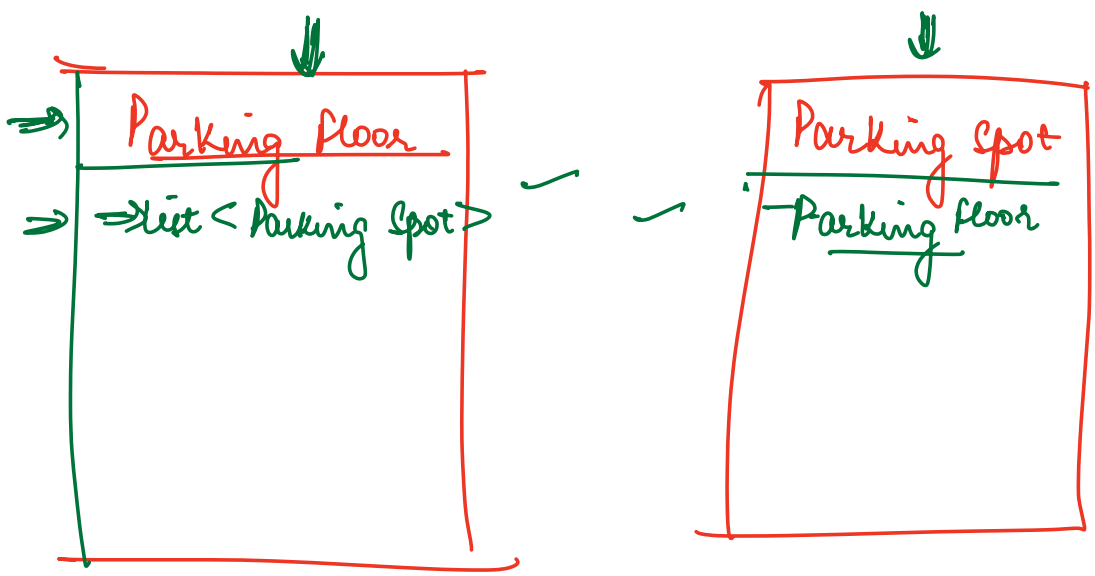
Agenda

1. Class Diagram of PL (leftover part)
2. Schema Design of PL
   → How to approach LD in a Machine Coding problem
3. Start code of PL
   → Code Models of PL

Class Diagram of Parking lot

How to decide in which class to put associations.



1. List < Parking Spot > in Parking Floor
2. Parking floor in Parking Spot

```
ParkingFloor pf = new ParkingFloor()
ParkingSpot ps = new ParkingSpot()
    pf.spots.add(ps)
    ps.floor = pf
```

