(Coding)

```
import java.awt.*;
import java.awt.event.*;
import java.util.*; // Import java.util.List and other utility classes

// Abstract class for Vehicle
abstract class Vehicle {
    private String id;
    private String brand;
    private String model;
    private double pricePerDay;
    private boolean isAvailable;

    public Vehicle(String id, String brand, String model, double pricePerDay) {
        this.id = id;
        this.brand = brand;
        this.model = model;
        this.pricePerDay = pricePerDay;
        this.isAvailable = true;
    }

    public String getId() {
        return id;
    }

    public String getBrand() {
```

```
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double getPricePerDay() {
        return pricePerDay;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void setAvailable(boolean available) {
        isAvailable = available;
    }

    public abstract String getVehicleType();

    @Override
    public String toString() {
        return getVehicleType() + " | ID: " + id + " | Brand: " + brand + " | Model: " +
model +
                " | Price/Day: $" + pricePerDay + " | Available: " + isAvailable;
    }
```

```
}

// Car subclass for Vehicle
class Car extends Vehicle {
    public Car(String id, String brand, String model, double pricePerDay) {
        super(id, brand, model, pricePerDay);
    }

    @Override
    public String getVehicleType() {
        return "Car";
    }
}

// Payment class
class Payment {
    private String paymentId;
    private double amount;
    private String paymentMethod;
    private boolean isSuccessful;

    public Payment(double amount, String paymentMethod) {
        this.paymentId = UUID.randomUUID().toString();
        this.amount = amount;
        this.paymentMethod = paymentMethod;
        this.isSuccessful = true; // For simplicity, all payments are marked successful
    }
```

```
    public String getPaymentId() {
        return paymentId;
    }

    @Override
    public String toString() {
        return "Payment ID: " + paymentId + " | Amount: $" + amount +
                " | Method: " + paymentMethod + " | Status: " + (isSuccessful ?
"Successful" : "Failed");
    }
}

// Reservation class
class Reservation {
    private String reservationId;
    private String customerName;
    private Vehicle vehicle;
    private int rentalDays;
    private double totalCost;
    private Payment payment;

    public Reservation(String customerName, Vehicle vehicle, int rentalDays,
Payment payment) {
        this.reservationId = UUID.randomUUID().toString();
        this.customerName = customerName;
        this.vehicle = vehicle;
```

```
        this.rentalDays = rentalDays;
        this.totalCost = rentalDays * vehicle.getPricePerDay();
        this.payment = payment;
        vehicle.setAvailable(false); // Mark the vehicle as unavailable
    }

    public String getReservationId() {
        return reservationId;
    }

    @Override
    public String toString() {
        return "Reservation ID: " + reservationId + " | Customer: " + customerName +
                " | Vehicle: " + vehicle.getBrand() + " " + vehicle.getModel() +
                " | Rental Days: " + rentalDays + " | Total Cost: $" + totalCost +
                " | Payment: [" + payment + "]";
    }
}

// CarRentalSystem class
class CarRentalSystem {
    private java.util.List<Vehicle> fleet; // Explicitly using java.util.List
    private java.util.List<Reservation> reservations; // Explicitly using java.util.List

    public CarRentalSystem() {
        fleet = new ArrayList<>(); // Using java.util.ArrayList
        reservations = new ArrayList<>(); // Using java.util.ArrayList
```

```
    }

    public void addVehicle(Vehicle vehicle) {
        fleet.add(vehicle);
    }

    public String displayFleet() {
        StringBuilder sb = new StringBuilder("\n--- Fleet Details ---\n");
        for (Vehicle vehicle : fleet) {
            sb.append(vehicle).append("\n");
        }
        return sb.toString();
    }

    public String displayAvailableVehicles() {
        StringBuilder sb = new StringBuilder("\n--- Available Vehicles ---\n");
