Low level Design

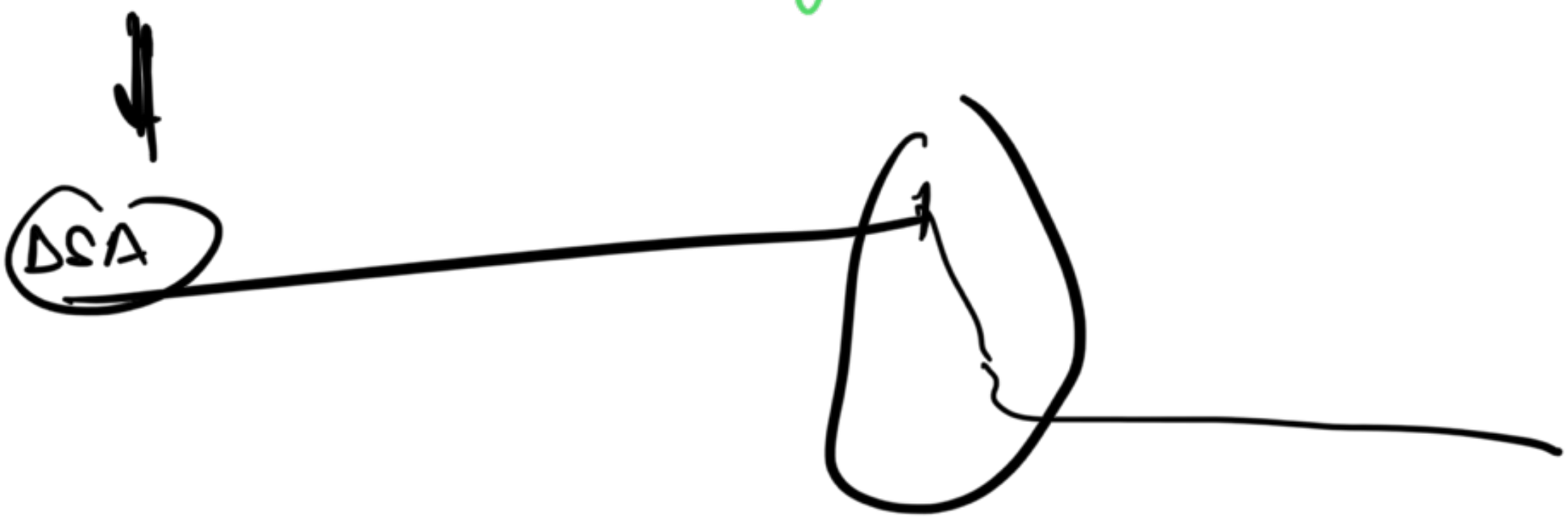
(Object Oriented Design)

Agenda

1. What is LLD
2. Why learn LLD
3. What kind of LLD interviews Happen
4. Curriculum of LLD at Scaler
5. Intro to OOP
6. Amazon Interview Question

→ Design a Bird

↳ Do an Object Oriented design of a bird.



In live class for us,

we are going to teach in Java

for other languages, we give bridge material.

→ Python

→ C++

→ JS + TS

→ Go lang

→ CPP

SWE

→ Java

CP/DSA

→ CPP

Python

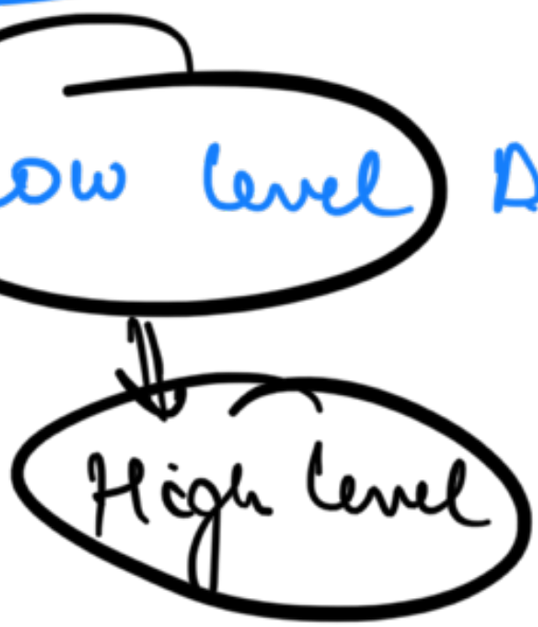
→ ~~Video~~

→ Java : The Complete Reference (Part 1)

