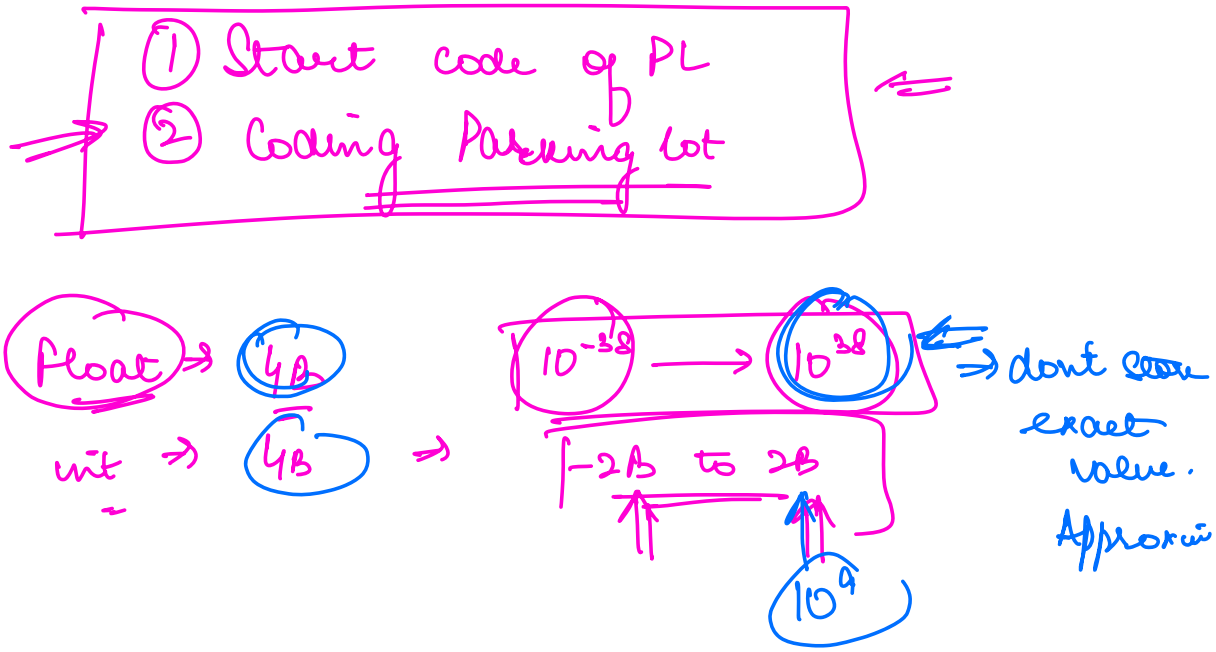
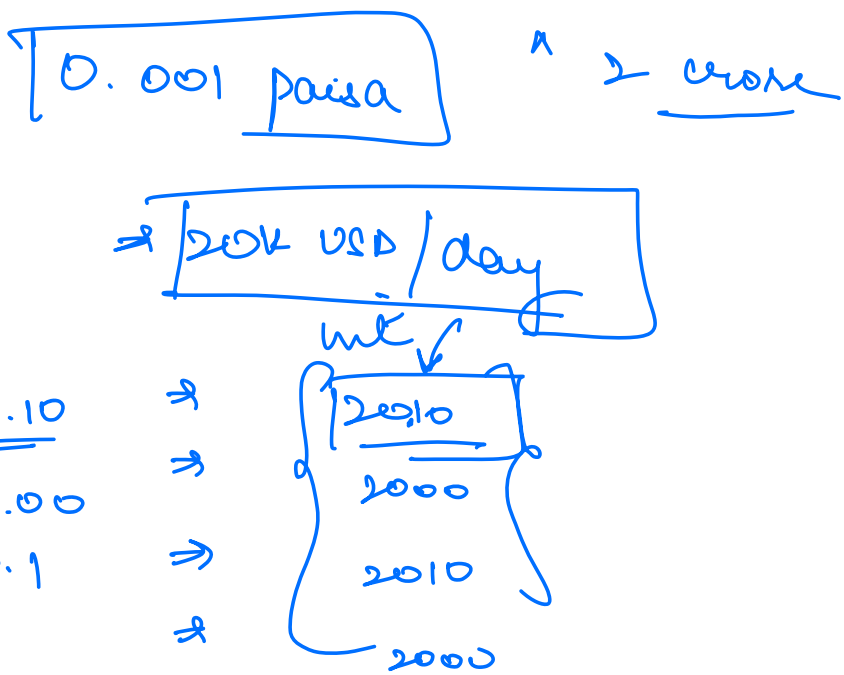
