

GROUP 5 - OOP Project Report

RESUBMIT

by Muhammad Erfan Syabil ESA

Submission date: 16-Jan-2024 07:44AM (UTC-0800)

Submission ID: 2271886379

File name: Group_5_OOP_Project_-_Report.pdf (1.38M)

Word count: 2940

Character count: 16406



UNIVERSITI TEKNOLOGI MALAYSIA

**OBJECT-ORIENTED PROGRAMMING
(SECJ2154)**

SEMESTER 1 2023/2024

**GROUP PROJECT
CAR RENTAL SYSTEM**

**MUHAMMAD ERFAN SYABIL BIN ESA (B23CS0055)
AMIRUL HANI BIN SYAFRIZON (B23CS0025)
SARANYA A/P JAYARAMA REDDY (B22EC3013)
ISWARY A/P SHANMUGAM (B22EC3004)**

SECTION 01

**Lecturer:
MADAM LIZAWATI MI YUSUF**

16rd JANUARY 2023

SECTION A: PROJECT DESCRIPTION

Synopsis:

The Car Rental System is a system designed to help customers have a sustainable, efficient car renting service. The system aims to simplify the system process by guaranteeing a simplistic and ideal useful rental process.

General Concept:

1) User Registration

If there is no customer information stored in the system, users must add or register before renting their car. The user will be able to see their saved details under all customer information.

2) Car Rental Confirmation and Car Rental History

Users can choose one of the available cars by providing both a pick-up location and a return location. User must enter a date and time to pick up their car. Once all of the necessary information has been completed, the confirmation details will be displayed and the user can see their rental history.

3) Pickup Location and Return Location

Users can view the car pick-up address and return address during the rental registration process.

How to Use the System:

1) First: View Available Car and Rental Prices

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 1

--- Available Car and Rental Prices ---

      Type      Make   Model     Year  Rental Price
1. Compact    Perodua Axia   (2022) RM50.0 per day
2. Sports     Ford   Mustang  (2022) RM300.0 per day
3. Sedan      Proton  Saga    (2022) RM80.0 per day
4. MPV        Honda  BRV     (2022) RM100.0 per day
```

Users are required to enter 1 from the Car Rental System Menu in order to view the availability of cars and rental prices. The system will display the details of the available car and rental price for per day..

2) Second: Add Customer Details

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 2

Enter customer name: Saranya Reddy
Enter customer license number: 1818
Enter customer address: Teluk Intan, Perak
Enter customer email: Sara
Enter customer phone number: 123

Customer details added:
Name: Saranya Reddy
License Number: 1818
Address: Teluk Intan, Perak
Email: Sara
Phone No: 123
```

Users are required to add their necessary details as in, customer name, customer license number, customer address, customer email and customer phone number. Once the user adds their details, the customer details will be shown in the system.

3) Third: View All Customer Details

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 3

--- All Customer Information ---

Customer 1
-----
Name: Erfan Syabil
License Number: 021026060451
Address: Puchong, Selangor
Email: muhderfan2610@gmail.com
Phone No: 0198525011
-----

Customer 2
-----
Name: Amirul Hani
License Number: 020814112378
Address: Gombak, Kuala Lumpur
Email: amirulhani02@gmail.com
Phone No: 0173223121
-----

Customer 3
-----
Name: Saranya Jayarama
License Number: 010515113278
Address: Ipoh, Perak
Email: saranyaJayarama@yahoo.com.my
Phone No: 0178324320
-----
```

Users can enter 3 from the Car Rental System Menu to view all the customer information. The system will display both previous customer information and new customer details.

4) Fourth: Rent A Car

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 4

--- All Customer Information ---

Customer 1
-----
Name: Erfan Syabil
License Number: 021026060451
Address: Puchong, Selangor
Email: muhderfan2610@gmail.com
Phone No: 0198525011
-----

Customer 2
-----
Name: Amirul Hani
License Number: 020814112378
Address: Gombak, Kuala Lumpur
Email: amirulhani02@gmail.com
Phone No: 0173223121
-----

Customer 3
-----
Name: Saranya Jayarama
License Number: 010515113278
Address: Ipoh, Perak
Email: saranyaJayarama@yahoo.com.my
Phone No: 0178324320
-----
```

```

Customer 4
-----
Name: Iswary Aish
License Number: 010715442781
Address: Kuantan, Pahang
Email: aish@email.com.my
Phone No: 0132454438
-----
Customer 5
-----
Name: Saranya Reddy
License Number: 1818
Address: Teluk Intan, Perak
Email: Sara
Phone No: 123
-----
Enter the index of the customer to make a reservation for: 5
--- Pickup Locations ---
1. 123 BP 11, Puchong, Selangor, 47120
2. 456 NP 32, Kulai, Johor, 81310
3. 789 Villa, Kuantan, Pahang, 43543
4. 102 Elm St, Gombak, Kuala Lumpur, 91232
5. 879 Oak St, Ipoh, Perak, 30000
6. 321 Pine St, Jeli, Kelantan, 54321
Enter the index of the pickup location: 5
--- Available Car and Rental Prices ---
Type      Make     Model      Year   Rental Price
1. Compact  Perodua Axia    (2022) RM50.0 per day
2. Sports   Ford       Mustang   (2022) RM300.0 per day
3. Sedan    Proton Saga    (2022) RM80.0 per day
4. MPV      Honda      BRV      (2022) RM100.0 per day
Enter the car index to reserve (1-4): 4

```

```

--- Return Locations ---
1. 123 BP 11, Puchong, Selangor, 47120
2. 456 NP 32, Kulai, Johor, 81310
3. 789 Villa, Kuantan, Pahang, 43543
4. 102 Elm St, Gombak, Kuala Lumpur, 91232
5. 879 Oak St, Ipoh, Perak, 30000
6. 321 Pine St, Jeli, Kelantan, 54321
Enter the index of the return location: 5
Enter appointment date (yyyy-MM-dd HH:mm:ss): 2024-01-18 12:00:00
Enter the duration of rental in days: 3
-----
Rental Successful for: Saranya Reddy
Name: Saranya Reddy
License Number: 1818
Address: Teluk Intan, Perak
Email: Sara
Phone Number: 123
Car rented: MPV      Honda      BRV      (2022)
Pickup date: Thu Jan 18 12:00:00 MYT 2024
Pickup location: 879 Oak St, Ipoh, Perak, 30000
Return location: 879 Oak St, Ipoh, Perak, 30000
Duration: 3 day(s)
Total: RM300.00
-----
```

Users have to enter 4 to rent a car. All the customer information will be shown in the system and users need to proceed by entering the index of the customer to make a reservation. Next, users need to select a pickup location followed by choosing a car according to their preference. Next, users are required to choose the return location by providing details such as appointment date, time and duration of rental days. Lastly, a confirmation message will be shown in the system indicating the successful rental for the specified customer along with the customer details, car rental details and the total payment.

5) View Rental History

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 5

--- Rental History ---

Name: Saranya Reddy
License Number: 1818
Address: Teluk Intan, Perak
Email: Sara
Phone Number: 123

Car rented: MPV      Honda   BRV      (2022)
Pickup date: Thu Jan 18 12:00:00 MYT 2024
Pickup location: 879 Oak St, Ipoh, Perak, 30000
Return location: 879 Oak St, Ipoh, Perak, 30000
Duration: 3 day(s)

