelQueue(Fname,Sname,VehicleNo,liters);
                    fuelStock -= liters;
                    setFuelIncome(num,liters);
                    break;
                }
            }else {
                System.out.println("Queue "+num+" is full now. check another queue or
try again later.");
                System.out.println("You will be added to a waiting list.");
                WaitingList.enqueue(Fname,Sname,VehicleNo,liters);
                break;
            }
        }
    }

    //Fuel income array
    public static void setFuelIncome(int index,int liters) {
        int income=liters*430;
        FuelIncome[index]+=income;
    }

    //108 STK - fuel stock Getter
    public static int getFuelStock() {
        return fuelStock;
    }

    //find the shortest queue
    public static int ShortestQueue(FuelQueue [][] queue){
        int shortestIndex=0;
        int NullCount=0;
        int temp=0;
        for (int row=0;row<queue.length;row++){
            for (int col=0;col<queue[row].length;col++){
                if (queue[row][col]==null){
                    NullCount+=1;
                }
            }
            if (NullCount>temp){
                System.out.println("short changed to "+row);
                shortestIndex=row;
            }
            temp=NullCount;
            NullCount=0;

        }
        return shortestIndex;
    }

    //103 or RCQ: Remove a customer from a Queue
    public static void remove_customer(){
        FuelQueue queue[][]=getFuelQueue();
```

```
        Scanner sc=new Scanner(System.in);
        int queue_no;
        int location;
        while (true){
            try {
                System.out.print("Enter queue number:");
                queue_no=sc.nextInt();
                if (queue_no<1 || queue_no>5){
                    System.out.println("Enter a valid queue number.");
                    continue;
                }
                System.out.print("Enter location:");
                location=sc.nextInt();
                if (location<1 || location>6){
                    System.out.println("Enter a valid queue number.");
                    sc.nextLine();//clear input
                    continue;
                }
                break;
            }catch (Exception e){
                System.out.println("Enter a number not a letter.");
                sc.nextLine();//clear input
                continue;
            }

        }
        passenger Passenger;
        Passenger=queue[queue_no-1][location-1];
        if (queue[queue_no-1][location-1]!=null){
            System.out.println(Passenger.getFirstName()+" removed.");
            FuelQueue.setFuelStock(Passenger.getLitersRequired());

            queue[queue_no-1][location-1]=null;

            queue=customers_moveFront(queue,queue_no); //moving customers to front
        }else {
            System.out.println("location "+location+" is Empty. pls try again with
correct location.");
        }

    }

    //move customer front
    private static FuelQueue[][] customers_moveFront(FuelQueue[][]queue,int queue_no){
        //reordering queue
        for (int i=0;i<queue[queue_no-1].length;i++){
            if (queue[queue_no-1][i]==null){
                for (int k=i+1; k<queue[queue_no-1].length; k++){
                    queue[queue_no-1][k-1] = queue[queue_no-1][k];
                }
                int lastVal=queue[queue_no-1].length-1;
                queue[queue_no-1][lastVal]=null;
                break;
            }
        }return queue;
    }

    //setter for fuel stock
    public static void setFuelStock(int liters) {
        FuelQueue.fuelStock += liters;
    }

    //104 or PCQ: Remove a served customer.
```

```
    public static void remove_served_customer(){
        FuelQueue queue[][]=getFuelQueue();
        boolean isNameEqual=false;
        int col;
        int row;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter name: ");
        String customer_name=sc.nextLine();
        try {
            for (row=0;row< queue.length;row++){
                for ( col=0;col< queue[row].length;col++){
                    if (queue[row][col]!=null &&
((queue[row][col].getFirstName()).equalsIgnoreCase(customer_name))){
                        System.out.println(queue[row][col].getFirstName()+" removed");
                        queue[row][col]=null;
                        isNameEqual=true;
                        break;
                    }
                }
                if (isNameEqual==true){
                    break;
                }
            }customers_moveFront(queue,row+1);
            int lastVal=queue[row].length-1;
            queue[row][lastVal]=WaitingList.dequeue();
            System.out.println(queue[row][lastVal].getFirstName()+" has been added to
Fuel queue "+(row+1)+", from waiting queue");

        }catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Entered name can't be found in fuel queue..");
        }