→ Udemy (Tim Buchalka)

What is LLD

LLD: Low level Design



→ ~~D~~ ← how the whole software system

- Overview
- Not going in detail
- Abstract idea

HLD: How diff infra layers work together to serve you data

⇒ DB

⇒ Cache

⇒ App Server

LLD: Drills into the code that is running

⇒ how to write good code

→ how to structure your project

→ how to handle concurrency

→ how to do testing

→ how to design good APIs

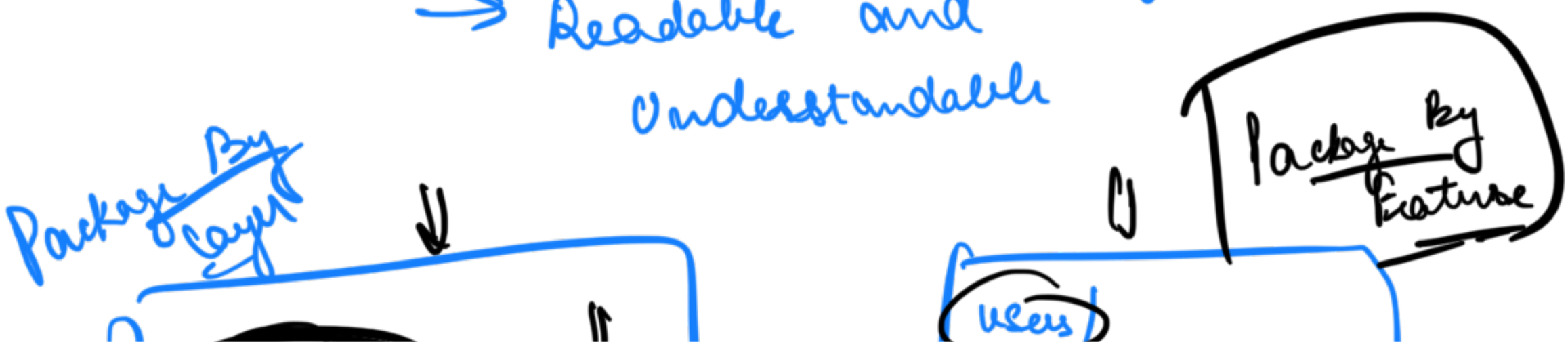
Good Code

→ Reusable

→ Maintainable ⇒ (Easy to fix issues. Easy to debug. Easy to upgrade)

→ Extensible ⇒ (Easy to add new features)

