import java.util.ArrayList;

import java.util.List;

// Abstract Class: Vehicle

public abstract class Vehicle {

private String licensePlate;

private String make;

private String model;

private double baseRentalRate;

public Vehicle(String licensePlate, String make, String model, double baseRentalRate) {

this.licensePlate = licensePlate;

this.make = make;

this.model = model;

this.baseRentalRate = baseRentalRate;

}

public abstract double calculateRentalCost(int days);

public abstract boolean isAvailableForRental();

// Getters and Setters

public String getLicensePlate() {

return licensePlate;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public double getBaseRentalRate() {

return baseRentalRate;

}

public void setLicensePlate(String licensePlate) {

this.licensePlate = licensePlate;

}

public void setMake(String make) {

this.make = make;

}

public void setModel(String model) {

this.model = model;

}

public void setBaseRentalRate(double baseRentalRate) {

if (baseRentalRate > 0) {

this.baseRentalRate = baseRentalRate;

} else {

throw new IllegalArgumentException("Rental rate must be positive.");

}

}

}

// Concrete Classes: Car, Motorcycle, Truck

public class Car extends Vehicle implements Rentable {

private boolean isConvertible;

public Car(String licensePlate, String make, String model, double baseRentalRate, boolean isConvertible) {

super(licensePlate, make, model, baseRentalRate);

this.isConvertible = isConvertible;

}

@Override

public double calculateRentalCost(int days) {

return getBaseRentalRate() \* days;

}

@Override

public boolean isAvailableForRental() {

// Logic to determine availability

return true; // Placeholder

}

@Override

public void rent(Customer customer, int days) {

// Logic to rent the car to the customer

System.out.println("Car rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

// Logic to process the return

System.out.println("Car returned.");

}

}

public class Motorcycle extends Vehicle implements Rentable {

private boolean hasSidecar;

public Motorcycle(String licensePlate, String make, String model, double baseRentalRate, boolean hasSidecar) {

super(licensePlate, make, model, baseRentalRate);

this.hasSidecar = hasSidecar;

}

@Override

public double calculateRentalCost(int days) {

return getBaseRentalRate() \* days \* 0.9; // Discount for motorcycles

}

@Override

public boolean isAvailableForRental() {

return true; // Placeholder

}

@Override

public void rent(Customer customer, int days) {

System.out.println("Motorcycle rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

System.out.println("Motorcycle returned.");

}

}

public class Truck extends Vehicle implements Rentable {

private double loadCapacity;

public Truck(String licensePlate, String make, String model, double baseRentalRate, double loadCapacity) {

super(licensePlate, make, model, baseRentalRate);

this.loadCapacity = loadCapacity;

}

@Override

public double calculateRentalCost(int days) {

return getBaseRentalRate() \* days + 50; // Additional fee for trucks

}

@Override

public boolean isAvailableForRental() {

return true; // Placeholder

}

@Override

public void rent(Customer customer, int days) {

System.out.println("Truck rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

System.out.println("Truck returned.");

}

}

// Interface: Rentable

public interface Rentable {

void rent(Customer customer, int days);

void returnVehicle();

}

// Supporting Class: Customer

public class Customer {

private String name;

private String driverLicense;

public Customer(String name, String driverLicense) {

this.name = name;

this.driverLicense = driverLicense;

}

public String getName() {

return name;

}

public String getDriverLicense() {

return driverLicense;

}

}

// Supporting Class: RentalAgency

public class RentalAgency {

private List<Vehicle> vehicles;

public RentalAgency() {

this.vehicles = new ArrayList<>();

}

public void addVehicle(Vehicle vehicle) {

vehicles.add(vehicle);

}

public Vehicle findVehicle(String licensePlate) {

for (Vehicle vehicle : vehicles) {

if (vehicle.getLicensePlate().equals(licensePlate)) {

return vehicle;

}

}

return null; // Not found

}

public void rentVehicle(String licensePlate, Customer customer, int days) {

Vehicle vehicle = findVehicle(licensePlate);

if (vehicle != null && vehicle.isAvailableForRental()) {

vehicle.rent(customer, days);

} else {

System.out.println("Vehicle not available.");

}

}

}

// Supporting Class: RentalTransaction

public class RentalTransaction {

private Customer customer;

private Vehicle vehicle;

private int rentalDays;

public RentalTransaction(Customer customer, Vehicle vehicle, int rentalDays) {

this.customer = customer;

this.vehicle = vehicle;

this.rentalDays = rentalDays;

}

public double calculateTotalCost() {

return vehicle.calculateRentalCost(rentalDays);

}

// Additional methods for transaction management

}