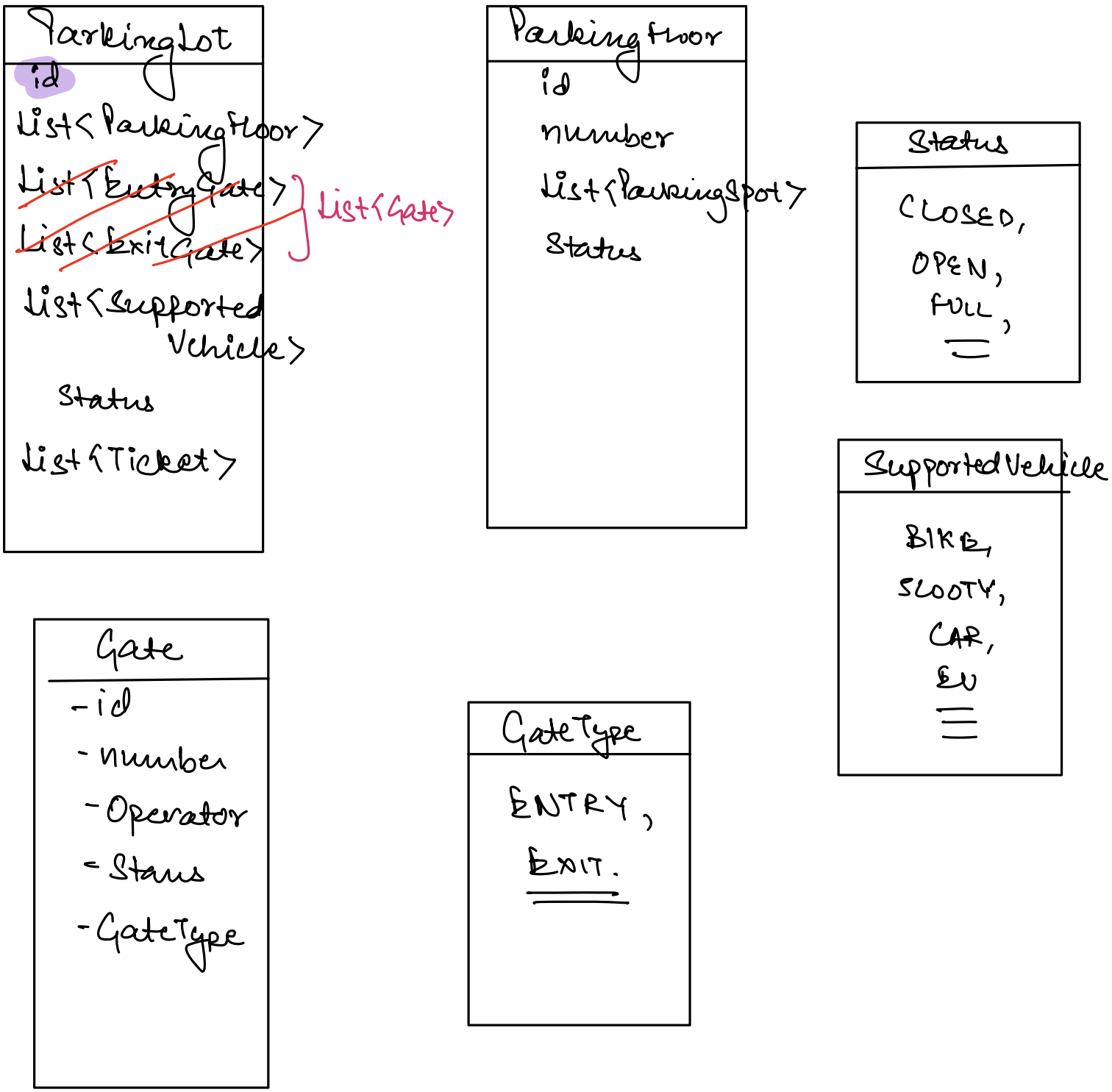
