Car Rental System

→ A car rental system is an application that manages the renting of automobile for short time. A car rental System keeps a list of all its clients in a database. A member can reserve a car for a certain number of days, hire a car, or return the car that was rented.

Expectation From the Interviewee

- What type of vehicle will that system support?
- How can we identify the Specific Vehicle?
- Is it possible to search a vehicle using (its name or type)
- Can we search for a vehicle by its model number?
- Does a car rental system assign a driver to its customer
- Does a car rental System provide roadside assistance to its customer?
- Can the member be able to cancel a reservations?
- which member is allowed to request a vehicle reservations Cancellation & when?
- How can customers pay at different branch locations & by different methods (Cash, Credit or cheque)?
- If there are multiple branches of the car rental system, how will the system keep track of the Customer having already paid at a particular branch

Requirement Collection of Car Rental System

R1: There can be two types of users in the car rental system i.e. customers & receptionists.

R2: The system should handle multiple types of vehicle. Initially, the system should cater to the following vehicle: cars, trucks, vans & motorcycles.

R3: There can be multiple subtypes of vehicle. The car type can be economy, luxury, standard & compact. The van type can be passenger or cargo. Moreover, the motorcycle type can be cruiser, touring or sports. The truck type can be light, medium or high-duty.

R4: The system should be able to keep a record of who reserved a particular vehicle & on which date the vehicle was issued.

R5: The system should be able to find out how many vehicles have been rented out by the specific customer.

R6: The customer should be able to cancel their reservation

R7: To keep track of all events related to the vehicle, the system should maintain a vehicle log.

R8: The system should allow the users to add service to the reservations like a driver, Wi-fi & roadside assistance.

R9: The system should allow the users to add equipment to the reservations like a ski rack, child seat & Navigation

R10: The system should send a notification to the customer & generate a fine if the vehicle is returned within the due date.

R11: The system should allow the user to search the vehicles by type or model

R12: A system should be able to manage the multiple branches of the car rental

R13: Every branch of the car rental system should have parking stalls to park the vehicles.

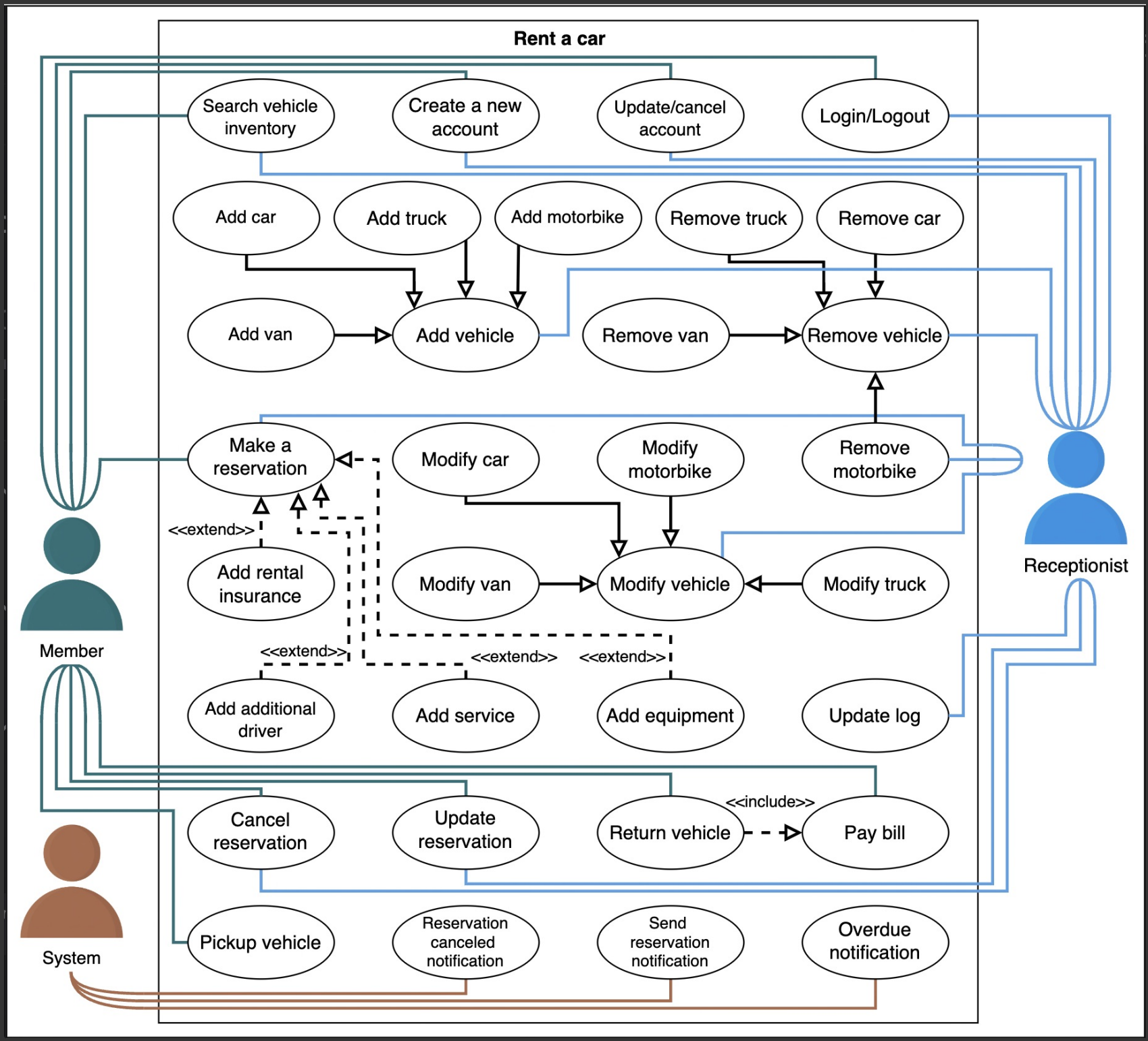
System

Primary Actors → Member

Secondary Actors → Receptionists, Worker, System

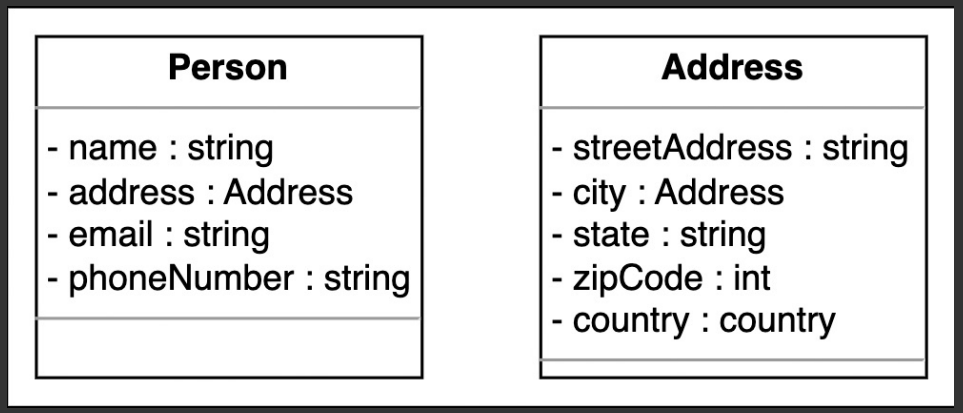
Member	Receptionist	System
Create a new account	Create a new account	Reservation canceled notification
Update/cancel account	Update/cancel account	Send reservation notification
Login/Logout	Login/Logout	Overdue notification
Search vehicle inventory	Search vehicle inventory	
Make a reservation	Make a reservation	
Cancel reservation	Cancel reservation	
Update reservation	Update reservation	
Pickup vehicle	Add vehicle	
Return vehicle	Remove vehicle	
Pay bill	Modify vehicle	
	Update log	

