Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module	: 4COSC010C.3: Software Development II
Module Leader	: Deshan Sumanathilaka
Date of submission	: 07/08/2022
Student ID	: 20211274 / w1903065
Student First Name	: Tharkana
Student Surname	: Bandara

Name : E.M.T.P.S. BANDARA

Student ID : 20211274

Test Cases

Task 1(Array Version)

	Test Case	Expected Result	Actual Result	Pass/Fail
1	When the code runs, Display the main menu.	Displays the main menu.	Displays the main menu.	Pass
2	(Queues Initialised correctly) After the program starts, 100 or VFQ	Displays 'N/A' for all queues.	Displays 'N/A' for queues.	Pass
3	(Add customer to a Queue) After the program starts, 102 or ACQ enter "tharkana" to 2 102 or ACQ enter "prabhath" to 3	Display 100 or VFQ _View all Fuel Queues option _ Pump 1 Queue: Pump 1 queue is N/A. Pump 2 Queue: Position 0 is occupied by tharkana. Pump 3 Queue: Position 0 is occupied by prabhath.	Display _ View all Fuel Queues option _ Pump 1 Queue: Pump 1 queue is N/A. Pump 2 Queue: Position 0 is occupied by tharkana. Pump 3 Queue: Position 0 is occupied by prabhath.	Pass
4	(View all empty queues) 101 or VEQ	Displays _ View all Empty Queues option _ Pump 1 queue is empty.	_ View all Empty Queues option Pump 1 queue is empty.	Pass

5	(Remove customer from queue) 103 or RPB enter queue number 2	100 or VFQ Display _ View all Fuel Queues option _ Pump 2 Queue: Position 0 is empty. Position 1 is empty. Position 2 is empty. Position 3 is empty. Position 4 is empty. Position 5 is empty.	Display _ View all Fuel Queues option_ Pump 2 Queue: Position 0 is empty. Position 1 is empty. Position 2 is empty. Position 3 is empty. Position 4 is empty. Position 5 is empty.	Pass
6	(View all empty queues) 101 or VEQ	Displays _ View all Empty Queues option _ Pump 1 queue is empty. Pump 2 queue is empty.	Displays _ View all Empty Queues option _ Pump 1 queue is empty. Pump 2 queue is empty.	Pass
7	(Add a customer to the queue) 102 or ACQ enter “avi” to queue 7	Displays error message “Please check your input again! Queue number should be between 1 and 3. ” range error.	Displays error message “ Please check your input again! Queue number should be between 1 and 3.” range error.	Pass
8	(Add a customer to Queue) 102 or ACQ enter “navini” to queue j	Displays _ Add the customer to a queue option _ Enter the queue number (1-3): j Please check the queue number! (Input should be an integer). Enter the queue number (1-3):	Displays _ Add the customer to a queue option _ Enter the queue number (1-3): j Please check the queue number! (Input should be an integer). Enter the queue number (1-3):	Pass

9	(Remove customer from the queue) 103 or RPB enter queue number 10	Displays _ Remove a customer from a Queue option _ Enter the queue number of the removed customer (1-3): 10 Please check the input! (Input should be between 1-3) Enter the queue number of the removed customer (1-3):	Displays _ Remove a customer from a Queue option _ Enter the queue number of the removed customer (1-3): 10 Please check the input! (Input should be between 1-3) Enter the queue number of the removed customer (1-3):	Pass
10	(Remove customer from the queue) 103 or RPB enter queue number w	Displays _ Remove a customer from a Queue option _ Enter the queue number of the removed customer (1-3): w Please check the input and try again! (Input should be an integer). Enter the queue number of the removed customer (1-3):	Displays _ Remove a customer from a Queue option _ Enter the queue number of the removed customer (1-3): w Please check the input and try again! (Input should be an integer). Enter the queue number of the removed customer (1-3):	Pass

11	(Remove a served customer.) 104 or PCQ	Displays _ Remove a served customer option _ Enter the queue number of the removed customer (1-3): 1 served customer successfully removed	Displays _ Remove a served customer option _ Enter the queue number of the removed customer (1-3): 1 served customer successfully removed	Pass
12	(View customers sorted in alphabetical order) 105 or VCS	Displays customers in alphabetical order.	Displays customers in alphabetical order.	Pass
13	(Store program data into the file) 106 or SPD	_ Store Program Data into file option _ Successfully stored data into the file.	_ Store Program Data into file option _ Successfully stored data into the file.	Pass
14	(Load program data from file) 107 or LPD	Displays the data in the file.	Displays the data in the file.	Pass

15	(View remaining Fuel stock) 108 or LPD	Displays the remaining fuel stock.	Displays the remaining fuel stock.	Pass
16	(Add Fuel stock) 109 or AFS	_ Add Fuel Stock _ Enter the fuel quantity which will be added to the stock: 100 100, new fuel has been added to the stock.	_ Add Fuel Stock _ Enter the fuel quantity which will be added to the stock: 100 100, new fuel has been added to the stock.	Pass
17	(Exit the program) 999 or EXT	Displays ("Thank you for using Fuel Center Programme") Stay safe!	Displays ("Thank you for using Fuel Center Programme") Stay safe!	Pass