    }

    //105 or VCS: View Customers Sorted in alphabetical order
    public static void sort_queue(){
        FuelQueue queue[][]=getFuelQueue();
        FuelQueue[][] copyArr=new FuelQueue[5][6]; //copying queue to a new array
        //copying queue to new array
        for (int row=0;row< queue.length;row++){
            for (int col=0;col< queue[row].length;col++){
                copyArr[row][col]=queue[row][col];
            }
        }

        for (int i = 0; i < copyArr.length; i++) {
            for (int j = 0; j < copyArr[i].length; j++) {
                for (int k = 0; k < copyArr[i].length - j - 1; k++) {
                    try {
                        if (copyArr[i][k].getFirstName().compareTo(copyArr[i][k +
1].getFirstName()) >0) {
                            // swapping elements
                            FuelQueue t = copyArr[i][k];
                            copyArr[i][k] = copyArr[i][k + 1];
                            copyArr[i][k + 1] = t;
                        }
                    }catch (NullPointerException n){
                        continue;
                    }

                }
            }
        }
```

```
        // printing the sorted queue
        for (int i = 0; i < copyArr.length; i++) {
            System.out.print("Queue "+(i+1)+" --> ");
            for (int j = 0; j < copyArr[i].length; j++)
                if (copyArr[i][j]!=null){
                    System.out.print(copyArr[i][j].getFirstName() + ", ");
                }

            System.out.println();
        }
    }


//-----------------------------------------------------------------
    //106 or SPD: Store Program Data into file.
    public static void store_data_toFile(){
        FuelQueue queue[][]=getFuelQueue();
        try {
            FileWriter data_file= new FileWriter("data.txt");
            for (int row=0;row<queue.length;row++){
                data_file.write("Queue "+(row+1)+"\n");
                for (int col=0;col<queue[row].length;col++){
                    data_file.write("Queue no, "+(row+1)+" location "+(col+1)+":\n");
                    if (queue[row][col]!=null){
                        data_file.write("\tFirst Name:
"+queue[row][col].getFirstName()+"\n\tSecond Name:
"+queue[row][col].getSecondName()+"\n\tVehicle No:
"+queue[row][col].getVehicleNo()+"\n\tLiters requied:
"+queue[row][col].getLitersRequired()+"\n");
                    }else {
                        data_file.write("\tEmpty space\n");
                    }
                }
                data_file.write("\n");
            }
            data_file.close();
            System.out.println("wrote to file..");
        }catch (IOException e){
            System.out.println("error occurred..");
        }
    }

    //107 or LPD: Load Program Data from file.
    public static void load_data_fromFile(){
        try {
            File dataFile=new File("data.txt");
            Scanner readFile=new Scanner(dataFile);
            String fileLine;
            while (readFile.hasNext()){
                fileLine=readFile.nextLine();
                System.out.println(fileLine);
            }
        }catch (IOException e){
            System.out.println("Error");
        }
    }


    //for 109 or AFS: Add Fuel Stock.
    public static void add_fuel_stock() {
        Scanner sc = new Scanner(System.in);
        int new_fuel_count;
```

```
        while (true) {
            try {
                System.out.print("Enter fuel amount to add: ");
                new_fuel_count = sc.nextInt();
                if (new_fuel_count < 0) {
                    System.out.println("invalid, should be grater than 0.");
                    continue;
                }
                break;
            } catch (Exception e) {
                System.out.println("Enter a valid amount.");
                sc.nextLine();//clear input
                continue;
            }
        }
        setFuelStock(new_fuel_count);
    }

    //110 or IFQ: display fuel income
    public static void DisplayFuelIncome(){
        for(int i=0;i<FuelIncome.length;i++){
            System.out.println("Fuel queue "+(i+1)+" income: "+FuelIncome[i]);
        }
    }

    //fuel income array Getter
    public static int[] getFuelIncome() {
        return FuelIncome;
    }
}
```

3.3.3 passenger.java

```
package com.example.w1912792_task04;

public class passenger {
    //class attributes
    private String FirstName;
    private String SecondName;
    private String VehicleNo;
    private int LitersRequired;
    private static int passengerCount=0;

    //class methods
    public passenger(String fName, String sName, String vehicleNo, int liters){
        this.FirstName=fName;
        this.SecondName=sName;
        this.VehicleNo =vehicleNo;
        this.LitersRequired=liters;
        passengerCount++;
    }

