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	: **7th of August 2022**
Student ID	: **20210887 / w1912890**
Student First Name	: **Lakindu**
Student Surname	: **Indipa**

"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	: **Lakindu Indipa**
Student ID	: **20210887 / w1912890**

Test Cases

Task 1 - Array Version:

	Test Case	Expected Result	Actual Result	Pass/Fail
1	View All Queues After the program starts, 100 or VFQ	Displays 'null' for all queues.	Displayed 'null' for all queues.	Pass
2	View All Empty Queues 101 or VEQ	Displays all the empty queues.	Displayed all the empty queues.	Pass
3	Add a customer to a Queue 102 or ACQ **Enter The Queue Number (1/2/3): 2** **Enter the name of the customer : Sarith**	"Sarith" should be added to Queue 2. Displays "Sarith was added to the Queue 2 successfully!" 10L from the stock should be reduced.	"Sarith" is added to Queue 2. "Sarith was added to the Queue 2 successfully!" is displayed. The stock was reduced by 10L.	Pass
4	Add a customer to a Queue by **inputting a LETTER as the queue number** 102 or ACQ **Enter The Queue Number (1/2/3): w**	Displays "The Input can only be a number! Please Try Again!". Program asks again the queue number.	Displayed "The Input can only be a number! Please Try Again!". The queue number is asked again by the program.	Pass
5	Add a customer to a Queue by **inputting a number which is NOT 1 or 2 or 3.** **Enter The Queue Number (1/2/3): 4** OR **Enter The Queue Number (1/2/3): -1**	Displays "The Queue number can only be 1 or 2 or 3! Please Try Again!". Program asks again the queue number.	Displayed "The Queue number can only be 1 or 2 or 3! Please Try Again!". The queue number is asked again by the program.	Pass
6	Add a customer to a Queue by **inputting a NUMBER as the customer name.** 102 or ACQ **Enter The Queue Number (1/2/3): 1** **Enter the name of the customer : 5**	Displays "The Name can only contain LETTERS! Please Try Again!". Program asks again for the name	Displayed "The Name can only contain LETTERS! Please Try Again!" The name is asked again by the program.	Pass
7	Add a customer to a Queue by **inputting NULL as the customer name.** 102 or ACQ **Enter The Queue Number (1/2/3): 1** **Enter the name of the customer : null**	Displays "Oops...null is not a valid name since it is a reserved keyword in JAVA. Please Try Again!". Program asks again for the name	Displayed "Oops...null is not a valid name since it is a reserved keyword in JAVA. Please Try Again!" The name is asked again by the program.	Pass
8	(After adding 3 more customers to Queue 2) Remove a customer from a Queue 103 or RCQ	The 2nd customer of the Queue 2 should be removed and the customers behind him should be shifted to the front.	The 2nd customer of the Queue 2 is removed and all the other customers behind have shifted to the front.	Pass

	Enter The Queue Number (1/2/3): 2 **Enter the position (0/1/2/3/4/5) : 1** ***NOTE:** The validations for the inputs are same as the **Test Cases 4 and 5.**	Displays "The customer at the position 1 was removed from the Queue 2 successfully!" 10L should be added back to the stock.	Displayed "The customer at the position 1 was removed from the Queue 2 successfully!" Stock was increased back by 10L.	
9	Remove a customer from a Queue who **does not exist** 103 or RCQ **Enter The Queue Number (1/2/3): 1** **Enter the position (0/1/2/3/4/5) : 3**	Displays "Oops...There's no customer at the position 3 in the Queue 1"	Displayed "Oops...There's no customer at the position 3 in the Queue 1"	Pass
10	Remove a served customer 104 or PCQ **Enter The Queue Number (1/2/3): 1** ***NOTE:** The validations for the input are same as the **Test Cases 4 and 5.**	Displays "The first customer was removed from the Queue 1 successfully!". The 1st customer in the queue 1 should be removed and all the other customers in the behind should be shifted to the front.	Displayed "The first customer was removed from the Queue 1 successfully!". The 1st customer is removed from the queue 1 and all the other customers in the behind is shifted to the front.	Pass
11	Remove a served customer from a Queue who **does not exist** 104 or PCQ **Enter The Queue Number (1/2/3): 3**	Displays "Oops...There are no customers in the Queue 3".	Displayed "Oops...There are no customers in the Queue 3".	Pass
12	View Customers Sorted in alphabetical order 105 or VCS	All the customers in the queues should be displayed in the alphabetical order.	The customers in all the queues are displayed in the alphabetical order.	Pass
13	Store Program Data into file. 106 or SPD	Displays "The data is processing...The data was stored successfully into the file." Program data should be stored in a text file called "Program_Data.txt".	Displayed "The data is processing...The data was stored successfully into the file." Program data is stored in a text file called "Program_Data.txt".	Pass
14	Load Program Data from file. 107 or LPD	Displays "The data is loading... The data was loaded successfully!". The loaded data should replace the existing data in the program.	Displayed "The data is loading... The data was loaded successfully!". The existing data is replaced by the loaded data.	Pass
15	Load Program Data from a file **before storing anything to the text file.** 107 or LPD	Displays "Oops...The text file is not available".	Displayed "Oops...The text file is not available".	Pass

16	Load Program Data from a file by **erasing everything in the text file.** 107 or LPD	Displays “Oops...There's no previously available data in the text file.”	Displayed “Oops...There's no previously available data in the text file.”	Pass
17	View Remaining Fuel Stock. 108 or STK	Displays the remaining stock which was calculated when adding a customer and removing a customer from a specific location.	The remaining stock is displayed correctly as, Remaining Fuel Stock : 6540 Liters out of 6600 Liters”	Pass
18	Add Fuel Stock. 109 or AFS **Enter the amount of the new stock : 30**	Displays the stock with the newly added fuel stock.	Displayed “The new fuel stock is 6570 Liters”.	Pass
19	Add Fuel Stock **which goes above the stock limit.** 109 or AFS **Enter the amount of the new stock : 100**	Displays “The amount of the fuel stock cannot exceed 6600 Liters! Please Try Again!” Program asks for the new stock again.	Displayed “The amount of the fuel stock cannot exceed 6600 Liters! Please Try Again!” The new stock is asked again by the program.	Pass
20	Add Fuel Stock as a **negative number.** 109 or AFS **Enter the amount of the new stock : -50**	Displays “The amount of the fuel stock cannot be a NEGATIVE NUMBER! Please Try Again!”. Program asks for the new stock again.	Displayed “The amount of the fuel stock cannot be a NEGATIVE NUMBER! Please Try Again!”. The new stock is asked again by the program.	Pass
21	Add Fuel Stock as a **Letter.** 109 or AFS **Enter the amount of the new stock : r**	Displays “The amount of the fuel stock can only be a NUMBER! Please Try Again!”. Program asks for the new stock again.	Displayed “The amount of the fuel stock can only be a NUMBER! Please Try Again!”. The new stock is asked again by the program.	Pass
22	Exit the Program. 999 or EXT	Displays “Exit the Program.” Program should end.	Displayed “Exit the Program.” Program is ended.	Pass
23	Enter an option which is NOT in the menu **Enter your choice : 99** OR **Enter your choice : -1**	Displays “Oops...There's no such a choice. Please read the instructions carefully and TRY AGAIN!”. The menu should display again.	Displayed “Oops...There's no such a choice. Please read the instructions carefully and TRY AGAIN!”. The menu is displayed again.	Pass
24	Add a customer to a Queue **when all the queues are full** 102 or ACQ **Enter The Queue Number (1/2/3): 2** **Enter the name of the customer : man**	Displays “Oops...The Queue is full!”.	Displayed “Oops...The Queue is full!”.	Pass

Task 2/3/4 - Class Version:

	Test Case	Expected Result	Actual Result	Pass/Fail
25	View All Queues After the program starts, 100 or VFQ	Displays 'EMPTY' for all queues including the waiting queue.	Displayed 'EMPTY' for all Queues.	Pass
26	View All Empty Queues 101 or VEQ	Displays all the empty queues.	Displayed all the empty queues.	Pass
27	Add a passenger called "Sarith" 102 or ACQ **Enter The First Name : Sarith** **Enter The Second Name : Wijesundera** **Enter The Vehicle Number : 1020** **Enter The Number of Liters Required : 10** ***NOTE:** Validations for first name and last name is same as the **Test Cases 6 and 7.** Validation for the vehicle number is same as the **Test Case 7.**	Displays "Sarith Wijesundera was added to the Queue 1 successfully!". "Sarith" should be added to the minimum queue. The stock should be reduced by 10 The income for the queue 1 should be increased.	Displayed "Sarith Wijesundera was added to the Queue 1 successfully!". "Sarith" is added to the minimum queue. The stock was reduced by 10. Queue 1 income is increased.	Pass
28	Add a passenger by **inputting the required liters as 0 or a negative number.** 102 or ACQ **Enter The Number of Liters Required : 0** OR **Enter The Number of Liters Required : -2**	Displays "The required fuel liters cannot be less than or equal to zero! Please Try Again!". Program asks for the required liters again.	Displayed "The required fuel liters cannot be less than or equal to zero! Please Try Again!". The required liters is asked again by the program.	Pass
29	Add a passenger by **inputting the required liters as more than 10.** 102 or ACQ **Enter The Number of Liters Required : 12**	Displays "Oops...The maximum amount of liters per each customer is 10L! Please Try Again!". Program asks for the required liters again.	Displayed "Oops...The maximum amount of liters per each customer is 10L! Please Try Again!". The required liters is asked again by the program.	Pass
30	**When the stock has only 7 Liters,** Add a passenger by **inputting the required liters as more than the existing stock.** 102 or ACQ **Enter The Number of Liters Required : 8**	Displays "The required fuel liters cannot exceed the existing amount of liters in the stock. Please Try Again!". Program asks for the required liters again.	Displayed "The required fuel liters cannot exceed the existing amount of liters in the stock. Please Try Again!". The required liters is asked again by the program.	Pass

