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	: Mr. Deshan Sumanathilaka
Date of submission	: 07/08/2022
Student ID	: 20211267 / w1899317
Student First Name	: Sasiru
Student Surname	: Perera

"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 : K. Mareen Sasiru Vishmika Perera

Student ID : 20211267

Test Cases

(1 – 8 Test Cases are for Array Version)

	Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
1	Fuel Queue Initialized Correctly After program starts, 100 or VFQ	Displays 'empty' for all queues.	Displays 'empty' for all Queues.	Pass
2	Add passenger "Jane" to Queue 2 102 or ACQ Enter Queue: 2 Enter Name: Jane	Display 'Jane added to the queue 2 successfully"	Display 'Jane added to the queue 2 successfully"	Pass
3	Add passenger "John" to Queue 3 102 or ACQ Enter Queue: 3 Enter Name: John	Display 'John added to the queue 3 successfully"	Display 'John added to the queue 3 successfully"	Pass
4	View all empty queue 101 or VEQ	Display all Queue available for passenger input	Display all Queue available for passenger input	Pass
5	Remove Customer from a queue 103 or RCQ Enter Queue: 2 Enter position: 1	Display Customer is removed (Jane will be removed)	Display Customer is removed (Jane will be removed)	Pass
6	Remove a Served customer 104 or RCQ Enter Queue: 3	Served Customer is removed (John as first customer will be removed)	Served Customer is removed	Pass
7	View remaining Fuel stock 108 or STK	6590L	6590L	Pass
8	Add fuel Stock 109 or AFS Enter new stock: 100	Stock updated (6690)	Stock updated	Pass

9	Add Customer data 102 or ACQ Enter Customer first name: Nimal Enter Customer Last name: Kumara Enter vehicle No: 2009 Enter stock need: 10	Nimal added Queue1	Nimal added Queue1	Pass
10	Add Customer data 102 or ACQ Enter Customer first name: Akash Enter Customer Last name: Nimantha Enter vehicle No: 2010 Enter stock need: 20	Akash added Queue2	Akash added Queue2	Pass
11	Add Customer data 102 or ACQ Enter Customer first name: Namal Enter Customer Last name: Perera Enter vehicle No: 3000 Enter stock need: 20	Namal added Queue3	Namal added Queue3	Pass
12	Add Customer data 102 or ACQ Enter Customer first name: Amila Enter Customer Last name: Kasun Enter vehicle No: 8787 Enter stock need: 10	Amila added Queue4	Amila added Queue4	Pass
13	Add Customer data 102 or ACQ Enter Customer first name: Shenal Enter Customer Last name: Rashen Enter vehicle No: 2003 Enter stock need: 100	Shenal added Queue5	shenal added Queue5	Pass

14	Add Customer data 102 or ACQ Enter Customer first name: Anna Enter Customer Last name: Perera Enter vehicle No: 2020 Enter stock need: 10	Anna added Queue1	Anna added Queue1	Pass
15	View remaining Fuel Stock 108	6430.0L	6430.0L	Pass
16	Select operation: eee	Invalid operation	Invalid operation	Pass
17	Save Data to a file 106 or SPD	Data Saved	Data Saved	Pass
18	Sort Data 105 or VCS	---------- Customer sorted A - Z by name ---------- Queue 1 -> Anna Perera Nimal Kumara Queue 2 -> Akash Nimantha Queue 3 -> Namal Perera Queue 4 -> Amila Kasun Queue 5 -> Shenal Rashen	---------- Customer sorted A - Z by name ---------- Queue 1 -> Anna Perera Nimal Kumara Queue 2 -> Akash Nimantha Queue 3 -> Namal Perera Queue 4 -> Amila Kasun Queue 5 -> Shenal Rashen	Pass
20	Exit program 999 or EXT	Program terminated	Program terminated	Pass

21	Load program data from a file 107 or LPD	Data loaded successfully	Data loaded successfully	Pass
22	Income of each Fuel Queue 110 or IFQ	Income Queue 1 : Rs 8600.0 Income Queue 2 : Rs 8600.0 Income Queue 3 : Rs 8600.0 Income Queue 4 : Rs 4300.0 Income Queue 5 : Rs 43000.0	Income Queue 1 : Rs 8600.0 Income Queue 2 : Rs 8600.0 Income Queue 3 : Rs 8600.0 Income Queue 4 : Rs 4300.0 Income Queue 5 : Rs 43000.0	Pass
23	Add fuel Stock 109 or AFS Enter new stock: rrr	Invalid Fuel Stock	Invalid Fuel Stock	Pass
24	Add Customer data 102 or ACQ Enter Customer first name: Kalpa Enter Customer Last name: Perera Enter vehicle No: 2010 Enter stock need: -89	Invalid Stock count	Invalid Stock count	Pass
25	View Fuel Stock low when hit the Limit 500L	Fuel Stock Low	Fuel Stock Low	Pass
26	Remove Customer from a location 103 Enter Queue: 1 Enter Position: 2	Customer is removed from queue 1(Anna)	Customer is removed from queue 1(Anna)	Pass

Save the Data to file, change max customer in each queue to 1, Rerun the program, Load Data

26	Add Customer to waiting Queue after all Queue are full 102 or ACQ Enter Customer first name: Kalpa Enter Customer Last name: Perera Enter vehicle No: 2050 Enter stock need: 100	Customer added to waiting Queue	Customer added to waiting Queue	Pass
27	Remove a served Customer with Waiting Queue implementation 104 or RCQ Select the queue: 2	Customer removed and first customer in Waiting queue add to the relevant queue (2) as last customer	Customer removed and first customer in Waiting queue add to the relevant queue (2) as last customer	Pass

