


Name: 1. MUHAMMAD IQHWAN MUSLIM 2. MD ISRAFIL SIDDIKY RIFAT 3. MUHAMMAD AMIN BIN MOHD SOFUAN		Section : 01
ID Number: 1. AM2307014172 2. AF2311015342 3. AM2311015359		
Lecturer: MADAM NOORNAJWA BINTI MD AMIN		Lab group / Tutorial group / Tutor (if applicable):
Course and Course Code: SWC3344-DATA STRUCTURE		Submission Date: 18/11/2024
Assignment No. / Title: GROUP PROJECT (30%)		Extension & Late submission: Disallowed
Assignment Type: GROUP	% of Assignment Mark 30%	Returning Date: 18/11/2024
Penalties: <ol style="list-style-type: none"> 10% of the original mark will be deducted for every one week period after the submission date. No work will be accepted after two weeks of the deadline. If you were unable to submit the coursework on time due to extenuating circumstances you may be eligible for an extension. Extension will not exceed one week. 		
<p>Declaration: I/we the undersigned confirm that I/we have read and agree to abide by these regulations on plagiarism and cheating. I/we confirm that this piece of work is my/our own. I/we consent to appropriate storage of our work for checking to ensure that there is no plagiarism/ academic cheating.</p> <p>Signature(s): <u> Rifat </u> </p> <p>Full Name(s): MUHAMMAD IKHWAN MUSLIM, MD ISRAFIL SIDDIKY RIFAT, MUHAMMAD AMIN BIN MOHD SOFUAN</p>		
<p>This section may be used for feedback or other information</p>		

TABLE OF CONTENTS

NO	CONTENTS	PAGE
1.0	INTRODUCTION	3
2.0	ANALYSIS	3-4
3.0	FLOWCHART AND URL DIAGRAM	5-6
4.0	Design of Data Structure Classes	7
5.0	Class Definitions	8-13
6.0	JAVA CODE AND SAMPLE INPUT	13-26
7.0	PROGRAM OUTPUT	25-29
8.0	LESSONS LEARNED, REFLECTIVE EXPERIENCE TOWARDS COMPLETING THE INDIVIDUAL/OVERALL TASK AND CONCLUSION	30-31
9.0	REFERENCES	31

1.0 INTRODUCTION

Introduction to the Vehicle Service Center Management System

In today's fast-paced world, efficient management of vehicle service centers is paramount. This project aims to develop a robust Vehicle Service Center Management System using Java, designed to streamline and automate the process of managing customer information, service requests, and transactions. The system is implemented with a command-line interface (CLI), making it accessible and straightforward to use.

Project Objectives

- **Centralized Customer Management:** Maintain detailed records of customers, including their personal information and service history.
- **Service Request Handling:** Efficiently manage and process service requests using a queue-based system.
- **Receipt Generation:** Automatically generate and display service receipts for customers.
- **Interactive CLI:** Provide an interactive command-line interface for users to add customers, process services, upload customer lists, and view customer status.

The Vehicle Service Center Management System offers a practical and efficient solution for managing vehicle service operations. By leveraging Java and a user-friendly CLI, the system ensures that service centers can handle customer requests promptly and maintain accurate records, ultimately enhancing customer satisfaction and operational efficiency.

2.0 ANALYSIS

The Vehicle Service Center Management System is a practical and effective solution for modernizing service center operations. By integrating automated processes and leveraging appropriate data structures, the system significantly improves efficiency, organization, and customer satisfaction. It also serves as an excellent demonstration of applying programming and data structure concepts to solve real-world challenges.

Key Features

1. **Add Customer:** Allows users to input customer details and their corresponding service requests.

2. **Process Services:** Processes all service requests in the system, ensuring efficient management of resources.
3. **Display Receipts:** Generates and displays detailed receipts for each processed service request.
4. **View Customer List:** Provides a comprehensive view of all customers, marking each as "Processed" or "Not Processed."
5. **Upload Customer List File:** Enables bulk upload of customer data from a file, streamlining the addition of multiple customers.
6. **Interactive CLI:** Facilitates easy interaction with the system, ensuring users can execute commands efficiently.