Use Case Diagram

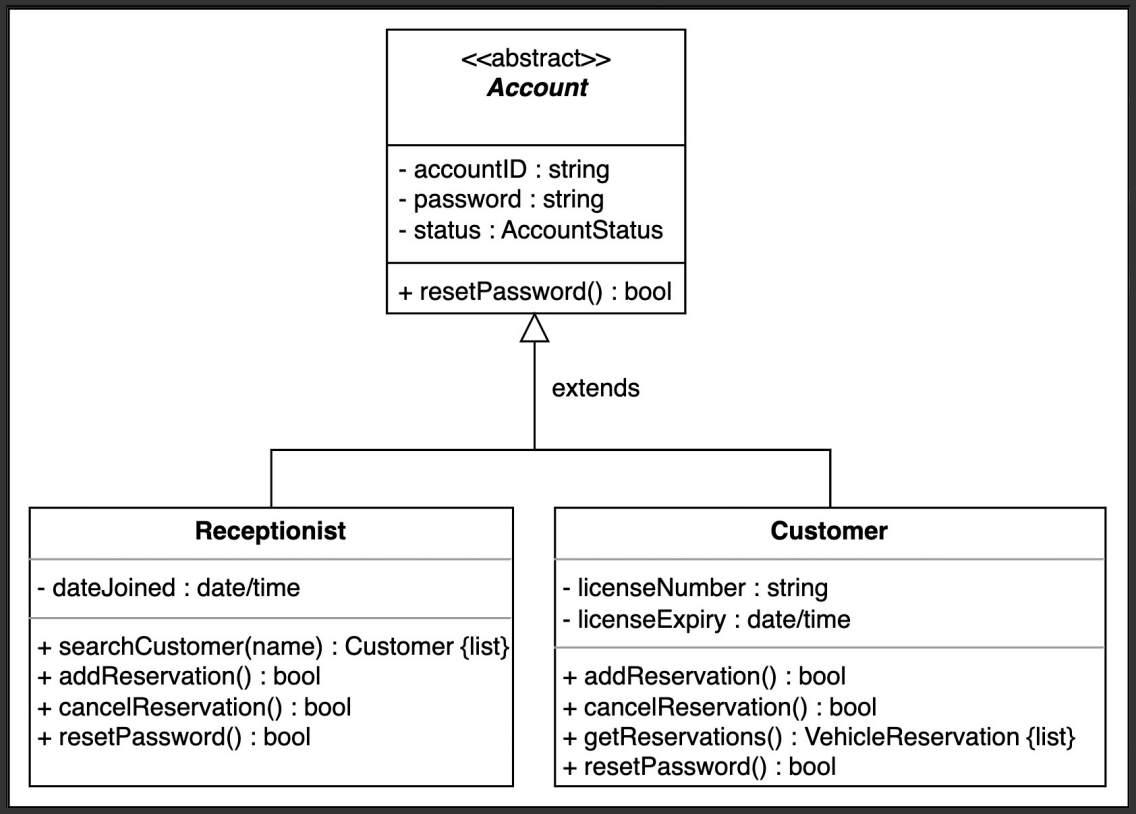


Class Diagram for Car Rental System

Address & Person



Account

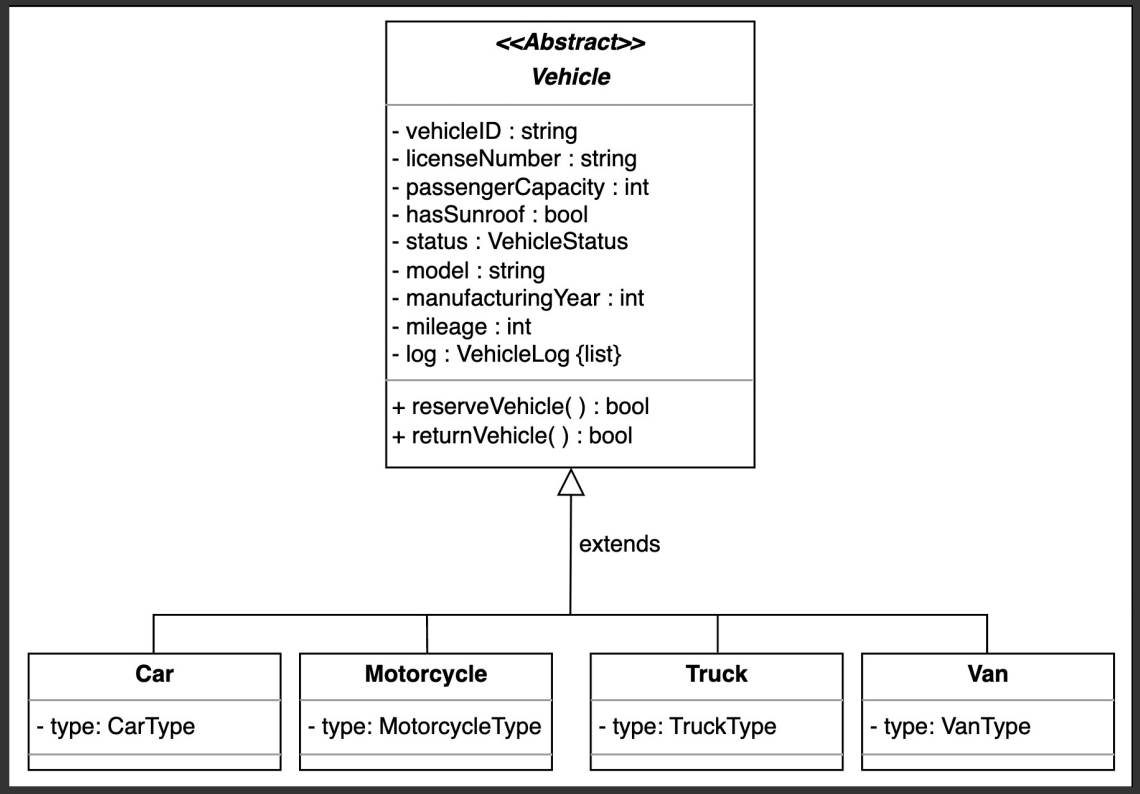


Driver

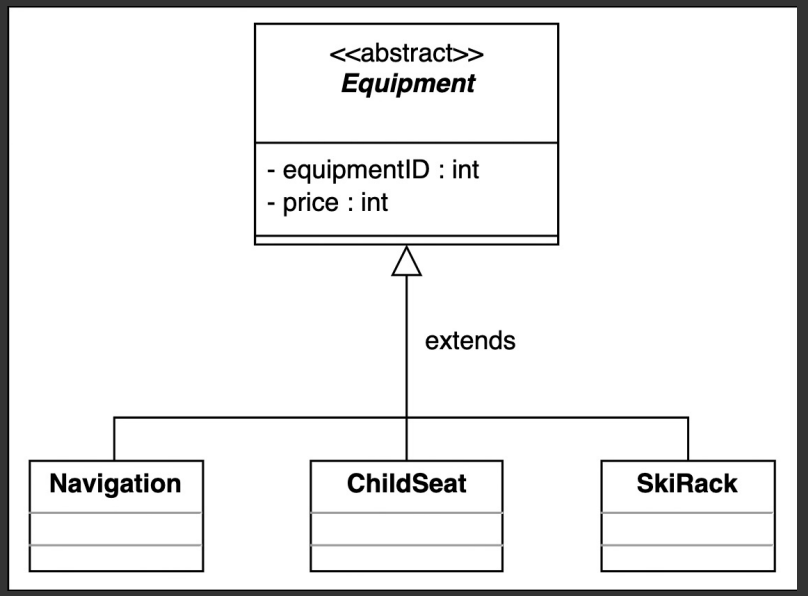
Driver

- driverID : int

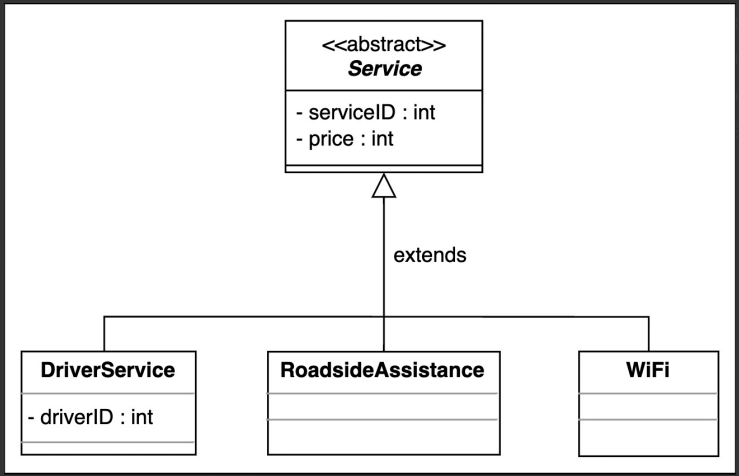
Vehicle



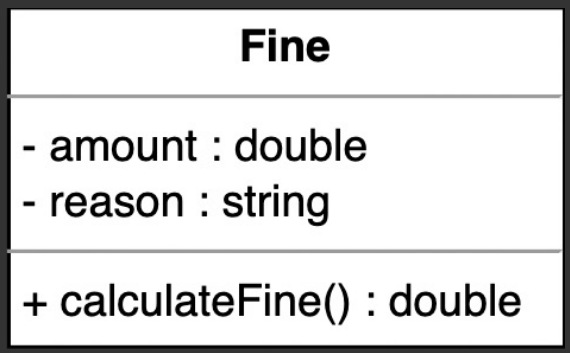
Equipment



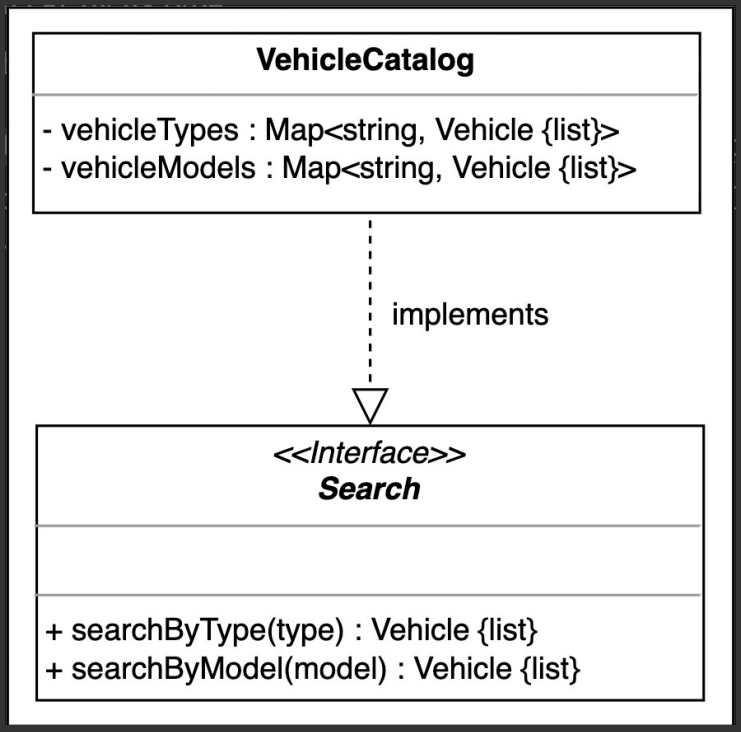
Service



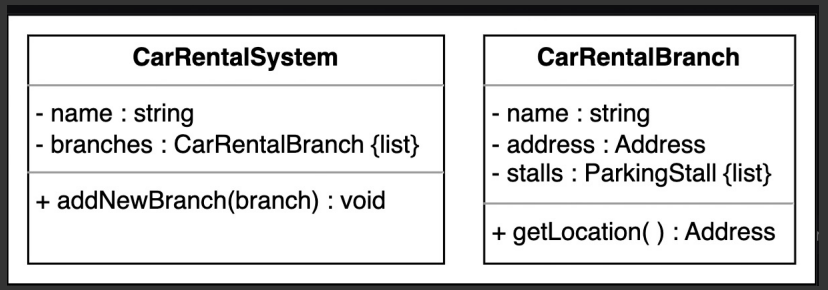
Fine



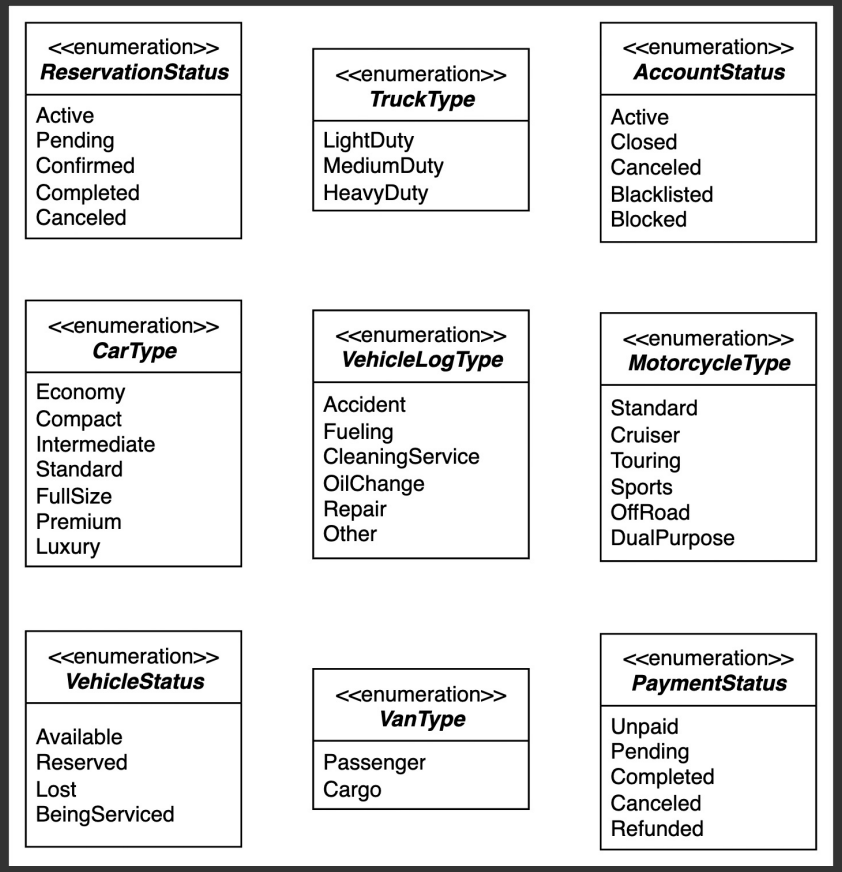
Vehicle Catalog



Car Rental & Branch

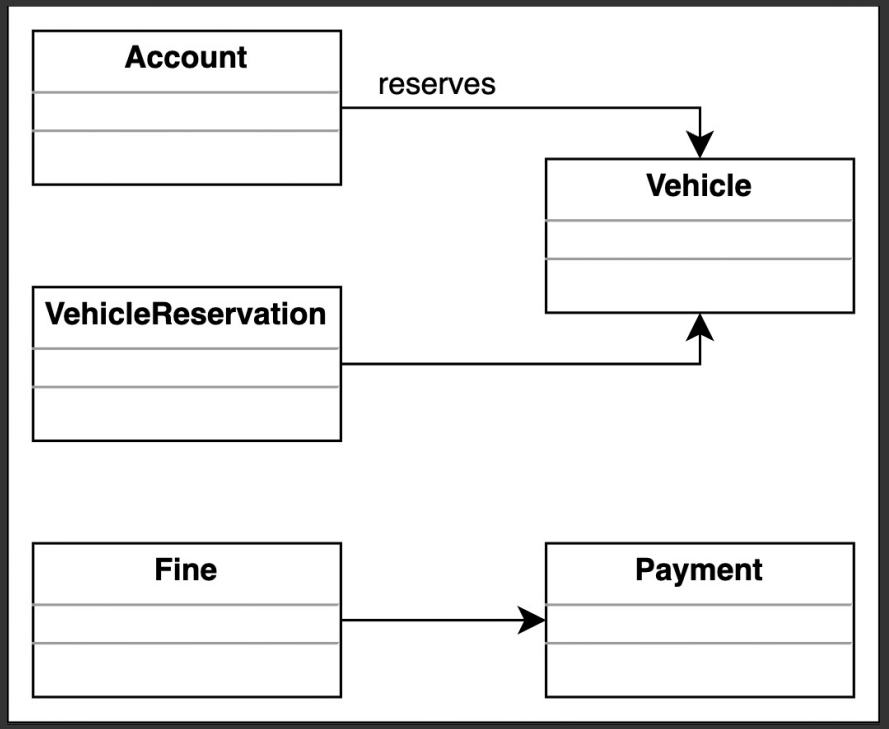


Enumerations

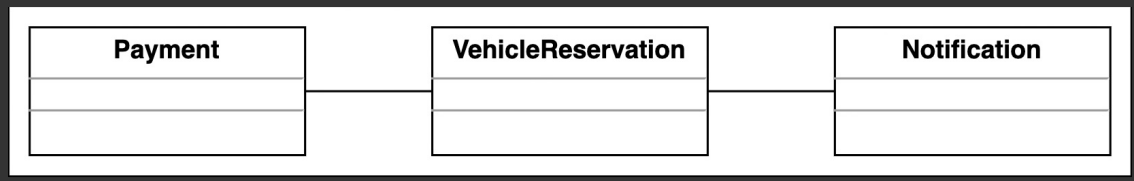


Associations

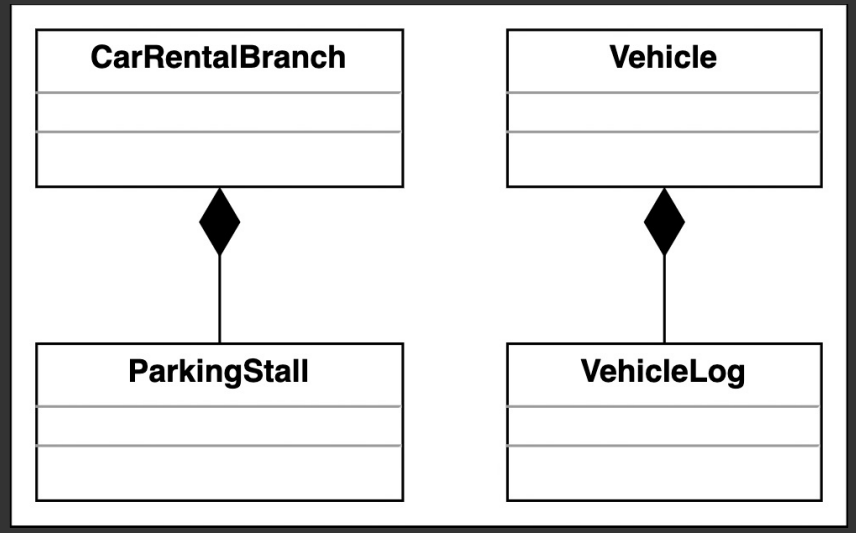
1. One-Way Association



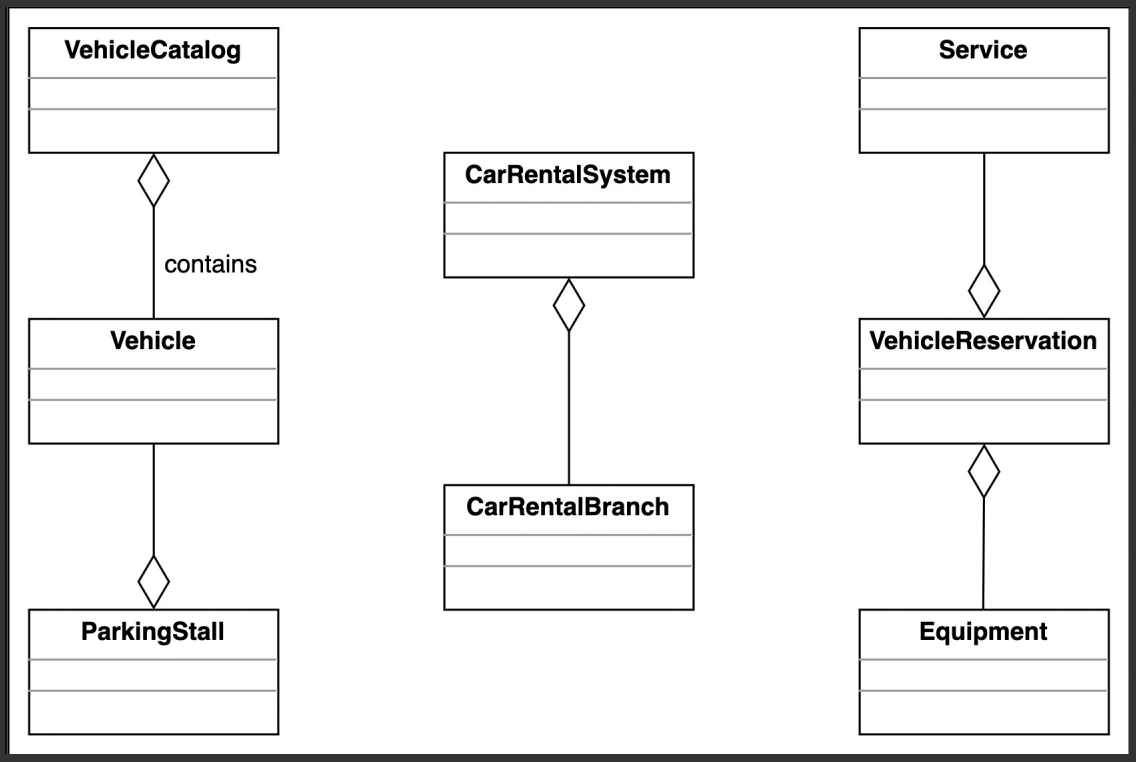
