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

The work in this page has been done by: **Saranya A/P Jayarama Reddy**, **Amirul Hani Bin Syafrizon** *Date:* **15/01/2024**

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**

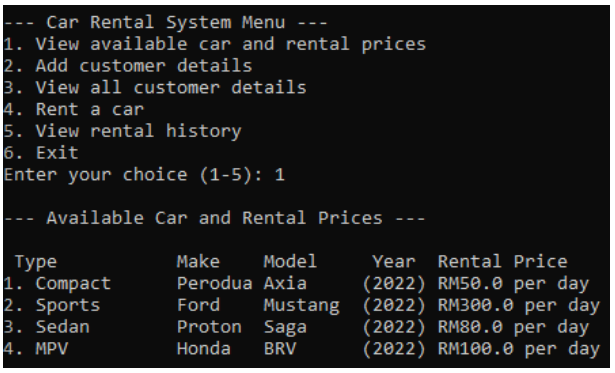
   Users can choose one of the available cars by providing both a pick-up location and a return location. User must enter a date and time to pick up their car. Once all of the necessary information has been completed, the confirmation details will be displayed and the user can see their rental history.

3) **Pickup Location and Return Location**

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

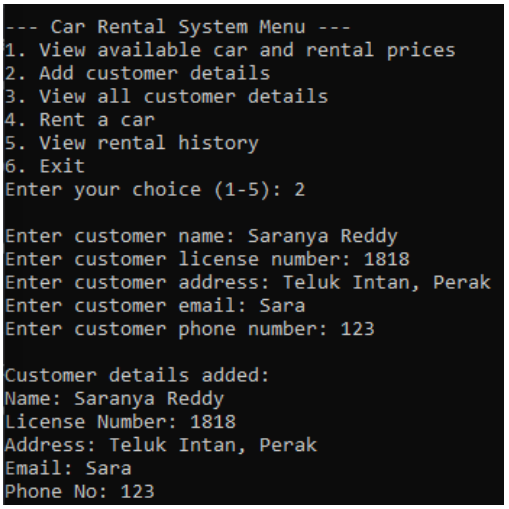
How to Use the System:

1) **First: View Available Car and Rental Prices**



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**



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.

