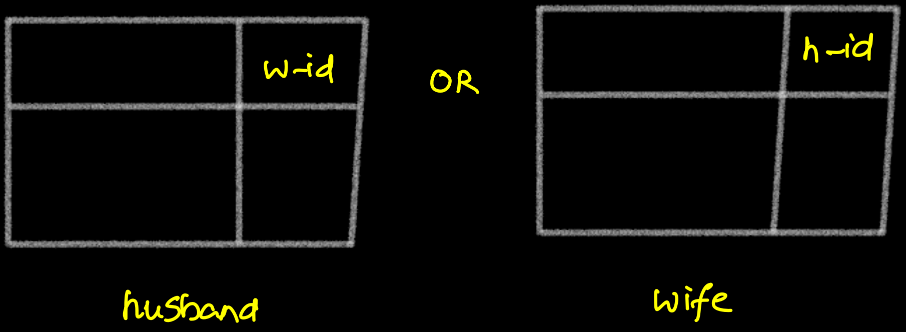
Agenda

1. Cardinalities
2. schema design
3. coding models
4. DTOs

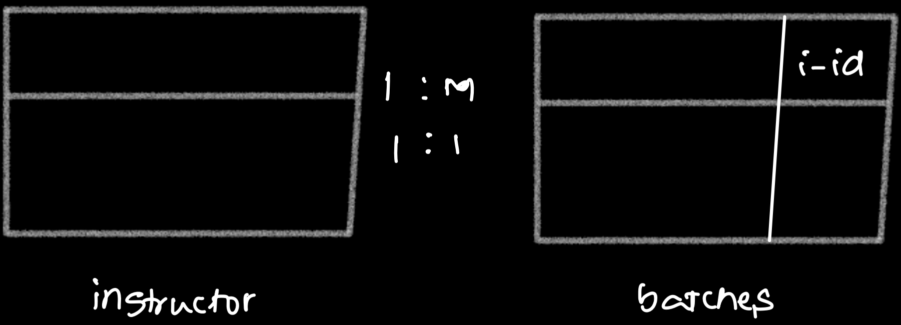
HW: create a new project and create models.

Cardinalities

1:1 : How to represent?

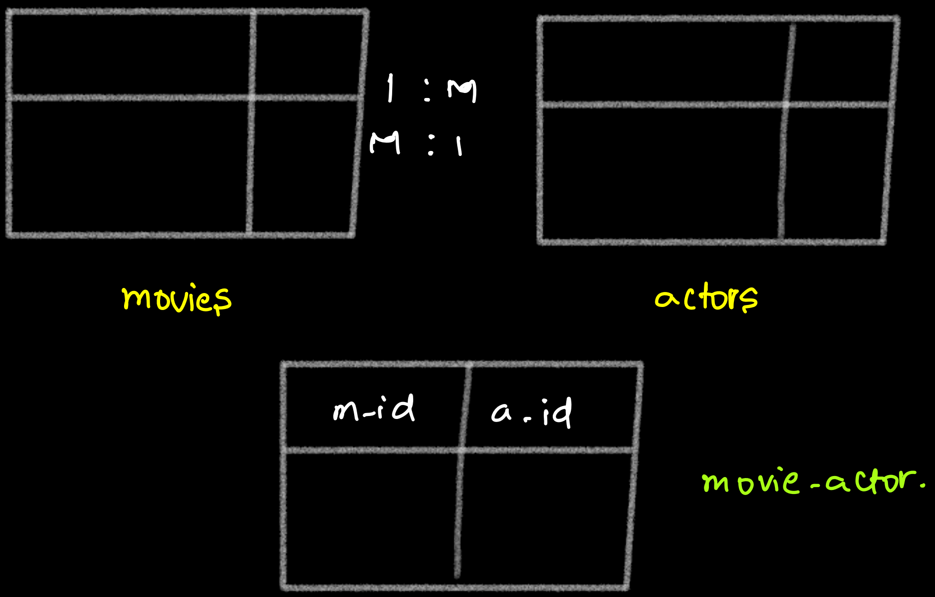


1:M
M:1 : How to represent?