Task 02 (Class version)

18	(Add customer to a Queue) After the program starts, 102 or ACQ enter "Malith" to 2 102 or ACQ enter "sandaru" to 3 102 or ACQ enter "chamoda" to 5	Display 100 or VFQ _View all Fuel Queues option _ Pump 1 Queue: Pump 1 queue is empty. Pump 2 Queue: Position 0 is occupied by Malith. Pump 3 Queue: Position 0 is occupied by sandaru. Pump 4 Queue: Pump 4 queue is empty.	Display 100 or VFQ _View all Fuel Queues option _ Pump 1 Queue: Pump 1 queue is empty. Pump 2 Queue: Position 0 is occupied by Malith. Pump 3 Queue: Position 0 is occupied by sandaru. Pump 4 Queue: Pump 4 queue is empty.	Pass
		Pump 5 Queue: Position 0 is occupied by chamoda.	Pump 5 Queue: Position 0 is occupied by chamoda.	

19	(Income of Fuel Queue) 110 or IFQ	_ Income of Fuel Queue _ Total income of fuel queue no.1 - Rs.4300 Total income of fuel queue no.2 - Rs.4300 Total income of fuel queue no.3 - Rs.4300 Total income of fuel queue no.4 - Rs.0 Total income of fuel queue no.5 - Rs.0	_ Income of Fuel Queue _ Total income of fuel queue no.1 - Rs.4300 Total income of fuel queue no.2 - Rs.4300 Total income of fuel queue no.3 - Rs.4300 Total income of fuel queue no.4 - Rs.0 Total income of fuel queue no.5 - Rs.0	Pass

Task 3(Waiting Queue)

20	(Add Customer to a Queue) 102 or ACQ enter "ravi jay" to the waiting queue.	_ Add customer to a Queue option _ Sorry! All the queues are full. Customer will be added to the waiting queue. Enter customer's first name: ravi Enter customer's second name: jay Enter customer's vehicle number: e3 Enter the required no of liters: 10 Passenger ravi jay successfully added to the waiting queue.	_ Add customer to a Queue option _ Sorry! All the queues are full. Customer will be added to the waiting queue. Enter customer's first name: ravi Enter customer's second name: jay Enter customer's vehicle number: e3 Enter the required no of liters: 10 Passenger ravi jay successfully added to the waiting queue.	Pass
21	(Remove a served customer) 104 or PCQ remove served customer from queue 2.	_ Remove a served customer option _ Enter the queue number of the served customer (1-5): 2 Customer wrd successfully removed from the queue 2 Passenger ravi successfully added to the 2	_ Remove a served customer option _ Enter the queue number of the served customer (1-5): 2 Customer wrd successfully removed from the queue 2 Passenger ravi successfully added to the 2	Pass

Discussion

Making a test case has as its major objective determining whether the program's numerous features are error- and bug-free. Test cases contribute to the program's quality improvement. Here, I used 21 test cases to ensure that the program was working (Tasks 1,2,3). I focused on validating the user-provided input and mostly paid attention to the methods' functioning.

I used at least one test case for each function in order to ensure that it runs without any bugs or errors. While working on the test cases, I found a number of small flaws, but I was able to control them all, and I rectified each and every one of them. To ensure that my test cases included every element of my program, I ran tests for each function one at a time.

After making sure that the functions were operating correctly, I began the data validation stage. I went through and double-checked every user entry to make sure the data type and range were entered properly.

But after running the test cases, I found many of flaws in my application, which made me realize how important it was to do test cases. And I'm hoping that my test cases covered every aspect of my program.

Code:

w1903065_arrays_only Task 01 – Array Version (Task1.java)

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

/**
 SD2 Course Work Task 01 (Array Version).
 Fuel Center Program.
 E.M.T.P.S.BANDARA - w1903065 (20211274).
 */
```

```
public class Task1{

    private static double FuelStock = 6600;
    private static String[][] q1_Customer = new String[6][2];//6 customer in 3
queue {name,status}
    private static String[][] q2_Customer = new String[6][2];//6 customer in 3
queue {name,status}
    private static String[][] q3_Customer = new String[6][2];//6 customer in 3
queue {name,status}
    private static double ServedQTY = 0;
    private static Scanner input = new Scanner(System.in);

    public static void main (String[] args) throws IOException {
        int loop = 0;

        while (loop < 1 ){
            println("");
            println("====================================");
            println("=======Fuel Station System==========");
            println("====================================");
            println("");
            System.out.println("100 or VFQ:\t View all Fuel Queues.");
            System.out.println("101 or VEQ:\t View all Empty Queues.");
            System.out.println("102 or ACQ:\t Add customer to a Queue.");
            System.out.println("103 or RCQ:\t Remove a customer from a
Queue.");
            System.out.println("104 or PCQ:\t Remove a served customer.");
            System.out.println("105 or VCS:\t View Customers Sorted in
alphabetical order");
            System.out.println("106 or SPD:\t Store Program Data into file.");
            System.out.println("107 or LPD:\t Load Program Data from file.");
            System.out.println("108 or STK:\t View Remaining Fuel Stock.");
            System.out.println("109 or AFS:\t Add Fuel Stock.");
            System.out.println("999 or EXT:\t Exit the Program.");
            System.out.println();


