+ methods
+ Cardinality

Agenda

1) Schema Design
   ⇒ Inheritance
   ⇒ Association
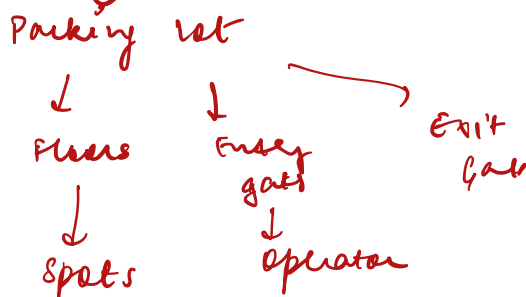
2) Models ⇒ coding

3) Controllers → user story
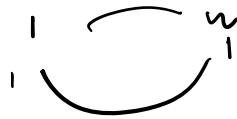
DTO

4) Repository
   ORM
   In-memory
   DB
   spring
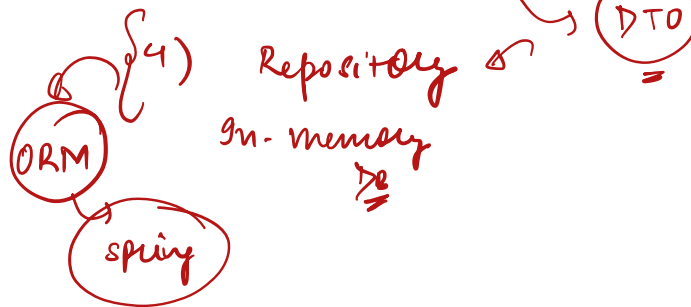
Schema Design

1) For every class ⇒ create a table
   in class dia

2) For all primitive ⇒ Add a column
   attributes   in the table

3) For all non ⇒ → Find cardinality
   primitive   → depending on
                    cardinality
                    represent it.

Parking lot

↓   ↓
Floors   Entry gates → Exit Gate

↓   ↓
Spots   Operator   Vehicle

DB   1 ⌢ n
                      1   1

parking_lot

id	name			address

parking_floors

id	floor_no.	parking_lot_id	status

Parking Spot Status

id	name
1	OCCUPIED
2	BLOCKED
3	AVA

parking_spot

id	row	col	parking_floor_id	status

ENUM

status
    id
OCCUPIED

Inheritance

Pen   id
             name
             price

Ball Pen   Gel Pen

ball radius   ink type

pens

id	name	price	pentype

1
2
3
...