⇓

③ Both

Double : ✓

Good : ✗

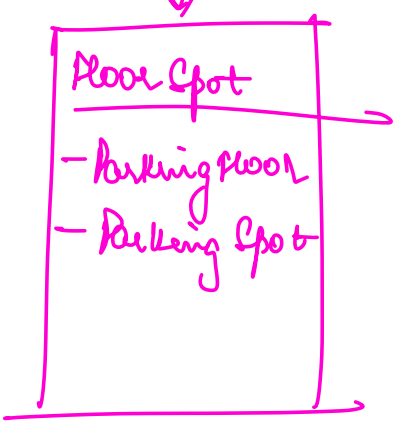
↳ Redundancy

↳ Inconsistency

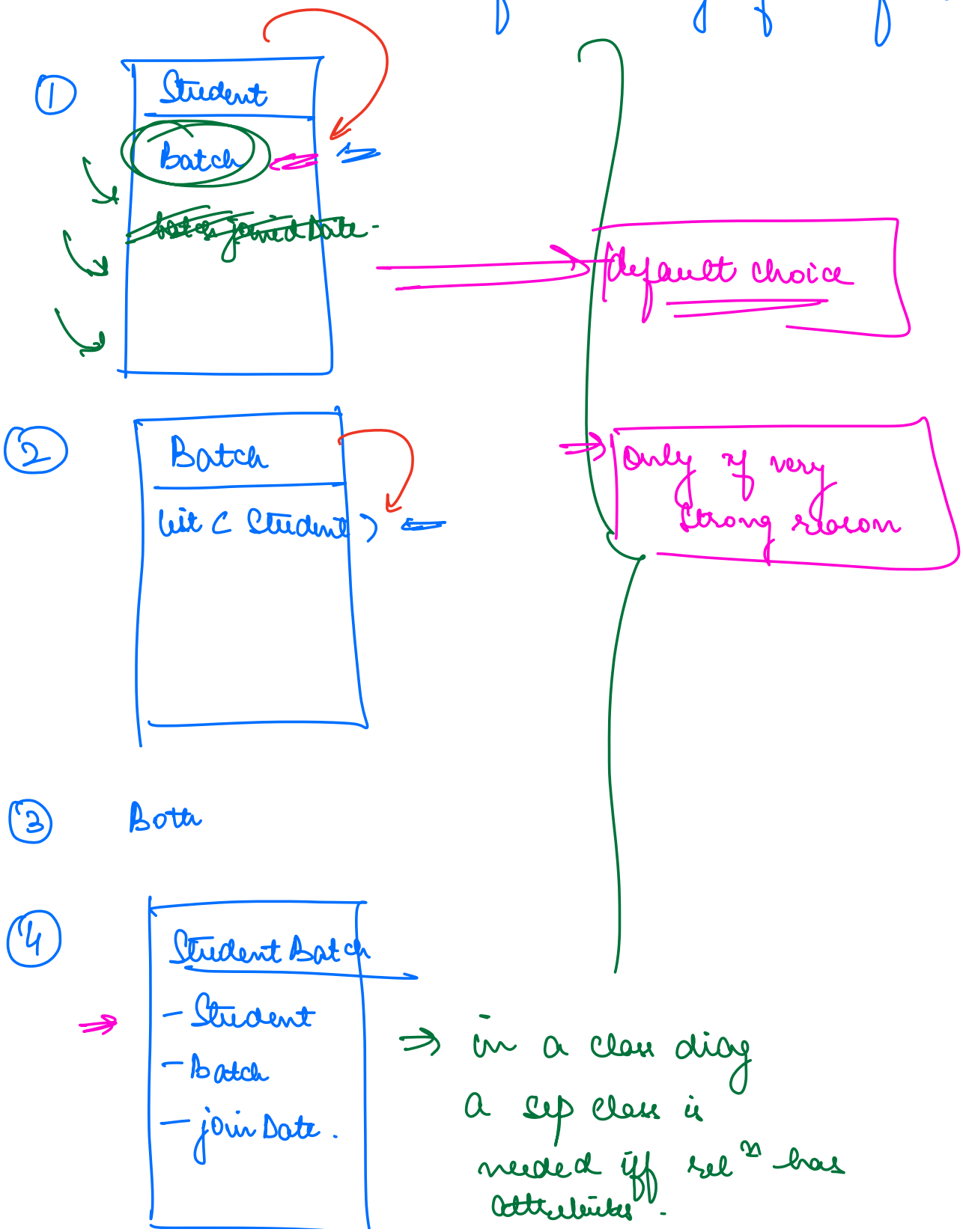
⇒ Problematic

⇓

④ Separate Class



Q^n What if for every spot, I want to store its distance from entry of its floor.



⇒ attrs of a rel^n store in rel^n class else it violates SRP.

Myth ⇒ But in a table, I can't store a list

⇒ Class Diagram ≠ Schema Design

Differences

→ In class diagram, every entity has object of other class.
→ Whereas in Schema design we have foreign keys (id)
→ In a class diagram I can store lists while in a table I can't.

⇒ When we build a real system, the work to convert a table to a class is handled by ORM (Object Rel^n Mapping)
→ ensures no redundancy
→ ensures no inconsistency.

⇒ Ideally choose which way to store assoc^{n} via access patterns.

" is there a sit^{n} where if I have a Student obj I might need their batch"
↳ get batch in Student class.

By default : @Lazy loading

① The best idea is to not have List < > attr in any model.

Batch b = db.getBatch()

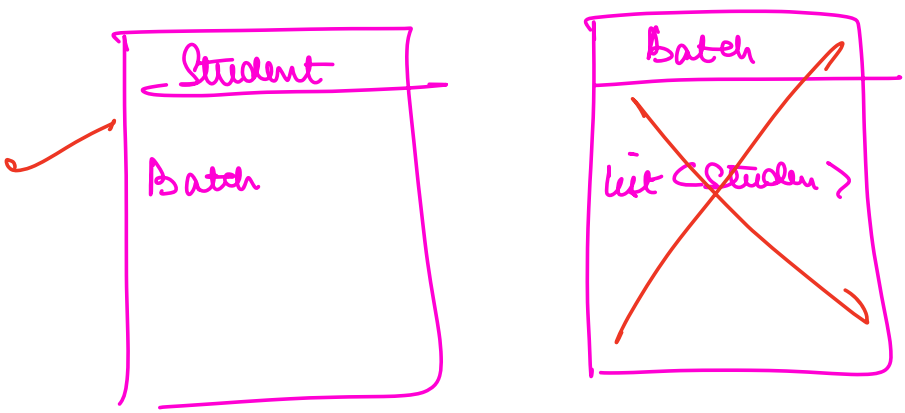
⇒ b.getStudents()