Discussion

I have created total 27 test cases. First 8 test cases for Array Version and all the other test cases are design for Class Version and Class Version with Waiting queue implementation. In Array Version I have inserted number of usernames and showed how it can be removed from a specific location or as the first served customer. In Class Version however I have added 6 customers to queues with their full details First name, Last name, Vehicle number and Stock they need as it shows when user insert data it will create the given passenger objects correctly and it always select the queue with the minimum length, and I have checked the remaining fuel stock value after to show stock count is calculated correctly. Some test cases are defined to show the input validation as if user entered alphabetical characters or any value smaller than 0 as stock value program will let you know the given value is invalid for Stock. Also, I have added test cases to show writing data to the file, reading and loading data back from the file and sorting the given data along with how it calculates income for each fuel queue. To check for the low stock count warning, I have changed the stock variable value to 550L as it will soon run out and display the warning to the screen, also to see that Waiting queue works correctly I have change max number of customers per queue for 1 as after 5 customer input for 6^{th} customer it will go for the waiting Queue and when served customer is removed from any one of the 5 queue the first customer in waiting queue will be moved to the relevant queue at last position.

I did not attempt the Task 4 Part of the assessment.

Code:

Task 1

FuelQueueArrayV.java

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

//Saving data will overwrite the current saved data.

public class FuelQueueArrayV {

    private static int stock = 6600;
    private static final int LIMIT = 500;

    public static void main(String[] args) {
        System.out.println(".......Please Select a operation from the below
menu.......");
        System.out.println("""

                100 or VFQ: View all Fuel Queues.
                101 or VEQ: View all Empty Queues.
                102 or ACQ: Add customer to a Queue.
                103 or RCQ: Remove a customer from a Queue. (From a specific
location)
                104 or PCQ: Remove a served customer.
                105 or VCS: View Customers Sorted in alphabetical order.
                106 or SPD: Store Program Data into file.
                107 or LPD: Load Program Data from file.
                108 or STK: View Remaining Fuel Stock.
                109 or AFS: Add Fuel Stock.
                999 or EXT: Exit the Program.

                """);

        Scanner input = new Scanner(System.in);
        //initialize arrays
        String[] queue1 = {};
        String[] queue2 = {};
        String[] queue3 = {};

        loop1:
        do {
            System.out.print("Select operation : ");
            String choice = input.nextLine();
            switch (choice) {
                case "100", "VFQ", "vfq" -> {
                    viewQueue(queue1, 1);
                    viewQueue(queue2, 2);
                    viewQueue(queue3, 3);
                }
                case "101", "VEQ", "veq" -> {
```

```
                    System.out.println("Available queue for customer input :
- ");
                    boolean emptyQueue1 = isArrayEmpty(queue1);
                    boolean emptyQueue2 = isArrayEmpty(queue2);
                    boolean emptyQueue3 = isArrayEmpty(queue3);
                    if (emptyQueue1) {
                        System.out.println("Queue 1");
                    }
                    if (emptyQueue2) {
                        System.out.println("Queue 2");
                    }
                    if (emptyQueue3) {
                        System.out.println("Queue 3");
                    }
                    if ((!emptyQueue1) & (!emptyQueue2) & (!emptyQueue3)) {
                        System.out.println("Non of the queue are available");
                    }
                }
                case "102", "ACQ", "acq" -> {
                    int qNum = inputValidation("Enter queue number : ");

                    if (qNum == 1) {
                        queue1 = addCustomer(queue1, qNum);
                    } else if (qNum == 2) {
                        queue2 = addCustomer(queue2, qNum);
                    } else {
                        queue3 = addCustomer(queue3, qNum);
                    }
                }
                case "103", "RCQ", "rcq" -> {

                    // When Customer removed from a specific location it will
take as he is leaving the queue and fuel stock will restore to the value
before he was assigned.

                    int qNum = inputValidation("Enter which queue you want to
remove the customer from : ");

                    if (qNum == 1) {
                        queue1 = removeCustomer(queue1, qNum, "rcq");
                    } else if (qNum == 2) {
                        queue2 = removeCustomer(queue2, qNum, "rcq");
                    } else {
                        queue3 = removeCustomer(queue3, qNum, "rcq");
                    }
                }
                case "104", "PCQ", "pcq" -> {
                    int qNum = inputValidation("Enter which queue have the
served customer : ");

                    if (qNum == 1) {
                        queue1 = removeCustomer(queue1, qNum, "pcq");
                    } else if (qNum == 2) {
                        queue2 = removeCustomer(queue2, qNum, "pcq");
                    } else {
                        queue3 = removeCustomer(queue3, qNum, "pcq");
                    }
```

```
                }
                case "105", "VCS", "vcs" -> {
                    System.out.println("Queue 1 A-Z Sorted :
"+Arrays.toString(sortArray(queue1,queue1.length)));
                    System.out.println("Queue 2 A-Z Sorted :
"+Arrays.toString(sortArray(queue2,queue2.length)));
                    System.out.println("Queue 3 A-Z Sorted :
"+Arrays.toString(sortArray(queue3,queue3.length)));

                }
                case "106", "SPD", "spd" -> {
                    try {
                        FileWriter myWriter = new FileWriter("Data.txt");
                        myWriter.write("Queue 1 : " + Arrays.toString(queue1)
+ "\n");
                        myWriter.write("Queue 2 : " + Arrays.toString(queue2)
+ "\n");
                        myWriter.write("Queue 3 : " + Arrays.toString(queue3)
+ "\n");
                        myWriter.write("Remaining fuel Stock : " + stock +
"\n");
                        myWriter.close();
                        System.out.println("Successfully written to the
file");
                    } catch (IOException e) {
                        System.out.println("Error when writing to the file");
                    }
                }
                case "107", "LPD", "lpd" -> {
                    try {
                        File inputFile = new File("Data.txt");
                        Scanner readFile = new Scanner(inputFile);
                        int lineNumber = 1;
                        while (readFile.hasNextLine()) {
                            String data = readFile.nextLine();
                            data = data.substring(10).replace('[', '
').replace(']', ' ').strip();
                            data = data.replace(" ", "");
                            if (lineNumber == 1) {
                                System.out.println("Queue 1 : " + data);
                                queue1 = data.split(",");
                            } else if (lineNumber == 2) {
                                System.out.println("Queue 2 : " + data);
                                queue2 = data.split(",");
                            } else if (lineNumber == 3) {
                                System.out.println("Queue 3 : " + data);
                                queue3 = data.split(",");
                            } else {
                                System.out.println(data);
                                stock = Integer.parseInt(data.substring(10));
                            }
                            lineNumber++;
                        }
                        System.out.println("File Data loaded successfully
:)");
                        readFile.close();
                    } catch (Exception e) {
```

