AGENDA

1.> code ParkingLOT

•> schema design

•> start Models/controller/ DTO's

start by 9:05 PM

*) SCHEMA DESIGN:

class → schema design

1·) Convert all classes → tables

2·> Add attributes

·) NON P. attributes

·> Primitive —

↳ cardinality → Represent

① ParkingLot

- List<ParkingFloor>
- List<Gate>
- int size
- id

② ParkingFloor

List<ParkingSpot>

floor_NO

③ ParkingSpot

no. ✓

vehicleType ×

spotStatus ×

floor-id ✓

④ Gate

gateNO

Operator

status

GateType

⑤ Ticket

spot ×

entryTime ✓

vehicle ×

Gate ×

operator ×

⑥

↓

ElectricParkingspot

- Meter

tables:

Parking_lot
Parking_Floors
gates
Parking_spots
vehicles
operators
tickets
bills
Payments

ENUMS
✓ # INHERITANCE

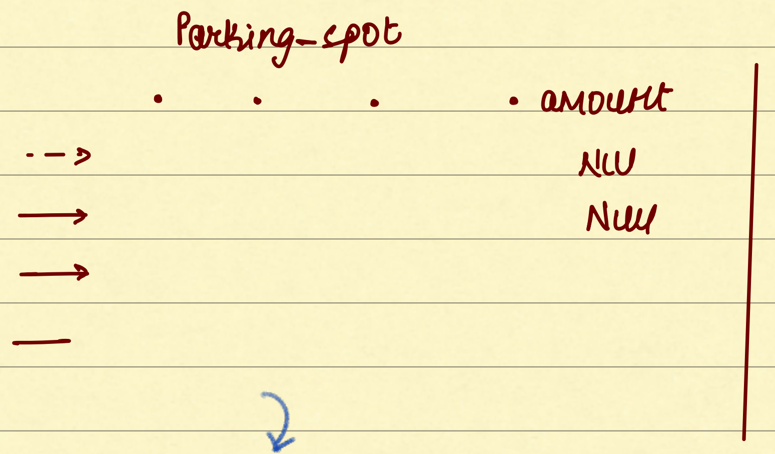
*) REPRESENTING INHERITANCE:

Parking spot

↓

evParkingspot

- Meter_amount



A1:> Parking-spot •id, vehicle , floor

ev-parking-spot •id, vehicle, floor, meter

PROBLEM:

both tables Needed.

A2:> **Better Approach:**

•id, vehicle , floor

•id, vehicle, floor, meter

① Parking-spot : id, vehicle, floor. vehicle-type

② electronic-parking-spot : Parking-spot-id, meter-amount

steps:

1:) crete table for Parent class

2:> create Mapping table for child class

*) HOW TO REPRESENT ENUMS:

```
ParkingSpotType {
    2W, 4W
}
```

EV-2W

A1:) park-spot-type → Mapping / lookup table

id	name
1	2W
2	4W
3	EV-2W

① Parking-spot : id, vehicle, floor. vehicle-type

*) SCHEMA DESIGN – PARKING LOT

① Parking_lots

id, size, status-id

② Parking-floors

id, floor-Number, parking-lot-id

③ Gates

id, gate-number, parking-lot-id

④ Parking-spots

id, floor_no, spot-number,

⑤ vehicles

id, number, owner-name

⑥ Operators

id, name, employee_id

⑦ tickets

id, entry_time, number

⑧ bills

id, exit_time, bill-amount

⑨ Payments

id, time, amount, reference_NO

(# S2) Relations b/w other tables

Parking_lot

① $\overset{①}{PL}$ <> $\overset{Ⓜ}{gates}$ → 1:M

Gates → parking_lot_id

② PL <> floor → 1:M

③ $\overset{1}{PL}$ <> $\overset{1}{status}$ → M:1

Parking_lot_status

id	value
1	OPEN
2	CLOSED

Parking floor:

$$\begin{tikzcd} PF \arrow[rr, bend left, "M"] & <> & slots \arrow[ll, bend left] & \longrightarrow & 1:M \end{tikzcd}$$