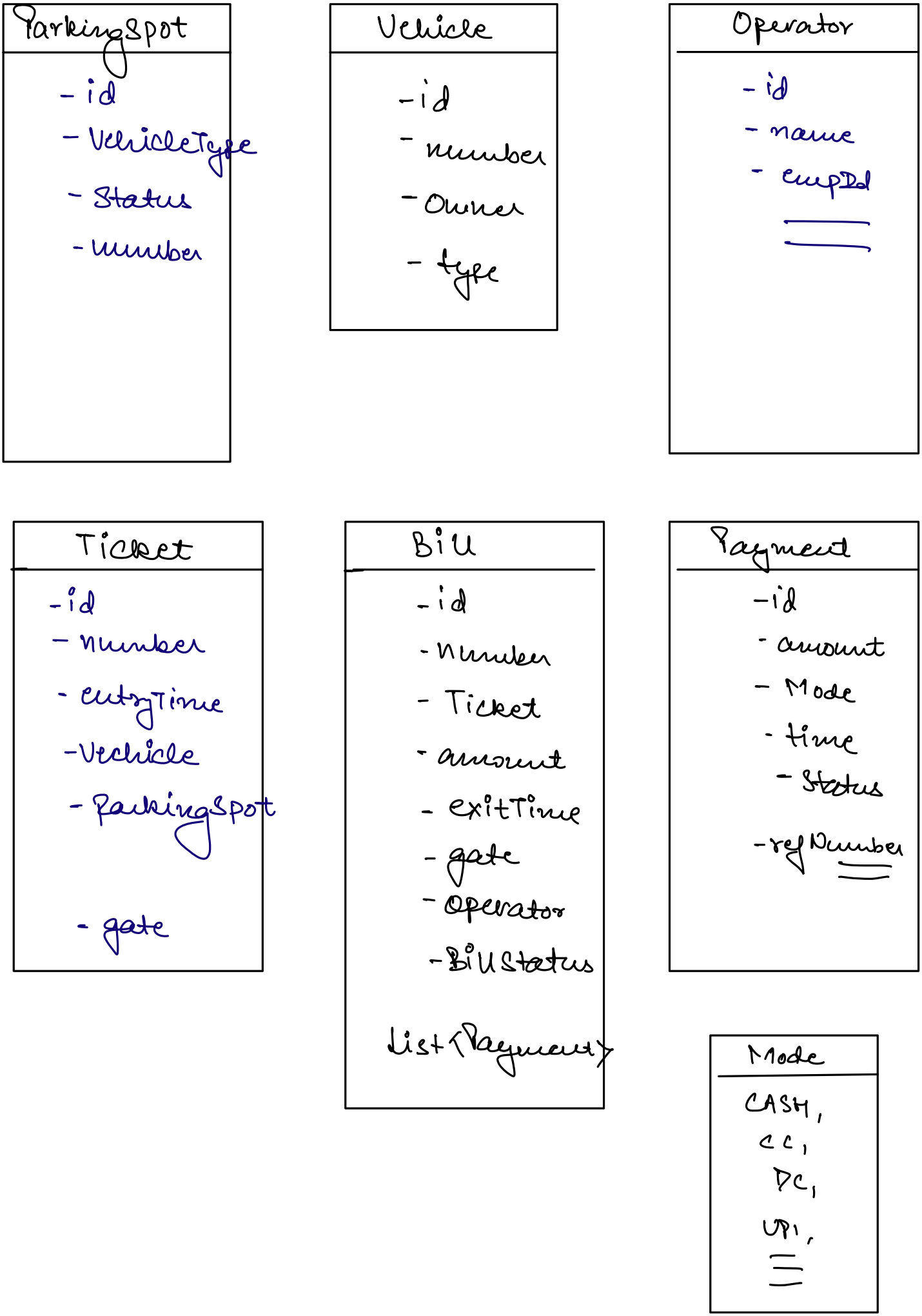
Agenda.