2. Two-way association



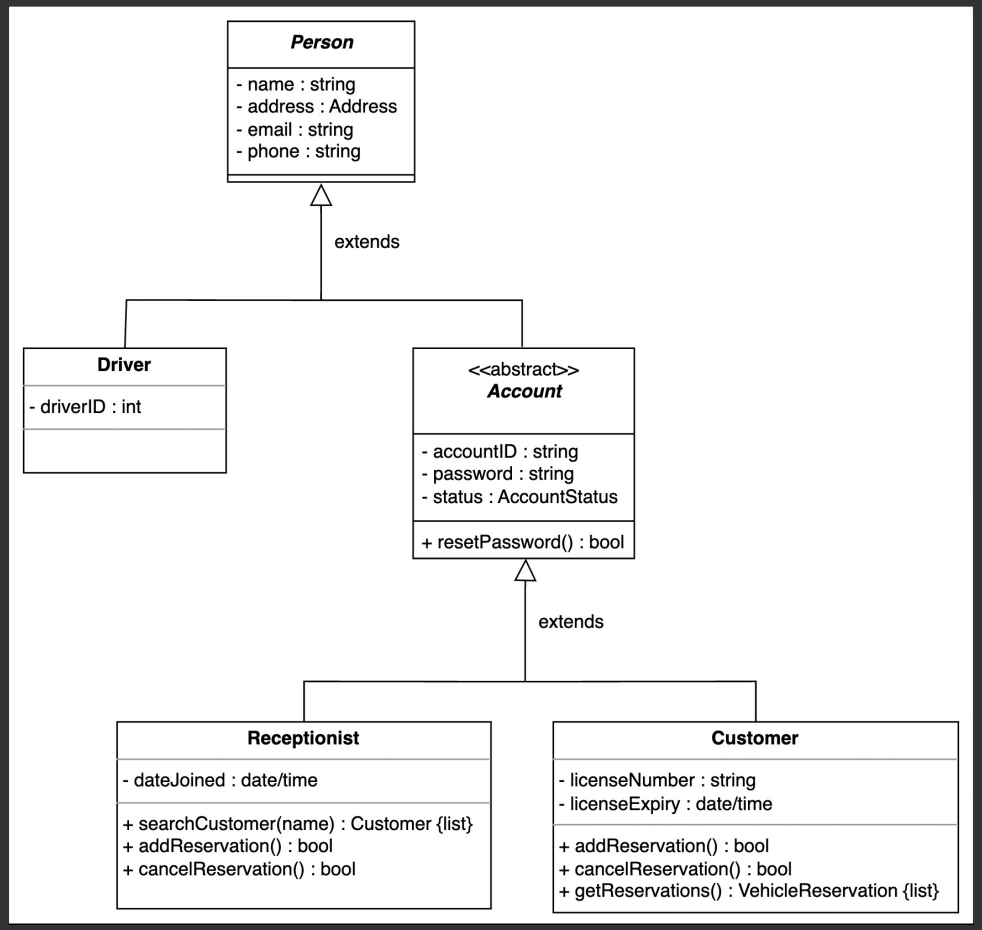
3. Composition

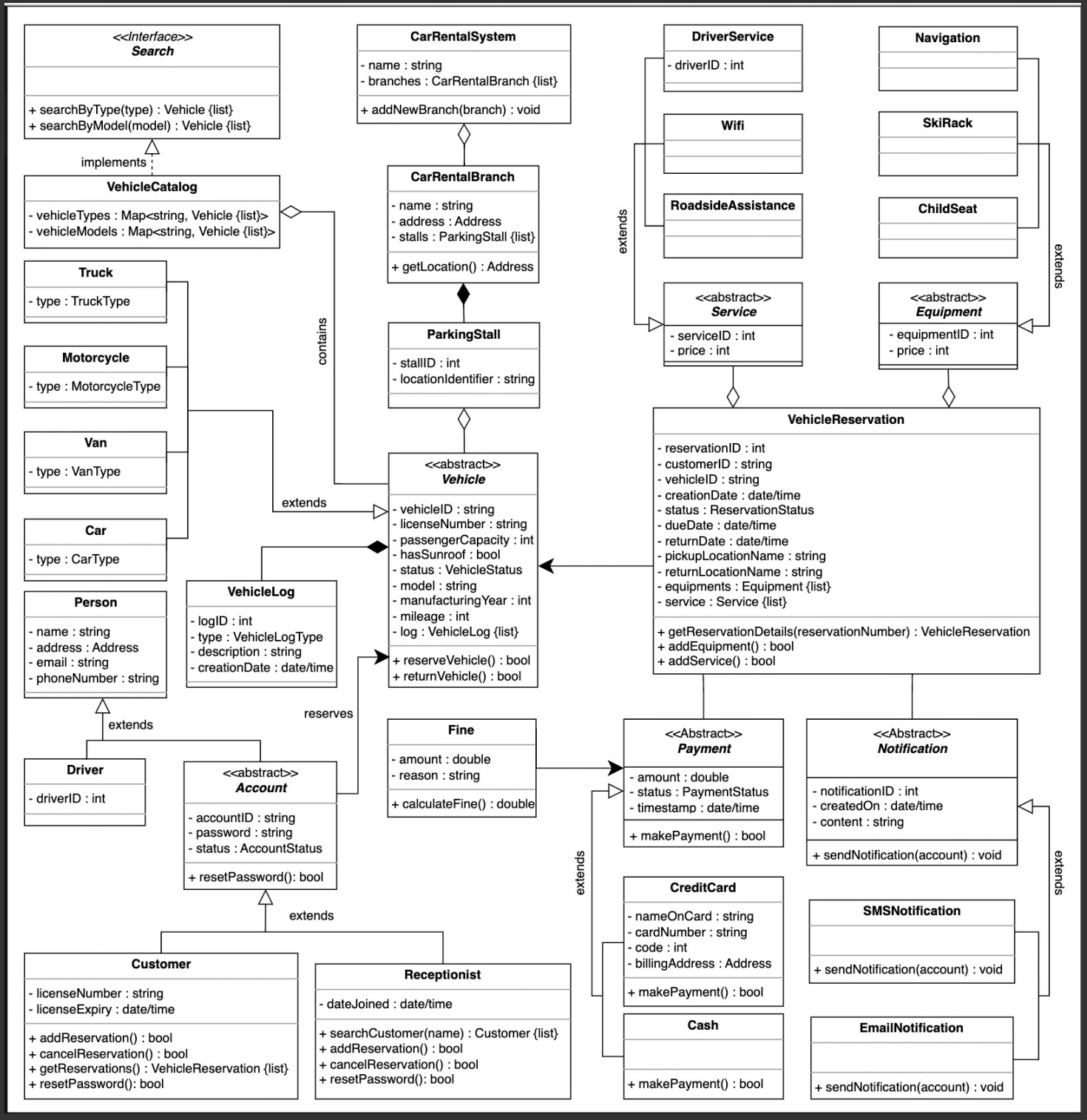


4. Aggregation



Inheritance

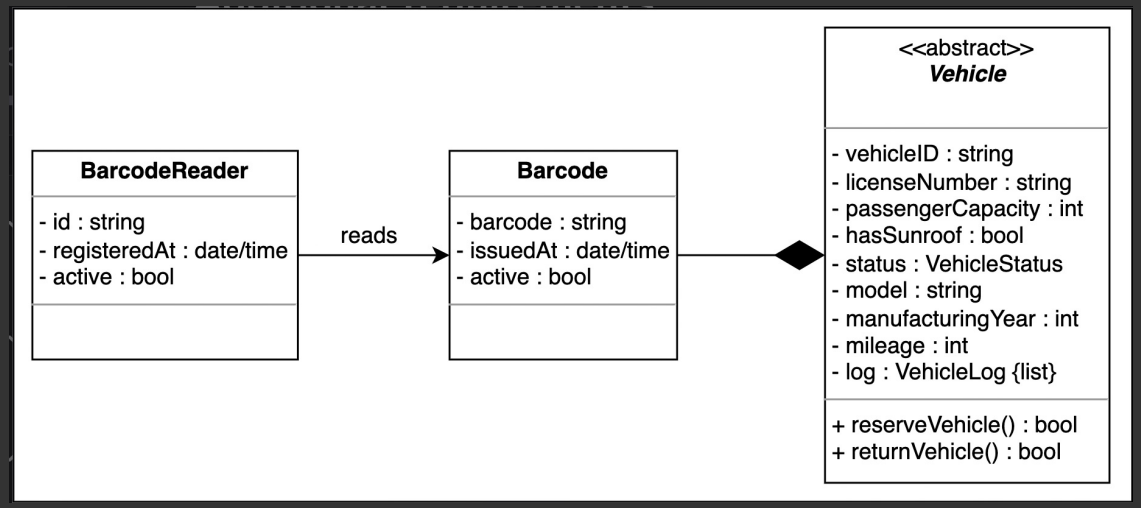



<<Interface>>
Search
+ searchByType(type) : Vehicle {list}
+ searchByModel(model) : Vehicle {list}
implements
CarRentalSystem
- name : string
- branches : CarRentalBranch {list}
+ addNewBranch(branch) : void
CarRentalBranch
- name : string
- address : Address
- stalls : ParkingStall {list}
+ getLocation() : Address
ParkingStall
- stallID : int
- locationIdentifier : string
VehicleCatalog
- vehicleTypes : Map<string, Vehicle {list}>
- vehicleModels : Map<string, Vehicle {list}>
contains
Truck
- type : TruckType
Motorcycle
- type : MotorcycleType
Van
- type : VanType
