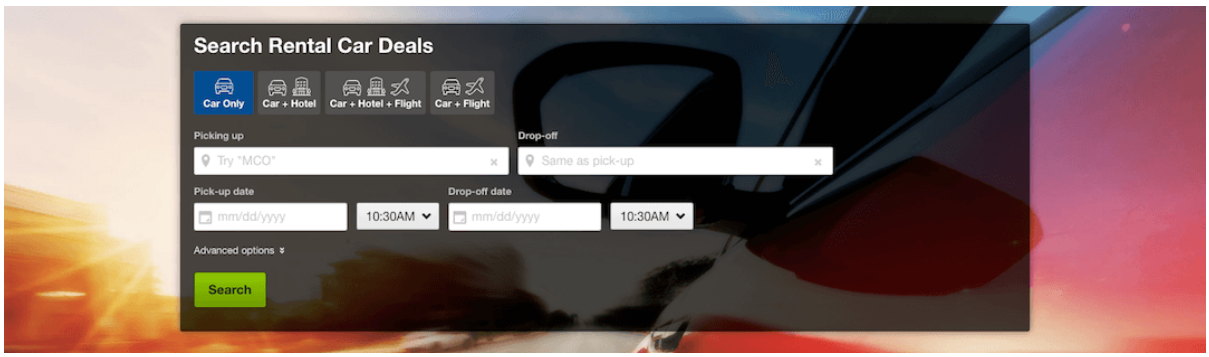
Design a Car Rental System

Let's design a car rental system where customers can rent vehicles.

A Car Rental System is a software built to handle the renting of automobiles for a short period of time, generally ranging from a few hours to a few weeks. A car rental system often has numerous local branches (to allow its user to return a vehicle to a different location), and primarily located near airports or busy city areas.



System Requirements

We will focus on the following set of requirements while designing our Car Rental System:

1. The system will support the renting of different automobiles like cars, trucks, SUVs, vans, and motorcycles.

2. Each vehicle should be added with a unique barcode and other details, including a parking stall number which helps to locate the vehicle.

3. The system should be able to retrieve information like which member took a particular vehicle or what vehicles have been rented out by a specific member.

4. The system should collect a late-fee for vehicles returned after the due date.
5. Members should be able to search the vehicle inventory and reserve any available vehicle.
6. The system should be able to send notifications whenever the reservation is approaching the pick-up date, as well as when the vehicle is nearing the due date or has not been returned within the due date.
7. The system will be able to read barcodes from vehicles.
8. Members should be able to cancel their reservations.
9. The system should maintain a vehicle log to track all events related to the vehicles.
10. Members can add rental insurance to their reservation.
11. Members can rent additional equipment, like navigation, child seat, ski rack, etc.
12. Members can add additional services to their reservation, such as roadside assistance, additional driver, wifi, etc.