            System.out.print("Please Enter action code : ");
            String code = input.next();


            if (code.equals("100") || code.equals("VFQ") ||
code.equals("vfq")) {
                viewQueue();
            }
            else if (code.equals("101") || code.equals("VEQ") ||
code.equals("veq")) {
                viewEmptyQueue();
            }
            else if (code.equals("102") || code.equals("ACQ") ||
code.equals("acq")) {
                addCustomer();
            }
            else if (code.equals("103") || code.equals("RCQ") ||
code.equals("rcq")) {
                removeCustomer();
            }
            else if (code.equals("104") || code.equals("PCQ") ||
code.equals("pcq")) {
                removeServedCustomer();
            }
            else if (code.equals("105") || code.equals("VCS") ||
code.equals("vcs")) {
                viewCustomerSorted();
            }
            else if (code.equals("106") || code.equals("SPD") ||
code.equals("spd")) {
                storeDataToFile();
            }
            else if (code.equals("107") || code.equals("LPD") ||
code.equals("lpd")) {
                loadDataInFile();
            }
            else if (code.equals("108") || code.equals("STK") ||
code.equals("stk")) {
                viewRemainingFuelStock();
            }
            else if (code.equals("109") || code.equals("AFS") ||
code.equals("afs")) {
                addFuelStock();
            }
            else if (code.equals("999") || code.equals("EXT") ||
code.equals("ext")) {
                break;
            }
            else{
                System.out.println("Invalid Option!!");
            }
        }
    }


    private static void viewQueue(){
        println("----------------------------");
        println("---------View Queue---------");
        println( "\n1st Queue customers:");
        for(int c=0; c<6; c++ ){
            String cusName = q1_Customer[c] == null?null:q1_Customer[c][0];
            println("\r\t"+(c+1)+"-"+ ((cusName == null)?"N/A": cusName));
        }
        println( "\n2nd Queue customers:");
        for(int c=0; c<6; c++ ){
            String cusName = q2_Customer[c] == null?null:q2_Customer[c][0];
            println("\r\t"+(c+1)+"-"+ ((cusName == null)?"N/A": cusName));
        }

        println( "\n3rd Queue customers:");
        for(int c=0; c<6; c++ ){
            String cusName = q3_Customer[c] == null?null:q3_Customer[c][0];
            println("\r\t"+(c+1)+"-"+ ((cusName == null)?"N/A": cusName));
        }
        println("----------------------------");

    }
    private static void viewEmptyQueue(){
        println("----------------------------");
        println("----View All Empty Queue----");

        boolean isEmpty_Q1=true;
        for(int c= 0; c<6; c++){
            isEmpty_Q1=q1_Customer[c] ==null || q1_Customer[c][0]==null ||
q1_Customer[c][0].length() == 0;
            if(!isEmpty_Q1)break;
        }

        boolean isEmpty_Q2=true;
        for(int c= 0; c<6; c++){
            isEmpty_Q2= q2_Customer[c] ==null || q2_Customer[c][0]==null ||
q2_Customer[c][0].length() == 0;
            if(!isEmpty_Q2)break;
        }

        boolean isEmpty_Q3=true;
        for(int c= 0; c<6; c++){
            isEmpty_Q3=q3_Customer[c] ==null ||  q3_Customer[c][0]==null ||
q3_Customer[c][0].length() == 0;
            if(!isEmpty_Q3)break;
        }

        print(isEmpty_Q1? "1st Queue is empty\n":"");
        print(isEmpty_Q2? "2nd Queue is empty\n":"");
        print(isEmpty_Q3? "3rd Queue is empty\n":"");

        println("----------------------------");
    }
    private static void addCustomer(){

        print("Please Enter Queue No:");
        int Q_number=-1;
        if(input.hasNextInt()){
            Q_number = input.nextInt();
        }else{
            println("Invalid Queue Number Entered!");
            return;
        }

        print("Please Enter Customer Name:");
        String name = input.next();

        int locationIndex = FindQueueLocation(Q_number);

        if(Q_number == 1){
            q1_Customer[locationIndex]=new String[] {name,"new"};
            System.out.println();
            System.out.println("Successfully added a customer");
        }else if(Q_number == 2){
            q2_Customer[locationIndex]=new String[] {name,"new"};
            System.out.println();
            System.out.println("Successfully added a customer");
        }else if(Q_number == 3){
            q3_Customer[locationIndex]=new String[] {name,"new"};
            System.out.println();
            System.out.println("Successfully added a customer");
        }