{ the query will be executed only when you run this line }

{ ② If a rel^{n} has attributes, rep via a mapping class. }

SUMMARY

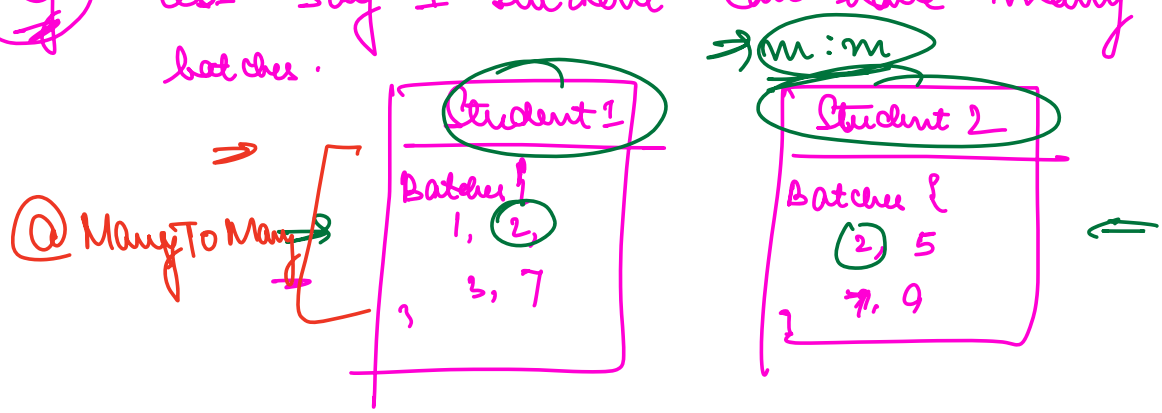
* If 1:1 / 1:m / m:1 ⇒ attribute on one of the sides.



* If m:m rel^{n} ⇒ we can't avoid list on any of the sides.

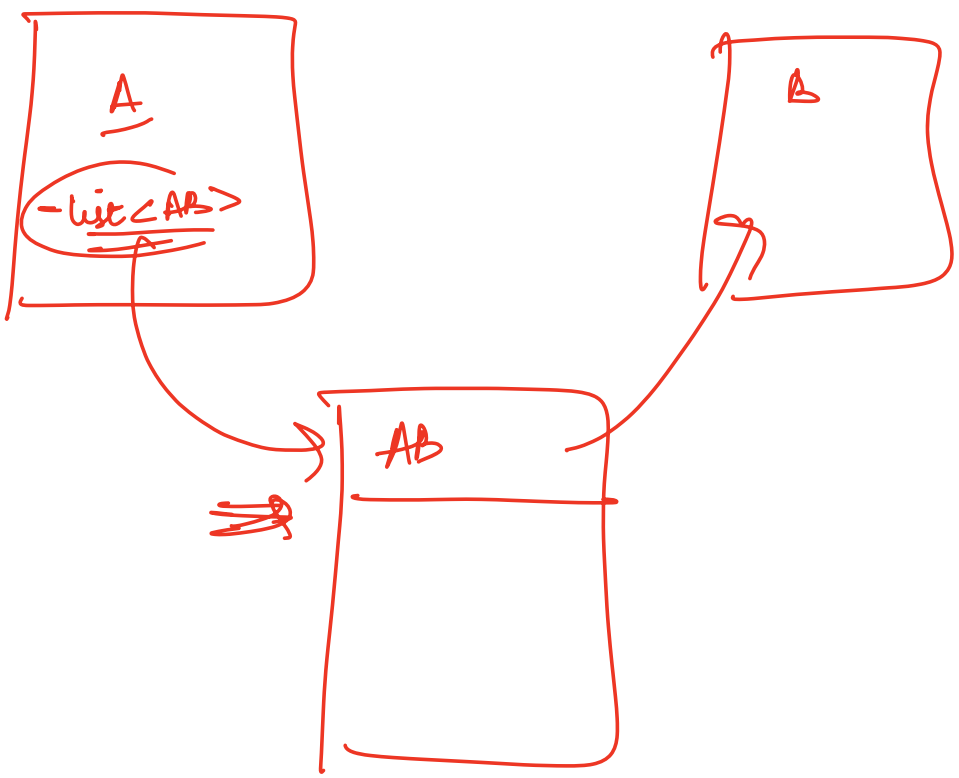
↳ Put a list on any of the sides!!!.

Eg Let's say 1 student can have many batches.

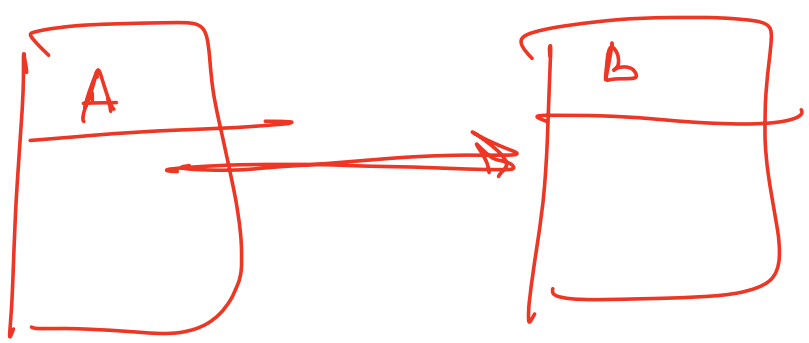


✳ What if rel^{n} has attrs =

⇒ Go for mapping class.