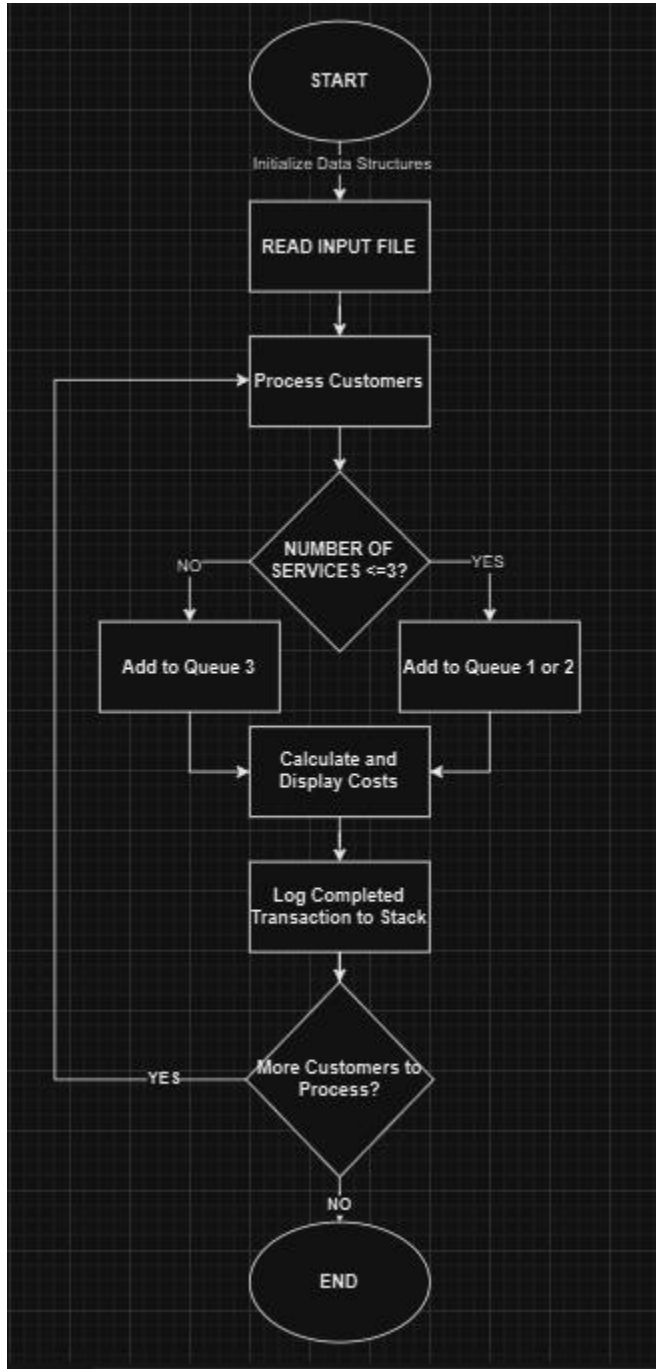
Implementation Overview

- **CustomerInfo Class:** Stores customer details and associated services.
- **ServiceInfo Class:** Represents individual service requests, including cost and completion time.
- **VehicleServiceCenter Class:** Manages the list of customers.
- **ServiceLane Class:** Handles the queue management for different service lanes and processes the requests.
- **VehicleServiceApplication Class:** Main class that provides the CLI for user interaction.

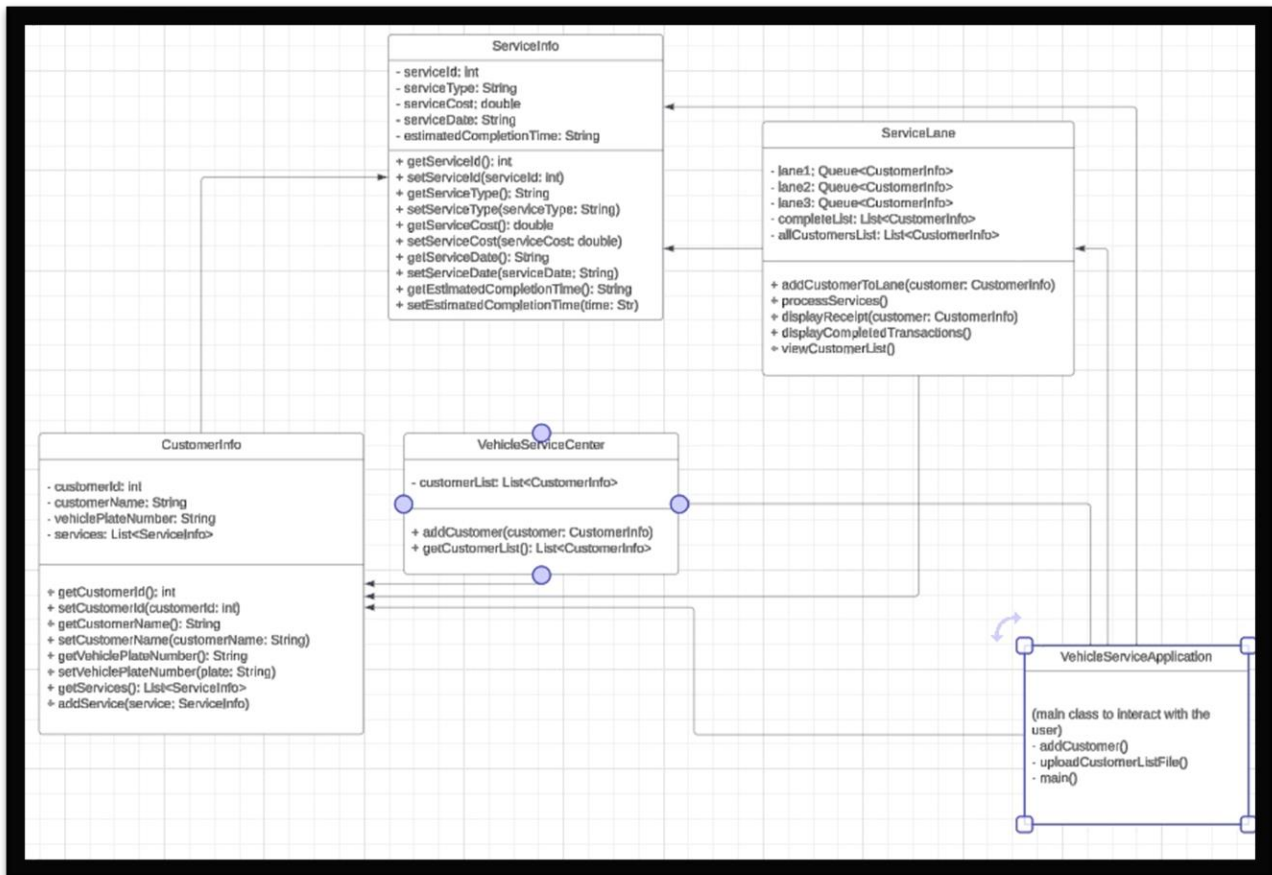
Usage

- **Adding Customers:** Users can add customers one by one through the CLI or upload a file containing multiple customer entries.
- **Processing Requests:** Service requests are processed in batches, ensuring efficient workflow.
- **Generating Receipts:** Detailed receipts for each customer are generated and displayed after processing.
- **Viewing Customers:** Users can view the entire customer list with their processing status, helping in tracking and managing service requests.

