Tutorial 4 Feedback
(Domain Model Class Diagram)

Some pointers

- How do we assess whether a design if correct?
  - Is it **able to model the various real-world relationship**?
    e.g. in this tutorial, do you know for each Trip:
    - Who is the Driver? Which vehicle is used? Who are the passengers?

  - The exact association also **depends on the multiplicity of the associations**
    - e.g. if each Driver drives only 1 vehicle, there's no need to associate Trip with the Vehicle (implicit understanding is that each Driver drives a specific Vehicle).
    - Conversely, if each Driver drives more than 1 vehicle, we must also associate the Trip with the Vehicle (since need to capture which vehicle is used for the Trip)

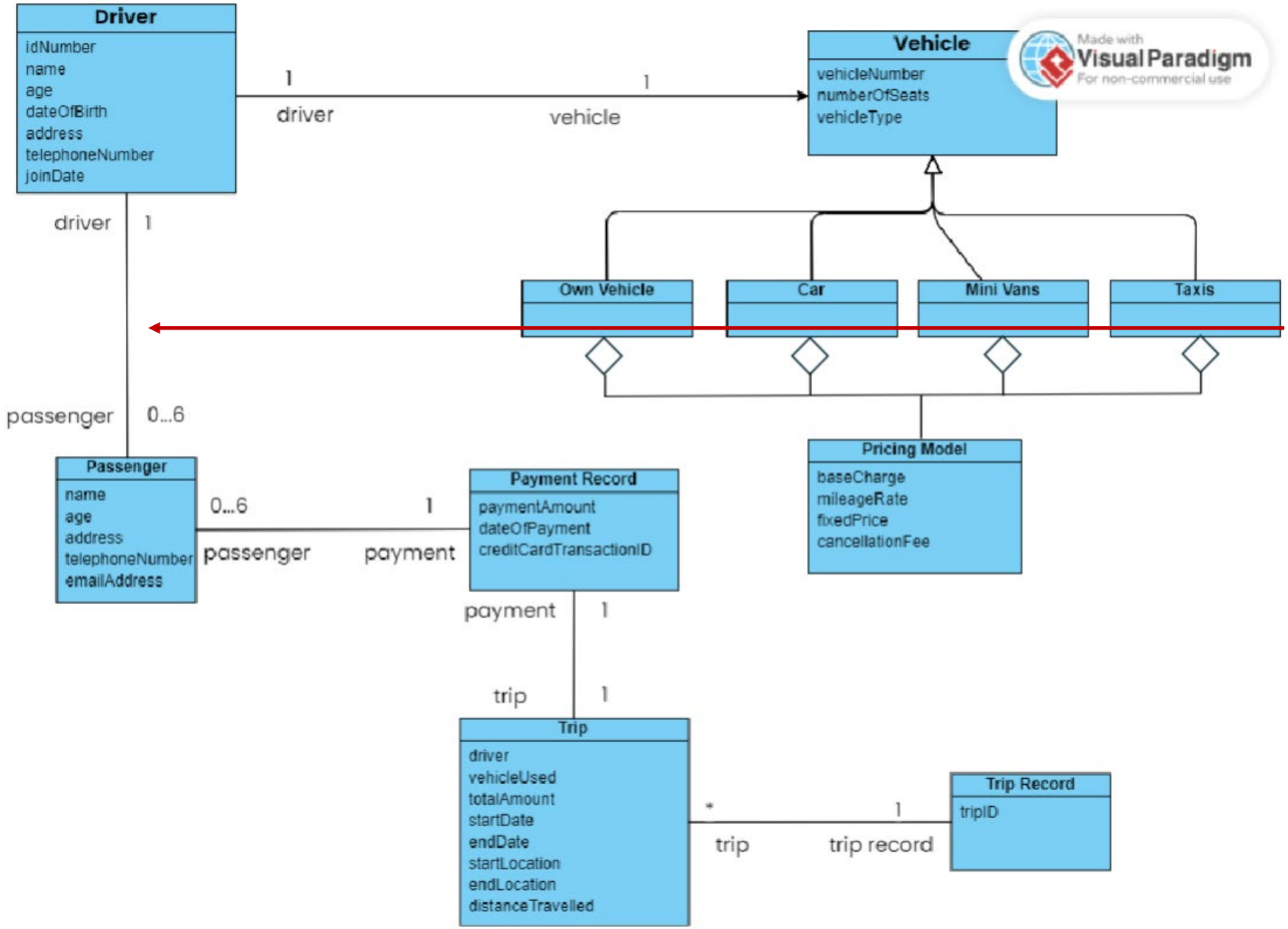
Some pointers

- Notations
  - Make sure attribute names, class names are proper identifier names (no spaces, “/” symbol, etc)
  - Make sure you include the rolenames (they are really just attribute names of the other class type)
  - Make sure every entity should have a primary key (that uniquely identify each entity)
  - Try to be consistent in the naming (singular form – “Driver” instead of “Drivers”)

Some pointers

- Other advice
  - If softcopy, use a drawing software
  - Avoid putting the association too close to each other (it’s hard to tell with absolute certainty the rolename/multiplicity for each association)

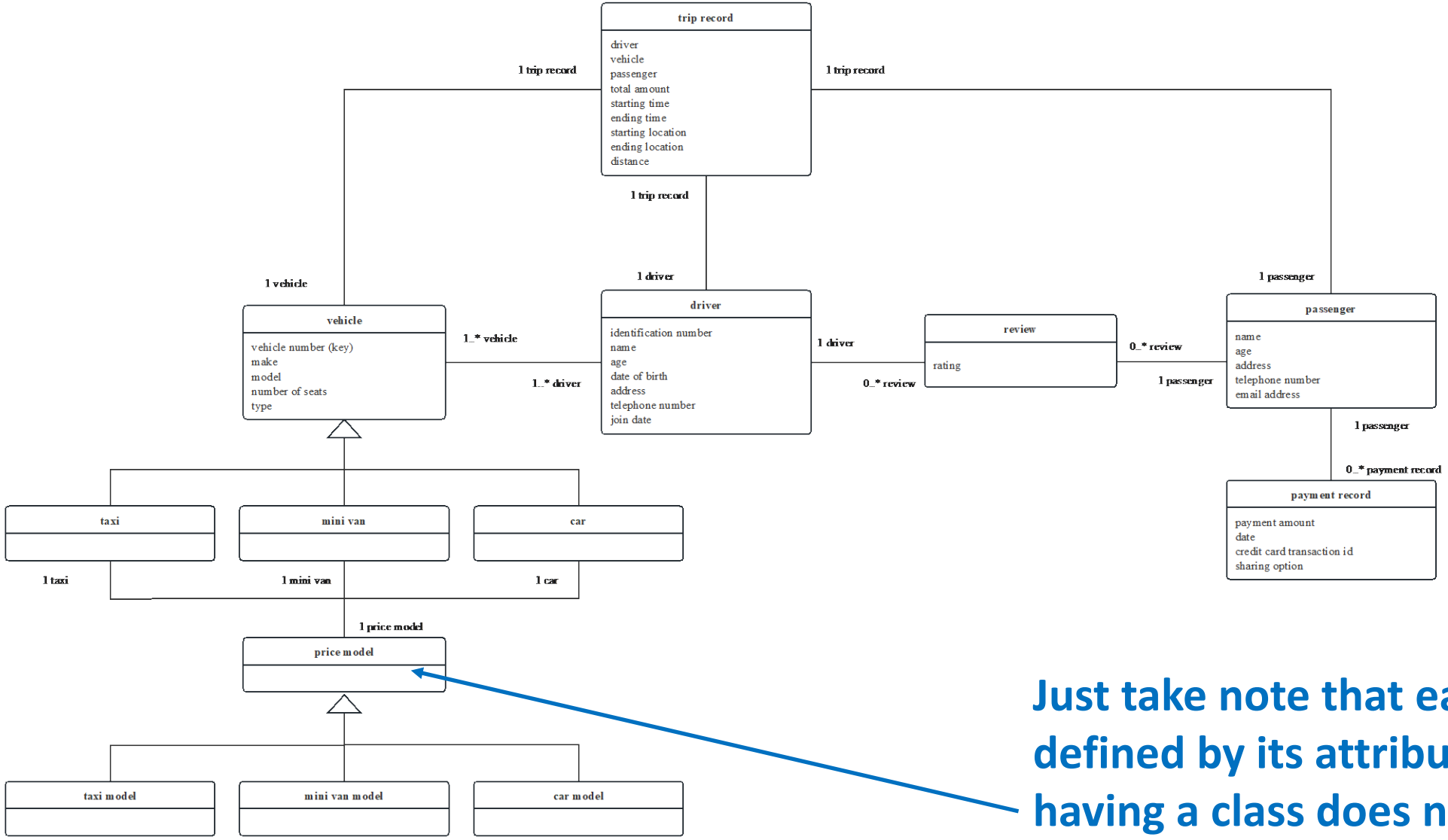
Ideally, the design should **reflect the real-world relationship** not just want to need to achieve now



When designing the entities, you should design it to reflect the real-world relationship, currently, this is saying:

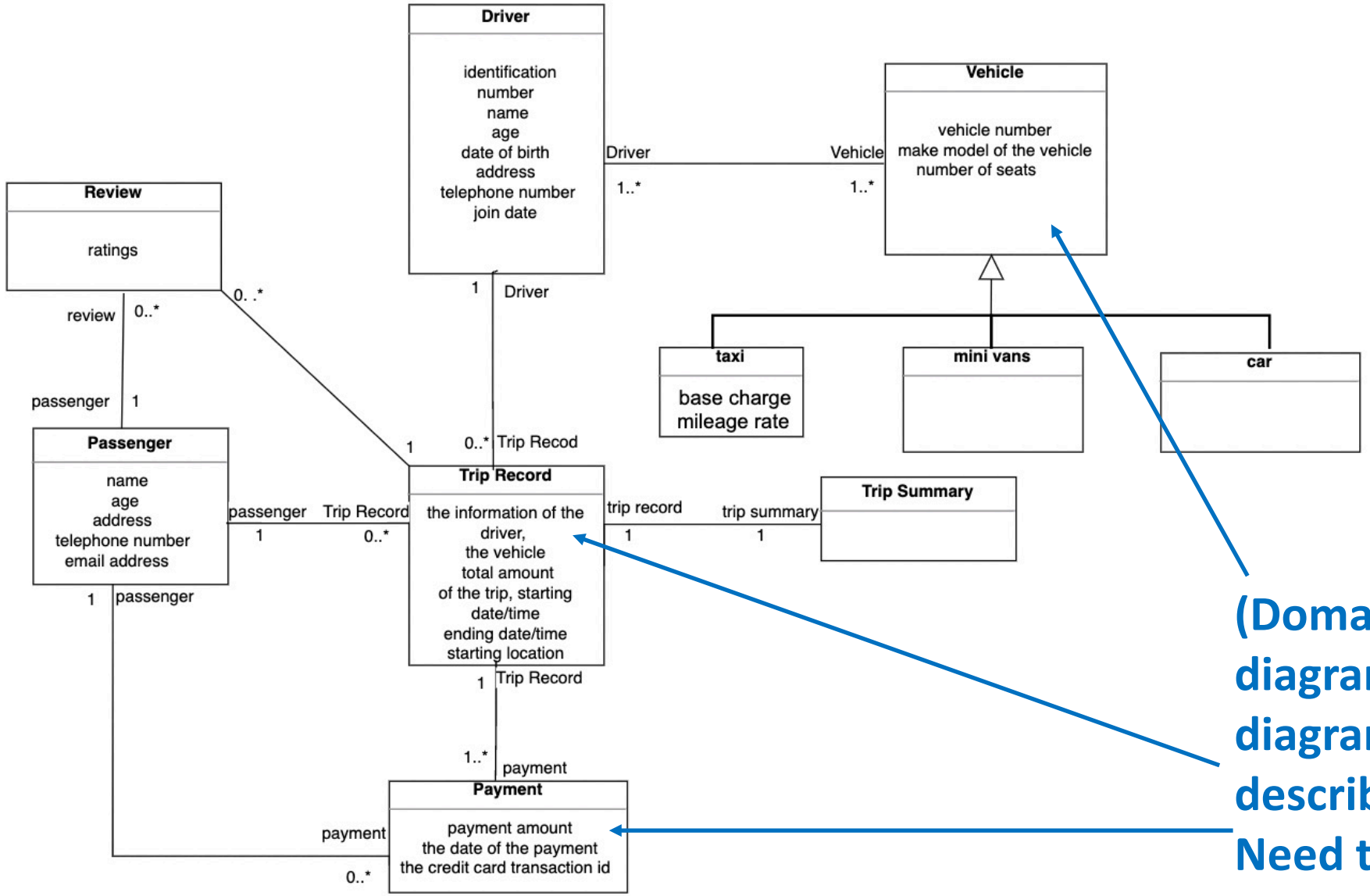
"1 Driver would have 0 to 6 passengers

but in real life, it's more of Passengers can book trips, in a trip there's a driver and a trip can have up to 6 passengers"

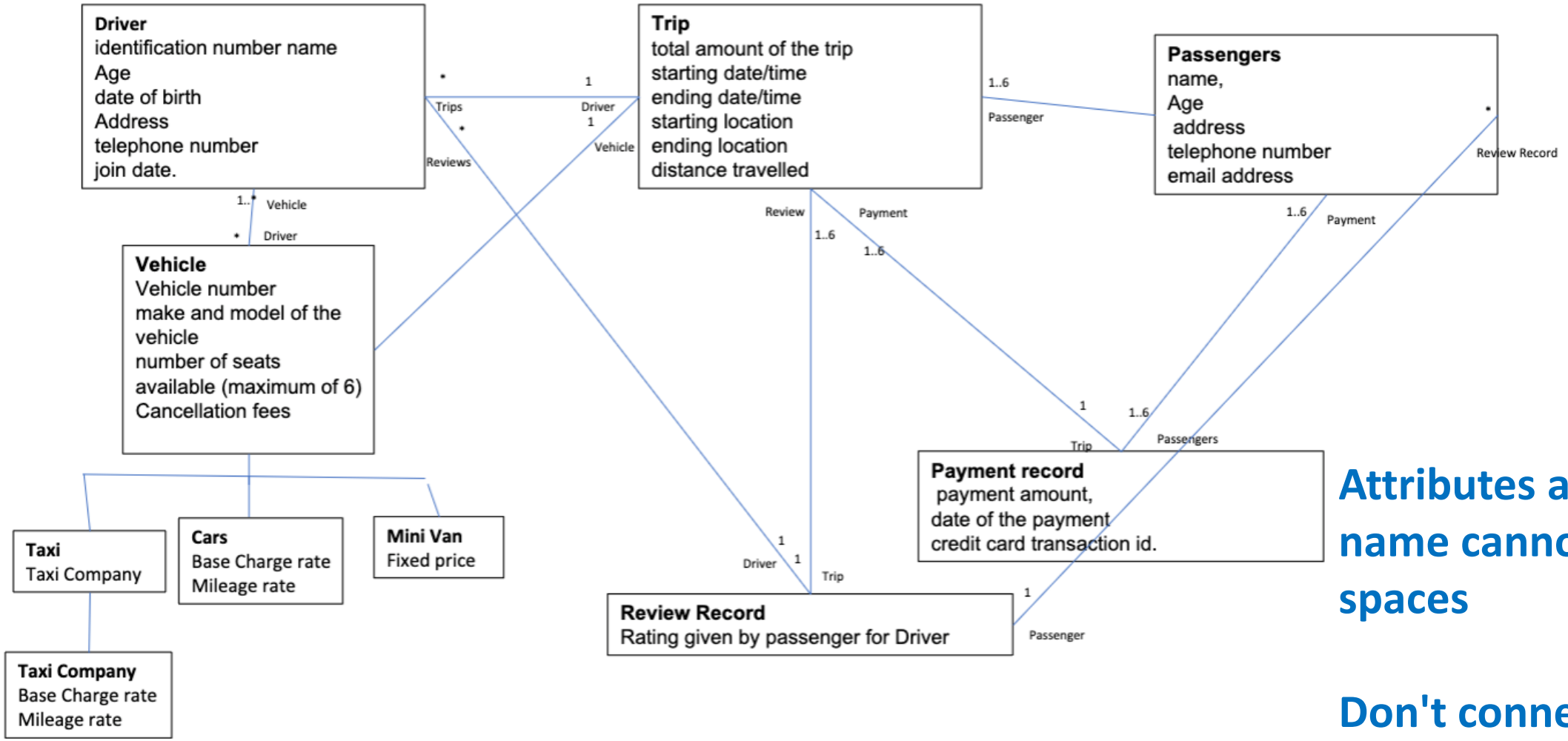


Just take note that each class is defined by its attribute. By just having a class does not automatically allow us to model the logic immediately

Domain Model Class Diagram is a Design diagram – **Notations are important** (describe how you will write the codes)

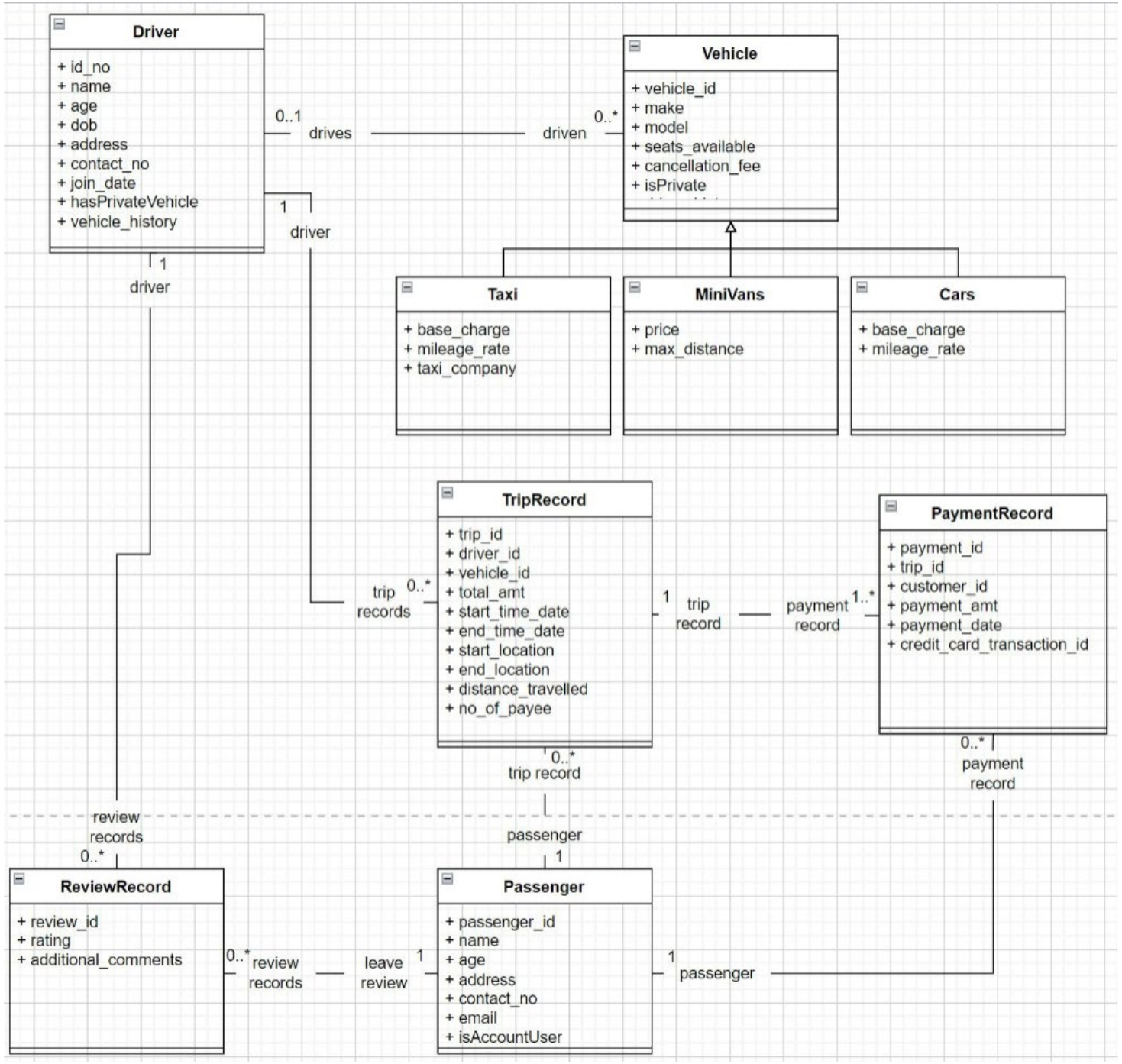


(Domain model) class diagram is a coding-specific diagram. We don't just describe the information. Need to put it down as attributes



