Design a Car Rental System

Let's design a Car Rental System

We'll cover the following:

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.

Car Rental System

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:

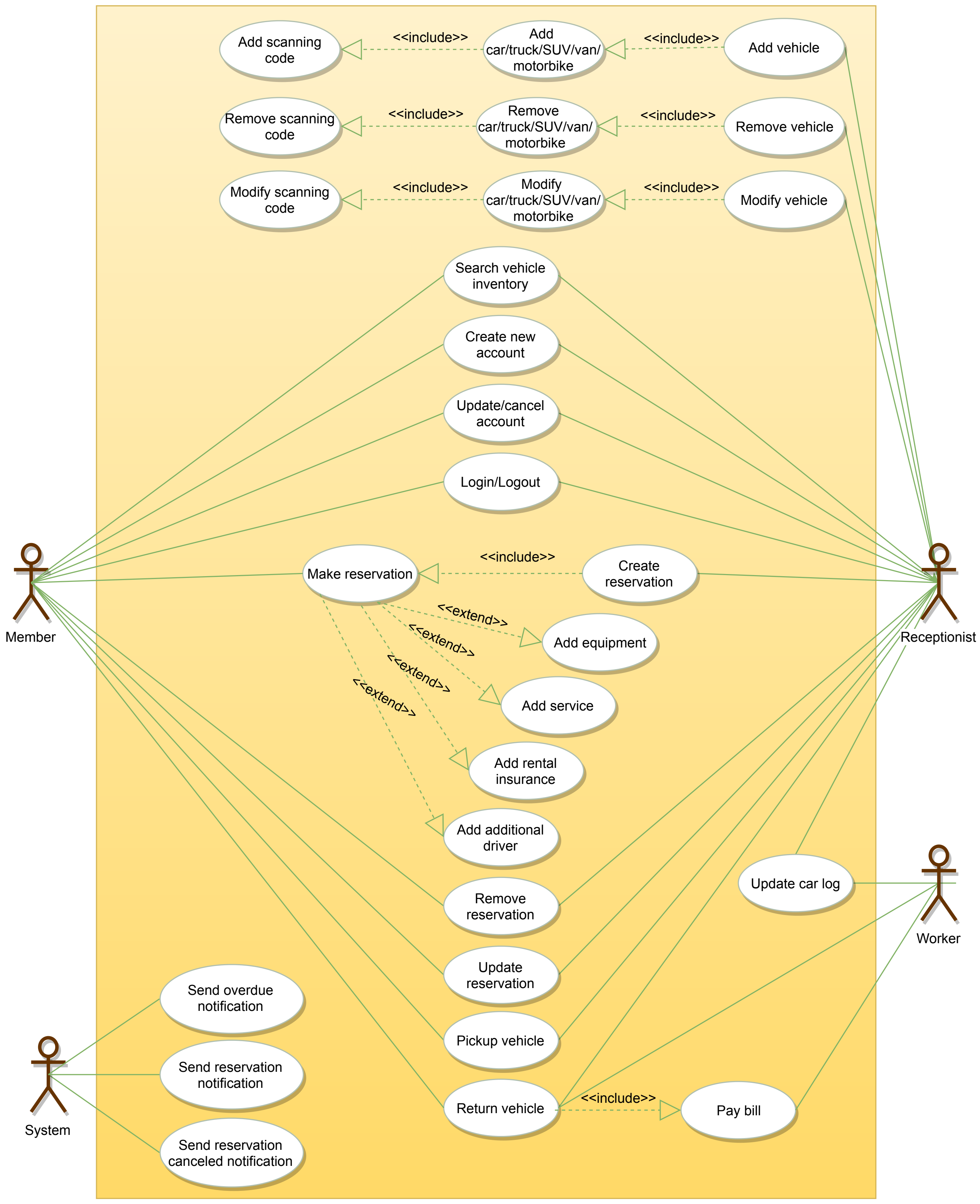
- **Receptionist:** Mainly responsible for adding and modifying vehicles and workers. Receptionists can also reserve vehicles.
- **Member:** All members can search the catalog, as well as reserve, pick-up, and return a vehicle.
- **System:** Mainly responsible for sending notifications about overdue 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.