    //getters
    public String getFirstName() {
        return FirstName;
    }

    public String getSecondName() {
        return SecondName;
    }
```

```
    public String getVehicleNo() {
        return VehicleNo;
    }

    public int getLitersRequired() {
        return LitersRequired;
    }

}
```

3.3.4 WaitingList.java

```
package com.example.w1912792_task04;
/*
referred from below sources:
https://www.programiz.com/dsa/circular-queue
https://github.com/im-mani-teckieshare/data-structure-algorithm-java/blob/main/data-
structure/src/circularqueue/CircularQueue.java
*/

public class WaitingList {
    //class attributes
    private static int capacity=6;
    private static int front = 0;
    private static int rear = -1;
    private static int size = 0;
    private static FuelQueue waiting_list[] = new FuelQueue[capacity];

    //class methods
    public WaitingList(){

    }

    public static void enqueue(String Fname,String Sname, String VehicleNo,int liters)
{
        if(isFull()) {
            System.out.println("Waiting Queue is full can't insert");
            return;
        }
        rear = (rear + 1) % capacity;
        waiting_list[rear] = new FuelQueue(Fname,Sname,VehicleNo,liters);//add
passenger to waiting list
        System.out.println(Fname+" has been added to waiting queue.");
        size++;
    }

    public static FuelQueue dequeue() {
        if(isEmpty()) {
            System.out.println("Queue is empty");
            return null;
        }
        FuelQueue data = waiting_list[front];
        front = (front + 1) % capacity;
        size--;
        return data;//return the first element of waiting queue
    }

    public static boolean isFull() {
        return size == capacity;
```

```
    }

    public static boolean isEmpty() {
        return size == 0;
    }

    //display waiting queue: for javafx
    public static String DisplayWaitingQueue(){
        String WaitingQueueDetails="";
        for (int i=0;i< waiting_list.length;i++){
            if (waiting_list[i]!=null) {
                WaitingQueueDetails += "\n" + (i + 1) + ")";
                WaitingQueueDetails += ("\n\tFirst name: " +
waiting_list[i].getFirstName());
                WaitingQueueDetails += ("\n\tSecond name: " +
waiting_list[i].getSecondName());
                WaitingQueueDetails += ("\n\tVehicle No: " +
waiting_list[i].getVehicleNo());
                WaitingQueueDetails += ("\n\tLiters required: " +
waiting_list[i].getLitersRequired() + "\n");
            }
        }
        return WaitingQueueDetails;
    }

    public static FuelQueue[] getWaiting_list() {
        return waiting_list;
    }
}
```

3.3.5 FuelStationApplication.java (fxml application)

```
package com.example.w1912792_task04;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class FuelStationApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(FuelStationApplication.class.getResource("HomePage.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 600, 400);
        stage.setTitle("Fuel Management System");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