3.0 FLOWCHART & URL DIAGRAM

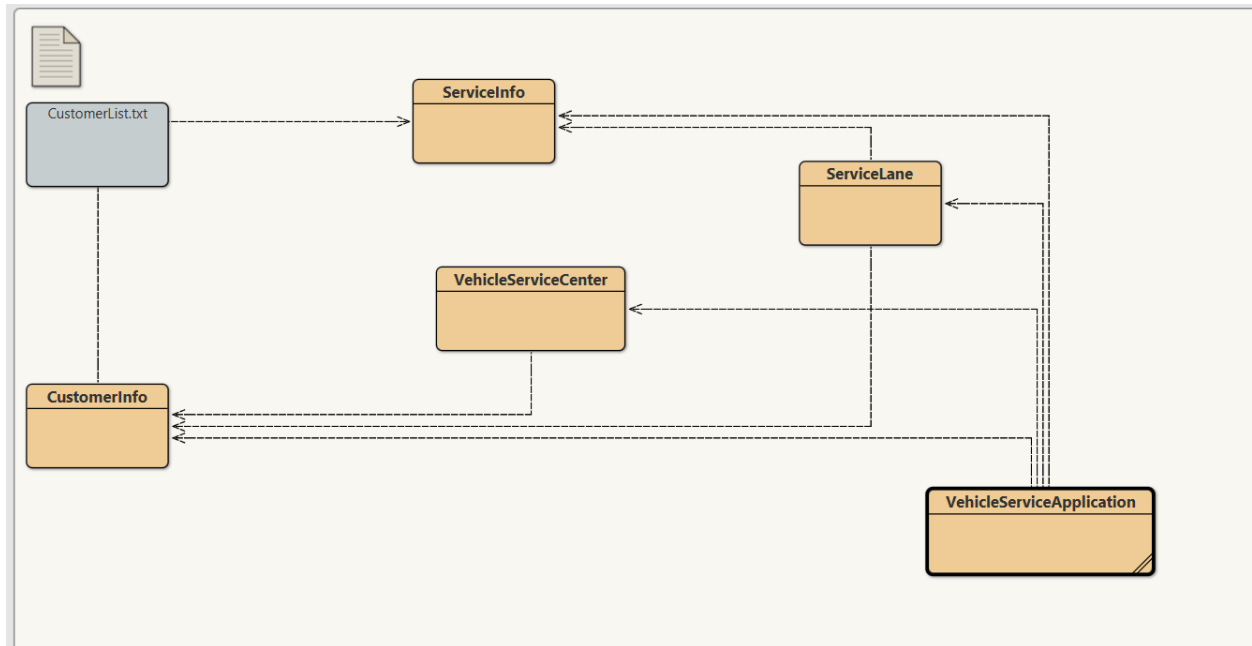


URL DIAGRAM



4.0 Design of Data Structure Classes

The **Vehicle Service Center Management System** uses three key data structures—**Queue**, **Stack**, and **LinkedList**—to manage service bookings, payment processing, and transaction logging effectively.



5.0 Class Definitions

1.CustomerInfo

```
1 import java.util.LinkedList;
2 import java.util.List;
3
4 class CustomerInfo {
5     private int customerId;
6     private String customerName;
7     private String vehiclePlateNumber;
8     private List<ServiceInfo> services;
9
10    public CustomerInfo(int customerId, String customerName, String vehiclePlateNumber) {
11        this.customerId = customerId;
12        this.customerName = customerName;
13        this.vehiclePlateNumber = vehiclePlateNumber;
14        this.services = new LinkedList<>();
15    }
16
17    // Getters and setters
18    public int getCustomerId() {
19        return customerId;
20    }
21
22    public void setCustomerId(int customerId) {
23        this.customerId = customerId;
24    }
25
26    public String getCustomerName() {
27        return customerName;
28    }
29
30    public void setCustomerName(String customerName) {
31        this.customerName = customerName;
32    }
33
34    public String getVehiclePlateNumber() {
35        return vehiclePlateNumber;
36    }
37
38    public void setVehiclePlateNumber(String vehiclePlateNumber) {
39        this.vehiclePlateNumber = vehiclePlateNumber;
40    }
41
42    public List<ServiceInfo> getServices() {
43        return services;
44    }
45
46    public void addService(ServiceInfo service) {
47        this.services.add(service);
48    }
49 }
```

2. ServiceInfo


```

1 import java.util.LinkedList;
2 import java.util.List;
3
4 class ServiceInfo {
5     private int serviceId;
6     private String serviceType;
7     private double serviceCost;
8     private String serviceDate;
9     private String estimatedCompletionTime;
10
11     public ServiceInfo(int serviceId, String serviceType, double serviceCost, String serviceDate, String estimatedCompletionTime) {
12         this.serviceId = serviceId;
13         this.serviceType = serviceType;
14         this.serviceCost = serviceCost;
15         this.serviceDate = serviceDate;
16         this.estimatedCompletionTime = estimatedCompletionTime;
17     }
18
19     // Getters and setters
20     public int getServiceId() {
21         return serviceId;
22     }
23
24     public void setServiceId(int serviceId) {
25         this.serviceId = serviceId;
26     }
27
28     public String getServiceType() {
29         return serviceType;
30     }
31
32     public void setServiceType(String serviceType) {
33         this.serviceType = serviceType;
34     }
35
36     public double getServiceCost() {
37         return serviceCost;
38     }
39
40     public void setServiceCost(double serviceCost) {
41         this.serviceCost = serviceCost;
42     }
43
44     public String getServiceDate() {
45         return serviceDate;
46     }
47
48     public void setServiceDate(String serviceDate) {
49         this.serviceDate = serviceDate;
50     }
51
52     public String getEstimatedCompletionTime() {
53         return estimatedCompletionTime;
54     }
55
56     public void setEstimatedCompletionTime(String estimatedCompletionTime) {
57         this.estimatedCompletionTime = estimatedCompletionTime;
58     }
59 }