→ Schema Design.

→ Implementation.

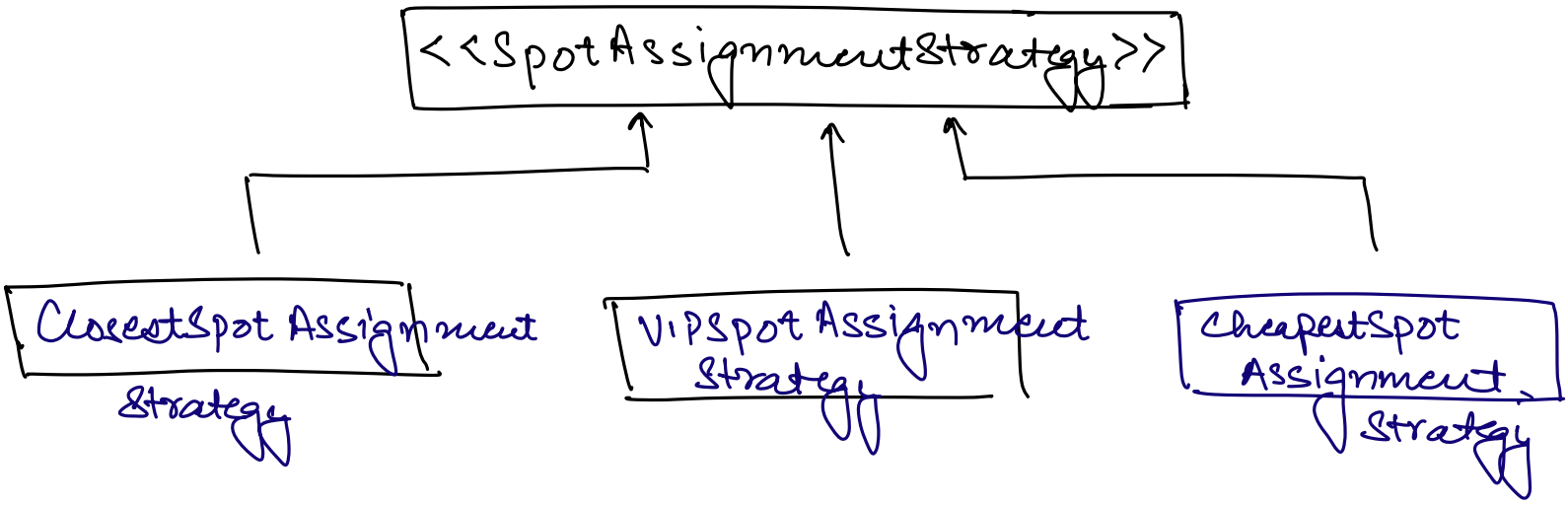
Class Diagram revision

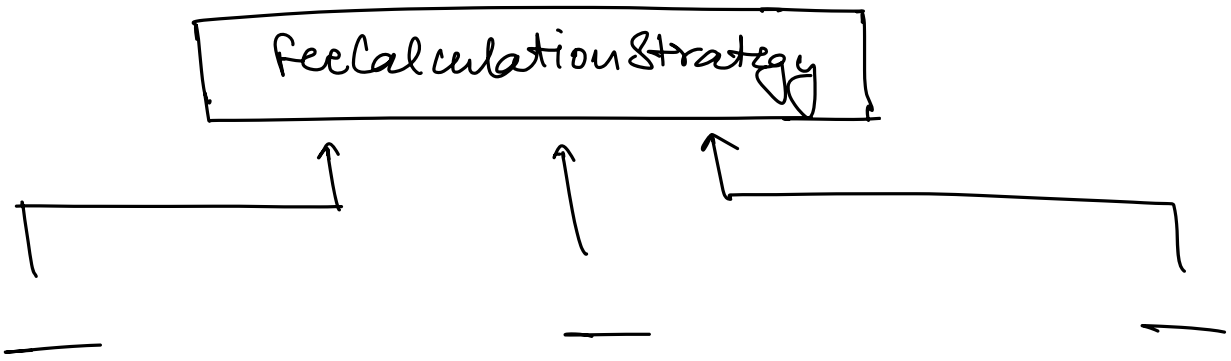



ParkingSpot
- id
- VehicleType
- Status
- number
Vehicle
-id
- number
- Owner
- type
Operator
- id
- name
- empId
Ticket
-id
- number
- entryTime
-Vehicle
- ParkingSpot
- gate
Bill
-id
- number
- Ticket
- amount
- exitTime
- gate
- Operator
-BillStatus
List<Payment>
Payment
-id
- amount
- Mode
- time
- Status
-refNumber
Mode
CASH,
CC,
DC,
UPI,