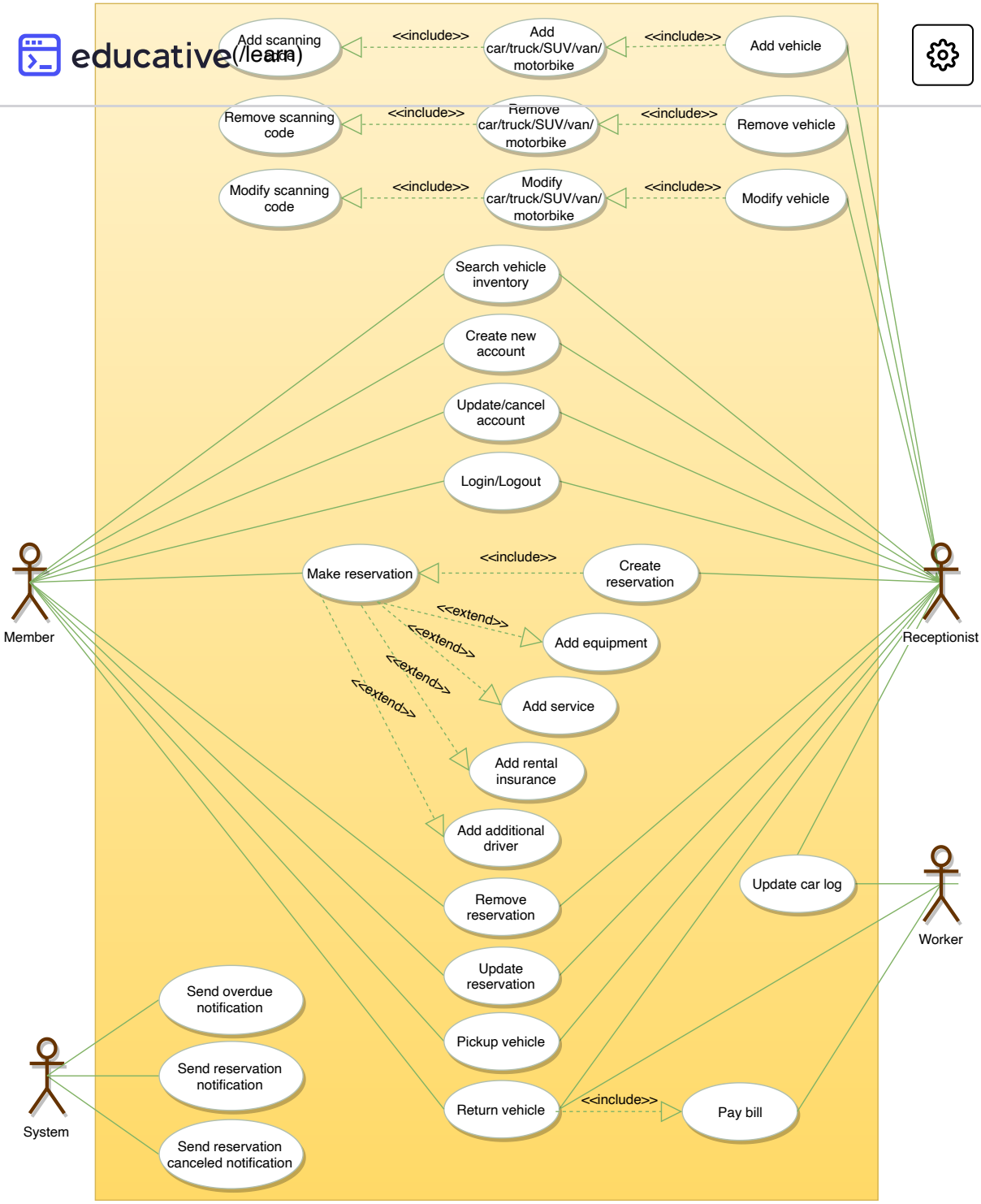
Use case diagram

We have four main Actors in our system:

- **Receptionist:** Mainly responsible for adding and modifying vehicles and workers. Receptionists can also reserve vehicles.
- **Member:** All members can search the catalog, as well as reserve, pick-up, and return a vehicle.
- **System:** Mainly responsible for sending notifications about overdue vehicles, canceled reservation, etc.
- **Worker:** Mainly responsible for taking care of a returned vehicle and updating the vehicle log.

Here are the top use cases of the Car Rental System:

- **Add/Remove/Edit vehicle:** To add, remove or modify a vehicle.
- **Search catalog:** To search for vehicles by type and availability.
- **Register new account/Cancel membership:** To add a new member or cancel an existing membership.
- **Reserve vehicle:** To reserve a vehicle.
- **Check-out vehicle:** To rent a vehicle.
- **Return a vehicle:** To return a vehicle which was checked-out to a member.
- **Add equipment:** To add an equipment to a reservation like navigation, child seat, etc.
- **Update car log:** To add or update a car log entry, such as refueling, cleaning, damage, etc.



Use case diagram

Class diagram

Here are the main classes of our Car Rental System:

- **CarRentalSystem:** The main part of the organization for which this software has been designed.

- **CarRentalLocation:** The car rental system will have multiple locations, each location will have attributes like ‘Name’ to distinguish it from any other locations and ‘Address’ which defines the address of the rental location.

- **Vehicle:** The basic building block of the system. Every vehicle will have a barcode, license plate number, passenger capacity, model, make, mileage, etc. Vehicles can be of multiple types, like car, truck, SUV, etc.

- **Account:** Mainly, we will have two types of accounts in the system, one will be a general member and the other will be a receptionist. Another account can be of the worker taking care of the returned vehicle.