```
                        System.out.println("Error when reading the file");
                    }
                }
                case "108", "STK", "stk" -> {
                    System.out.println("Remaining fuel " + stock + "L");
                }
                case "109", "AFS", "afs" -> {
                    stock = addingFuelStock(stock);
                    System.out.println("Stock is updated " + stock);
                }
                case "999", "EXT", "ext" -> {
                    System.out.println("Program terminated");
                    break loop1;
                }
                default -> {
                    System.out.println("Invalid Operation try again");
                }
            }
            if (stock <= LIMIT) {
                System.out.printf("Remaining Stock is %d. Please refill !!!
%n", stock);
            }
        } while (true);
    }


    // function for view all fuel queue
    public static void viewQueue(String [] queue,int num){
        boolean value = true;
        System.out.printf("Queue %d : - ",num);
        for (String i : queue){
            if (i != null){
                System.out.print(i+" ");
                value=false;
            }
        }
        if (value){
            System.out.print("empty");
        }
        System.out.println();
    }


    // function for checking an array is empty
    public static boolean isArrayEmpty(String [] queue) {
        return queue.length < 6;
    }


    //function to add customer to queue
    public static String[] addCustomer(String[] queue,int qNum){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter customer name : ");
        String customer = sc.nextLine();
        //Convert first letter to uppercase
        customer =
String.valueOf(customer.charAt(0)).toUpperCase()+customer.substring(1);
```

```
        int n = queue.length;
        int i;
        if (n<6){
            String[] newArr = new String[n+1];
            for (i = 0; i < n; i++) {
                newArr[i] = queue[i];
            }
            newArr[n] = customer;
            System.out.printf("%s added to queue %d
successfully.%n",customer,qNum);
            FuelQueueArrayV.stock = stockCount(FuelQueueArrayV.stock,"acq");
            return newArr;
        }
        System.out.println("Queue is full");
        return queue;
    }



    // removing a customer from a specific location in a queue && removing
the first served customer
    public static String[] removeCustomer(String[] queue, int qNum,String
operation) {
        int index=0;
        int n = queue.length;
        if(operation.equalsIgnoreCase("rcq")) {

            Scanner sc = new Scanner(System.in);
            System.out.printf("Enter the place in the queue %d : ", qNum);
            index = sc.nextInt();
            index -= 1;
            // If the array is empty
            // or the index is not in array range
            // return the original array
            if (index < 0 || index >= n) {
                System.out.println("no customer in that place");
                return queue;
            }

            // Create another array of size one less
            String[] newArray = new String[n - 1];

            // Copy the elements except the index
            // from original array to the other array
            for (int i = 0, k = 0; i < n; i++) {

                // if the index is
                // the removal element index
                if (i == index) {
                    System.out.printf("Customer "+queue[index]+" removed from
the queue %d.%n",qNum);
                    continue;
                }

                newArray[k++] = queue[i];
            }
            //update the stock
```

```
            FuelQueueArrayV.stock = stockCount(FuelQueueArrayV.stock,"rcq");
            // return the resultant array
            return newArray;
        }else{
            // remove first customer
            String[] newArray = new String[n - 1];
            for (int i = 0, k = 0; i < n; i++) {
                if (i == index) {
                    System.out.println("Served Customer is removed.");
                    continue;
                }
                newArray[k++] = queue[i];
            }
            return newArray;
        }
    }


    // add fuel stock
    public static int addingFuelStock(int STOCK){
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.print("Enter additional stock : ");
            try {
                int addStock = sc.nextInt();
                if (addStock<0){
                    System.out.println("invalid");
                    continue;
                }
                return STOCK + addStock;
            } catch (Exception e) {
                System.out.println("Please enter a valid stock count");
            }
        }
    }


    // Calculate stock
    public static int stockCount(int stock,String operation) {
        if (operation.equalsIgnoreCase("acq")) {
            stock -= 10;
        } else {
            stock+=10;
        }
        return stock;
    }


    // Validate user input for queues
    public static int inputValidation(String op){
        Scanner scanner = new Scanner(System.in);

        while (true){
            try{
                System.out.print(op);
                String qNum = scanner.nextLine();
                int Num = Integer.parseInt(qNum);
```

```
                if ((1<=Num)&&(3>=Num)){
                    return Num;
                }else{
                    System.out.println("Queue number you entered out of range
enter a valid Queue number [1,2,3]");
                }
            }catch (Exception e){
                System.out.println("Enter a number 1,2 or 3");
            }
        }
    }



    // Use the bubble sort to sort the array(fuel queue)
    public static String[] sortArray(String[] queue, int length) {
        String temp;

        // Sorting strings using bubble sort
        for (int j = 0; j < length - 1; j++)
        {
            for (int i = j + 1; i < length; i++)
            {
                if (queue[j].compareTo(queue[i]) > 0)
                {
                    temp = queue[j];
                    queue[j] = queue[i];
                    queue[i] = temp;
                }
            }
        }
        return queue;
    }
}

