

Object Oriented



[modularity]



use

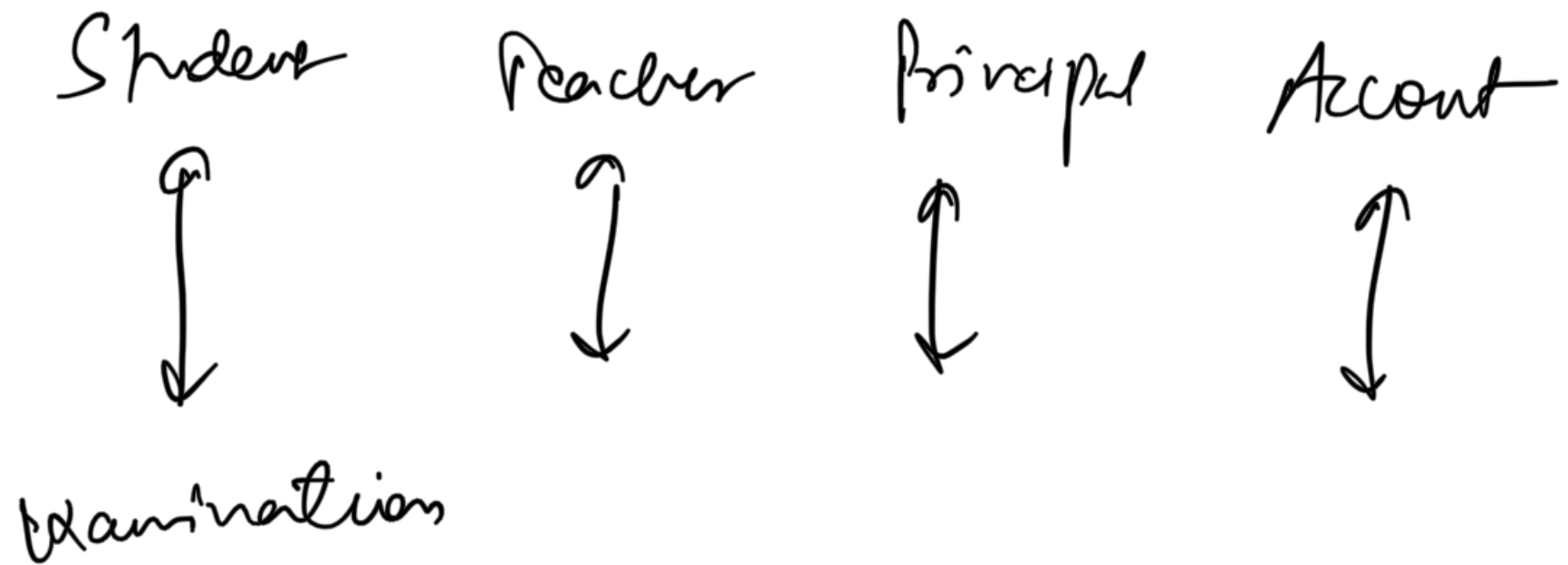
easy to debug

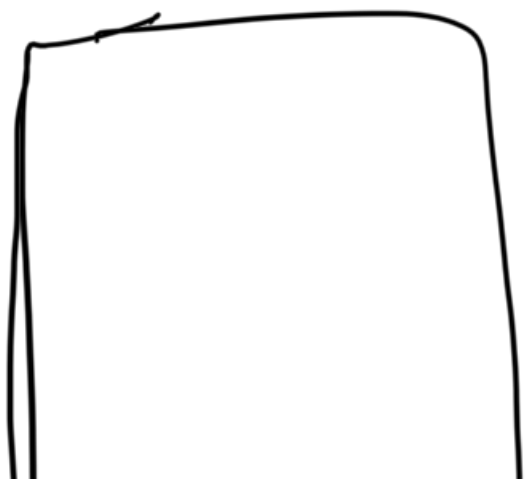
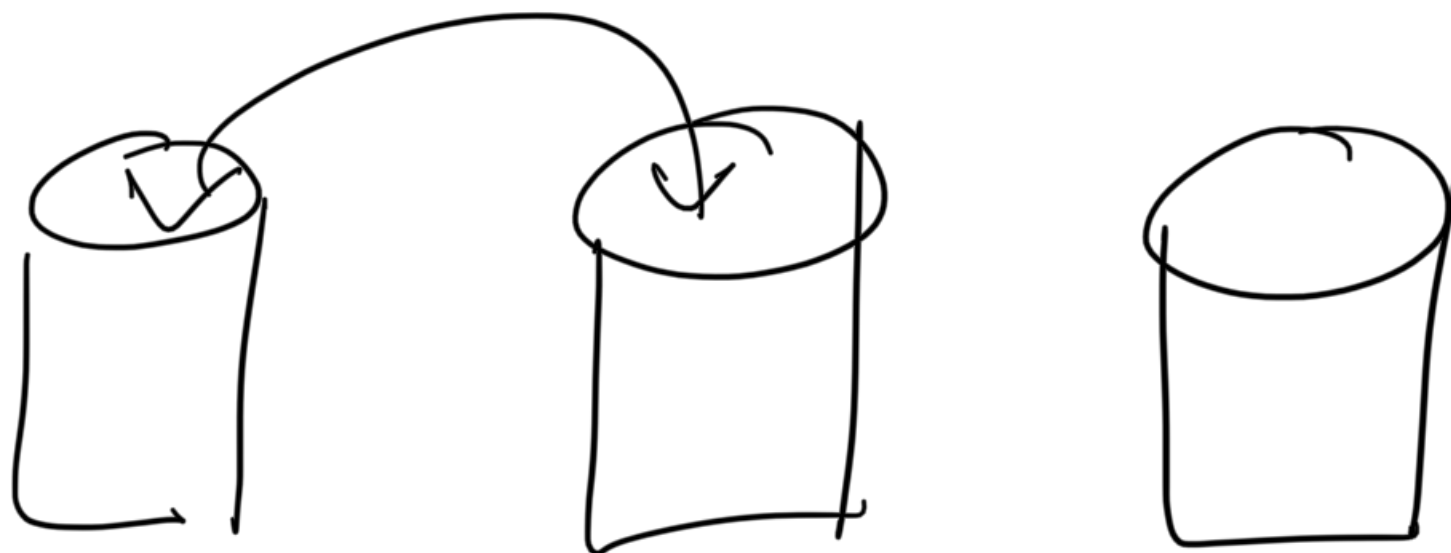
easy to break & use again