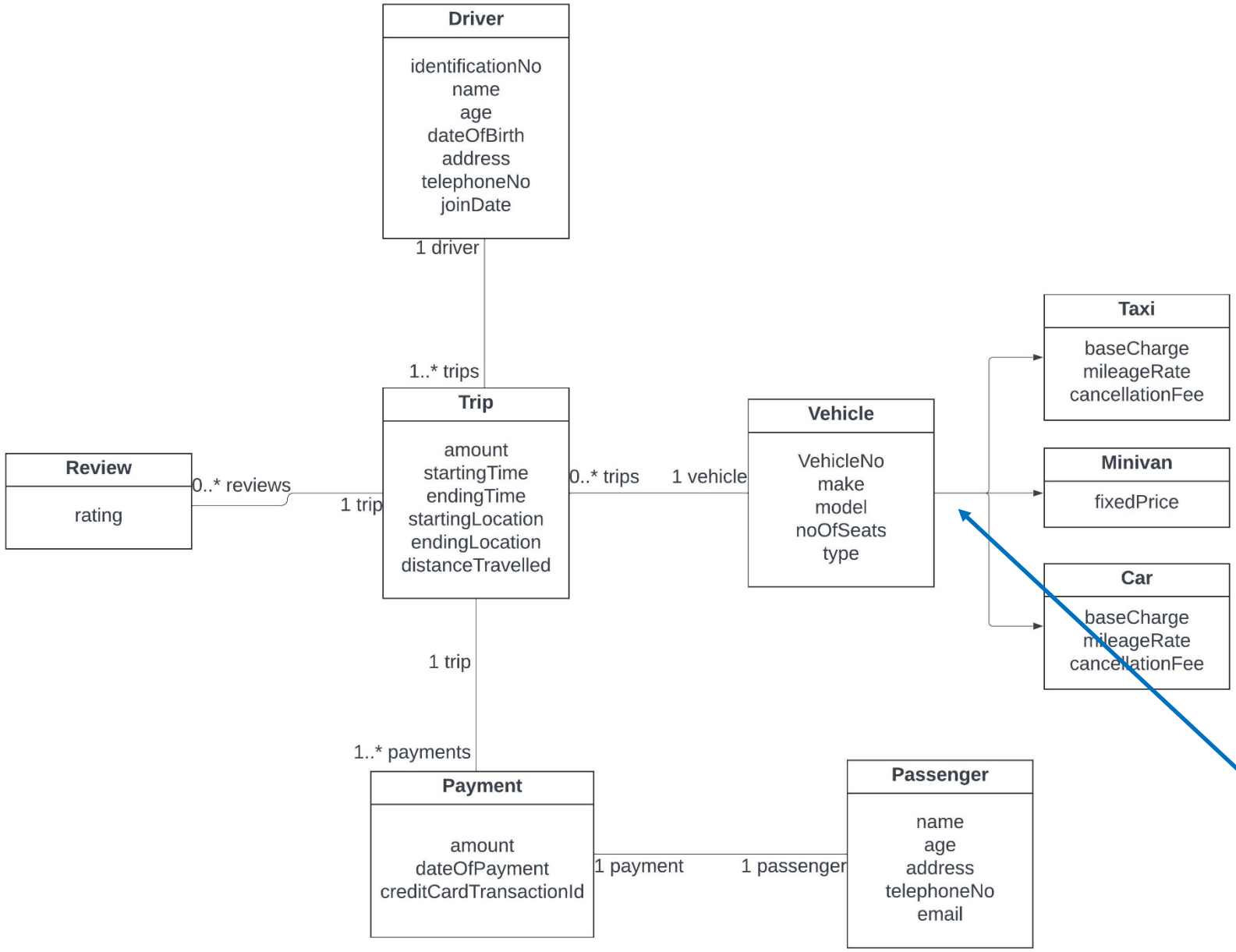
Attributes and class name cannot have spaces

Don't connect the associations so closely (it's hard to tell the multiplicity or rolename is for the associations)



Don’t use multiple lines for rolenames

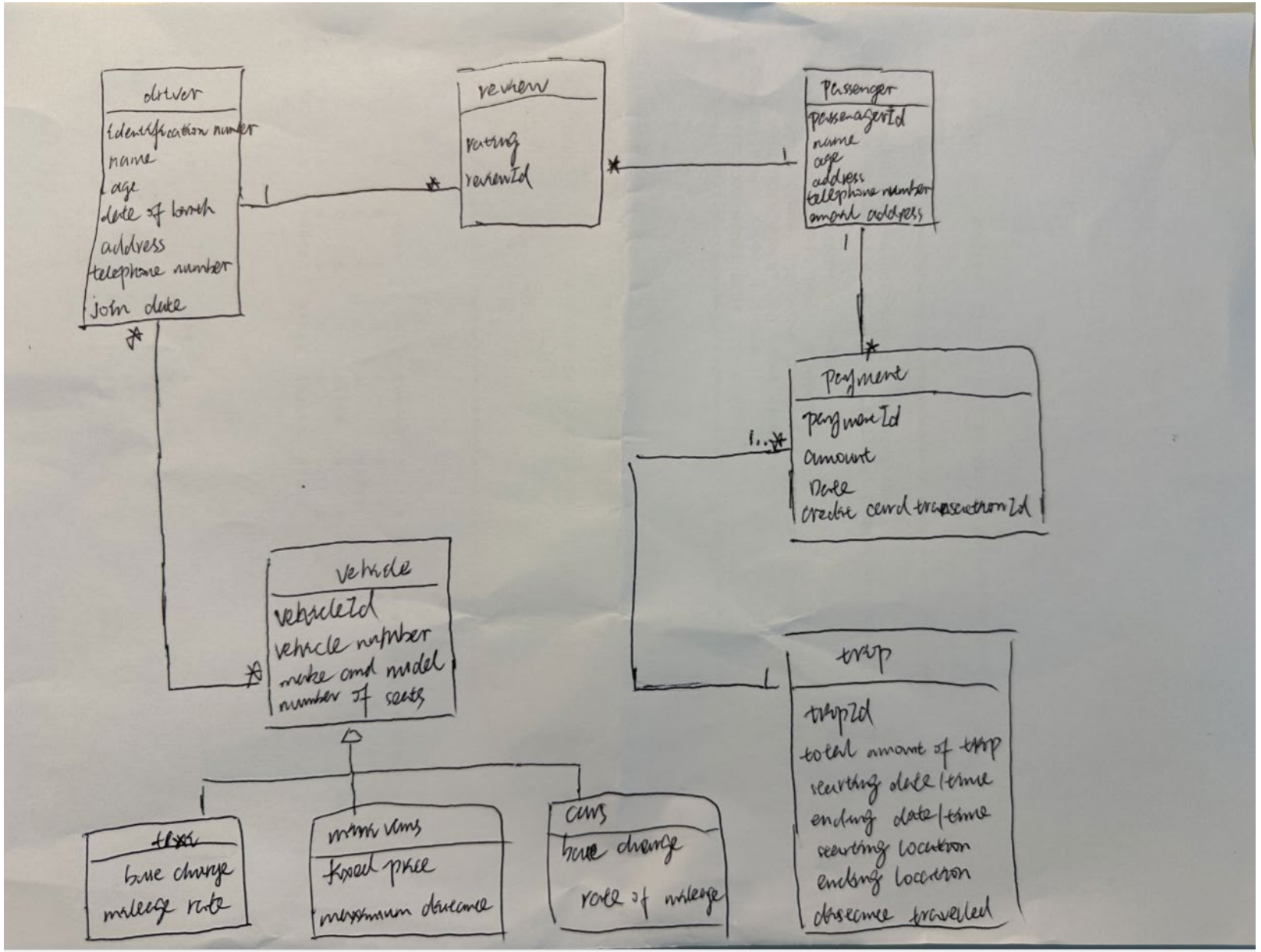
TripRecord should be associated with Vehicle



Should be using generalization symbol

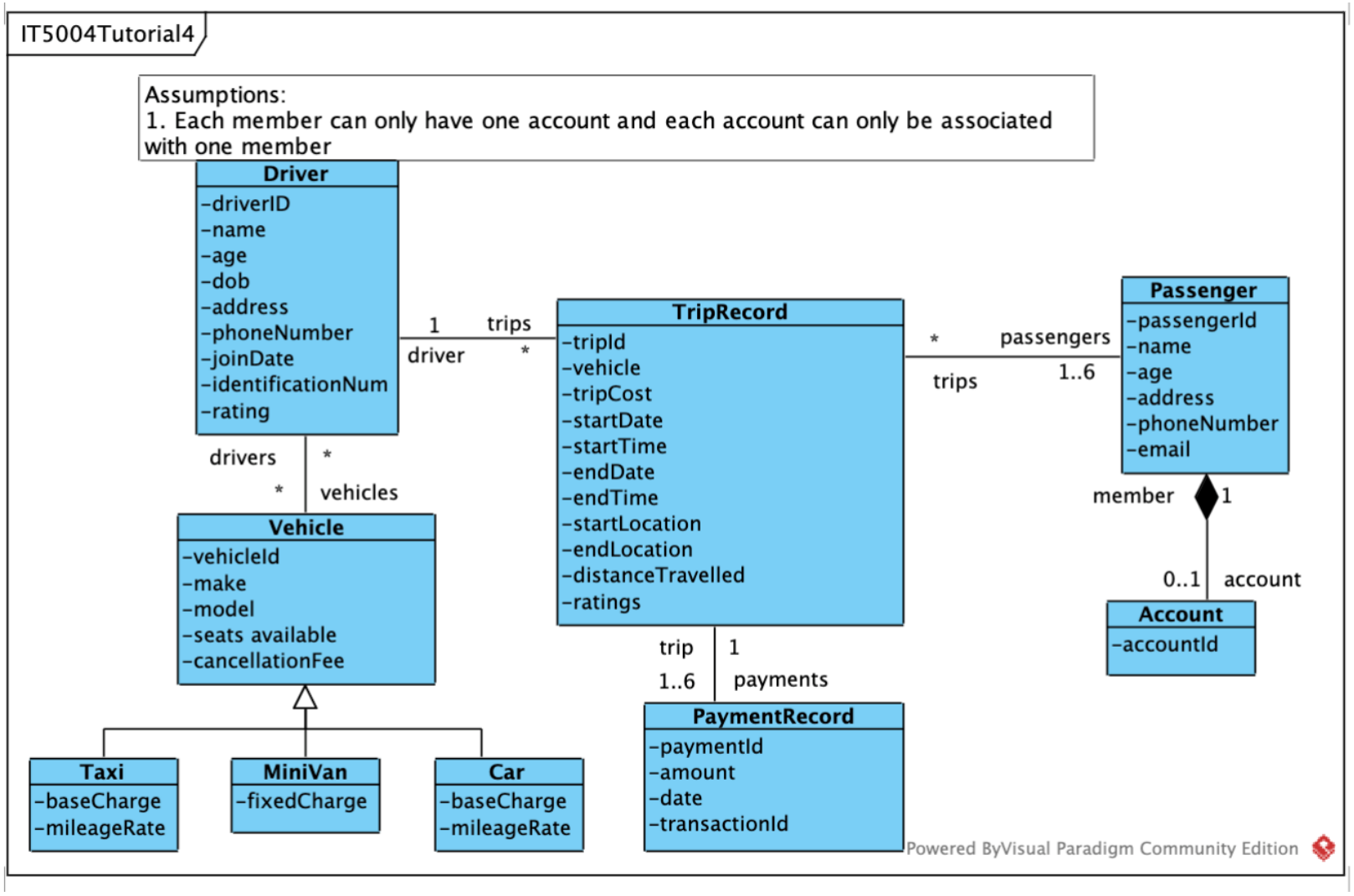
Association is not the same as generalization