```

3.ServiceLane

```
1 import java.util.LinkedList;
2 import java.util.Queue;
3 import java.util.ArrayList;
4 import java.util.List;
5
6 class ServiceLane {
7     private Queue<CustomerInfo> lane1;
8     private Queue<CustomerInfo> lane2;
9     private Queue<CustomerInfo> lane3;
10    private List<CustomerInfo> completeList;
11    private List<CustomerInfo> allCustomersList; // To keep track of all customers
12
13    public ServiceLane() {
14        lane1 = new LinkedList<>();
15        lane2 = new LinkedList<>();
16        lane3 = new LinkedList<>();
17        completeList = new ArrayList<>();
18        allCustomersList = new ArrayList<>();
19    }
20
21    public void addCustomerToLane(CustomerInfo customer) {
22        allCustomersList.add(customer); // Add to all customers list
23        if (customer.getServices().size() <= 3) {
24            if (lane1.size() <= lane2.size()) {
25                lane1.add(customer);
26            } else {
27                lane2.add(customer);
28            }
29        } else {
30            lane3.add(customer);
31        }
32    }
33
34    public void processServices() {
35        System.out.println("Processing Lane 1:");
36        processLane(lane1);
37        System.out.println("Processing Lane 2:");
38        processLane(lane2);
39        System.out.println("Processing Lane 3:");
40        processLane(lane3);
41    }
42
43    private void processLane(Queue<CustomerInfo> lane) {
44        while (!lane.isEmpty()) {
45            CustomerInfo customer = lane.poll();
46            completeList.add(customer);
47            displayReceipt(customer);
48        }
49    }
50
51    private void displayReceipt(CustomerInfo customer) {
52        double totalCost = 0;
53        for (ServiceInfo service : customer.getServices()) {
54            totalCost += service.getServiceCost();
55        }
56        System.out.printf("%-15d %-25s %-20s %-15.2f\n", customer.getCustomerId(), customer.getCustomerName(), customer.getVehiclePlateNumber(), totalCost);
57        System.out.println("-----");
58    }
59
60    public void displayCompletedTransactions() {
61        System.out.println("List of Receipts:");
62        System.out.printf("%-15s %-25s %-20s %-15s\n", "Customer ID", "Customer Name", "Vehicle Plate Number", "Total Service Cost");
63        System.out.println("-----");
64        for (CustomerInfo customer : completeList) {
65            displayReceipt(customer);
66        }
67    }
68 }
```

```

68
69 public void viewCustomerList() {
70     System.out.println("Customer List:");
71     System.out.printf("%-15s %-25s %-20s %-10s\n", "Customer ID", "Customer Name", "Vehicle Plate Number", "Status");
72     System.out.println("-----");
73     for (CustomerInfo customer : allCustomersList) {
74         String status = completeList.contains(customer) ? "Processed" : "Not Processed";
75         System.out.printf("%-15d %-25s %-20s %-10s\n", customer.getCustomerId(), customer.getCustomerName(), customer.getVehiclePlateNumber(), status);
76     }
77 }
78 }

```

5. VehicleServiceCenter

```

1 import java.util.LinkedList;
2 import java.util.List;
3
4 class VehicleServiceCenter {
5     private List<CustomerInfo> customerList;
6
7     public VehicleServiceCenter() {
8         customerList = new LinkedList<>();
9     }
10
11     public void addCustomer(CustomerInfo customer) {
12         customerList.add(customer);
13     }
14
15     public List<CustomerInfo> getCustomerList() {
16         return customerList;
17     }
18 }
19

```

5. VehicleServiceApplication

```

1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4 import java.util.Scanner;
5 import java.util.StringTokenizer;
6
7 public class VehicleServiceApplication {
8     public static void main(String[] args) {
9         Scanner scanner = new Scanner(System.in);
10         VehicleServiceCenter serviceCenter = new VehicleServiceCenter();
11         ServiceLane serviceLane = new ServiceLane();
12
13         while (true) {
14             System.out.println("Vehicle Service Center Management System");
15             System.out.println("1. Add Customer");
16             System.out.println("2. Process Services");
17             System.out.println("3. Display Receipts");
18             System.out.println("4. View Customer List");
19             System.out.println("5. Upload Customer List File");
20             System.out.println("6. Exit");
21             System.out.print("Enter your choice: ");
22             int choice = scanner.nextInt();
23             scanner.nextLine(); // Consume newline
24

```

```

24
25 switch (choice) {
26     case 1:
27         addCustomer(serviceCenter, serviceLane, scanner);
28         break;
29     case 2:
30         serviceLane.processServices();
31         break;
32     case 3:
33         serviceLane.displayCompletedTransactions();
34         break;
35     case 4:
36         serviceLane.viewCustomerList();
37         break;
38     case 5:
39         uploadCustomerListFile(serviceCenter, serviceLane, scanner);
40         break;
41     case 6:
42         System.out.println("Exiting... Goodbye!");
43         scanner.close();
44         return;
45     default:
46         System.out.println("Invalid choice. Please try again.");
47 }
48 }
49 }