        //check fuel estimation
        double fuel_need=0;
        for(int c =0; c<6; c++){
            boolean flag = q1_Customer[c][0]!=null &&
q1_Customer[c][0].length() != 0;
            if(flag){
                fuel_need+=10;
            }
        }
        for(int c =0; c<6; c++){
            boolean flag = q2_Customer[c][0]!=null &&
q2_Customer[c][0].length() != 0;
            if(flag){
                fuel_need+=10;
            }
        }
        for(int c =0; c<6; c++){
            boolean flag = q2_Customer[c][0]!=null &&
q2_Customer[c][0].length() != 0;
            if(flag){
                fuel_need+=10;
            }
        }
        if((FuelStock - fuel_need )<=500){
            println("\n\tWarning - Reached 500L of fuel!");
        }

    }
    private static int FindQueueLocation(int Q_number){
        if(Q_number == 1){
            //find queue location
            for(int c =0; c<6; c++){
                boolean flag = q1_Customer[c][0]==null ||
q1_Customer[c][0].length() == 0;
                if(flag){
                    return c;
                }
            }
        }
        else if(Q_number == 2){
            //find queue location
            for(int c =0; c<6; c++){
                boolean flag = q2_Customer[c][0]==null ||
q2_Customer[c][0].length() == 0;
                if(flag){
                    return c;
                }
            }
        }
        else if(Q_number == 3){
            //find queue location
            for(int c =0; c<6; c++){
                boolean flag = q3_Customer[c][0]==null ||
q3_Customer[c][0].length() == 0;
                if(flag){
                    return c;
                }
            }
        }else{
            println("Invalid Queue Number!");
        }
        return -1;
    }
    private static void removeCustomer(){
        print("Please Enter Queue No:");
        int Q_number=-1;
        if(input.hasNextInt()){
            Q_number = input.nextInt();
        }else{
            println("Invalid Queue Number Entered!");
            return;
        }

        if(Q_number>3 || Q_number<1){
            println("Invalid Queue Number Entered!");
            return;
        }
        //----
        print("Please Enter Location No:");
        int Q_location=-1;
        if(input.hasNextInt()){
            Q_location = input.nextInt();
        }else{
            println("Invalid Queue Location!");
            return;
        }

        if(Q_location>6 || Q_location<1){
            println("Invalid Queue Location Entered!");
            return;
        }

        if(Q_number == 1){
            q1_Customer[Q_location-1]=new String[]{null,null};
        }else if(Q_number ==2 ){
            q2_Customer[Q_location-1]=new String[]{null,null};
        }else if(Q_number ==3 ){
            q3_Customer[Q_location-1]=new String[]{null,null};
        }
        println("Customer Removed Successfully");
    }
    private static void removeServedCustomer(){
        println("Start Served Customer Remove Process..");
        for(int c =0; c<6; c++){
            if( q1_Customer[c][0]!=null && !q1_Customer[c][1].equals("new") ){
                q1_Customer[c] =new String[]{null,null};
            }
            if( q2_Customer[c][0]!=null && !q2_Customer[c][1].equals("new") ){
                q2_Customer[c] =new String[]{null,null};
            }
            if( q3_Customer[c][0]!=null && !q3_Customer[c][1].equals("new") ){
                q3_Customer[c] =new String[]{null,null};
            }
        }
        println("Reordering Queue...");
        //reset queue
        int lastEmptyIndex=-1;
        for(int c =0; c<6; c++){
            boolean flag = q1_Customer[c][0]==null ||
q1_Customer[c][0].length() == 0;
            if(!flag && lastEmptyIndex != -1 ){
                q1_Customer[lastEmptyIndex]=q1_Customer[c];
                q1_Customer[c]=new String[]{null,null};
            }
            if(flag){
                lastEmptyIndex = c;
            }
        }
        lastEmptyIndex=-1;
        for(int c =0; c<6; c++){
            boolean flag = q2_Customer[c][0]==null ||
q2_Customer[c][0].length() == 0;
            if(!flag && lastEmptyIndex != -1 ){
                q2_Customer[lastEmptyIndex]=q2_Customer[c];
                q2_Customer[c]=new String[]{null,null};
            }
            if(flag){
                lastEmptyIndex = c;
            }
        }
        lastEmptyIndex=-1;
        for(int c =0; c<6; c++){
            boolean flag = q3_Customer[c][0]==null ||
q3_Customer[c][0].length() == 0;
            if(!flag && lastEmptyIndex != -1 ){
                q3_Customer[lastEmptyIndex]=q3_Customer[c];
                q3_Customer[c]=new String[]{null,null};
            }
            if(flag){
                lastEmptyIndex = c;
            }
        }
        println("End Served Customer Remove Process!");