31	Add a passenger by **inputting the required liters as a letter/word.** 102 or ACQ **Enter The Number of Liters Required : er**	Displays "The required fuel liters can only be a NUMBER! Please Try Again!". Program asks for the required liters again.	Displayed "The required fuel liters can only be a NUMBER! Please Try Again!". The required liters is asked again by the program.	Pass
32	Remove a customer from a Queue. 103 or RCQ **Enter The Queue Number (1/2/3/4/5): 4** **Enter the position (0/1/2/3/4/5) : 3** ***NOTE:** The validations for the inputs are same as the **Test Cases 4 and 5.**	Displays "The customer at the position 3 was removed from the Queue 4 successfully! The next customer at The Waiting Queue was added to the empty space successfully!". The mentioned customer in the queue should be removed and the customers behind that should be shifted to the front. The number of required liters of the removed customer should be added back to the stock. The income based on the required liters of the removed customer should be reduced from the income of the queue. If there are customers in the waiting queue, the next customer should be added at the end of the Queue 4. The queue income should increase.	Displayed "The customer at the position 3 was removed from the Queue 4 successfully! The next customer at The Waiting Queue was added to the empty space successfully!". The customers behind him is shifted to the front. The stock is added back. The income is reduced based on the required liters of the removed customer. The first customer in the waiting queue is added at the end of the queue 4. The queue income is increased.	Pass
33	Remove a served customer. 104 or PCQ **Enter The Queue Number (1/2/3/4/5): 2** ***NOTE:** The validations for the input are same as the **Test Cases 4 and 5.**	Displays "The first customer was removed from the Queue 2 successfully! The next customer at The Waiting Queue was added to the empty space successfully!" The 1st customer in the queue 2 should be removed and all the other customers in the behind should be shifted to the front. If there are customers in the waiting queue, the next customer should be added at the end of the Queue 2. The queue income should increase.	Displayed "The first customer was removed from the Queue 2 successfully! The next customer at The Waiting Queue was added to the empty space successfully!". The 1st customer is removed from the queue 2 and all the other customers in the behind is shifted to the front. The first customer in the waiting queue is added at the end of the	Pass

			queue 2. The queue income is increased.	
34	View Customers Sorted in alphabetical order 105 or VCS **NOTE:** If there are customers with the same first name, then those will be sorted based on their last name.	All the customers in the queues should be displayed in the alphabetical order.	The customers in all the queues are displayed in the alphabetical order.	Pass
	NOTE: 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.** All the above-mentioned options are **same as the Test Cases in the array version** (Test Cases 13 – 21)**.** For an addition, only the incomes of each queue are stored and loaded from the text file.			
35	Income of each Fuel Queue. 110 or IFQ	Incomes of all the 5 queues should be displayed separately.	All the incomes are displayed correctly.	Pass
36	Add a customer to a Queue **when all the queues are full** 102 or ACQ **Enter The First Name : Namal** **Enter The Second Name : Silva** **Enter The Vehicle Number : 3001** **Enter The Number of Liters Required : 4**	Displays “Oops...All The Fuel Queues are Full! You'll be added to The Waiting Queue!” Displays “Namal Silva was added to the Waiting Queue successfully!” Customer should be added to the waiting queue. Stock should be reduced.	Displayed “Oops...All The Fuel Queues are Full! You'll be added to The Waiting Queue!” Displayed “Namal Silva was added to the Waiting Queue successfully!” Customer is added to the waiting queue. Stock is reduced.	Pass
37	Exit the Program. 999 or EXT	Displays “Exit the Program.” Program should end. The GUI should automatically appear.	Displayed “Exit the Program.” Program is ended. The GUI is appeared automatically.	Pass
38	**GUI:** Clicks on ‘View All Queues’ button	The menu window should be closed and a new window should be opened.	The menu window is being closed and a new window is being opened to display the status of the queues.	Pass
39	**GUI:** Clicks on ‘Update’ button	All the data including the customer info, stock and incomes of each fuel queue should be displayed on each slot.	All the required data is successfully displayed.	Pass

40	**GUI:** Clicks on ‘Update’ button when **there are no data available.**	An error alert window should appear saying that “There are no previously available data”.	An error alert window appeared and displayed that “There are no previously available data”.	Pass
41	**GUI:** Clicks on ‘Go Back’ button	The View All Queues window should be closed and the Menu window should be opened back.	The View All Queues window is being closed and the Menu window is being Opened back.	Pass
42	**GUI:** Clicks on ‘Search a Customer’ button	The menu window should be closed and a new window should be opened.	The menu window is being closed and a new window is being opened with a search bar.	Pass
43	**GUI:** Types a first name in the search bar and clicks on the ‘Search’ button.	If it’s a match, all the customers with that first name should be displayed in an information alert window one after one. If it’s not a match, an information alert window appears and displays that “No Result. Oops…There are no customers with that name”.	All the customers with that first name are displayed in an information alert window one after one. An information alert window appeared and displayed that “No Result. Oops…There are no customers with that name”.	Pass
44	**GUI:** Clicks on the ‘Exit’ button.	The GUI window should be closed.	The GUI window is closed.	Pass

Discussion

Array Version: (Test Cases 1 to 24 inclusive)

Test Cases **1** and **2** display the view all queues and the view all empty queues respectively. The test cases **3,4,5,6,7 and 24** show all the possible validations when adding a customer to a queue. Test cases **8 and 9** contain the possible outcomes when removing a customer from a queue while test cases **10 and 11** contains the possible outcomes when removing a served customer. Test case **12** displays the sorting order. The test cases **13 to 21** include all the possible outcomes and the validations of Store Program Data into a file, Load Program Data from a file, View Remaining Fuel Stock and Add Fuel Stock which is mostly the same in the class version also. The test case **22** is about exiting from the program and the test case **23** displays the validation outcome when the user enters an invalid menu option. Therefore, all these test cases cover all the possible outcomes in the array version.

Class Version: (Test Cases 25 to 44 inclusive)

Test Cases **25 and 26** display the view all queues and the view all empty queues respectively. The test cases **27 to 31 inclusive** show all the possible validations when adding a customer to a queue. Test cases **32 and 33** are about removing a customer from a specific location and removing a served customer respectively. Test case **34** shows the alphabetical order in the class version. Since the menu options, *106 to 109 inclusive* are the same as the test cases in the array version, those are not being repeated again in the class version. Test case **35** displays the incomes of all the fuel queues which is a unique feature in the class version. Test case **36** is about adding customers to the waiting list when all the other queues are full. Test case **37** is about the GUI which appears when exiting the program. Test case **38 to 44** covers all the possible outcomes and validations in the GUI.

Assumptions:

I assumed when a customer is added to a queue, the required number of liters should be reduced from the stock. At the same time, when a customer is being removed from a specific location, the stock should be added back to the stock and the queue income should be reduced. As per my assumptions, I made my GUI appear after the program ends.

Code :