easy to enhance

4 Classes → each responsible task

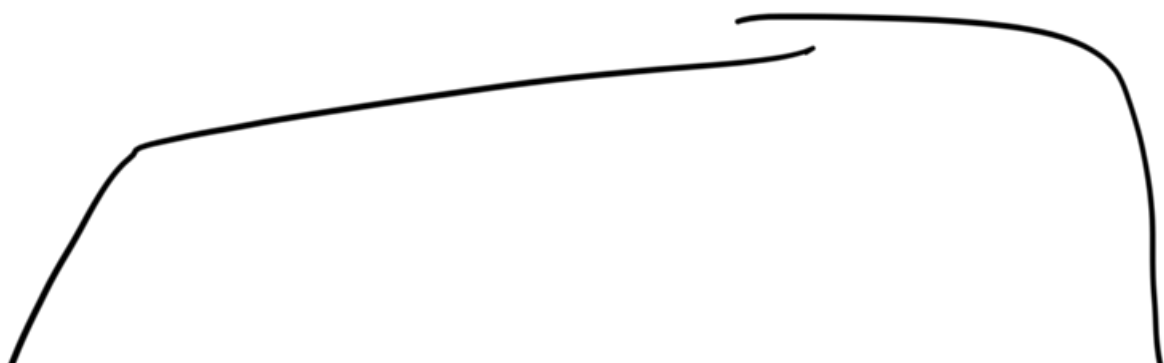
SMS → school







Swift ; Swift Drive

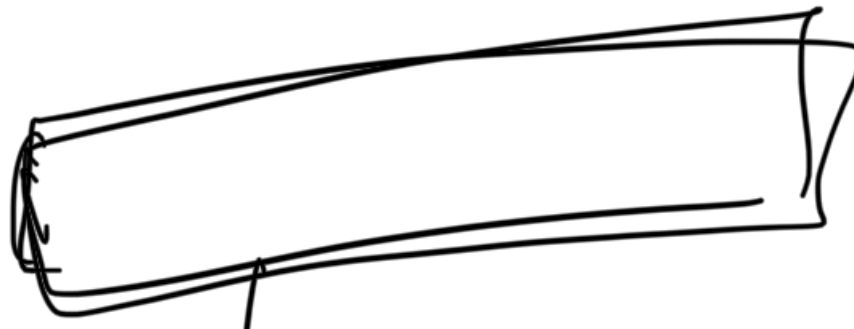
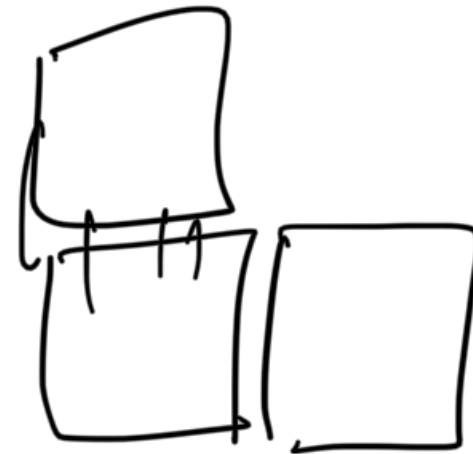




Punk

Lego

[ Surfer  
Surfer Drive





bricks  
cement  
pop

PRD → Design → Development



on other board  
camera

GPU  
CPU

NF-E  
WiFi

display

battery

→ s/w

RAM

touch panel

→ mobile engineer

Design ↑

sensors

I/O - sim & card slots

buttons

mic

memory  
IP block

flash light

Blue tooth

speakers

Design



What  
is the  
problem  
statement?