Depending on the multiplicity of the association, you should associate the entities to each other to capture important information (e.g. who is the Driver, which vehicle is used, etc)



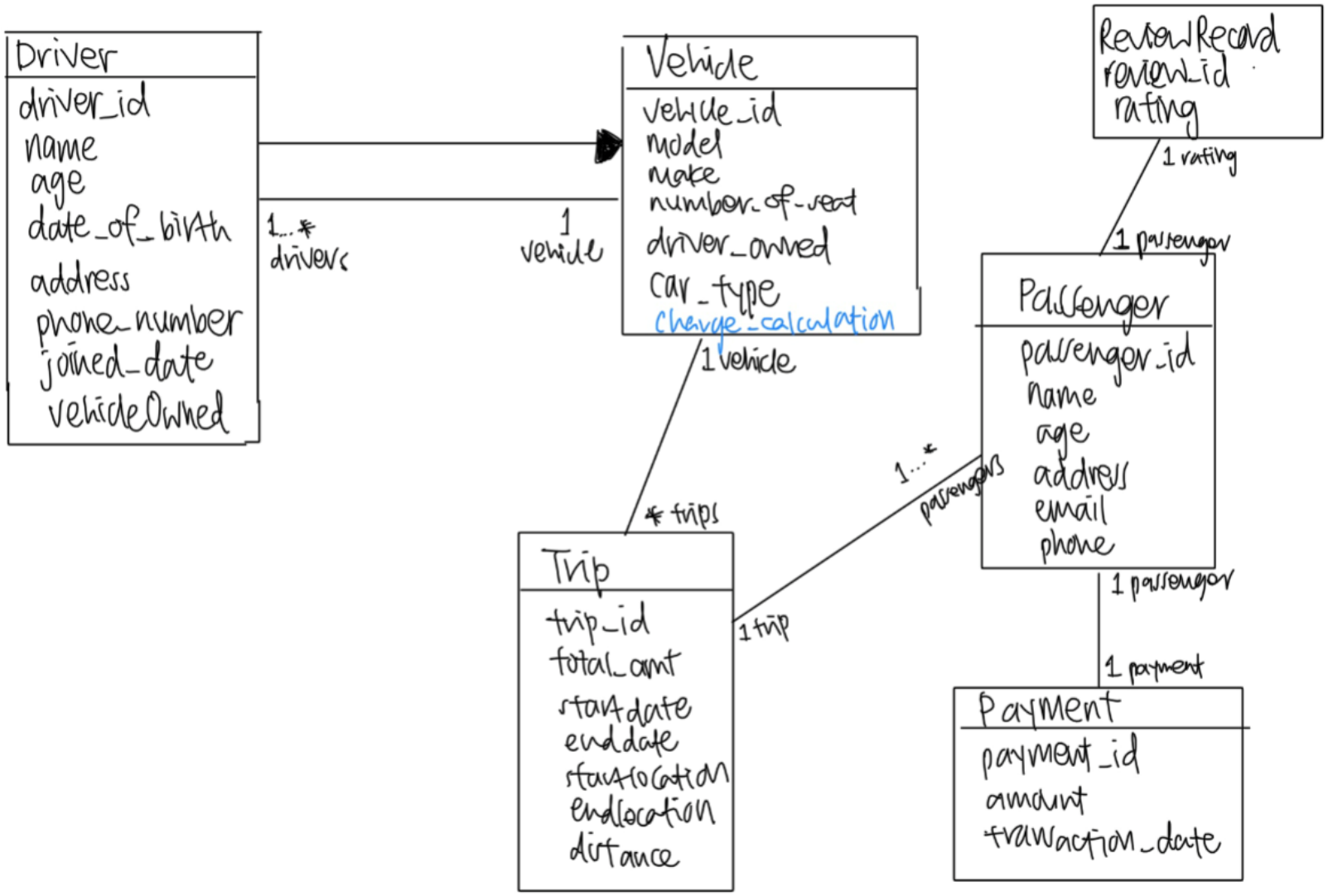
Missing rolenames for the entities

Trip should be linked with the Passenger, Vehicle and Passenger



TripRecord should be linked with Vehicle since a Driver can drive multiple vehicles

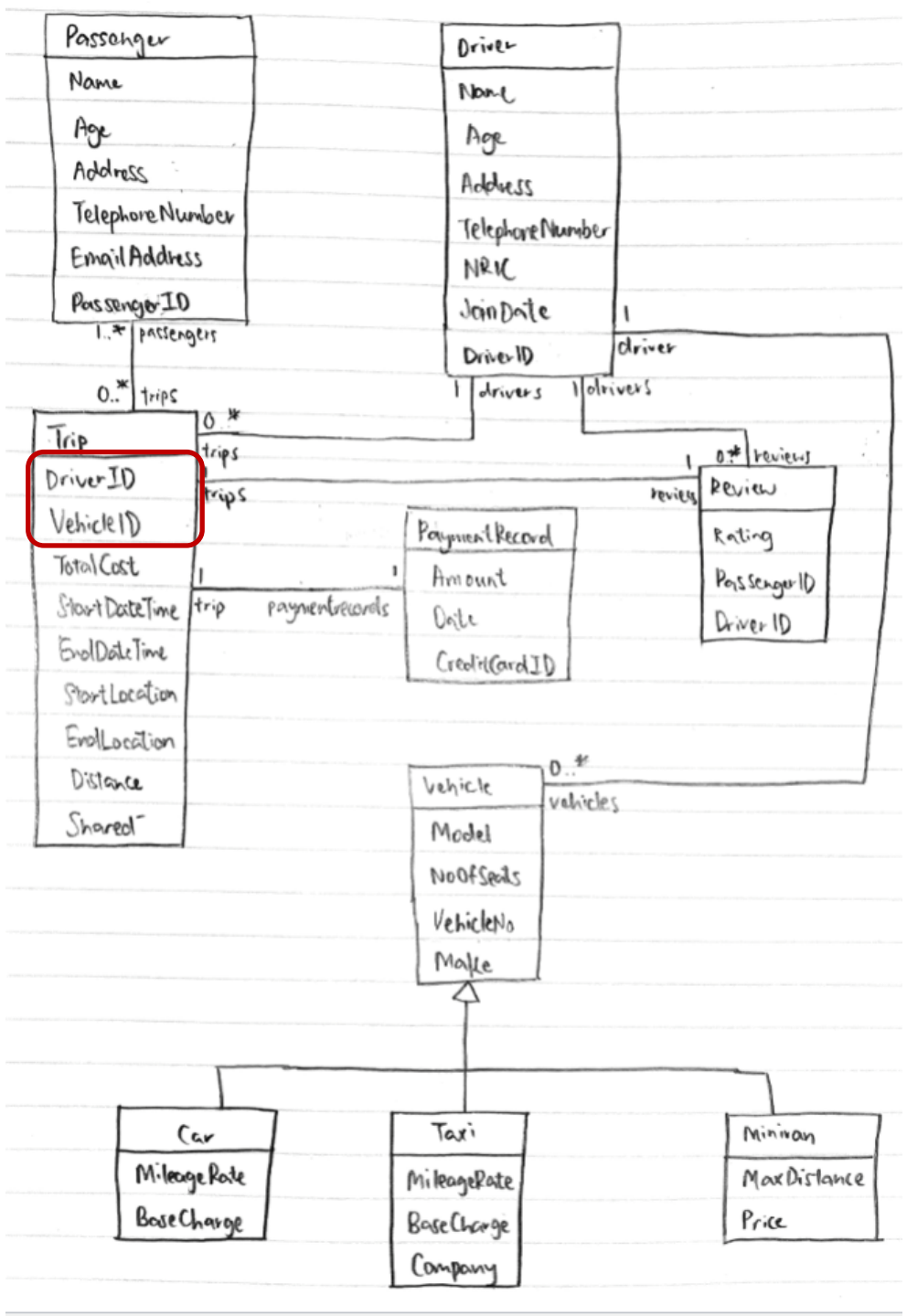
PaymentRecord should be linked with Passenger



Trip should be linked to Driver since a Vehicle can be driven by multiple Drivers