Car
- type : CarType
extends
<<abstract>>
Vehicle
- vehicleID : string
- licenseNumber : string
- passengerCapacity : int
- hasSunroof : bool
- status : VehicleStatus
- model : string
- manufacturingYear : int
- mileage : int
- log : VehicleLog {list}
+ reserveVehicle() : bool
+ returnVehicle() : bool
VehicleLog
- logID : int
- type : VehicleLogType
- description : string
- creationDate : date/time
Person
- name : string
- address : Address
- email : string
- phoneNumber : string
extends
Driver
- driverID : int
reserves
<<abstract>>
Account
- accountID : string
- password : string
- status : AccountStatus
+ resetPassword(): bool
extends
Customer
- licenseNumber : string
- licenseExpiry : date/time
+ addReservation() : bool
+ cancelReservation() : bool
+ getReservations() : VehicleReservation {list}
+ resetPassword(): bool
Receptionist
- dateJoined : date/time
+ searchCustomer(name) : Customer {list}
+ addReservation() : bool
+ cancelReservation() : bool
+ resetPassword(): bool
Fine
- amount : double
- reason : string
+ calculateFine() : double
DriverService
- driverID : int
Wifi
RoadsideAssistance
extends
<<abstract>>
Service
- serviceID : int
- price : int
Navigation
SkiRack
ChildSeat
extends
<<abstract>>
Equipment
- equipmentID : int
- price : int
VehicleReservation
- reservationID : int
- customerID : string
- vehicleID : string
- creationDate : date/time
- status : ReservationStatus
- dueDate : date/time
- returnDate : date/time
- pickupLocationName : string
- returnLocationName : string