→ Readable and Understandable



Controller

→ User Contro

→ Image Cont

Service

→ User Servic

→ Image Serv

Repository

→ User Repo

→ Image Repo

Spring Boot

Django

RoR

User Controll

User Servic

User Repo

Image

Image Contro

Image Servi

Image Repo

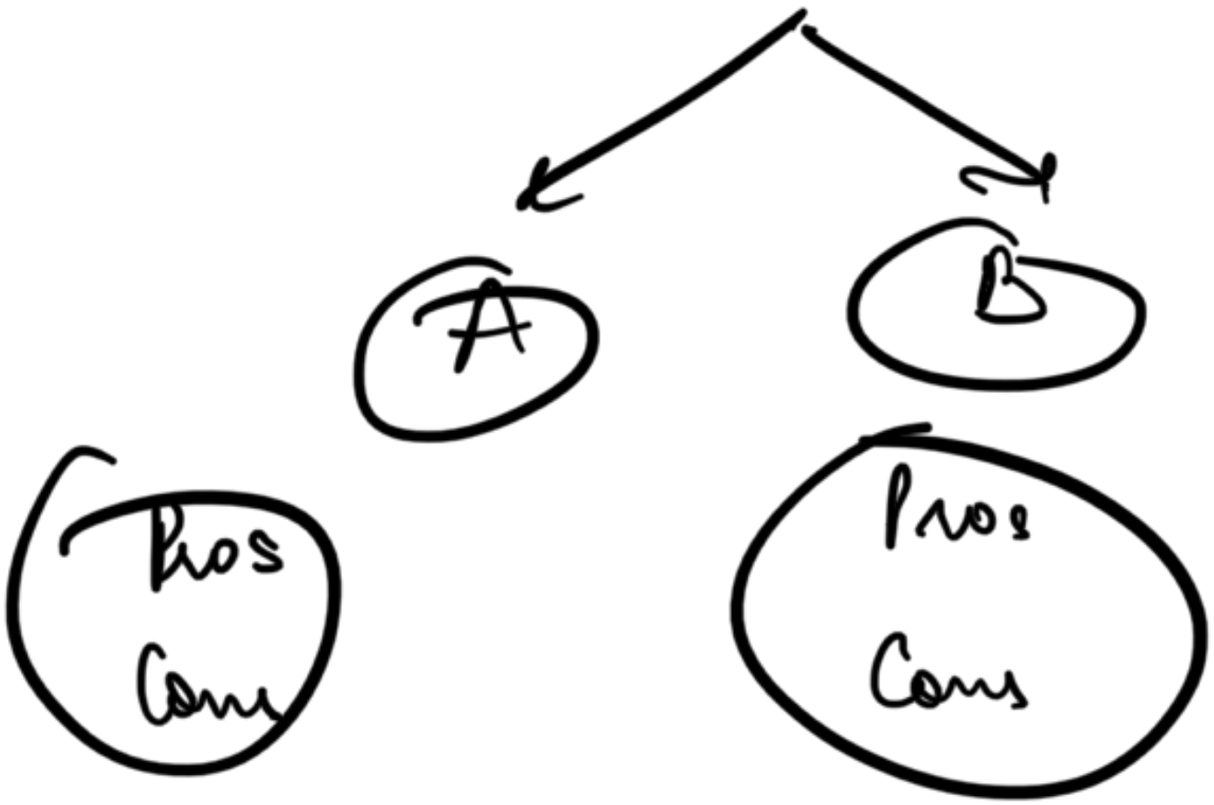
Angular

Flutter

Google

ibrary

LLD: Good implementation of systems

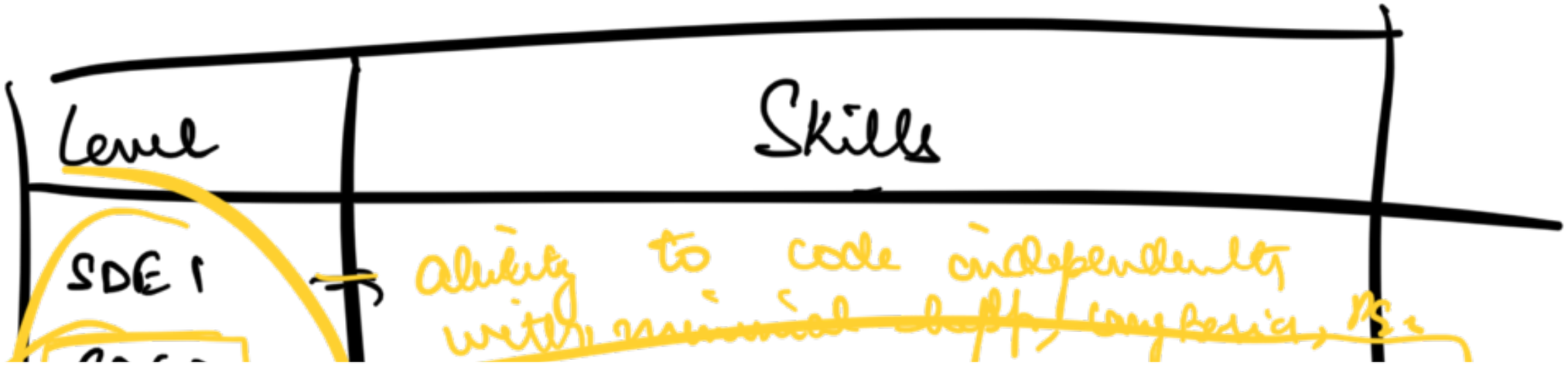


Why learn DS

① Career

② Interview —

Engineering Ladder



SDE 2 →

SDE 3 → Ability to take a fairly complex project and do the design for it with minimal help

Architect →

IA → Ability to lead multiple projects, handle cross team comm, com

VP Eng →

demonstrate LLD skills consistently for atleast 6 months

MNCs → X

Startups → ✓

Interviews

	YOE	DSA	LLD	HLD
SDE1	0 – 2	4 ✓	✓	X
SDE2	3 – 5	3 ✓	✓	X Startup

SDE 3	6 – 10	2			

Types of LLD Interviews

Theoretical Rounds

- What is Singleton DP
- OOP
- [illegible]

Design Round / LLD Round / OOP Round

- Involve a case study for which [illegible]

Machine Coding Round

1. Detailed PS
2. Already given req. in PS

Design ...