Array Version (Task_1.java)

```
package com.example.prog_cw;

//Importing the packages
import java.util.*;
import java.io.*;
public class Task_1 {
    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);   //Defining a variable for
the Scanner
        String[] queue_1 = new String[6];      //Initializing 3 arrays for
queues
        String[] queue_2 = new String[6];
        String[] queue_3 = new String[6];
        boolean[] stock_Update = new boolean[1];    //Initializing a separate
array to get a boolean value to decide whether the fuel stock should be
updated or not
        int stock = 6600;      //Defining the fuel stock
        //Displaying the menu
        mainLoop:
        while (true){
            System.out.println("\n-------------------------------------------
---------------------------------");
            System.out.println("Menu of The Fuel Queue Management System");
            System.out.println("\nEnter,");
            System.out.println("\t100 or VFQ : View all Fuel Queues.");
            System.out.println("\t101 or VEQ : View all Empty Queues.");
            System.out.println("\t102 or ACQ : Add customer to a Queue.");
            System.out.println("\t103 or RCQ : Remove a customer from a
Queue.");
            System.out.println("\t104 or PCQ : Remove a served customer.");
            System.out.println("\t105 or VCS : View Customers Sorted in
alphabetical order");
            System.out.println("\t106 or SPD : Store Program Data into
file.");
            System.out.println("\t107 or LPD : Load Program Data from
file.");
            System.out.println("\t108 or STK : View Remaining Fuel Stock.");
            System.out.println("\t109 or AFS : Add Fuel Stock.");
            System.out.println("\t999 or EXT : Exit the Program.");

            //Getting the user choice for the menu
            System.out.print("\nEnter your choice : ");
            String choice = input.next();

            //Using switch case to call the methods
            switch (choice.toUpperCase()){
                case "100":
                case "VFQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View all Fuel Queues.");
                    all_Queues(queue_1,queue_2,queue_3);
                    break;
                case "101":
                case "VEQ":
                    System.out.println("\n--------------------------------------
------------------------------------------");
                    System.out.println("View all Empty Queues.");
                    empty_Queues(queue_1,queue_2,queue_3);
                    break;
                case "102":
                case "ACQ":
                    System.out.println("\n--------------------------------------
------------------------------------------");
                    System.out.println("Add customer to a Queue.\n");

System.out.println(add_Customer(stock_Update,input,queue_1,queue_2,queue_3));
                    if (stock_Update[0]){   //If true, the stock will be
updated (reduce by 10)
                        stock -= 10;
                    }
                    if (stock <= 500){      //If the stock is less than or
equals to 500, a warning message will be displayed
                        System.out.println("\nWARNING:\n\tThe fuel stock has
reached a value of 500 Liters");
                    }
                    //I assumed when a customer is added to a queue, that
customer is already being served. Therefore, I removed 10 liters from the
stock
                    break;
                case "103":
                case "RCQ":
                    System.out.println("\n--------------------------------------
------------------------------------------");
                    System.out.println("Remove a customer at a specific
location from a Queue.\n");

System.out.println(remove_Customer(input,stock_Update,queue_1,queue_2,queue_3
));
                    if (stock_Update[0]){   //If true, the stock will be
updated (increase by 10)
                        if ((stock + 10) > 6600){
                            stock = 6600;
                        }else {
                            stock += 10;
                        }
                    }
                    break;
                case "104":
                case "PCQ":
                    System.out.println("\n--------------------------------------
------------------------------------------");
                    System.out.println("Remove a served customer.\n");

System.out.println(remove_Served_Customer(input,queue_1,queue_2,queue_3));
                    //I assumed that when removing a served customer from a
queue, it's practical to remove the first customer in the queue.
                    break;
                case "105":
                case "VCS":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View Customers Sorted in alphabetical
order\n");
                    System.out.println("Queue 1:");
                    sort_Customers(queue_1);
                    System.out.println("\nQueue 2:");
                    sort_Customers(queue_2);
                    System.out.println("\nQueue 3:");
                    sort_Customers(queue_3);
                    break;
                case "106":
                case "SPD":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Store Program Data into file.\n");

System.out.println(store_Data(stock,queue_1,queue_2,queue_3));
                    break;
                case "107":
                case "LPD":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Load Program Data from file.\n");
                    stock = load_Data(stock,queue_1,queue_2,queue_3);
                    break;
                case "108":
                case "STK":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View Remaining Fuel Stock.\n");
                    System.out.println("Remaining Fuel Stock : " + stock + "
Liters out of 6600 Liters");
                    break;
                case "109":
                case "AFS":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Add Fuel Stock.\n");
                    stock = add_Stock(input, stock);
                    System.out.println("NOTE:\n\tThe new fuel stock is " +
stock + " Liters");
                    break;
                case "999":
                case "EXT":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Exit the Program.");
                    break mainLoop;
                default:
                    System.out.println("ERROR:\n\tOops...There's no such a
choice.\n\tPlease read the instructions carefully and TRY AGAIN!");
            }
        }
    }
    //View all Fuel Queues
    public static void all_Queues(String[] queue_1, String[] queue_2,
String[] queue_3){
        //This method displays all the queues
        System.out.println("\nQueue 1:\n" + Arrays.toString(queue_1));
        System.out.println("\nQueue 2:\n" + Arrays.toString(queue_2));
        System.out.println("\nQueue 3:\n" + Arrays.toString(queue_3));
    }

    //View all Empty Queues
    public static void empty_Queues(String[] queue_1, String[] queue_2,
String[] queue_3){
        //This method checks if there are any empty spaces in each of the
arrays.
        //If there is at least one empty space, then that specific array will
be considered and displayed as an empty queue.
        for (String i: queue_1){
            if (i == null){
                System.out.println("\nQueue 1:\n" +
Arrays.toString(queue_1));
                break;
            }
        }
        for (String i: queue_2){
            if (i == null){
                System.out.println("\nQueue 2:\n" +
Arrays.toString(queue_2));
                break;
            }
        }
        for (String i: queue_3){
            if (i == null){
                System.out.println("\nQueue 3:\n" +
Arrays.toString(queue_3));
                break;
            }
        }
    }

    //Get The Queue Number
    public static int get_Queue_Number(Scanner input){
        //This method validates the user input for the queue number which can
only be 1/2/3.
        //This method is used to get the queue number when adding a customer,
removing a customer and removing a served customer.
        int queue;
        while (true) {
            try {
                System.out.print("Enter The Queue Number (1/2/3): ");
                queue = input.nextInt();
                if (!(1 <= queue && queue <= 3)) {     //If the queue number
is not a number between 1 and 3 inclusive, then an error message will be
displayed.
                    System.out.println("ERROR:\n\tThe Queue number can only
be 1 or 2 or 3!\n\tPlease Try Again!\n");
                    input.nextLine();
                }else {
                    break;
                }
            } catch (Exception e) {     //If the user input for the queue
number is not an integer value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe Input can only be a
number!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
        return queue;
    }

    //Updating the array after a customer was added
    public static int update_Array(String[] queue, String name, boolean[]
stock_Update, int is_Full){
        //This method adds a new customer to the first empty space in the
queue
        for (int i = 0; i < queue.length; i++){
            if (queue[i] == null){      //If the space is empty, then it will
replace that empty space with the customer name.
                queue[i] = name;
                stock_Update[0] = true;    //Since the stock_Update has a true
value, the fuel stock will be reduced by 10 in the main method.
                break;
            }else {
                is_Full++;       //If it's not an empty space, then the
is_Full will be increased by 1
                stock_Update[0] = false;
            }
        }
        return is_Full;
    }

    //Validating the user input name
    public static boolean nameValidate(String name){
        char[] charArray = name.toCharArray();
        for (char i : charArray){
            if (!Character.isAlphabetic(i)){
                return true;
            }
        }
        return false;
    }

    //Add customer to a Queue
    public static String add_Customer(boolean[] stock_Update, Scanner input,
String[] queue_1, String[] queue_2, String[] queue_3){
        //This method gets the queue number and the customer name from the
user and adds the customer to a queue.
        int is_Full = 0;
        String name;
        int queue = get_Queue_Number(input);    //Calling the get_Queue_Number
method to validate the queue number.
        do {
            System.out.print("Enter the name of the customer : ");     //If
user inputs 'null' as the customer name, it will display an error message.
            name = input.next();
            if (name.equals("null")){
                System.out.println("NOTE:\n\tOops..." + name + " is not a
valid name since it is a reserved keyword in JAVA.\n\tPlease Try Again!\n");
            } else if (nameValidate(name)) {
                System.out.println("NOTE:\n\tThe Name can only contain
LETTERS! Please Try Again!\n");
            } else {
                break;
            }
        }while (true);
        switch (queue){
            case 1:
                is_Full = update_Array(queue_1, name, stock_Update, is_Full);
//Calling the update_Array method to update the queue.
                break;
            case 2:
                is_Full = update_Array(queue_2, name, stock_Update, is_Full);
                break;
            case 3:
                is_Full = update_Array(queue_3, name, stock_Update, is_Full);
                break;
        }
        if (is_Full == 6){      //If it's true, it will display a message
saying the queue is full.
            return "NOTE:\n\tOops...The Queue is full!";
        }else {
            return "NOTE:\n\t" + name + " was added to the Queue " + queue +
" successfully!";
        }
    }

    //Shifting the queue after a customer was removed
    public static void shift_Queue(int position, String[] queue){
        //This method shifts the remaining customers to the left, when a
customer was removed.
        if (position == 5){
            queue[position] = null;
        }else {
            for (int i = 0; i < queue.length; i++){
                if ((position + i + 1) == 5){   //This only becomes true for
the 4th index. If it's true, the index 4 will be replaced by the 5th index.
                    queue[4] = queue[5];
                    queue[5] = null;
                    break;
                }else {
                    queue[position + i] = queue[position + i + 1];    //The
index will be replaced by the index after that.
                }
            }
        }
    }

    //Remove a customer from a Queue (Specific Position)
    public static String remove_Customer(Scanner input, boolean[]
stock_Update, String[] queue_1, String[] queue_2, String[] queue_3){
        //This method removes a customer from the queue by index.
        int position;
        int queue = get_Queue_Number(input);    //Calling the
get_Queue_Number method to validate the queue number.
        while (true){
            try{
                System.out.print("Enter the position (0/1/2/3/4/5) : ");
                position = input.nextInt();
                if (!(0 <= position && position <= 5)){     //If the position
is not a number between 0 and 5 inclusive, then an error message will be
displayed.
                    System.out.println("ERROR:\n\tThe Position can only be a
number between 0 to 6 inclusive!\n\tPlease Try Again!\n");
                    input.nextLine();
                } else {
                    break;
                }
            } catch (Exception e){      //If the position is not an integer
value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe Input can only be a
number!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
        switch (queue){
            case 1:
                if (queue_1[position] != null){     //If the position is not
null, then it calls the shift_Queue method to replace the positions
                    shift_Queue(position,queue_1);
                }else {
                    stock_Update[0] = false;      //Since there's no customer
at that position, the fuel stock won't be updated.
                    return "NOTE:\n\tOops...There's no customer at the
position " + position + " in the Queue " + queue;
                }
                break;
            case 2:
                if (queue_2[position] != null){
                    shift_Queue(position,queue_2);
                }else {
                    stock_Update[0] = false;
                    return "NOTE:\n\tOops...There's no customer at the
position " + position + " in the Queue " + queue;
                }
                break;
            case 3:
                if (queue_3[position] != null){
                    shift_Queue(position,queue_3);
                }else {
                    stock_Update[0] = false;
                    return "NOTE:\n\tOops...There's no customer at the
position " + position + " in the Queue " + queue;
                }
                break;
        }
        stock_Update[0] = true;     //Since the stock_Update has a true
value, the fuel stock will be increased by 10 in the main method.
        return "NOTE:\n\tThe customer at the position " + position + " was
removed from the Queue " + queue + " successfully!";
```

```
    }

    //Remove a served customer from a Queue
    public static String remove_Served_Customer(Scanner input, String[]
queue_1, String[] queue_2, String[] queue_3) {
        //This method removes the first served customer from the queue.
        int queue = get_Queue_Number(input);    //Calling the
get_Queue_Number method to validate the queue number.
        switch (queue){
            case 1:
                if (queue_1[0] != null){
                    shift_Queue(0,queue_1);
                }else {
                    return "NOTE:\n\tOops...There are no customers in the
Queue " + queue;
                }
                break;
            case 2:
                if (queue_2[0] != null){
                    shift_Queue(0,queue_2);
                }else {
                    return "NOTE:\n\tOops...There are no customers in the
Queue " + queue;
                }
                break;
            case 3:
                if (queue_3[0] != null){
                    shift_Queue(0,queue_3);
                }else {
                    return "NOTE:\n\tOops...There are no customers in the
Queue " + queue;
                }
                break;
        }
        return "NOTE:\n\tThe first customer was removed from the Queue " +
queue + " successfully!";
    }

    //View Customers Sorted in alphabetical order
    public static void sort_Customers(String[] queue){
        //This method sorts the customer names in the alphabetical order
without using the library sort routine.
        int name_Count = 0, max_Count, clone;
        String[] sorted_Queue = new String[6];  //Initializing a new queue to
store the sorted customer names.
        for (String x: queue) {
            if (x != null) {     //This loop determines how many customer
names are there in the queue without the null values.
                name_Count++;
            }
        }
        for (String name : queue) {
            if (name != null) {
                name = name.toLowerCase();
                max_Count = 0;    //max_Count variable increases the count
when a name appears at the first in the alphabetical order than the other
names in the queue.
```

```
                clone = 0;    //clone variable counts how many repeated names
are there in the queue.
                //The below loop compares each name in the queue with all the
names in the same queue.
                for (String compare_Name : queue) {
                    if (compare_Name != null){
                        compare_Name = compare_Name.toLowerCase();
                        if (!name.equals(compare_Name)){   //If the name is
not equals to the comparison name and null, then the comparison statement
will run.
                            if (name.compareToIgnoreCase(compare_Name) < 0) {
                                max_Count++;    //If the name is at the first
in the alphabetical order than the comparison name, then it increases by 1.
                            }
                        }else if (name.equalsIgnoreCase(compare_Name)){
                            clone++;    //If the name is same as the
comparison name, then the clone increases by 1.
                        }
                    }
                }
                for (int i = 0; i < name_Count; i++) {
                    if (max_Count == ((name_Count - 1) - i)) {
                        sorted_Queue[i] = name;
                        for (int j = 0; j < clone; j++){
                            sorted_Queue[i-j] = name;   //If there are
customers with the same name, then those will be appeared next to the other.
                        }
                    }
                }
            }
        }
        System.out.println(Arrays.toString(sorted_Queue));    //Displaying
the sorted queue.
    }

    //Store data into a file
    public static String store_Data(int stock, String[] queue_1, String[]
queue_2, String[] queue_3) throws IOException{
        //This method stores the program data into a text file.
        try {
            FileWriter file = new FileWriter("Program_Data.txt");   //Creates
a text file to store data.
            file.write((Arrays.toString(queue_1)).replaceAll("[\\[\\],]","")
+ "\n");   //Converting arrays to string and replacing all the "[ , ]"
symbols with empty spaces.
            file.write((Arrays.toString(queue_2)).replaceAll("[\\[\\],]","")
+ "\n");   //Storing all the queues in the text file.
            file.write((Arrays.toString(queue_3)).replaceAll("[\\[\\],]","")
+ "\n");

            file.write(String.valueOf(stock) + "\n");   //Converting the fuel
stock to a string and storing it in the text file.
            file.close();
        }catch (Exception e){
            System.out.println("ERROR:\n\t" + e);
        }
        System.out.println("The data is processing...\n");      //Displaying
a message to the user that the process has happened successfully.
```

```
        return "NOTE:\n\tThe data was stored successfully into the file.";
    }

    //Load data from a file
    public static int load_Data(int stock, String[] queue_1, String[]
queue_2, String[] queue_3) throws IOException{
        //This method loads the data from a text file.
        int line_Count = 0;
        try {
            File file = new File("Program_Data.txt");   //Opening the text
file.
            Scanner read_File = new Scanner(file);    //Reads the content in
the text file.
            while (read_File.hasNext()){
                line_Count++;   //line_Count counts how many lines are there
in the text file.
                switch (line_Count){
                    case 1:
                        String[] toArray1 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                        for (int i = 0; i < queue_1.length; i++){
                            if (toArray1[i].equals("null")){    //In this for
loop, the values in the converted array will be cloned to the queue.
                                queue_1[i] = null;
                            } else {
                                queue_1[i] = toArray1[i];
                            }
                        }
                        break;
                    case 2:
                        String[] toArray2 = read_File.nextLine().split(" ");
                        for (int i = 0; i < queue_2.length; i++){
                            if (toArray2[i].equals("null")){
                                queue_2[i] = null;
                            } else {
                                queue_2[i] = toArray2[i];
                            }
                        }
                        break;
                    case 3:
                        String[] toArray3 = read_File.nextLine().split(" ");
                        for (int i = 0; i < queue_3.length; i++){
                            if (toArray3[i].equals("null")){
                                queue_3[i] = null;
                            } else {
                                queue_3[i] = toArray3[i];
                            }
                        }
                        break;
                    case 4:
                        stock = Integer.parseInt(read_File.nextLine());
                        break;
                }
            }
            if (line_Count == 0){       //If there are no lines in the text
file, an error message will be displayed.
                System.out.println("NOTE:\n\tOops...There's no previously
available data in the text file.");
            }else {
                System.out.println("The data is loading...\n");
//Displaying a message to the user that the process has happened
successfully.
                System.out.println("NOTE:\n\tThe data was loaded
successfully!");
            }
        }catch (IOException e){     //If there's no existing text file with
that name, an error message will be displayed.
            System.out.println("ERROR:\n\tOops...The text file is not
available");
        }
        return stock;
    }

    //Add Fuel Stock
    public static int add_Stock(Scanner input, int stock){
        //This method adds new fuel stock to the existing stock.
        while (true){
            try {
                System.out.print("Enter the amount of the new stock : ");
                int new_Stock = input.nextInt();     //Getting the amount of
the new stock from the user.
                if ((stock + new_Stock) > 6600){     //If the stock is greater
than 6600, then an error message will be displayed.
                    System.out.println("ERROR:\n\tThe amount of the fuel
stock cannot exceed 6600 Liters!\n\tPlease Try Again!\n");
                    input.nextLine();
                } else if (new_Stock < 0) {     //If the stock is a negative
number, an error message will be displayed.
                    System.out.println("ERROR:\n\tThe amount of the fuel
stock cannot be a NEGATIVE NUMBER!\n\tPlease Try Again!\n");
                    input.nextLine();
                } else {
                    stock += new_Stock;
                    break;
                }
            }catch (Exception e){      //If the fuel stock is not an integer
value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe amount of the fuel stock
can only be a NUMBER!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
        return stock;
    }
}
```