Each passenger can have multiple Payment

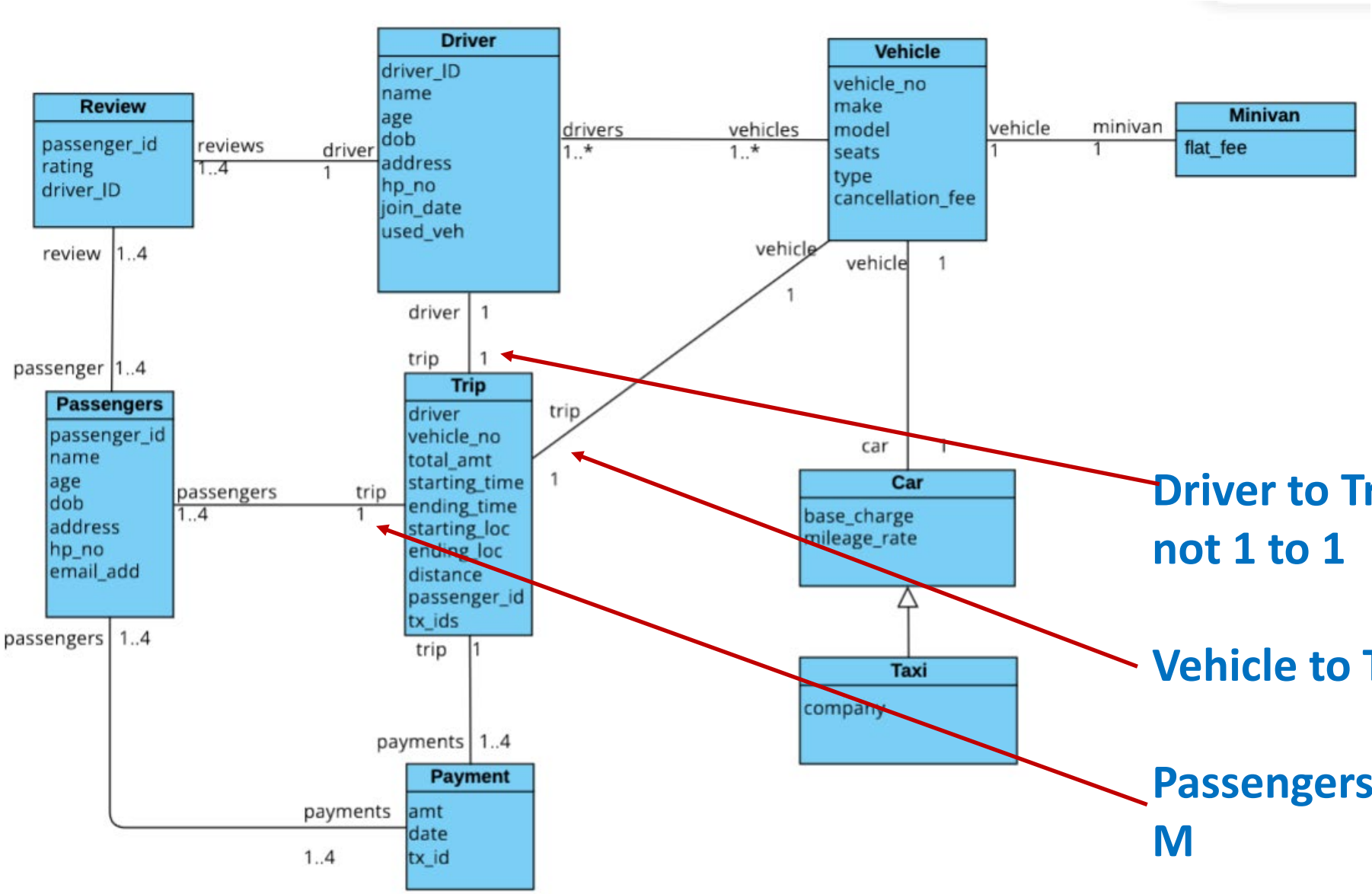
Payment should be linked with Trip



Trip should be associated with Vehicle. Why?

Don't store the IDs of the associated entities (e.g. DriverID, VehicleID):

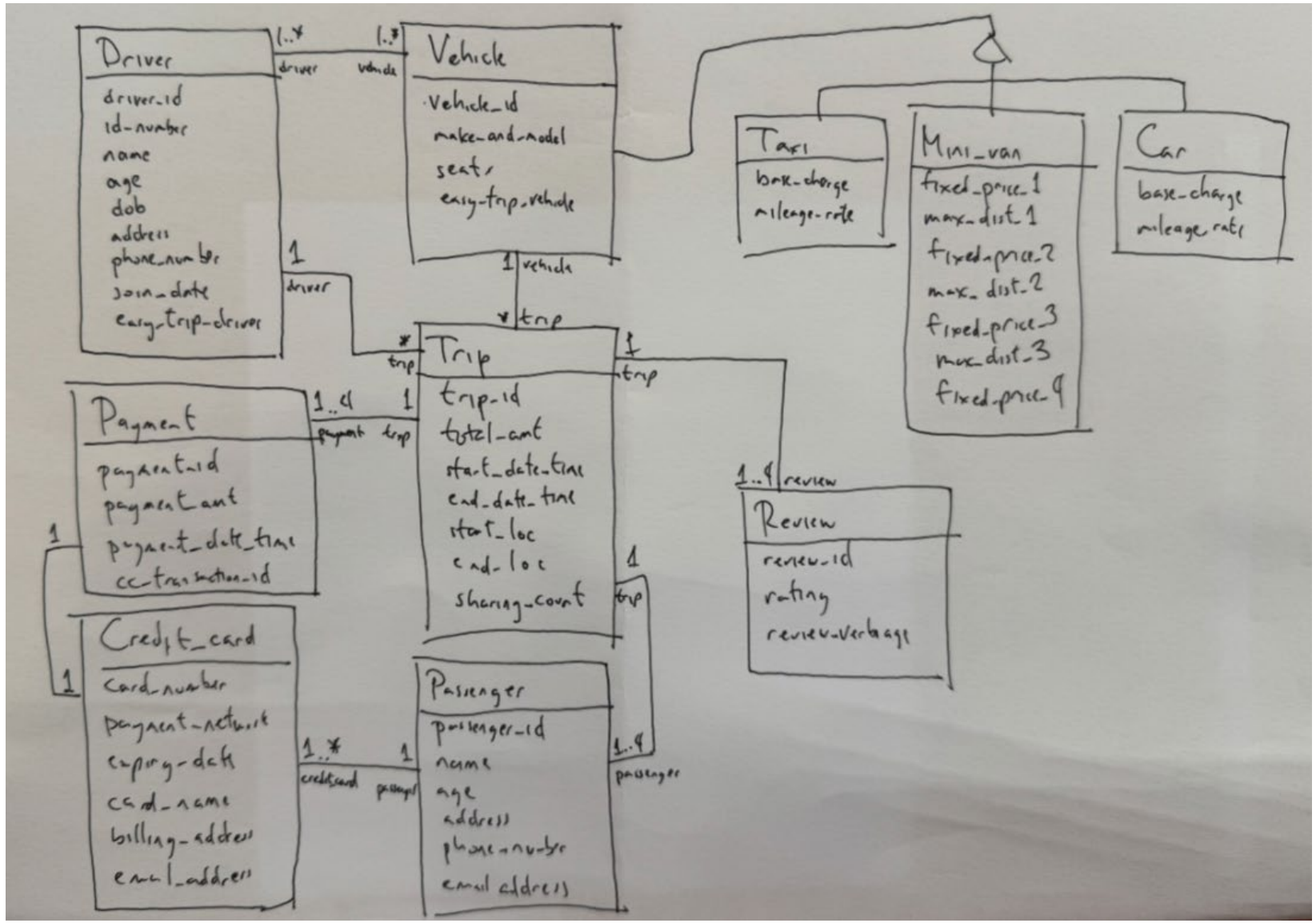
- **Think associating objects not foreign key (like in Database)**
- **That's what rolenames are**



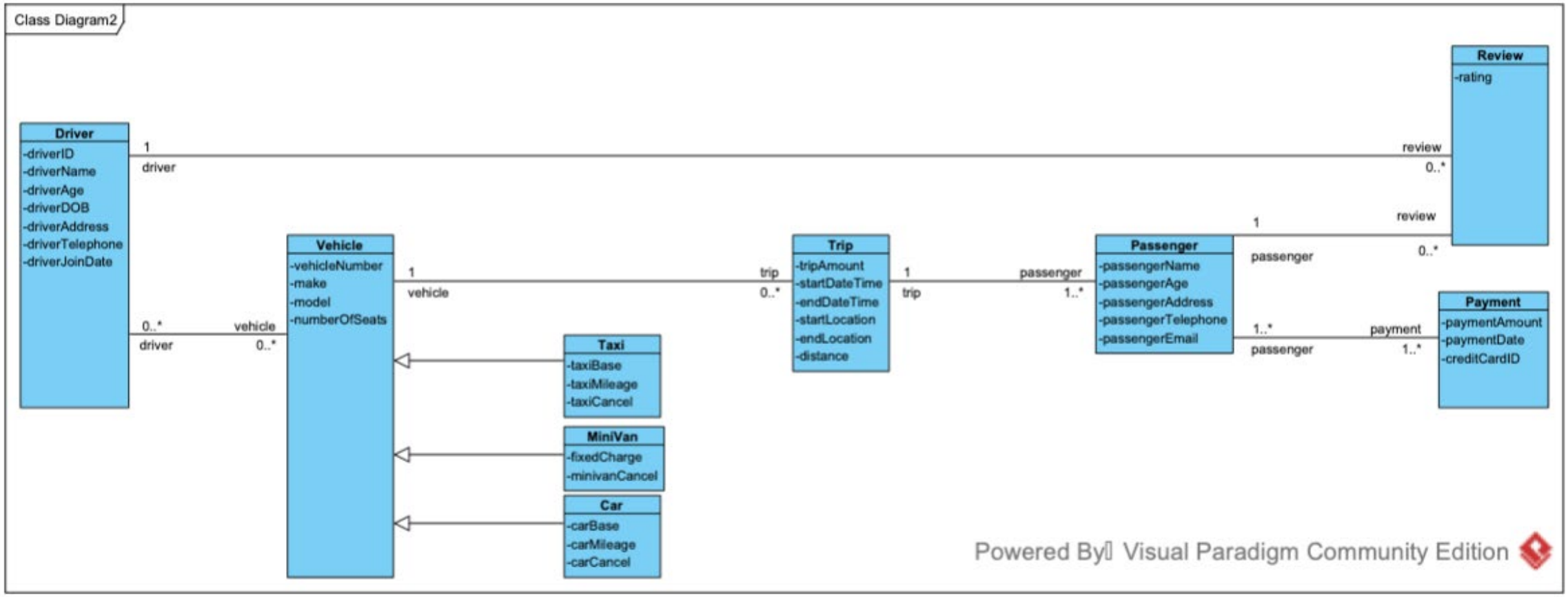
Driver to Trip should be 1 to M not 1 to 1