Put the id of '1' side on 'm' side.

M:M : How to represent?



How to create schema diagram from class diagram.

1. For each class representing an entity, we will create a new table
2. For each primitive attribute, we add a column in the table
3. For non-primitive attributes.-
   1. Find the cardinality
   2. Represent the cardinality

Class Movie
- id
- title
- dateofRel
- List <Actor>

id	title	date

movie

m-id	a-id

movie_actor

Relations in class dia

1. IS-A relation - inheritance [we'll learn today].
2. HAS-A relation [same as cardinalities]

How to create tables?.-

①

Pen

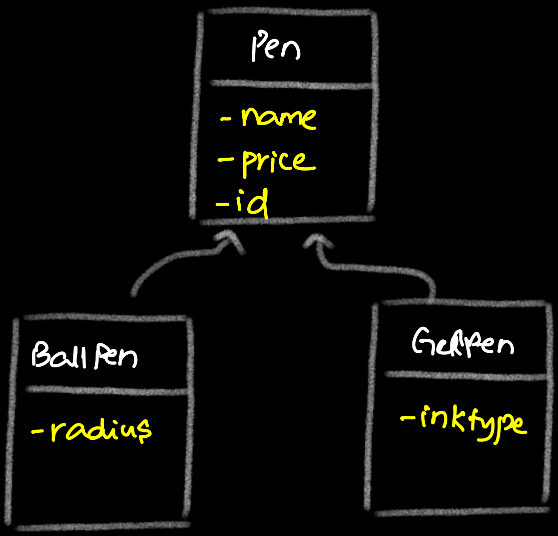
id	name	price

Ball Pen

id	name	price	radius

Gel Pen

id	name	price	inktype



Issues ; Get all the pens.

1. Iterate on all the 3 tables
2. If a new type of pen is added, it'll lead to modified queries
3. Data redundancy.

2: Preferred method.

Pen

id	name	price	type

Get me the name of all Pens. -

1. Select * from Pens

Ball Pen

radius	pen_id

Gel Pen

inktype	pen-id

How to represent enums?

ex: Player

id	name	player-type

HUMAN
BOT

} How to represent.

1. Store player type as strings

→ Easy to read
→ No joins involved

→ Error prone
→ More memory
→ String comparision.