Design Patterns

Spot Assignment





Adapter ⇒ Integrate 3rd party PG.

Schema Design.

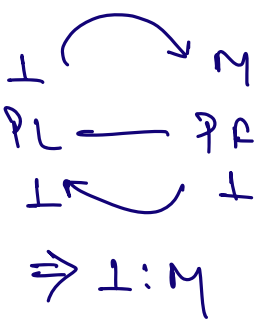
1. for all the classes in the class diagram, create a corresponding table in the DB.
2. for primitive (simple) attributes in these classes, represent as a column in the table.
3. for non-primitive attributes ⇒ relationship b/w the entities.
   - i) find the cardinality of that rel^n.
   - ii) Apply the rule to represent the cardinality in the DB.

⇒ Cardinality Rules.

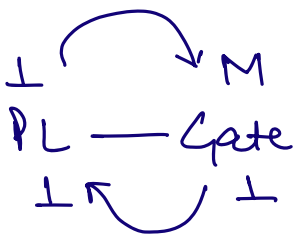
1 : 1 ⇒ Id of one side on other side.

1 : M | M : 1 ⇒ Id of (1) on (M) side.

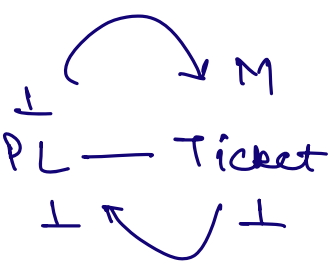
M : M ⇒ Mapping table.