3.3.6 task04_controller (fxml controller)

```
package com.example.w1912792_task04;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

import java.io.IOException;
import java.util.Queue;

public class task04_controller {

    @FXML
    private TextArea ViewFuelQueue;
    @FXML
    private TextArea ViewWaitingQueue;
    @FXML
    private TextField search_input;
    @FXML
    private TextArea search_results;
    @FXML
    private TextArea FuelPumpIncome;
    @FXML
    private TextField TotalFuelIncome;

    //Scenes switching
    @FXML
    public void switchToHome(ActionEvent event) throws IOException {
        //identify the previous stage and close it
        Parent root=FXMLLoader.load(getClass().getResource("HomePage.fxml"));
        Stage currentWindow=(Stage)((Node)event.getSource()).getScene().getWindow();
        Scene newScene = new Scene(root,600,400);
        currentWindow.setScene(newScene);
        currentWindow.show();
    }
    public void switchToViewQueues(ActionEvent event)throws IOException{
        Parent root=FXMLLoader.load(getClass().getResource("ViewAllQueues.fxml"));
        Stage currentWindow=(Stage)((Node)event.getSource()).getScene().getWindow();
        Scene newScene = new Scene(root,600,400);
        currentWindow.setScene(newScene);
        currentWindow.show();
    }
    public void switchToSearch(ActionEvent event)throws IOException{
        Parent root=FXMLLoader.load(getClass().getResource("SearchPassengers.fxml"));
        Stage currentWindow=(Stage)((Node)event.getSource()).getScene().getWindow();
        Scene newScene = new Scene(root,600,400);
        currentWindow.setScene(newScene);
        currentWindow.show();
    }
    public void switchToFuelIncome(ActionEvent event)throws IOException{
        Parent root=FXMLLoader.load(getClass().getResource("FuelIncome.fxml"));
        Stage currentWindow=(Stage)((Node)event.getSource()).getScene().getWindow();
        Scene newScene = new Scene(root,600,400);
        currentWindow.setScene(newScene);
        currentWindow.show();
    }
```

```
    @FXML
    //Displaying Fuel Queue
    protected void FuelViewBtn_onClick() {
        FuelQueue Queue[][] = FuelQueue.getFuelQueue();
        String FuelQueueDetails="";
        for (int row=0;row<Queue.length;row++){
            FuelQueueDetails+="Queue No: "+(row+1)+"\n";
            for (int col=0;col<Queue[row].length;col++){
                FuelQueueDetails+=(col+1)+") Queue no, "+(row+1)+" location
"+(col+1)+":\n";
                if (Queue[row][col]!=null){
                    FuelQueueDetails+=("\tFirst Name:
"+Queue[row][col].getFirstName()+"\n\tSecond Name:
"+Queue[row][col].getSecondName()+"\n\tVehicle No:
"+Queue[row][col].getVehicleNo()+"\n\tLiters requied:
"+Queue[row][col].getLitersRequired()+"\n");
                }else {
                    FuelQueueDetails+=("\tEmpty space\n");
                }
            }FuelQueueDetails+="\n";
        }
        ViewFuelQueue.setText(FuelQueueDetails);
    }

    @FXML
    //display waiting queue
    protected void WaitingViewBtn_onClick(){
        String WaitingQueueDetails="";
        int count=1;
        FuelQueue waitingQueue[]=WaitingList.getWaiting_list();
        System.out.println(waitingQueue);
        if (WaitingList.isEmpty()){//check waiting queue is empty
            WaitingQueueDetails+="\n\tWaiting Queue is empty";
        }else {
            WaitingQueueDetails+=WaitingList.DisplayWaitingQueue();
        }

        ViewWaitingQueue.setText(WaitingQueueDetails);
    }


    @FXML
    //get search value and display results
    protected void SearchPassenger(){
        String input_name=search_input.getText();
        FuelQueue queue[][] = FuelQueue.getFuelQueue();
        FuelQueue waiting[]=WaitingList.getWaiting_list();
        boolean isNameEqual=false;
        String output_text="";

            for (int row=0;row< queue.length;row++){
                for (int col=0;col< queue[row].length;col++){
                    if (queue[row][col]!=null &&
((queue[row][col].getFirstName()).equalsIgnoreCase(input_name))){
                        output_text+=queue[row][col].getFirstName()+"'s Details:";
                        output_text+="\n\tFirst Name:
"+queue[row][col].getFirstName();
                        output_text+="\n\tSecond Name:
"+queue[row][col].getSecondName();
                        output_text+="\n\tVehicle No:
"+queue[row][col].getVehicleNo();
                        output_text+="\n\tLiters Required:
```

```
"+queue[row][col].getLitersRequired();
                        isNameEqual=true;
                        break;
                    }
                }
                if (isNameEqual==true){
                    break;
                }
            }
            for (int i=0;i< waiting.length;i++){
                if (waiting[i]!=null &&
((waiting[i].getFirstName()).equalsIgnoreCase(input_name))){
                    output_text+=waiting[i].getFirstName()+"'s Details:";
                    output_text+="\n\tFirst Name: "+waiting[i].getFirstName();
                    output_text+="\n\tSecond Name: "+waiting[i].getSecondName();
                    output_text+="\n\tVehicle No: "+waiting[i].getVehicleNo();
                    output_text+="\n\tLiters Required:
"+waiting[i].getLitersRequired();
                    isNameEqual=true;
                    break;
                }
            }
            if (isNameEqual==false){
                output_text+=("Entered name can't be found in fuel queue");
            }

        search_results.setText(output_text);
    }

    //display fuel income
    @FXML
    protected void DisplayFuelIncome(){
        String DisplayPumpIncome="";
        int TotalIncome=0;
        int[] IncomeList=FuelQueue.getFuelIncome();
        for (int i=0;i<IncomeList.length;i++){
            DisplayPumpIncome+="Fuel Pump No "+(i+1)+" Income-> "+IncomeList[i]+"\n";
            TotalIncome+=IncomeList[i];
        }
        FuelPumpIncome.setText(DisplayPumpIncome);
        TotalFuelIncome.setText("Total income is, "+TotalIncome);
    }
}
```