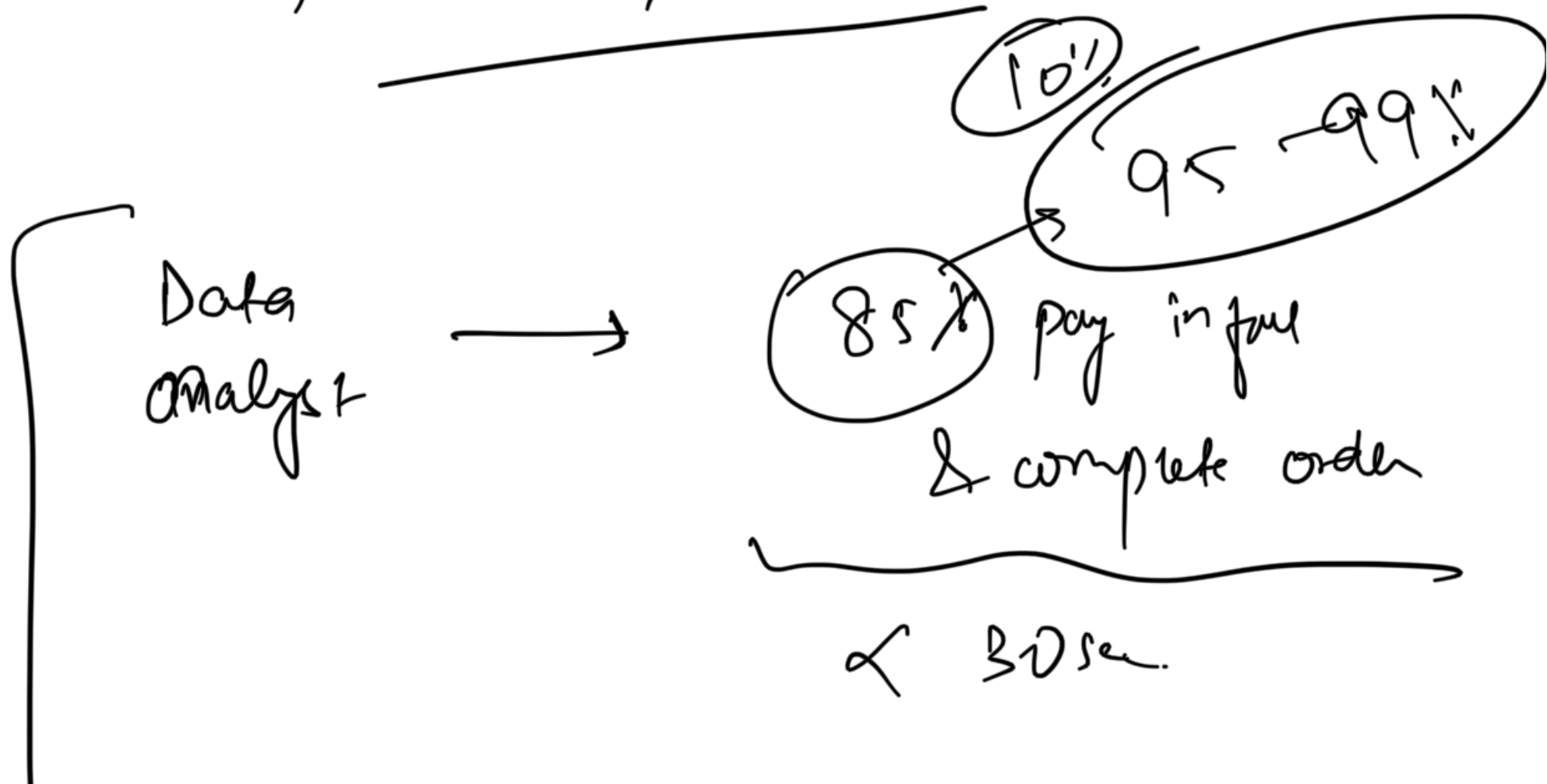
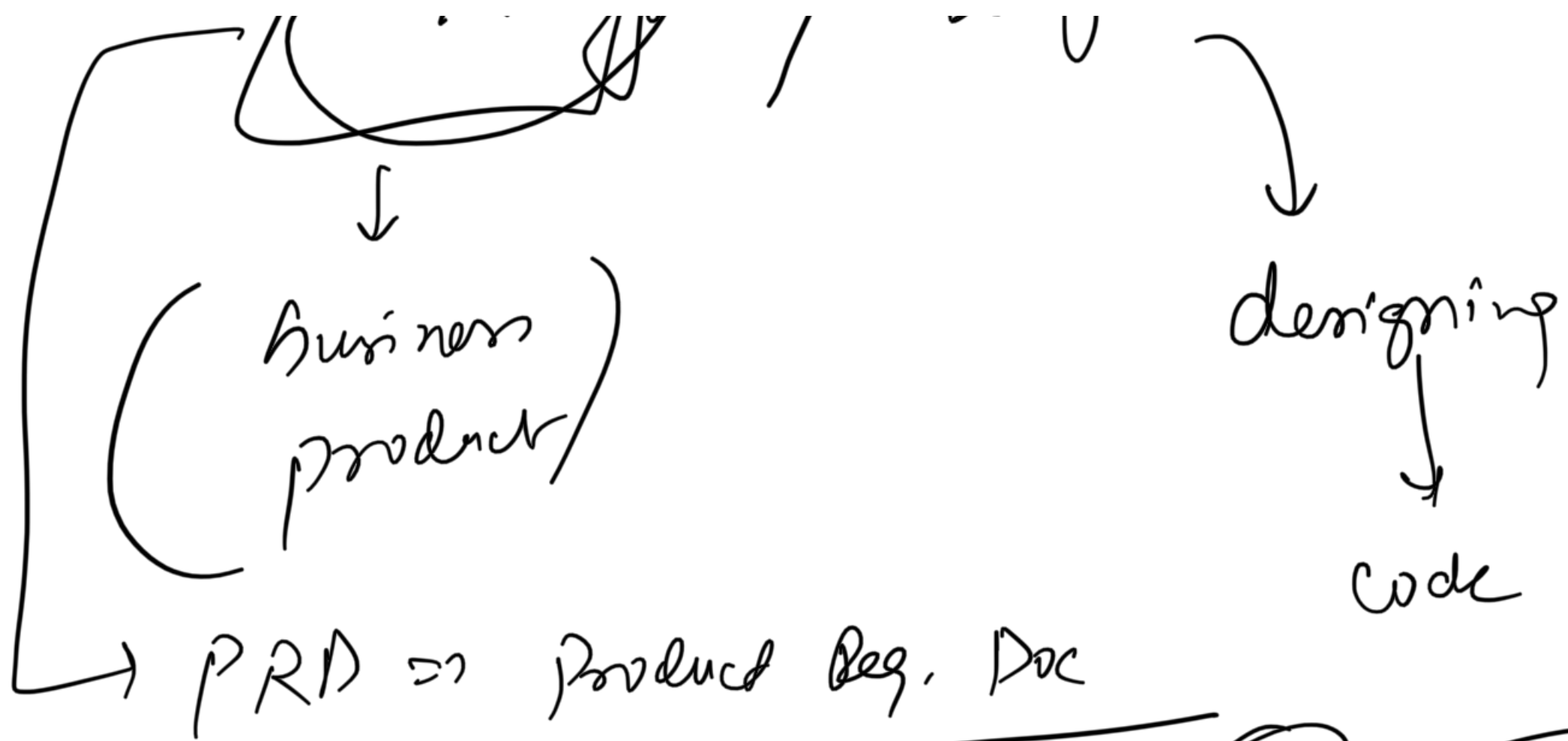
Design  
a solution