    }

    private static void viewCustomerSorted(){

        String[] customers = new String[18];
        int ind=0;
        for(int i=0;i<6;i++){
            if(q1_Customer[i][0]==null || q1_Customer[i][0].length() ==0){
                continue;
            }
            customers[ind++] = q1_Customer[i][0];
        }
        for(int i=0;i<6;i++){
            if(q2_Customer[i][0]==null || q2_Customer[i][0].length() ==0){
                continue;
            }
            customers[ind++] = q2_Customer[i][0];
        }

        for(int i=0;i<6;i++){
            if(q3_Customer[i][0]==null || q3_Customer[i][0].length() ==0){
                continue;
            }
            customers[ind++] = q3_Customer[i][0];
        }

        for(int i=0;i<18;i++){
            for(int j=0;j<=i;j++){
                if(customers[i] == null || customers[j]==null){
                    continue;
                }
                if (customers[i].compareTo(customers[j]) < 0) //compares two
elements of the array
                {
                    String temp = customers[i];
                    customers[i] = customers[j];
                    customers[j] = temp;  //rearranged in a way to make sure
it is in alphabetical order
                }
            }
        }
        println("-----Customer List------\n");
        boolean flagHasCustomer=false;
        for(int i=0; i< 18; i++){
            if(customers[i] == null){
                continue;
            }
            flagHasCustomer =true;
            println("\t"+(i+1)+" - "+customers[i]);
        }
        if(!flagHasCustomer){
            println("!!!Empty Queue Customers!!!");
        }
    }
    private static void storeDataToFile()throws IOException {
        File obj = new File("fuelStation.txt");
        obj.createNewFile();  //A new file is created using the method
        FileWriter theWriter = new FileWriter(obj.getName());
        for(int x = 0; x<q1_Customer.length; x++){
            theWriter.write("1@"+x+"@"+q1_Customer[x][0]+"@"+q1_Customer[x][1]
+ "\n");
        }
        for(int x = 0; x<q2_Customer.length; x++){
            theWriter.write("2@"+x+"@"
+q2_Customer[x][0]+"@"+q2_Customer[x][1] + "\n");
        }
        for(int x = 0; x<q3_Customer.length; x++){
            theWriter.write("3@"+x+"@"+q3_Customer[x][0]+"@"+q3_Customer[x][1]
+ "\n");
```

```
        }
        println("Fuel Station Details Stored");
        theWriter.close();
    }
    private static void loadDataInFile() throws IOException {


        File obj = new File("fuelStation.txt");
        Scanner theReader = new Scanner(obj);

        while (theReader.hasNextLine()) {
            String data = theReader.nextLine();
            String[] spData= data.split("@",-2);

            if(spData[0].equals("1")){
                q1_Customer[Integer.parseInt(spData[1])] =new String[]
{spData[2],spData[3]};
            }
            if(spData[0].equals("2")){
                q2_Customer[Integer.parseInt(spData[1])] =new String[]
{spData[2],spData[3]};
            }
            if(spData[0].equals("3")){
                q3_Customer[Integer.parseInt(spData[1])] =new String[]
{spData[2],spData[3]};
            }

        }
        println("Loaded Fuel Station Details");
    }
    private static void viewRemainingFuelStock(){
        println("\n\tRemaining Fuel Stock -"+ FuelStock +"L \n");
    }
    private static void addFuelStock(){
        println("Enter Fuel Qty:");
        if(input.hasNextDouble()){
            double qty = input.nextDouble();
            FuelStock+= qty;
            println("Current Stock - " + FuelStock);
        }
        else{
            println("Invalid Value Entered!");
        }
    }
    private static void FuelServe(){
        print("Please Enter Queue No:");
        int Q_number=-1;
        if(input.hasNextInt()){
            Q_number = input.nextInt();
        }else{
            println("Invalid Queue Number Entered!");
            return;
        }

        if(Q_number>3 || Q_number<1){
            println("Invalid Queue Number Entered!");
            return;
        }
        //----
        print("Please Enter QTY:");
        double qty=0;
        if(input.hasNextDouble()){
            qty = input.nextDouble();
        }else{
            println("Invalid Fuel QTY!");
            return;
        }

        if(qty<0){
            println("Invalid Fuel QTY!");
            return;
        }
        if(qty>FuelStock){
            println("This QTY is not available!\n"+"Current QTY -
"+FuelStock+"L" );
            return;
        }

        if(Q_number==1){
            for(int c =0; c<6; c++){
                if( q1_Customer[c][0]!=null && q1_Customer[c][1].equals("new")
){

                    q1_Customer[c] =new String[] {q1_Customer[c][0],"Served"};
                    break;
                }
            }
        }
        if(Q_number==2){
            for(int c =0; c<6; c++){
                if( q2_Customer[c][0]!=null && q2_Customer[c][1].equals("new")
){

                    q2_Customer[c] =new String[] {q2_Customer[c][0],"Served"};
                    break;
                }
            }
        }
        if(Q_number==3){
            for(int c =0; c<6; c++){
                if( q3_Customer[c][0]!=null && q3_Customer[c][1].equals("new")
){

                    q3_Customer[c] =new String[] {q3_Customer[c][0],"Served"};
                    break;
                }
            }
        }
        FuelStock -=qty;
        println("\n\tFuel Served to customer!");
    }
    private static void print(String text){System.out.print(text);}
    private static void println(String text){System.out.println(text);}
}
```

Task 02/03 – Class Version & Waiting List

Fuelmanagement.java