- **VehicleReservation:** This class will be responsible for managing reservations for a vehicle.

- **Notification:** Will take care of sending notifications to members.

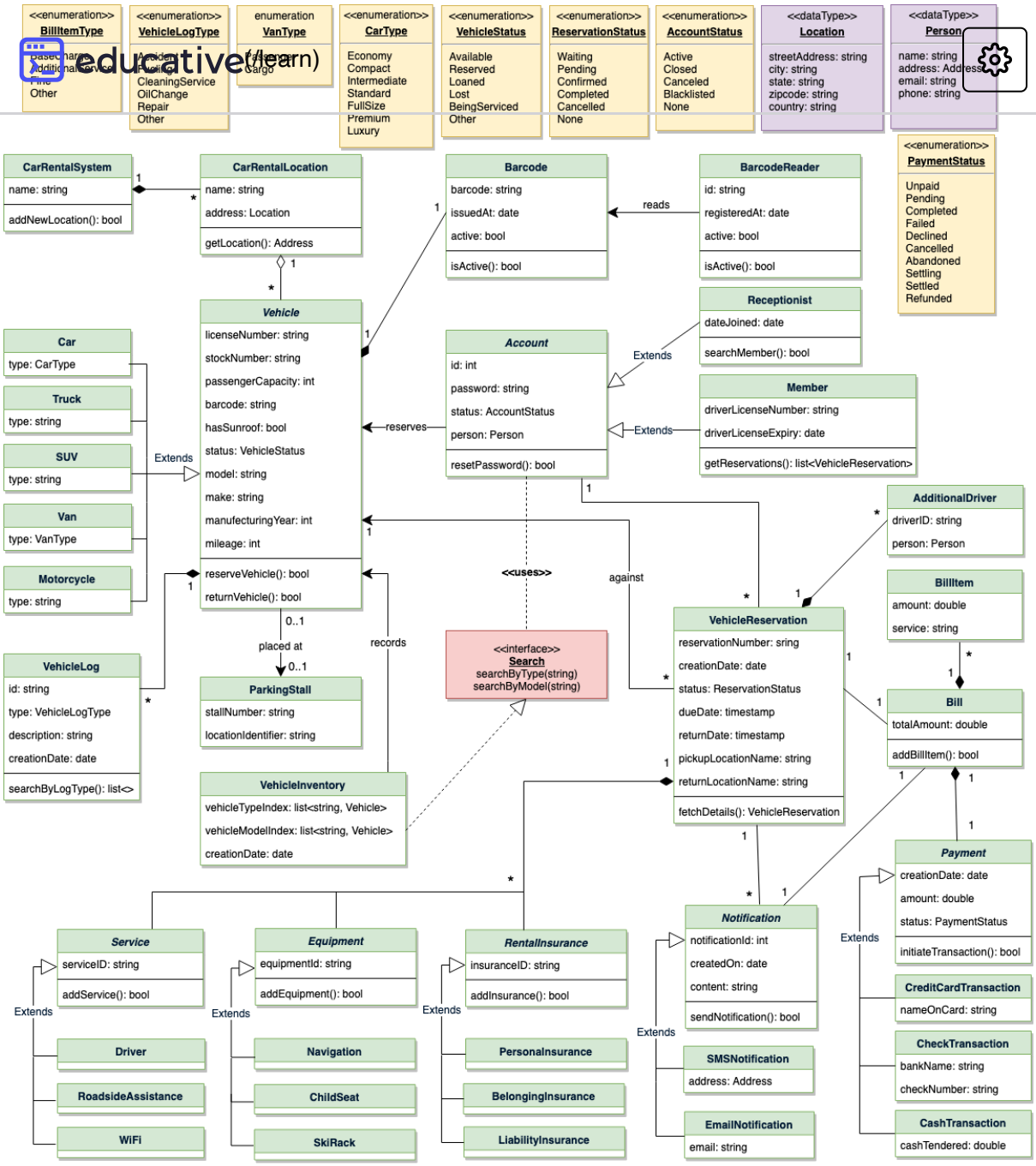
- **VehicleLog:** To keep track of all the events related to a vehicle.

- **RentalInsurance:** Stores details about the various rental insurances that members can add to their reservation.