Total: RM300.00
-----
```

Users can enter 5 to view their rental history. The rental history will display customer details, car rental details and the total payment.

6) Exit

Users can enter 6 to exit from the car rental system.

Objective:

The objective of the Car Rental System is to ease the car rental process for customers. The system aims to enhance the overall experience of renting a car by providing a user-friendly and simple interface, ensuring accurate documentation and facilitating efficient management for rental companies.

Scope

The Car Rental System's scope includes various functions that help in renting cars efficiently and user-friendly. This system is designed to satisfy the needs of customers.

Below is the description of the scope:

1) Customer registration

- Users have the option to register customers and view their details under customer information.

2) Car rental confirmation and car rental history

- Users can select cars and specify pickup and return location.
- Confirmation details and rental history can be previewed.

3) Pickup address and return address

- Users can view details about pickup and return location.

4) System usage steps:

- View available car and rental prices
- Add customer details
- View all customer details
- Rent a car
- View rental history
- Exit

Workflow:

Pseudocode RentalApp:

1. **Begin**

2. **Initialize** RentalSystem object: rentalSystem

3. **Initialize** Customer object: customer (set to null initially)

4. **Loop** until the user chooses to **exit**:

a. **Display** the Car Rental System Menu:

 1. View available vehicles and rental prices

 2. Add user details

 3. View all customer details

 4. Rent a car

 5. View rental history

 6. Exit

 7. Enter your choice (1-6):

 6
b. **Prompt** the user to enter their choice (1-6).

 3
c. **Read** the user's choice.

d. **Switch** based on the user's choice:

 - Case 1:

 - Call rentalSystem.displayAvailableCar()

 - Case 2:

 - Prompt the **user** to enter customer details (name, licenseNumber, address, email, phoneNumber).

 - Create a new Customer object with the entered details.

 - Call rentalSystem.addUserDetails(customer)

 - Print "Customer details added: "

 - Print "Name: " + customer.getName()

 - Print "License Number: " + customer.getLicenseNumber()

- Print "Address: " + customer.getAddress()
- Print "Email: " + customer.getEmail()
- Print "Phone No: " + customer.getPhoneNumber()

- Case 3:
 - Call rentalSystem.printAllCustomers()

- Case 4:
 - IF rentalSystem.hasCustomerDetails() // user details (customer) exist:
 - Call rentalSystem.printAllCustomers().
 - Prompt user to enter index of customer
 - IF customerIndex >= 1 && customerIndex <=

 rentalSystem.getCustomers().size()

 - Get selected customer based on index
 - Call rentalSystem.displayPickupLocations()
 - Prompt user to enter index of pickup location
 - Get selected pickup location
 - Call rentalSystem.displayAvailableCar()
 - Prompt user to enter index of vehicle to reserve
 - Get selected car based on index
 - Call rentalSystem.displayReturnLocations()
 - Prompt user to enter index of return location
 - Get selected return location
 - Schedule an appointment by calling rentalSystem.scheduleAppointment()
 - Prompt user to enter rental duration in days
 - Get entered rental duration
 - Call rentalSystem.rentVehicle(customer, selectedCar, appointment, pickupLocation, returnLocation, rentalDays)

 - ELSE Print “Invalid customer index.”

- ELSE Print "Please add user details first."
- Case 5:
 - Call rentalSystem.displayRentals()
- Case 6:
 - Print message "Exiting Rental System. Thank you!".
- Default:
 - Print message "Invalid choice. Please enter a number between 1 and 6".

6. Handle exceptions if they occur
7. Close scanner by calling rentalSystem.closeScanner()

End

OO Concepts:**Class and Object:**

Class: Represents a blueprint for creating objects. For example, classes like Car, SportCar, Customer, Appointment, CompactCar, Location, MPVCar, Rentable, Rental, RentalApp, RentalSystem and SedanCar.

Object: Instances of classes. For instance, an object of the SportCar class represents a specific sports car.

Inheritance:

Usage: Inheritance is utilized to model an "is-a" relationship between classes. For example, SportCar, MPVCar, SedanCar, and CompactCar extend the Car class, inheriting common properties and behaviors.

Abstraction:

Usage: Abstract classes (Car) and interfaces (Rentable) are used to abstract common properties and behaviors, allowing for code reusability and the definition of a common interface for rental rates.

Encapsulation:

Usage: Data hiding and encapsulation are achieved by making the fields of classes (Car, Customer, etc.) private and providing public methods to access and manipulate them. This protects the internal state of objects.

Polymorphism:

Usage: Polymorphism is demonstrated through method overriding. For example, the `getRentalRate` method is overridden in each vehicle class (`SportCar`, `MPVCar`, `SedanCar` and `CompactCar`) to provide different rental rates.

Composition:

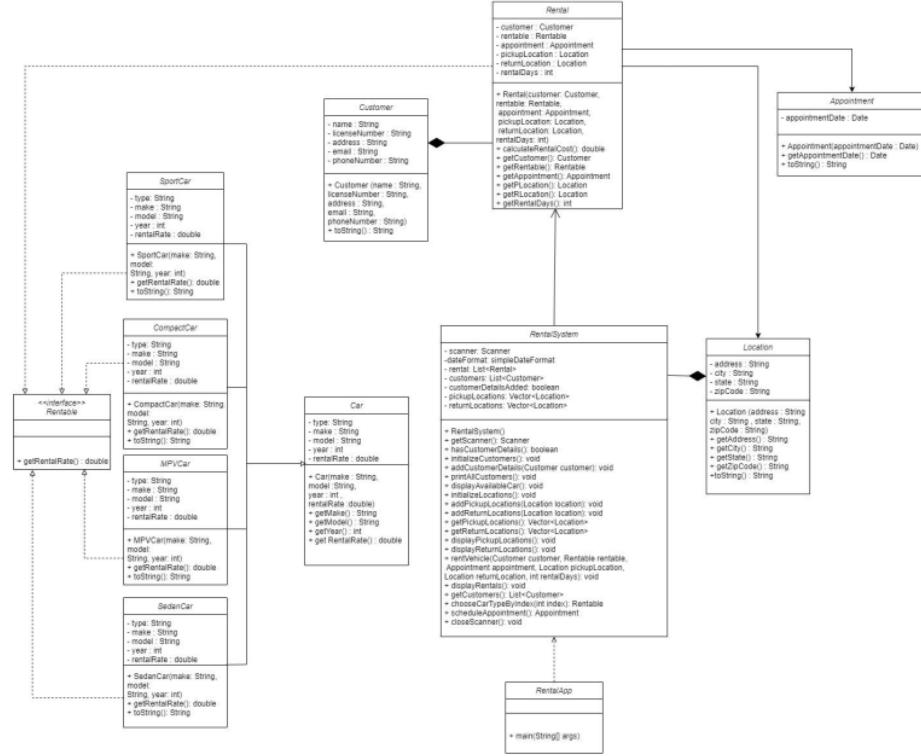
Usage: The system uses composition to build complex objects. For example, a `Rental` object includes instances of `Customer`, `Rentable` (vehicle), `Appointment`, and `Location`.

Interface:

Usage: The `Rentable` interface defines a common method (`getRentalRate`) that is implemented by car classes (`SportCar`, `MPVCar`, `SedanCar`, and `CompactCar`). This allows for a consistent way to retrieve rental rates.

These object-oriented concepts contribute to the design principles of modularity, extensibility, and maintainability. The system's structure is organized around well-defined classes, promoting code reuse and flexibility. Each class encapsulates its functionality, and relationships between classes reflect real-world associations, enhancing the system's modeling capabilities.

SECTION B: CLASS DIAGRAMS



Class Rental System:

Attributes	Description
scanner	An instance of the Scanner class for input handling.
dateFormat	An instance of the SimpleDateFormat class for date formatting.
rental	A list containing Rental objects.
customers	A list containing Customer objects.
customerDetailsAdded	A boolean indicating whether customer details have been added.
pickupLocations	A Vector containing Location objects for pickup.
returnLocations	A Vector containing Location objects for return.
Methods	Description
RentalSystem()	Constructor for initializing the RentalSystem.
getScanner(): Scanner	Getter method for the Scanner instance.

<code>hasCustomerDetails(): boolean</code>	Checks if customer details have been added.
<code>initializeCustomers(): void</code>	Initializes the list of customers.
<code>addCustomerDetails(Customer customer): void</code>	Adds customer details to the list.
<code>printAllCustomers(): void</code>	Prints details of all customers.
<code>displayAvailableCars(): void</code>	Displays available cars for rent.
<code>initializeLocations(): void</code>	Initializes pickup and return locations.
<code>addPickupLocations(Location location): void</code>	Adds a pickup location to the list.
<code>addReturnLocations(Location location): void</code>	Adds a return location to the list.
<code>getPickupLocations(): Vector<Location></code>	Getter method for pickup locations.
<code>getReturnLocations(): Vector<Location></code>	Getter method for return locations.
<code>displayPickupLocations(): void</code>	Displays available pickup locations.
<code>displayReturnLocations(): void</code>	Displays available return locations.
<code>rentVehicle(Customer customer, Rentable rentable, Appointment appointment, Location pickupLocation, Location returnLocation, int rentalDays): void</code>	Rents a vehicle for a specified duration.
<code>displayRentals(): void</code>	Displays details of all rentals.
<code>getCustomers(): List<Customer></code>	Getter method for the list of customers.
<code>chooseCarTypeByIndex(int index): Rentable</code>	Selects a Rentable object based on the index.
<code>scheduleAppointment(): Appointment</code>	Creates and returns an Appointment object.
<code>closeScanner(): void</code>	Closes the Scanner instance.

Class Rental App:

Attributes	Description
Method	Description
main(String[] args)	The main method that serves as the entry point for the RentalApp. It is responsible for executing the rental application, taking command-line arguments (if any), and coordinating the overall functionality of the rental system.

Class Customer:

Attributes	Description
name (String)	The name of the customer.
licenseNumber (String)	The license number associated with the customer.
address (String)	The address of the customer.
email (String)	The email address of the customer.
phoneNumber (String)	The phone number of the customer.
Method	Description
toString() : String:	This method returns a string representation of the customer, encapsulating their information. It is commonly used for debugging, logging, or displaying customer details in a user-friendly format. The method overrides the default <code>toString()</code> implementation to provide a customized representation of the Customer object.

Class Rental:

Attributes	Description
customer (Customer)	The customer associated with the rental.
rentable (Rentable)	The item being rented.
appointment (Appointment)	The scheduled appointment for the rental.
pickupLocation (Location)	The location where the customer picks up the rented item.
returnLocation (Location)	The location where the customer returns the rented item.
rentalDays (int)	The number of days for which the item is rented.
Method	Description
calculateRentalCost () : double	Calculates and returns the rental cost based on the rental duration and the specific rentable item.
getCustomer() : Customer	Returns the associated customer.
getRentable() : Rentable	Returns the rented item.
getAppointment() : Appointment	Returns the scheduled appointment for the rental.
getPLocation() : Location	Returns the pickup location.
getRLocation() : Location	Returns the return location.
getRentalDays() : int	Returns the number of days for which the item is rented.

Class Location:

Attributes	Description
address (String)	The street address of the location.
city (String)	The city where the location is situated.
state (String)	The state or region of the location.
zipCode (String)	The ZIP code associated with the location.
Method	Description
getAddress() : String	Returns the street address of the location.
getCity() : String	Returns the city of the location.
getState() : String	Returns the state or region of the location.
getZipCode() : String	Returns the ZIP code of the location.

<code>toString() : String</code>	Returns a string representation of the location, providing a concise and readable summary of its address, city, state, and ZIP code.
----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------

Class Appointment:

Attributes	Description
<code>appointmentDate (Date)</code>	The date and time of the scheduled appointment.
Method	Description
<code>getAppointmentDate() : Date</code>	Returns the date and time of the scheduled appointment.
<code>toString() : String</code>	Returns a string representation of the appointment, providing a concise and readable summary of its date and time.

Class Rentable:

Attributes	Description
Method	Description
<code>getRentalRate() : double</code>	Returns the rental rate for the object. Classes implementing this interface will provide their own logic for determining the rental rate.

Class SportCar:

Attributes	Description
<code>type(String)</code>	The type of the car
<code>make (String)</code>	The make or brand of the sports car.
<code>model (String)</code>	The model name of the sports car.
<code>year (int)</code>	The manufacturing year of the sports car.
Method	Description
<code>getRentalRate() : double</code>	Returns the rental rate specific to the sports car.
<code>toString() : String</code>	Returns a string representation of the sports car, providing a concise and readable summary of its make, model, and year.

Class CompactCar:

Attributes	Description
type(String)	The type of the car
make (String)	The make or brand of the compact car.
model (String)	The model name of the compact car.
year (int)	The manufacturing year of the compact car.
Method	Description
getRentalRate() : double	Returns the rental rate specific to the compact car.
toString() : String	Returns a string representation of the compact car, providing a concise and readable summary of its make, model, and year.

Class MPVCar:

Attributes	Description
type(String)	The type of the car
make (String)	The make or brand of the MPV car.
model (String)	The model name of the MPV car.
year (int)	The manufacturing year of the MPV car.
Method	Description
getRentalRate() : double	Returns the rental rate specific to the MPV car.
toString() : String	Returns a string representation of the MPV car, providing a concise and readable summary of its make, model, and year.

Class SedanCar:

Attributes	Description
type(String)	The type of the car
make (String)	The make or brand of the Sedan car.
model (String)	The model name of the Sedan car.
year (int)	The manufacturing year of the Sedan car.
Method	Description
getRentalRate() : double	Returns the rental rate specific to the Sedan car.
toString() : String	Returns a string representation of the Sedan car, providing a concise and readable summary of its make, model, and year.

Class Car:

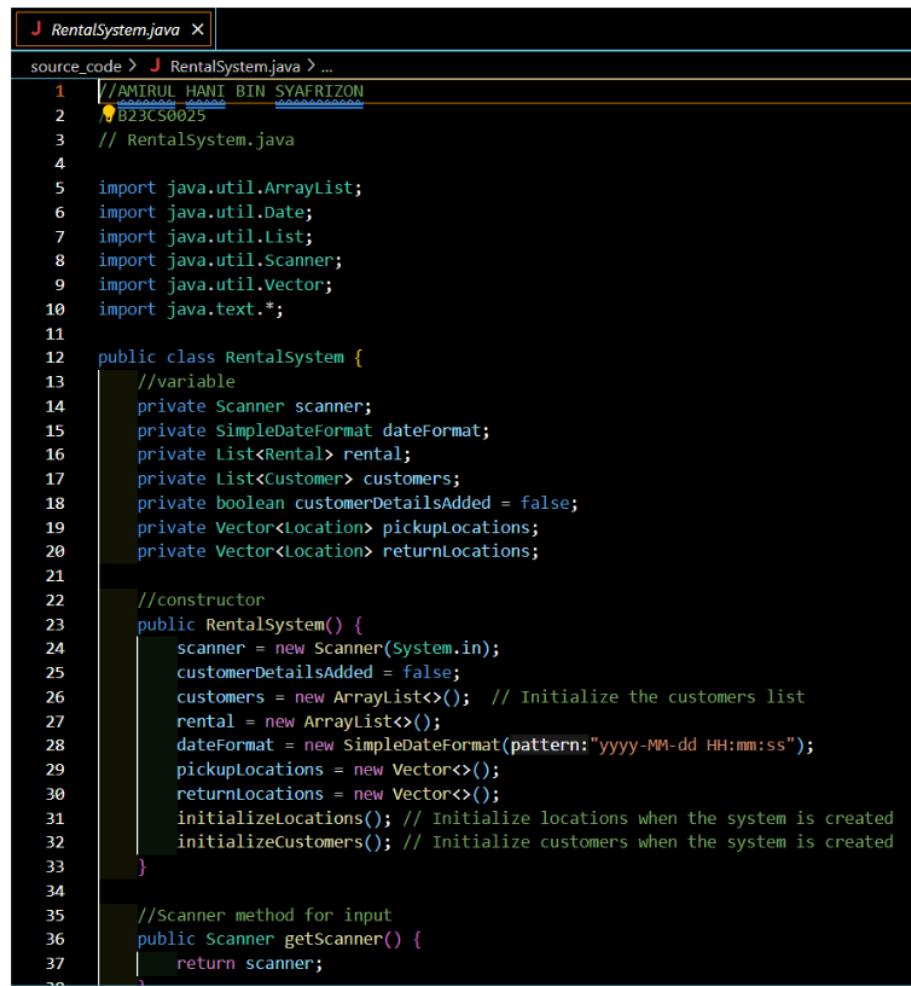
Attributes	Description
type(String)	The type of the car
make (String)	The make or brand of the car.
model (String)	The model name of the car.
year (int)	The manufacturing year of the car.
rentalRate (double)	The rental rate associated with the car.
Method	Description
getMake() : String	Returns the make of the car.
getModel() : String	Returns the model name of the car.
getYear() : int	Returns the manufacturing year of the car.
getRentalRate() : double	Returns the rental rate associated with the car.

SECTION C: SOURCE CODE AND USER MANUAL

In this section, you need to provide the source code of your system. The source code serves as a way to present the object-oriented concepts used in your system. You are required to provide the user manual of your system. The user manual describes how to use the system and the flow of your system, i.e. shows the example of input and the expected output.

SOURCE CODE:

1. RentalSystem.java



A screenshot of a Java code editor showing the `RentalSystem.java` file. The code is a Java class named `RentalSystem` with various fields and methods. The code is color-coded, and line numbers are visible on the left. A cursor is positioned at the start of the class definition. The code includes imports for `java.util.ArrayList`, `java.util.Date`, `java.util.List`, `java.util.Scanner`, `java.util.Vector`, and `java.text.*`. It defines private variables for `scanner`, `dateFormat`, `rental`, `customers`, `customerDetailsAdded`, `pickupLocations`, and `returnLocations`. The constructor initializes these variables and calls `initializeLocations()` and `initializeCustomers()`. The `getScanner()` method returns the `scanner` object.