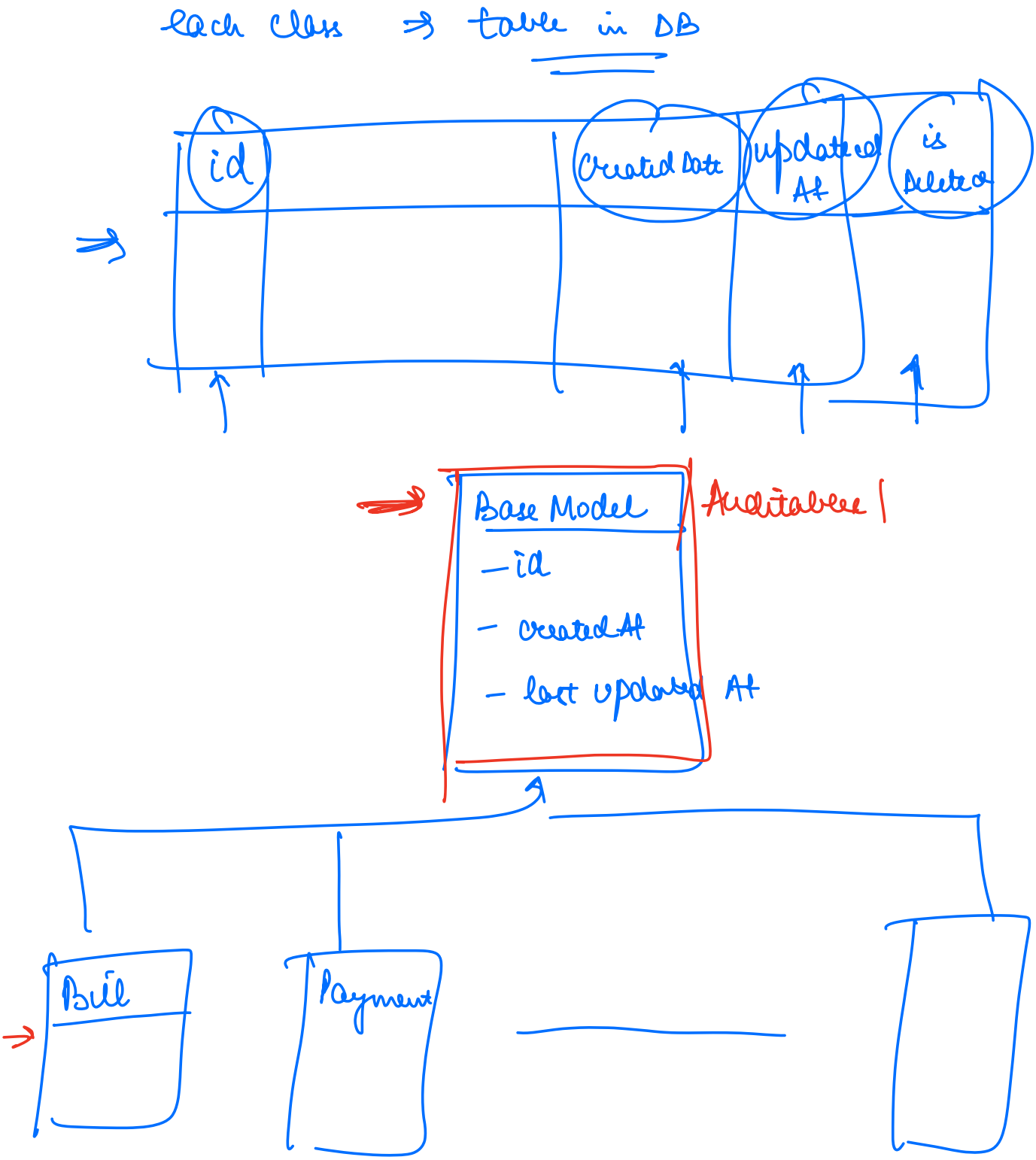
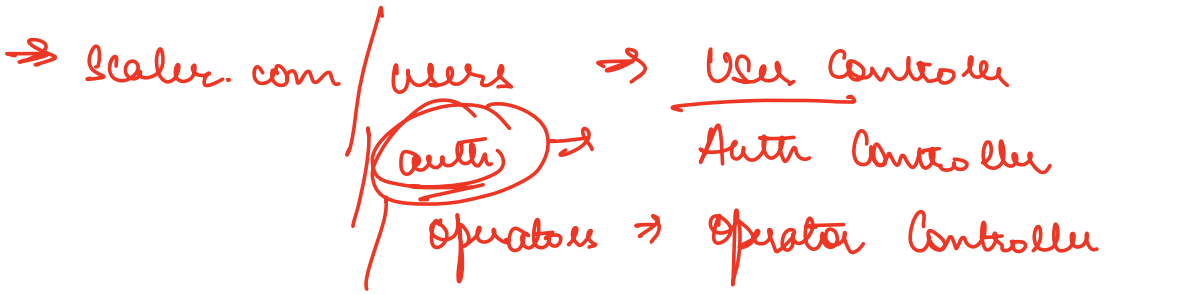