Vehicle to Trip should be 1 to M

Passengers to Trip should be M to M



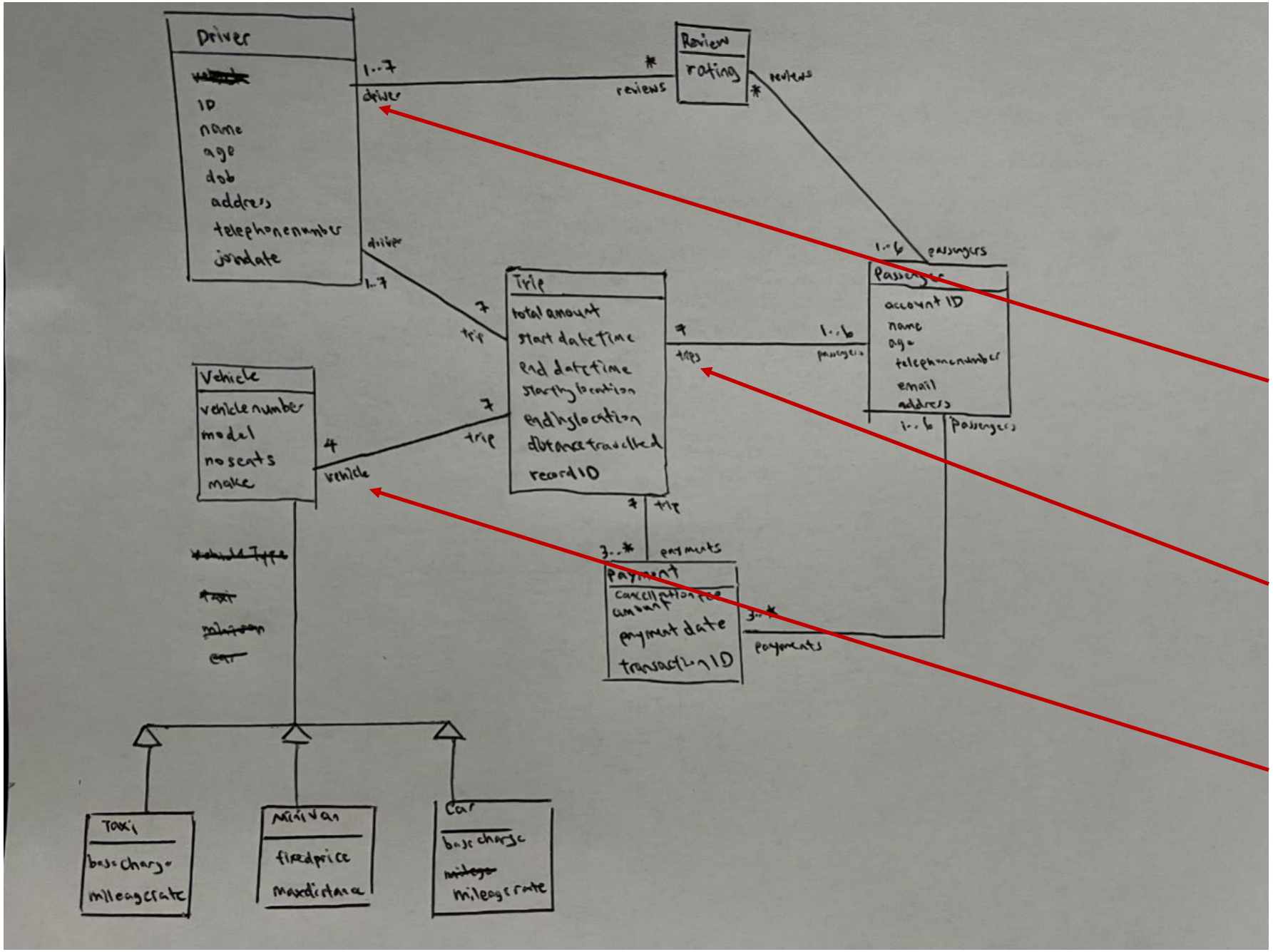
Review should be associated with Passenger



Missing primary key for review, payment, passenger, etc

Trip should be associated with Driver

Payment should be associated with Trip



Overall, it seems like you might have a misunderstanding of multiplicity

Each Review should be associated with just 1 Driver

Each Passenger should he multiple trips (not just 7)

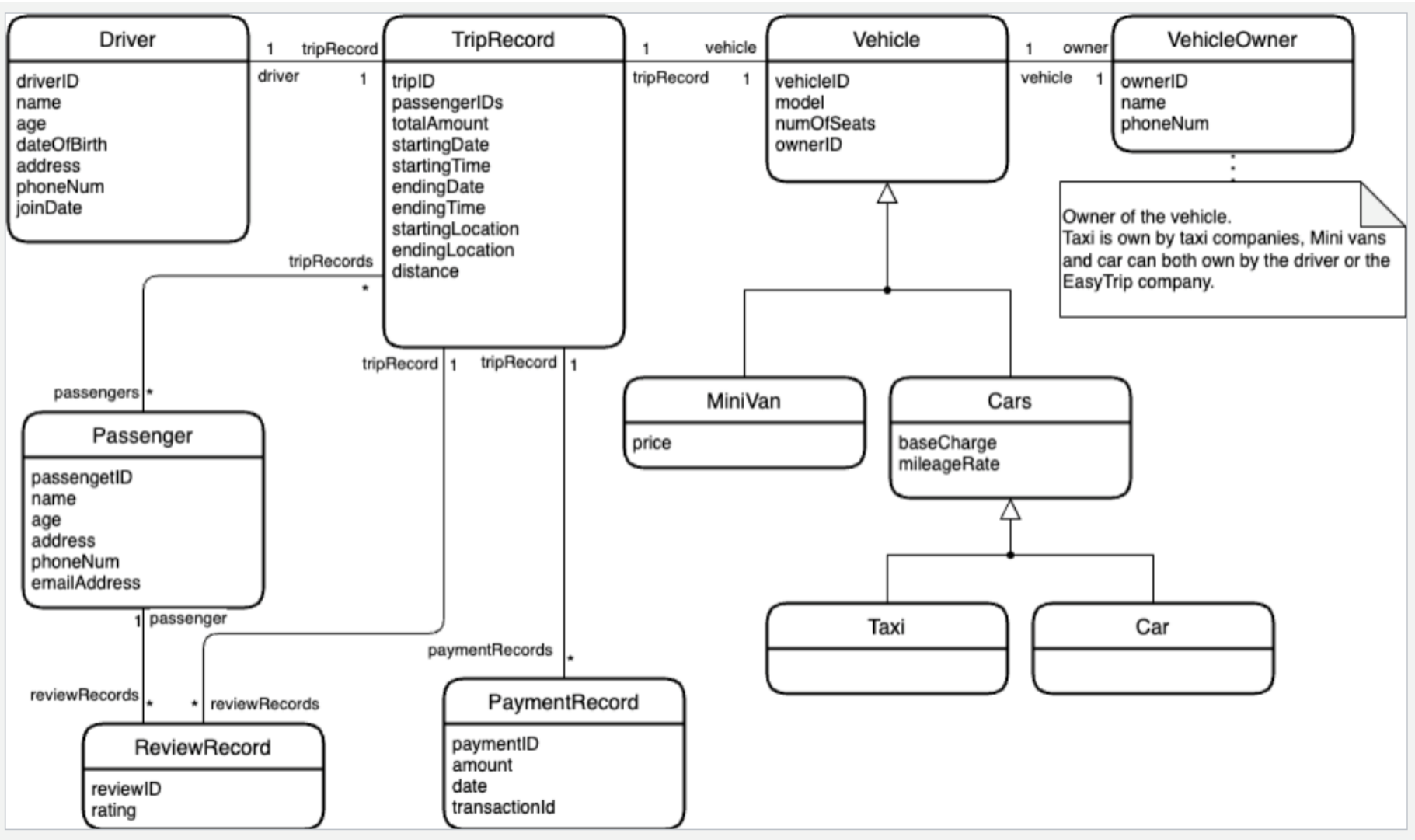
Trip should be associated with 1 Vehicle

Try to be consistent in the naming:

Driver instead of **Drivers**

Driver instead of **DriverRecord**

Uppercase for class name, etc



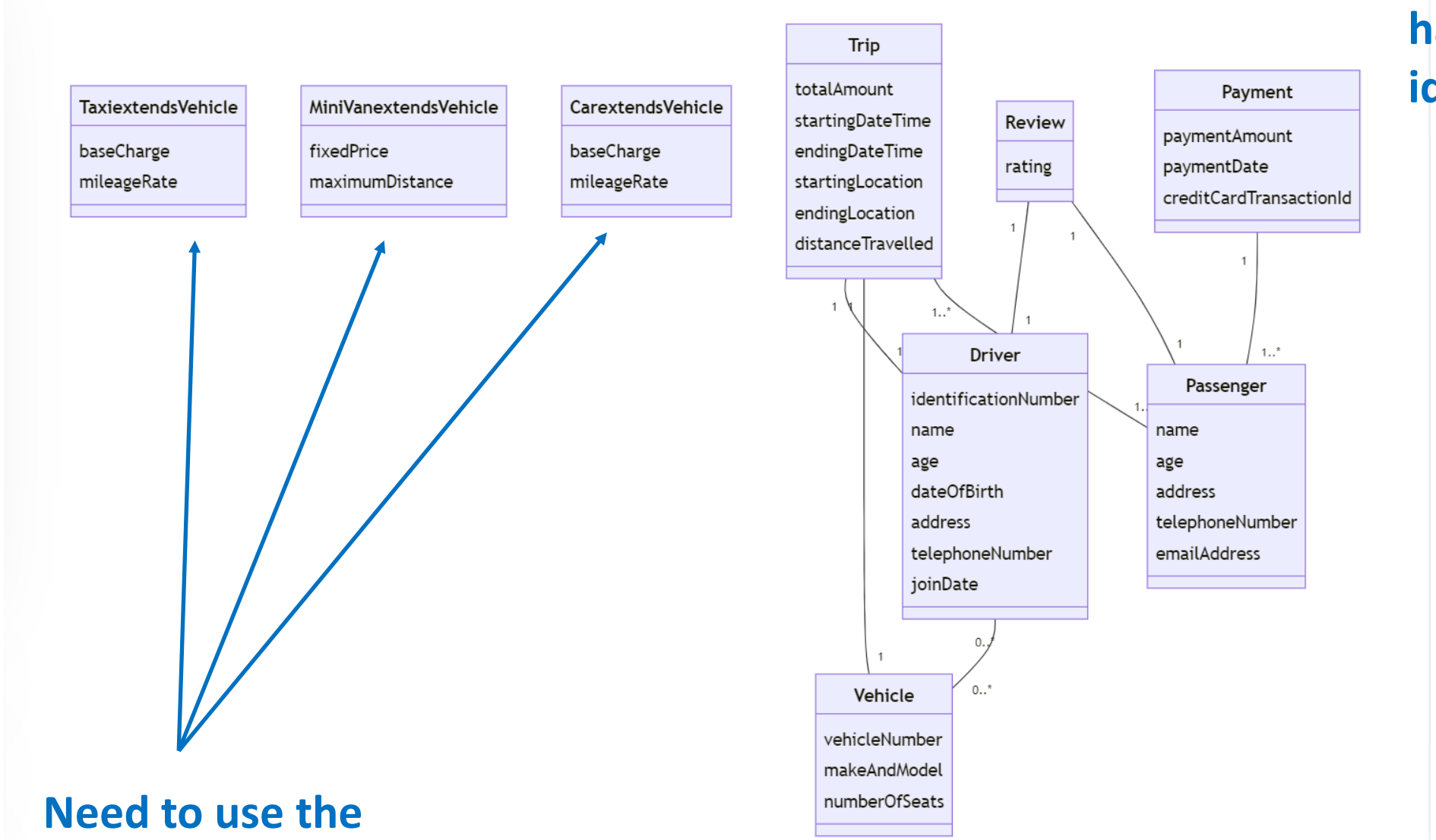
Driver should be associated with Vehicle