→ Design Patte

* Rare in startups

* Common in Mic/

Amazon

→ 30 – 45 min

you have to design

→ 45 min

→ Abstract PS

→ Design a class

Design Splitwise

→ ① Gather Req

② Clarify Req

③ Design

No CODE

→ Use Case Diag

Class Diag

WC: Code a single class

2 – 2.5 hrs

→ ✗

→ ✓

→ ✓

30 min

→ Expected to write an E2E working

code

2 hrs

Handle Connection

Write test cases

Tech Rounds

→ DSA

→ LLD

→ HLD

Managerial / Leadership Rou

→ 1 Part

→ Use Design Pattern

→ Good design

→ Good Project Structure

↓

100% allowed to use google

10:30 PM

Scaler Course in LLD

- ⇒ OOPC (2 classes)
- ⇒ SOLID Design Principle
- ⇒ Design Patterns (10 most frequent ones)
  - → 3 classes
- ⇒ UML Diagrams & Schema Design
- ⇒ Concurrency

Code

- All Design and Testing
- How to approach LLD interview Problem ⟸
- Case Studies / Interview Question

① Entities (1 of them)
- Pen
- N/B
- ATM

② Games (2 of them)
- Chess
- TicTac Toe
- Snake & Ladders

→ Card Game

(3) Real Systems

→ BMS

→ PayTM (3 of them)

→ Parking lot

→ Splitwise

→ MailChimp

(4) Engineering Problems

→ Distributed Cache

→ Distributed Queue (1)

→ Mirage Quiz

Assignments

1. Objective Question (MCQ / Single word Answer)
2. Design Reviews (TA + 3 Peers)
3. Code Review (TA + 3 Peers)
4. Code Submission

JS + Python + Java

Intro to OOP

Subject

Action

Active Voice

Noman

is teaching us

I am learning us

Object oriented programming

Procedure Oriented

Program is nothing but a series of procedures executed in a particular order.

Each procedure may call other procedures.

Execution of program starts from a special procedure main()

Series of instructions to perform a particular action by taking some input data and might return some output. ← function

Procedural Prog language : C

⇒ Student in C

```
struct Student {
    int age;
    String name;
    String batch;
};
```

=

int Student_age

String Student_name

String Student_batch;

Student.age = 10

Student.Name = "Name"

Student.batch = "

⇒ changeStudentBatch(Student, newBatch)

Student.batch = newBatch;

}

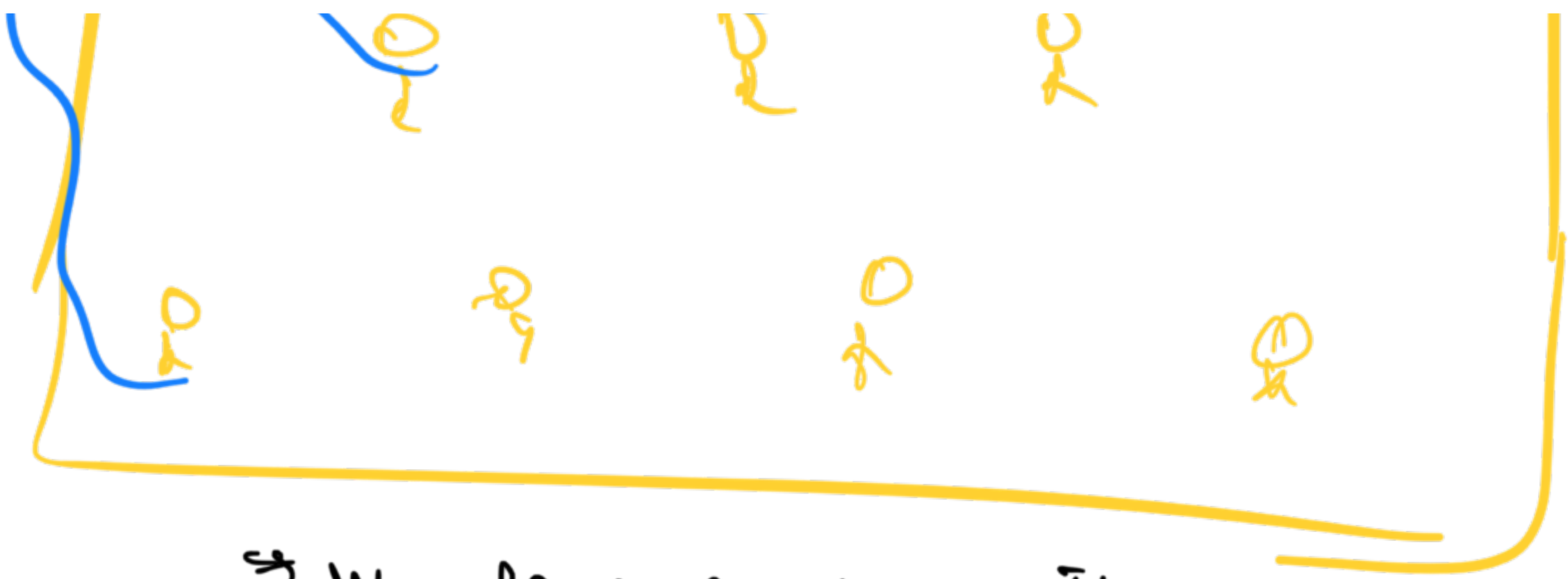
In Procedural Prog languages ⇒ Data is just