→ What to build (Analysis)

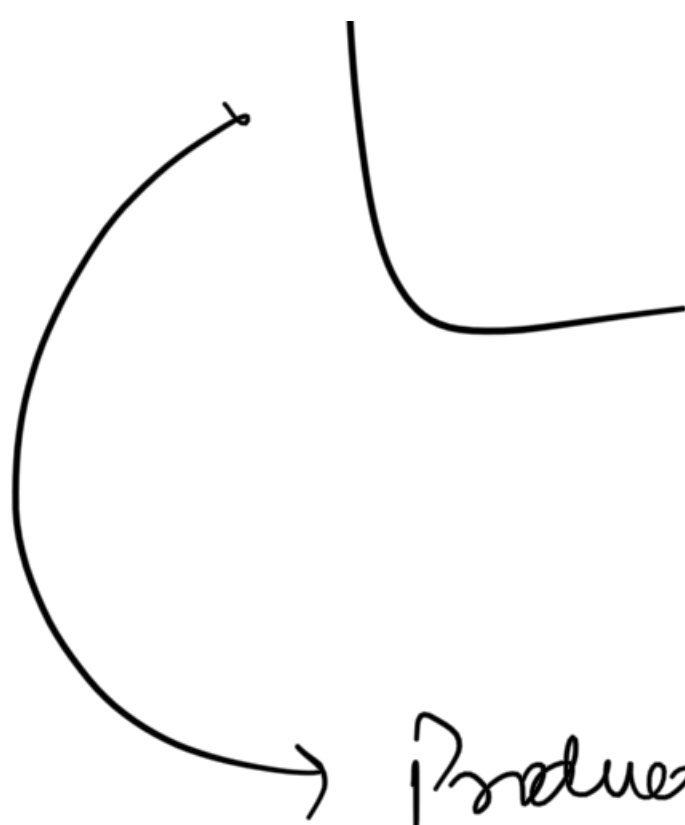
→ How to build it (Design)