```
source_code > J RentalSystem.java > ...
1 //AMIRUL HANI BIN SYAFRIZON
2 // B23CS0025
3 // RentalSystem.java
4
5 import java.util.ArrayList;
6 import java.util.Date;
7 import java.util.List;
8 import java.util.Scanner;
9 import java.util.Vector;
10 import java.text.*;
11
12 public class RentalSystem {
13     //variable
14     private Scanner scanner;
15     private SimpleDateFormat dateFormat;
16     private List<Rental> rental;
17     private List<Customer> customers;
18     private boolean customerDetailsAdded = false;
19     private Vector<Location> pickupLocations;
20     private Vector<Location> returnLocations;
21
22     //constructor
23     public RentalSystem() {
24         scanner = new Scanner(System.in);
25         customerDetailsAdded = false;
26         customers = new ArrayList<>(); // Initialize the customers list
27         rental = new ArrayList<>();
28         dateFormat = new SimpleDateFormat(pattern:"yyyy-MM-dd HH:mm:ss");
29         pickupLocations = new Vector<>();
30         returnLocations = new Vector<>();
31         initializeLocations(); // Initialize locations when the system is created
32         initializeCustomers(); // Initialize customers when the system is created
33     }
34
35     //Scanner method for input
36     public Scanner getScanner() {
37         return scanner;
38     }
}
```