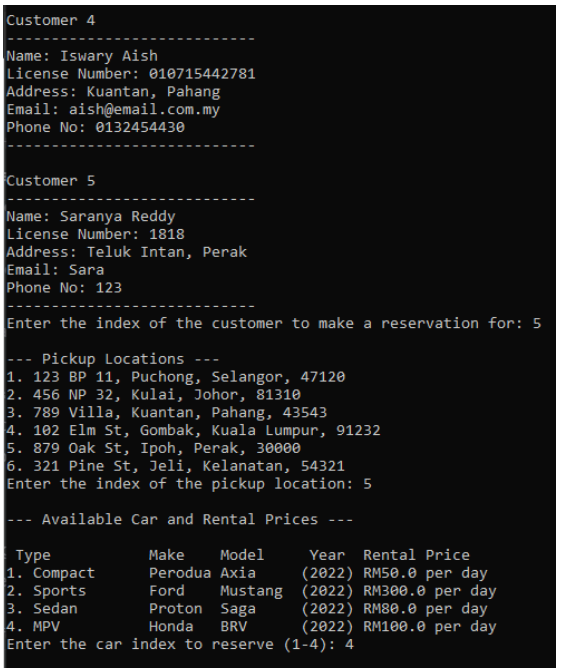
3) Third: View All Customer Details

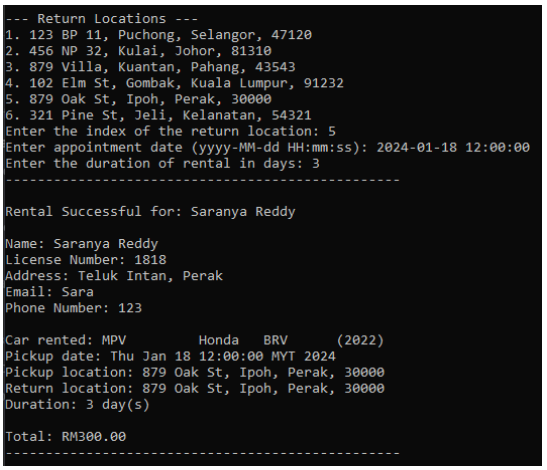
```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 3

--- 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

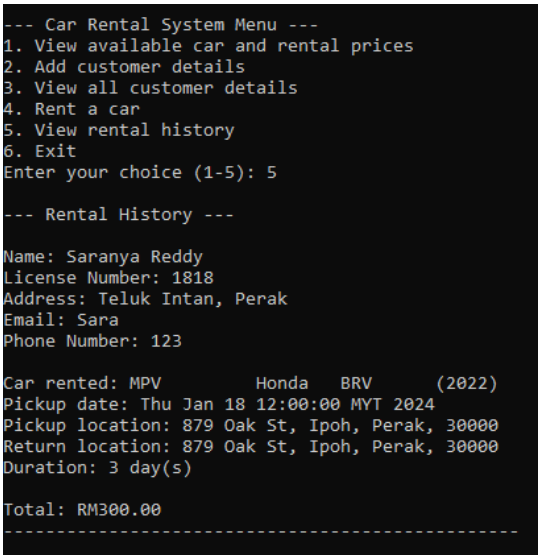
```
--- Car Rental System Menu ---
1. View available car and rental prices
2. Add customer details
3. View all customer details
4. Rent a car
5. View rental history
6. Exit
Enter your choice (1-5): 4

--- 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
----------------------------
```





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



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):

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

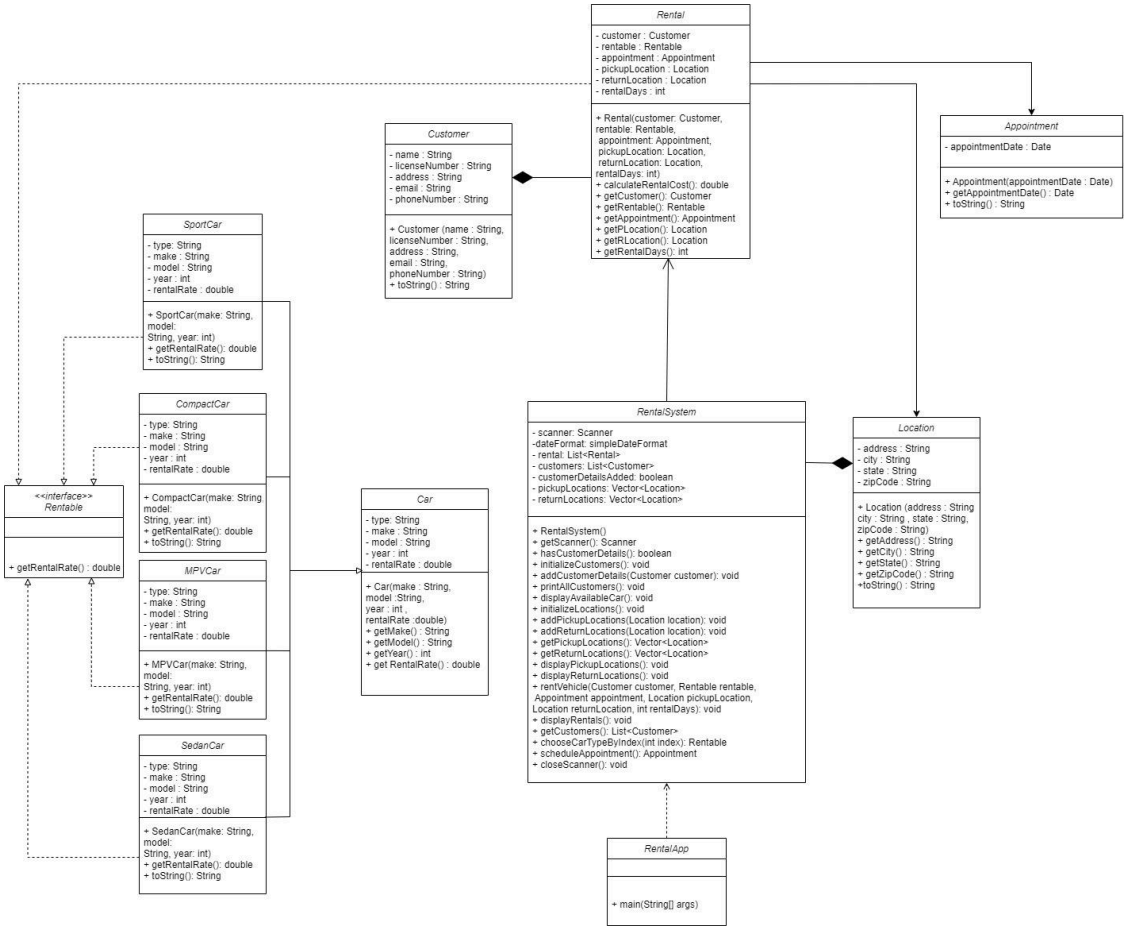
Interface:

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.

The work in this page has been done by: **Iswary A/P Shanmugam** *Date:* **15/01/2024**

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.

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

The work in this page has been done by : Iswary A/P Shanmugam *Date:* **15/01/2024**

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 toString() 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.

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

Class Rentable:

Attributes	**Description**
Method	**Description**
getRentalRate() : double	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**
type(String)	The type of the car
make (String)	The make or brand of the sports car.
model (String)	The model name of the sports car.
year (int)	The manufacturing year of the sports car.
Method	**Description**
getRentalRate() : double	Returns the rental rate specific to the sports car.
toString() : String	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.

The work in this page has been done by: **MUHAMMAD ERFAN SYABIL BIN ESA** *Date:* **16/01/2023**

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

RentalSystem.java

source_code > RentalSystem.java > ...