```

```

50
51 private static void addCustomer(VehicleServiceCenter serviceCenter, ServiceLane serviceLane, Scanner scanner) {
52     System.out.print("Enter Customer ID: ");
53     int customerId = scanner.nextInt();
54     scanner.nextLine(); // Consume newline
55     System.out.print("Enter Customer Name: ");
56     String customerName = scanner.nextLine();
57     System.out.print("Enter Vehicle Plate Number: ");
58     String vehiclePlateNumber = scanner.nextLine();
59
60     CustomerInfo customer = new CustomerInfo(customerId, customerName, vehiclePlateNumber);
61
62     System.out.print("Enter the number of services: ");
63     int numServices = scanner.nextInt();
64     scanner.nextLine(); // Consume newline
65
66     for (int i = 0; i < numServices; i++) {
67         System.out.print("Enter Service ID: ");
68         int serviceId = scanner.nextInt();
69         scanner.nextLine(); // Consume newline
70         System.out.print("Enter Service Type: ");
71         String serviceType = scanner.nextLine();
72         System.out.print("Enter Service Cost: ");
73         double serviceCost = scanner.nextDouble();
74         scanner.nextLine(); // Consume newline
75         System.out.print("Enter Service Date: ");
76         String serviceDate = scanner.nextLine();
77         System.out.print("Enter Estimated Completion Time: ");
78         String estimatedCompletionTime = scanner.nextLine();
79
80         ServiceInfo service = new ServiceInfo(serviceId, serviceType, serviceCost, serviceDate, estimatedCompletionTime);
81         customer.addService(service);
82     }

```

```

83     serviceCenter.addCustomer(customer);
84     serviceLane.addCustomerToLane(customer);
85     System.out.println("Customer added successfully.");
86 }
87
88
89 private static void uploadCustomerListFile(VehicleServiceCenter serviceCenter, ServiceLane serviceLane, Scanner scanner) {
90     System.out.print("Enter the file path: ");
91     String filePath = scanner.nextLine();
92
93     try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
94         String line;
95         while ((line = br.readLine()) != null) {
96             StringTokenizer tokenizer = new StringTokenizer(line, ",");
97             int customerId = Integer.parseInt(tokenizer.nextToken());
98             String customerName = tokenizer.nextToken();
99             String vehiclePlateNumber = tokenizer.nextToken();
100
101             CustomerInfo customer = new CustomerInfo(customerId, customerName, vehiclePlateNumber);
102
103             while (tokenizer.hasMoreTokens()) {
104                 int serviceId = Integer.parseInt(tokenizer.nextToken());
105                 String serviceType = tokenizer.nextToken();
106                 double serviceCost = Double.parseDouble(tokenizer.nextToken());
107                 String serviceDate = tokenizer.nextToken();
108                 String estimatedCompletionTime = tokenizer.nextToken();
109
110                 ServiceInfo service = new ServiceInfo(serviceId, serviceType, serviceCost, serviceDate, estimatedCompletionTime);
111                 customer.addService(service);
112             }
113
114             serviceCenter.addCustomer(customer);
115             serviceLane.addCustomerToLane(customer);
116         }
117         System.out.println("Customer list uploaded successfully.");
118     } catch (IOException e) {
119         System.out.println("Error reading file: " + e.getMessage());
120     }
121 }
122 }
123

```

6.0 Java code and sample input and output.

1.CustomerInfo

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

```
import java.util.List;
```

```

class CustomerInfo {
    private int customerId;
    private String customerName;
    private String vehiclePlateNumber;
    private List<ServiceInfo> services;
}

```

```
public CustomerInfo(int customerId, String customerName, String vehiclePlateNumber) {  
    this.customerId = customerId;  
    this.customerName = customerName;  
    this.vehiclePlateNumber = vehiclePlateNumber;  
    this.services = new LinkedList<>();  
}
```

// Getters and setters

```
public int getCustomerId() {  
    return customerId;  
}
```

```
public void setCustomerId(int customerId) {  
    this.customerId = customerId;  
}
```

```
public String getCustomerName() {  
    return customerName;  
}
```

```
public void setCustomerName(String customerName) {  
    this.customerName = customerName;  
}
```

```
public String getVehiclePlateNumber() {  
    return vehiclePlateNumber;  
}
```

```
public void setVehiclePlateNumber(String vehiclePlateNumber) {  
    this.vehiclePlateNumber = vehiclePlateNumber;  
}
```

```

    }

    public List<ServiceInfo> getServices() {
        return services;
    }

    public void addService(ServiceInfo service) {
        this.services.add(service);
    }
}

```

2. ServiceInfo

```

import java.util.LinkedList;
import java.util.List;

class ServiceInfo {
    private int serviceId;
    private String serviceType;
    private double serviceCost;
    private String serviceDate;
    private String estimatedCompletionTime;

    public ServiceInfo(int serviceId, String serviceType, double serviceCost, String serviceDate,
String estimatedCompletionTime) {
        this.serviceId = serviceId;
        this.serviceType = serviceType;
        this.serviceCost = serviceCost;
        this.serviceDate = serviceDate;
        this.estimatedCompletionTime = estimatedCompletionTime;
    }
}

```

```
// Getters and setters
public int getServiceId() {
    return serviceId;
}

public void setServiceId(int serviceId) {
    this.serviceId = serviceId;
}

public String getServiceType() {
    return serviceType;
}

public void setServiceType(String serviceType) {
    this.serviceType = serviceType;
}

public double getServiceCost() {
    return serviceCost;
}

public void setServiceCost(double serviceCost) {
    this.serviceCost = serviceCost;
}

public String getServiceDate() {
    return serviceDate;
}
```



```

public void setServiceDate(String serviceDate) {
    this.serviceDate = serviceDate;
}

public String getEstimatedCompletionTime() {
    return estimatedCompletionTime;
}

public void setEstimatedCompletionTime(String estimatedCompletionTime) {
    this.estimatedCompletionTime = estimatedCompletionTime;
}
}

