Car Rental System

Explore the world of car rentals with the Car Rental System, a Java-based console application that combines learning and simulation. 🌟

Features

- 🚀 **Rent a Car**: Experience the ease of renting cars through an interactive console.
- 🔄 **Return a Car**: Effortlessly return previously rented cars and update availability.
- 👥 **Customer Management**: Add new customers and maintain customer records.
- 🚗 **Car Management**: Manage cars, brands, models, and pricing details.
- 📝 **Rental History**: Keep track of rentals, customers, and rental durations.

Getting Started

Prerequisites

- Java Development Kit (JDK) 8 or above
- IDE or text editor (e.g., IntelliJ IDEA, Eclipse, or Visual Studio Code)
- Git (optional, for cloning the repository)

Clone the Repository

```
git clone https://github.com/prabhatthakuryt/Car-Rental-System.git
```

Compile and Run

1. Navigate to the project directory.
2. Compile the Java files:

   ```
   javac -d . Main.java
   ```

3. Run the application:

   ```
   java carrentalmanagementsystem.Main
   ```

Code Overview

Main Components

1. Car Class

Represents a car with attributes like ID, brand, model, price per day, and availability.

```
class Car {
    private String carId;
    private String brand;
    private String model;
    private double basePricePerDay;
    private boolean isAvailable;

    public Car(String carId, String brand, String model, double
basePricePerDay) {
        this.carId = carId;
        this.brand = brand;
        this.model = model;
        this.basePricePerDay = basePricePerDay;
        this.isAvailable = true;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void rent() {
        isAvailable = false;
    }

    public void returnCar() {
        isAvailable = true;
    }

    public double calculatePrice(int rentalDays) {
        return basePricePerDay * rentalDays;
    }
}
```

2. Customer Class

Handles customer details like ID and name.

```
class Customer {
    private String customerId;
    private String name;

    public Customer(String customerId, String name) {
        this.customerId = customerId;
        this.name = name;
    }
}
```

3. Rental Class

Manages rental information, including the car, customer, and rental duration.

```
class Rental {
    private Car car;
    private Customer customer;
    private int days;

    public Rental(Car car, Customer customer, int days) {
        this.car = car;
        this.customer = customer;
        this.days = days;
    }
}
```

4. CarRentalSystem Class

Core functionality for renting and returning cars, managing customers, and displaying the menu.

```
class CarRentalSystem {
    private List<Car> cars = new ArrayList<>();
    private List<Customer> customers = new ArrayList<>();
    private List<Rental> rentals = new ArrayList<>();

    public void menu() {
        // Interactive console-based menu implementation
    }
}
```

5. Main Class

Initializes the system with sample cars and starts the menu.

```
public class Main {
    public static void main(String[] args) {
        CarRentalSystem rentalSystem = new CarRentalSystem();

        Car car1 = new Car("C001", "Toyota", "Camry", 60.0);
        Car car2 = new Car("C002", "Honda", "Accord", 70.0);
        Car car3 = new Car("C003", "Mahindra", "Thar", 150.0);

        rentalSystem.addCar(car1);
        rentalSystem.addCar(car2);
        rentalSystem.addCar(car3);

        rentalSystem.menu();
    }
}
```