```java
//AMIRUL HANI BIN SYAFRIZON
//B23CS0025
// RentalSystem.java

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Scanner;
import java.util.Vector;
import java.text.*;

public class RentalSystem {
    //variable
    private Scanner scanner;
    private SimpleDateFormat dateFormat;
    private List<Rental> rental;
    private List<Customer> customers;
    private boolean customerDetailsAdded = false;
    private Vector<Location> pickupLocations;
    private Vector<Location> returnLocations;

    //constructor
    public RentalSystem() {
        scanner = new Scanner(System.in);
        customerDetailsAdded = false;
        customers = new ArrayList<>();  // Initialize the customers list
        rental = new ArrayList<>();
        dateFormat = new SimpleDateFormat(pattern:"yyyy-MM-dd HH:mm:ss");
        pickupLocations = new Vector<>();
        returnLocations = new Vector<>();
        initializeLocations(); // Initialize locations when the system is created
        initializeCustomers(); // Initialize customers when the system is created
    }

    //Scanner method for input
    public Scanner getScanner() {
        return scanner;
```

```
    }

    //Check whether is there customer detail
    public boolean hasCustomerDetails() {
        return customerDetailsAdded;
    }

    //Initialize customer
    public void initializeCustomers(){
        customers.add(new Customer(name:"Erfan Syabil", licenseNumber:"021026060451", address:"Puchong, Selangor", email:"muhderfan2610@gmail.com", phoneNumber:"0198525011"));
        customers.add(new Customer(name:"Amirul Hani", licenseNumber:"020814112378", address:"Gombak, Kuala Lumpur", email:"amirulhani02@gmail.com", phoneNumber:"0173223121"));
        customers.add(new Customer(name:"Saranya Red", licenseNumber:"010515113278", address:"Ipoh, Perak", email:"saranyaJayarama@yahoo.com.my", phoneNumber:"0178324320"));
        customers.add(new Customer(name:"Iswary Aish", licenseNumber:"010715442781", address:"Kuantan, Pahang", email:"aish@email.com.my", phoneNumber:"0132454430"));
        customerDetailsAdded = true;
    }

    //To add user in the Customer List
    public void addCustomerDetails(Customer customer) {
        customerDetailsAdded = true;
        customers.add(customer);
    }

    //Display all Customer in the list
    public void printAllCustomers() {
        System.out.println(x:"\n--- All Customer Information ---");
        System.out.println(x:"-------------------------------------------------------------------------------");
        System.out.println("\tName \t" + "License Number \t\t" + "Address \t\t" + "Email \t\t" + "Phone No");
        System.out.println(x:"-------------------------------------------------------------------------------");
        int i = 1;
        for (Customer customer : customers) {
            System.out.println(i + ". " + customer.getName() + "\t" + customer.getLicenseNumber() + "\t" + customer.getAddress() + "\t" + customer.getEmail()+ "\t" + customer.getPhoneNumber());
            System.out.println(x:"-------------------------------------------------------------------------------");
            i++;
        }
```

```
    }

    //Display all available Car by retrieving the list in chooseCarTypeByIndex() method
    public void displayAvailableCar() {
        System.out.println(x:"\n--- Available Car and Rental Prices ---");
        System.out.println("\n Type" + "\t\t" + "Make" + "\t" + "Model" + "\t" + "  Year" + "\t" + "Rental Price");
        for (int i = 1; i <= 4; i++) {
            Rentable car = chooseCarTypeByIndex(i);
            System.out.println(i + ". " + car.toString() + "\t" + "RM" + car.getRentalRate() + " per day");
        }
    }