//reference(bubble sort) :- https://www.geeksforgeeks.org/sorting-strings-
using-bubble-sort-2/
```

Task 2

FuelQueueManagementSys.java

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

public class FuelQueueManagementSys {

    public static void main(String[] args) {

        System.out.println(".......Please Select a operation from the below
menu.......");
        System.out.println("""

                100 or VFQ: View all Fuel Queues.
                101 or VEQ: View all Empty Queues.
                102 or ACQ: Add customer to a Queue.
                103 or RCQ: Remove a customer from a Queue. (From a specific
location)
                104 or PCQ: Remove a served customer.
                105 or VCS: View Customers Sorted in alphabetical order.
                106 or SPD: Store Program Data into file.
                107 or LPD: Load Program Data from file.
                108 or STK: View Remaining Fuel Stock.
                109 or AFS: Add Fuel Stock.
                110 or IFQ: Income of each fuel queue.
                999 or EXT: Exit the Program.

                """);

        //array is created to hold 5 FuelQueue objects
        FuelQueue[] refQueue = new FuelQueue[5];
        for(int p=0;p<5;p++){
            refQueue[p] = new FuelQueue();
        }

        Scanner input = new Scanner(System.in);
        loop1:
        do {
            //get the user input for operation
            System.out.print("Select operation : ");
            String choice = input.nextLine();
            switch (choice){
                case "100", "VFQ", "vfq" -> {
                    viewAllQueue(refQueue[0],1);
                    viewAllQueue(refQueue[1],2);
                    viewAllQueue(refQueue[2],3);
```

```
                    viewAllQueue(refQueue[3],4);
                    viewAllQueue(refQueue[4],5);
                }
                case "101", "VEQ", "veq" -> {
                    System.out.print("Fuel queue 1 : ");
                    refQueue[0].emptyQueue();
                    System.out.print("Fuel queue 2 : ");
                    refQueue[1].emptyQueue();
                    System.out.print("Fuel queue 3 : ");
                    refQueue[2].emptyQueue();
                    System.out.print("Fuel queue 4 : ");
                    refQueue[3].emptyQueue();
                    System.out.print("Fuel queue 5 : ");
                    refQueue[4].emptyQueue();

                }
                case "102", "ACQ", "acq" -> {

                    //selecting the queue with the minimum length.
                    ArrayList<Integer> size =new ArrayList<>();
                    size.add(refQueue[0].sizeOfQueue());
                    size.add(refQueue[1].sizeOfQueue());
                    size.add(refQueue[2].sizeOfQueue());
                    size.add(refQueue[3].sizeOfQueue());
                    size.add(refQueue[4].sizeOfQueue());
                    int min = Collections.min(size);

                    //getting user inputs
                    System.out.print("Enter customer first name : ");
                    String fn = input.nextLine();
                    fn =
String.valueOf(fn.charAt(0)).toUpperCase()+fn.substring(1);
                    System.out.print("Enter customer last name : ");
                    String ln = input.nextLine();
                    ln =
String.valueOf(ln.charAt(0)).toUpperCase()+ln.substring(1);
                    System.out.print("Enter vehicle number : ");
                    String vn = input.nextLine();
                    double noLiters;
                    System.out.print("Enter no of liters need : ");
                    try {
                        noLiters = input.nextDouble();
                        input.nextLine();
                        if(noLiters<=0){
                            System.out.println("Invalid");
                            continue;
                        }
                    }catch (Exception e){
                        System.out.println("Invalid input for liters");
                        input.nextLine();
                        continue;
                    }
                    // adding customer object to each of the queue
                    if (refQueue[0].sizeOfQueue() == min){
                        refQueue[0].addCustomer(fn,ln,vn,noLiters,1);
                    } else if (refQueue[1].sizeOfQueue() == min) {
                        refQueue[1].addCustomer(fn,ln,vn,noLiters,2);
```

```
                    } else if (refQueue[2].sizeOfQueue() == min) {
                        refQueue[2].addCustomer(fn,ln,vn,noLiters,3);
                    } else if (refQueue[3].sizeOfQueue() == min) {
                        refQueue[3].addCustomer(fn,ln,vn,noLiters,4);
                    } else {
                        refQueue[4].addCustomer(fn, ln, vn, noLiters,5);
                    }

                }
                case "103", "RCQ", "rcq" -> {
                    int queueNum = inputValidation("Enter which queue you
want to remove the customer from : ");
                    if (queueNum == 1){

refQueue[0].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 2) {

refQueue[1].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 3) {

refQueue[2].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 4) {

refQueue[3].removeCustomer(indexValidation(queueNum));
                    } else {

refQueue[4].removeCustomer(indexValidation(queueNum));
                    }

                }
                case "104", "PCQ", "pcq" -> {
                    int queueNum = inputValidation("Enter which queue you
want to remove the customer from : ");
                    if (queueNum == 1){
                        refQueue[0].removeServed();
                    } else if (queueNum == 2) {
                        refQueue[1].removeServed();
                    } else if (queueNum == 3) {
                        refQueue[2].removeServed();
                    } else if (queueNum == 4) {
                        refQueue[3].removeServed();
                    } else {
                        refQueue[4].removeServed();
                    }

                }
                case "105", "VCS", "vcs" -> {
                    System.out.println("----------- Customer sorted A - Z by
name -----------");
                    refQueue[0].sortAlpha("Queue 1 ->");
                    refQueue[1].sortAlpha("Queue 2 ->");
                    refQueue[2].sortAlpha("Queue 3 ->");
                    refQueue[3].sortAlpha("Queue 4 ->");
                    refQueue[4].sortAlpha("Queue 5 ->");

                }
                case "106", "SPD", "spd" -> {
```

```
                    try {
                        FileWriter myWriter = new FileWriter("Data.txt");
                        refQueue[0].writingTOFile("Queue 1 ->",myWriter);
                        refQueue[1].writingTOFile("Queue 2 ->",myWriter);
                        refQueue[2].writingTOFile("Queue 3 ->",myWriter);
                        refQueue[3].writingTOFile("Queue 4 ->",myWriter);
                        refQueue[4].writingTOFile("Queue 5 ->",myWriter);
                        FuelQueue.writingFuelStock(myWriter);
                        System.out.println("Successfully written to the file
:)");
                        myWriter.close();
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
                case "107", "LPD", "lpd" -> {
                    try {
                        int count=0;
                        File inputFile = new File("Data.txt");
                        Scanner readFile = new Scanner(inputFile);
                        while (readFile.hasNextLine()) {
                            String data = readFile.nextLine();
                            String section1 =data.substring(0,5);

if(!((section1.equals("Queue"))||(section1.equals("Remai")))){
                                String[] x = data.split(" ");
                                if (count==1){

refQueue[0].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),1 );
                                }else if (count==2) {

refQueue[1].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),2 );
                                }else if (count==3) {

refQueue[2].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),3 );
                                }else if (count==4) {

refQueue[3].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),4 );
                                }else {

refQueue[4].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),5 );
                                }
                            }else{
                                count++;
                            }
                        }
                        System.out.println("Data Loaded successfully :)");
                    }catch (Exception e) {
                        System.out.println("Error when reading the file");
                    }

                }
                case "108", "STK", "stk" -> {
                    System.out.println(FuelQueue.getStock());
                }
                case "109", "AFS", "afs" -> {
                    System.out.print("Enter fuel stock : ");
```

```
                    double newStock;
                    try {
                        newStock = input.nextDouble();
                        input.nextLine();
                    }catch (Exception e){
                        input.nextLine();
                        System.out.println("Invalid input for Stock");
                        continue;
                    }
                    if(newStock<0){
                        System.out.println("invalid stock");
                        continue;
                    }
                    FuelQueue.setStock(newStock);
                    System.out.println("Fuel Stock updated");

                }
                case "110", "IFQ", "ifq" -> {
                    System.out.println("Income Queue 1 : Rs "+
refQueue[0].income());
                    System.out.println("Income Queue 2 : Rs "+
refQueue[1].income());
                    System.out.println("Income Queue 3 : Rs "+
refQueue[2].income());
                    System.out.println("Income Queue 4 : Rs "+
refQueue[3].income());
                    System.out.println("Income Queue 5 : Rs "+
refQueue[4].income());

                }
                case "999", "EXT", "ext" -> {
                    System.out.println("Program terminated");
                    break loop1;


                }
                default -> {
                    System.out.println("Invalid Operation try again");

                }
            }
            if (FuelQueue.getStock() <= FuelQueue.LIMIT) {
                System.out.printf("Remaining Stock is %.2f --> Please refill
!!! %n", FuelQueue.getStock());
            }
        }while (true);
    }

    // method related to 100
    public static void viewAllQueue(FuelQueue queue,int num){
        System.out.printf("Queue %d : - %n",num);
        queue.viewCustomer();
    }

    //method related to 103,104

    // input validation for which queue(1-5)
```

```
    public static int inputValidation(String op){
        Scanner scanner = new Scanner(System.in);

        while (true){
            try{
                System.out.print(op);
                String qNum = scanner.nextLine();
                int Num = Integer.parseInt(qNum);
                if ((Num<=5)&&(1<=Num)){
                    return Num;
                }else{
                    System.out.println("Queue number you entered out of range
enter a valid Queue number [1,2,3]");
                }
            }catch (Exception e){
                System.out.println("Enter a number 1,2,3,4 or 5");
            }
        }
    }

    //input validation for index of the queue(array)
    public static int indexValidation(int num){
        Scanner scanner = new Scanner(System.in);
        do{
            try{
                System.out.printf("Enter the position of %d queue you want to
remove the customer:",num);
                String qNum = scanner.nextLine();
                int Num = Integer.parseInt(qNum);
                if ((Num<=6)&&(1<=Num)){
                    return Num;
                }else{
                    System.out.println("index number out of range try number
less or equal to 6");
                }
            }catch (Exception e){
                System.out.println("Enter a number");
            }
        }while (true);
    }


}
```