2. create a mapping table

player-type

id	value
1	HUMAN
2	BOT

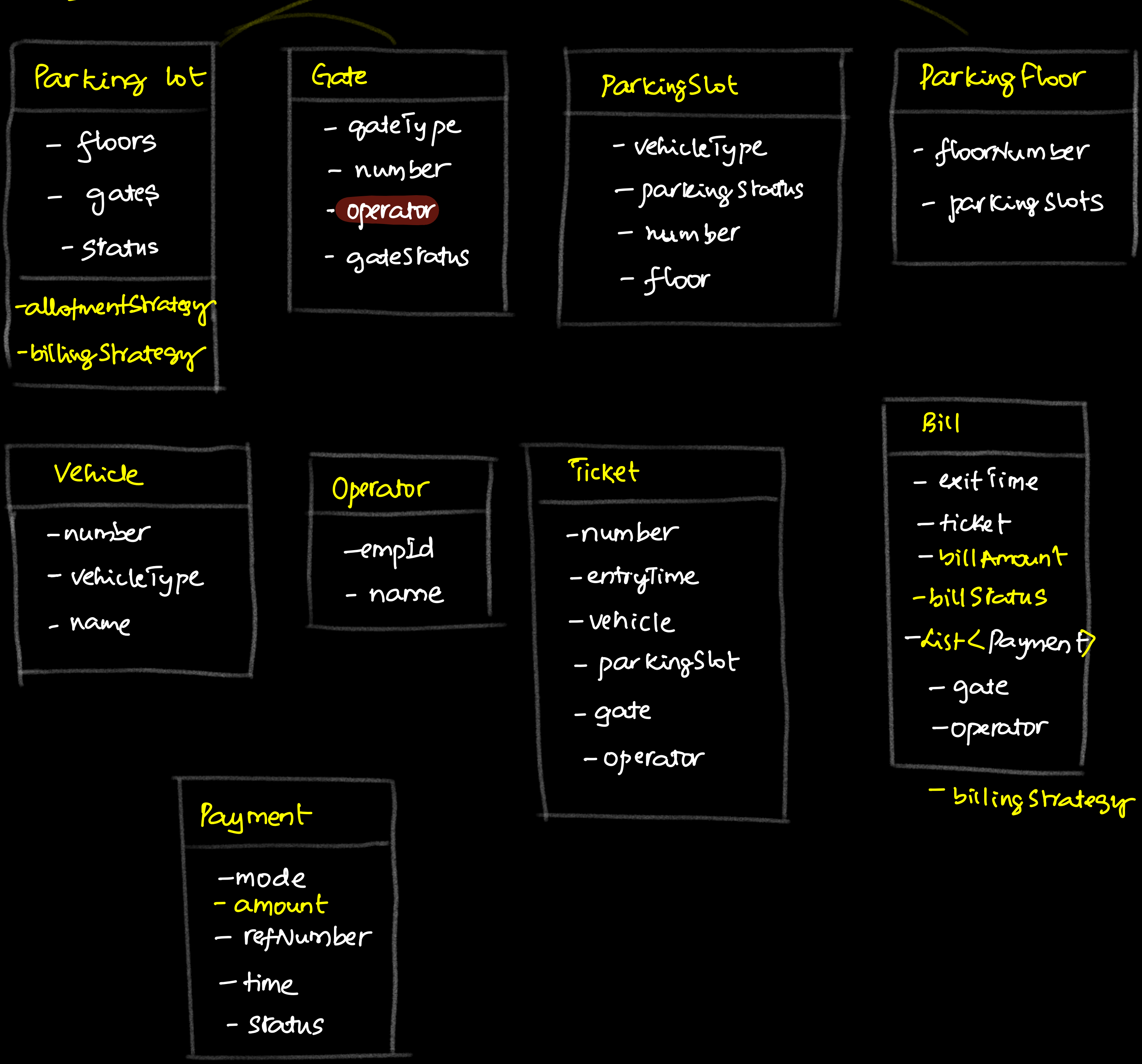
Player.

id	name	player-type-id
1	Keer	1
2	GPT	2

Cons:

Joins are involved.

Class diagram



Tables

parking_lot

parking-floor

gate

parking_spot

vehicle

operator

ticket

bill

payment

① Primitives becomes direct columns

parking_lot

id	parking-status-id	allot-strat-id	bill-strat-id

parking-floor

id	floor-number	parking-lot-id

gate

id	number	parking-lot-id	gate-type-id	operator-id	gate-status-id

parking-spot

id	number	parking-floor

vehicle

id	number	ownerName

operator

id	empId	name

ticket

id	number	entryTime

bill

id	exit_time	bill-amount

payment

id	amount	ref No	time

Relations

Parking lot	Floor
1	M
1	1

1 : M

Parking lot	Gate
1	M
1	1

1 : M

Parking lot	status
1	1
M	1

M : 1

Parking lot	allot Strat
1	1
M	1

M : 1

Parking lot	bill strat
1	1
M	1

M : 1

ParkingFloor	ParkingSlot
1	M
1	1

1 : M

Gate	GateType
1	1
M	1

M : 1

Gate	Operator
1	1
1	1

1 : 1

Gate	GateStatus
1	1
M	1

M : 1

ParkingSlot	ParkingFloor
1	1
M	1

M : 1

Ignore already handled relations.

Rest is homework.

parking-status

id	name

allotment Strategy

id	name

billing Strategy

id	name

Gate-type

id	name

Gate-status

id	name

DTOs

issue Ticket

TicketController.

public ________ issueTicket (________ input)

userDetails, gateDetails, opeDetails -- etc.

create DTOs → Data Transfer Objects.

POJO → Plain old java object.

Depending on what UI needs, you'll create a DTO.

Ticket Issue Response

- ticket Id
- user Id
- vehicle Id
- message
- error