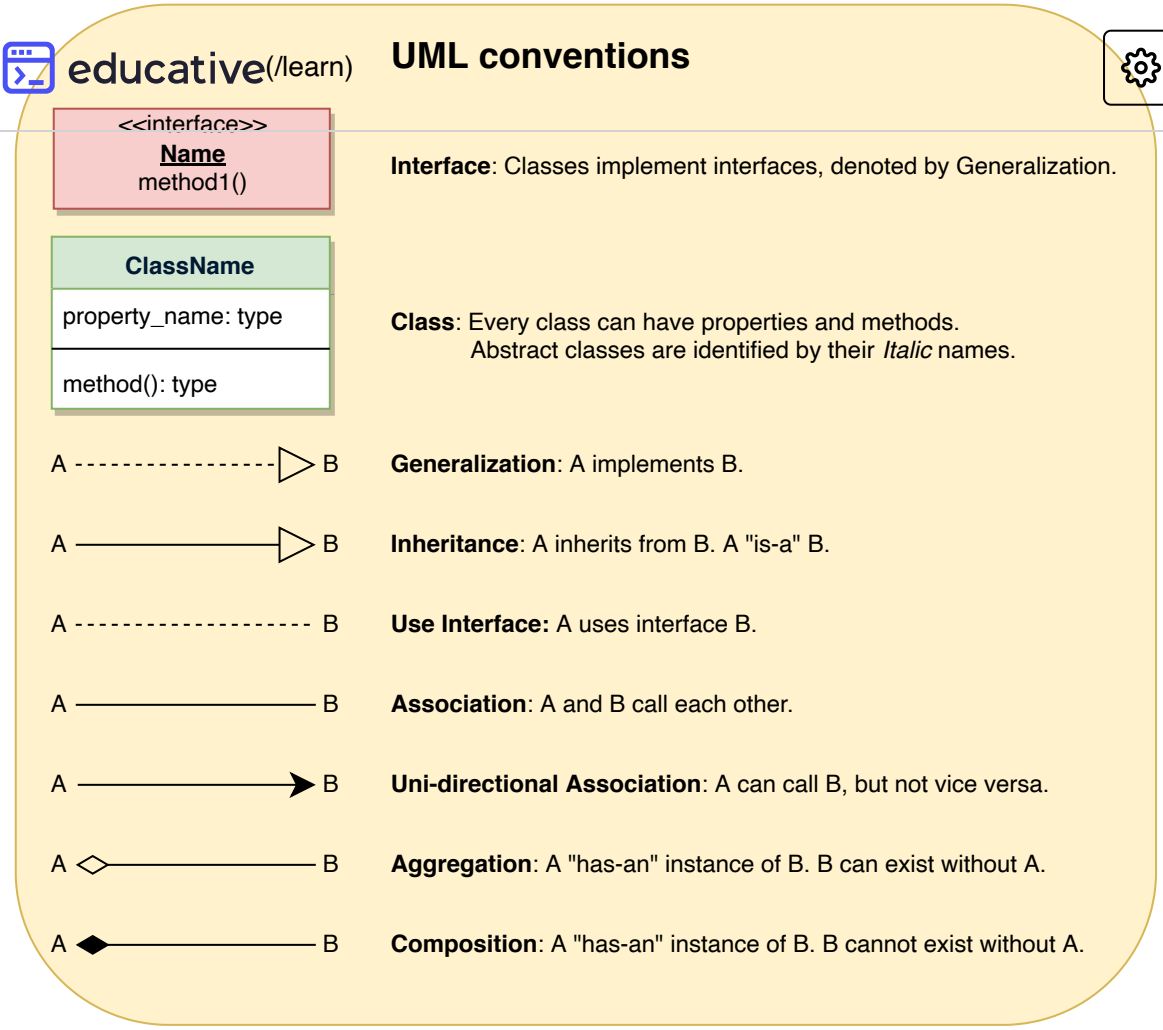
- **Equipment:** Stores details about the various types of equipment that members can add to their reservation.

- **Service:** Stores details about the various types of service that members can add to their reservation, such as additional drivers, roadside assistance, etc.

- **Bill:** Contains different bill-items for every charge for the reservation.