Here is the use case diagram of our Car Rental System:

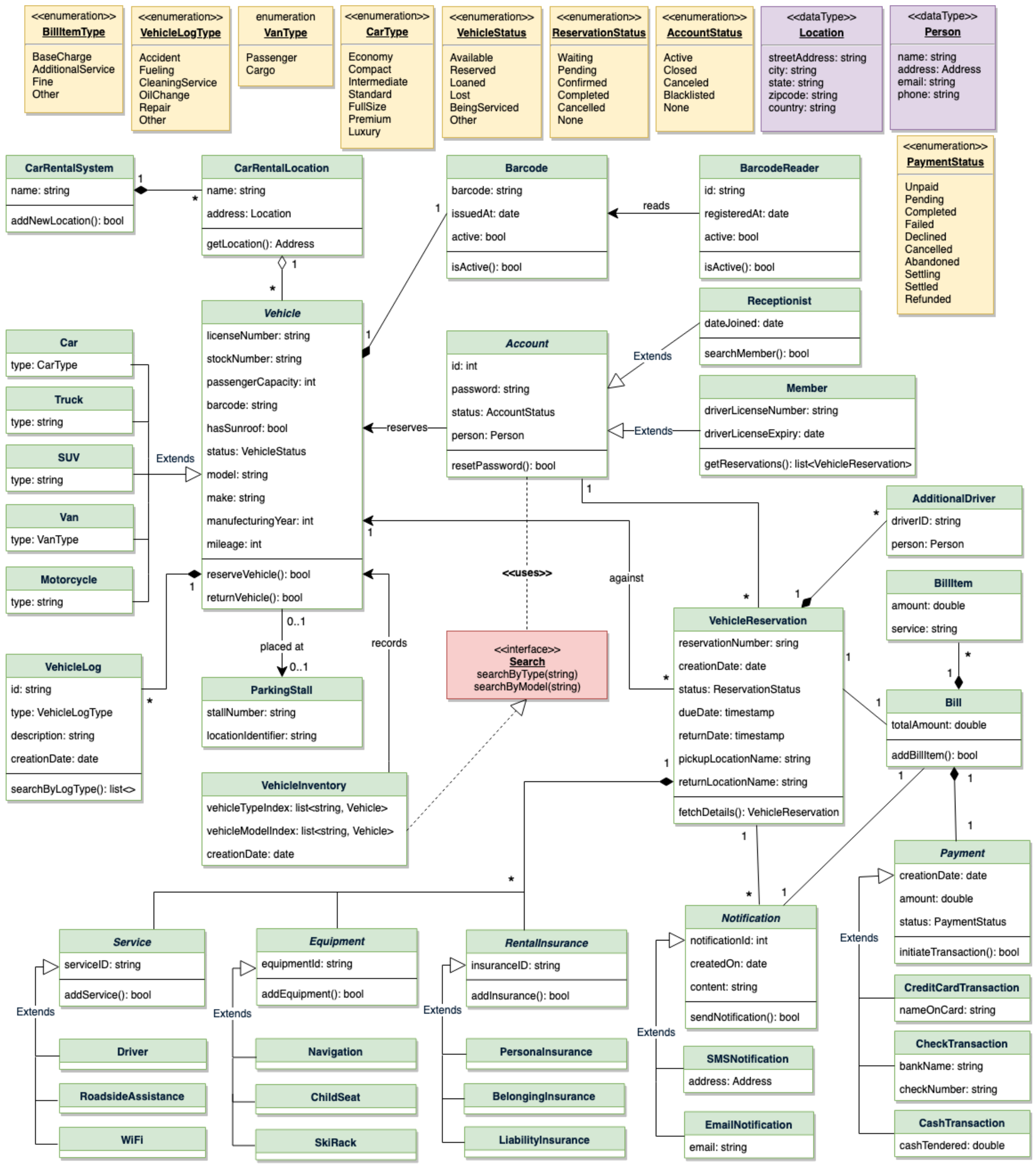


Use Case Diagram for Car Rental System