Overall Code : Main.java

```
package carrentalmanagementsystem;


import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;


class Car {

    private String carId;

    private String brand;

    private String model;

    private double basePricePerDay;

    private boolean isAvailable;


    public Car(String carId, String brand, String model, double basePricePerDay) {

        this.carId = carId;

        this.brand = brand;

        this.model = model;

        this.basePricePerDay = basePricePerDay;

        this.isAvailable = true;

    }

    public String getCarId() {

        return carId;

    }


    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double calculatePrice(int rentalDays) {
        return basePricePerDay * rentalDays;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void rent() {
        isAvailable = false;
    }

    public void returnCar() {
        isAvailable = true;
    }
}

class Customer {
    private String customerId;
    private String name;
    public Customer(String customerId, String name) {

        this.customerId = customerId;

        this.name = name;

    }


    public String getCustomerId() {

        return customerId;

    }


    public String getName() {

        return name;

    }

}


class Rental {

    private Car car;

    private Customer customer;

    private int days;


    public Rental(Car car, Customer customer, int days) {

        this.car = car;

        this.customer = customer;

        this.days = days;

    }


    public Car getCar() {
        return car;
    }

    public Customer getCustomer() {
        return customer;
    }

    public int getDays() {
        return days;
    }
}

class CarRentalSystem {
    private List<Car> cars;
    private List<Customer> customers;
    private List<Rental> rentals;

    public CarRentalSystem() {
        cars = new ArrayList<>();
        customers = new ArrayList<>();
        rentals = new ArrayList<>();
    }

    public void addCar(Car car) {
        cars.add(car);
    }
```

```
public void addCustomer(Customer customer) {

    customers.add(customer);

}


public void rentCar(Car car, Customer customer, int days) {

    if (car.isAvailable()) {

        car.rent();

        rentals.add(new Rental(car, customer, days));


    } else {

        System.out.println("Car is not available for rent.");

    }

}


public void returnCar(Car car) {

    car.returnCar();

    Rental rentalToRemove = null;

    for (Rental rental : rentals) {

        if (rental.getCar() == car) {

            rentalToRemove = rental;

            break;

        }

    }

    if (rentalToRemove != null) {

        rentals.remove(rentalToRemove);


    } else {
            System.out.println("Car was not rented.");
        }
    }

    public void menu() {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("===== Car Rental System =====");
            System.out.println("1. Rent a Car");
            System.out.println("2. Return a Car");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            if (choice == 1) {
                System.out.println("\n== Rent a Car ==\n");
                System.out.print("Enter your name: ");
                String customerName = scanner.nextLine();

                System.out.println("\nAvailable Cars:");
                for (Car car : cars) {
                    if (car.isAvailable()) {
                        System.out.println(car.getCarId() + " - " + car.getBrand() + " " + car.getModel());
                    }
```

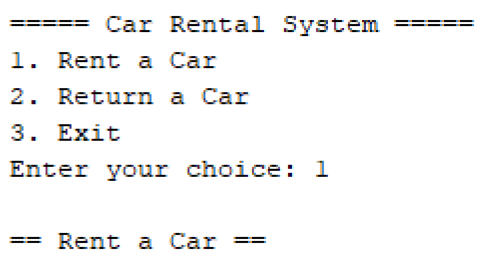
```java
}

System.out.print("\nEnter the car ID you want to rent: ");
String carId = scanner.nextLine();

System.out.print("Enter the number of days for rental: ");
int rentalDays = scanner.nextInt();
scanner.nextLine(); // Consume newline

Customer newCustomer = new Customer("CUS" + (customers.size() + 1), customerName);
addCustomer(newCustomer);

Car selectedCar = null;
for (Car car : cars) {
   if (car.getCarId().equals(carId) && car.isAvailable()) {
       selectedCar = car;
       break;
   }
}

if (selectedCar != null) {
   double totalPrice = selectedCar.calculatePrice(rentalDays);
   System.out.println("\n== Rental Information ==\n");
   System.out.println("Customer ID: " + newCustomer.getCustomerId());
   System.out.println("Customer Name: " + newCustomer.getName());
   System.out.println("Car: " + selectedCar.getBrand() + " " + selectedCar.getModel());
   System.out.println("Rental Days: " + rentalDays);
```

```
            System.out.printf("Total Price: $%.2f%n", totalPrice);


            System.out.print("\nConfirm rental (Y/N): ");
            String confirm = scanner.nextLine();


            if (confirm.equalsIgnoreCase("Y")) {
                rentCar(selectedCar, newCustomer, rentalDays);
                System.out.println("\nCar rented successfully.");
            } else {
                System.out.println("\nRental canceled.");
            }
        } else {
            System.out.println("\nInvalid car selection or car not available for rent.");
        }
    } else if (choice == 2) {
        System.out.println("\n== Return a Car ==\n");
        System.out.print("Enter the car ID you want to return: ");
        String carId = scanner.nextLine();


        Car carToReturn = null;
        for (Car car : cars) {
            if (car.getCarId().equals(carId) && !car.isAvailable()) {
                carToReturn = car;
                break;
            }
        }
```