PaymentRecord should be associated with Passenger

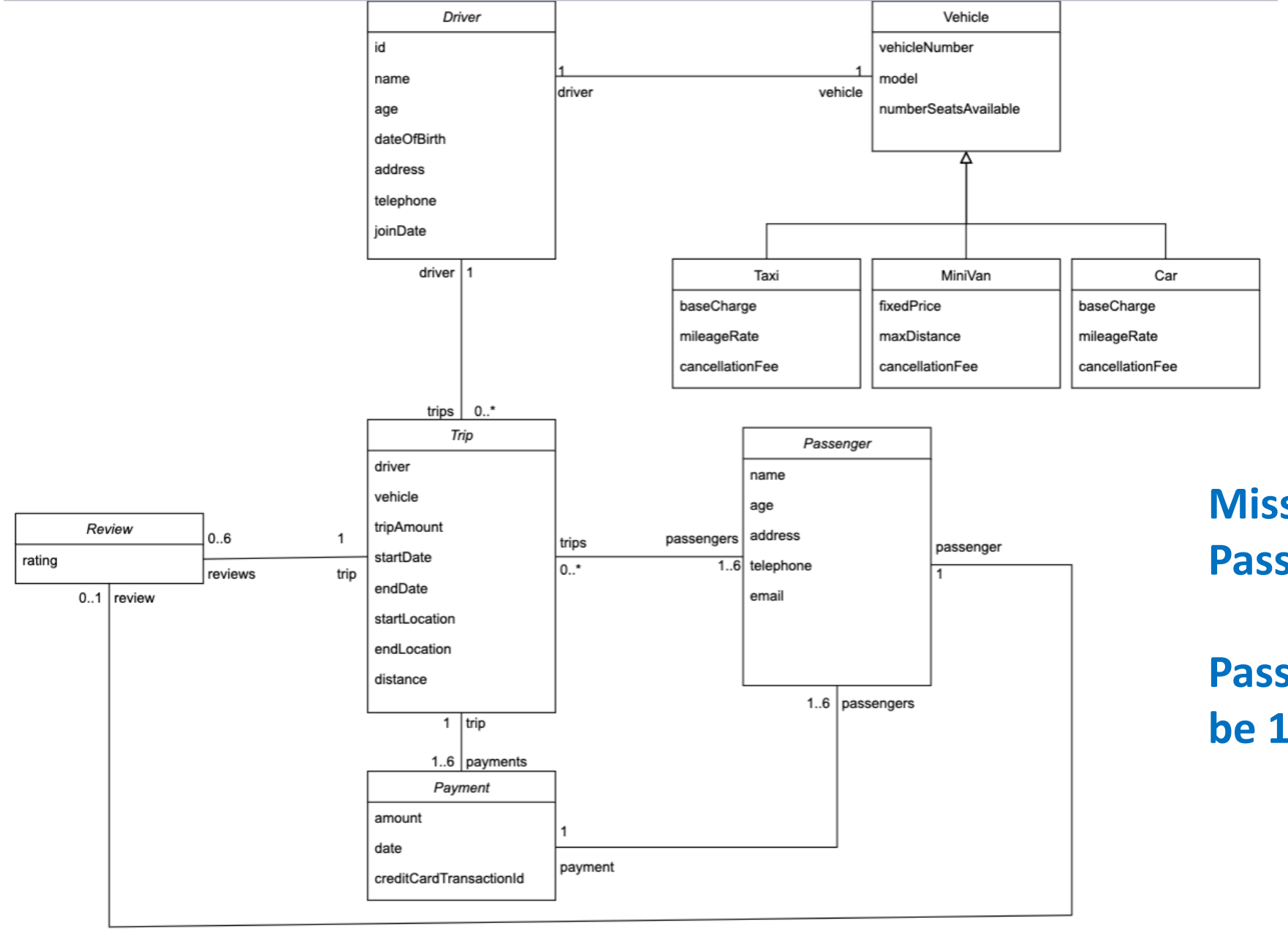
Try to be consistent in the naming (probably everything not ending with XXXRecord)

Make sure **every entity class has a primary key** (ID field e.g. driverId, passengerId, etc)

Make sure every class has a primary key (i.e. id field)

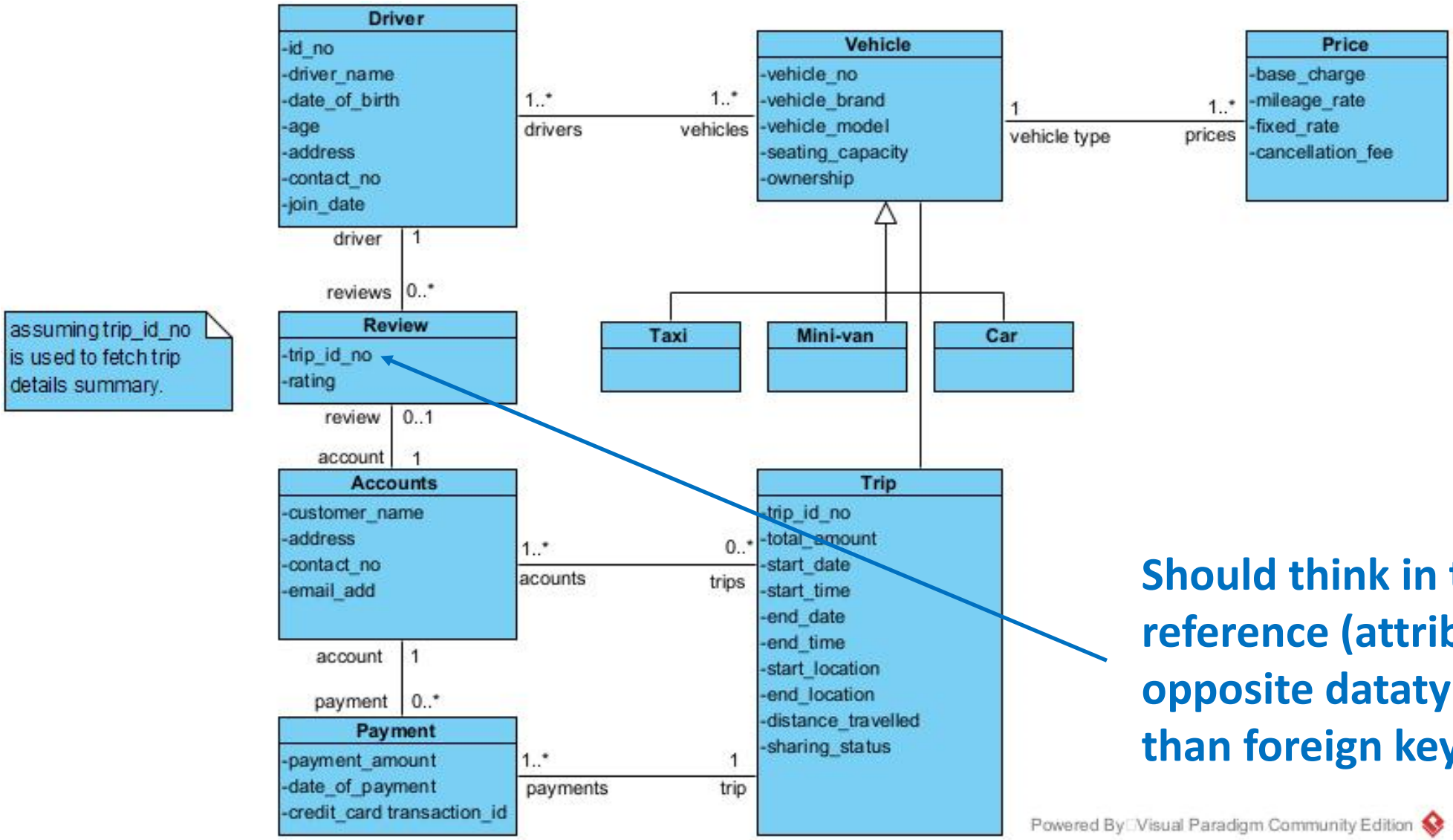


Need to use the generalization symbol



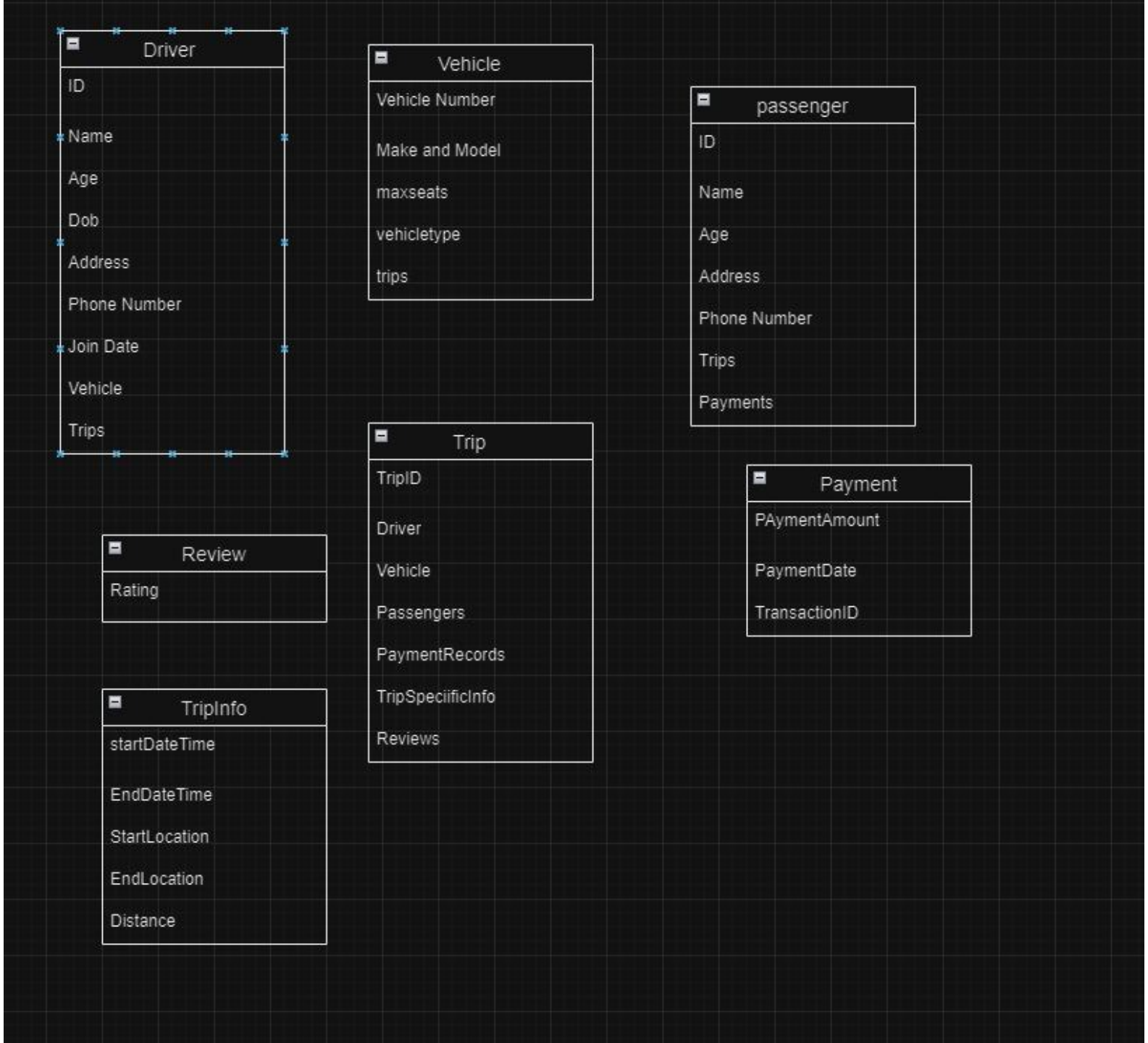
Missing primary key for Passenger, Review, Trip