⇒ 1 : M



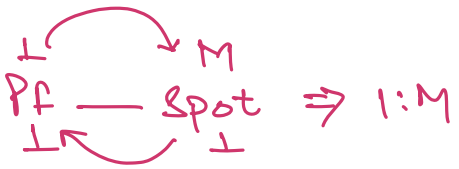
⇒ 1 : M



⇒ 1 : M

parking_lots

id

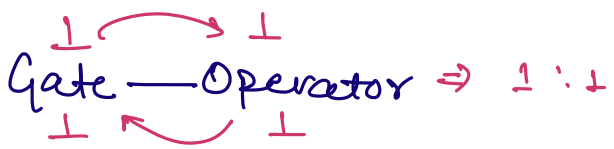


parking_floors

id	number	parking_lot_id

parking_spots

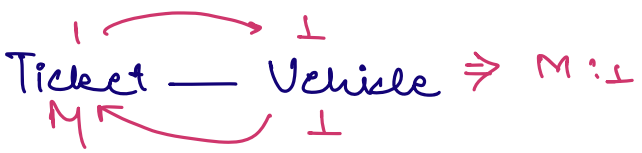
	parking_floor_id	



gates

id	number	parking_lot_id	operator_id

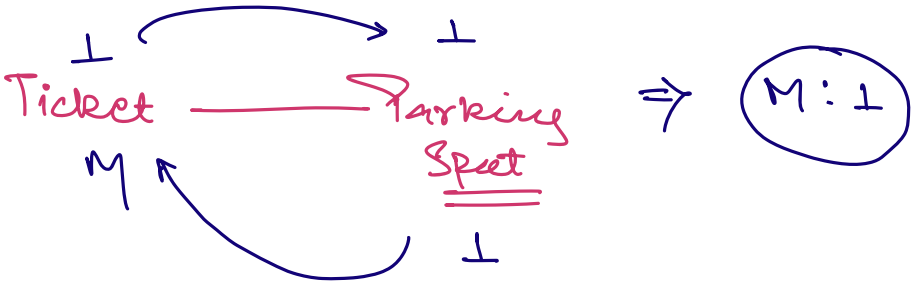
operators

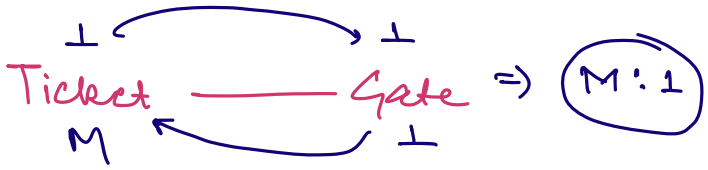


tickets

id	number	entry_time	parking_lot_id	vehicle_id	parking_spot_id

gate_id ...



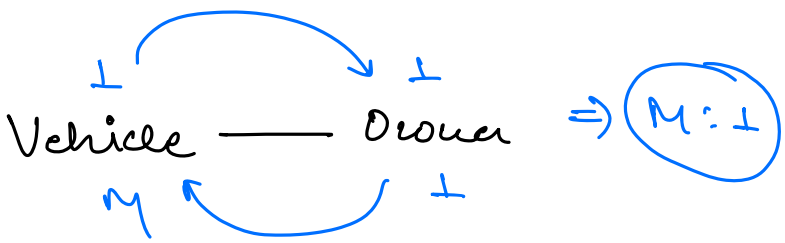


bills

Payments

Vehicles

id	number	ownerid



Vehicle — VehicleType

① String values.

Vehicles

String

id	number	Owner_id	type
1	KA01M1234	100	SEDAN
2	HR16X1234	200	Hatchback
3	———	501	SUV
4	———	420	EV

SUV → XUV

Cons

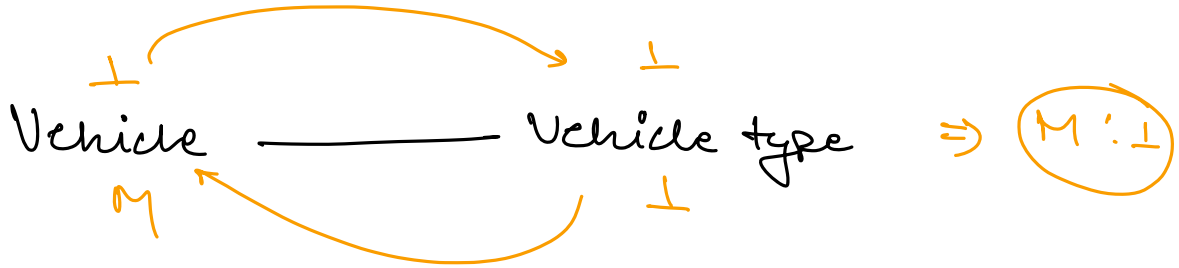
1) huge space wastage
2) Chances of typo.
3) Making any change will be difficult.
4) String comparisons are costly.

⇒ Store it as a mapping table.

⇒ for every enum create a separate table with only two columns id & value.