```

3.ServiceLane

```

import java.util.LinkedList;
import java.util.Queue;
import java.util.ArrayList;
import java.util.List;

class ServiceLane {
    private Queue<CustomerInfo> lane1;
    private Queue<CustomerInfo> lane2;
    private Queue<CustomerInfo> lane3;
    private List<CustomerInfo> completeList;
    private List<CustomerInfo> allCustomersList; // To keep track of all customers

    public ServiceLane() {
        lane1 = new LinkedList<>();
        lane2 = new LinkedList<>();
    }
}

```

```

lane3 = new LinkedList<>();
completeList = new ArrayList<>();
allCustomersList = new ArrayList<>();
}

public void addCustomerToLane(CustomerInfo customer) {
    allCustomersList.add(customer); // Add to all customers list
    if (customer.getServices().size() <= 3) {
        if (lane1.size() <= lane2.size()) {
            lane1.add(customer);
        } else {
            lane2.add(customer);
        }
    } else {
        lane3.add(customer);
    }
}

```

```

public void processServices() {
    System.out.println("Processing Lane 1:");
    processLane(lane1);
    System.out.println("Processing Lane 2:");
    processLane(lane2);
    System.out.println("Processing Lane 3:");
    processLane(lane3);
}

```

```

private void processLane(Queue<CustomerInfo> lane) {
    while (!lane.isEmpty()) {
        CustomerInfo customer = lane.poll();
    }
}

```

```

        completeList.add(customer);
        displayReceipt(customer);
    }
}

private void displayReceipt(CustomerInfo customer) {
    double totalCost = 0;
    for (ServiceInfo service : customer.getServices()) {
        totalCost += service.getServiceCost();
    }
    System.out.printf("%-15d %-25s %-20s %-15.2f%n", customer.getCustomerId(),
customer.getCustomerName(), customer.getVehiclePlateNumber(), totalCost);
    System.out.println("-----");
}

public void displayCompletedTransactions() {
    System.out.println("List of Receipts:");
    System.out.printf("%-15s %-25s %-20s %-15s%n", "Customer ID", "Customer Name",
"Vehicle Plate Number", "Total Service Cost");
    System.out.println("-----");
    for (CustomerInfo customer : completeList) {
        displayReceipt(customer);
    }
}

public void viewCustomerList() {
    System.out.println("Customer List:");
    System.out.printf("%-15s %-25s %-20s %-10s%n", "Customer ID", "Customer Name",
"Vehicle Plate Number", "Status");
    System.out.println("-----");
    for (CustomerInfo customer : allCustomersList) {

```

```

        String status = completeList.contains(customer) ? "Processed" : "Not Processed";

        System.out.printf("%-15d %-25s %-20s %-10s%n", customer.getCustomerId(),
customer.getCustomerName(), customer.getVehiclePlateNumber(), status);
    }
}
}

```

1. **Class: VehicleServiceCenter**

```

import java.util.LinkedList;
import java.util.List;

class VehicleServiceCenter {
    private List<CustomerInfo> customerList;

    public VehicleServiceCenter() {
        customerList = new LinkedList<>();
    }

    public void addCustomer(CustomerInfo customer) {
        customerList.add(customer);
    }

    public List<CustomerInfo> getCustomerList() {
        return customerList;
    }
}

```

2. **VehicleServiceApplication**

```

import java.io.BufferedReader;

```

```

import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;
import java.util.StringTokenizer;

public class VehicleServiceApplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        VehicleServiceCenter serviceCenter = new VehicleServiceCenter();
        ServiceLane serviceLane = new ServiceLane();

        while (true) {
            System.out.println("Vehicle Service Center Management System");
            System.out.println("1. Add Customer");
            System.out.println("2. Process Services");
            System.out.println("3. Display Receipts");
            System.out.println("4. View Customer List");
            System.out.println("5. Upload Customer List File");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    addCustomer(serviceCenter, serviceLane, scanner);
                    break;
                case 2:
                    serviceLane.processServices();
                    break;
            }
        }
    }
}

```

```

        case 3:
            serviceLane.displayCompletedTransactions();
            break;
        case 4:
            serviceLane.viewCustomerList();
            break;
        case 5:
            uploadCustomerListFile(serviceCenter, serviceLane, scanner);
            break;
        case 6:
            System.out.println("Exiting... Goodbye!");
            scanner.close();
            return;
        default:
            System.out.println("Invalid choice. Please try again.");
    }
}
}

```

```

private static void addCustomer(VehicleServiceCenter serviceCenter, ServiceLane
serviceLane, Scanner scanner) {

```

```

    System.out.print("Enter Customer ID: ");
    int customerId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Customer Name: ");
    String customerName = scanner.nextLine();
    System.out.print("Enter Vehicle Plate Number: ");
    String vehiclePlateNumber = scanner.nextLine();

```

```

    CustomerInfo customer = new CustomerInfo(customerId, customerName,
vehiclePlateNumber);

```

```

System.out.print("Enter the number of services: ");
int numServices = scanner.nextInt();
scanner.nextLine(); // Consume newline

for (int i = 0; i < numServices; i++) {
    System.out.print("Enter Service ID: ");
    int serviceId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Service Type: ");
    String serviceType = scanner.nextLine();
    System.out.print("Enter Service Cost: ");
    double serviceCost = scanner.nextDouble();
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Service Date: ");
    String serviceDate = scanner.nextLine();
    System.out.print("Enter Estimated Completion Time: ");
    String estimatedCompletionTime = scanner.nextLine();

    ServiceInfo service = new ServiceInfo(serviceId, serviceType, serviceCost, serviceDate,
estimatedCompletionTime);
    customer.addService(service);
}

serviceCenter.addCustomer(customer);
serviceLane.addCustomerToLane(customer);
System.out.println("Customer added successfully.");
}

private static void uploadCustomerListFile(VehicleServiceCenter serviceCenter, ServiceLane
serviceLane, Scanner scanner) {