```java
package fuelmanagement;

import java.util.*;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;

/**
 SD2 Course Work Task 02/03 (Classes Version & waiting List).
 Fuel Center Program.  E.M.T.P.S. Bandara- w1903065 (20211274)
 */


public class FuelManagement {

    private double FuelStock = 6600;
    // private static String[][] q1_Customer = new String[6][2];//6 customer
in 3 queue {name,status}
    // private static String[][] q2_Customer = new String[6][2];//6 customer
in 3 queue {name,status}
    // private static String[][] q3_Customer = new String[6][2];//6 customer
in 3 queue {name,status}

    private FuelQueue[] feulQ = new FuelQueue[5];
    private ArrayList<Passenger> waitingList = new ArrayList<Passenger>();

    private double ServedQTY = 0;
    private double fuelPrice = 430;
    private Scanner input = new Scanner(System.in);

    public FuelManagement() {
        for (int i = 0; i < feulQ.length; i++) {
            feulQ[i] = new FuelQueue();
        }
    }

    public static void main(String[] args) throws IOException,
ClassNotFoundException {
        int loop = 0;
        FuelManagement fuelManagement = new FuelManagement();
        while (loop < 1) {
            println("");
            println("====================================");
            println("=======Fuel Station System==========");
            println("====================================");
            println("");
            println("100 or VFQ:\t View all Fuel Queues.");
            println("101 or VEQ:\t View all Empty Queues.");
            println("102 or ACQ:\t Add customer to a Queue.");
            println("103 or RCQ:\t Remove a customer from a Queue.");
            println("104 or PCQ:\t Remove a served customer.");
            println("105 or VCS:\t View Customers Sorted in alphabetical
order");
            println("106 or SPD:\t Store Program Data into file.");
            println("107 or LPD:\t Load Program Data from file.");
            println("108 or STK:\t View Remaining Fuel Stock.");
            println("109 or AFS:\t Add Fuel Stock.");
            println("999 or EXT:\t Exit the Program.");
            println("110 or IFQ:\t print the income of each Fuel queue.");
            println("");

            print("Please Enter action code : ");
            String code = fuelManagement.input.next();

            if (code.equals("100") || code.equals("VFQ") ||
code.equals("vfq")) {
                fuelManagement.viewQueue();
            } else if (code.equals("101") || code.equals("VEQ") ||
code.equals("veq")) {
                fuelManagement.viewEmptyQueue();
            } else if (code.equals("102") || code.equals("ACQ") ||
code.equals("acq")) {
                fuelManagement.addCustomer();
            } else if (code.equals("103") || code.equals("RCQ") ||
code.equals("rcq")) {
                fuelManagement.removeCustomer();
            } else if (code.equals("104") || code.equals("PCQ") ||
code.equals("pcq")) {
                fuelManagement.removeServedCustomer();
            } else if (code.equals("105") || code.equals("VCS") ||
code.equals("vcs")) {
                fuelManagement.viewCustomerSorted();
            } else if (code.equals("106") || code.equals("SPD") ||
code.equals("spd")) {
                fuelManagement.storeDataToFile();
            } else if (code.equals("107") || code.equals("LPD") ||
code.equals("lpd")) {
                fuelManagement.loadDataInFile();
            } else if (code.equals("108") || code.equals("STK") ||
code.equals("stk")) {
                fuelManagement.viewRemainingFuelStock();
            } else if (code.equals("109") || code.equals("AFS") ||
code.equals("afs")) {
                fuelManagement.addFuelStock();
            } else if (code.equals("000") || code.equals("SRV") ||
code.equals("srv")) {
                fuelManagement.FuelServe();
            } else if (code.equals("110") || code.equals("IFQ") ||
code.equals("ifq")) {
                fuelManagement.PrintQIncome();
            } else if (code.equals("999") || code.equals("EXT") ||
code.equals("ext")) {
                break;
            } else {
                println("Invalid Option!!");
            }
        }
    }
    private void viewQueue() {
        println("----------------------------");
        println("---------View Queue---------");

        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];
            println("\n Queue No. " + (q + 1) + " customers:");

            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                String cusName = passenger == null ? "N/A" :
passenger.firstName + " " + passenger.secondName + " - " +passenger.noOfLiters
+ "L";
                println(cusName);
            }
        }
        println("----------------------------");

    }

    private void viewEmptyQueue() {
        println("----------------------------");
        println("----View All Empty Queue----");