```

38     }
39
40     //Check whether is there customer detail
41     public boolean hasCustomerDetails() {
42         return customerDetailsAdded;
43     }
44
45     //Initialize customer
46     public void initializeCustomers() {
47         customers.add(new Customer(name:"ErFan Syabill", licenseNumber:"021026060451", address:"Puchong, Selangor", email:"muhenderfan261@gmail.com", phoneNumber:"0198525011"));
48         customers.add(new Customer(name:"Amirul Hanif", licenseNumber:"028814112378", address:"Gombak, Kuala Lumpur", email:"amirulhanif02@gmail.com", phoneNumber:"0173223121"));
49         customers.add(new Customer(name:"Saranya Red", licenseNumber:"018515113279", address:"Ipoh, Perak", email:"saranyaayaramag@yahoo.com.my", phoneNumber:"0178324528"));
50         customers.add(new Customer(name:"Iswary Aish", licenseNumber:"010715442701", address:"Kuantan, Pahang", email:"dish@email.com.my", phoneNumber:"0132454430"));
51         customerDetailsAdded = true;
52     }
53
54     //To add user in the Customer List
55     public void addCustomerDetails(Customer customer) {
56         customerDetailsAdded = true;
57         customers.add(customer);
58     }
59
60     //Display all Customer in the list
61     public void printAllCustomers() {
62         System.out.println(x"\n--- All Customer Information ---");
63         System.out.println(x"-----");
64         System.out.println("Name \t" + "License Number \t" + "Address \t" + "Email \t" + "Phone No");
65         System.out.println(x"-----");
66         int i = 1;
67         for (Customer customer : customers) {
68             System.out.println(i + "\t" + customer.getName() + "\t" + customer.getLicenseNumber() + "\t" + customer.getAddress() + "\t" + customer.getEmail() + "\t" + customer.getPhoneNumber());
69             System.out.println(x"-----");
70             i++;
71         }
72     }
73
74     //Display all available Car by retrieving the list in chooseCarTypeByIndex() method
75     public void displayAvailableCar() {
76         System.out.println(x"\n--- Available Car and Rental Prices ---");
77         System.out.println("\t" + "Type" + "\t" + "Make" + "\t" + "Model" + "\t" + "Year" + "\t" + "Rental Price");
78         for (int i = 1; i <= 4; i++) {
79             Rentable car = chooseCarTypeByIndex(i);
80             System.out.println(i + ". " + car.toString() + "\t" + "RM" + car.getRentalRate() + " per day");
81         }
82     }
83
84     //Initialize pickup and return location
85     private void initializeLocations() {
86         //Pickup Location
87         pickupLocations.add(new Location(address:"123 BP 11", city:"Puchong", state:"Selangor", zipCode:"47120"));
88         pickupLocations.add(new Location(address:"456 NP 32", city:"Kulai", state:"Johor", zipCode:"81310"));
89         pickupLocations.add(new Location(address:"789 Villa", city:"Kuantan", state:"Pahang", zipCode:"43543"));
90         pickupLocations.add(new Location(address:"102 Elm St", city:"Gombak", state:"Kuala Lumpur", zipCode:"91232"));
91         pickupLocations.add(new Location(address:"879 Oak St", city:"Ipoh", state:"Perak", zipCode:"30000"));
92         pickupLocations.add(new Location(address:"321 Pine St", city:"Jeli", state:"Kelantan", zipCode:"54321"));
93
94         //Return Location
95         returnLocations.add(new Location(address:"123 BP 11", city:"Puchong", state:"Selangor", zipCode:"47120"));
96         returnLocations.add(new Location(address:"456 NP 32", city:"Kulai", state:"Johor", zipCode:"81310"));
97         returnLocations.add(new Location(address:"789 Villa", city:"Kuantan", state:"Pahang", zipCode:"43543"));
98         returnLocations.add(new Location(address:"102 Elm St", city:"Gombak", state:"Kuala Lumpur", zipCode:"91232"));
99         returnLocations.add(new Location(address:"879 Oak St", city:"Ipoh", state:"Perak", zipCode:"30000"));
100        returnLocations.add(new Location(address:"321 Pine St", city:"Jeli", state:"Kelantan", zipCode:"54321"));
101    }
102
103    //Add pickup location into the vector location
104    public void addPickupLocation(Location location) {
105        pickupLocations.add(location);

```