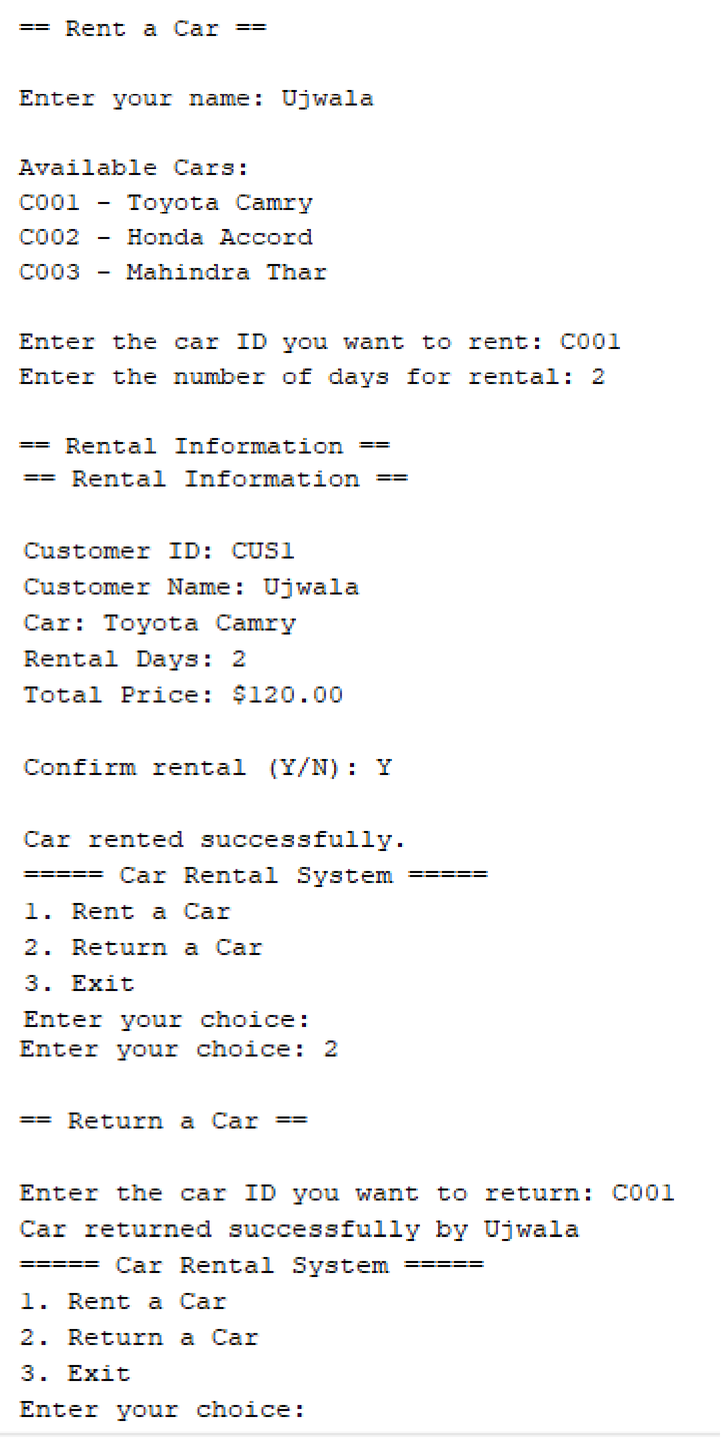
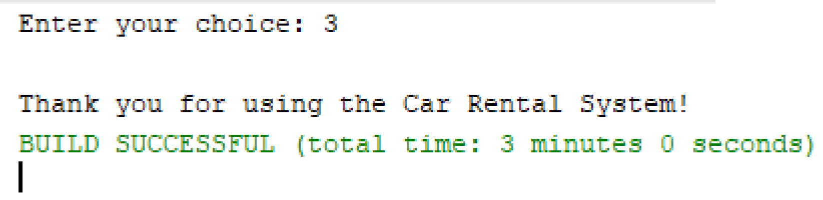
```java
                if (carToReturn != null) {
                    Customer customer = null;
                    for (Rental rental : rentals) {
                        if (rental.getCar() == carToReturn) {
                            customer = rental.getCustomer();
                            break;
                        }
                    }

                    if (customer != null) {
                        returnCar(carToReturn);
                        System.out.println("Car returned successfully by " + customer.getName());
                    } else {
                        System.out.println("Car was not rented or rental information is missing.");
                    }
                } else {
                    System.out.println("Invalid car ID or car is not rented.");
                }
            } else if (choice == 3) {
                break;
            } else {
                System.out.println("Invalid choice. Please enter a valid option.");
            }
        }

        System.out.println("\nThank you for using the Car Rental System!");
    }
```

```
}
public class Main{
    public static void main(String[] args) {
        CarRentalSystem rentalSystem = new CarRentalSystem();


        Car car1 = new Car("C001", "Toyota", "Camry", 60.0); // Different base price per day for each car
        Car car2 = new Car("C002", "Honda", "Accord", 70.0);
        Car car3 = new Car("C003", "Mahindra", "Thar", 150.0);
        rentalSystem.addCar(car1);
        rentalSystem.addCar(car2);
        rentalSystem.addCar(car3);


        rentalSystem.menu();
    }
}
```