① Start code of PL
② Coding Parking lot
float → 4B
int → 4B
10^{-38} → 10^{38}
-2B to 2B
10^9
⇒ dont store exact value. Approx

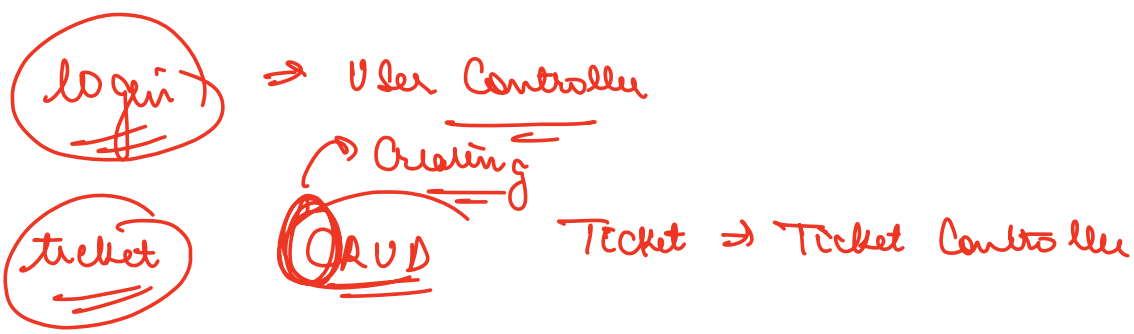

0.001 paisa
1 crore
Google
⇒ 20K USD / day
int
$ 20.10 ⇒ 2010
$ 20.00 ⇒ 2000
$ -20.1 ⇒ 2010
$ 20 ⇒ 2000
2009 == 2010
every prog lang stores floats like this
IEEE 754
Aashish ≠ Ashish


Each Class ⇒ table in DB
id
Created Date
Updated At
is Deleted
Base Model
Auditables
- id
- created At
- last updated At
Bill
Payment

Break till 10:12PM

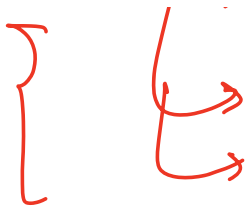


Rule of Thumb { Name the controller based on what entity there is a CRUD opⁿ on }



User Controller

User login(String email, String pass)
- placement kota
- internal Scam
- plagiarism Scam
- ...



Never return internal models outside.