```

```

System.out.print("Enter the file path: ");
String filePath = scanner.nextLine();

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
    String line;
    while ((line = br.readLine()) != null) {
        StringTokenizer tokenizer = new StringTokenizer(line, ",");
        int customerId = Integer.parseInt(tokenizer.nextToken());
        String customerName = tokenizer.nextToken();
        String vehiclePlateNumber = tokenizer.nextToken();

        CustomerInfo customer = new CustomerInfo(customerId, customerName,
vehiclePlateNumber);

        while (tokenizer.hasMoreTokens()) {
            int serviceId = Integer.parseInt(tokenizer.nextToken());
            String serviceType = tokenizer.nextToken();
            double serviceCost = Double.parseDouble(tokenizer.nextToken());
            String serviceDate = tokenizer.nextToken();
            String estimatedCompletionTime = tokenizer.nextToken();

            ServiceInfo service = new ServiceInfo(serviceId, serviceType, serviceCost,
serviceDate, estimatedCompletionTime);
            customer.addService(service);
        }

        serviceCenter.addCustomer(customer);
        serviceLane.addCustomerToLane(customer);
    }

    System.out.println("Customer list uploaded successfully.");
} catch (IOException e) {

```



```
        System.out.println("Error reading file: " + e.getMessage());
    }
}
}
```

7.0 OUTPUT:

```
Vehicle Service Center Management System
1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit
Enter your choice: 1
Enter Customer ID: 1
Enter Customer Name: Rifat
Enter Vehicle Plate Number: AHJ9874
Enter the number of services: 1
Enter Service ID: 1002
Enter Service Type: OIL CHANGE
Enter Service Cost: 300
Enter Service Date: 2024-10-20
Enter Estimated Completion Time: 2 HR
Customer added successfully.
```

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 1

Enter Customer ID: 2

Enter Customer Name: AMIN

Enter Vehicle Plate Number: WRF5678

Enter the number of services: 4

Enter Service ID: 1004

Enter Service Type: OIL CHANGE

Enter Service Cost: 500

Enter Service Date: 2024-06-12

Enter Estimated Completion Time: 5 HOUR

Enter Service ID: 1

Enter Service Type: ENGINE CHECK

Enter Service Cost: 300

Enter Service Date: 2024-06-12

Enter Estimated Completion Time: 2 HR

Enter Service ID: 1004

Enter Service Type: BREAKING CHECK

Enter Service Cost: 100

Enter Service Date: 2024-07-12

Enter Estimated Completion Time: 2 HR

Enter Service ID: 1004

Enter Service Type: VEHICLE PAINT

Enter Service Cost: 600

Enter Service Date: 2024-09-13

Enter Estimated Completion Time: 7 HR

Customer added successfully.

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 2

Processing Lane 1:

1	Rifat	AHJ9874	300.00
---	-------	---------	--------

Processing Lane 2:

Processing Lane 3:

2	AMIN	WRF5678	1500.00
---	------	---------	---------

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 3

List of Receipts:

Customer ID	Customer Name	Vehicle Plate Number	Total Service Cost
1	Rifat	AHJ9874	300.00
2	AMIN	WRF5678	1500.00

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 4

Customer List:

Customer ID	Customer Name	Vehicle Plate Number	Status
1	Rifat	AHJ9874	Processed
2	AMIN	WRF5678	Processed

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 5

Enter the file path: D:\SEM-3 UPTM\SWC3344\Vehicle service Management\VehicleServiceManagement - Copy\CustomerList.txt

Customer list uploaded successfully.

Vehicle Service Center Management System

1. Add Customer
2. Process Services
3. Display Receipts
4. View Customer List
5. Upload Customer List File
6. Exit

Enter your choice: 4

Customer List:

Customer ID	Customer Name	Vehicle Plate Number	Status
1	Rifat	AHJ9874	Processed
2	AMIN	WRF5678	Processed
1	Ahmad Zaki	ABC123	Not Processed
2	Nur Aisyah	XYZ789	Not Processed
3	Lim Wei Hong	DEF456	Not Processed
4	Sharifah Aminah	GHI111	Not Processed
5	Chew Kam Fai	JKL222	Not Processed
6	Siti Rahmah	MNO333	Not Processed
7	Kavitha Rajan	PQR444	Not Processed
8	Shanmuganathan Pillai	STU555	Not Processed
9	Wong Mei Ling	VWX666	Not Processed
10	Mohammad Faizal	YYY777	Not Processed
11	Chan Siew Ping	ABC124	Not Processed
12	Hafizah Binti Abdullah	XYZ780	Not Processed
13	Ahmad Shah	DEF457	Not Processed
14	Tan Lee Kuan	GHI112	Not Processed
15	Rashidah Binti Samad	JKL223	Not Processed
16	Lim Boon Keat	MNO334	Not Processed
17	Siti Aisyah	PQR445	Not Processed
18	Thamilarasi Subramaniam	STU556	Not Processed
19	Raja Norazlin	VWX667	Not Processed
20	Lau Shing Fung	YYY778	Not Processed
21	Mohd Azlan	ABC125	Not Processed
22	Chong Mei Ling	XYZ781	Not Processed
23	Nurul Huda	DEF458	Not Processed
24	Leong Siew Ling	GHI113	Not Processed
25	Ahmad Nizam	JKL224	Not Processed
26	Wan Siti Mariam	MNO335	Not Processed
27	Heng Soon Chuan	PQR446	Not Processed