Class Version (Passenger.java)

```java
package com.example.prog_cw;

public class Passenger {
    //Defining four variables to store passenger details
    private String firstName = null;
    private String lastName = null;
    private String vehicleNo = null;
    private String liters = null;


    //Defining getters and setters
    public String getFirstName(){
        return firstName;
    }

    public void setFirstName(String newFirstName){
        this.firstName = newFirstName;
    }

    public String getLastName(){
        return lastName;
    }

    public void setLastName(String newLastName)  {
        this.lastName = newLastName;
    }

    public String getVehicleNo(){
        return vehicleNo;
    }

    public void setVehicleNo(String newVehicleNo){
        this.vehicleNo = newVehicleNo;
    }

    public String getLiters(){
        return liters;
    }

    public void setLiters(String newLiters){
        this.liters = newLiters;
    }
}
```

Class Version (FuelQueue.java)

```
package com.example.prog_cw;

import java.util.ArrayList;

public class FuelQueue {
    //Defining objects for 6 passengers
    Passenger passenger_1 = new Passenger();
    Passenger passenger_2 = new Passenger();
    Passenger passenger_3 = new Passenger();
    Passenger passenger_4 = new Passenger();
    Passenger passenger_5 = new Passenger();
    Passenger passenger_6 = new Passenger();

    //Creating an array of Passenger Objects
    Passenger[] passengerObj  = {passenger_1, passenger_2, passenger_3,
passenger_4, passenger_5, passenger_6};

    //Defining a variable to store the income of each fuel queue
    private int queue_income = 0;

    //Defining the getter and the setter for the income
    public int getQueue_income() {
        return queue_income;
    }
    public void setQueue_income(int queue_income) {
        this.queue_income = queue_income;
    }

    //This method is to display all the passengers in a queue.
    public void viewAllQueues(){
        for (int i = 0; i < 6; i++) {
            System.out.println("\nCustomer " + (i+1));
            if (passengerObj[i].getFirstName() == null){
                System.out.println("EMPTY");
            }else {
                System.out.println("\tFirst Name     : " +
passengerObj[i].getFirstName());
                System.out.println("\tLast Name      : " +
passengerObj[i].getLastName());
                System.out.println("\tVehicle No     : " +
passengerObj[i].getVehicleNo());
                System.out.println("\tRequired Liters: " +
passengerObj[i].getLiters());
            }
        }
    }

    //This method is to display all the empty passengers in the queue.
    public void emptyQueues(){
        int maxCount = 0;
        for (int i = 0; i < 6; i++){
            if (passengerObj[i].getFirstName() == null){
                System.out.println("\nCustomer " + (i+1));
                System.out.println("EMPTY");
            }else {
                maxCount++;
            }
        }
        if (maxCount == 6){
            System.out.println("NOTE:\n\tQueue is Full!");
        }
    }


    //Get null count of the queue
    public int getNullCount(){
        //This method counts the null values in each queue
        int nullCount = 0;
        for (int i = 0; i < 6; i++){
            if (passengerObj[i].getFirstName() == null){
                nullCount += 1;
            }
        }
        return nullCount;
    }

    //This method assigns all the details of the customer to the passenger
object
    public void addCustomerToQueue(int queueNo,String[] details){
        for (int i = 0; i < 6; i++){
            if (passengerObj[i].getFirstName() == null){
                passengerObj[i].setFirstName(details[0]);
                passengerObj[i].setLastName(details[1]);
                passengerObj[i].setVehicleNo(details[2]);
                passengerObj[i].setLiters(details[3]);
                setQueue_income(getQueue_income() + (430 *
Integer.parseInt(details[3])));

                if (queueNo == 1){
                    System.out.println("NOTE:\n\t" +
passengerObj[i].getFirstName() + " " + passengerObj[i].getLastName() + " was
added to the Queue 1 successfully!");
                }else if (queueNo == 2){
                    System.out.println("NOTE:\n\t" +
passengerObj[i].getFirstName() + " " + passengerObj[i].getLastName() + " was
added to the Queue 2 successfully!");
                }else if (queueNo == 3){
                    System.out.println("NOTE:\n\t" +
passengerObj[i].getFirstName() + " " + passengerObj[i].getLastName() + " was
added to the Queue 3 successfully!");
                }else if (queueNo == 4){
                    System.out.println("NOTE:\n\t" +
passengerObj[i].getFirstName() + " " + passengerObj[i].getLastName() + " was
added to the Queue 4 successfully!");
                }else if (queueNo == 5){
                    System.out.println("NOTE:\n\t" +
passengerObj[i].getFirstName() + " " + passengerObj[i].getLastName() + " was
added to the Queue 5 successfully!");
                }
                break;
```

```java
            }
        }
    }

    //This method is to remove a customer from a specific index
    public int removeFromIndex(int queueNo, int position, int stock,
ArrayList<String[]> waiting){
        if (passengerObj[position].getFirstName() != null){
            if (stock + Integer.parseInt(passengerObj[position].getLiters())
> 6600){
                stock = 6600;
            }else {
                stock +=
Integer.parseInt(passengerObj[position].getLiters());
            }
            setQueue_income(getQueue_income() - (430 *
Integer.parseInt(passengerObj[position].getLiters())));
            shift_Queue(position);
            System.out.println("NOTE:\n\tThe customer at the position " +
position + " was removed from the Queue " + queueNo + " successfully!" +
addFromWaitingQueue(waiting));
        }else {
            System.out.println("NOTE:\n\tOops...There's no customer at the
position " + position + " in the Queue " + queueNo);
        }
        return stock;
    }


    //This method removes a served customer
    public void removeServedCustomer(int queueNo, ArrayList<String[]>
waiting){
        if (passengerObj[0].getFirstName() != null){
            shift_Queue(0);
            System.out.println("NOTE:\n\tThe first customer was removed from
the Queue " + queueNo + " successfully!" + addFromWaitingQueue(waiting));
        }else {
            System.out.println("NOTE:\n\tOops...There are no customers in the
Queue " + queueNo);
        }
    }

    //This method is to add a customer to a queue from the waiting list when
a customer was removed from a queue.
    public String addFromWaitingQueue(ArrayList<String[]> waiting){
        String message = "";
        if (waiting.size() > 0) {
            for (int i = 0; i < 6; i++) {
                if (passengerObj[i].getFirstName() == null) {
                    passengerObj[i].setFirstName(waiting.get(0)[0]);
                    passengerObj[i].setLastName(waiting.get(0)[1]);
                    passengerObj[i].setVehicleNo(waiting.get(0)[2]);
                    passengerObj[i].setLiters(waiting.get(0)[3]);
                    setQueue_income(getQueue_income() + (430 *
Integer.parseInt(waiting.get(0)[3])));
                    waiting.remove(0);
                    message = "\n\tThe next customer at The Waiting Queue was
added to the empty space successfully!";
                }
            }
        }
        return message;
    }

    //Shifting the queue after a customer was removed
    public void shift_Queue(int position){
        //This method shifts the remaining customers to the left, when a
customer was removed.
        if (position == 5){
            allToNull(5);
        }else {
            for (int i = 0; i < 6; i++){
                if ((position + i + 1) == 5){    //This only becomes true for
the 4th index. If it's true, the index 4 will be replaced by the 5th index.

passengerObj[4].setFirstName(passengerObj[5].getFirstName());

passengerObj[4].setLastName(passengerObj[5].getLastName());

passengerObj[4].setVehicleNo(passengerObj[5].getVehicleNo());
                    passengerObj[4].setLiters(passengerObj[5].getLiters());
                    allToNull(5);
                    break;
                }else {
                    passengerObj[position +
i].setFirstName(passengerObj[position + i + 1].getFirstName());
                    passengerObj[position +
i].setLastName(passengerObj[position + i + 1].getLastName());
                    passengerObj[position +
i].setVehicleNo(passengerObj[position + i + 1].getVehicleNo());
                    passengerObj[position +
i].setLiters(passengerObj[position + i + 1].getLiters());
                    //The index will be replaced by the index after that.
                }
            }
        }
    }

    //This method is to assign the details of a customer to null value when
shifting the customers
    public void allToNull(int i){
        passengerObj[i].setFirstName(null);
        passengerObj[i].setLastName(null);
        passengerObj[i].setVehicleNo(null);
        passengerObj[i].setLiters(null);
    }


    //View Customers Sorted in alphabetical order
    public void sort_Customers(String[][] queue){
        //This method sorts the customer names in the alphabetical order
without using the library sort routine.
        int name_Count = 0, max_Count, clone;
        ArrayList<String[]> store = new ArrayList<>();
        String[][] sorted_Queue = new String[6][4];  //Initializing a new
queue to store the sorted customer names.
        for (String[] strings : queue) {
            if (strings[0] != null) {     //This loop determines how many
customer names are there in the queue without the null values.
                name_Count++;
            }
        }
        for (String[] name : queue) {
            if (name[0] != null) {
                name[0] = name[0].toLowerCase();
                clone = 0;
                ArrayList <String[]> cloneList = new ArrayList<>();
                max_Count = 0;    //max_Count variable increases the count
when a name appears at the first in the alphabetical order than the other
names in the queue.
                //The below loop compares each name in the queue with all the
names in the same queue.
                for (String[] compare_Name : queue) {
                    if (compare_Name[0] != null){
                        compare_Name[0] = compare_Name[0].toLowerCase();
                        if (!name[0].equals(compare_Name[0])) {   //If the
name is not equals to the comparison name and null, then the comparison
statement will run.
                            if (name[0].compareTo(compare_Name[0]) < 0) {
                                max_Count++;    //If the name is at the first
in the alphabetical order than the comparison name, then it increases by 1.
                            }
                        }else if (name[0].equalsIgnoreCase(compare_Name[0])){
                            name[1] = name[1].toLowerCase();
                            compare_Name[1] = compare_Name[1].toLowerCase();
//If the first name of the customers are equal, then it will compare from the
last name.
                            if (name[1].compareTo(compare_Name[1]) < 0) {
                                max_Count++;
                            } else if (name[1].equals(compare_Name[1])){
                                clone++;
                                if (clone > 1){
                                    if (!store.contains(name)){
                                        store.add((name));
                                        cloneList.add(name);
                                    }
                                    if (!store.contains(compare_Name)){
                                        store.add(compare_Name);
                                        cloneList.add(compare_Name);
                                    }
                                }
                            }
                        }
                    }
                }
                for (int i = 0; i < name_Count; i++) {
                    if (max_Count == ((name_Count - 1) - i)) {
                        if (!store.contains(name)){
                            sorted_Queue[i] = name;
                        }else {
                            for (int j = 0; j < cloneList.size(); j++){
                                sorted_Queue[i-j] =
cloneList.get(cloneList.size()-(j+1));    //If there are customers with the
same name, then those will be appeared next to the other.
                            }
                        }
                    }
                }
            }
        }
        //Displaying the sorted queue.
        for (int i = 0; i < 6; i++) {
            System.out.println("\nCustomer " + (i+1));
            if (sorted_Queue[i][0] == null){
                System.out.println("EMPTY");
            }else {
                System.out.println("\tFirst Name      : " +
sorted_Queue[i][0]);
                System.out.println("\tLast Name       : " +
sorted_Queue[i][1]);
                System.out.println("\tVehicle No      : " +
sorted_Queue[i][2]);
                System.out.println("\tRequired Liters: " +
sorted_Queue[i][3]);
            }
        }
    }

}
```