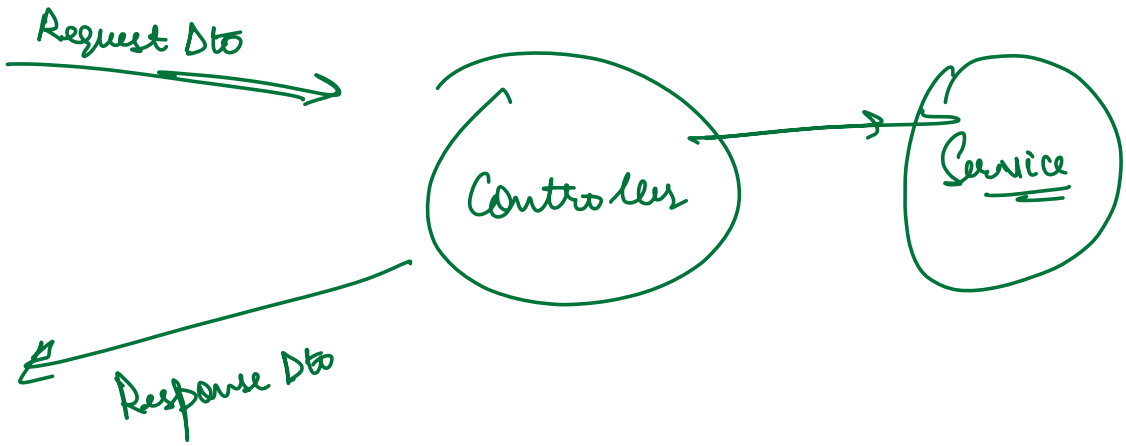
DTO : Data Transfer Obj

→ Objects / Classes that are used just for comm for client → controller

And controller → client

Controller

→ it receives request DTO
→ it gets attr from DTO
→ it calls services with those attr
→ it gets return from service
→ it creates response dto
→ return dto to client



TicketController {

GenTicketRespDto generateTicket(GenTicketReqDto);

}

Never take DTOs

TicketService {

genTicket( );

}

SCALER

UserController {

LoginRespDto login(LoginReqDto) {

UserService.login(——).