    //Initialize pickup and return location
    private void initializeLocations() {
        //Pickup Location
        pickupLocations.add(new Location(address:"123 BP 11", city:"Puchong", state:"Selangor", zipCode:"47120"));
        pickupLocations.add(new Location(address:"456 NP 32", city:"Kulai", state:"Johor", zipCode:"81310"));
        pickupLocations.add(new Location(address:"789 Villa", city:"Kuantan", state:"Pahang", zipCode:"43543"));
        pickupLocations.add(new Location(address:"102 Elm St", city:"Gombak", state:"Kuala Lumpur", zipCode:"91232"));
        pickupLocations.add(new Location(address:"879 Oak St", city:"Ipoh", state:"Perak", zipCode:"30000"));
        pickupLocations.add(new Location(address:"321 Pine St", city:"Jeli", state:"Kelanatan", zipCode:"54321"));

        //Return Location
        returnLocations.add(new Location(address:"123 BP 11", city:"Puchong", state:"Selangor", zipCode:"47120"));
        returnLocations.add(new Location(address:"456 NP 32", city:"Kulai", state:"Johor", zipCode:"81310"));
        returnLocations.add(new Location(address:"879 Villa", city:"Kuantan", state:"Pahang", zipCode:"43543"));
        returnLocations.add(new Location(address:"102 Elm St", city:"Gombak", state:"Kuala Lumpur", zipCode:"91232"));
        returnLocations.add(new Location(address:"879 Oak St", city:"Ipoh", state:"Perak", zipCode:"30000"));
        returnLocations.add(new Location(address:"321 Pine St", city:"Jeli", state:"Kelanatan", zipCode:"54321"));
    }

    //Add pickup location into the vector location
    public void addPickupLocation(Location location) {
        pickupLocations.add(location);
```

```
    }

    //Add return location into the vector location
    public void addReturnLocation(Location location) {
        returnLocations.add(location);
    }

    //Retrieve pickup location object
    public Vector<Location> getPickupLocations() {
        return pickupLocations;
    }

    //Retrieve return location object
    public Vector<Location> getReturnLocations() {
        return returnLocations;
    }

    //Display pickup location
    public void displayPickupLocations() {
        System.out.println(x:"\n--- Pickup Locations ---");
        int i = 1;
        for (Location pickupLocation : pickupLocations) {
            System.out.println(i + ". " + pickupLocation.getAddress() +  ", " + pickupLocation.getCity() + ", " + pickupLocation.getState() + ", " + pickupLocation.getZipCode());
            i++;
        }
    }

    //Display return location
    public void displayReturnLocations() {
        System.out.println(x:"\n--- Return Locations ---");
```

```
        int i = 1;
        for (Location returnLocation : returnLocations) {
            System.out.println(i + ". " + returnLocation.getAddress() +  ", " + returnLocation.getCity() + ", " + returnLocation.getState() + ", " + returnLocation.getZipCode());
            i++;
        }
    }


    //Rent a vehicle and after that produce output containing info about customer, car rented, pickup time, pickup and return location, duration and total.
    public void rentVehicle(Customer customer, Rentable rentable, Appointment appointment, Location pickupLocation, Location returnLocation, int rentalDays) {
        Rental rental = new Rental(customer, rentable, appointment, pickupLocation, returnLocation, rentalDays);
        this.rental.add(rental);

        System.out.println(x:"-----------------------------------------------");
        System.out.println("\nRental Successful for: " + customer.getName());

        System.out.println("\nName: " + customer.getName());
        System.out.println("License Number: " + customer.getLicenseNumber());
        System.out.println("Address: " + customer.getAddress());
        System.out.println("Email: " + customer.getEmail());
        System.out.println("Phone Number: " + customer.getPhoneNumber());

        System.out.println("\nCar rented: " + rentable);
        System.out.println("Pickup date: " + rental.getAppointment().getAppointmentDate());
        System.out.println("Pickup location: " + rental.getPLocation().toString());
        System.out.println("Return location: " + rental.getPLocation().toString());
        System.out.println("Duration: " + rental.getRentalDays() + " day(s)");
        System.out.printf(format:"\nTotal: RM%.2f", rental.calculateRentalCost());
        System.out.println(x:"\n-----------------------------------------------");
    }

    //To display rental history
    public void displayRentals() {
        System.out.println(x:"\n--- Rental History ---");
        for (Rental rental : rental) {
            System.out.println("\nName: " + rental.getCustomer().getName());
            System.out.println("License Number: " + rental.getCustomer().getLicenseNumber());
            System.out.println("Address: " + rental.getCustomer().getAddress());
            System.out.println("Email: " + rental.getCustomer().getEmail());
            System.out.println("Phone Number: " + rental.getCustomer().getPhoneNumber());