Analysis / Design

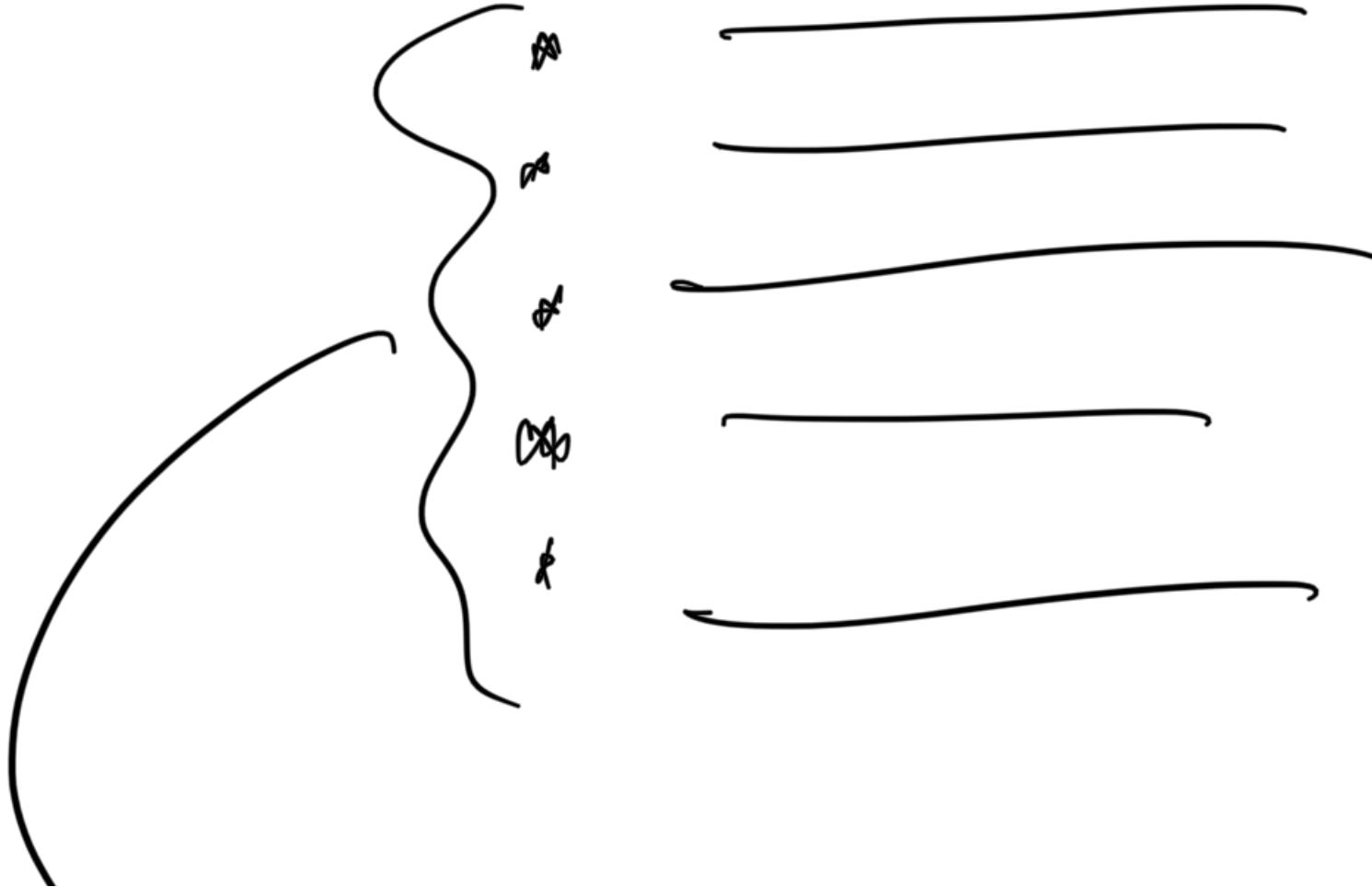




15% > 30



Product/Business → ??? What



Developer

Req. Doc

⇒

Design  
Code

Procedural

OOPs

not modular

modular

top-down

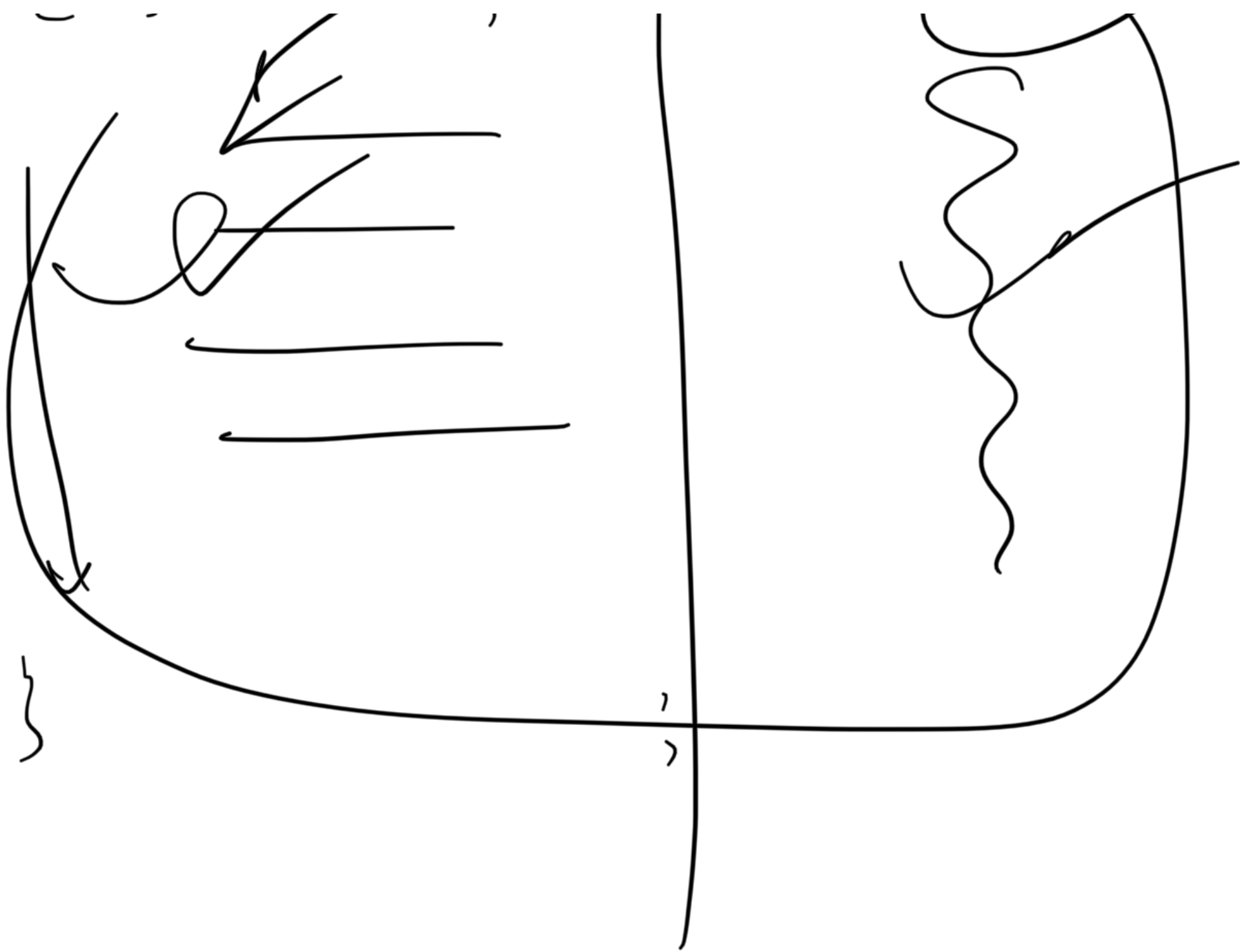
modular approach

C

Java C++