}

}

UserService {

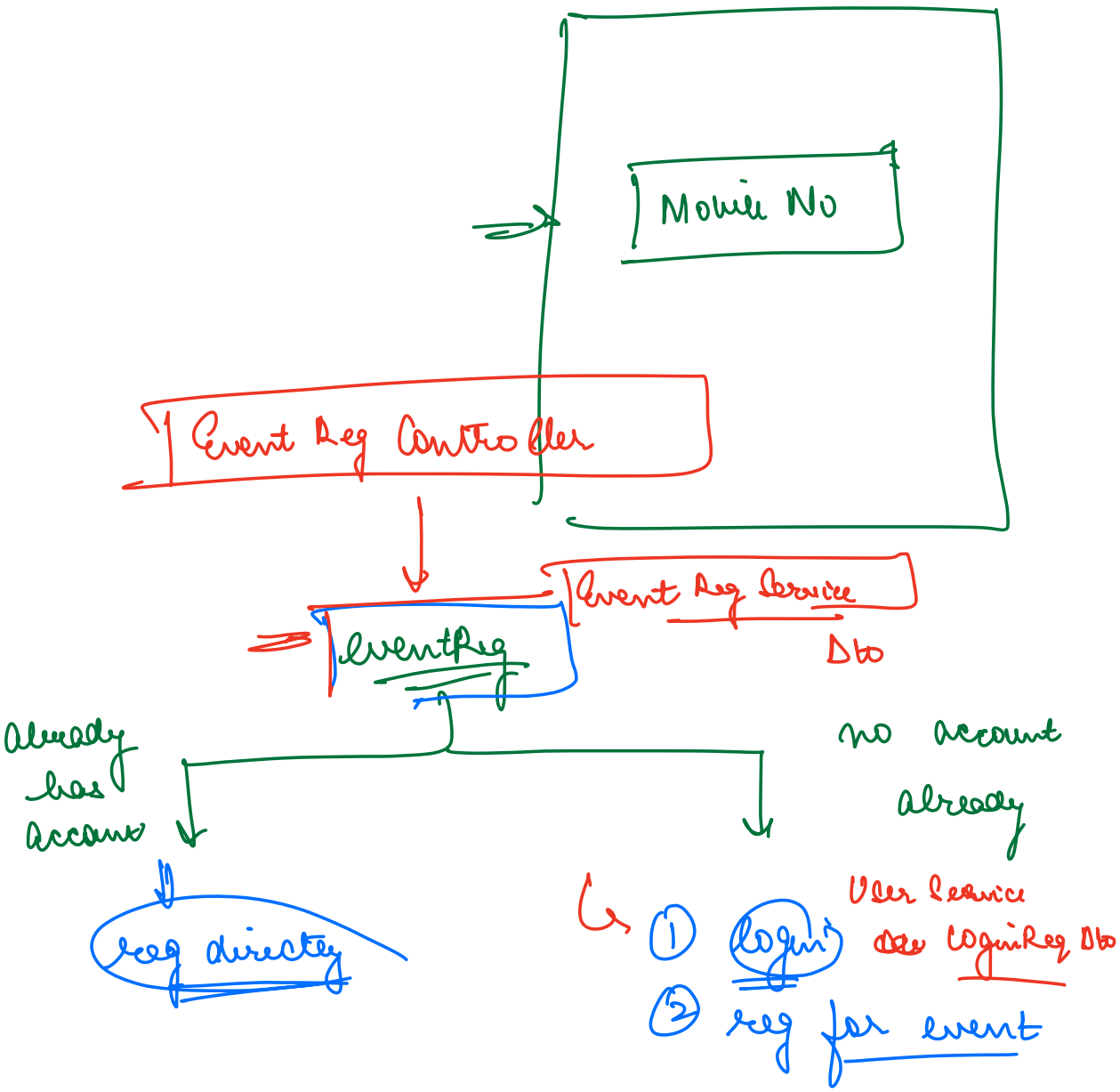
(A) login(——(B)——) {

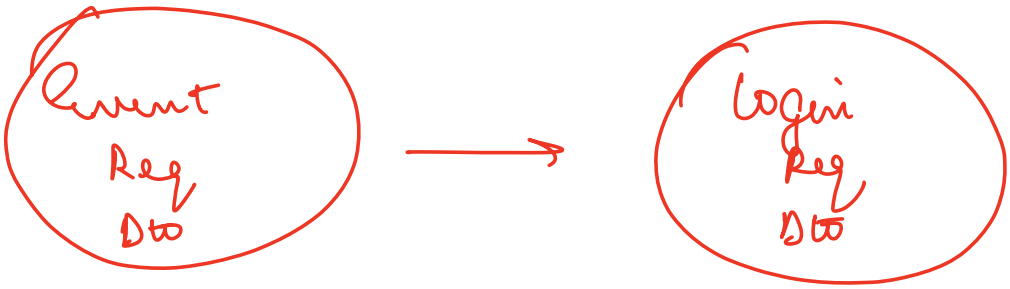
→

}

}

When a new person comes to Coder
(someone who doesn't already have
an account)