vehicle_types

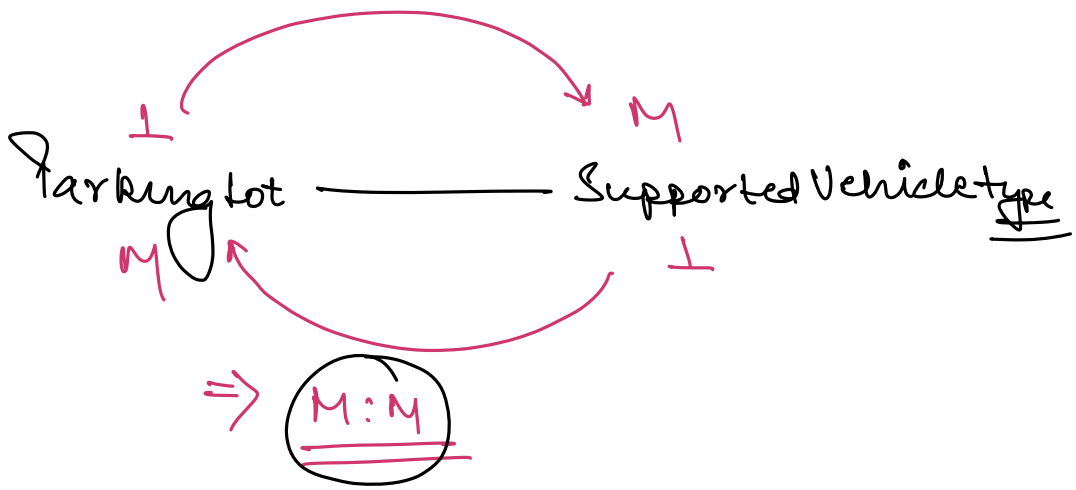
id	value
1	Hatchback
2	SEDAN
3	~~SUV~~ XUV



Vehicles

SUV → XUV

id	number	Owner_id	Vehicle_type_id
1	KA01M1234	100	1
2	HR16X1234	200	1
3	———	501	2
4	———	420	3
			⋮



parking_lot_vehicle_types

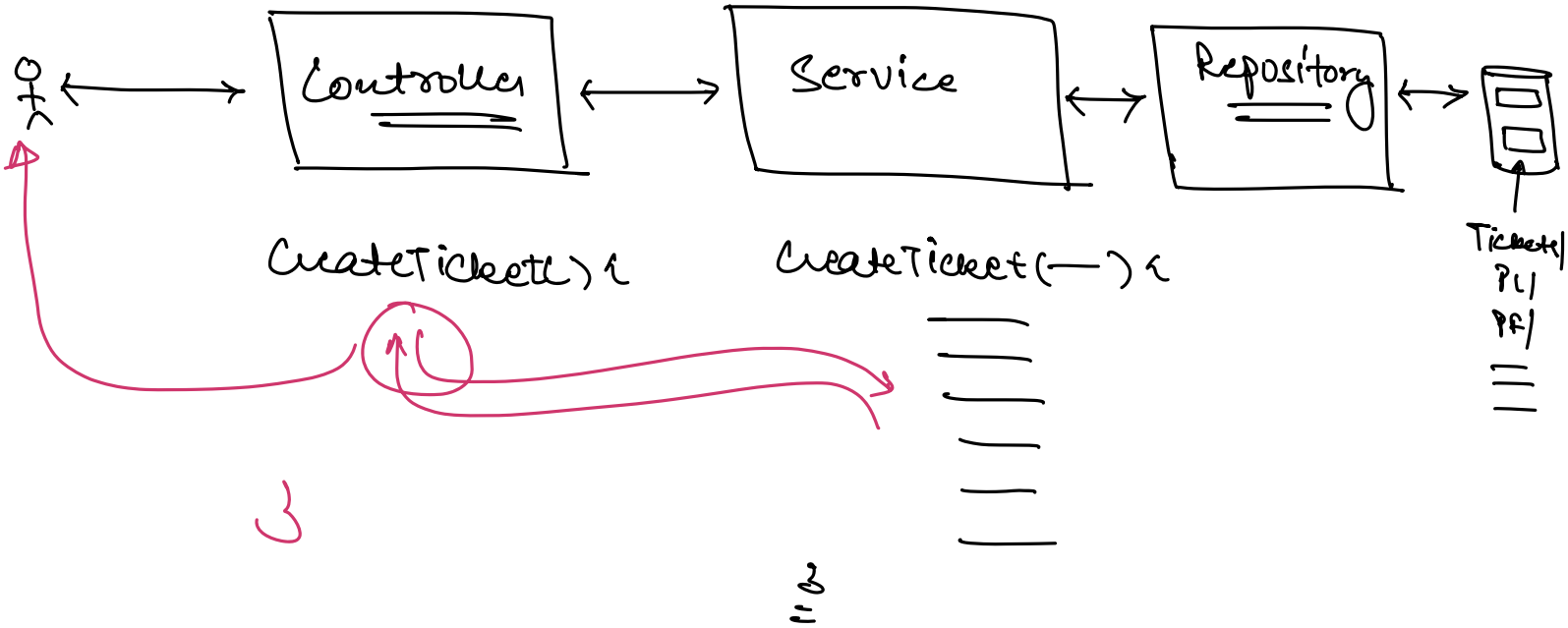
parking_lot_id	vehicle_type_id

Code:

⇒ MVC.