Gates

1 1
Gate <> GateType → M:1
M 1
↓

Gate_type

id,	value
1	ENTRY
2	EXIT

1 1
Gate <> Operator → 1:1

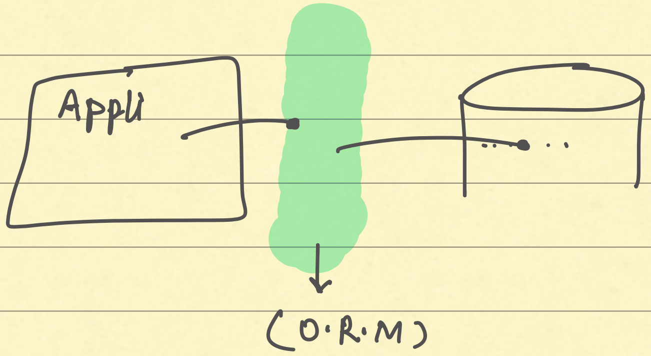
Parking_spots

1
P spot <> vehicle Type
M 1 ↓
(M:1)

M:1 / 1:M → id of (1) side ON (m) side

*) MODELS:

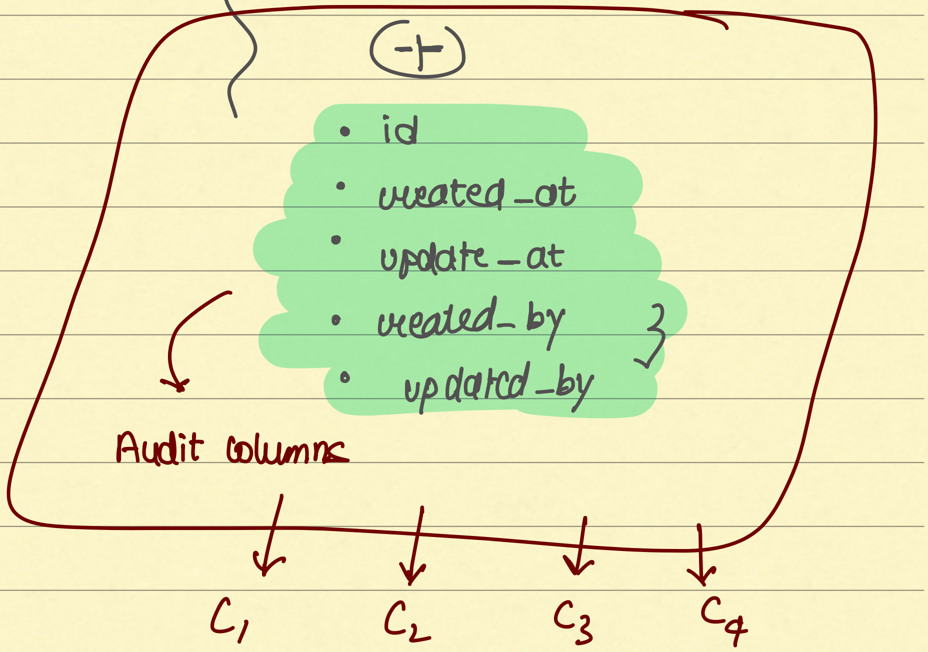
Models → classes (classes + Enums + ln...)