a second world entity

Data has no control over itself.

Anyone (any procedure) can change whatever attributes of data they want.

Anyone can write a procedure to do anything with the data that even the data might not want to.



→ We have no self with

→ Robots being controlled by Someon

→ We are puppets

→ with control of a puppet holder.

OOP ⇒ Humans also by default don't think of

the world of someone making a entity take an action

Humans by default think of the world as different Dec entities performing Actions on themselves.

Procedures shouldn't be the central part:

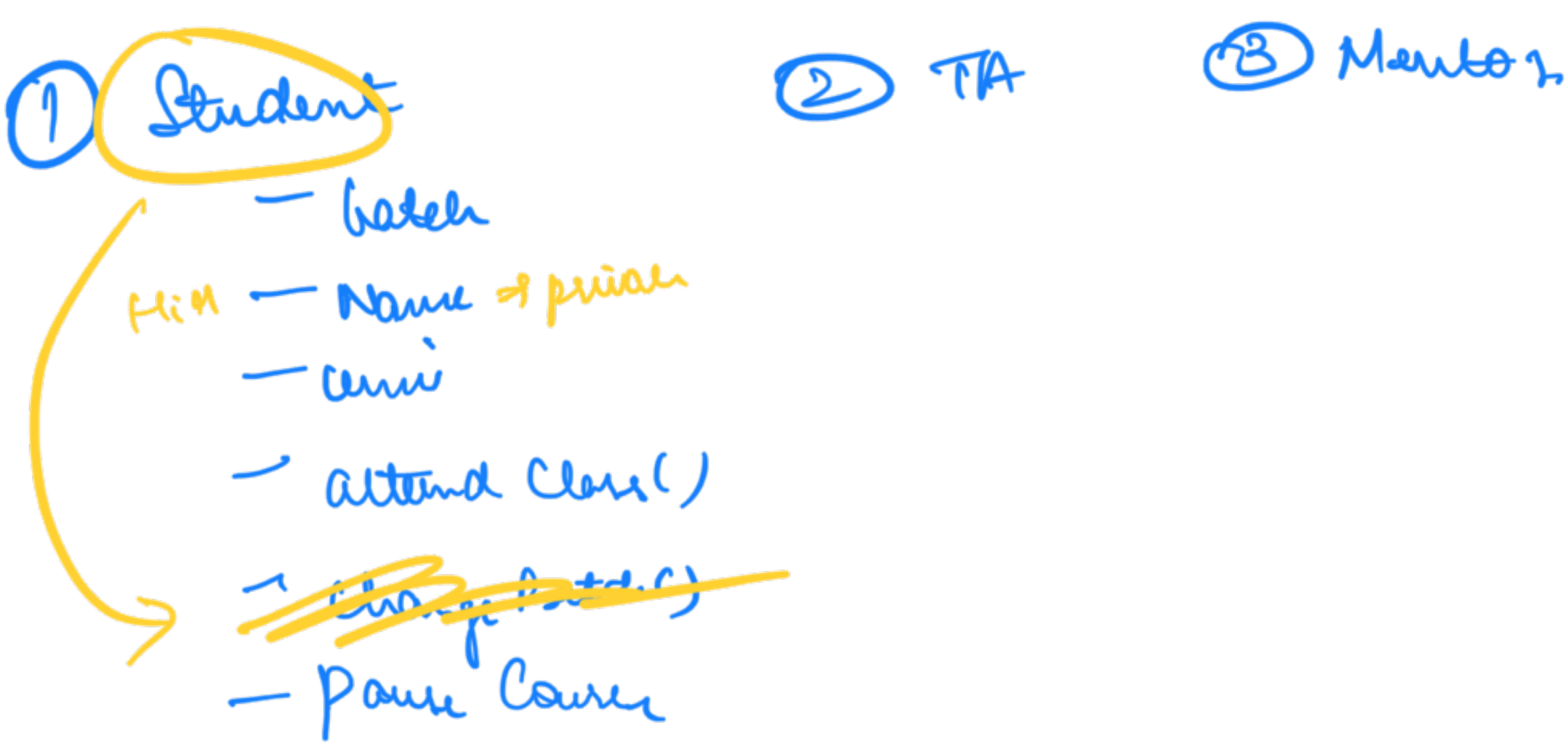
are the central part.