            System.out.println("\nCar rented: " + rental.getRentable());
            System.out.println("Pickup date: " + rental.getAppointment().getAppointmentDate());
            System.out.println("Pickup location: " + rental.getPLocation().toString());
            System.out.println("Return location: " + rental.getPLocation().toString());
```

```
            System.out.println("Duration: " + rental.getRentalDays() + " day(s)");
            System.out.printf(format:"\nTotal: RM%.2f", rental.calculateRentalCost());
            System.out.println(x:"\n-------------------------------------------------");
        }
    }

    //Close scanner to prevent resource leak
    public void closeScanner() {
        scanner.close();
    }

    //Initialize car into arrayList and return the list index
    public Rentable chooseCarTypeByIndex(int carIndex) {
        // Choose a vehicle based on index
        List<Rentable> availableCar = new ArrayList<>();
        availableCar.add(new CompactCar(type:"Compact",make:"Perodua", model:"Axia", year:2022));
        availableCar.add(new SportCar(type:"Sports", make:"Ford", model:"Mustang", year:2022));
        availableCar.add(new SedanCar(type:"Sedan", make:"Proton", model:"Saga", year:2022));
        availableCar.add(new MPVCar(type:"MPV\t", make:"Honda", model:"BRV", year:2022));

        if (carIndex >= 1 && carIndex <= availableCar.size()) {
            return availableCar.get(carIndex - 1);
        } else {
            throw new IllegalArgumentException(s:"Invalid vehicle index.");
        }
    }

    //Set appointment using appropriate format of date and time
    public Appointment scheduleAppointment() {
        System.out.print(s:"Enter appointment date (yyyy-MM-dd HH:mm:ss): ");
        String dateString = scanner.nextLine();

        try {
            Date appointmentDate = dateFormat.parse(dateString);
            return new Appointment(appointmentDate);
        } catch (ParseException e) {
            System.out.println(x:"Invalid date format. Defaulting to current date and time.");
            return new Appointment(new Date());
        }
    }

    //Retrieve customer object.
    public List<Customer> getCustomers() {
        return customers;
    }
}
```

2. RentalApp.java

RentalApp.java

source_code > RentalApp.java > ...

```java
//AMIRUL HANI BIN SYAFRIZON
//B23CS0025
// RentalApp.java

public class RentalApp {
    Run | Debug
    public static void main(String[] args) {
        RentalSystem rentalSystem = new RentalSystem();

        try {
            int choice;
            Customer customer = null;
            do {
                System.out.println(x:"\n--- Car Rental System Menu ---");
                System.out.println(x:"1. View available car and rental prices");
                System.out.println(x:"2. Add customer details");
                System.out.println(x:"3. View all customer details");
                System.out.println(x:"4. Rent a car");
                System.out.println(x:"5. View rental history");
                System.out.println(x:"6. Exit");
                System.out.print(s:"Enter your choice (1-5): ");

                choice = rentalSystem.getScanner().nextInt();
                rentalSystem.getScanner().nextLine();

                switch (choice) {
                    case 1:
                        // View available vehicles and rental prices
                        rentalSystem.displayAvailableCar();
                        break;
                    case 2:
                        // Add user details
                        System.out.print(s:"\nEnter customer name: ");
                        String name = rentalSystem.getScanner().nextLine();

                        System.out.print(s:"Enter customer license number: ");
                        String licenseNumber = rentalSystem.getScanner().nextLine();
```

RentalApp.java

source_code > RentalApp.java > ...