Passenger.java

```
public class Passenger {

    private final String firstName;
    private final String secondName;
    private final String VehicleNo;
    private final double noLitersNeed;

    public Passenger(String firstName, String secondName, String vehicleNo,
double noLitersNeed) {
        this.firstName = firstName;
        this.secondName = secondName;
        this.VehicleNo = vehicleNo;
        this.noLitersNeed = noLitersNeed;
    }


    public double getNoLitersNeed() {
        return noLitersNeed;
    }

    // display all the details about a customer
    public void viewCustomerDetails(){
        System.out.printf("%s    %s    %s
%.2f",firstName,secondName,VehicleNo,noLitersNeed);
    }
    // to return customer details
    public String getDetails(){
        return firstName +" "+ secondName+" "+VehicleNo+" "+noLitersNeed;
    }

    public String customerName(){
        return firstName+ " "+ secondName;
    }

}
```

FuelQueue.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

public class FuelQueue {

    //creating a ArrayList as object type to hold Passenger object with all
the details
    private final ArrayList<Passenger> queue;
    private static double stock = 6600;
    public final static double LIMIT = 500;


    //constructor
    public FuelQueue() {
        this.queue=new ArrayList<>();
    }


    //adding a customer to a queue
    public void addCustomer(String fn,String ln,String vn,double noLiters,int
qNum){
        if (queue.size()<6){
            queue.add(new Passenger(fn,ln,vn,noLiters));
            stock -=noLiters;
            System.out.println(fn + " added to queue "+ qNum);
        }else {
            System.out.println("Queues are Full");
        }
    }