ball pen

pen_id	ball_radius

ball

PK

3
5

gel pen

pen_id	ink type

2
4
6

pen

ball pen
id name price
ball radius

independent

gel pen
id name pric
ink type
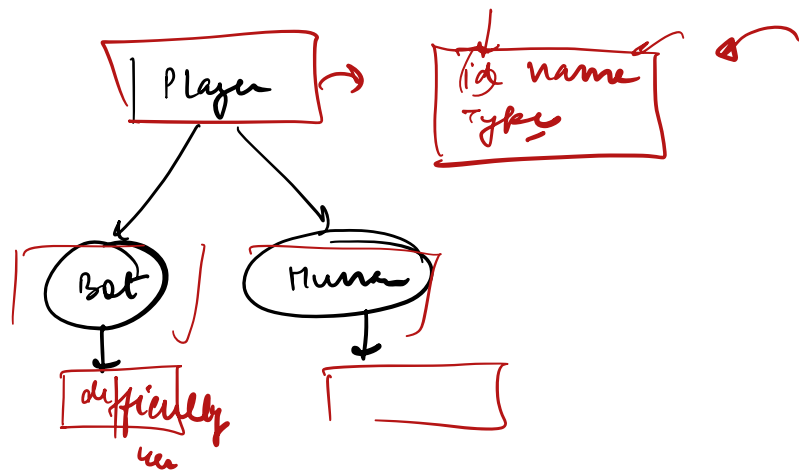
UNION
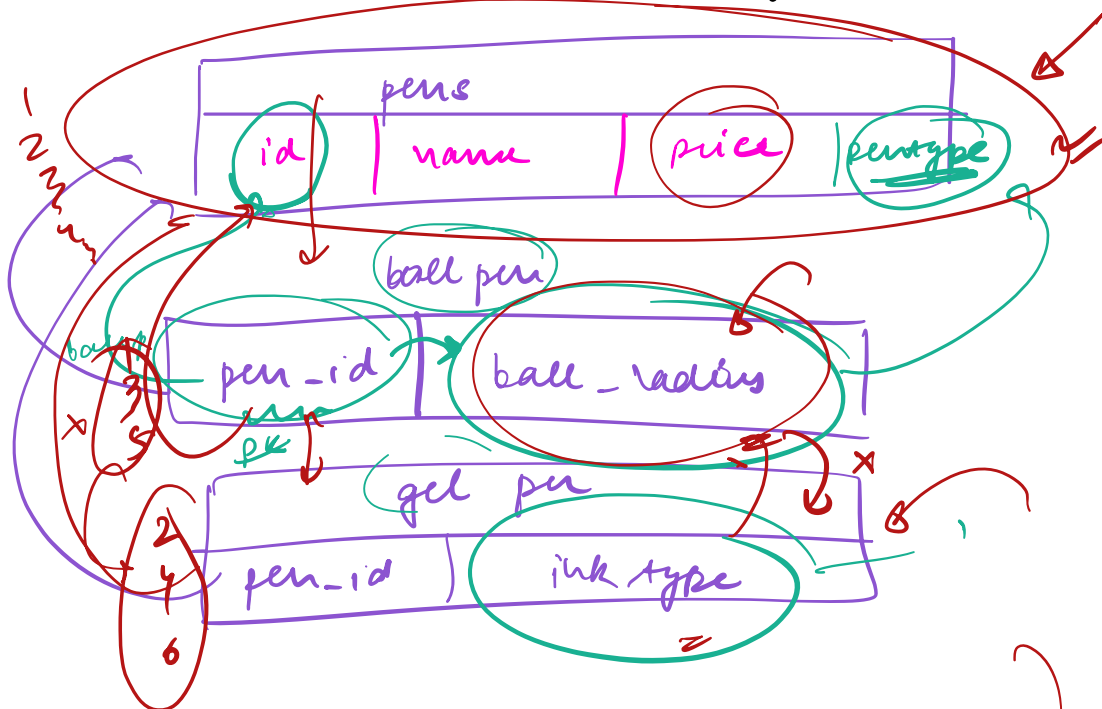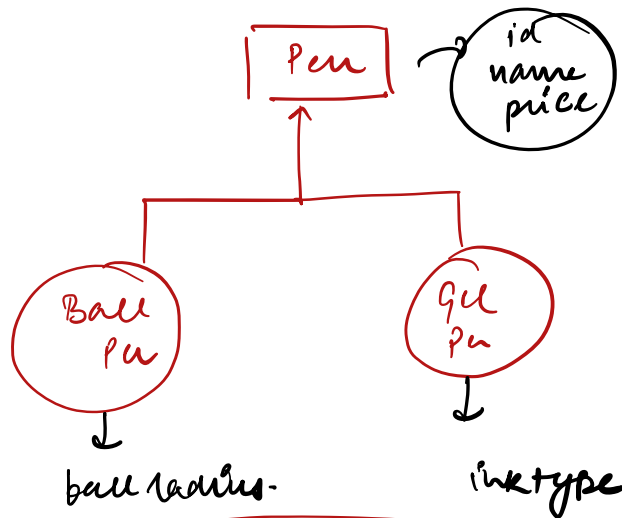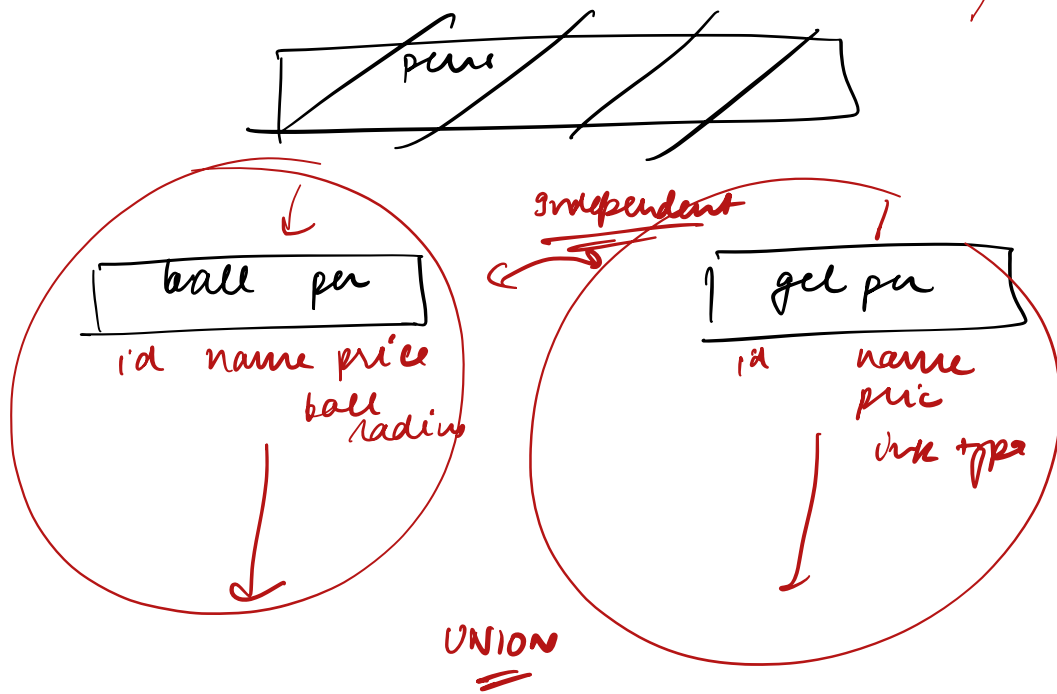
1) Create a table for parent class

2) All common attribute :) parent table

3) Remaining specific child class attributes :) children table

Player → id name Type

Bot
difficulty
...

Human

(Ticket)

(CRUD)

Ticket controller {

(Issue Ticket Response DTO) == {Issue Ticket} (_____) {

DTO →

Issue Ticket Request DTO

DTO

}

}

Return type and parameters?

DTO :) Data Transfer Object

(Prejince)

Issue Ticket (vehicle no, type, _____)

(Ticket)

issue Ticket (dto) → { class dto {

vehicle

type

parking controll. issue Ticket ("6536" , "CAR")

parking controller . issue Ticket (Dto. builder)

dto. get vehicle type

cet. _____

10:15 pm