that performs an action

Anyone (Any thing) / I

And has some attributes.

→ Anything involved in working of the system -



① Entities can decide what do they want or not want to expose

In OOP

Entities are in complete control

→ Decide what actions can be taken on it

→ Decide what are actions allowed to see

Program is nothing but diff methods being called from diff entities in an order

Student · make up()

. brush()

. draw()

. dress

Case Study

Interview Problem at Amazon

PS : Do an OOP design of a Build.

↳ Design a software system that manage

birds.

Bird ⇐ Way to represent an entity

Colour

Name

no of Eyes

breed ⇐

fly()

walk()

eat()

⇐ attributes

⇐ behaviours

fly() {

if breed == pigeon

elif breed == 'parrot':

elif breed == 'hen':

else:

)

No

Test Cases

Violation of 'S' of SOLID

Single Responsibility Principle

Every code unit should be responsible

for only one thing

Violation of 'O' principle of SOLID

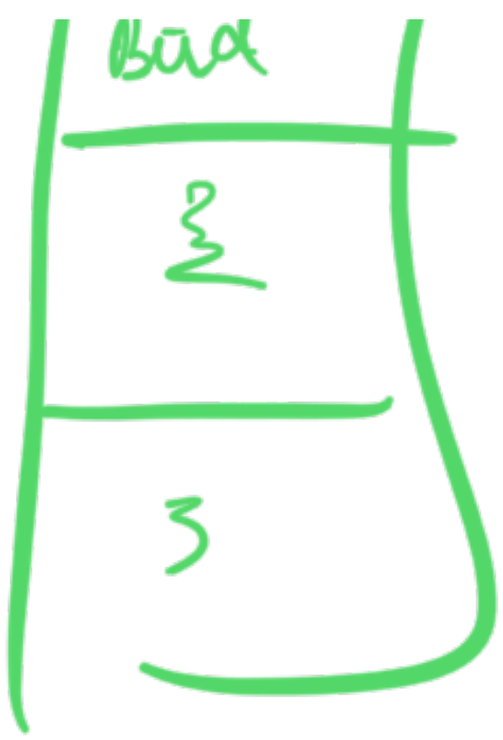
⇒ Open/Close Principle

⇒ Whenever I add a new feature, I "Extensibility"