Class Version (Main.java)

```
package com.example.prog_cw;

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

import javafx.application.Application;

public class Main {
    //Initializing The Stock and the getter and the setter
    static int stock = 6600;
    public static int getStock() {
        return stock;
    }
    public static void setStock(int stock) {
        Main.stock = stock;
    }


    public static void main(String[] args) throws IOException {

        //Defining an object for the Scanner
        Scanner input = new Scanner(System.in);

        //Creating an array of FuelQueue Objects
        FuelQueue[] queueObj = new FuelQueue[5];

        //Creating actual FuelQueue Objects
        for (int i = 0; i < 5; i++){
            queueObj[i] = new FuelQueue();
        }

        ArrayList<String[]> waiting = new ArrayList<>();


        //Displaying the menu
        mainLoop:
        while (true){
            System.out.println("\n-------------------------------------------
---------------------------------");
            System.out.println("Menu of The Fuel Queue Management System");
            System.out.println("\nEnter,");
            System.out.println("\t100 or VFQ : View all Fuel Queues.");
            System.out.println("\t101 or VEQ : View all Empty Queues.");
            System.out.println("\t102 or ACQ : Add customer to a Queue.");
            System.out.println("\t103 or RCQ : Remove a customer from a
Queue.");
            System.out.println("\t104 or PCQ : Remove a served customer.");
            System.out.println("\t105 or VCS : View Customers Sorted in
alphabetical order");
            System.out.println("\t106 or SPD : Store Program Data into
file.");
            System.out.println("\t107 or LPD : Load Program Data from 
file.");
            System.out.println("\t108 or STK : View Remaining Fuel Stock.");
            System.out.println("\t109 or AFS : Add Fuel Stock.");
            System.out.println("\t110 or IFQ : Income of each Fuel Queue.");
            System.out.println("\t999 or EXT : Exit the Program.");

            //Getting the user choice for the menu
            System.out.print("\nEnter your choice : ");
            String choice = input.next();

            //Using switch case to call the methods
            switch (choice.toUpperCase()){
                case "100":
                case "VFQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View all Fuel Queues.");
                    for (int i = 0; i < 5; i++){
                        System.out.println("\n*************************
Queue " + (i+1) + " *************************");
                        queueObj[i].viewAllQueues();
                    }
                    System.out.println("\n************************* Waiting 
Queue *************************");
                    for (int j = 0; j < waiting.size(); j++){
                        System.out.println("\nWaiting Customer " + (j+1));
                        System.out.println("\tFirst Name     : " + 
waiting.get(j)[0]);
                        System.out.println("\tLast Name      : " + 
waiting.get(j)[1]);
                        System.out.println("\tVehicle No     : " + 
waiting.get(j)[2]);
                        System.out.println("\tRequired Liters: " + 
waiting.get(j)[3]);
                    }
                    if (waiting.size() == 0){
                        System.out.println("EMPTY\n\tCurrently there are no 
customers in the Waiting Queue");
                    }

                    break;
                case "101":
                case "VEQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View all Empty Queues.");
                    for (int i = 0; i < 5; i++){
                        System.out.println("\n*************************
Queue " + (i+1) + " *************************");
                        queueObj[i].emptyQueues();
                    }
                    if (waiting.size() == 0){
                        System.out.println("\n*************************
Waiting Queue *************************");
                        System.out.println("EMPTY\n\tCurrently there are no 
customers in the Waiting Queue");
                    }

                    break;
                case "102":
                case "ACQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Add customer to a Queue.\n");
                    validateBeforeAddCustomer(queueObj, waiting);

                    break;
                case "103":
                case "RCQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Remove a customer at a specific
location from a Queue.\n");
                    remove_Customer(input, queueObj, waiting);

                    break;
                case "104":
                case "PCQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Remove a served customer.\n");
                    remove_Served_Customer(input, queueObj, waiting);

                    break;
                case "105":
                case "VCS":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View Customers Sorted in alphabetical
order\n");
                    callSortFunc(queueObj);

                    break;
                case "106":
                case "SPD":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Store Program Data into file.\n");
                    store_Data(queueObj, waiting);

                    break;
                case "107":
                case "LPD":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Load Program Data from file.\n");
                    load_Data(queueObj, waiting);

                    break;
                case "108":
                case "STK":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("View Remaining Fuel Stock.\n");
                    System.out.println("Remaining Fuel Stock : " + getStock()
+ " Liters out of 6600 Liters");

                    break;
                case "109":
                case "AFS":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Add Fuel Stock.\n");
                    addStock(input);

                    break;
                case "110":
                case "IFQ":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Income of each Fuel Queue.");
                    System.out.println("\nQueue 1 --> Rs." +
queueObj[0].getQueue_income());
                    System.out.println("\nQueue 2 --> Rs." +
queueObj[1].getQueue_income());
                    System.out.println("\nQueue 3 --> Rs." +
queueObj[2].getQueue_income());
                    System.out.println("\nQueue 4 --> Rs." +
queueObj[3].getQueue_income());
                    System.out.println("\nQueue 5 --> Rs." +
queueObj[4].getQueue_income());

                    break;
                case "999":
                case "EXT":
                    System.out.println("\n-----------------------------------
-----------------------------------------");
                    System.out.println("Exit the Program.");
                    break mainLoop;
                default:
                    System.out.println("ERROR:\n\tOops...There's no such a
choice.\n\tPlease read the instructions carefully and TRY AGAIN!");
            }
        }
        //Store all the existing data in a text file
        gui_data(queueObj, waiting);

        //Launch the GUI
        Application.launch(Task_4_Application.class, args);
    }

    //Find the minimum queue
    public static FuelQueue getMinimumQueue(FuelQueue[] queueObj){
        //This method adds a customer to the queue with the minimum length
        int queue_1_nullCount = queueObj[0].getNullCount();
        int queue_2_nullCount = queueObj[1].getNullCount();
        int queue_3_nullCount = queueObj[2].getNullCount();
        int queue_4_nullCount = queueObj[3].getNullCount();
        int queue_5_nullCount = queueObj[4].getNullCount();
```

```java
        if ((queue_1_nullCount >= queue_2_nullCount) && (queue_1_nullCount >=
queue_3_nullCount) && (queue_1_nullCount >= queue_4_nullCount) &&
(queue_1_nullCount >= queue_5_nullCount)){
            return queueObj[0];
        } else if ((queue_2_nullCount > queue_1_nullCount) &&
(queue_2_nullCount >= queue_3_nullCount) && (queue_2_nullCount >=
queue_4_nullCount) && (queue_2_nullCount >= queue_5_nullCount)){
            return queueObj[1];
        } else if ((queue_3_nullCount > queue_1_nullCount) &&
(queue_3_nullCount > queue_2_nullCount) && (queue_3_nullCount >=
queue_4_nullCount) && (queue_3_nullCount >= queue_5_nullCount)){
            return queueObj[2];
        } else if ((queue_4_nullCount > queue_1_nullCount) &&
(queue_4_nullCount > queue_2_nullCount) && (queue_4_nullCount >
queue_3_nullCount) && (queue_4_nullCount >= queue_5_nullCount)){
            return queueObj[3];
        } else{
            return queueObj[4];
        }
    }

    //This method pass the inputs to the queue with the minimum length
    public static void addCustomer(FuelQueue[] queueObj){
        int queueNo = 0;
        FuelQueue addObj = getMinimumQueue(queueObj);
        if (addObj == queueObj[0]){
            queueNo = 1;
        }else if (addObj == queueObj[1]){
            queueNo = 2;
        }else if (addObj == queueObj[2]){
            queueNo = 3;
        }else if (addObj == queueObj[3]){
            queueNo = 4;
        }else if (addObj == queueObj[4]){
            queueNo = 5;
        }
        String[] details = getCustomerInputs();
        addObj.addCustomerToQueue(queueNo, details);
    }

    //This method validates if the stock is finished or the queues are full
    public static void validateBeforeAddCustomer(FuelQueue[] queueObj,
ArrayList<String[]> waiting){
        if (getStock() == 0){
            System.out.println("NOTE:\n\tOops...The fuel is OUT OF STOCK.
Please come back again when a new stock arrives.\n\tThank You!");

        }
        else if ((queueObj[0].passengerObj[5].getFirstName() != null) &&
(queueObj[1].passengerObj[5].getFirstName() != null) &&
(queueObj[2].passengerObj[5].getFirstName() != null) &&
(queueObj[3].passengerObj[5].getFirstName() != null) &&
(queueObj[4].passengerObj[5].getFirstName() != null)) {
            System.out.println("NOTE:\n\tOops...All The Fuel Queues are
Full!\n\tYou'll be added to The Waiting Queue!\n");
```

```
            String[] details = getCustomerInputs();
            waiting.add(details);
            System.out.println("NOTE:\n\t" + details[0] + " " + details[1] +
" was added to the Waiting Queue successfully!");

        }
        else {
            addCustomer(queueObj);
        }
    }

    //This method gets all the customer details from the user
    public static String[] getCustomerInputs(){
        //Defining an object for the Scanner
        Scanner input = new Scanner(System.in);

        String[] details = new String[4];
        do {
            System.out.print("Enter The First Name : ");
            String fName = input.next();
            if (nameValidate(fName)){
                System.out.println("NOTE:\n\tFirst Name can only contain
LETTERS! Please Try Again!\n");
                continue;
            }
            if (fName.equals("null")){
                System.out.println("NOTE:\n\tOops..." + fName + " is not a
valid name since it is a reserved keyword in JAVA.\n\tPlease Try Again!\n");
            }else {
                details[0] = fName;
                break;
            }
        }while (true);

        do {
            System.out.print("Enter The Second Name : ");
            String lName = input.next();
            if (nameValidate(lName)){
                System.out.println("NOTE:\n\tLast Name can only contain
LETTERS! Please Try Again!\n");
                continue;
            }
            if (lName.equals("null")){
                System.out.println("NOTE:\n\tOops..." + lName + " is not a
valid name since it is a reserved keyword in JAVA.\n\tPlease Try Again!\n");
            }else {
                details[1] = lName;
                break;
            }
        }while (true);

        do {
            System.out.print("Enter The Vehicle Number : ");
            String vehicleNo = input.next();
            if (vehicleNo.equals("null")){
                System.out.println("NOTE:\n\tOops..." + vehicleNo + " is not
a valid input since it is a reserved keyword in JAVA.\n\tPlease Try
```

```
Again!\n");
            }else {
                details[2] = vehicleNo;
                break;
            }
        }while (true);

        while (true){
            try {
                System.out.print("Enter The Number of Liters Required : ");
                int liters = input.nextInt();
                if (liters <= 0){
                    System.out.println("ERROR:\n\tThe required fuel liters
cannot be less than or equal to zero!\n\tPlease Try Again!\n");
                } else if (liters > 10) {
                    System.out.println("ERROR:\n\tOops...The maximum amount
of liters per each customer is 10L!\n\tPlease Try Again!\n");
                } else if (liters > stock) {