View : UI

Controllers : Entry point of our app".



⇒ Code Models.

Entities | Classes that we have come up in the Class diagram.

createTicket(——){

}

TicketController {

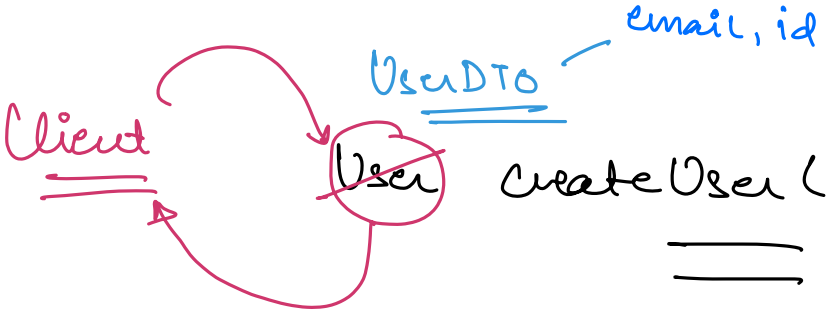
Ticket createTicket(——){

——
——
——
——

}

}

UserController {



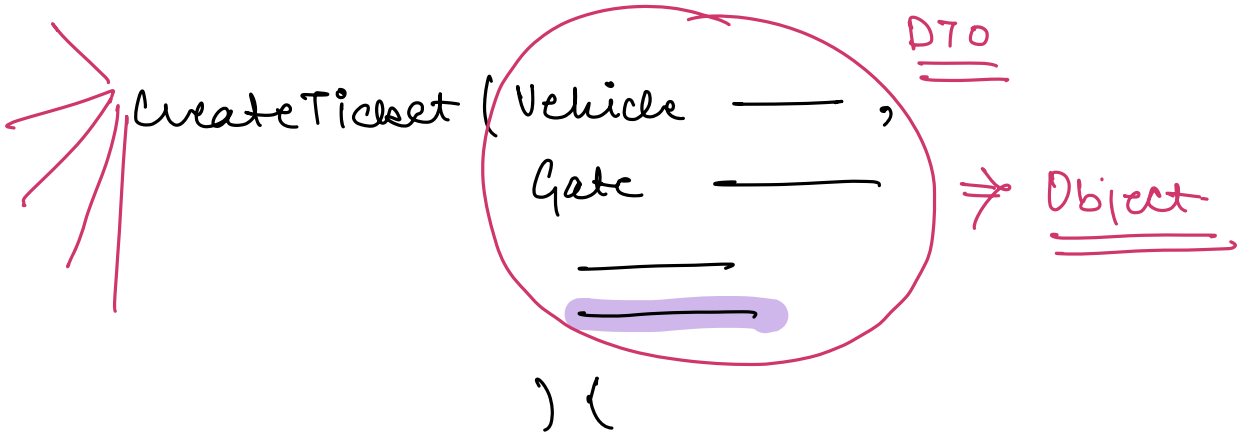
createUser(——){

——
——
——
——

}

}

⇒ DTO : Data Transfer Object.



3

i) Addition of param.

⇒ If we want to add a new parameter in the input for a API

ii) Non mandatory parameters.

⇒ for input params as well we can create DTO's.

CreateTicket ResponseDTO CreateTicket (CreateTicketRequestDTO dto) {

}

CreateTicketRequestDTO {

Gate gate

Vehicle vehicle

}

}

NOTE: In Controllers, for input / output we'll use DTO's.

10:45 pm