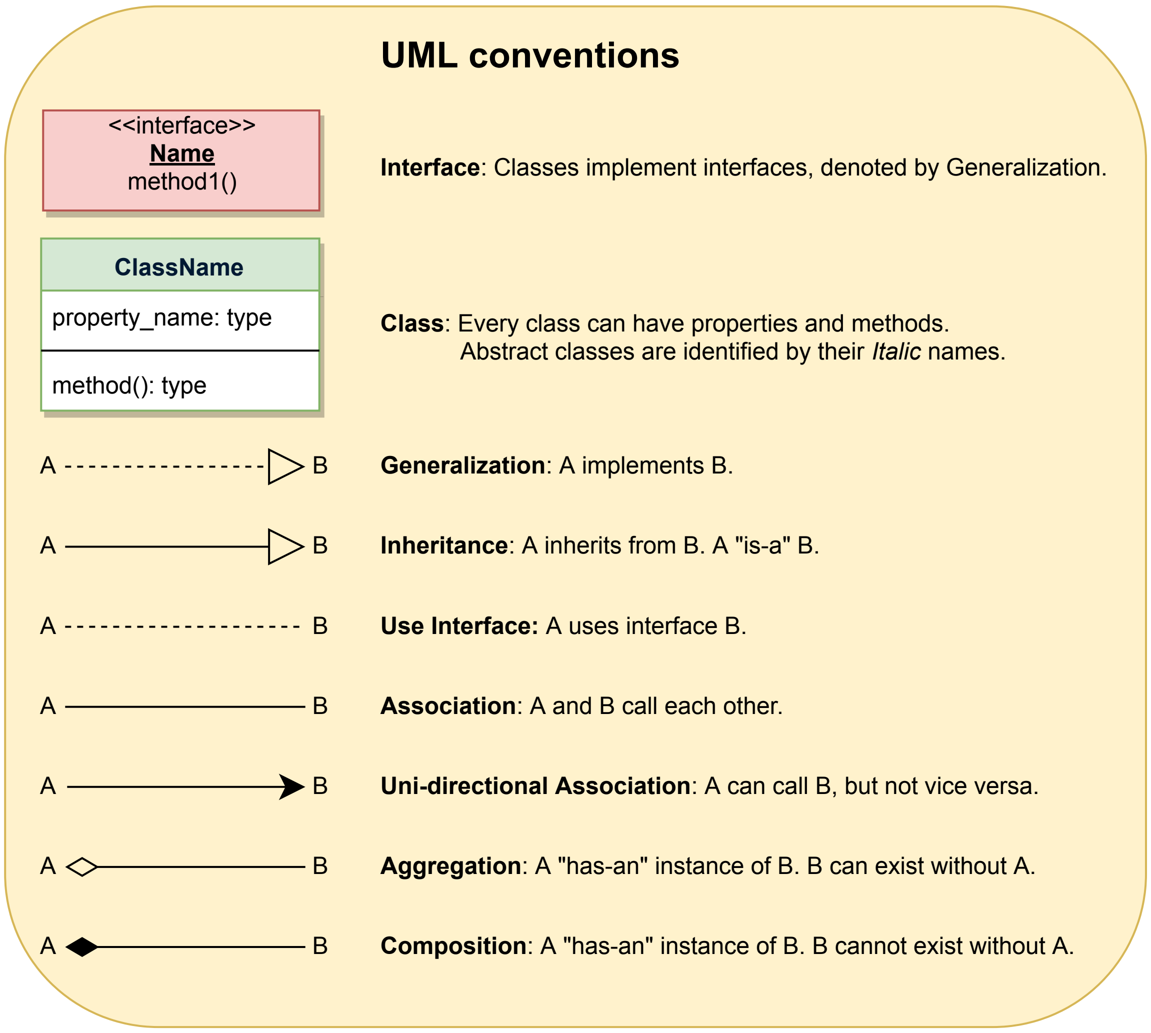
Class Diagram

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.