Passenger to Payment should be 1 to many

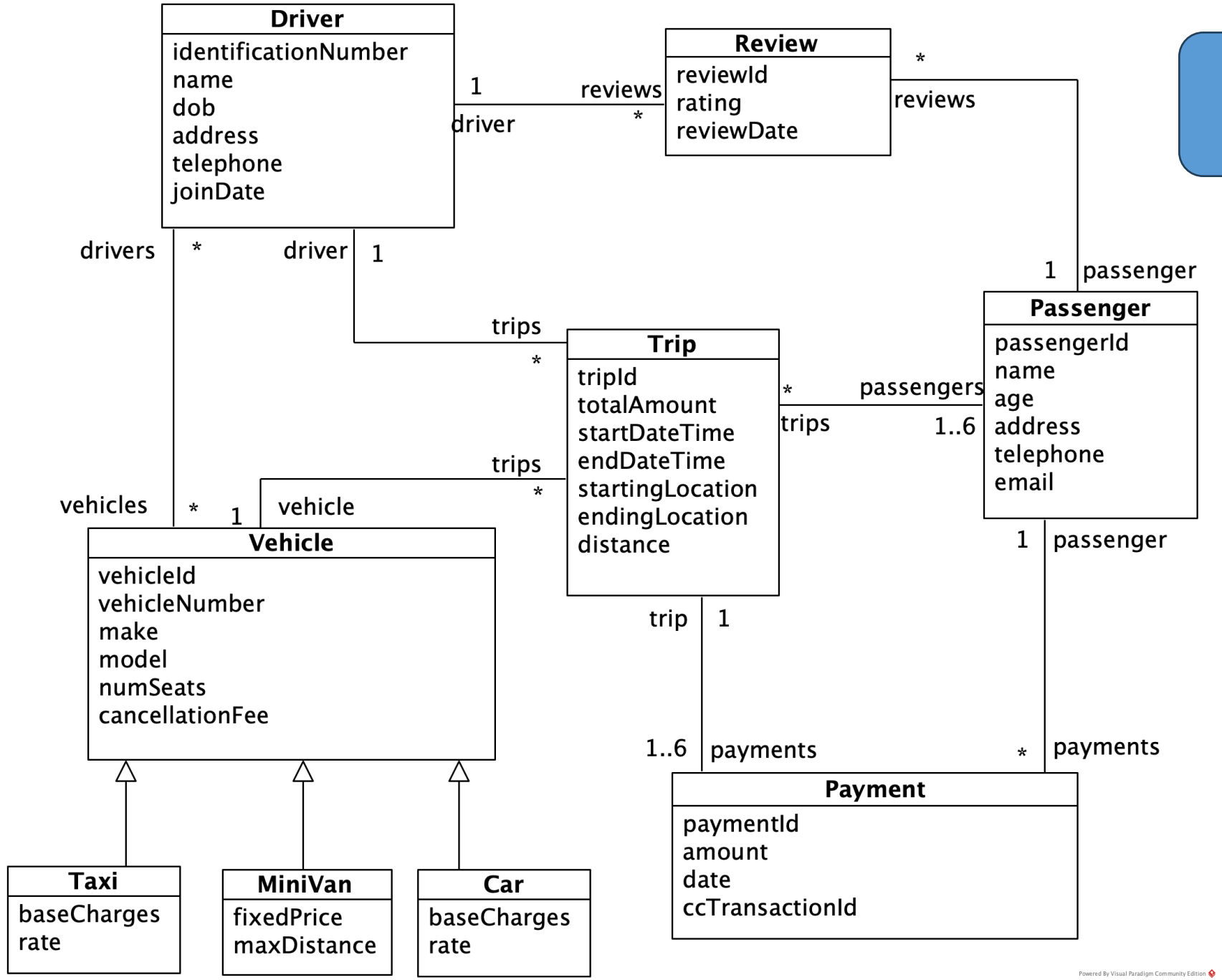


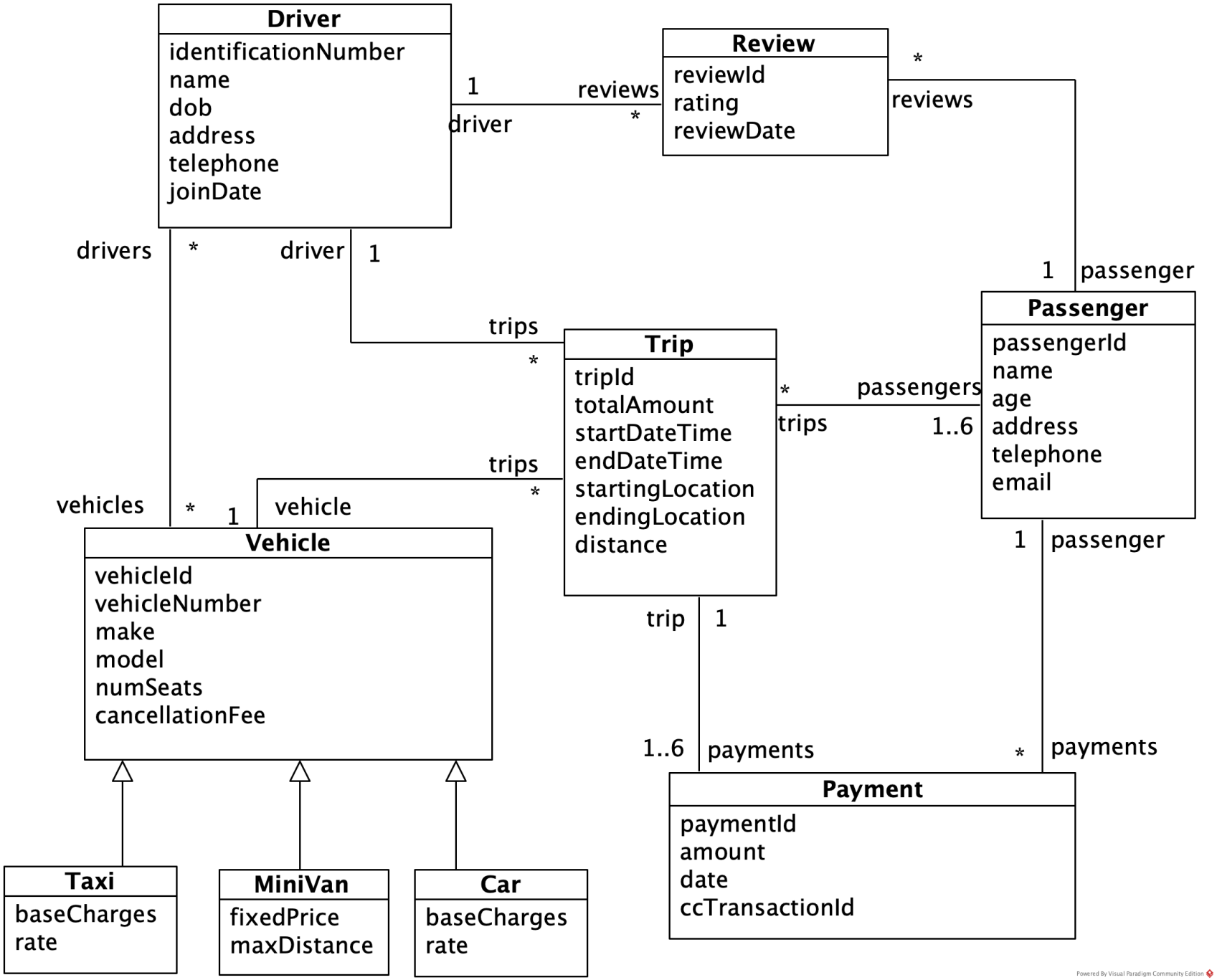
Should think in terms of reference (attribute of the opposite datatype rather than foreign key)

In most cases, the entities in a Domain Model Class Diagram are all linked up



Classes should be mostly linked to each other


Driver
identificationNumber
name
dob
address
telephone
joinDate
Review
reviewId
rating
reviewDate
1
reviews
*
driver
*
reviews
drivers
*
driver
1
1
passenger
Passenger
passengerId
name
age
address
telephone
email
trips
*
Trip
tripId
totalAmount
startDateTime
endDateTime
startingLocation
endingLocation
distance
*
passengers
trips
1..6
trips
*
vehicles
*
1
vehicle
Vehicle
vehicleId
vehicleNumber
make
model
numSeats
cancellationFee
1
passenger
trip
1
1..6
payments
*
payments
Payment
paymentId
amount
date
ccTransactionId
Taxi
baseCharges
rate
MiniVan
fixedPrice
maxDistance
Car
baseCharges
rate
Sample Solution
Powered By Visual Paradigm Community Edition



Sample Solution

Some questions to consider:

- **Why do we need certain association?**
- **What's the implication of each multiplicity?**
- **What can a design encode/not encode?**