27	Heng Soon Chuan	PQR446	Not Processed
28	Zulkifli Basha	STU557	Not Processed
29	Lim Chuan Hong	VWX668	Not Processed
30	Siti Nurhaliza	YYY779	Not Processed
31	Teoh Wai Leong	ABC126	Not Processed
32	Ali Basha	XYZ782	Not Processed
33	Chua Siew Fun	DEF459	Not Processed
34	Rozita Binti Ismail	GHI114	Not Processed
35	Foo Siew Fong	JKL225	Not Processed
36	Nadia Zakaria	MN0336	Not Processed
37	Ong Yew Weng	PQR447	Not Processed
38	Ahmad Nazri	STU558	Not Processed
39	Azlina Binti Omar	VWX669	Not Processed
40	Chan Hock Chuan	YYY780	Not Processed
41	Zainuddin Basha	ABC127	Not Processed
42	Low Siew Ping	XYZ783	Not Processed
43	Roslan Basha	DEF460	Not Processed
44	Lim Boon Hong	GHI115	Not Processed
45	Siti Mariam	JKL226	Not Processed
46	Thavamani Rajan	MN0337	Not Processed
47	Mohd Fadzil	PQR448	Not Processed
48	Wong Siew Fun	STU559	Not Processed
49	Zaiton Binti Abdullah	VWX670	Not Processed
50	Lim Boon Kiat	YYY781	Not Processed
51	Siti Khadijah	ABC128	Not Processed
52	Yong Siew Fong	XYZ784	Not Processed
53	Fazlina Binti Ismail	DEF461	Not Processed
54	Ong Boon Hong	GHI116	Not Processed
55	Ali Basha	JKL227	Not Processed
56	Loh Siew Ping	MN0338	Not Processed
57	Siti Rahayu	PQR449	Not Processed
58	Lim Boon Siew	STU560	Not Processed
59	Chan Siew Fun	VWX671	Not Processed
60	Zainuddin Basha	YYY782	Not Processed
61	Low Siew Fong	ABC129	Not Processed
62	Ahmad Nazri	XYZ785	Not Processed
63	Raja Norazlin	DEF462	Not Processed
64	Tan Ah Lek	GHI117	Not Processed
65	Roslina Binti Omar	JKL228	Not Processed
66	Chong Siew Fong	MN0339	Not Processed
67	Ahmad Zainuddin	PQR450	Not Processed
68	Leong Boon Keat	STU561	Not Processed

8.0 Lessons learned, reflective experience towards completing the individual/overall tasks and Conclusion

Amin: Engaging in this project was both challenging and rewarding. It allowed me to put my understanding of data structures, such as LinkedLists, Queues, and Stacks, into practice while developing a Java application designed for a vehicle service center. This experience extended beyond just coding—it involved finding effective solutions for real-world issues, like streamlining service bookings and managing customer payments.

Working with my group underscored the importance of strong communication and collaboration. We divided tasks efficiently, supported one another, and ensured that each component fit together cohesively. Creating UML diagrams and planning the overall system architecture helped me visualize the complete design before jumping into coding, highlighting the significance of preparation.

The project was not without its challenges. Processing extensive data and managing deadlines pushed me out of my comfort zone, but overcoming these obstacles made the end result all the more satisfying. This project has deepened my practical understanding of applying data structures and has equipped me with new insights that I am eager to carry forward into future work.

Ighwan: Working on this project was a challenging yet rewarding experience. It gave me the chance to apply what I've learned about data structures like LinkedLists, Queues, and Stacks to create a functional Java application for a vehicle service center. This wasn't just about coding—it was about solving real problems, like organizing service bookings and managing payments efficiently.

Collaborating with my group taught me how important communication and teamwork are in completing a complex project. We divided tasks, supported each other, and made sure everything came together smoothly. Designing UML diagrams and planning out the system helped me see the bigger picture before diving into the code.

There were definitely hurdles along the way—processing large data files and managing time under tight deadlines weren't easy. But pushing through these challenges made the final outcome even more satisfying. This project gave me a clearer understanding of how to apply my skills in real-world scenarios, and I'm excited to build on what I've learned.

Rifat:

This project deepened my understanding of OOP principles, such as encapsulation, abstraction, inheritance, and polymorphism. Implementing real-world entities like Booking, Payment, and Transaction as classes allowed me to appreciate how OOP simplifies complex systems by organizing data and behavior.

Designing the system involved analyzing requirements and translating them into functional modules. For example, ensuring seamless integration between bookings, payments, and transaction logs required careful planning of relationships and methods.

Working on this project was an enriching experience that strengthened my programming, problem-solving, and software design skills. It gave me a deeper appreciation for the power of programming in addressing real-world challenges and prepared me for tackling more advanced projects in the future.

9.0 References

- Github link for our project : <https://github.com/israfilrifat/Vehicle-Service-Center.git>
- CustomerList : <D:\\SEM-3 UPTM\\SWC3344\\Vehicle service Management\\VehicleServicemanagement - Copy\\CustomerList.txt>
- Oracle provides official Java documentation that explains classes, objects, collections, and data structures like Queue, Stack, and LinkedList.
<https://docs.oracle.com/javase/>
- BlueJ's integrated development environment helps visualize class diagrams and manage small-scale Java applications.
<https://bluej.org/>
- The basics of UML diagrams and how to structure a system using object-oriented principles can be found here:
<https://www.uml-diagrams.org/>
<https://refactoring.guru/design-patterns>
- Tutorials on implementing and using Queue, Stack, and LinkedList.
<https://www.geeksforgeeks.org/the-java-collections-framework/>
<https://www.javatpoint.com/java-collections>
- References to tools like Lucidchart, Draw.io, and StarUML for creating flowcharts and UML diagrams:
- Lucidchart UML Tutorials
- [Draw.io](#) Guide
- Explore similar open-source projects or seek help from the developer community for specific implementation queries.
<https://github.com/>
<https://stackoverflow.com/>