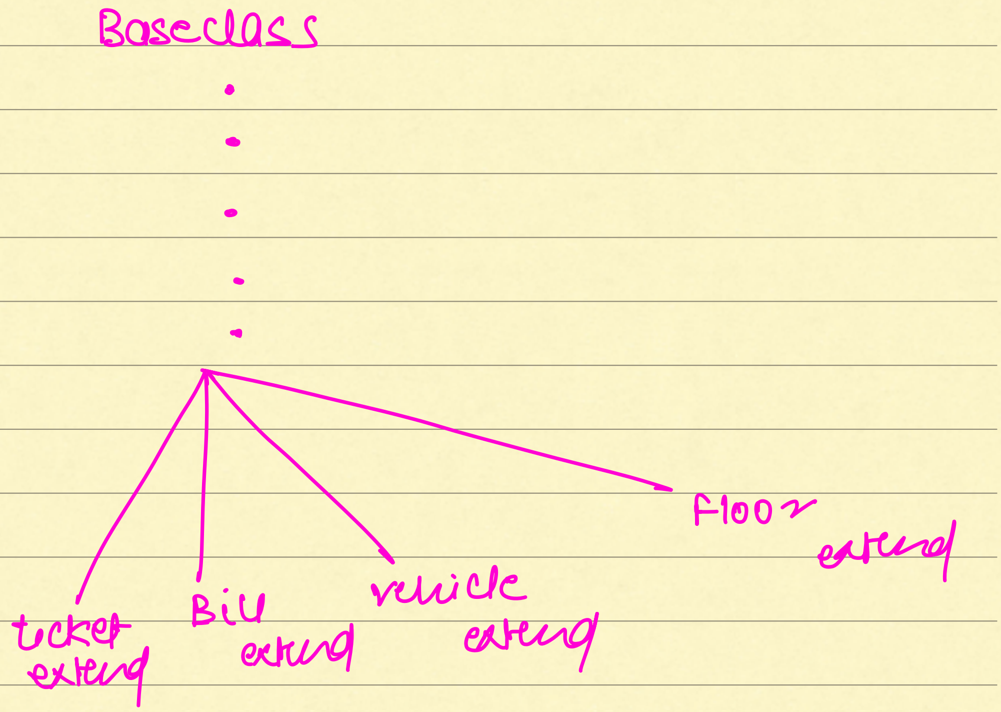
parking lot
spot
Floor
vehicle

→ classes

①



(2)



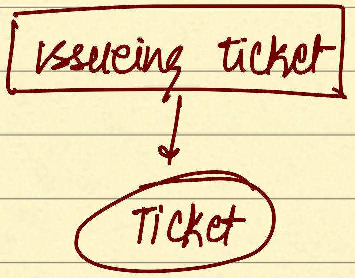
(3) create Getters / setters : for model classes.

(4) ticket → Represent table in D/B

↓

POJO

usecase:

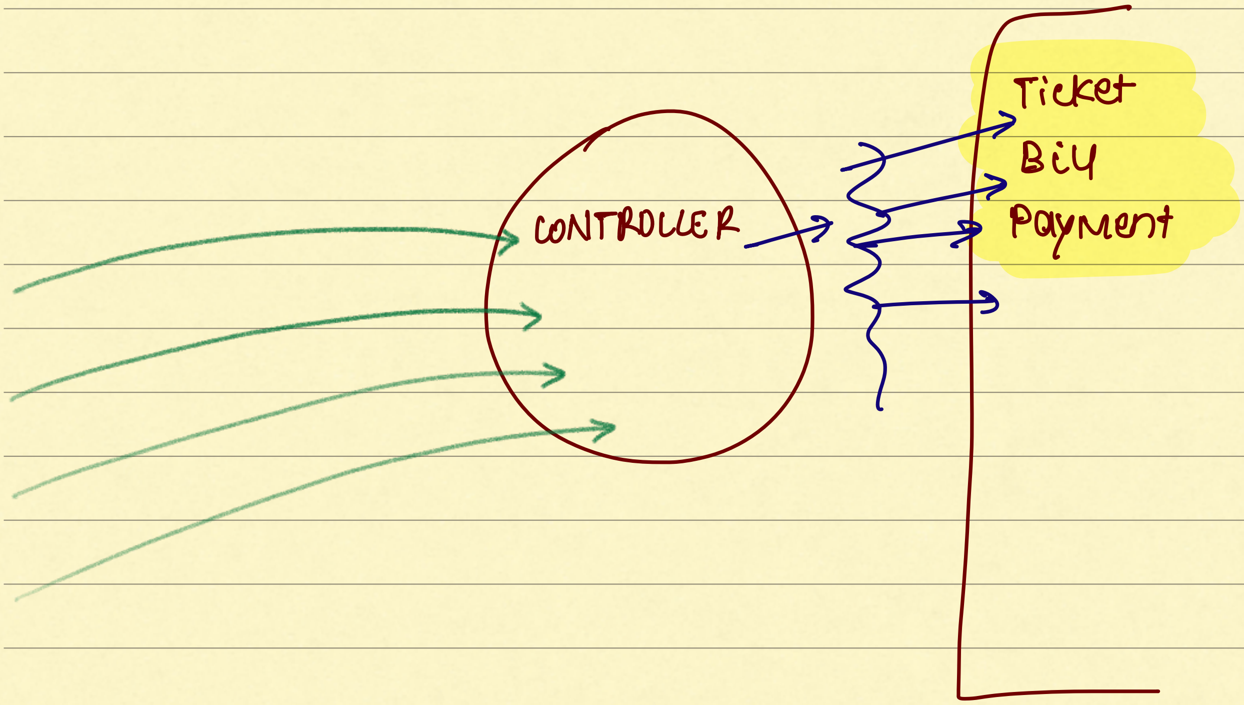


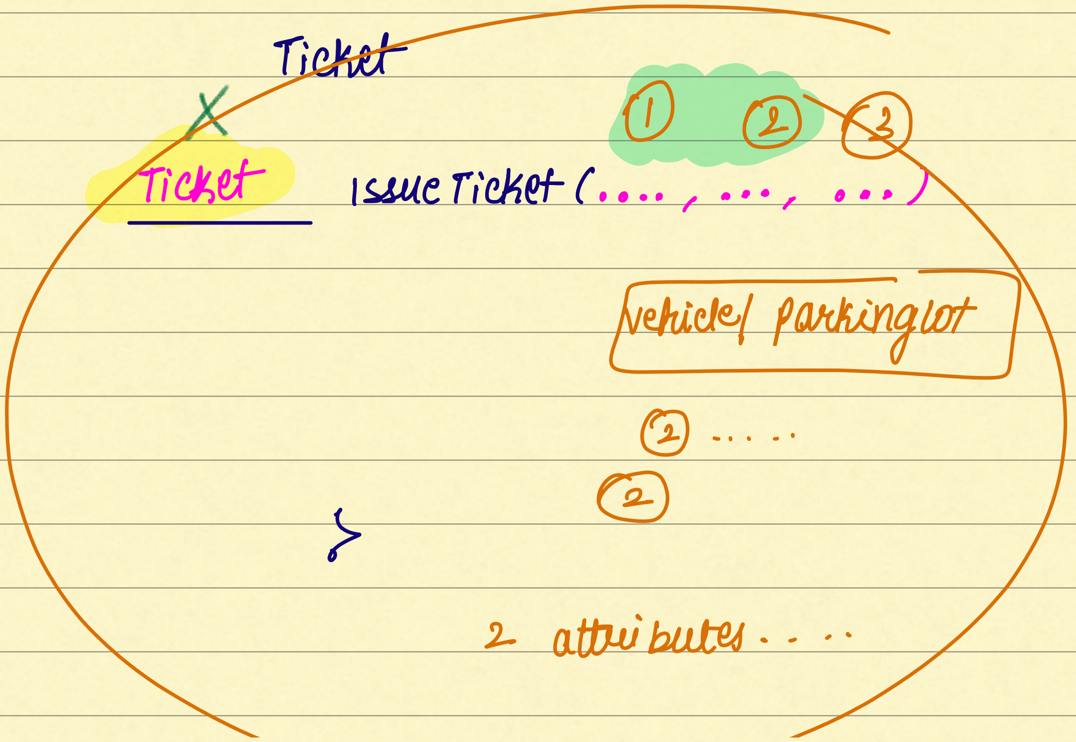
— TicketController

```
TicketController {

    issueticket(){
        //  → service
    }

    cancelTicket(){

    }

}
```

*) D.T.O :

(Data transfer object)





3 attributes . . .