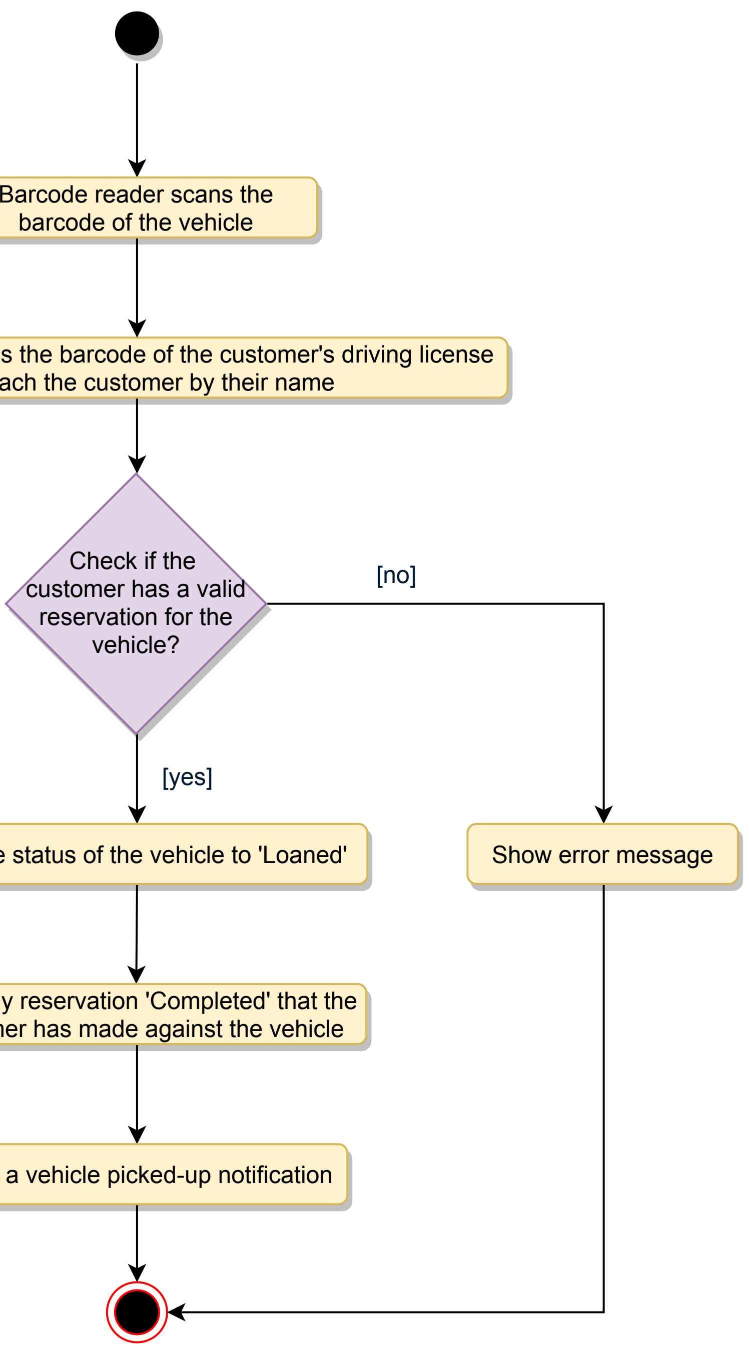
Class Diagram for Car Rental System


UML conventions
<<interface>>
Name
method1()
Interface: Classes implement interfaces, denoted by Generalization.
ClassName
property_name: type
method(): type
Class: Every class can have properties and methods. Abstract classes are identified by their Italic names.
A B
Generalization: A implements B.
A B
Inheritance: A inherits from B. A "is-a" B.
A B
Use Interface: A uses interface B.
A B
Association: A and B call each other.
A B
Uni-directional Association: A can call B, but not vice versa.
A B
Aggregation: A "has-an" instance of B. B can exist without A.
A B
Composition: A "has-an" instance of B. B cannot exist without A.