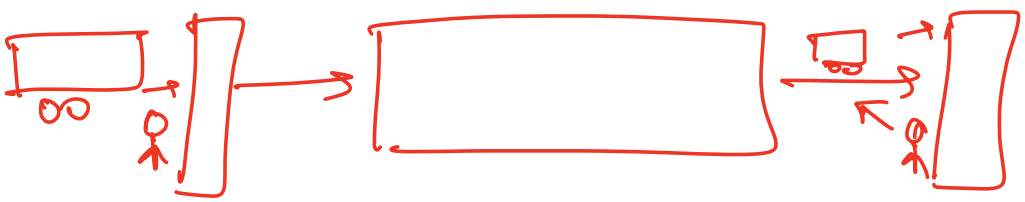
V/S

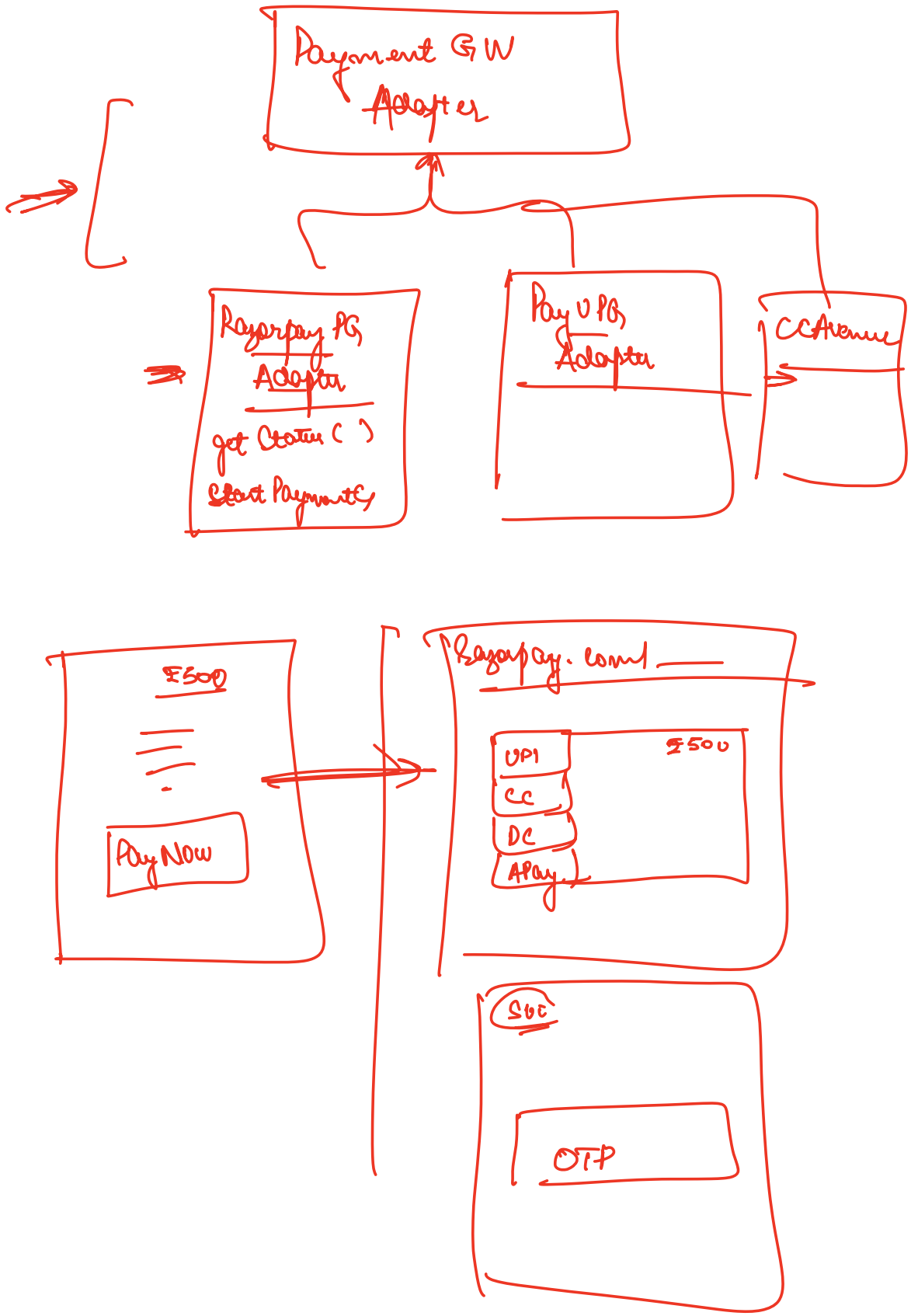


→ Schema design ≠ Class diagram.