    //remove customer from a specific place
    public void removeCustomer(int index){
        index=index-1;
        try {
            double prvLiters = queue.get(index).getNoLitersNeed();
            stock += prvLiters;
            queue.remove(index);
            System.out.println("Customer Successfully removed");
        }catch (Exception e){
            System.out.println("No customer in that location");
        }
    }


    //remove served customer
    public void removeServed(){
        queue.remove(0);
        System.out.println("Customer Successfully removed");
    }
```

```
    //view customer in a queue one by one
    public void viewCustomer(){
        for(Passenger i : queue){
            i.viewCustomerDetails();
            System.out.println();
        }
    }



    // Show empty queue
    public void emptyQueue(){
        if(queue.size()<=6){
            System.out.println("Available for data entry");
        }else{
            System.out.println("full");
        }
    }



    //to get the size of each queue
    public int sizeOfQueue(){
        return queue.size();
    }

    //get stock
    public static double getStock() {
        return stock;
    }

    // update fuel stock
    public static void setStock(double stock) {
        FuelQueue.stock += stock;
    }

    //income of each fuel queue
    public double income(){
        double totalIncome = 0;
        // price of a fuel liter : Rs 430.00
        double PRICE = 430.00;
        for(Passenger i : queue){
            totalIncome = totalIncome + i.getNoLitersNeed() * PRICE;
        }
        return totalIncome;
    }

    //writing data of each queue to a file (Data.txt)
    public void writingTOFile(String massage,FileWriter myWriter){
        try {
            myWriter.write(massage+"\n");
            for (Passenger i : queue) {
                myWriter.write(i.getDetails()+"\n");
            }
        }catch (IOException e){
            System.out.println("Error when writing to the file :(");
```

```
        }
    }

    //writing current fuel stock to the file (Data.txt)
    public static void writingFuelStock(FileWriter myWriter){
        try {
            myWriter.write("Remaining fuel stock : "+stock + "\n");
        }catch (IOException e){
            System.out.println();
        }
    }

