Car Rental System Documentation

1. Introduction

This program is a simple car rental management system built with Java. It shows how object-oriented programming (OOP) concepts like encapsulation, composition, and collections can be applied to model cars, customers, and rental transactions.

The system allows:

- Adding cars and customers
- Renting cars for a number of days
- Returning cars
- Managing all data in lists (`ArrayList`)

2. Class Design

Car

- Purpose: Represents a car available for rent
- Fields: `carId`, `brand`, `model`, `basePricePerDay`, `isAvailable`
- Methods:
  - `calculatePrice(int days)` → returns total rental cost
  - `rent()` → marks car as unavailable
  - `returnCar()` → marks car as available
  - Getters for car details

Customer

- Purpose: Represents a customer renting a car
- Fields: `customerId`, `name`
- Methods: getters for customer details

Rental

- Purpose: Represents a rental transaction linking a Car and a Customer
- Fields: `car`, `customer`, `days`

- Methods: getters for rental details

CarRentalSystem

- Purpose: Manages cars, customers, and rentals
- Fields: `cars` list, `customers` list, `rentals` list
- Methods:
  - `addCar(Car car)` → adds a car to the system
  - `addCustomer(Customer customer)` → registers a customer
  - `rentCar(Car car, Customer customer, int days)` → processes rental if car is available
  - `returnCar(Car car)` → handles car return and removes rental record
  - `menu()` → interactive console menu for renting and returning cars

Main

- Purpose: Entry point of the program
- Flow:
  - Creates a `CarRentalSystem` object
  - Adds sample cars (`Toyota`, `Honda`, `Mahindra`)
  - Starts the interactive menu

3. Program Flow

1. Create the car rental system
2. Add cars to the system
3. Show menu options:
   a. Rent a car (enter customer name, car ID, rental days)
   b. Return a car (enter car ID)
   c. Exit the system
4. Process rentals and returns, updating availability and rental records
5. Display rental information and confirmation messages

4. Example Output

```
===== Car Rental System =====
1. Rent a Car
2. Return a Car
3. Exit
```

```
Enter your choice: 1

== Rent a Car ==

Enter your name: Alice

Available Cars:
C001 - Toyota Camry
C002 - Honda Accord
C003 - Mahindra Thar

Enter the car ID you want to rent: C002
Enter the number of days for rental: 3

== Rental Information ==

Customer ID: CUS1
Customer Name: Alice
Car: Honda Accord
Rental Days: 3
Total Price: $210.00

Confirm rental (Y/N): Y

Car rented successfully.
```

5. Key Takeaways

- Encapsulation: Car and Customer details are private, accessed via getters
- Composition: Rental objects link together Car and Customer instances
- Collections: ArrayList makes it easy to manage groups of cars, customers, and rentals
- Interactive Flow: The menu method shows how user input drives program logic
- Reusability: Classes are modular and can be extended (e.g., adding more car types or customer features)