3.3.7 FXML and CSS files

- HomePage.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="400.0" prefWidth="600.0" styleClass="background"
stylesheets="@style.css" xmlns="http://javafx.com/javafx/18"
```

```
   xmlns:fx="http://javafx.com/fxml/1"
   fx:controller="com.example.w1912792_task04.task04_controller">
      <children>
         <Label layoutX="95.0" layoutY="57.0" styleClass="Home_Title"
   stylesheets="@style.css" text="Fuel Management System " textAlignment="CENTER">
            <font>
               <Font size="28.0" />
            </font>
         </Label>
         <VBox layoutX="211.0" layoutY="129.0" prefHeight="192.0"
   prefWidth="178.0" style="-fx-alignment: center; -fx-spacing: 40px;">
            <children>
               <Button mnemonicParsing="false" onAction="#switchToViewQueues"
   styleClass="Home_Btns" stylesheets="@style.css" text="View Queues"
   textFill="#2e2b2b">
                  <font>
                     <Font name="Arial" size="12.0" />
                  </font>
               </Button>
               <Button mnemonicParsing="false" onAction="#switchToSearch"
   styleClass="Home_Btns" stylesheets="@style.css" text="Search Passengers"
   textAlignment="CENTER" />
               <Button mnemonicParsing="false" onAction="#switchToFuelIncome"
   styleClass="Home_Btns" stylesheets="@style.css" text="Fuel income" />
            </children>
         </VBox>
      </children>
   </AnchorPane>
```

- ViewAllQueues.fxml

```
   <?xml version="1.0" encoding="UTF-8"?>

   <?import javafx.scene.control.Button?>
   <?import javafx.scene.control.Label?>
   <?import javafx.scene.control.Tab?>
   <?import javafx.scene.control.TabPane?>
   <?import javafx.scene.control.TextArea?>
   <?import javafx.scene.layout.AnchorPane?>
   <?import javafx.scene.text.Font?>

   <AnchorPane prefHeight="400.0" prefWidth="600.0" styleClass="background"
   stylesheets="@style.css" xmlns="http://javafx.com/javafx/18"
   xmlns:fx="http://javafx.com/fxml/1"
   fx:controller="com.example.w1912792_task04.task04_controller">
      <children>
         <TabPane layoutY="47.0" prefHeight="352.0" prefWidth="600.0"
   tabClosingPolicy="UNAVAILABLE">
           <tabs>
             <Tab text="Fuel Queue">
               <content>
                 <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0"
   prefWidth="200.0">
                        <children>
                           <Button layoutX="14.0" layoutY="9.0"
   mnemonicParsing="false" onAction="#FuelViewBtn_onClick" styleClass="View_Btn"
   stylesheets="@style.css" text="View" />
                           <TextArea fx:id="ViewFuelQueue" layoutX="17.0"
   layoutY="47.0" prefHeight="266.0" prefWidth="323.0" />
                           <Button layoutX="247.0" layoutY="9.0"
```

```
   mnemonicParsing="false" onAction="#switchToHome" styleClass="GoBack_Btn"
   stylesheets="@style.css" text="Go Back" />
                        </children></AnchorPane>
                </content>
              </Tab>
              <Tab text="Waiting Queue">
                <content>
                  <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0"
   prefWidth="200.0">
                        <children>
                           <Button layoutX="14.0" layoutY="14.0"
   mnemonicParsing="false" onAction="#WaitingViewBtn_onClick"
   styleClass="View_Btn" stylesheets="@style.css" text="View" />
                           <TextArea fx:id="ViewWaitingQueue" layoutX="17.0"
   layoutY="56.0" prefHeight="229.0" prefWidth="258.0" />
                           <Button layoutX="183.0" layoutY="14.0"
   mnemonicParsing="false" onAction="#switchToHome" styleClass="GoBack_Btn"
   stylesheets="@style.css" text="Go Back" />
                        </children></AnchorPane>
                </content>
              </Tab>
            </tabs>
          </TabPane>
          <Label layoutX="205.0" layoutY="7.0" styleClass="Page_Title"
   stylesheets="@style.css" text="View All Queues">
             <font>
                <Font size="20.0" />
             </font>
          </Label>
       </children>
   </AnchorPane>
```

- SearchPassengers.fxml