    public void sortAlpha(String massage){
        System.out.println(massage);
        ArrayList<String> sorted =new ArrayList<>();
        for(Passenger i : queue){
            sorted.add(i.customerName());
        }
        Collections.sort(sorted);
        for(String e : sorted){
            System.out.println(e);
        }
    }
}
```

Task 3

FuelQueueManagementSys.java

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

//Assumption :- Waiting queue can hold 20 customers
public class FuelQueueManagementSys {

    public static void main(String[] args) {

        System.out.println(".......Please Select a operation from the below
menu.......");
        System.out.println("""

                100 or VFQ: View all Fuel Queues.
                101 or VEQ: View all Empty Queues.
                102 or ACQ: Add customer to a Queue.
                103 or RCQ: Remove a customer from a Queue. (From a specific
location)
                104 or PCQ: Remove a served customer.
                105 or VCS: View Customers Sorted in alphabetical order.
                106 or SPD: Store Program Data into file.
                107 or LPD: Load Program Data from file.
                108 or STK: View Remaining Fuel Stock.
                109 or AFS: Add Fuel Stock.
                110 or IFQ: Income of each fuel queue.
                999 or EXT: Exit the Program.

                """);

        //array is created to hold 5 FuelQueue objects
        FuelQueue[] refQueue = new FuelQueue[5];
        for(int p=0;p<5;p++) {
            refQueue[p] = new FuelQueue();
        }

        Scanner input = new Scanner(System.in);
        loop1:
        do {
            //get the user input for operation
            System.out.print("Select operation : ");
            String choice = input.nextLine();
            switch (choice){
                case "100", "VFQ", "vfq" -> {
                    viewAllQueue(refQueue[0],1);
                    viewAllQueue(refQueue[1],2);
                    viewAllQueue(refQueue[2],3);
                    viewAllQueue(refQueue[3],4);
                    viewAllQueue(refQueue[4],5);
```

```
                }
                case "101", "VEQ", "veq" -> {
                    System.out.print("Fuel queue 1 : ");
                    refQueue[0].emptyQueue();
                    System.out.print("Fuel queue 2 : ");
                    refQueue[1].emptyQueue();
                    System.out.print("Fuel queue 3 : ");
                    refQueue[2].emptyQueue();
                    System.out.print("Fuel queue 4 : ");
                    refQueue[3].emptyQueue();
                    System.out.print("Fuel queue 5 : ");
                    refQueue[4].emptyQueue();
                }
                case "102", "ACQ", "acq" -> {

                    //selecting the queue with the minimum length.
                    ArrayList<Integer> size =new ArrayList<>();
                    size.add(refQueue[0].sizeOfQueue());
                    size.add(refQueue[1].sizeOfQueue());
                    size.add(refQueue[2].sizeOfQueue());
                    size.add(refQueue[3].sizeOfQueue());
                    size.add(refQueue[4].sizeOfQueue());
                    int min = Collections.min(size);

                    //getting user inputs
                    System.out.print("Enter customer first name : ");
                    String fn = input.nextLine();
                    fn =
String.valueOf(fn.charAt(0)).toUpperCase()+fn.substring(1);
                    System.out.print("Enter customer last name : ");
                    String ln = input.nextLine();
                    ln =
String.valueOf(ln.charAt(0)).toUpperCase()+ln.substring(1);
                    System.out.print("Enter vehicle number : ");
                    String vn = input.nextLine();
                    double noLiters;
                    System.out.print("Enter no of liters need : ");
                    try {
                        noLiters = input.nextDouble();
                        input.nextLine();
                        if(noLiters<=0){
                            System.out.println("Invalid");
                            continue;
                        }
                    }catch (Exception e){
                        System.out.println("Invalid input for liters");
                        input.nextLine();
                        continue;
                    }
                    // adding customer object to each of the queue
                    if (refQueue[0].sizeOfQueue() == min){
                        refQueue[0].addCustomer(fn,ln,vn,noLiters,1);
                    } else if (refQueue[1].sizeOfQueue() == min) {
                        refQueue[1].addCustomer(fn,ln,vn,noLiters,2);
                    } else if (refQueue[2].sizeOfQueue() == min) {
                        refQueue[2].addCustomer(fn,ln,vn,noLiters,3);
                    } else if (refQueue[3].sizeOfQueue() == min) {
```

```
                        refQueue[3].addCustomer(fn,ln,vn,noLiters,4);
                    } else {
                        refQueue[4].addCustomer(fn, ln, vn, noLiters,5);
                    }
                }
                case "103", "RCQ", "rcq" -> {
                    int queueNum = inputValidation("Enter which queue you
want to remove the customer from : ");
                    if (queueNum == 1){

refQueue[0].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 2) {

refQueue[1].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 3) {

refQueue[2].removeCustomer(indexValidation(queueNum));
                    } else if (queueNum == 4) {

refQueue[3].removeCustomer(indexValidation(queueNum));
                    } else {

refQueue[4].removeCustomer(indexValidation(queueNum));
                    }
                }
                case "104", "PCQ", "pcq" -> {
                    int queueNum = inputValidation("Enter which queue you
want to remove the customer from : ");
                    if (queueNum == 1){
                        refQueue[0].removeServed();
                    } else if (queueNum == 2) {
                        refQueue[1].removeServed();
                    } else if (queueNum == 3) {
                        refQueue[2].removeServed();
                    } else if (queueNum == 4) {
                        refQueue[3].removeServed();
                    } else {
                        refQueue[4].removeServed();
                    }
                }
                case "105", "VCS", "vcs" -> {
                    System.out.println("----------- Customer sorted A - Z by
name -----------");
                    refQueue[0].sortAlpha("Queue 1 ->");
                    refQueue[1].sortAlpha("Queue 2 ->");
                    refQueue[2].sortAlpha("Queue 3 ->");
                    refQueue[3].sortAlpha("Queue 4 ->");
                    refQueue[4].sortAlpha("Queue 5 ->");
                }
                case "106", "SPD", "spd" -> {
                    try {
                        FileWriter myWriter = new FileWriter("Data.txt");
                        refQueue[0].writingTOFile("Queue 1 ->",myWriter);
                        refQueue[1].writingTOFile("Queue 2 ->",myWriter);
                        refQueue[2].writingTOFile("Queue 3 ->",myWriter);
                        refQueue[3].writingTOFile("Queue 4 ->",myWriter);
                        refQueue[4].writingTOFile("Queue 5 ->",myWriter);
```

```
                        FuelQueue.writingFuelStock(myWriter);
                        System.out.println("Successfully written to the file
:)");
                        myWriter.close();
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
                case "107", "LPD", "lpd" -> {
                    try {
                        int count=0;
                        File inputFile = new File("Data.txt");
                        Scanner readFile = new Scanner(inputFile);
                        while (readFile.hasNextLine()) {
                            String data = readFile.nextLine();
                            String section1 =data.substring(0,5);

if(!((section1.equals("Queue"))||(section1.equals("Remai")))){
                                String[] x = data.split(" ");
                                if (count==1){

refQueue[0].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),1 );
                                }else if (count==2) {

refQueue[1].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),2 );
                                }else if (count==3) {

refQueue[2].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),3 );
                                }else if (count==4) {

refQueue[3].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),4 );
                                }else {

refQueue[4].addCustomer(x[0],x[1],x[2],Double.parseDouble(x[3]),5 );
                                }
                            }else{
                                count++;
                            }
                        }
                        System.out.println("Data Loaded successfully :)");
                    }catch (Exception e) {
                        System.out.println("Error when reading the file");
                    }
                }
                case "108", "STK", "stk" -> {
                    System.out.println(FuelQueue.getStock());
                }
                case "109", "AFS", "afs" -> {
                    System.out.print("Enter fuel stock : ");
                    double newStock;
                    try {
                        newStock = input.nextDouble();
                        input.nextLine();
                    }catch (Exception e){
                        input.nextLine();
                        System.out.println("Invalid input for Stock");
                        continue;
```

```
                    }
                    if(newStock<0){
                        System.out.println("invalid stock");
                        continue;
                    }
                    FuelQueue.setStock(newStock);
                    System.out.println("Fuel Stock updated");
                }
                case "110", "IFQ", "ifq" -> {
                    System.out.println("Income Queue 1 : Rs "+
refQueue[0].income());
                    System.out.println("Income Queue 2 : Rs "+
refQueue[1].income());
                    System.out.println("Income Queue 3 : Rs "+
refQueue[2].income());
                    System.out.println("Income Queue 4 : Rs "+
refQueue[3].income());
                    System.out.println("Income Queue 5 : Rs "+
refQueue[4].income());
                }
                case "999", "EXT", "ext" -> {
                    System.out.println("Program terminated");
                    break loop1;
                }
                default -> {
                    System.out.println("Invalid Operation try again");
                }
            }
            //each iteration check stock count is low
            if (FuelQueue.getStock() <= FuelQueue.LIMIT) {
                System.out.printf("Remaining Stock is %.2f --> Please refill
!!! %n", FuelQueue.getStock());
            }
        }while (true);
    }

    // method related to 100
    public static void viewAllQueue(FuelQueue queue,int num){
        System.out.printf("Queue %d : - %n",num);
        queue.viewCustomer();
    }

    //method related to 103,104

    // input validation for which queue(1-5)
    public static int inputValidation(String op){
        Scanner scanner = new Scanner(System.in);

        while (true){
            try{
                System.out.print(op);
                String qNum = scanner.nextLine();
                int Num = Integer.parseInt(qNum);
                if ((Num<=5)&&(1<=Num)){
                    return Num;
                }else{
                    System.out.println("Queue number you entered out of range
```

```
enter a valid Queue number [1,2,3]");
                }
            }catch (Exception e){
                System.out.println("Enter a number 1,2,3,4 or 5");
            }
        }
    }

    // validation for index of the queue(array)
    public static int indexValidation(int num){
        Scanner scanner = new Scanner(System.in);
        do{
            try{
                System.out.printf("Enter the position of %d queue you want to
remove the customer:",num);
                String qNum = scanner.nextLine();
                int Num = Integer.parseInt(qNum);
                if ((Num<=6)&&(1<=Num)){
                    return Num;
                }else{
                    System.out.println("index number out of range try number
less or equal to 6");
                }
            }catch (Exception e){
                System.out.println("Enter a number");
            }
        }while (true);
    }


}
```