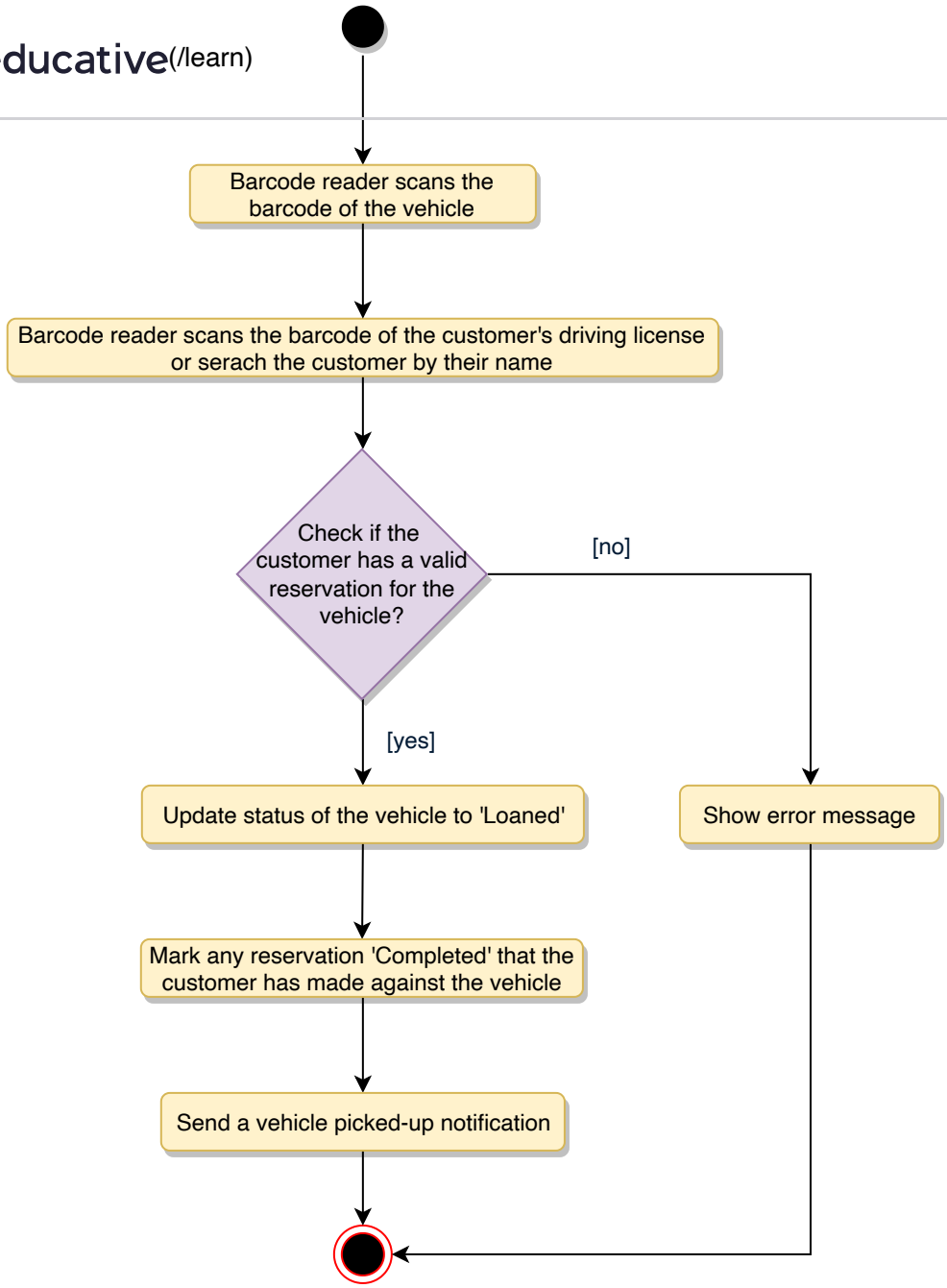


Class diagram

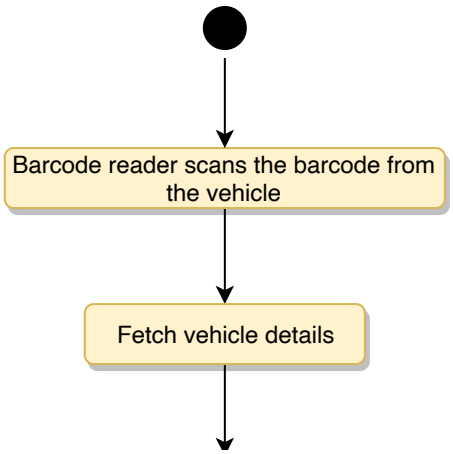


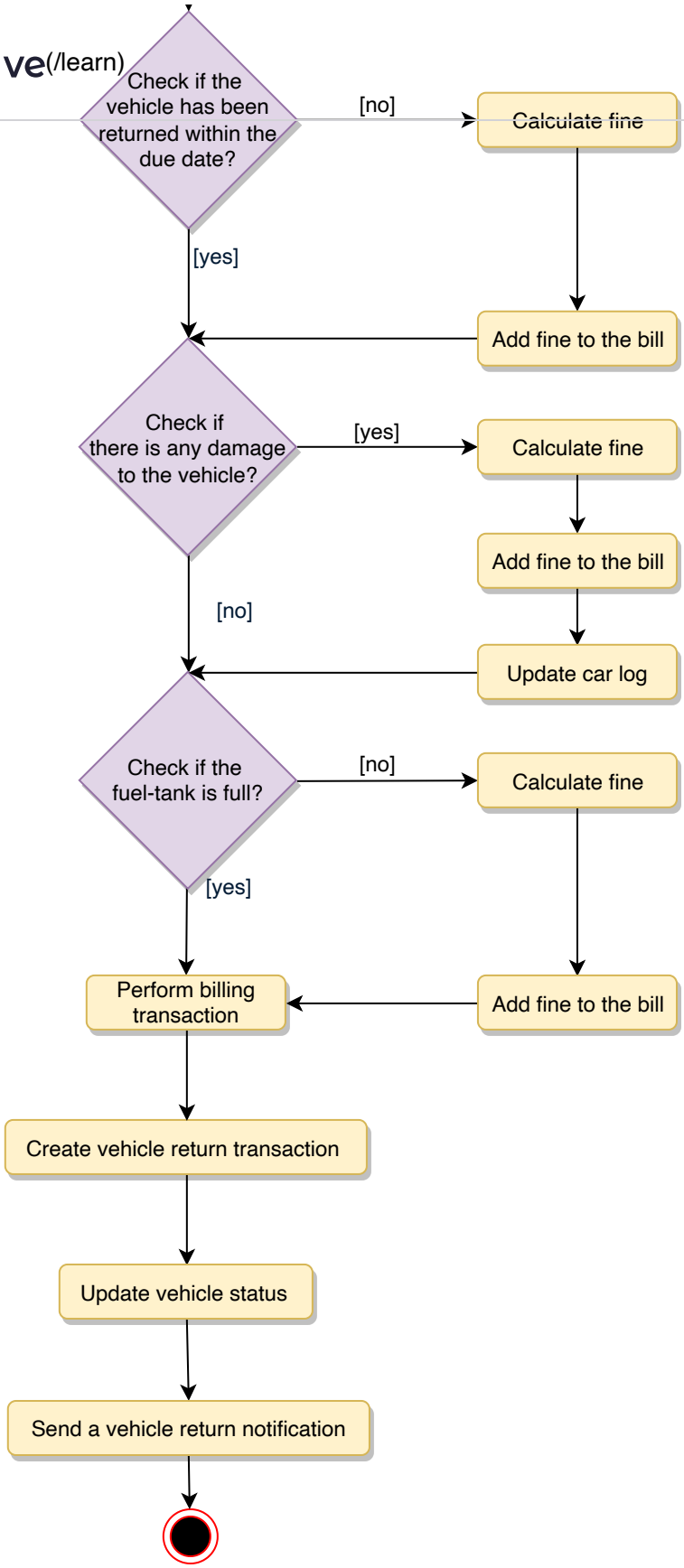
Activity diagrams

Pick up a vehicle: Any member can perform this activity. Here are the steps to pick up a vehicle:



Return a vehicle: Any worker can perform this activity. While returning a vehicle, the system must collect a late fee from the member if the return date is after the due date. Here are the steps for returning a vehicle:





Code

Here is the high-level definition for the classes described above.