```
   <?xml version="1.0" encoding="UTF-8"?>

   <?import javafx.scene.control.Button?>
   <?import javafx.scene.control.Label?>
   <?import javafx.scene.control.TextArea?>
   <?import javafx.scene.control.TextField?>
   <?import javafx.scene.layout.AnchorPane?>
   <?import javafx.scene.text.Font?>

   <AnchorPane prefHeight="400.0" prefWidth="600.0" styleClass="background"
   stylesheets="@style.css" xmlns="http://javafx.com/javafx/18"
   xmlns:fx="http://javafx.com/fxml/1"
   fx:controller="com.example.w1912792_task04.task04_controller">
      <children>
         <TextField fx:id="search_input" layoutX="172.0" layoutY="65.0"
   promptText="Enter Passenger's name" />
         <Button layoutX="353.0" layoutY="61.0" mnemonicParsing="false"
   onAction="#SearchPassenger" styleClass="View_Btn" stylesheets="@style.css"
   text="View" />
         <TextArea fx:id="search_results" layoutX="172.0" layoutY="120.0"
   prefHeight="200.0" prefWidth="255.0" />
         <Button layoutX="335.0" layoutY="339.0" mnemonicParsing="false"
   onAction="#switchToHome" styleClass="GoBack_Btn" stylesheets="@style.css"
   text="Go Back" />
         <Label layoutX="164.0" layoutY="14.0" styleClass="Page_Title"
   stylesheets="@style.css" text="Search Passengers Details">
            <font>
               <Font size="24.0" />
```

```
            </font>
         </Label>
      </children>
   </AnchorPane>
```

- FuelIncome.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="400.0" prefWidth="600.0" styleClass="background"
stylesheets="@style.css" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.w1912792_task04.task04_controller">
   <children>
      <TextField fx:id="TotalFuelIncome" layoutX="225.0" layoutY="299.0" />
      <Label layoutX="227.0" layoutY="271.0" text="Total Fuel income:">
         <font>
            <Font size="18.0" />
         </font></Label>
      <TextArea fx:id="FuelPumpIncome" layoutX="116.0" layoutY="120.0"
prefHeight="114.0" prefWidth="396.0" />
      <Button layoutX="116.0" layoutY="77.0" mnemonicParsing="false"
onAction="#DisplayFuelIncome" styleClass="View_Btn" stylesheets="@style.css"
text="View" />
      <Button layoutX="419.0" layoutY="77.0" mnemonicParsing="false"
onAction="#switchToHome" styleClass="GoBack_Btn" stylesheets="@style.css"
text="Go Back" />
      <Label layoutX="233.0" layoutY="26.0" styleClass="Page_Title"
stylesheets="@style.css" text="Fuel Income" />
   </children>
</AnchorPane>
```

- Style.css

```
/*common styles*/
.background{
-fx-background-color:linear-gradient(to right, #2193b0, #6dd5ed);
}
.View_Btn{
-fx-background-color:#134261;
-fx-background-radius:25px;
-fx-text-fill: #fff;
-fx-cursor: hand;
-fx-padding:8px 25px;
}
.GoBack_Btn{
-fx-background-color:#2b1887;
-fx-background-radius:25px;
-fx-text-fill: #fff;
```

```
    -fx-cursor: hand;
    -fx-padding:8px 25px;
    }
    .View_Btn:hover{
    -fx-background-color:#eb1887;
    }
    .GoBack_Btn:hover{
    -fx-background-color:#eb1887;
    }
    .Page_Title{
    -fx-font-size:25px;
    -fx-font-weight:bold;
    }


    /*Home Page style*/
    .Home_Btns{
    -fx-background-color: #89f0ae;
    -fx-border-radius:50%;
    -fx-scale-x:1.5;
    -fx-scale-y:1.5;
    -fx-font-weight:bold;
    -fx-cursor: hand;
    }
    .Home_Title{
    -fx-font-size:35px;
    }
```