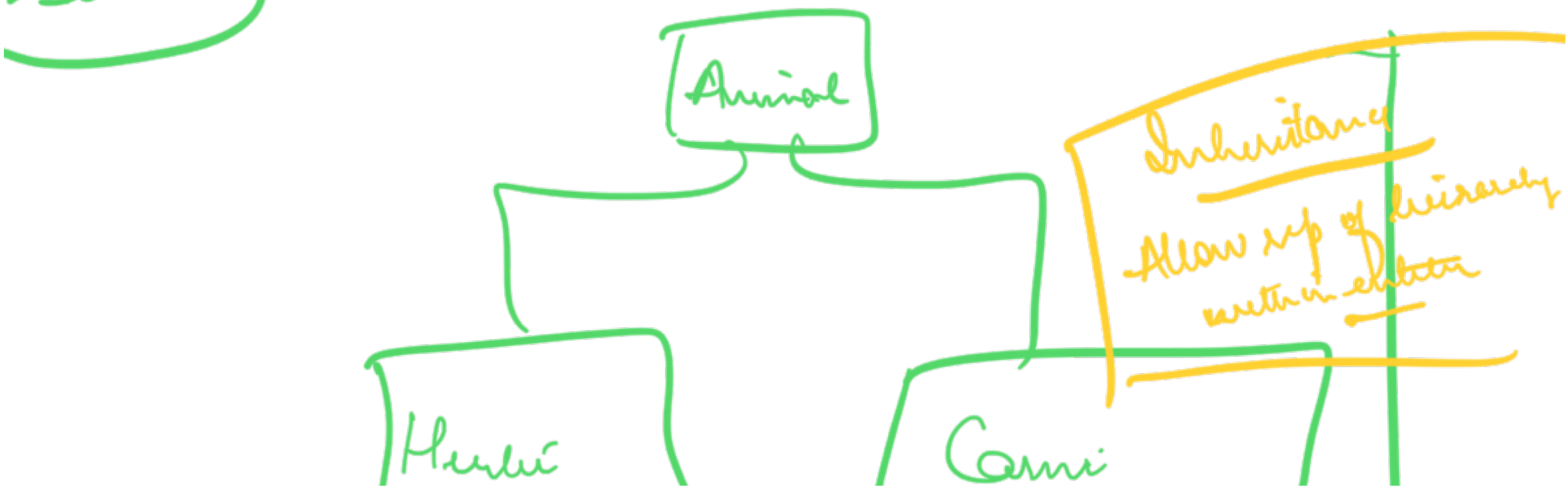
① It should be easy to add new features

② I shouldn't be required to modify already existing code.