                    System.out.println("ERROR:\n\tThe required fuel liters
cannot exceed the existing amount of liters in the stock\n\tPlease Try
Again!\n");
                } else {
                    details[3] = String.valueOf(liters);
                    setStock(getStock() - liters);
                    break;
                }
            }catch (Exception e){
                System.out.println("ERROR:\n\tThe required fuel liters can
only be a NUMBER!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
        return details;
    }

    //This method validates whether the user input name contains any other
symbols except letters
    public static boolean nameValidate(String name){
        char[] charArray = name.toCharArray();
        for (char i : charArray){
            if (!Character.isAlphabetic(i)){
                return true;
            }
        }
        return false;
    }

    //Add stock
    public static void addStock(Scanner input){
        //This method adds new fuel stock to the existing stock.
        while (true){
            try {
                System.out.print("Enter the amount of the new stock : ");
                int new_Stock = input.nextInt();     //Getting the amount of
the new stock from the user.
                if ((getStock() + new_Stock) > 6600){    //If the stock is
greater than 6600, then an error message will be displayed.
```

```
                    System.out.println("ERROR:\n\tThe amount of the fuel 
stock cannot exceed 6600 Liters!\n\tPlease Try Again!\n");
                    input.nextLine();
                } else if (new_Stock < 0) {     //If the stock is a negative 
number, an error message will be displayed.
                    System.out.println("ERROR:\n\tThe amount of the fuel 
stock cannot be a NEGATIVE NUMBER!\n\tPlease Try Again!\n");
                    input.nextLine();
                } else {
                    setStock(getStock()+new_Stock);
                    System.out.println("NOTE:\n\t" + new_Stock + " liters 
have been added to the stock successfully!");
                    break;
                }
            }catch (Exception e){       //If the fuel stock is not an integer 
value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe amount of the fuel stock 
can only be a NUMBER!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
    }

    //Get The Queue Number
    public static int get_Queue_Number(Scanner input){
        //This method validates the user input for the queue number which can 
only be 1/2/3/4/5.
        //This method is used to get the queue number when removing a 
customer and removing a served customer.
        int queueNo;
        while (true) {
            try {
                System.out.print("Enter The Queue Number (1/2/3/4/5): ");
                queueNo = input.nextInt();
                if (!(1 <= queueNo && queueNo <= 5)) {     //If the queue 
number is not a number between 1 and 5 inclusive, then an error message will 
be displayed.
                    System.out.println("ERROR:\n\tThe Queue number can only 
be 1 or 2 or 3 or 4 or 5!\n\tPlease Try Again!\n");
                    input.nextLine();
                }else {
                    break;
                }
            } catch (Exception e) {     //If the user input for the queue 
number is not an integer value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe Input can only be a 
number!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
        return queueNo;
    }

    //Remove a customer from a Queue (Specific Position)
    public static void remove_Customer(Scanner input, FuelQueue[] queueObj, 
ArrayList<String[]> waiting) {
        //This method removes a customer from the queue by index.
        int queueNo = get_Queue_Number(input);    //Calling the
get_Queue_Number method to validate the queue number.
        while (true) {
            try {
                System.out.print("Enter the position (0/1/2/3/4/5) : ");
                int position = input.nextInt();
                if (!(0 <= position && position <= 5)) {      //If the
position is not a number between 0 and 5 inclusive, then an error message
will be displayed.
                    System.out.println("ERROR:\n\tThe Position can only be a
number between 0 to 6 inclusive!\n\tPlease Try Again!\n");
                    input.nextLine();
                }else {
                    if (queueNo == 1){
                        setStock(queueObj[0].removeFromIndex(queueNo,
position, getStock(), waiting));
                    }else if (queueNo == 2){
                        setStock(queueObj[1].removeFromIndex(queueNo,
position, getStock(), waiting));
                    }else if (queueNo == 3){
                        setStock(queueObj[2].removeFromIndex(queueNo,
position, getStock(), waiting));
                    }else if (queueNo == 4){
                        setStock(queueObj[3].removeFromIndex(queueNo,
position, getStock(), waiting));
                    }else if (queueNo == 5){
                        setStock(queueObj[4].removeFromIndex(queueNo,
position, getStock(), waiting));
                    }
                    break;
                }
            }catch (Exception e) {      //If the position is not an integer
value, then an error message will be displayed.
                System.out.println("ERROR:\n\tThe Input can only be a
number!\n\tPlease Try Again!\n");
                input.nextLine();
            }
        }
    }


    //Remove a served customer from a Queue
    public static void remove_Served_Customer(Scanner input, FuelQueue[]
queueObj, ArrayList<String[]> waiting){
        //This method removes the first served customer from the queue.
        int queueNo = get_Queue_Number(input);    //Calling the
get_Queue_Number method to validate the queue number.
        if (queueNo == 1){
            queueObj[0].removeServedCustomer(queueNo, waiting);
        }else if (queueNo == 2){
            queueObj[1].removeServedCustomer(queueNo, waiting);
        }else if (queueNo == 3){
            queueObj[2].removeServedCustomer(queueNo, waiting);
        }else if (queueNo == 4){
            queueObj[3].removeServedCustomer(queueNo, waiting);
        }else if (queueNo == 5){
            queueObj[4].removeServedCustomer(queueNo, waiting);
        }
    }


    //This method calls the sort method in the FuelQueue objects
    public static void callSortFunc(FuelQueue[] queueObj){
        for (int i = 0; i < 5; i++){
            System.out.println("\n************************* Queue " + (i+1)
+ " *************************");
            String[][] queue = new String[6][4];
            for (int j = 0; j < 6; j++){
                queue[j][0] = queueObj[i].passengerObj[j].getFirstName();
                queue[j][1] = queueObj[i].passengerObj[j].getLastName();
                queue[j][2] = queueObj[i].passengerObj[j].getVehicleNo();
                queue[j][3] = queueObj[i].passengerObj[j].getLiters();
            }
            queueObj[i].sort_Customers(queue);
        }
    }

    //Store program data to a text file
    public static void store_Data(FuelQueue[] queueObj, ArrayList<String[]>
waiting){
        //This method stores the program data into a text file.
        try {
            FileWriter file = new FileWriter("Class_Version_Data.txt");
//Creates a text file to store data.

            for (int i = 0; i < 5; i++){
                for (int j = 0; j < 6; j++){
                    file.write(queueObj[i].passengerObj[j].getFirstName() + "
");

                    file.write(queueObj[i].passengerObj[j].getLastName() + "
");

                    file.write(queueObj[i].passengerObj[j].getVehicleNo() + "
");

                    file.write(queueObj[i].passengerObj[j].getLiters() +
"\n");
                }
            }

            file.write(getStock() + "\n");   //Storing the fuel stock in the
text file.
            for (int i = 0; i < 5; i++){           //Storing the fuel pump
incomes in the text file.
                file.write(queueObj[i].getQueue_income() + "\n");
            }

            for (String[] strings : waiting) {  //Storing customers in the
waiting queue
                file.write(strings[0] + " ");
                file.write(strings[1] + " ");
                file.write(strings[2] + " ");
                file.write(strings[3] + "\n");
            }
            file.close();
        }catch (Exception e){
            System.out.println("ERROR:\n\t" + e);
        }
        System.out.println("The data is processing...\n");       //Displaying
a message to the user that the process has happened successfully.
        System.out.println("NOTE:\n\tThe data was stored successfully into
the file.");
    }


    //Load data from a text file
    public static void load_Data(FuelQueue[] queueObj, ArrayList<String[]>
waiting){
        //This method loads the data from a text file.
        try {
            File file = new File("Class_Version_Data.txt");   //Opening the
text file.
            Scanner read_File = new Scanner(file);     //Reads the content in
the text file.

            int line_Count = 0;
            for (int i = 0; i < 5; i++){
                for (int j = 0; j <6; j++){
                    line_Count++;   //line_Count counts how many lines are
there in the text file.
                    String[] toArray = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                    if (toArray[0].equals("null")){
                        queueObj[i].passengerObj[j].setFirstName(null);
                        queueObj[i].passengerObj[j].setLastName(null);
                        queueObj[i].passengerObj[j].setVehicleNo(null);
                        queueObj[i].passengerObj[j].setLiters(null);
                    }else {
                        queueObj[i].passengerObj[j].setFirstName(toArray[0]);
                        queueObj[i].passengerObj[j].setLastName(toArray[1]);
                        queueObj[i].passengerObj[j].setVehicleNo(toArray[2]);
                        queueObj[i].passengerObj[j].setLiters(toArray[3]);
                    }
                }
            }

            for (int i = 0; i < 2; i++){
                if (line_Count == 30){
                    setStock(Integer.parseInt(read_File.nextLine()));
                    line_Count++;   //line_Count counts how many lines are
there in the text file.
                }else if (line_Count > 30){
                    for (int k = 0; k < 5; k++){

queueObj[k].setQueue_income(Integer.parseInt(read_File.nextLine()));
                        line_Count++;   //line_Count counts how many lines
are there in the text file.
                    }
                }
            }

            while (read_File.hasNext()){    //Loading the waiting queue
customers
```

```
                String[] toArray = read_File.nextLine().split(" ");
                waiting.add(toArray);
            }

            if (line_Count == 0){        //If there are no lines in the text
file, an error message will be displayed.
                System.out.println("NOTE:\n\tOops...There's no previously
available data in the text file.");
            }else {
                System.out.println("The data is loading...\n");
//Displaying a message to the user that the process has happened
successfully.
                System.out.println("NOTE:\n\tThe data was loaded
successfully!");
            }
        }catch (IOException e){      //If there's no existing text file with
that name, an error message will be displayed.
            System.out.println("ERROR:\n\tOops...The text file is not
available");
        }
    }


    //This method is to store all the existing data to a text file when the
program ends, which can be used in the controller to access the program data.
    public static void gui_data(FuelQueue[] queueObj, ArrayList<String[]>
waiting) throws IOException {
        //This method stores the program data into a text file.
        FileWriter file = new FileWriter("Gui_Data.txt");   //Creates a text
file to store data.

        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 6; j++) {
                file.write(queueObj[i].passengerObj[j].getFirstName() + " ");
                file.write(queueObj[i].passengerObj[j].getLastName() + " ");
                file.write(queueObj[i].passengerObj[j].getVehicleNo() + " ");
                file.write(queueObj[i].passengerObj[j].getLiters() + "\n");
            }
        }

        file.write(getStock() + "\n");   //Storing the fuel stock in the text
file.
        for (int i = 0; i < 5; i++) {           //Storing the fuel pump
incomes in the text file.
            file.write(queueObj[i].getQueue_income() + "\n");
        }

        for (String[] strings : waiting) {  //Storing customers in the
waiting queue
            file.write(strings[0] + " ");
            file.write(strings[1] + " ");
            file.write(strings[2] + " ");
            file.write(strings[3] + "\n");
        }
        file.close();
    }
}
```