⇒ Even if we rep m:m via a list on 1 of the sides, in table it will be rep via a mapping table.

Design Patterns in Parking lot




Payment GW
Adapter
Razorpay PG
Adapter
getStatus()
startPayment()
PayU PG
Adapter
CCAvenue
₹500
PayNow
Razorpay.com
UPI
CC
DC
APay
₹500
SBC
OTP

+Factory for creating Adapter / Strategy.

Payment GW Req Builder
- setCard()
- setAPI()

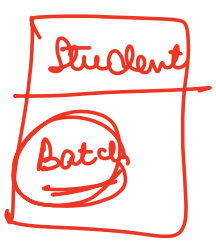
//Builder ⇒ Create
// HTTP
/ Request

Schema Design of Parking lot

Create Schema design from the class diagram.

① For every class (resp entities) that is there in your class diagram, create a table for each of those.

② For primitive attrs in those classes (int, String, date, etc) put them as it is as a column in the corresponding table.



③ For non primitive attrs:

a.) Find the cardinality of rel^n

b.) Depending on cardinality rep it:

1:1 ⇒ putting id of any 1 side on other side.

1:m
Or
m:1

⇒ id of 1 side on m side

Student
Batch.

1 — 1
St : Batch

M — 1
M: 1

Students

⇒

id	name	——	batch_id

batch

id	name	——	List<Student_id>

m:m ⇒ Mapping Table

How to rep enums

① Create new table
② just have a string

⇒

① If we store as string

parking_spots

id	number	status
1	103	AVAILABLE
		BOOKED
		BOOKED
		AVAILABLE
		BOOOKED
		BOOKED

Cons
⇒ extra storage
⇒ updates expensive.

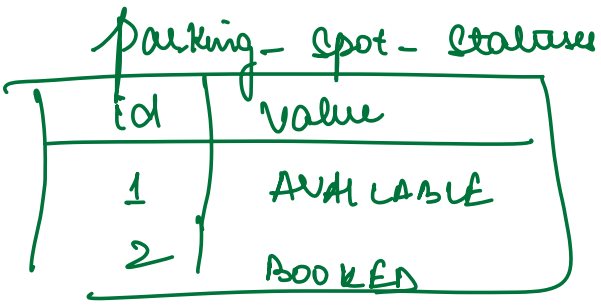
Pro
⇒ easy to understand

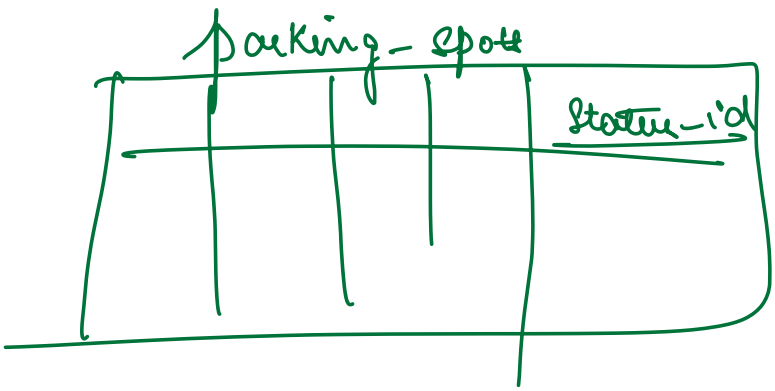
② [RECOMMENDED]

Create a table for every enum.
With 2 col^n:

id	value

for every possible value of that enum, store as a rep row in the table.





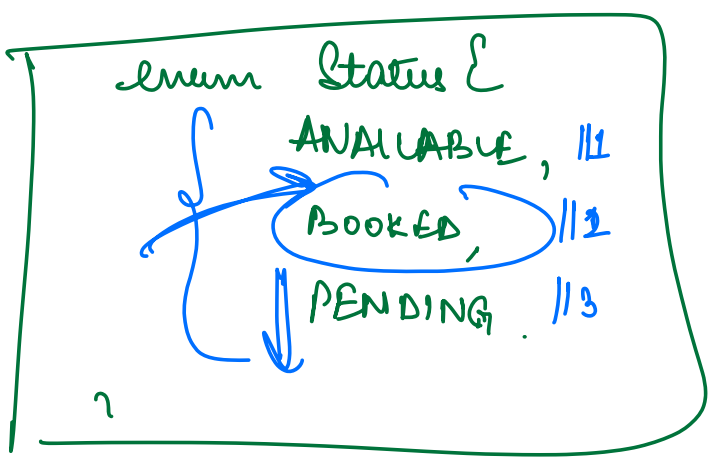
Pro

① No Extra Storage.