```
74                          rentalSystem.displayPickupLocations();
75                          System.out.print(s:"Enter the index of the pickup location: ");
76                          int pickupIndex = rentalSystem.getScanner().nextInt();
77                          rentalSystem.getScanner().nextLine();
78
79                          Location pickupLocation = rentalSystem.getPickupLocations().get(pickupIndex - 1);
80
81
82                          //Choose car to rent
83                          rentalSystem.displayAvailableCar();
84                          System.out.print(s:"Enter the car index to reserve (1-4): ");
85                          int carIndex = rentalSystem.getScanner().nextInt();
86                          rentalSystem.getScanner().nextLine();
87                          Rentable selectedCar = rentalSystem.chooseCarTypeByIndex(carIndex);
88
89                          // Choose return location
90                          rentalSystem.displayReturnLocations();
91                          System.out.print(s:"Enter the index of the return location: ");
92                          int returnIndex = rentalSystem.getScanner().nextInt();
93                          rentalSystem.getScanner().nextLine();
94
95                          Location returnLocation = rentalSystem.getReturnLocations().get(returnIndex - 1);
96
97                          Appointment appointment = rentalSystem.scheduleAppointment();
98
99                          System.out.print(s:"Enter the duration of rental in days: ");
100                         int rentalDays = rentalSystem.getScanner().nextInt();
101                         rentalSystem.getScanner().nextLine();
102
103                         rentalSystem.rentVehicle(customer, selectedCar, appointment, pickupLocation, returnLocation, rentalDays);
104                     } else {
105                         System.out.println(x:"Invalid customer index.");
106                     }
107                 } else {
108                     System.out.println(x:"Please add user details first.");
109                 }
110                 break;
```

RentalApp.java

source_code > RentalApp.java > ...

```
111                 case 5:
112                     // View rental history
113                     rentalSystem.displayRentals();
114                     break;
115                 case 6:
116                     System.out.println(x:"Exiting Rental System. Thank you!");
117                     break;
118                 default:
119                     System.out.println(x:"Invalid choice. Please enter a number between 1 and 5.");
120             }
121         } while (choice != 6);
122
123     } catch (Exception e) {
124         System.out.println("An error occurred: " + e.getMessage());
125     } finally {
126         // Close the scanner
127         rentalSystem.closeScanner();
128     }
129 }
130 }
131
```

3. Rental.java

```java
//SARANYA A/P JAYARAMA REDDY (B22EC3013)

public class Rental {
    private Customer customer;
    private Rentable rentable;
    private Appointment appointment;
    private Location pickupLocation;
    private Location returnLocation;
    private int rentalDays;

    public Rental(Customer customer, Rentable rentable, Appointment appointment, Location pickupLocation, Location returnLocation, int rentalDays){
        this.customer = customer;
        this.rentable = rentable;
        this.appointment = appointment;
        this.pickupLocation = pickupLocation;
        this.returnLocation = returnLocation;
        this.rentalDays = rentalDays;
    }

    //Method to calculate the total rental
    public double calculateRentalCost(){
        return rentable.getRentalRate() * rentalDays;
        }

    //Getter method
    public Customer getCustomer() {
        return customer;
        }

    public Rentable getRentable() {
        return rentable;
        }

    public Appointment getAppointment() {
        return appointment;
        }

    public Location getPLocation(){
        return pickupLocation;
        }

    public Location getRLocation(){
        return returnLocation;
        }

    public int getRentalDays(){
        return rentalDays;
        }
}
```

4. Rentable.java

Rentable.java

source_code > Rentable.java > Rentable

```java
//SARANYA A/P JAYARAMA REDDY (B22EC3013)

public interface Rentable {
    double getRentalRate();
}

```

5. Location.java

Location.java

source_code > Location.java > ...

```java
//ISWARY:B22EC3004

public class Location {
    private String address;
    private String city;
    private String state;
    private String zipCode;

    public Location(String address, String city, String state, String zipCode) {
        this.address = address;
        this.city = city;
        this.state = state;
        this.zipCode = zipCode;
    }

    // Getter methods
    public String getAddress() {
        return address;
    }

    public String getCity() {
        return city;
    }

    public String getState() {
        return state;
    }

    public String getZipCode() {
        return zipCode;
    }

    //toString method to get location as a String
    public String toString(){
        return getAddress() + ", " + getCity() + ", " + getState() + ", " + getZipCode();
    }
}
```

6. Appoinment.java

Appointment.java

source_code > Appointment.java > ...

```java
//Iswary:B22EC3004

import java.util.Date;

public class Appointment {
    private Date appointmentDate;

    public Appointment(Date appointmentDate) {
        this.appointmentDate = appointmentDate;
    }

    public Date getAppointmentDate() {
        return appointmentDate;
    }
}
```

7. Customer.java

Customer.java

source_code > Customer.java > ...

```java
//SARANYA A/P JAYARAMA REDDY (B22EC3013)

public class Customer {
    private String name;
    private String licenseNumber;
    private String address;
    private String email;
    private String phoneNumber;

    public Customer(String name, String licenseNumber, String address, String email, String phoneNumber) {
        this.name = name;
        this.licenseNumber = licenseNumber;
        this.address = address;
        this.email = email;
        this.phoneNumber = phoneNumber;
    }

    //Getter method
    public String getName(){
        return name;
    }

    public String getLicenseNumber(){
        return licenseNumber;
    }

    public String getAddress(){
        return address;
    }

    public String getEmail(){
        return email;
    }

    public String getPhoneNumber(){
        return phoneNumber;
    }
}
```

8. Car.java

Car.java

source_code > Car.java > ...