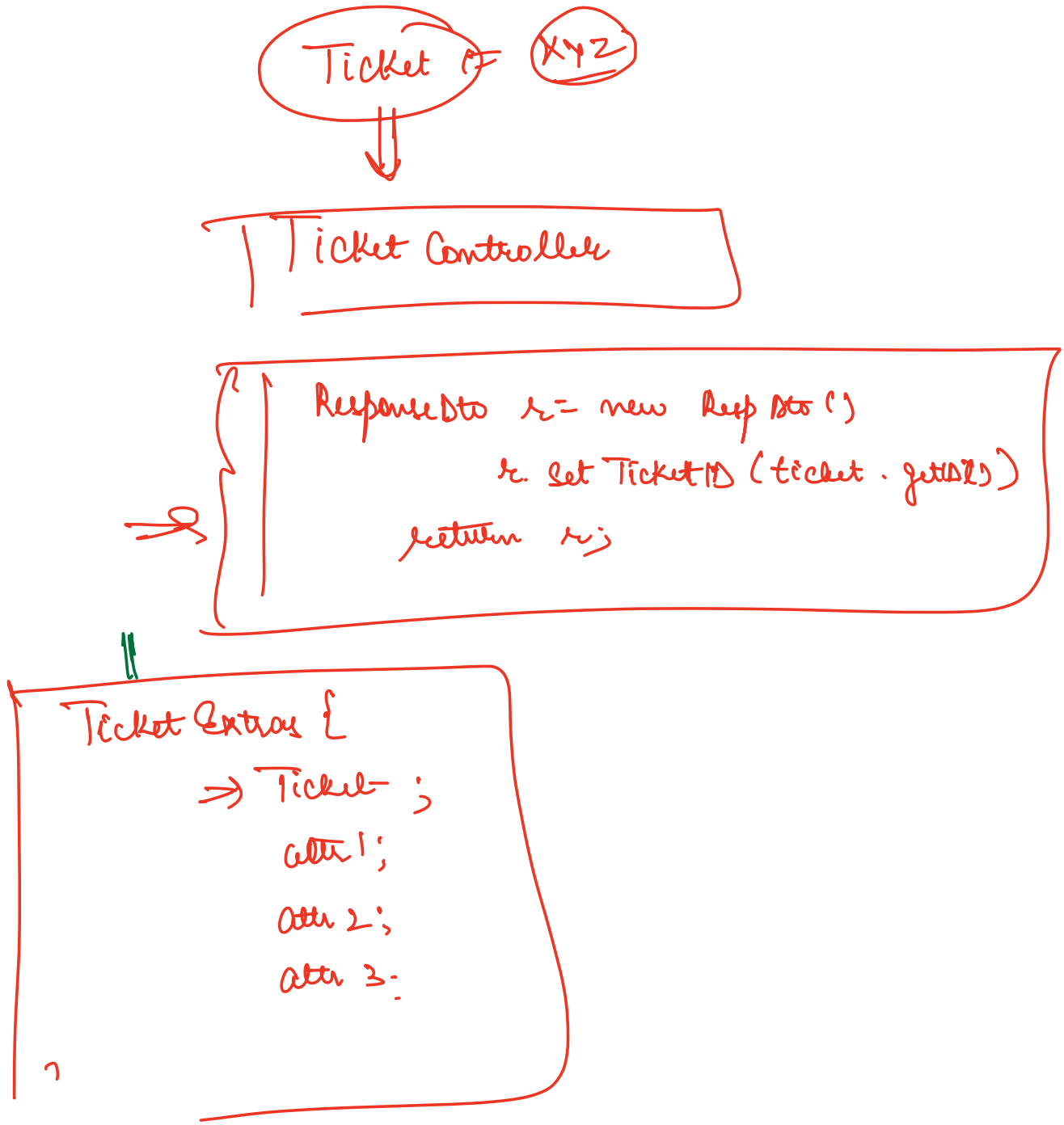


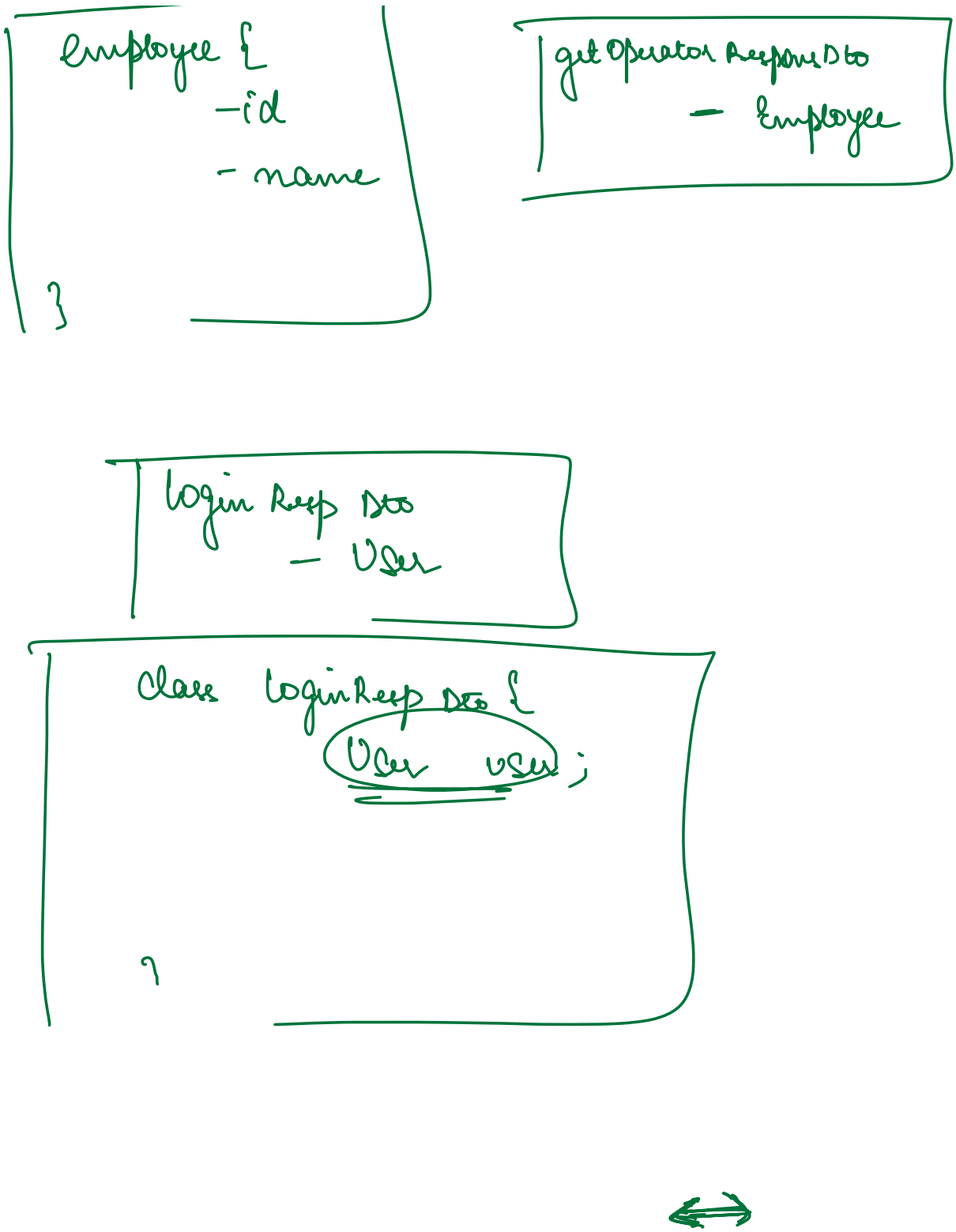


If 1 Service can be used by many Controllers eg login Service at Scaler is used from multiple places

it doesn't make sense to hard code params of a Service to a Dto

- ⇒ Service should always take real obj
- Only controllers should take Dto.


Ticket
XYZ
Ticket Controller
ResponseDto r = new RespDto()
r.setTicketID(ticket.getID())
return r;
TicketExtras {
Ticket ;
attr1;
attr2;
attr3;
}


Employee {
-id
- name
}
getOperatorResponseDto
- Employee
loginResponseDto
- User
class loginResponseDto {
User user;
}

Ticket Controller {

→ createTicket (CreateTicketReqDto)

getTicket (GetTicketReqDto)