Passenger.java

```
public class Passenger {

    private final String firstName;
    private final String secondName;
    private final String VehicleNo;
    private final double noLitersNeed;

    public Passenger(String firstName, String secondName, String vehicleNo,
double noLitersNeed) {
        this.firstName = firstName;
        this.secondName = secondName;
        this.VehicleNo = vehicleNo;
        this.noLitersNeed = noLitersNeed;
    }


    public double getNoLitersNeed() {
        return noLitersNeed;
    }

    // display all the details about a customer
    public void viewCustomerDetails(){
        System.out.printf("%s    %s    %s
%.2f",firstName,secondName,VehicleNo,noLitersNeed);
    }
    // to return customer details
    public String getDetails(){
        return firstName +" "+ secondName+" "+VehicleNo+" "+noLitersNeed;
    }
    public String customerName(){
        return firstName+ " "+ secondName;
    }

}
```

FuelQueue.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

public class FuelQueue {

    //creating a ArrayList as object type to hold Passenger object with all
the details
    private final ArrayList<Passenger> queue;
    private static double stock = 6600;
    public final static double LIMIT = 500;


    //constructor
    public FuelQueue() {
        this.queue=new ArrayList<>();
    }

    //WaitingQ waitingLine = new WaitingQ();
    //adding a customer to a queue
    public void addCustomer(String fn,String ln,String vn,double noLiters,int
qNum){
        if (queue.size()<6){
            queue.add(new Passenger(fn,ln,vn,noLiters));
            stock -=noLiters;
            System.out.println(fn + " added to queue "+ qNum);
        }else {
            // when all the queues are full following code will run
            System.out.println("Queues are full customer will be added to
waiting queue");

            WaitingQ.addCustomerWaitingQ(fn,ln,vn,noLiters);
        }
    }


    //remove customer from a specific place
    public void removeCustomer(int index){
        index=index-1;
        try {
            double prvLiters = queue.get(index).getNoLitersNeed();
            stock += prvLiters;
            queue.remove(index);
            System.out.println("Customer Successfully removed");
        }catch (Exception e){
            System.out.println("No customer in that location");
        }
    }

    //remove served customer
    public void removeServed(){
        queue.remove(0);
        System.out.println("Customer Successfully removed");
```

```
        if (WaitingQ.getWaitingQueue().size()!=0)
        {
            queue.add(WaitingQ.getFirstCustomer());
        }
    }

    //view customer in a queue one by one
    public void viewCustomer(){
        for(Passenger i : queue){
            i.viewCustomerDetails();
            System.out.println();
        }
    }

    // Show empty queue
    public void emptyQueue(){
        if(queue.size()<6){
            System.out.println("Available for data entry");
        }else{
            System.out.println("full");
        }
    }

    //to get the size of each queue
    public int sizeOfQueue(){
        return queue.size();
    }

    //get stock
    public static double getStock() {
        return stock;
    }

    // update fuel stock
    public static void setStock(double stock) {
        FuelQueue.stock += stock;
    }

    //income of each fuel queue
    public double income(){
        double totalIncome = 0;
        // price of a fuel liter : Rs 430.00
        double PRICE = 430.00;
        for(Passenger i : queue){
            totalIncome = totalIncome + i.getNoLitersNeed() * PRICE;
        }
        return totalIncome;
    }

    //writing data of each queue to a file (Data.txt)
    public void writingTOFile(String massage,FileWriter myWriter){
        try {
            myWriter.write(massage+"\n");
            for (Passenger i : queue) {
                myWriter.write(i.getDetails()+"\n");
            }
        }catch (IOException e){
```

```
            System.out.println("Error when writing to the file :(");
        }
    }

    //writing current fuel stock to the file (Data.txt)
    public static void writingFuelStock(FileWriter myWriter){
        try {
            myWriter.write("Remaining fuel stock : "+stock + "\n");
        }catch (IOException e){
            System.out.println();
        }
    }

    //sorting
    public void sortAlpha(String massage){
        System.out.println(massage);
        ArrayList<String> sorted =new ArrayList<>();
        for(Passenger i : queue){
            sorted.add(i.customerName());
        }
        Collections.sort(sorted);
        for(String e : sorted){
            System.out.println(e);
        }
    }
}
```

WaitingQ.java

```
import java.util.LinkedList;
import java.util.Queue;

public class WaitingQ {

    private static final Queue<Passenger> waitingQ = new LinkedList<>();

   //adding extra customer to waiting queue
    public static void addCustomerWaitingQ(String fn,String ln,String
vn,double noLiters){
        if (waitingQ.size()<20) {
            waitingQ.offer(new Passenger(fn, ln, vn, noLiters));
            System.out.println("Customer added to waiting queue !");
        }else {
            System.out.println("Waiting queue reach its limit customer will
not be added :(");
        }
    }

    //once a customer is removed first customer in the waiting queue will be
added to the removed customer position
    public static Passenger getFirstCustomer(){
        Passenger temp=waitingQ.poll();
        FuelQueue.setStock(-temp.getNoLitersNeed());
        return temp;
    }

    //getter
    public static Queue<Passenger> getWaitingQueue() {
        return waitingQ;
    }

}
```

<<END>>