```

106     }
107
108     //Add return location into the vector location
109     public void addReturnLocation(Location location) {
110         returnLocations.add(location);
111     }
112
113     //Retrieve pickup location object
114     public Vector<Location> getPickupLocations() {
115         return pickupLocations;
116     }
117
118     //Retrieve return location object
119     public Vector<Location> getReturnLocations() {
120         return returnLocations;
121     }
122
123     //Display pickup location
124     public void displayPickupLocations() {
125         System.out.println("n--- Pickup Locations ---");
126         int i = 1;
127         for (Location pickupLocation : pickupLocations) {
128             System.out.print(i + ". " + pickupLocation.getAddress() + ", " + pickupLocation.getCity() + ", " + pickupLocation.getState() + ", " + pickupLocation.getZipCode());
129             i++;
130         }
131     }
132
133     //Display return location
134     public void displayReturnLocations() {
135         System.out.println("n--- Return Locations ---");
136
136         int i = 1;
137         for (Location returnLocation : returnLocations) {
138             System.out.print(i + ". " + returnLocation.getAddress() + ", " + returnLocation.getCity() + ", " + returnLocation.getState() + ", " + returnLocation.getZipCode());
139             i++;
140         }
141     }
142
143
144     //Rent a vehicle and after that produce output containing info about customer, car rented, pickup time, pickup and return location, duration and total
145     public void rentVehicle(Customer customer, Rentable rentable, Appointment appointment, Location pickupLocation, Location returnLocation, int rentalDays) {
146         Rental rental = new Rental(customer, rentable, appointment, pickupLocation, returnLocation, rentalDays);
147         this.rental.add(rental);
148
149         System.out.println("-----");
150         System.out.println("Rental Successful For: " + customer.getName());
151
152         System.out.println("Name: " + customer.getName());
153         System.out.println("License Number: " + customer.getLicenseNumber());
154         System.out.println("Address: " + customer.getAddress());
155         System.out.println("Email: " + customer.getEmail());
156         System.out.println("Phone Number: " + customer.getPhoneNumber());
157
158         System.out.println("Car rented: " + rentable);
159         System.out.println("Pickup date: " + rental.getAppointment().getAppointmentDate());
160         System.out.println("Pickup location: " + rental.getLocation().toString());
161         System.out.println("Return location: " + rental.getReturnLocation().toString());
162         System.out.println("Duration: " + rental.getRentalDays() + " day(s)");
163         System.out.printf("Total: %.2f", rental.calculateRentalCost());
164         System.out.println("-----");
165     }
166
167     //To display rental history
168     public void displayRentals() {
169         System.out.println("n--- Rental History ---");
170         for (Rental rental : rentals) {
171             System.out.println("Name: " + rental.getCustomer().getName());
172             System.out.println("License Number: " + rental.getCustomer().getLicenseNumber());
173             System.out.println("Address: " + rental.getCustomer().getAddress());
174             System.out.println("Email: " + rental.getCustomer().getEmail());
175             System.out.println("Phone Number: " + rental.getCustomer().getPhoneNumber());
176
177             System.out.println("Car rented: " + rental.getRental());
178             System.out.println("Pickup date: " + rental.getAppointment().getAppointmentDate());
179             System.out.println("Pickup location: " + rental.getPLocation().toString());
180             System.out.println("Return location: " + rental.getRLocation().toString());
181         }
182     }

```

```

181     |         System.out.println("Duration: " + rental.getRentalDays() + " day(s)");
182     |         System.out.printf(format:"\nTotal: RM%.2f", rental.calculateRentalCost());
183     |         System.out.println(x:"\n-----");
184   }
185 }
186
187 //Close scanner to prevent resource leak
188 public void closeScanner() {
189   scanner.close();
190 }
191
192 //Initialize car into arrayList and return the list index
193 public Rentable chooseCarTypeByIndex(int carIndex) {
194   // Choose a vehicle based on index
195   List<Rentable> availableCar = new ArrayList<>();
196   availableCar.add(new CompactCar(type:"Compact",make:"Perodua", model:"Axia", year:2022));
197   availableCar.add(new SportCar(type:"Sports", make:"Ford", model:"Mustang", year:2022));
198   availableCar.add(new SedanCar(type:"Sedan", make:"Proton", model:"Saga", year:2022));
199   availableCar.add(new MPVCar(type:"MPV", make:"Honda", model:"BRV", year:2022));
200
201   if (carIndex >= 1 && carIndex <= availableCar.size()) {
202     return availableCar.get(carIndex - 1);
203   } else {
204     throw new IllegalArgumentException(s:"Invalid vehicle index.");
205   }
206 }
207
208 //Set appointment using appropriate format of date and time
209 public Appointment scheduleAppointment() {
210   System.out.print(s:"Enter appointment date (yyyy-MM-dd HH:mm:ss): ");
211   String dateString = scanner.nextLine();
212
213   try {
214     Date appointmentDate = dateFormat.parse(dateString);
215     return new Appointment(appointmentDate);
216   } catch (ParseException e) {
217     System.out.println(x:"Invalid date format. Defaulting to current date and time.");
218     return new Appointment(new Date());
219   }
220 }
221
222 //Retrieve customer object.
223 public List<Customer> getCustomers() {
224   return customers;
225 }
226 }
```

2. RentalApp.java

The screenshot shows a Java code editor with the file 'RentalApp.java' open. The code is a simple application for a car rental system. It starts with comments identifying the author and date. The main method initializes a 'RentalSystem' object and enters a loop where it prints a menu of six options: viewing available cars, adding customer details, viewing all customer details, renting a car, viewing rental history, and exiting. It then reads the user's choice and executes the corresponding action. The code uses standard Java syntax with imports for System.out.println and Scanner.

```
1 //ANTRUL HANI BIN SYAFRIZON
2 //B23CS0025
3 // RentalApp.java
4
5 public class RentalApp {
6     Run | Debug
7     public static void main(String[] args) {
8         RentalSystem rentalSystem = new RentalSystem();
9
10    try {
11        int choice;
12        Customer customer = null;
13        do {
14            System.out.println("\n--- Car Rental System Menu ---");
15            System.out.println("1. View available car and rental prices");
16            System.out.println("2. Add customer details");
17            System.out.println("3. View all customer details");
18            System.out.println("4. Rent a car");
19            System.out.println("5. View rental history");
20            System.out.println("6. Exit");
21            System.out.print("Enter your choice (1-6): ");
22
23            choice = rentalSystem.getScanner().nextInt();
24            rentalSystem.getScanner().nextLine();
25
26            switch (choice) {
27                case 1:
28                    // View available vehicles and rental prices
29                    rentalSystem.displayAvailableCar();
30                    break;
31                case 2:
32                    // Add user details
33                    System.out.print("\nEnter customer name: ");
34                    String name = rentalSystem.getScanner().nextLine();
35
36                    System.out.print("Enter customer license number: ");
37                    String licenseNumber = rentalSystem.getScanner().nextLine();
38
39            }
40        } while (choice != 6);
41    } catch (Exception e) {
42        e.printStackTrace();
43    }
44 }
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
```

```

J RentalApp.java X
source_code > J RentalApp.java > ...
174 rentalSystem.displayPickupLocations();
175 System.out.print("Enter the index of the pickup location: ");
176 int pickupIndex = rentalSystem.getScanner().nextInt();
177 rentalSystem.getScanner().nextLine();
178
179 Location pickupLocation = rentalSystem.getPickupLocations().get(pickupIndex - 1);
180
181
182 //Choose car to rent
183 rentalSystem.displayAvailableCar();
184 System.out.print("Enter the car index to reserve (1-4): ");
185 int carIndex = rentalSystem.getScanner().nextInt();
186 rentalSystem.getScanner().nextLine();
187 rentalSystem.getScanner().nextLine();
188 Rentable selectedCar = rentalSystem.chooseCarTypeByIndex(carIndex);
189
190
191 // Choose return location
192 rentalSystem.displayReturnLocations();
193 System.out.print("Enter the index of the return location: ");
194 int returnIndex = rentalSystem.getScanner().nextInt();
195 rentalSystem.getScanner().nextLine();
196
197 Location returnLocation = rentalSystem.getReturnLocations().get(returnIndex - 1);
198
199 Appointment appointment = rentalSystem.scheduleAppointment();
200
201 System.out.print("Enter the duration of rental in days: ");
202 int rentalDays = rentalSystem.getScanner().nextInt();
203 rentalSystem.getScanner().nextLine();
204
205 rentalSystem.rentVehicle(customer, selectedCar, appointment, pickupLocation, returnLocation, rentalDays);
206 } else {
207 | System.out.println("Invalid customer index.");
208 }
209
210 break;
211
212 case 5:
213 | // View rental history
214 | rentalSystem.displayRents();
215 | break;
216
217 case 6:
218 | System.out.println("Exiting Rental System. Thank you!");
219 | break;
220 default:
221 | System.out.println("Invalid choice. Please enter a number between 1 and 5.");
222 }
223 } while (choice != 6);
224
225 } catch (Exception e) {
226 | System.out.println("An error occurred: " + e.getMessage());
227 } finally {
228 | // Close the scanner
229 | rentalSystem.closeScanner();
230 }
231

```

3. Rental.java