~~class Calculator {~~

~~class Addition {~~



OOA  $\rightarrow$  OO Analysis

OOD  $\rightarrow$  OO Design

[blueprints]



Knowledge  
transfer

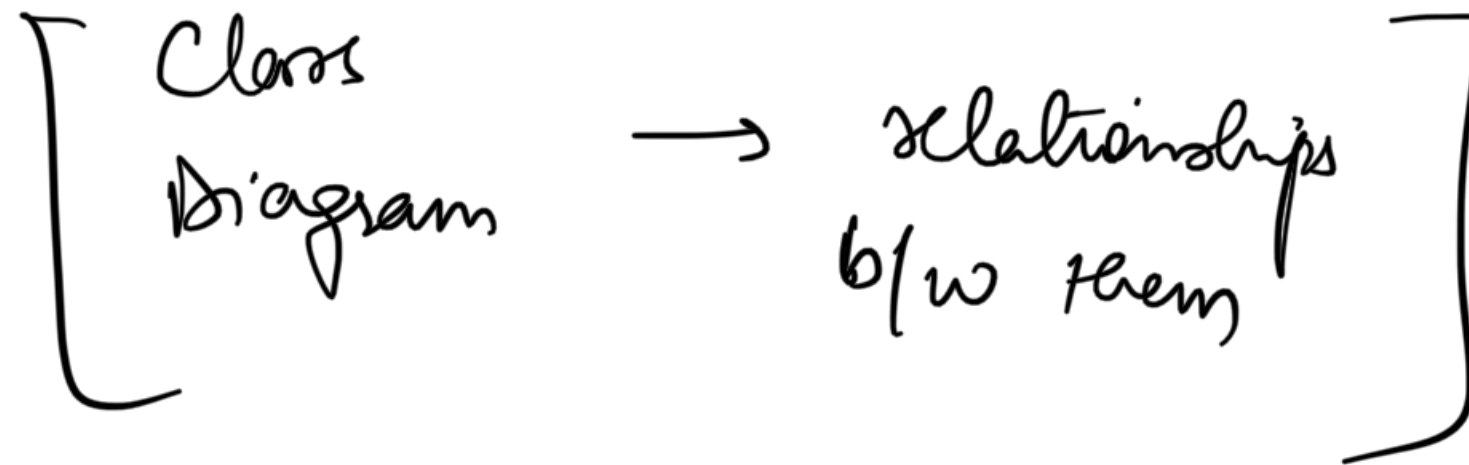
- \* easy to develop
- \* keep everyone at same level
- \* easy to give KT