Con

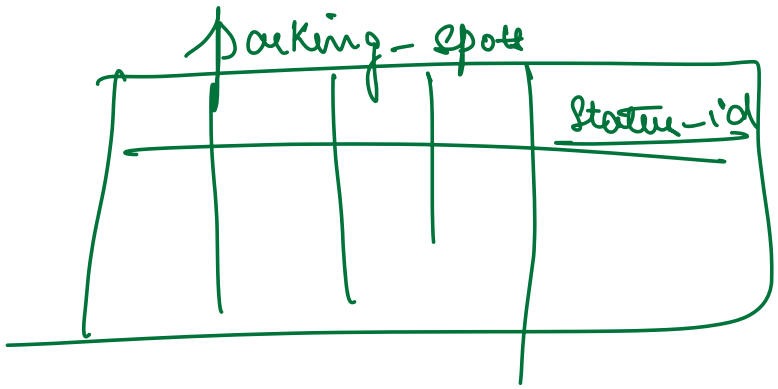
Will req join

③ No extra table



⇒ Don't even store enum in DB

⇒ Instead, assume id to be in seq of enum value.



Con:

→ enum is added in b/w

→

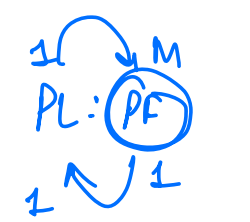
Schema design of Parking lot

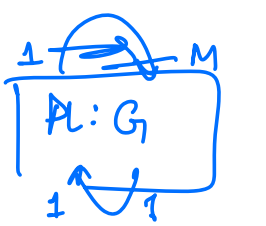
(1) For every class in class diag/ enum create table.

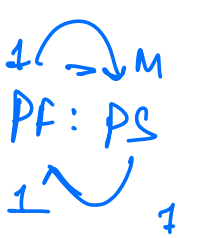
(2) Put primitive attrs of classes in corr tables.