```
Rental.java X
source code > J Rental.java > ...
1 //SARANYA A/P JAYARAMA REDDY (B22EC3013)
2
3 public class Rental {
4     private Customer customer;
5     private Rentable rentable;
6     private Appointment appointment;
7     private Location pickupLocation;
8     private Location returnLocation;
9     private int rentalDays;
10
11     public Rental(Customer customer, Rentable rentable, Appointment appointment, Location pickupLocation, Location returnLocation, int rentalDays){
12         this.customer = customer;
13         this.rentable = rentable;
14         this.appointment = appointment;
15         this.pickuplocation = pickupLocation;
16         this.returnlocation = returnlocation;
17         this.rentalDays = rentalDays;
18     }
19
20     //Method to calculate the total rental
21     public double calculateRentalCost(){
22         return rentable.getRentalRate() * rentalDays;
23     }
24
25     //Getter method
26     public Customer getCustomer() {
27         return customer;
28     }
29
30     public Rentable getRentable() {
31         return rentable;
32     }
33
34     public Appointment getAppointment() {
35         return appointment;
36     }
37
38     public Location getLocation(){
39         return pickupLocation;
40     }
41
42     public Location getRLocation(){
43         return returnLocation;
44     }
45
46     public int getRentalDays(){
47         return rentalDays;
48     }
49 }
```

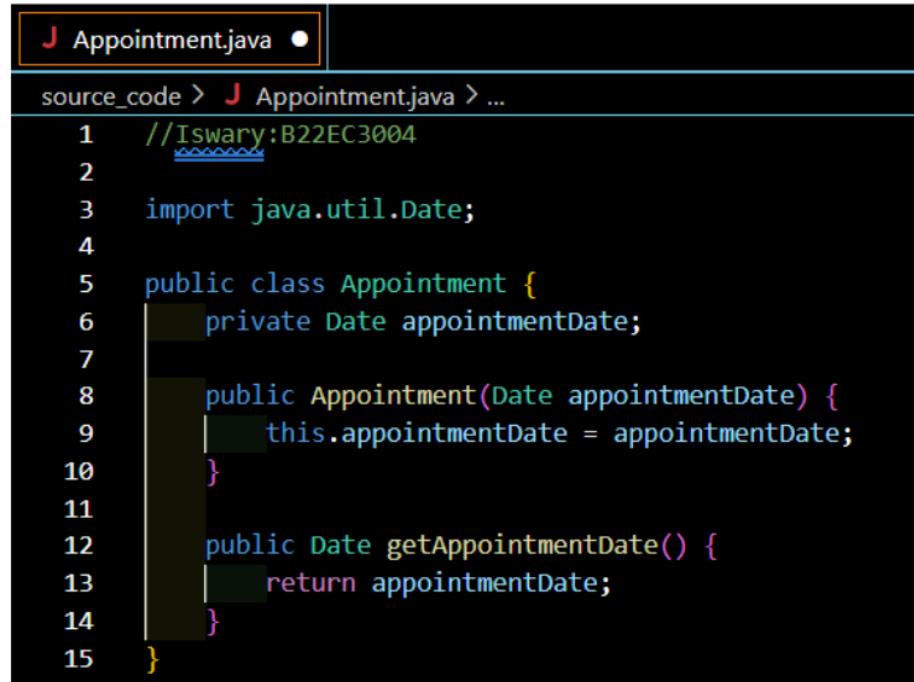
4. Rentable.java

```
source_code > J Rentable.java > Rentable
1 //SARANYA A/P JAYARAMA REDDY (B22EC3013)
2
3 public interface Rentable {
4     double getRentalRate();
5 }
6
```

5. Location.java

```
source_code > J Location.java > ...
1 //ISWARY:B22EC3004
2
3 public class Location {
4     private String address;
5     private String city;
6     private String state;
7     private String zipCode;
8
9     public Location(String address, String city, String state, String zipCode) {
10         this.address = address;
11         this.city = city;
12         this.state = state;
13         this.zipCode = zipCode;
14     }
15
16     // Getter methods
17     public String getAddress() {
18         return address;
19     }
20
21     public String getCity() {
22         return city;
23     }
24
25     public String getState() {
26         return state;
27     }
28
29     public String getZipCode() {
30         return zipCode;
31     }
32
33     //toString method to get location as a String
34     public String toString(){
35         return getAddress() + ", " + getcity() + ", " + getstate() + ", " + getzipCode();
36     }
37 }
```

6. Appoinment.java

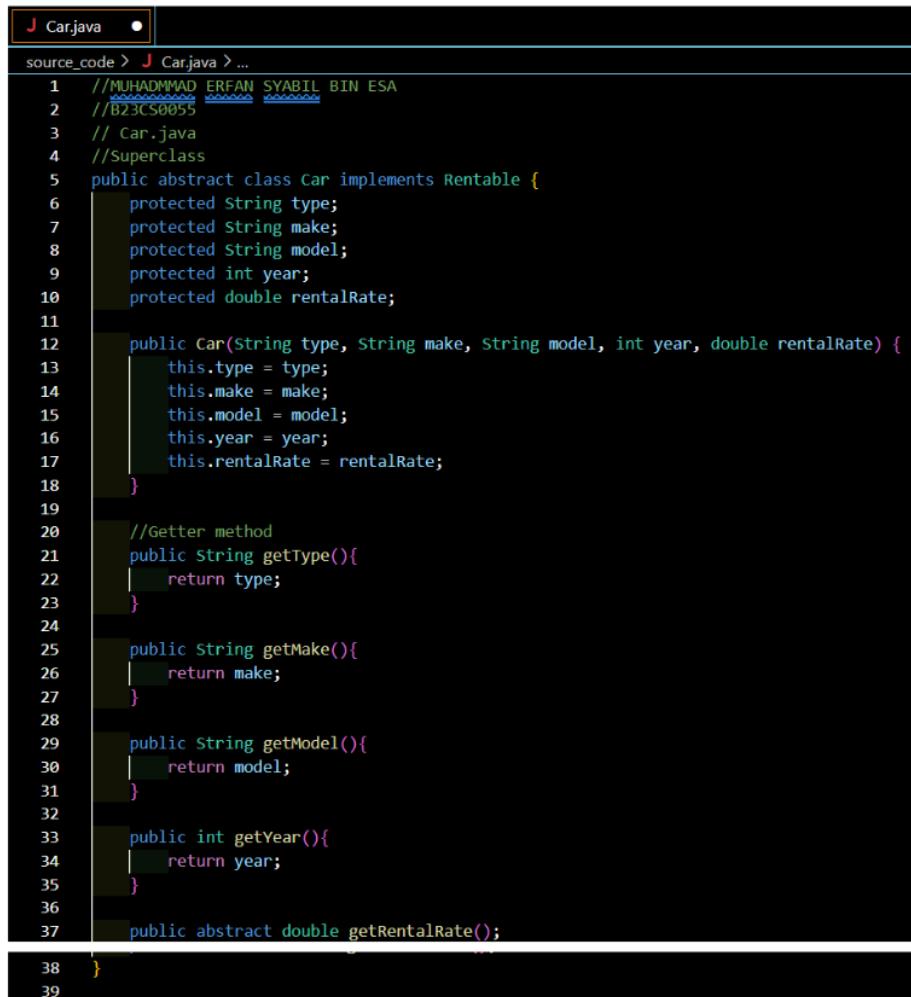


```
source_code > J Appointment.java > ...
1  //Iswary:B22EC3004
2
3  import java.util.Date;
4
5  public class Appointment {
6      private Date appointmentDate;
7
8      public Appointment(Date appointmentDate) {
9          this.appointmentDate = appointmentDate;
10     }
11
12     public Date getAppointmentDate() {
13         return appointmentDate;
14     }
15 }
```

7. Customer.java