        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];

            boolean isEmpty_Q = true;
            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                isEmpty_Q = passenger == null;
                if (!isEmpty_Q) {
                    break;
                }
            }
            if (isEmpty_Q) {
                print("Queue " + (q + 1) + " is empty\n");
            }
        }
        println("----------------------------");
    }

    private void addCustomer() {

        int minQNo = 0;
        int minLength = -1;
        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQ = feulQ[q];

            for (int c = 0; c < fuelQ.passengers.length; c++) {
                Passenger passenger = fuelQ.passengers[c];
                if (passenger == null) {
                    if (minLength > c) {
                        minLength = c;
                        minQNo = q;
                    } else if (minLength == -1) {
                        minLength = c;
                        minQNo = q;
                    }
                    break;
                }
                if (fuelQ.passengers.length - 1 == c) {
                    minQNo = q + 1;
                }
            }
        }

        print("Please Enter Customer First Name:");
        String fname = input.next();

        print("Please Enter Customer Second Name:");
        String sname = input.next();

        print("Please Enter Vehicle  No:");
        String vNo = input.next();

        print("Please Enter Required QTY (Liters):");
        double rQTY;
        if (input.hasNextDouble()) {
            rQTY = input.nextDouble();
        } else {
            println("Invalid Required QTY Entered!");
            return;
        }
        if (minQNo >= feulQ.length) {
            waitingList.add(new Passenger(fname, sname, vNo, rQTY));
            println("\nSuccessfully added a customer to waiting list");
            return;
        }
        int locationIndex = FindQueueLocation(minQNo);
        FuelQueue fuelQueue = feulQ[minQNo];
        if (locationIndex == -1) {
            println("\n Invalid Location found");
            return;
        }
        fuelQueue.passengers[locationIndex] = new Passenger(fname, sname, vNo,
rQTY);
        println("\nSuccessfully added a customer to queue");

        //check fuel estimation
        double fuel_need = 0;

        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQ = feulQ[q];

            for (int c = 0; c < fuelQ.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                if (passenger != null) {
                    fuel_need += passenger.noOfLiters;
                }
            }

            if ((FuelStock - fuel_need) <= 500) {
                println("\n\tWarning - Reached 500L of fuel!");
            }
        }
    }

    private int FindQueueLocation(int Q_number) {

        if (feulQ.length < Q_number || Q_number < 0) {
            println("Invalid Queue Number!");
            return -1;
        }
        FuelQueue fuelQueue = feulQ[Q_number];
        for (int i = 0; i < fuelQueue.passengers.length; i++) {
            if (fuelQueue.passengers[i] == null) {
                return i;
            }
        }
        return 0;
    }

    private void removeCustomer() {
        print("Please Enter Queue No:");
        int Q_number = -1;
        if (input.hasNextInt()) {
            Q_number = input.nextInt();
        } else {
            println("Invalid Queue Number Entered!");
            return;
        }

        if (Q_number > 3 || Q_number < 1) {
            println("Invalid Queue Number Entered!");
            return;
        }
        //----
        print("Please Enter Location No:");
        int Q_location = -1;
        if (input.hasNextInt()) {
            Q_location = input.nextInt();
        } else {
            println("Invalid Queue Location!");
            return;
        }

        if (Q_location > 6 || Q_location < 1) {
            println("Invalid Queue Location Entered!");
            return;
        }
        feulQ[Q_number - 1].passengers[Q_location - 1] = null;

        println("Customer Removed Successfully");
    }

    private void removeServedCustomer() {
        println("Strat Served Customer Remove Process..");
        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];
            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                if (passenger == null) {
                    continue;
                }
                if (!passenger.status.equals("New")) {
                     fuelQueue.passengers[c] = null;
                }
            }
        }

        println("Reordering Queue...");
        //reset queue
        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];
            int lastEmptyIndex = -1;
            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];

                boolean flag = passenger == null;
                if (!flag && lastEmptyIndex != -1) {
                    fuelQueue.passengers[lastEmptyIndex] =
fuelQueue.passengers[c];
                    fuelQueue.passengers[c] = null;
                }
                if (flag) {
                    lastEmptyIndex = c;
                }
            }
        }
        //add waiting list to q
        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];
            int lastEmptyIndex = -1;
            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                if (passenger == null && !waitingList.isEmpty()) {
                    fuelQueue.passengers[c] = waitingList.get(0);
                    waitingList.remove(0);
                }
                if (waitingList.isEmpty()) {
                    break;
                }
            }
        }
        println("End Served Customer Remove Process!");

    }

    private void viewCustomerSorted() {

        String[] customers = new String[feulQ.length *
feulQ[0].passengers.length];
        int ind = 0;

        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];