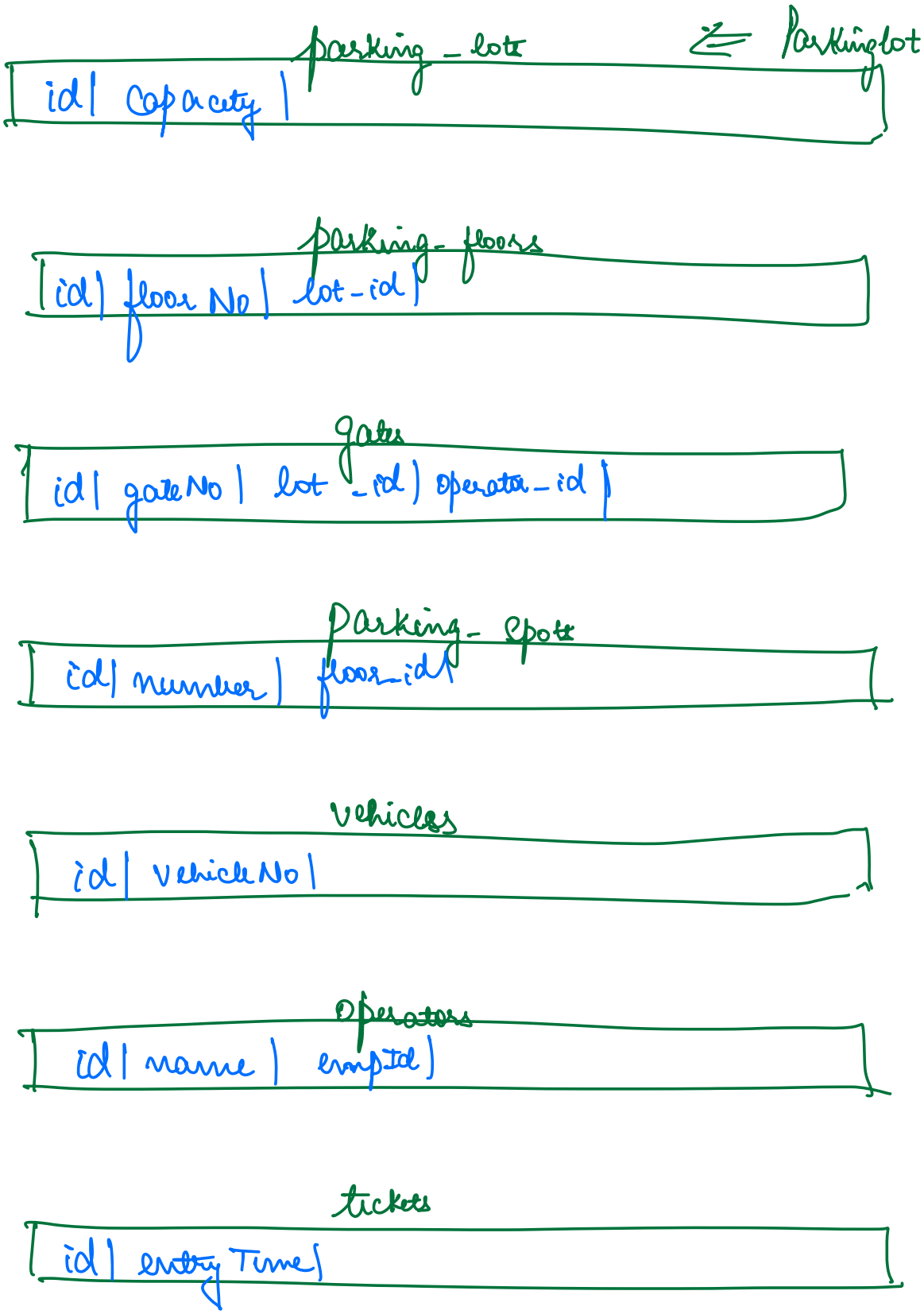
(3) rel^n b/w 2 tables

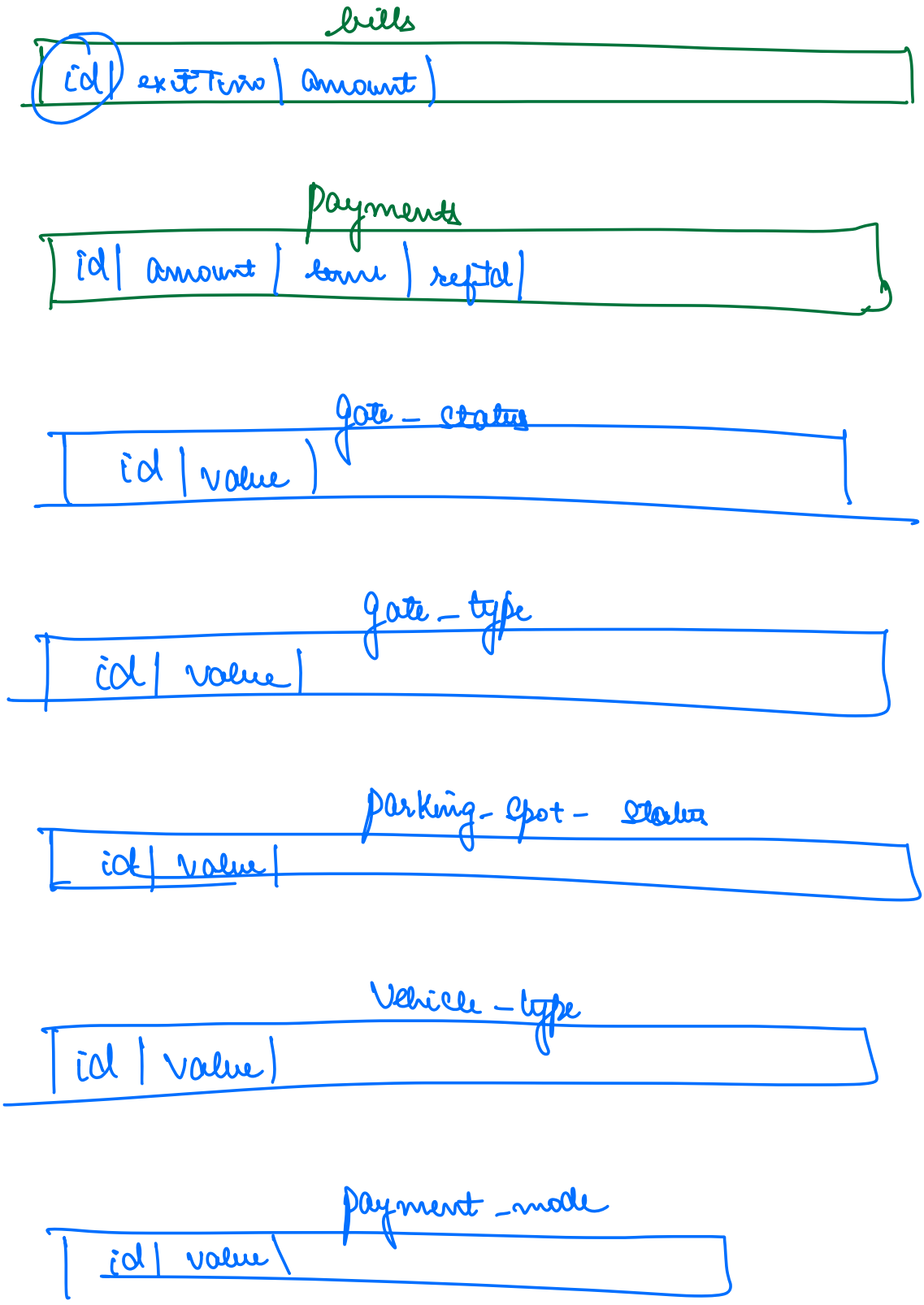
G:O

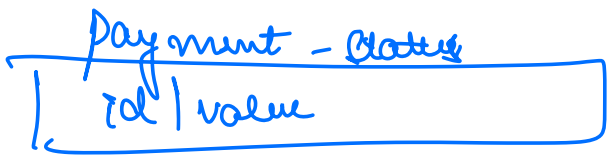


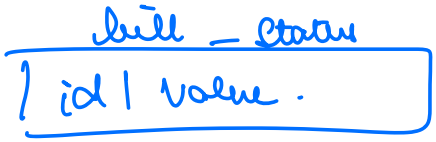


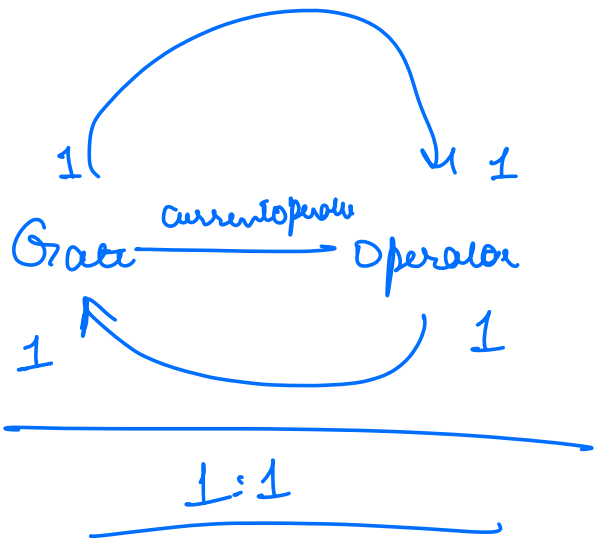


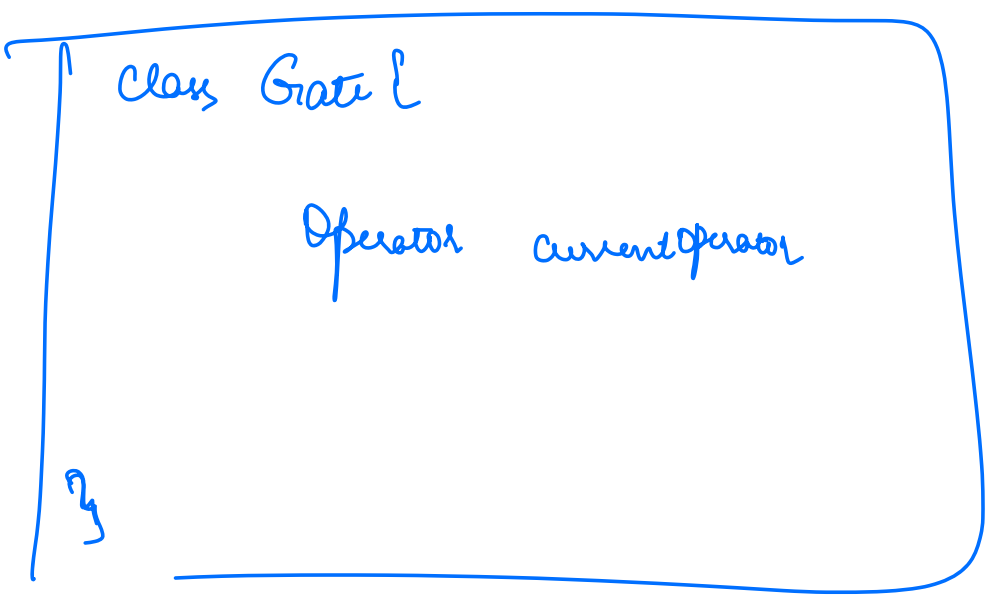

parking_lots
id | capacity |
parking_floors
id | floor No | lot_id |
gates
id | gate No | lot_id | operator_id |
parking_spots
id | number | floor_id |
vehicles
id | vehicle No |
operators
id | name | emp_id |
tickets
id | entry Time |


bills
id | exitTime | amount
payments
id | amount | time | refId
gate_status
id | value
gate_type
id | value
parking_spot_status
id | value
vehicle_type
id | value
payment_mode
id | value


Payment - Status
id | value


Bill - Status
id | value .


1
1
Gate
currentOperator
Operator
1
1
1 : 1


Class Gate {
Operator currentOperator
}

H/W

1. Complete Schema Design
2. Code Models.

... / models /

— . java
— . java

⇒ (only setters and getters/setters)

Update gates
Set operator_id = 123
where gate_id = —

Operator — duties

| op-id | st-time | en-time |