GUI (Task_4_Application.java)

```java
package com.example.prog_cw;

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

import java.io.IOException;

public class Task_4_Application extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(Task_4_Application.class.getResource("menu.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 480, 400);
        stage.setTitle("Fuel Queue Management System");
        stage.setScene(scene);
        stage.show();
    }


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

    }
}
```

GUI (Task_4_Controller.java)

```
package com.example.prog_cw;

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

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Optional;
import java.util.ResourceBundle;
import java.util.Scanner;


public class Task_4_Controller implements Initializable {
    //Defining a 3D array to store all the customers and their details in
queues
    String[][][] store = new String[5][6][4];

    //OnAction for the 'Exit' button
    public void quit(ActionEvent event) throws IOException {
        //Deletes the text file when exiting from the GUI
        String filePath =
"/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/Gui_Data.txt";
        Path path = Paths.get(filePath);
        try {
            Files.delete(path);
        }catch (Exception e){
            System.out.println("");
        }

        Stage previousWindow = (Stage)
((Node)event.getSource()).getScene().getWindow();
        previousWindow.close();
    }

    //OnAction for the 'View All Queues' button
    @FXML
    public void view_all(ActionEvent event) throws IOException {
        Stage newStage = new Stage();
        Parent root =
FXMLLoader.load(getClass().getResource("view_all_queues.fxml"));
        Scene newScene = new Scene(root, 1080, 700);
        newStage.setTitle("View All Queues");
        newStage.setScene(newScene);
        newStage.show();

        //Identify previous stage and close it
        Stage previousWindow = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        previousWindow.close();
    }

    //OnAction for the 'Search a Customer' button
    @FXML
    public void search(ActionEvent event) throws IOException {
        Stage newStage = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("search.fxml"));
        Scene newScene = new Scene(root, 480, 400);
        newStage.setTitle("Search a Customer");
        newStage.setScene(newScene);
        newStage.show();

        //Identify previous stage and close it
        Stage previousWindow = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        previousWindow.close();
    }

    //OnAction for the 'Go Back' button
    public void back(ActionEvent event) throws IOException {
        Stage newStage = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("menu.fxml"));
        Scene newScene = new Scene(root, 480, 400);
        newStage.setTitle("Fuel Queue Management System");
        newStage.setScene(newScene);
        newStage.show();

        //Identify previous stage and close it
        Stage previousWindow = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        previousWindow.close();
    }

    //FXid for the text-field of search bar
    @FXML
    private TextField searchBar;

    //OnAction for the 'Search' button in the 'Search a Customer' window.
    public void onSearch(ActionEvent event) throws  IOException{
        //Load the program data from the text file
        try {
            File file = new File("Gui_Data.txt");   //Opening the text file.
            Scanner read_File = new Scanner(file);     //Reads the content in
the text file.

            for (int i = 0; i < 6; i++){
                String[] toArray1 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                store[0][i] = toArray1;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray2 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                store[1][i] = toArray2;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray3 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                store[2][i] = toArray3;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray4 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                store[3][i] = toArray4;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray5 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                store[4][i] = toArray5;
            }
            int foundCount = 0;
            int notFoundCount = 0;
            String userSearch = searchBar.getText().toLowerCase();
            for (int i = 0; i < store.length; i++) {
                for (int j = 0; j < 6; j++) {
                    if (store[i][j][0].toLowerCase().equals(userSearch)) {
                        foundCount++;
                        Alert alert = new Alert(Alert.AlertType.INFORMATION);
//If a search found, an information alert window appears
                        alert.setTitle("Search Found");
                        alert.setHeaderText("Here's the result " +
foundCount);
                        alert.setContentText("First Name :  " +
store[i][j][0] + "\nLast Name :  " + store[i][j][1] + "\nVehicle No :  " +
store[i][j][2] + "\nLiters :  " + store[i][j][3] + "\n\nQueue Number :  " +
(i+1) + "\nPosition :  " + j + " (Starting from '0')");
                        Optional<ButtonType> result = alert.showAndWait();
                    }else {
                        notFoundCount++;
                    }
                }
            }
            if (notFoundCount == 30){
                Alert alert = new Alert(Alert.AlertType.INFORMATION);   //If
a search is not found, an information alert window appears
                alert.setTitle("Search Not Found");
                alert.setHeaderText("No Result");
                alert.setContentText("Oops...There are no customers with the
name '" + userSearch + "'");
                Optional<ButtonType> result = alert.showAndWait();
            }
```

```
        } catch (FileNotFoundException e) {
            Alert alert = new Alert(Alert.AlertType.ERROR);  //If the text
file is not found, an error alert window appears
            alert.setTitle("Error");
            alert.setHeaderText("Error");
            alert.setContentText("There are no previously available data");
            Optional<ButtonType> result=alert.showAndWait();
        }
    }

    //OnAction for the 'Update' button in the 'View All Queues' window.
    public void click(ActionEvent event) throws IOException {
        load_Data();
    }

    //FXid's in the 'View All Queues' window.
    @FXML
    private TextArea queue_1;
    @FXML
    private TextArea queue_2;
    @FXML
    private TextArea queue_3;
    @FXML
    private TextArea queue_4;
    @FXML
    private TextArea queue_5;
    @FXML
    private TextArea waiting_queue;
    @FXML
    private Label stock;
    @FXML
    private Label income_1;
    @FXML
    private Label income_2;
    @FXML
    private Label income_3;
    @FXML
    private Label income_4;
    @FXML
    private Label income_5;

    //Load program data from the text file.
    public void load_Data()  {
        try {
            File file = new File("Gui_Data.txt");   //Opening the text file.
            Scanner read_File = new Scanner(file);    //Reads the content in
the text file.

            for (int i = 0; i < 6; i++){
                String[] toArray1 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                if (toArray1[0].equals("null")) {    //In this for loop, the
values in the converted array will be cloned to the queue.
                    queue_1.appendText("Customer 0" + String.valueOf(i+1) +
"\n\t EMPTY\n\n");
                } else {
                    queue_1.appendText("Customer " + 0+(i+1) + "\n");
                    queue_1.appendText("\tFirst Name : " + toArray1[0] +
"\n");
                    queue_1.appendText("\tLast Name : " + toArray1[1] +
"\n");
                    queue_1.appendText("\tVehicle No : " + toArray1[2] +
"\n");
                    queue_1.appendText("\tLiters : " + toArray1[3] + "\n\n");
                }
                store[0][i] = toArray1;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray2 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                if (toArray2[0].equals("null")) {    //In this for loop, the
values in the converted array will be cloned to the queue.
                    queue_2.appendText("Customer 0" + String.valueOf(i+1) +
"\n\t EMPTY\n\n");
                } else {
                    queue_2.appendText("Customer " + 0+(i+1) + "\n");
                    queue_2.appendText("\tFirst Name : " + toArray2[0] +
"\n");
                    queue_2.appendText("\tLast Name : " + toArray2[1] +
"\n");
                    queue_2.appendText("\tVehicle No : " + toArray2[2] +
"\n");
                    queue_2.appendText("\tLiters : " + toArray2[3] + "\n\n");
                }
                store[1][i] = toArray2;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray3 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                if (toArray3[0].equals("null")) {    //In this for loop, the
values in the converted array will be cloned to the queue.
                    queue_3.appendText("Customer 0" + String.valueOf(i+1) +
"\n\t EMPTY\n\n");
                } else {
                    queue_3.appendText("Customer " + 0+(i+1) + "\n");
                    queue_3.appendText("\tFirst Name : " + toArray3[0] +
"\n");
                    queue_3.appendText("\tLast Name : " + toArray3[1] +
"\n");
                    queue_3.appendText("\tVehicle No : " + toArray3[2] +
"\n");
                    queue_3.appendText("\tLiters : " + toArray3[3] + "\n\n");
                }
                store[2][i] = toArray3;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray4 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                if (toArray4[0].equals("null")) {    //In this for loop, the
values in the converted array will be cloned to the queue.
                    queue_4.appendText("Customer 0" + String.valueOf(i+1) +
"\n\t EMPTY\n\n");
                } else {
                    queue_4.appendText("Customer " + 0+(i+1) + "\n");
                    queue_4.appendText("\tFirst Name : " + toArray4[0] +
"\n");
                    queue_4.appendText("\tLast Name : " + toArray4[1] +
"\n");
                    queue_4.appendText("\tVehicle No : " + toArray4[2] +
"\n");
                    queue_4.appendText("\tLiters : " + toArray4[3] + "\n\n");
                }
                store[3][i] = toArray4;
            }

            for (int i = 0; i < 6; i++){
                String[] toArray5 = read_File.nextLine().split(" ");
//Converting each line which is a string to an array.
                if (toArray5[0].equals("null")) {    //In this for loop, the
values in the converted array will be cloned to the queue.
                    queue_5.appendText("Customer 0" + String.valueOf(i+1) +
"\n\t EMPTY\n\n");
                } else {
                    queue_5.appendText("Customer " + 0+(i+1) + "\n");
                    queue_5.appendText("\tFirst Name : " + toArray5[0] +
"\n");
                    queue_5.appendText("\tLast Name : " + toArray5[1] +
"\n");
                    queue_5.appendText("\tVehicle No : " + toArray5[2] +
"\n");
                    queue_5.appendText("\tLiters : " + toArray5[3] + "\n\n");
                }
                store[4][i] = toArray5;
            }

            stock.setText(read_File.nextLine());    //Reads the stock
            income_1.setText(read_File.nextLine()); //Reads all the incomes
of the queues
            income_2.setText(read_File.nextLine());
            income_3.setText(read_File.nextLine());
            income_4.setText(read_File.nextLine());
            income_5.setText(read_File.nextLine());

            int count = 0;
            while (read_File.hasNext()){    //Loading the waiting queue
customers
                count++;
                String[] toArray = read_File.nextLine().split(" ");
                waiting_queue.appendText("Waiting Customer " + count + "\n");
                waiting_queue.appendText("\tFirst Name : " + toArray[0] +
"\n");
                waiting_queue.appendText("\tLast Name : " + toArray[1] +
"\n");
                waiting_queue.appendText("\tVehicle No : " + toArray[2] +
"\n");
                waiting_queue.appendText("\tLiters : " + toArray[3] +
"\n\n");
            }
            if (count == 0){
                waiting_queue.appendText("EMPTY");
            }


        } catch (FileNotFoundException e) {
            Alert alert = new Alert(Alert.AlertType.ERROR);    //If the text
file is not found, an error alert window appears
            alert.setTitle("Error");
            alert.setHeaderText("Error");
            alert.setContentText("There are no previously available data");
            Optional<ButtonType> result=alert.showAndWait();
        }
    }

    //Initialize method
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {

    }
}
```

GUI (menu.fxml)