            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                if (passenger == null) {
                    continue;
                }
                customers[ind++] = passenger.firstName + " " +
passenger.secondName;
            }
        }
        for (int i = 0; i < 18; i++) {
            for (int j = 0; j <= i; j++) {
                if (customers[i] == null || customers[j] == null) {
                    continue;
                }
                if (customers[i].compareTo(customers[j]) < 0) //compares two
elements of the array
                {
                    String temp = customers[i];
                    customers[i] = customers[j];
                    customers[j] = temp;  //rearranged in a way to make sure
it is in alphabetical order
                }
            }
        }
        println("-----Customer List------\n");
        boolean flagHasCustoemr = false;
        for (int i = 0; i < 18; i++) {
            if (customers[i] == null) {
                continue;
            }
            flagHasCustoemr = true;
            println("\t" + (i + 1) + " - " + customers[i]);
        }
        if (!flagHasCustoemr) {
            println("!!!Empty Queue Customers!!!");
        }
    }

    private void storeDataToFile()  {

        try {
            FileWriter writer = new FileWriter("DetailsOfFuelStation");
            writer.write("Fuel remaining stock: "+ FuelStock);
            for (int i = 0; i < feulQ.length; i++) {
                writer.write("\nQueue "+ (i+1) +" data\n");
                for (int j = 0; j < feulQ[i].passengers.length; j++) {
                    writer.write(feulQ[i].passengers[j]+"\n");
                }
                writer.write("");
            }
            writer.close();
            System.out.println("Successfully stored data into the file.");
        }
        catch (IOException e) {     //Runs if there was an error in file
creating or writing
```

```
            System.out.println("An error occurred while storing data into the
file. Please try again later.");
            e.printStackTrace();    //Tool used to handle exceptions and
errors (gives the line number and class name where exception happened)
        }
    }

    private void loadDataInFile() throws IOException, ClassNotFoundException {

        String fileName = "fuelStationClassV.txt";
        FileInputStream fin = new FileInputStream(fileName);
        ObjectInputStream ois = new ObjectInputStream(fin);
        FuelManagement fuelManagement = (FuelManagement) ois.readObject();
        ois.close();
        this.FuelStock = fuelManagement.FuelStock;
        this.ServedQTY = fuelManagement.ServedQTY;
        this.feulQ = fuelManagement.feulQ;
        this.fuelPrice = fuelManagement.fuelPrice;
    }

    private void viewRemainingFuelStock() {
        println("\n\tRemaining Fuel Stock :- " + FuelStock + "L \n");
    }

    private void addFuelStock() {
        println("Enter Fuel Qty:");
        if (input.hasNextDouble()) {
            double qty = input.nextDouble();
            if (qty + FuelStock > 6600) {
                println("Can't add fuel stock!, exceed capacity.");
                return;
            }
            FuelStock += qty;
            println("Current Stock - " + FuelStock);
        } else {
            println("Invalid Value Entered!");
        }
    }

    private void FuelServe() {
        print("Please Enter Queue No:");
        int Q_number = -1;
        if (input.hasNextInt()) {
            Q_number = input.nextInt();
        } else {
            println("Invalid Queue Number Entered!");
            return;
        }

        if (Q_number > feulQ.length || Q_number < 1) {
            println("Invalid Queue Number Entered!");
            return;
        }

        double qty = 0;
        for (int c = 0; c < feulQ[Q_number-1].passengers.length; c++) {
            Passenger passenger = feulQ[Q_number-1].passengers[c];
            if (passenger != null && passenger.status.equals("New")) {
                if (passenger.noOfLiters > FuelStock) {
                    println("This QTY is not available!\n" + "Currnet QTY - "
+ FuelStock + "L");
                    return;
                }
                qty = passenger.noOfLiters;
                passenger.status = "Served";
                break;
            }
        }

        FuelStock -= qty;
        println("\n\tFuel Served to customer!");
    }

    public void PrintQIncome() {
        for (int q = 0; q < feulQ.length; q++) {
            FuelQueue fuelQueue = feulQ[q];
            double income = 0;
            for (int c = 0; c < fuelQueue.passengers.length; c++) {
                Passenger passenger = fuelQueue.passengers[c];
                if (passenger == null) {
                    continue;
                }
                income += passenger.noOfLiters * fuelPrice * 1.0;
            }
            println("Queue " + (q + 1) + "Estimate price Rs. " + income);
        }
    }

    private static void print(String text) {
        System.out.print(text);
    }

    private static void println(String text) {
        System.out.println(text);
    }
}
```

Passenger.java

```
package fuelmanagement;

public class Passenger {

    public String firstName;
    public String secondName;
    public String vehicleNo;
    public double noOfLiters;
    public String status;

    public Passenger() {
    }
    public Passenger(String firstName, String secondName, String vehicleNo,
double noOfLiters) {
        this.firstName = firstName;
        this.secondName = secondName;
        this.vehicleNo = vehicleNo;
        this.noOfLiters = noOfLiters;
        this.status= "New";
    }

    @Override
    public String toString() {
        return "Passenger{" +
                "firstName='" + firstName + '\'' +
                ", secondName='" + secondName + '\'' +
                ", vehicleNo='" + vehicleNo + '\'' +
                ", noOfLiters=" + noOfLiters +
                ", status='" + status + '\'' +
                '}';
    }
}
```

FuelQueue.java

```
package fuelmanagement;

public class FuelQueue {
    public Passenger[]  passengers= new Passenger[6];

}
```