```
J Customer.java X
source_code > J Customer.java > ...
1 //SARANYA A/P JAYARAMA REDDY (B22EC3013)
2
3 public class Customer {
4     private String name;
5     private String licenseNumber;
6     private String address;
7     private String email;
8     private String phoneNumber;
9
10    public Customer(String name, String licenseNumber, String address, String email, String phoneNumber) {
11        this.name = name;
12        this.licenseNumber = licenseNumber;
13        this.address = address;
14        this.email = email;
15        this.phoneNumber = phoneNumber;
16    }
17
18    //Getter method
19    public String getName(){
20        return name;
21    }
22
23    public String getLicenseNumber(){
24        return licenseNumber;
25    }
26
27    public String getAddress(){
28        return address;
29    }
30
31    public String getEmail(){
32        return email;
33    }
34
35    public String getPhoneNumber(){
36        return phoneNumber;
37    }
38 }
```

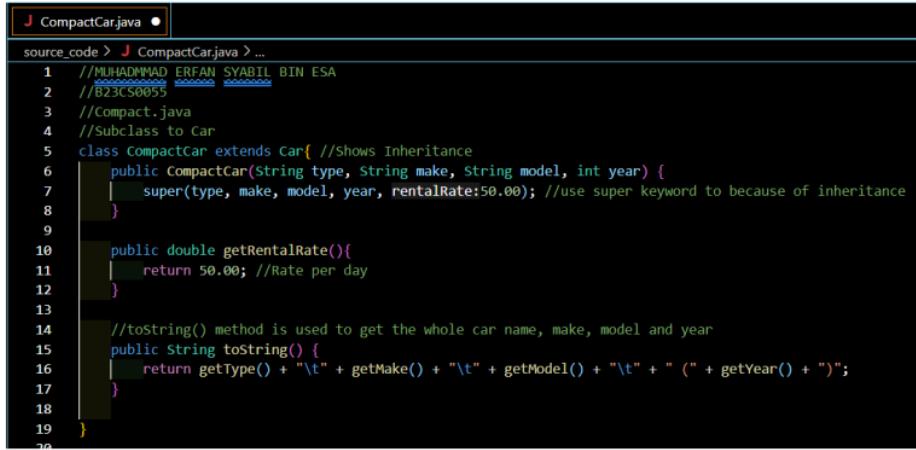
8. Car.java



The screenshot shows a Java code editor with the file 'Car.java' open. The code is a Java class definition for a car, implementing the 'Rentable' interface. It includes fields for type, make, model, year, and rental rate, along with getter methods for each. The code is color-coded for syntax highlighting.

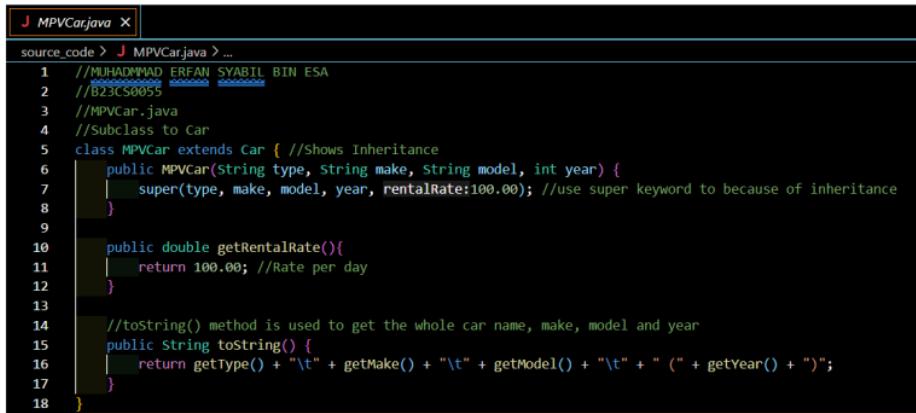
```
source_code > J Car.java > ...
1  //MUHAMMAD ERFAN SYABIL BIN ESA
2  //B23CS0055
3  // Car.java
4  //Superclass
5  public abstract class Car implements Rentable {
6      protected String type;
7      protected String make;
8      protected String model;
9      protected int year;
10     protected double rentalRate;
11
12     public Car(String type, String make, String model, int year, double rentalRate) {
13         this.type = type;
14         this.make = make;
15         this.model = model;
16         this.year = year;
17         this.rentalRate = rentalRate;
18     }
19
20     //Getter method
21     public String getType(){
22         return type;
23     }
24
25     public String getMake(){
26         return make;
27     }
28
29     public String getModel(){
30         return model;
31     }
32
33     public int getYear(){
34         return year;
35     }
36
37     public abstract double getRentalRate();
38 }
39
```

9. CompactCar.java



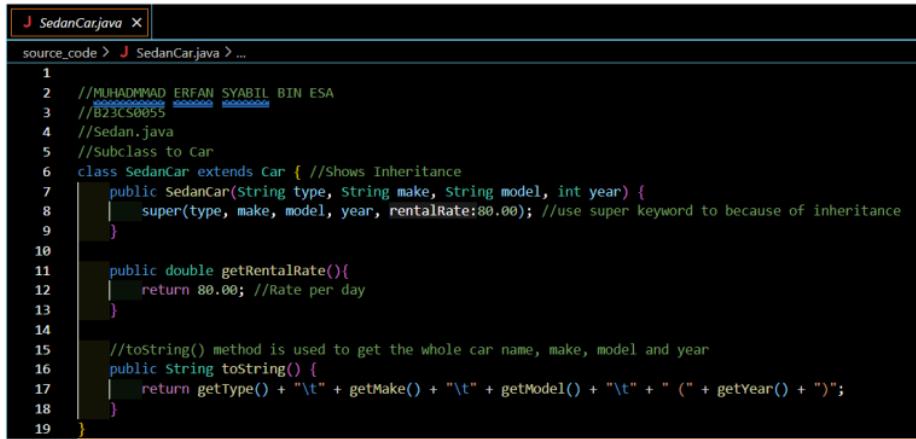
```
J CompactCar.java •
source_code > J CompactCar.java > ...
1 //MUHAMMAD ERFAN SYABIL BIN ESA
2 //B23CS0055
3 //Compact.java
4 //Subclass to Car
5 class CompactCar extends Car{ //Shows Inheritance
6     public CompactCar(String type, String make, String model, int year) {
7         super(type, make, model, year, rentalRate:50.00); //use super keyword to because of inheritance
8     }
9
10    public double getRentalRate(){
11        return 50.00; //Rate per day
12    }
13
14    //toString() method is used to get the whole car name, make, model and year
15    public String toString() {
16        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + "(" + getYear() + ")";
17    }
18
19 }
```

10. MPVCar.java



```
J MPVCar.java X
source_code > J MPVCar.java > ...
1 //MUHAMMAD ERFAN SYABIL BIN ESA
2 //B23CS0055
3 //MPVCar.java
4 //Subclass to Car
5 class MPVCar extends Car { //Shows Inheritance
6     public MPVCar(String type, String make, String model, int year) {
7         super(type, make, model, year, rentalRate:100.00); //use super keyword to because of inheritance
8     }
9
10    public double getRentalRate(){
11        return 100.00; //Rate per day
12    }
13
14    //toString() method is used to get the whole car name, make, model and year
15    public String toString() {
16        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + "(" + getYear() + ")";
17    }
18 }
```

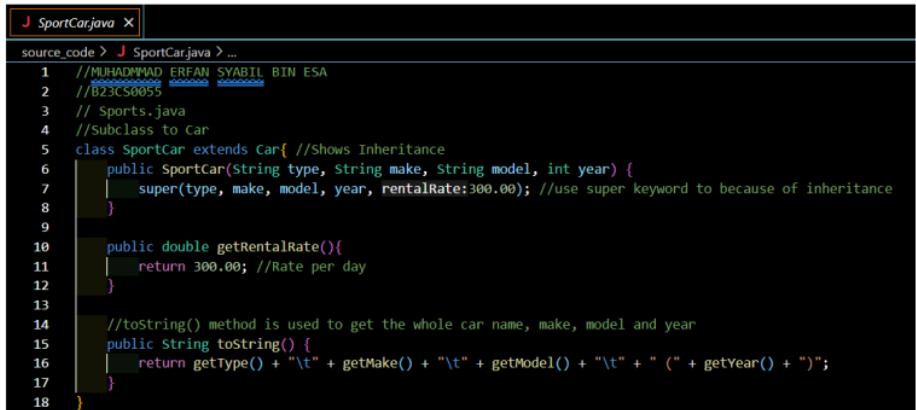
11. SedanCar.java



The screenshot shows a Java code editor window titled "J SedanCar.java". The code is as follows:

```
1 //MUHAMMAD ERFAN SYABIL BIN ESA
2 //B23CS0055
3 //Sedan.java
4 //Subclass to Car
5 class SedanCar extends Car { //Shows Inheritance
6     public SedanCar(String type, String make, String model, int year) {
7         super(type, make, model, year, rentalRate:80.00); //use super keyword to because of inheritance
8     }
9
10    public double getRentalRate(){
11        return 80.00; //Rate per day
12    }
13
14
15    //toString() method is used to get the whole car name, make, model and year
16    public String toString() {
17        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + "(" + getYear() + ")";
18    }
19 }
```

12. SportCar.java



The screenshot shows a Java code editor window titled "J SportCar.java". The code is as follows:

```
1 //MUHAMMAD ERFAN SYABIL BIN ESA
2 //B23CS0055
3 // Sports.java
4 //Subclass to Car
5 class SportCar extends Car{ //Shows Inheritance
6     public SportCar(String type, String make, String model, int year) {
7         super(type, make, model, year, rentalRate:300.00); //use super keyword to because of inheritance
8     }
9
10    public double getRentalRate(){
11        return 300.00; //Rate per day
12    }
13
14    //toString() method is used to get the whole car name, make, model and year
15    public String toString() {
16        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + "(" + getYear() + ")";
17    }
18 }
```

USER MANUAL:

Main menu:

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6):
```

When the user first runs the program, it is greeted with the simple main menu. It has 6 choices and the purposes of those choices are:

1. User can view the available cars for rent and their daily rental prices.
2. The user can add a customer detail such as their name, license number, address, email, and phone number.
3. The user can view all the multiple customer details.
4. This is the essential part of the system where the user can rent a vehicle.
5. User can track their rental history with this.
6. Exit the program.