        for (Vehicle vehicle : fleet) {
            if (vehicle.isAvailable()) {
                sb.append(vehicle).append("\n");
            }
        }
        return sb.toString();
    }

    public String makeReservation(String customerName, String vehicleId, int
rentalDays, String paymentMethod) {
        Optional<Vehicle> vehicleOpt = fleet.stream()
```

```
            .filter(v -> v.getId().equals(vehicleId) && v.isAvailable())
            .findFirst();

        if (vehicleOpt.isPresent()) {
            Vehicle vehicle = vehicleOpt.get();
            Payment payment = new Payment(rentalDays * vehicle.getPricePerDay(),
paymentMethod);
            Reservation reservation = new Reservation(customerName, vehicle,
rentalDays, payment);
            reservations.add(reservation);
            return "Reservation successful!\n" + reservation;
        } else {
            return "Vehicle not available for reservation!";
        }
    }

    public String displayReservations() {
        StringBuilder sb = new StringBuilder("\n--- Reservations ---\n");
        for (Reservation reservation : reservations) {
            sb.append(reservation).append("\n");
        }
        return sb.toString();
    }
}

// Main class (CarRentalGUI)
public class CarRentalGUI {
```

```
public static void main(String[] args) {
    // Create CarRentalSystem instance
    CarRentalSystem system = new CarRentalSystem();
    system.addVehicle(new Car("CAR001", "Toyota", "Corolla", 50));
    system.addVehicle(new Car("CAR002", "Honda", "Civic", 60));
    system.addVehicle(new Car("CAR003", "Ford", "Focus", 55));
    system.addVehicle(new Car("CAR004", "Chevrolet", "Malibu", 65));
    system.addVehicle(new Car("CAR005", "BMW", "X5", 120));
    system.addVehicle(new Car("CAR006", "Audi", "A4", 110));

    // Create frame and set layout
    Frame frame = new Frame("Car Rental System");
    frame.setSize(600, 400);
    frame.setLayout(new FlowLayout());

    // Create TextArea for displaying information
    TextArea outputArea = new TextArea("", 15, 50,
TextArea.SCROLLBARS_VERTICAL_ONLY);
    outputArea.setEditable(false);
    frame.add(outputArea);

    // Create Buttons and TextFields for interaction
    Button btnDisplayFleet = new Button("Display Fleet");
    Button btnDisplayAvailable = new Button("Display Available Vehicles");
    Button btnMakeReservation = new Button("Make Reservation");
    Button btnDisplayReservations = new Button("Display Reservations");
```

```
        TextField nameField = new TextField(20);
        TextField vehicleIdField = new TextField(20);
        TextField rentalDaysField = new TextField(5);
        TextField paymentMethodField = new TextField(20);

        frame.add(new Label("Customer Name:"));
        frame.add(nameField);
        frame.add(new Label("Vehicle ID:"));
        frame.add(vehicleIdField);
        frame.add(new Label("Rental Days:"));
        frame.add(rentalDaysField);
        frame.add(new Label("Payment Method:"));
        frame.add(paymentMethodField);

        frame.add(btnDisplayFleet);
        frame.add(btnDisplayAvailable);
        frame.add(btnMakeReservation);
        frame.add(btnDisplayReservations);

        // Button Actions
        btnDisplayFleet.addActionListener(e ->
outputArea.setText(system.displayFleet()));
        btnDisplayAvailable.addActionListener(e ->
outputArea.setText(system.displayAvailableVehicles()));

        btnMakeReservation.addActionListener(e -> {
            String name = nameField.getText();
```

```
            String vehicleId = vehicleIdField.getText();
            int rentalDays = Integer.parseInt(rentalDaysField.getText());
            String paymentMethod = paymentMethodField.getText();
            outputArea.setText(system.makeReservation(name, vehicleId, rentalDays,
paymentMethod));
        });

        btnDisplayReservations.addActionListener(e ->
outputArea.setText(system.displayReservations()));

        // Display the frame
        frame.setVisible(true);

        // Close the frame on exit
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }
}
```