Enums, data types and constants: Here are the required enums, data types, and constants:

Java | Python

```
public enum BillItemType {
  BASE_CHARGE, ADDITIONAL_SERVICE, FINE, OTHER
}

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

public enum VanType {
  PASSENGER, CARGO
}

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

public enum VehicleStatus {
  AVAILABLE, RESERVED, LOANED, LOST, BEING_SERVICED, OTHER
}

public enum ReservationStatus {
  ACTIVE, PENDING, CONFIRMED, COMPLETED, CANCELLED, NONE
}

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

public enum PaymentStatus {
  UNPAID, PENDING, COMPLETED, FILLED, DECLINED, CANCELLED, ABANDONED, SETTL
}

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

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

Account, Member, Receptionist, and Additional Driver: These classes represent different people that interact with our system:

Java | Python

```
// For simplicity, we are not defining getter and setter functions. The rea
// assume that all class attributes are private and accessed through their
// public getter method and modified only through their public setter metho

public abstract class Account {
  private String id;
  private String password;
  private AccountStatus status;
  private Person person;

  public boolean resetPassword();
}

public class Member extends Account {
  private int totalVehiclesReserved;

  public List<VehicleReservation> getReservations();
}

public class Receptionist extends Account {
  private Date dateJoined;

  public List<Member> searchMember(String name);
}

public class AdditionalDriver {
  private String driverID;
  private Person person;
}
```

CarRentalSystem and CarRentalLocation: These classes represent the top level classes:

Java | Python

```
public class CarRentalLocation {
  private String name;
  private Address location;

  public Address getLocation();
}

public class CarRentalSystem {
  private String name;
  private List<CarRentalLocation> locations;

  public boolean addNewLocation(CarRentalLocation location);
}
```

Vehicle, VehicleLog, and VehicleReservation: To encapsulate a vehicle, log, and reservation. The VehicleReservation class will be responsible for processing the reservation and return of a vehicle:

Java | Python

```
public abstract class Vehicle {
  private String licenseNumber;
  private String stockNumber;
  private int passengerCapacity;
  private String barcode;
  private boolean hasSunroof;
  private VehicleStatus status;
  private String model;
  private String make;
  private int manufacturingYear;
  private int mileage;

  private List<VehicleLog> log;

  public boolean reserveVehicle();

  public boolean returnVehicle();
}

public class Car extends Vehicle {
  private CarType type;
}

public class Van extends Vehicle {
  private VanType type;
}

public class Truck extends Vehicle {
  private String type;
}

// We can have similar definition for other vehicle types

//...

public class VehicleLog {
  private String id;
  private VehicleLogType type;
  private String description;
  private Date creationDate;

  public bool update();

  public List<VehicleLogType> searchByLogType(VehicleLogType type);
}

public class VehicleReservation {
  private String reservationNumber;
  private Date creationDate;
  private ReservationStatus status;
  private Date dueDate;
  private Date returnDate;
  private String pickupLocationName;
  private String returnLocationName;
```

```
  private int customerID;
  private Vehicle vehicle;
  private Bill bill;
  private List<AdditionalDriver> additionalDrivers;
  private List<Notification> notifications;
  private List<RentalInsurance> insurances;
  private List<Equipment> equipments;
  private List<Service> services;

  public static VehicleReservation fetchReservationDetails(String reservati

  public List<Passenger> getAdditionalDrivers();
}
```

VehicleInventory and Search: VehicleInventory will implement an interface ‘Search’ to facilitate the searching of vehicles:

Java | Python

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

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

  public List<Vehicle> searchByType(String query) {
    // return all vehicles of the given type.
    return vehicleTypes.get(query);
  }

  public List<Vehicle> searchByModel(String query) {
    // return all vehicles of the given model.
    return vehicleModels.get(query);
  }
}
```

← Back (/courses/grokking-the-object-oriented-design-interview/RM8ol3m2o3w)
Design an Online Stock Brokerage Sy…

Mark as Completed

Next → (/courses/grokking-the-object-oriented-design-interview/gxMOMDVKlBk)
Design LinkedIn

Stuck? Get DISCUSS
Send
4