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.paint.Color?>
<?import javafx.scene.paint.RadialGradient?>
<?import javafx.scene.paint.Stop?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-
background-color: white; -fx-border-width: 5; -fx-border-color: gold;"
xmlns="http://javafx.com/javafx/18" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.prog_cw.Task_4_Controller">
   <children>
      <ImageView fitHeight="331.0" fitWidth="591.0" layoutX="-60.0"
layoutY="57.0">
         <image>
            <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_2_menu.png" />
         </image>
      </ImageView>
      <Label layoutX="59.0" layoutY="31.0" text="Fuel Queue Management
System">
         <font>
            <Font name="Lucida Grande" size="23.0" />
         </font>
      </Label>
      <Button layoutX="166.0" layoutY="135.0" mnemonicParsing="false"
onAction="#view_all" prefHeight="26.0" prefWidth="136.0" style="-fx-
background-color: #34c9eb#34c9eb #34c9eb#34c9eb;" text="View All Queues">
         <font>
            <Font name="Arial Bold" size="13.0" />
         </font>
      </Button>
      <Button layoutX="166.0" layoutY="182.0" mnemonicParsing="false"
onAction="#search" style="-fx-background-color: #34eb43#34eb43
#34eb43#34eb43;" text="Search a Customer">
         <font>
            <Font name="Arial Bold" size="13.0" />
         </font>
      </Button>
      <Button layoutX="165.0" layoutY="228.0" mnemonicParsing="false"
onAction="#quit" prefHeight="26.0" prefWidth="136.0" style="-fx-background-
color: #ff7070#ff7070 #ff7070#ff7070;" text="Exit">
         <font>
            <Font name="Arial Bold" size="13.0" />
         </font>
      </Button>
      <Label layoutX="183.0" layoutY="282.0" text="Menu">
         <font>
            <Font name="Avenir Next Bold" size="35.0" />
         </font>
         <textFill>
            <RadialGradient centerX="0.5" centerY="0.5" radius="0.5">
               <stops>
                  <Stop>
                     <color>
                        <Color red="1.0" green="0.94035506248474l2" />
                     </color>
                  </Stop>
                  <Stop offset="1.0">
                     <color>
                        <Color red="0.9736841917037964"
green="0.9334145784378052" blue="0.3696393668651581" />
                     </color>
                  </Stop>
               </stops>
            </RadialGradient>
         </textFill>
      </Label>
   </children>
</AnchorPane>
```

GUI (view_all_queues.fxml)

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<ScrollPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="1555.0" prefWidth="1061.0"
xmlns="http://javafx.com/javafx/18" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.prog_cw.Task_4_Controller">
  <content>
    <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="1632.0"
prefWidth="1062.0" style="-fx-border-color: gold; -fx-border-width: 3;">
         <children>
            <ImageView fitHeight="166.0" fitWidth="232.0" layoutX="-18.0"
layoutY="172.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/queue.png" />
               </image>
            </ImageView>
            <Button layoutX="866.0" layoutY="513.0" mnemonicParsing="false"
onAction="#back" style="-fx-background-color: #ff7070;" text="Go Back">
               <font>
                  <Font name="Arial Italic" size="13.0" />
               </font>
            </Button>
            <Label layoutX="307.0" layoutY="123.0" text="Queue 1">
               <font>
                  <Font name="Arial Black" size="19.0" />
               </font>
            </Label>
            <Label layoutX="307.0" layoutY="416.0" text="Queue 2">
               <font>
                  <Font name="Arial Black" size="19.0" />
               </font>
            </Label>
            <Label layoutX="308.0" layoutY="697.0" text="Queue 3">
               <font>
                  <Font name="Arial Black" size="19.0" />
               </font>
            </Label>
            <Label layoutX="307.0" layoutY="967.0" text="Queue 4">
               <font>
                  <Font name="Arial Black" size="19.0" />
               </font>
            </Label>
            <Label layoutX="314.0" layoutY="1258.0" text="Queue 5">
               <font>
                  <Font name="Arial Black" size="19.0" />
               </font>
            </Label>
            <Separator layoutX="534.0" layoutY="-18.0" orientation="VERTICAL"
prefHeight="1668.0" prefWidth="0.0" />
            <ImageView fitHeight="66.0" fitWidth="72.0" layoutX="628.0"
layoutY="104.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_view_all.png" />
               </image>
            </ImageView>
            <Label layoutX="677.0" layoutY="54.0" text="Incomes of Fuel
Queues">
               <font>
                  <Font name="Lucida Grande" size="18.0" />
               </font>
            </Label>
            <Label layoutX="722.0" layoutY="127.0" text="Queue 1">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label layoutX="820.0" layoutY="125.0" text="\$">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <ImageView fitHeight="66.0" fitWidth="72.0" layoutX="628.0"
layoutY="186.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_view_all.png" />
               </image>
            </ImageView>
            <Label layoutX="721.0" layoutY="209.0" text="Queue 2">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label layoutX="820.0" layoutY="207.0" text="\$">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <ImageView fitHeight="66.0" fitWidth="72.0" layoutX="628.0"
layoutY="261.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_view_all.png" />
               </image>
            </ImageView>
            <Label layoutX="721.0" layoutY="284.0" text="Queue 3">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label layoutX="820.0" layoutY="282.0" text="\$">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <ImageView fitHeight="66.0" fitWidth="72.0" layoutX="628.0"
layoutY="337.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_view_all.png" />
               </image>
            </ImageView>
            <Label layoutX="721.0" layoutY="360.0" text="Queue 4">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label layoutX="820.0" layoutY="358.0" text="\$">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <ImageView fitHeight="66.0" fitWidth="72.0" layoutX="628.0"
layoutY="416.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_pump_view_all.png" />
               </image>
            </ImageView>
            <Label layoutX="721.0" layoutY="439.0" text="Queue 5">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label layoutX="820.0" layoutY="437.0" text="\$">
               <font>
                  <Font size="19.0" />
               </font>
            </Label>
            <Label layoutX="152.0" layoutY="79.0" text="Details of The
Customers">
               <font>
                  <Font name="Lucida Grande" size="18.0" />
               </font>
            </Label>
            <Separator layoutX="536.0" layoutY="606.0" prefHeight="4.0"
prefWidth="533.0" />
            <Label layoutX="638.0" layoutY="890.0" text="Customers in The
Waiting Queue">
```

```
               <font>
                  <Font name="Lucida Grande" size="18.0" />
               </font>
            </Label>
            <Label layoutX="684.0" layoutY="655.0" text="Remaining Fuel
Stock">
               <font>
                  <Font name="Lucida Grande" size="18.0" />
               </font>
            </Label>
            <Separator layoutX="532.0" layoutY="845.0" prefHeight="4.0"
prefWidth="533.0" />
            <ImageView fitHeight="165.0" fitWidth="200.0" layoutX="810.0"
layoutY="682.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/fuel_indicator.png" />
               </image>
            </ImageView>
            <Label layoutX="686.0" layoutY="731.0" text="out of">
               <font>
                  <Font name="Avenir Next Bold" size="19.0" />
               </font>
            </Label>
            <Label layoutX="752.0" layoutY="734.0" text="6600L">
               <font>
                  <Font name="Arial Bold Italic" size="19.0" />
               </font>
            </Label>
            <Label layoutX="663.0" layoutY="734.0" text="L">
               <font>
                  <Font name="Arial Bold Italic" size="19.0" />
               </font>
            </Label>
            <TextArea fx:id="queue_1" layoutX="198.0" layoutY="160.0"
prefHeight="202.0" prefWidth="303.0" promptText="Empty" />
            <TextArea fx:id="queue_2" layoutX="199.0" layoutY="449.0"
prefHeight="202.0" prefWidth="303.0" promptText="Empty" />
            <ImageView fitHeight="166.0" fitWidth="232.0" layoutX="-12.0"
layoutY="476.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/queue.png" />
               </image>
            </ImageView>
            <TextArea fx:id="queue_3" layoutX="199.0" layoutY="725.0"
prefHeight="202.0" prefWidth="303.0" promptText="Empty" />
            <ImageView fitHeight="166.0" fitWidth="232.0" layoutX="-12.0"
layoutY="743.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/queue.png" />
               </image>
            </ImageView>
            <TextArea fx:id="queue_4" layoutX="205.0" layoutY="1004.0"
prefHeight="202.0" prefWidth="303.0" promptText="Empty" />
            <ImageView fitHeight="166.0" fitWidth="232.0" layoutY="1022.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/queue.png" />
               </image>
            </ImageView>
            <TextArea fx:id="queue_5" layoutX="205.0" layoutY="1294.0"
prefHeight="202.0" prefWidth="303.0" promptText="Empty" />
            <ImageView fitHeight="166.0" fitWidth="232.0" layoutY="1312.0">
               <image>
                  <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/queue.png" />
               </image>
            </ImageView>
            <Button layoutX="438.0" layoutY="30.0" mnemonicParsing="false"
onAction="#click" style="-fx-background-color: #99ff00;" text="Update" />
            <Label fx:id="stock" layoutX="614.0" layoutY="734.0"
text=".....">
               <font>
                  <Font name="Arial Bold Italic" size="19.0" />
               </font>
            </Label>
            <Label fx:id="income_1" layoutX="843.0" layoutY="127.0"
text=".....">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label fx:id="income_2" layoutX="841.0" layoutY="209.0"
text=".....">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label fx:id="income_3" layoutX="841.0" layoutY="284.0"
text=".....">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label fx:id="income_4" layoutX="842.0" layoutY="360.0"
text=".....">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <Label fx:id="income_5" layoutX="842.0" layoutY="439.0"
text=".....">
               <font>
                  <Font size="16.0" />
               </font>
            </Label>
            <TextArea fx:id="waiting_queue" layoutX="630.0" layoutY="967.0"
```

```
prefHeight="529.0" prefWidth="303.0" promptText="Empty" />
            <Label layoutX="68.0" layoutY="31.0" text="Please Click On Update
If The Data is not Visible" textFill="RED">
               <font>
                  <Font name="Avenir Next Bold" size="15.0" />
               </font>
            </Label>
         </children></AnchorPane>
  </content>
</ScrollPane>
```

GUI (search.fxml)

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="400.0" prefWidth="600.0" style="-fx-border-width: 3;
-fx-border-color: gold;" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.prog_cw.Task_4_Controller">
   <children>
      <ImageView fitHeight="305.0" fitWidth="459.0" layoutX="-10.0"
layoutY="63.0">
         <image>
            <Image
url="file:/Users/sarithwijesundera/Desktop/w1912785_Prog_CW/src/main/java/com
/example/Images/maths.png" />
         </image>
      </ImageView>
      <Label layoutX="52.0" layoutY="32.0" text="Search The Details of a
Customer">
         <font>
            <Font name="Lucida Grande" size="21.0" />
         </font>
      </Label>
      <TextField fx:id="searchBar" layoutX="83.0" layoutY="166.0"
prefHeight="26.0" prefWidth="196.0" promptText="Search" />
      <Button layoutX="300.0" layoutY="166.0" mnemonicParsing="false"
onAction="#onSearch" text="Search" />
      <Button layoutX="372.0" layoutY="343.0" mnemonicParsing="false"
onAction="#back" style="-fx-background-color: #ff7070;" text="Go Back">
         <font>
            <Font name="Arial Italic" size="13.0" />
         </font>
      </Button>
      <Label layoutX="120.0" layoutY="122.0" text="Search By The First Name">
         <font>
            <Font name="Avenir Next Bold" size="15.0" />
         </font>
      </Label>
   </children>
</AnchorPane>
```

<<END>>