updateTicket (

}

Ticket Service {

{ CreateTicketRespDto createTicket (Show, List<Seat>) }

}

}

Signup Controller {

signup (SignupReqDto) }

}

Signup Service {

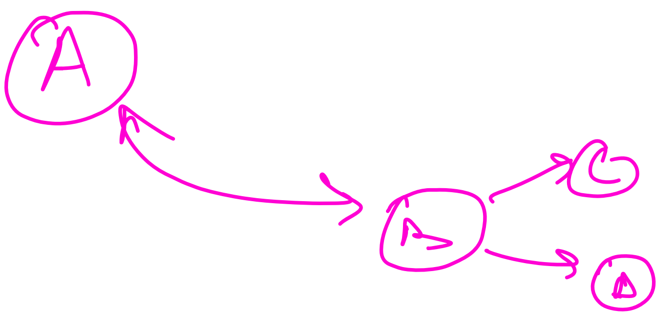
Signup (Signup Req Dto

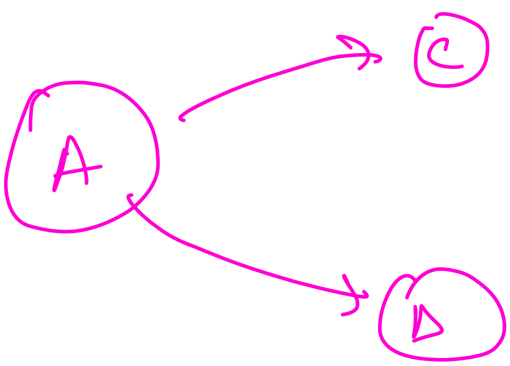
Create Ticket Dto

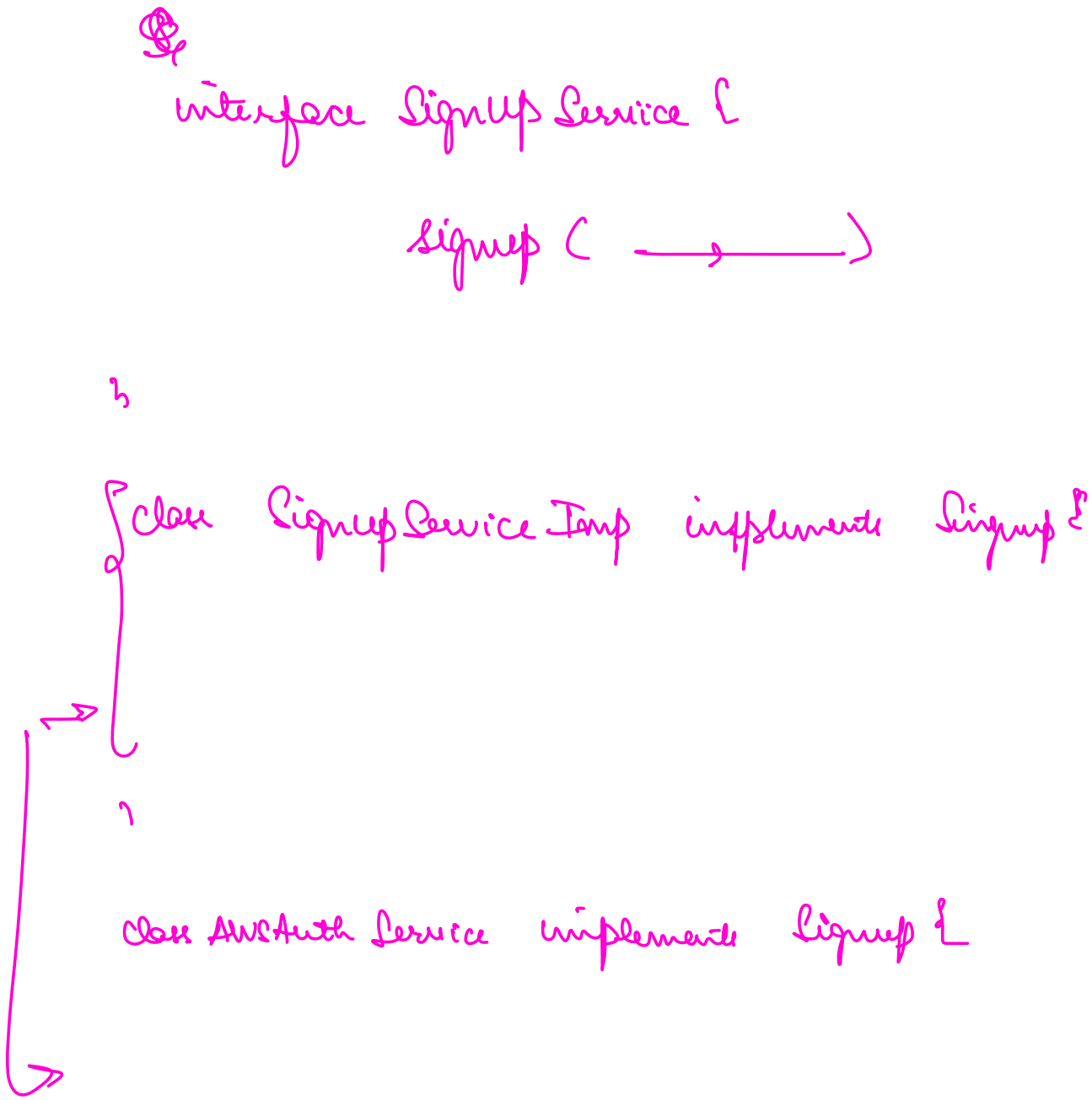
→ Shows

)

}






interface SignUpService {
signup (——→ ———)
}
class SignupServiceImp implements Signup {
}
class AWSAuthService implements Signup {

interface Something {

do (Object)

}

class A implements Something {

do (Object o) {

}

}

An Image of 1 color
Service = 10 - 20
lines