User must enter one of the choices based on their needs.

Error (User entered none of the choices)

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 7
Invalid choice. Please enter a number between 1 and 6.

--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6):
```

4

If a user entered a choice number other than 1 to 6 it will display:

“Invalid choice. Please enter a number between 1 to 6”

It will loop back to the main menu.

Display available car:

```
Enter your choice (1-6): 1

--- Available Car and Rental Prices ---

      Type      Make   Model     Year  Rental Price
1. Compact    Perodua Axia    (2022) RM50.0 per day
2. Sports     Ford    Mustang  (2022) RM300.0 per day
3. Sedan      Proton  Saga    (2022) RM80.0 per day
4. MPV        Honda   BRV    (2022) RM100.0 per day

--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): █
```

If the user enters 1 as one of their choices, it will display a table of all the available cars based on type, make, model, year and their rental price per day.

Add customer details:

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 2

Enter customer name: Amirah
Enter customer license number: 020813060296
Enter customer address: Dungun, Terengganu
Enter customer email: nuramierah@gmail.com
Enter customer phone number: 0179213496

Customer details added:
Name: Amirah
License Number: 020813060296
Address: Dungun, Terengganu
Email: nuramierah@gmail.com
Phone No: 0179213496
```

If the user enters 2 as one of their choices, it will allow the user to enter customer details. Start with the customer name, license number, address, email and phone number.

After that, it will print back the output of the information that the user entered.

Display all of the customers information

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 3

--- All Customer Information ---
-----
Name      License Number          Address                Email           Phone No
-----
1. Erfan Syabil 021026060451    Puchong, Selangor     muhderfan2610@gmail.com 0198525011
-----
2. Amirul Hani 020814112378    Gombak, Kuala Lumpur   amirulhani02@gmail.com 0173223121
-----
3. Saranya Red 010515113278    Ipoh, Perak         saranyaJayarama@yahoo.com.my 0178324320
-----
4. Iswary Aish 010715442781    Kuantan, Pahang       aish@email.com.my        0132454430
-----
5. Amirah     020813060296    Dungun, Terengganu   nuramierah@gmail.com 0179213496
```

If the user enters 3 as one of their choices, it will print out a list of customers, allowing the user to view all customer details.

Rent a car

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 4
```

```
--- All Customer Information ---
```

Name	License Number	Address	Email	Phone No
1. Erfan Syabil	021026060451	Puchong, Selangor	muhderfan2610@gmail.com	0198525011
2. Amirul Hani	020814112378	Gombak, Kuala Lumpur	amirulhani02@gmail.com	0173223121
3. Saranya Red	010515113278	Ipoh, Perak	saranyaJayarama@yahoo.com.my	0178324320
4. Iswary Aish	010715442781	Kuantan, Pahang	aish@email.com.my	0132454430
5. Amirah	020813060296	Dungun, Terengganu	nuramierah@gmail.com	0179213496

```
Enter the index of the customer to make a rental for: 2
```

```
--- Pickup Locations ---
1. 123 BP 11, Puchong, Selangor, 47120
2. 456 NP 32, Kulai, Johor, 81310
3. 789 Villa, Kuantan, Pahang, 43543
4. 102 Elm St, Gombak, Kuala Lumpur, 91232
5. 879 Oak St, Ipoh, Perak, 30000
6. 321 Pine St, Jeli, Kelantan, 54321
Enter the index of the pickup location: 4
```

```
--- Available Car and Rental Prices ---  


| Type       | Make    | Model   | Year   | Rental Price    |
|------------|---------|---------|--------|-----------------|
| 1. Compact | Perodua | Axia    | (2022) | RM50.0 per day  |
| 2. Sports  | Ford    | Mustang | (2022) | RM300.0 per day |
| 3. Sedan   | Proton  | Saga    | (2022) | RM80.0 per day  |
| 4. MPV     | Honda   | BRV     | (2022) | RM100.0 per day |

  
Enter the car index to rent (1-4): 3  
  
--- Return Locations ---  


1. 123 BP 11, Puchong, Selangor, 47120
2. 456 NP 32, Kulai, Johor, 81310
3. 879 Villa, Kuantan, Pahang, 43543
4. 102 Elm St, Gombak, Kuala Lumpur, 91232
5. 879 Oak St, Ipoh, Perak, 30000
6. 321 Pine St, Jeli, Kelantan, 54321

  
Enter the index of the return location: 4  
Enter appointment date (yyyy-MM-dd HH:mm:ss): 2024-02-01 16:00:00  
Enter the duration of rental in days: 3  
  
-----  
  
Rental Successful for: Amirul Hani  
  
Name: Amirul Hani  
License Number: 020814112378  
Address: Gombak, Kuala Lumpur  
Email: amirulhani02@gmail.com  
Phone Number: 0173223121  
  
Car rented: Sedan      Proton Saga      (2022)  
Pickup date: Thu Feb 01 16:00:00 MYT 2024  
Pickup location: 102 Elm St, Gombak, Kuala Lumpur, 91232  
Return location: 102 Elm St, Gombak, Kuala Lumpur, 91232  
Duration: 3 day(s)  
  
Total: RM240.00  
-----
```

Choice 4 allows users to rent a car. First of all, users must choose a customer whom they want to rent a car by inputting the index of the customer list. After that, they must choose a pickup location by entering a number based on the list shown. Next, choose a car, user must enter their preferred choices by the number of the available car list. A return location must be chosen. User must enter their appointment date and time to pick up their car. Lastly, enter a number for their rental duration.

Output will be shown, informing the user their rental is successful and printing out their rental information along with the total rental cost.

View rental history

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 5

--- Rental History ---

Name: Amirul Hani
License Number: 020814112378
Address: Gombak, Kuala Lumpur
Email: amirulhani02@gmail.com
Phone Number: 0173223121

Car rented: Sedan      Proton Saga      (2022)
Pickup date: Thu Feb 01 16:00:00 MYT 2024
Pickup location: 102 Elm St, Gombak, Kuala Lumpur, 91232
Return location: 102 Elm St, Gombak, Kuala Lumpur, 91232
Duration: 3 day(s)

Total: RM240.00
-----
```

User can see their rental history by pressing 5. Output will be shown containing the customer information and their rental information.

Exit

```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-6): 6
Exiting Rental System. Thank you!
```

If user wishes to exit the program, press 6.

GROUP 5 - OOP Project Report RESUBMIT

ORIGINALITY REPORT

4%
SIMILARITY INDEX

2%
INTERNET SOURCES

0%
PUBLICATIONS

4%
STUDENT PAPERS

PRIMARY SOURCES

1	www.mapquestapi.com Internet Source	2%
2	Submitted to Asia Pacific International College Student Paper	1 %
3	Submitted to Regent Independent School and Sixth Form College Student Paper	1 %
4	Submitted to Informatics Education Limited Student Paper	<1 %
5	Submitted to Info Myanmar College Student Paper	<1 %
6	Submitted to Suffolk County Community College Student Paper	<1 %

Exclude quotes

Off

Exclude bibliography

Off

Exclude matches

Off