UML

app. diagram. not

# Modelling Language



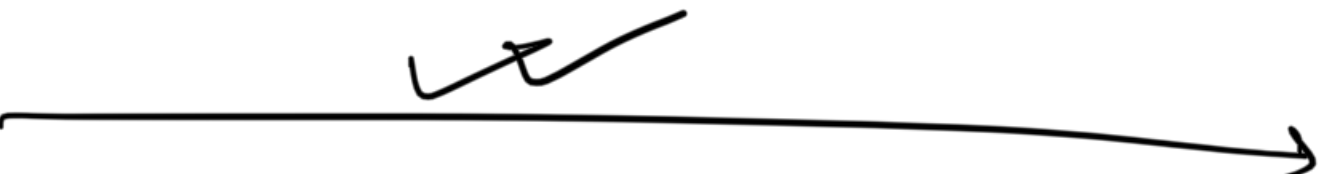
Employee

int id  
string name  
string designation

boolean applyleaves (int days)  
void approveleaves ()  
void dowork ()


String department  
double salary  
int no of leaves

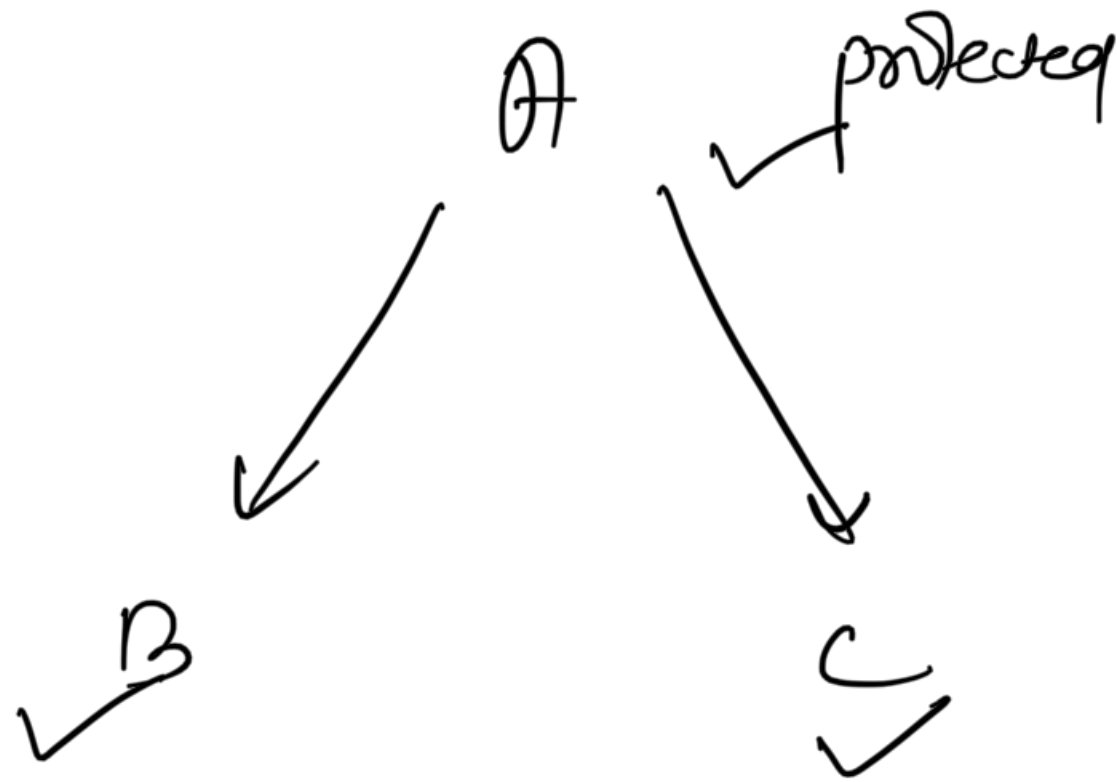
String getDetails()

(+) public  $\alpha$  

(-) private  $\alpha$  

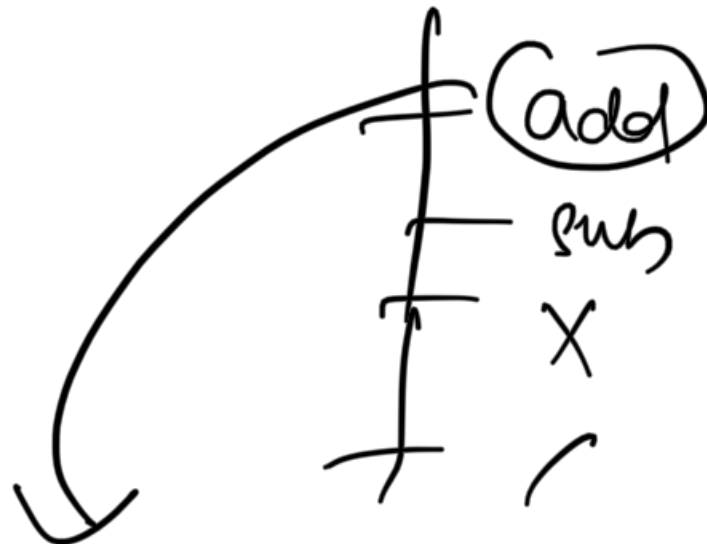
(#) protected same class  $\rightarrow$  child class  $\checkmark$  anywhere  $\times$