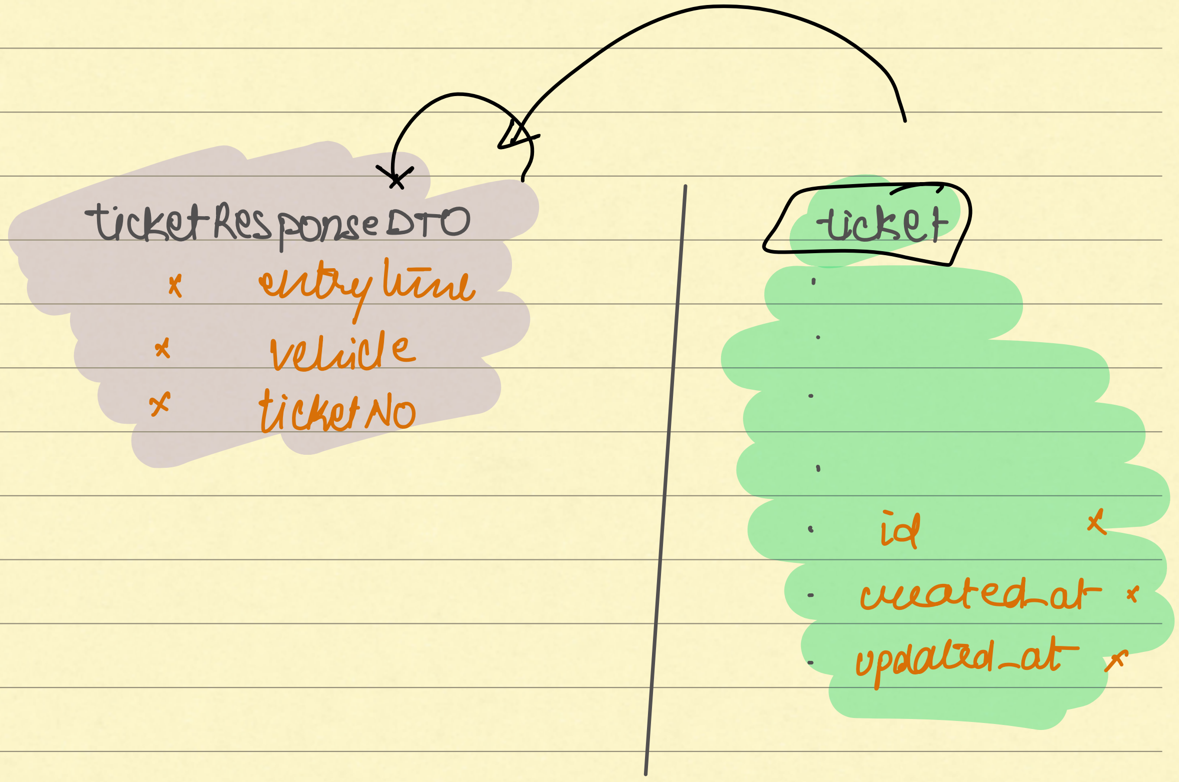
↓

DTO

data transfer object

1.) Request DTO

2.) Response DTO



ticketResponseDTO

- × entry time
- × vehicle
- × ticket No

ticket

- .
- .
- .
- .
- . id ×
- . created_at ×
- . updated_at ×

TicketResponseDTO IssueTicket (TicketRequestDTO dto) {

. . -

J