- equipments : Equipment {list}
- service : Service {list}
+ getReservationDetails(reservationNumber) : VehicleReservation
+ addEquipment() : bool
+ addService() : bool
<<Abstract>>
Payment
- amount : double
- status : PaymentStatus
- timestamp : date/time
+ makePayment() : bool
extends
CreditCard
- nameOnCard : string
- cardNumber : string
- code : int
- billingAddress : Address
+ makePayment() : bool
Cash
+ makePayment() : bool
<<Abstract>>
Notification
- notificationID : int
- createdOn : date/time
- content : string
+ sendNotification(account) : void
extends
SMSNotification
+ sendNotification(account) : void
EmailNotification
+ sendNotification(account) : void

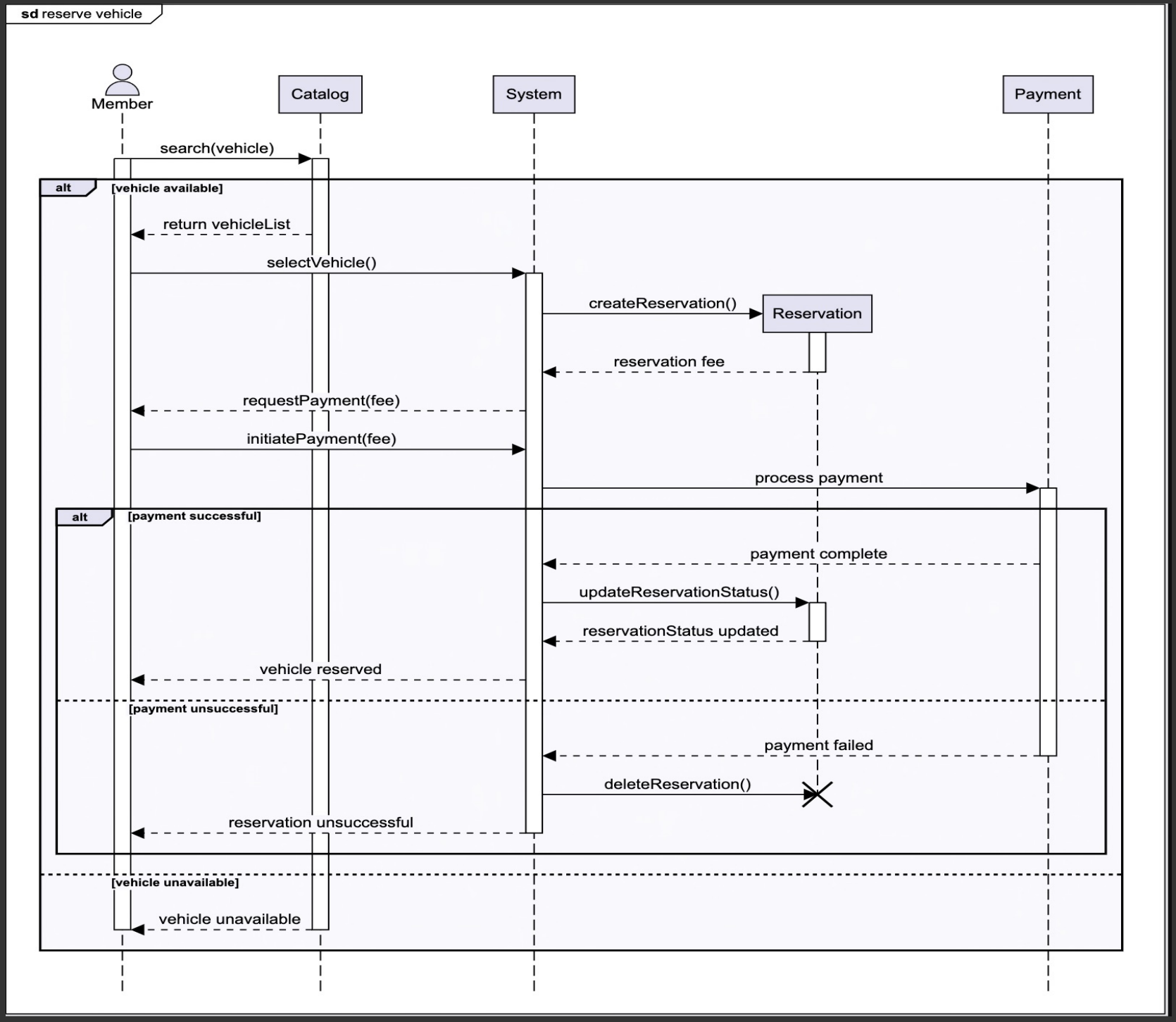
We can use Decorator Design pattern for our Car Rental System