default / package-private  $\alpha$   inside  $\checkmark$   
outside  $\times$



1) Calculator

4 classes





method

OOP

⇒

class for respective  
task

modular

OOP

→

attributes

→

methods

analysis

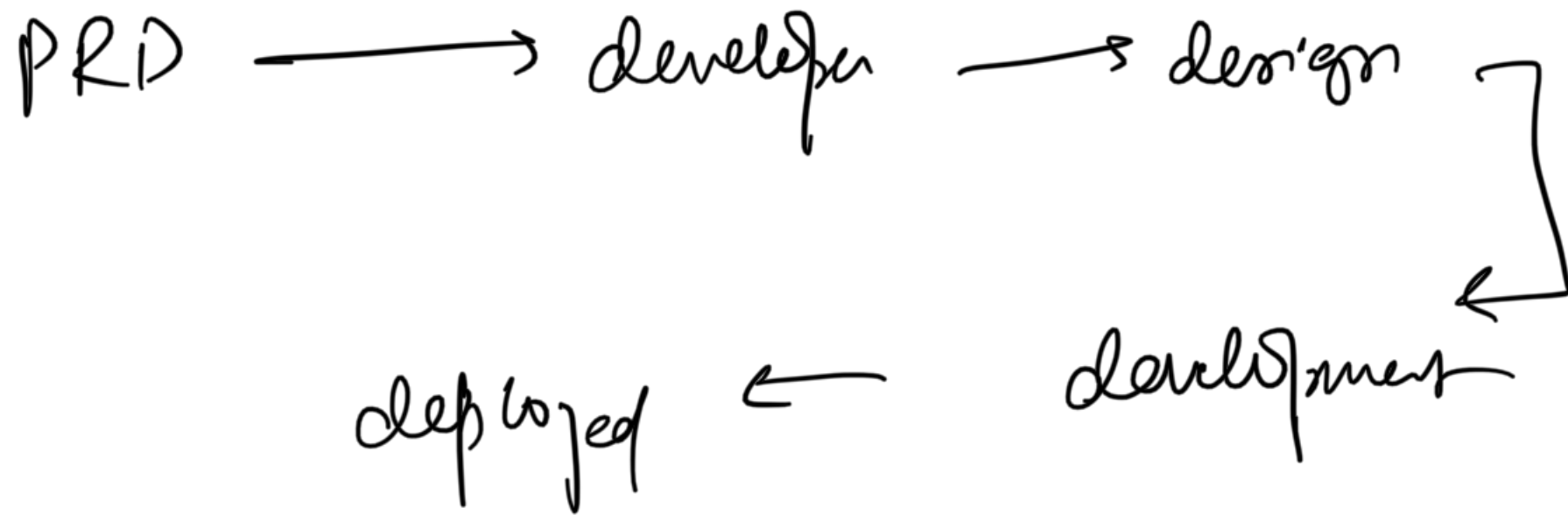
&

design

analysis

DOA

UUR

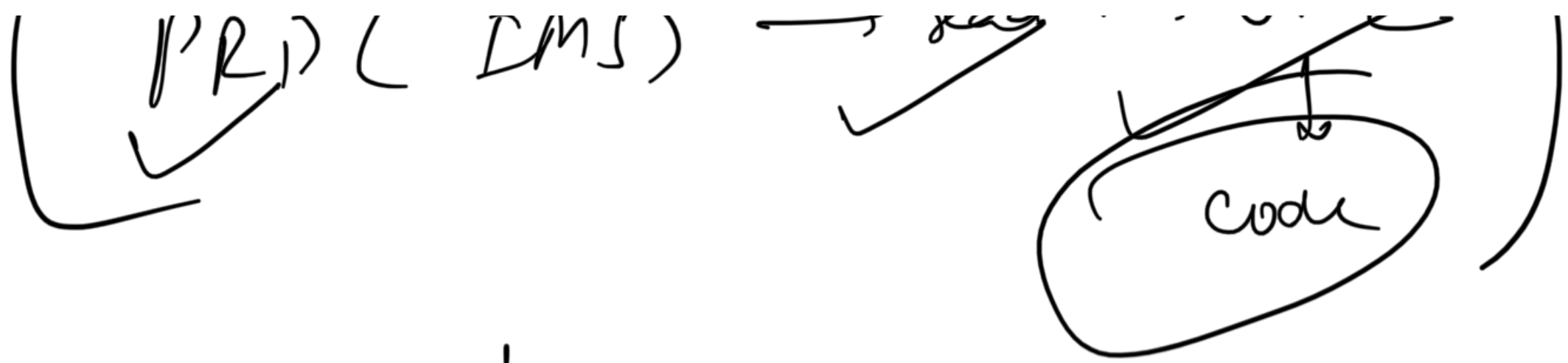


Blueprint

UML → app diagram

Employee





Backend  
frontend