Output:

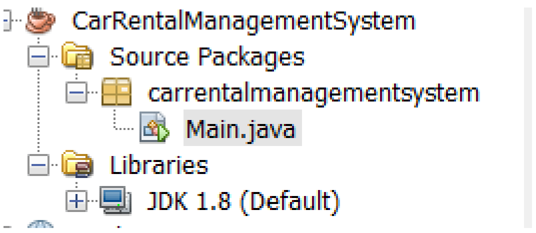


```
===== Car Rental System =====
1. Rent a Car
2. Return a Car
3. Exit
Enter your choice: 1

== Rent a Car ==
```

```
== Rent a Car ==

Enter your name: Ujwala

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

Enter the car ID you want to rent: C001
Enter the number of days for rental: 2

== Rental Information ==
== Rental Information ==

Customer ID: CUS1
Customer Name: Ujwala
Car: Toyota Camry
Rental Days: 2
Total Price: $120.00

Confirm rental (Y/N): Y

Car rented successfully.
===== Car Rental System =====
1. Rent a Car
2. Return a Car
3. Exit
Enter your choice:
Enter your choice: 2

== Return a Car ==

Enter the car ID you want to return: C001
Car returned successfully by Ujwala
===== Car Rental System =====
1. Rent a Car
2. Return a Car
3. Exit
Enter your choice:
Enter your choice: 3

Thank you for using the Car Rental System!
BUILD SUCCESSFUL (total time: 3 minutes 0 seconds)
```

Project Hierarchy:



```
CarRentalManagementSystem
  Source Packages
    carrentalmanagementsystem
      Main.java
  Libraries
    JDK 1.8 (Default)
```

How It Works

Renting a Car

1. Displays a list of available cars.
2. Prompts the user to select a car and specify rental duration.
3. Calculates the total price and confirms the rental.

Returning a Car

1. Prompts the user to input the car ID.
2. Updates the car's availability status and removes the rental record.

Future Enhancements 🚘

- ✅ Support multiple customers renting the same car simultaneously.
- Implement date-based pricing adjustments.
- 🎨 Develop a graphical user interface (GUI) for enhanced user experience.

Conclusion

The **Car Rental System** is an interactive console application that demonstrates object-oriented programming concepts through real-world scenarios. It's an excellent project for mastering Java while solving practical challenges.

Start your journey today and contribute to the project's growth! 🚗✨

Thank You…