UML for Car Rental System

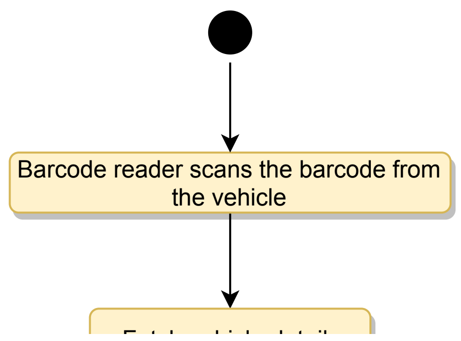
Activity Diagrams

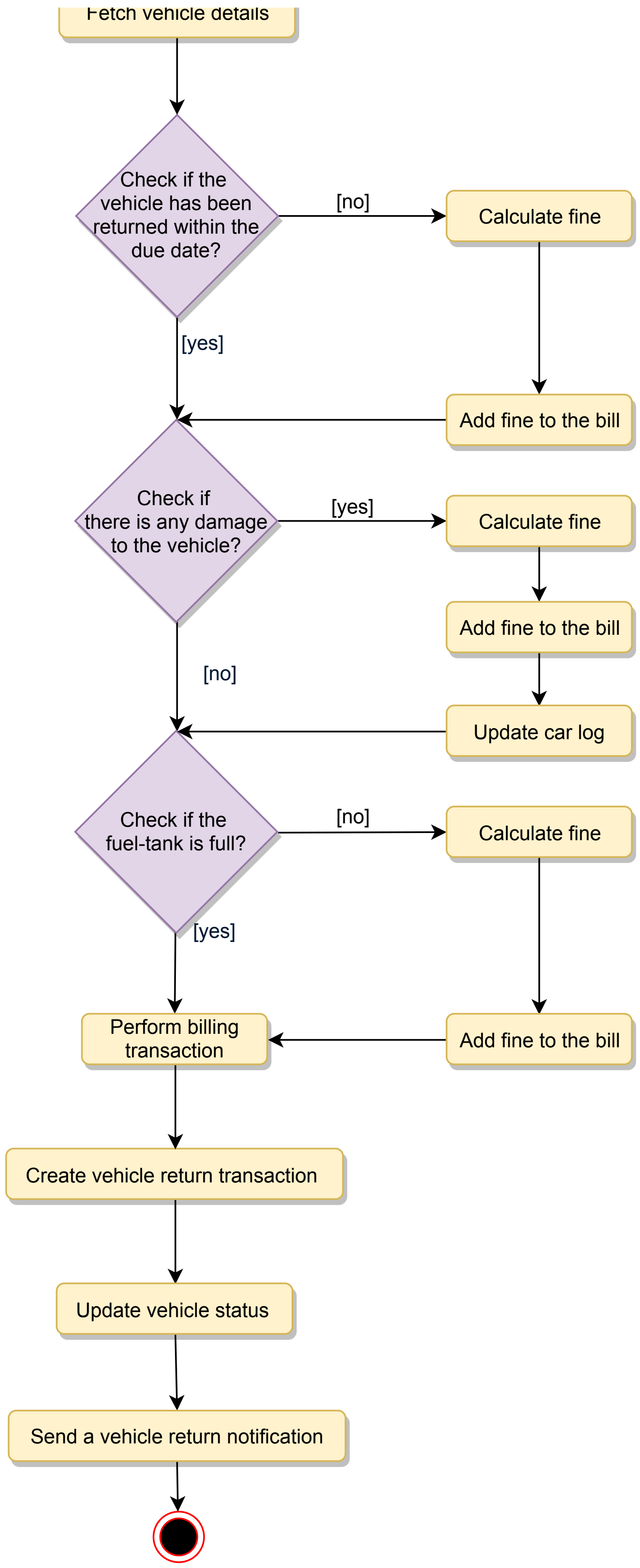
Pick up a vehicle: Any member can perform this activity. Here are the steps to pick up a vehicle:


Barcode reader scans the barcode of the vehicle
Barcode reader scans the barcode of the customer's driving license or serach the customer by their name
Check if the customer has a valid reservation for the vehicle?
[no]
[yes]
Update status of the vehicle to 'Loaned'
Show error message
Mark any reservation 'Completed' that the customer has made against the vehicle
Send a vehicle picked-up notification

Activity Diagram for Car Rental System Pick Up

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:





Fetch vehicle details
Check if the vehicle has been returned within the due date?
[no]
Calculate fine
[yes]
Add fine to the bill
Check if there is any damage to the vehicle?
[yes]
Calculate fine
Add fine to the bill
[no]
Update car log
Check if the fuel-tank is full?
[no]
Calculate fine
[yes]
Perform billing transaction
Add fine to the bill
Create vehicle return transaction
Update vehicle status
Send a vehicle return notification

Activity Diagram for Car Rental System Return

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:

```python
from enum import Enum


class BillItemType(Enum):
    BASE_CHARGE, ADDITIONAL_SERVICE, FINE, OTHER = 1, 2, 3, 4


class VehicleLogType(Enum):
    ACCIDENT, FUELING, CLEANING_SERVICE, OIL_CHANGE, REPAIR, OTHER = 1, 2, 3,
4, 5, 6


class VanType(Enum):
    PASSENGER, CARGO = 1, 2


class CarType(Enum):
    ECONOMY, COMPACT, INTERMEDIATE, STANDARD, FULL_SIZE, PREMIUM, LUXURY = 1,
2, 3, 4, 5, 6, 7


class VehicleStatus(Enum):
    AVAILABLE, RESERVED, LOANED, LOST, BEING_SERVICED, OTHER = 1, 2, 3, 4, 5, 6


class ReservationStatus(Enum):
    ACTIVE, PENDING, CONFIRMED, COMPLETED, CANCELLED, NONE = 1, 2, 3, 4, 5, 6


class AccountStatus(Enum):
    ACTIVE, CLOSED, CANCELED, BLACKLISTED, BLOCKED = 1, 2, 3, 4, 5


class PaymentStatus(Enum):
    UNPAID, PENDING, COMPLETED, FILLED, DECLINED, CANCELLED, ABANDONED,
SETTLING, SETTLED, REFUNDED = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10


class Address:
    def __init__(self, street, city, state, zip_code, country):
        self.__street_address = street
        self.__city = city
        self.__state = state
        self.__zip_code = zip_code
        self.__country = country
```

```python
class Person():
    def __init__(self, name, address, email, phone):
        self.__name = name
        self.__address = address
        self.__email = email
        self.__phone = phone
```