→ Discount Decorator
→ Peak Season Decorator
→ Damage fine Decorator
→ Partially filled fuel tank fine decorator

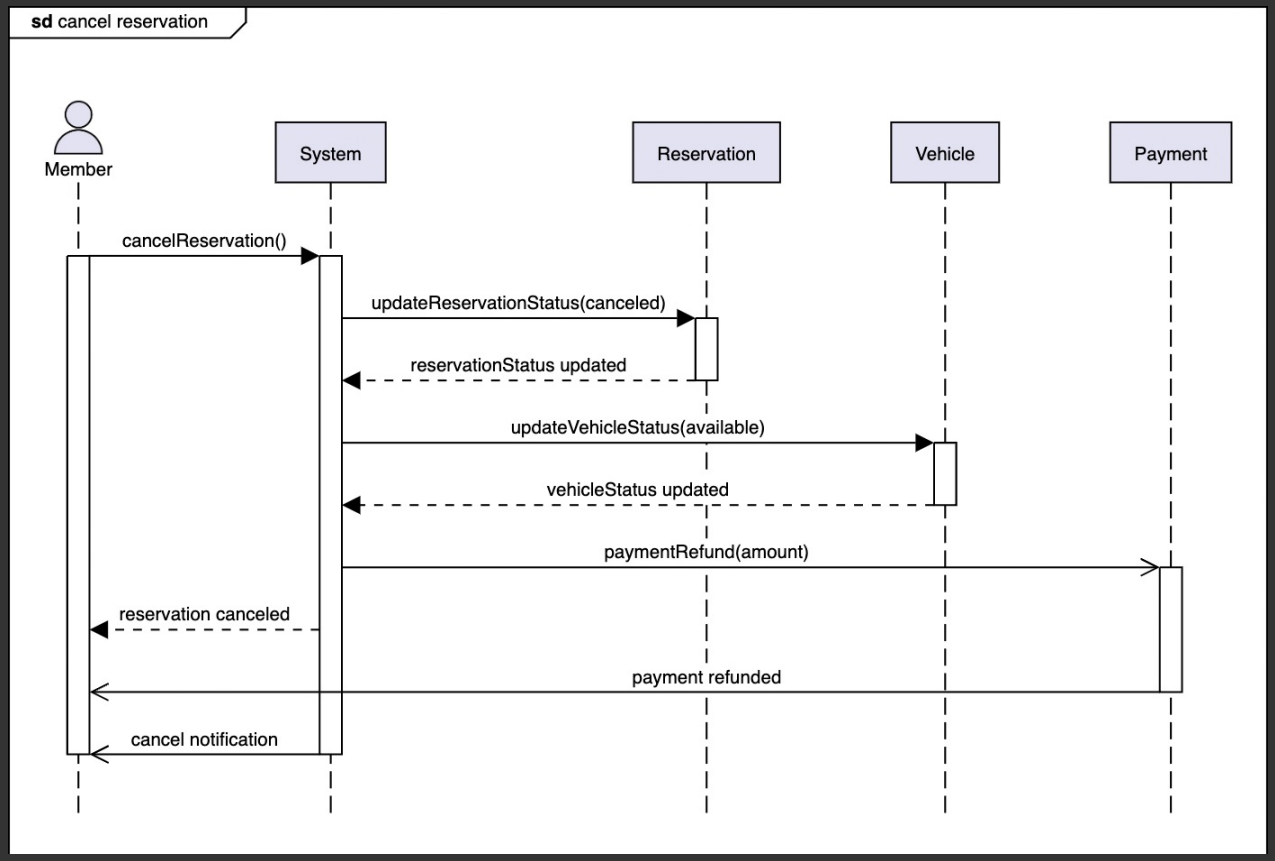
Additional Requirements

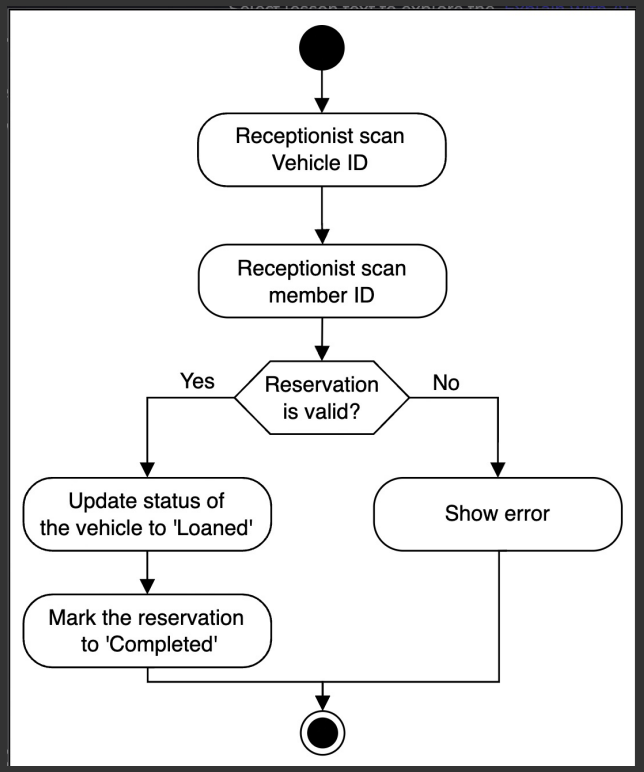


Vehicle Reservation → Sequence diagram

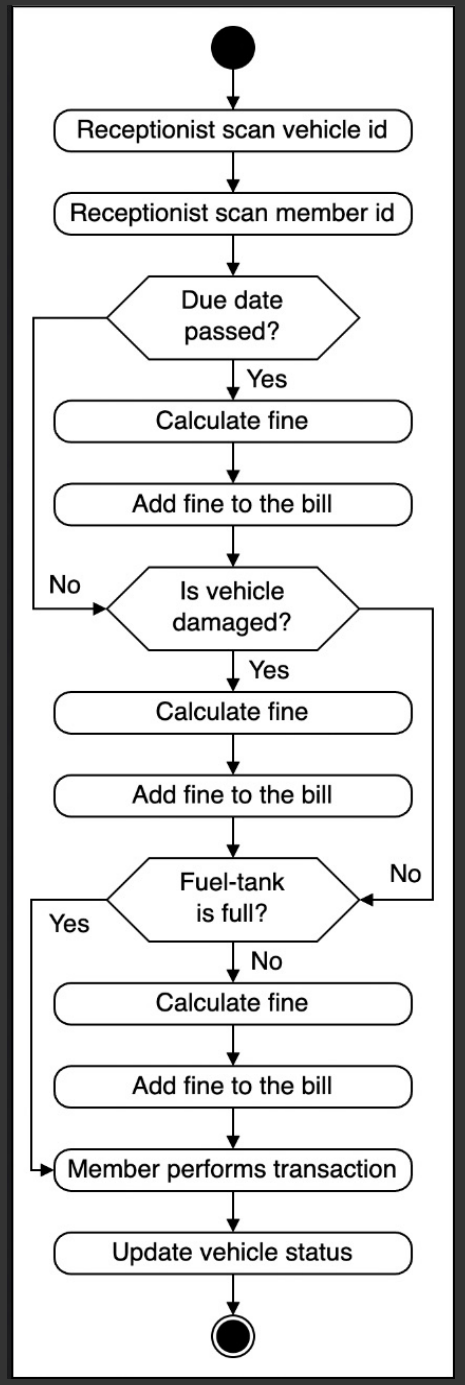


Cancel Reservations





Activity Diagram of Vehicle pickup System



activity diagram for vehicle return.