```java
//MUHADMMAD ERFAN SYABIL BIN ESA
//B23CS0055
// Car.java
//Superclass
public abstract class Car implements Rentable {
    protected String type;
    protected String make;
    protected String model;
    protected int year;
    protected double rentalRate;

    public Car(String type, String make, String model, int year, double rentalRate) {
        this.type = type;
        this.make = make;
        this.model = model;
        this.year = year;
        this.rentalRate = rentalRate;
    }

    //Getter method
    public String getType(){
        return type;
    }

    public String getMake(){
        return make;
    }

    public String getModel(){
        return model;
    }

    public int getYear(){
        return year;
    }

    public abstract double getRentalRate();
}

```

9. CompactCar.java

CompactCar.java

source_code > CompactCar.java > ...

```java
//MUHADMMAD ERFAN SYABIL BIN ESA
//B23CS0055
//Compact.java
//Subclass to Car
class CompactCar extends Car{ //Shows Inheritance
    public CompactCar(String type, String make, String model, int year) {
        super(type, make, model, year, rentalRate:50.00); //use super keyword to because of inheritance
    }

    public double getRentalRate(){
        return 50.00; //Rate per day
    }

    //toString() method is used to get the whole car name, make, model and year
    public String toString() {
        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + " (" + getYear() + ")";
    }

}
```

10. MPVCar.java

MPVCar.java

source_code > MPVCar.java > ...

```java
//MUHADMMAD ERFAN SYABIL BIN ESA
//B23CS0055
//MPVCar.java
//Subclass to Car
class MPVCar extends Car { //Shows Inheritance
    public MPVCar(String type, String make, String model, int year) {
        super(type, make, model, year, rentalRate:100.00); //use super keyword to because of inheritance
    }

    public double getRentalRate(){
        return 100.00; //Rate per day
    }

    //toString() method is used to get the whole car name, make, model and year
    public String toString() {
        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + " (" + getYear() + ")";
    }
}
```

11. SedanCar.java

SedanCar.java

source_code > SedanCar.java > ...

```java

//MUHADMMAD ERFAN SYABIL BIN ESA
//B23CS0055
//Sedan.java
//Subclass to Car
class SedanCar extends Car { //Shows Inheritance
    public SedanCar(String type, String make, String model, int year) {
        super(type, make, model, year, rentalRate:80.00); //use super keyword to because of inheritance
    }

    public double getRentalRate(){
        return 80.00; //Rate per day
    }

    //toString() method is used to get the whole car name, make, model and year
    public String toString() {
        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + " (" + getYear() + ")";
    }
}
```

12. SportCar.java

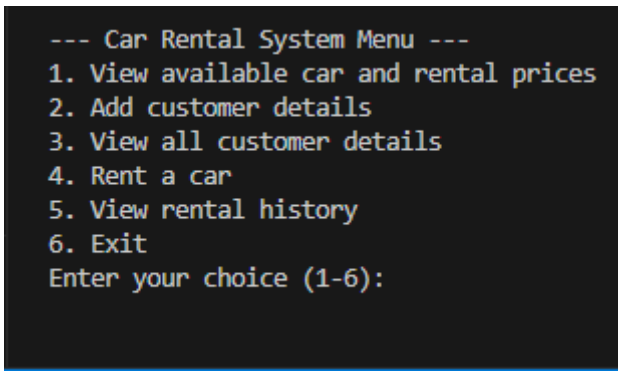
SportCar.java

source_code > SportCar.java > ...

```java
//MUHADMMAD ERFAN SYABIL BIN ESA
//B23CS0055
// Sports.java
//Subclass to Car
class SportCar extends Car{ //Shows Inheritance
    public SportCar(String type, String make, String model, int year) {
        super(type, make, model, year, rentalRate:300.00); //use super keyword to because of inheritance
    }