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

```python
from abc import ABC
from .constants import AccountStatus


# For simplicity, we are not defining getter and setter functions. The reader
can
# assume that all class attributes are private and accessed through their
respective
# public getter methods and modified only through their public methods
function.


class Account(ABC):
    def __init__(self, id, password, person, status=AccountStatus.NONE):
        self.__id = id
        self.__password = password
        self.__status = AccountStatus.NONE
        self.__person = person

    def reset_password(self):
        None


class Member(Account):
    def __init__(self):
        self.__total_vehicles_reserved = 0

    def get_reservations(self):
        None


class Receptionist(Account):
    def __init__(self, date_joined):
        self.__date_joined = date_joined

    def search_member(self, name):
        None


class AdditionalDriver:
    def __init__(self, id, person):
        self.__driver_id = id
        self.__person = person
```

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

```python
class CarRentalLocation:
    def __init__(self, name, address):
        self.__name = name
        self.__location = address

    def get_location(self):
        return self.__location


class CarRentalSystem:
    def __init__(self, name):
        self.__name = name
        self.__locations = []

    def add_new_location(self, location):
        None
```

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:

```python
from abc import ABC
from datetime import datetime
from .constants import ReservationStatus


class Vehicle(ABC):
    def __init__(self, license_num, stock_num, capacity, barcode, has_sunroof,
status, model, make, manufacturing_year,
                 mileage):
        self.__license_number = license_num
        self.__stock_number = stock_num
        self.__passenger_capacity = capacity
        self.__barcode = barcode
        self.__has_sunroof = has_sunroof
        self.__status = status
        self.__model = model
        self.__make = make
        self.__manufacturing_year = manufacturing_year
        self.__mileage = mileage
        self.__log = []

    def reserve_vehicle(self):
        None

    def return_vehicle(self):
        None


class Car(Vehicle):
    def __init__(self, license_num, stock_num, capacity, barcode, has_sunroof,
status, model, make, manufacturing_year,
                 mileage, type):
        super().__init__(license_num, stock_num, capacity, barcode,
                         has_sunroof, status, model, make, manufacturing_year,
mileage)
        self.__type = type


class Van(Vehicle):
    def __init__(self, license_num, stock_num, capacity, barcode, has_sunroof,
status, model, make, manufacturing_year,
                 mileage, type):
        super().__init__(license_num, stock_num, capacity, barcode,
                         has_sunroof, status, model, make, manufacturing_year,
mileage)
        self.__type = type
```

```
class Truck(Vehicle):
    def __init__(self, license_num, stock_num, capacity, barcode, has_sunroof,
status, model, make, manufacturing_year,
                 mileage, type):
        super().__init__(license_num, stock_num, capacity, barcode,
                         has_sunroof, status, model, make, manufacturing_year,
mileage)
        self.__type = type


# We can have similar definition for other vehicle types

# ...

class VehicleLog:
    def __init__(self, id, type, description, creation_date):
        self.__id = id
        self.__type = type
        self.__description = description
        self.__creation_date = creation_date

    def update(self):
        None

    def search_by_log_type(self, type):
        None


class VehicleReservation:
    def __init__(self, reservation_number):
        self.__reservation_number = reservation_number
        self.__creation_date = datetime.date.today()
        self.__status = ReservationStatus.ACTIVE
        self.__due_date = datetime.date.today()
        self.__return_date = datetime.date.today()
        self.__pickup_location_name = ""
        self.__return_location_name = ""

        self.__customer_id = 0
        self.__vehicle = None
        self.__bill = None
        self.__additional_drivers = []
        self.__notifications = []
        self.__insurances = []
        self.__equipments = []
        self.__services = []
    def fetch_reservation_details(self, reservation_number):
        None

    def get_additional_drivers(self):
        return self.__additional_drivers
```

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

```
from abc import ABC


class Search(ABC):
    def search_by_type(self, type):
        None

    def search_by_model(self, model):
        None


class VehicleInventory(Search):
    def __init__(self):
        self.__vehicle_types = {}
        self.__vehicle_models = {}

    def search_by_type(self, query):
        # return all vehicles of the given type.
        return self.__vehicle_types.get(query)

    def search_by_model(self, query):
        # return all vehicles of the given model.
        return self.__vehicle_models.get(query)
```