Code for Car Rental System

Enumerations

```
// definition of enumerations used in the car rental system
enum VehicleStatus {
  AVAILABLE,
  RESERVED,
  LOST,
  BEING_SERVICED
}

enum AccountStatus {
  ACTIVE,
  CLOSED,
  CANCELED,
  BLACKLISTED,
  BLOCKED
}

enum ReservationStatus {
  ACTIVE,
  PENDING,
  CONFIRMED,
  COMPLETED,
  CANCELED
}

enum PaymentStatus {
  UNPAID,
  PENDING,
  COMPLETED,
  CANCELED,
  REFUNDED
}

enum VanType {
  PASSENGER,
  CARGO
}
```

```
enum CarType {
  ECONOMY,
  COMPACT,
  INTERMEDIATE,
  STANDARD,
  FULL_SIZE,
  PREMIUM,
  LUXURY
}

enum MotorcycleType {
  STANDARD,
  CRUISER,
  TOURING,
  SPORTS,
  OFF_ROAD,
  DUAL_PURPOSE
}

enum TruckType {
  LIGHT_DUTY,
  MEDIUM_DUTY,
  HEAVY_DUTY
}

enum VehicleLogType {
  ACCIDENT,
  FUELING,
  CLEANING_SERVICE,
  OIL_CHANGE,
  REPAIR,
  OTHER
}
```

2. Address, person & driver

```
public class Address {
  private String streetAddress;
  private String city;
  private String state;
  private int zipCode;
  private String country;
}

public class Person {
  private String name;
  private Address address;
  private String email;
  private String phoneNumber;
}

public class Driver extends Person {
    private int driverId;
}
```

3. Account

```
public abstract class Account extends Person {
    private String accountId;
    private String password;
    private AccountStatus status;

    public abstract boolean resetPassword();
}

public class Receptionist extends Account {
    private Date dateJoined;

    public List<Customer> searchCustomer(String name);
    public boolean addReservation();
    public boolean cancelReservation();
    public boolean resetPassword() {
        // definition
    }
}

public class Customer extends Account {
    private String licenseNumber;
    private Date licenseExpiry;

    public boolean addReservation();
    public boolean cancelReservation();
    public List<VehicleReservation> getReservations();
    public boolean resetPassword() {
        // definition
    }
```

→ 4. Vehicle

```
// Vehicle is an abstract class
public abstract class Vehicle {
  private String vehicleId;
  private String licenseNumber;
  private int passengerCapacity;
  private boolean hasSunroof;
  private VehicleStatus status;
  private String model;
  private int manufacturingYear;
  private int mileage;
  private List<VehicleLog> log;

  public boolean reserveVehicle();
  public boolean returnVehicle();
}

public class Car extends Vehicle {
  private CarType carType;
}

public class Van extends Vehicle {
  private VanType vanType;
}

public class Truck extends Vehicle {
  private TruckType truckType;
}

public class Motorcycle extends Vehicle {
  private MotorcycleType motorcycleType;
}
```

5. Equipment

```
// Equipment is an abstract class
public abstract class Equipment {
    private int equipmentId;
    private int price;
}

public class Navigation extends Equipment {
}

public class ChildSeat extends Equipment {
}

public class SkiRack extends Equipment {
}
```

6. Service

```
// Service is an abstract class
public abstract class Service {
    private int serviceId;
    private int price;
}

public class DriverService extends Service {
    private int driverId;
}

public class RoadsideAssistance extends Service {
}

public class WiFi extends Service {
}
```

7. Payment

```
// Payment is an abstract class
public abstract class Payment {
  // Data members
  private double amount;

  // The Date datatype represents and deals
with both date and time.
  private Date timestamp;
  private PaymentStatus status;

  public abstract boolean makePayment();
}

public class Cash extends Payment {
    public boolean makePayment() {
        // functionality
    }
}

public class CreditCard extends Payment {
    // Data members
    private String nameOnCard;
    private String cardNumber;
    private String billingAddress;
    private int code;

    public boolean makePayment() {
        // functionality
    }
}
```

8. Vehicle log & Vehicle Reservation

```
public class VehicleLog {
  private int logId;
  private VehicleLogType logType;
  private String description;
  private Date creationDate;
}

public class VehicleReservation {
  private int reservationId;
  private String customerId;
  private String vehicleId;
  private Date creationDate;
  private ReservationStatus status;
  private Date dueDate;
  private Date returnDate;
  private String pickupLocation;
  private String returnLocation;

  private List<Equipment> equipments;
  private List<Service> services;

  public static VehicleReservation getReservationDetails();
  public boolean addEquipment();
  public boolean addService();
}
```

9. Notification

```
// Notification is an abstract class
public abstract class Notification {
    private int notificationId;
    // The Date data type represents and deals with
both date and time.
    private Date createdOn;
    private String content;

    public abstract void sendNotification(Account
account);
}

class SmsNotification extends Notification {

    public void sendNotification(Account account) {
        // functionality
    }
}

class EmailNotification extends Notification {

    public void sendNotification(Account account) {
        // functionality
    }
}
```

10. Parking Stall & Fine

```
public class ParkingStall {
  private int stallId;
  private String locationIdentifier;
}

public class Fine {
  private double amount;
  private String reason;
  public double calculateFine();
}
```

11. Search Interface & Vehicle Catalog

```
public interface Search {
  public List<Vehicle> searchByType(String type);
  public List<Vehicle> searchByModel(String model);
}

public class VehicleCatalog implements Search {
  private HashMap<String, List<Vehicle>> vehicleTypes;
  private HashMap<String, List<Vehicle>> vehicleModels;

  // to return all vehicles of the given type.
  public List<Vehicle> searchByType(String type) {
    // functionality
  }

  // to return all vehicles of the given model.
  public List<Vehicle> searchByModel(String model) {
    // functionality
  }
}
```

12. Car Rental System & Car Rental branch

```
public class CarRentalBranch {
  private String name;
  private Address address;
  private List<ParkingStall> stalls;

  public Address getLocation();
}

public class CarRentalSystem {
  private String name;
  private List<CarRentalBranch> branches;

  public void addNewBranch(CarRentalBranch branch);

  // The CarRentalSystem is a singleton class that ensures it will have only one active instance at a
time
    private static CarRentalSystem system = null;

    // Created a static method to access the singleton instance of CarRentalSystem class
    public static CarRentalSystem getInstance() {
        if (system == null) {
            system = new CarRentalSystem();
        }
        return system;
    }
}
```