    public double getRentalRate(){
        return 300.00; //Rate per day
    }

    //toString() method is used to get the whole car name, make, model and year
    public String toString() {
        return getType() + "\t" + getMake() + "\t" + getModel() + "\t" + " (" + getYear() + ")";
    }
}
```

USER MANUAL:

Main menu:

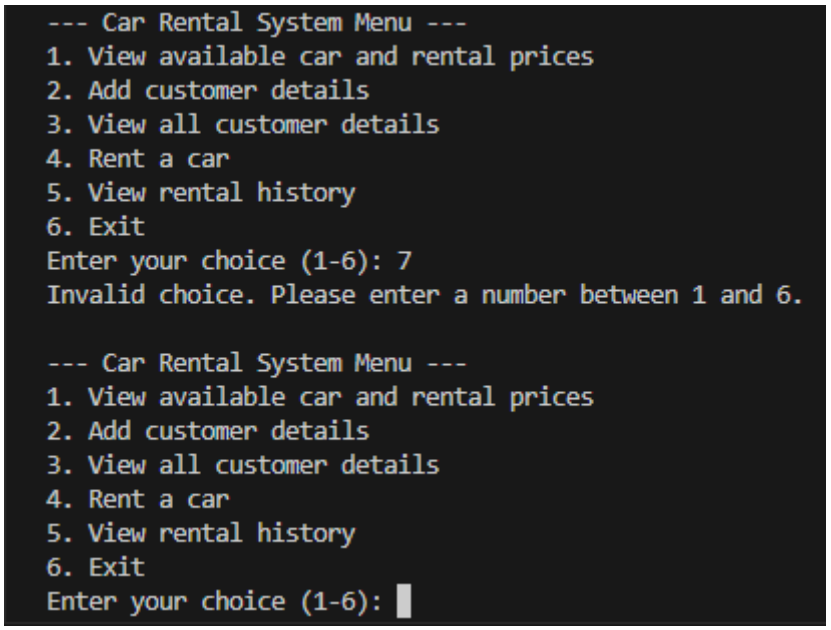


When the user first runs the program, it is greeted with the simple main menu.
It has 6 choices and the purposes of those choices are:

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)

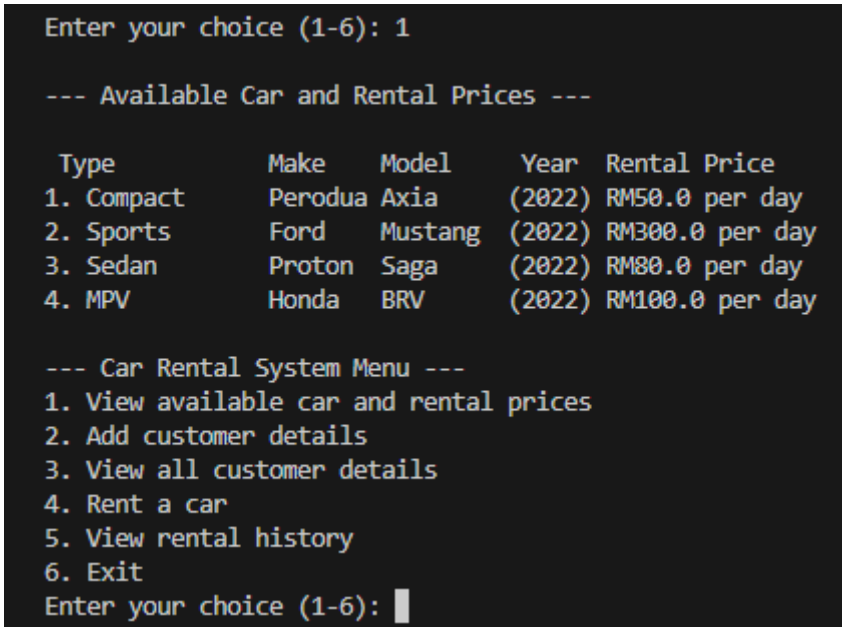


If a user entered a choice number other than 1 to 6 it will display:

"**Invalid choice. Please enter a number between 1 to 6**"

It will loop back to the main menu.

Display available car:



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, Kelanatan, 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, Kelanatan, 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.

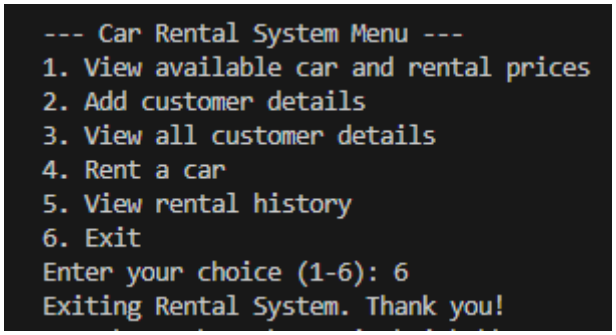
Output will be shown, informing the user their rental is successful and printing out their rental information along with the total rental cost.

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



If user wishes to exit the program, press 6.