Sol 1


Bird
2
3

Sol 2

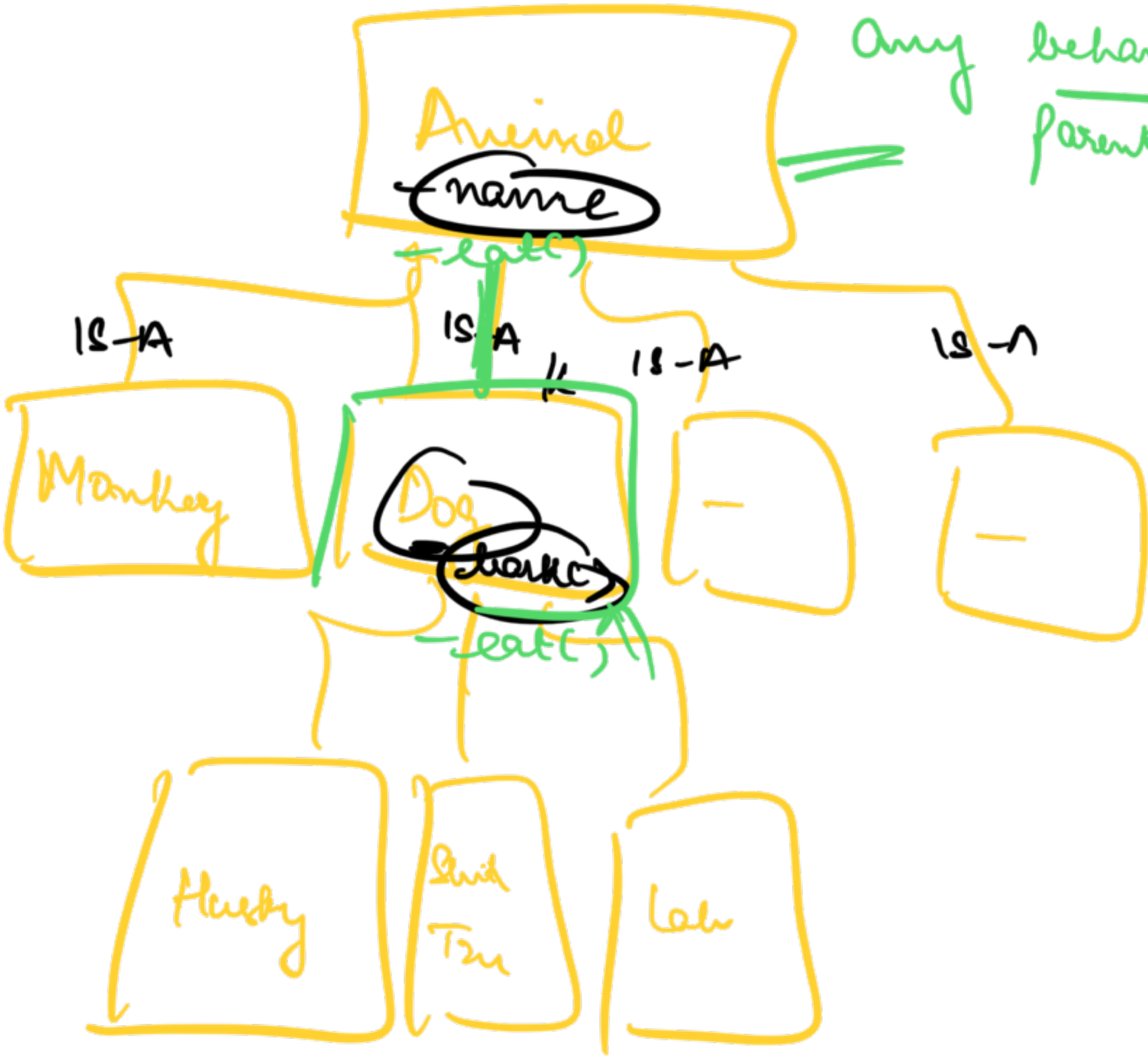

Animal
Inheritance
Allow use of hierarchy
within class
Herbi
Carni

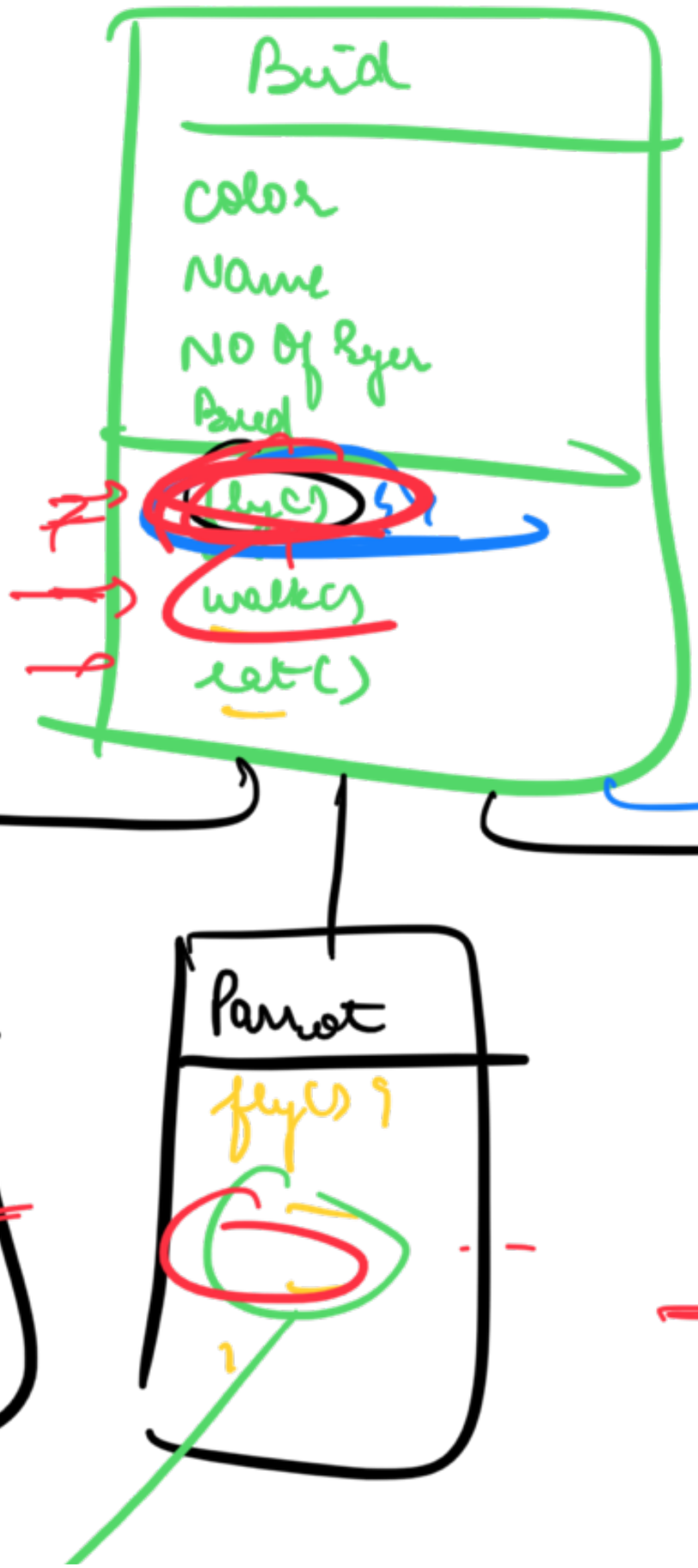
→ Allows child classes to inherit all the attributes of parent class

→ Allows child classes to add their Additional features.

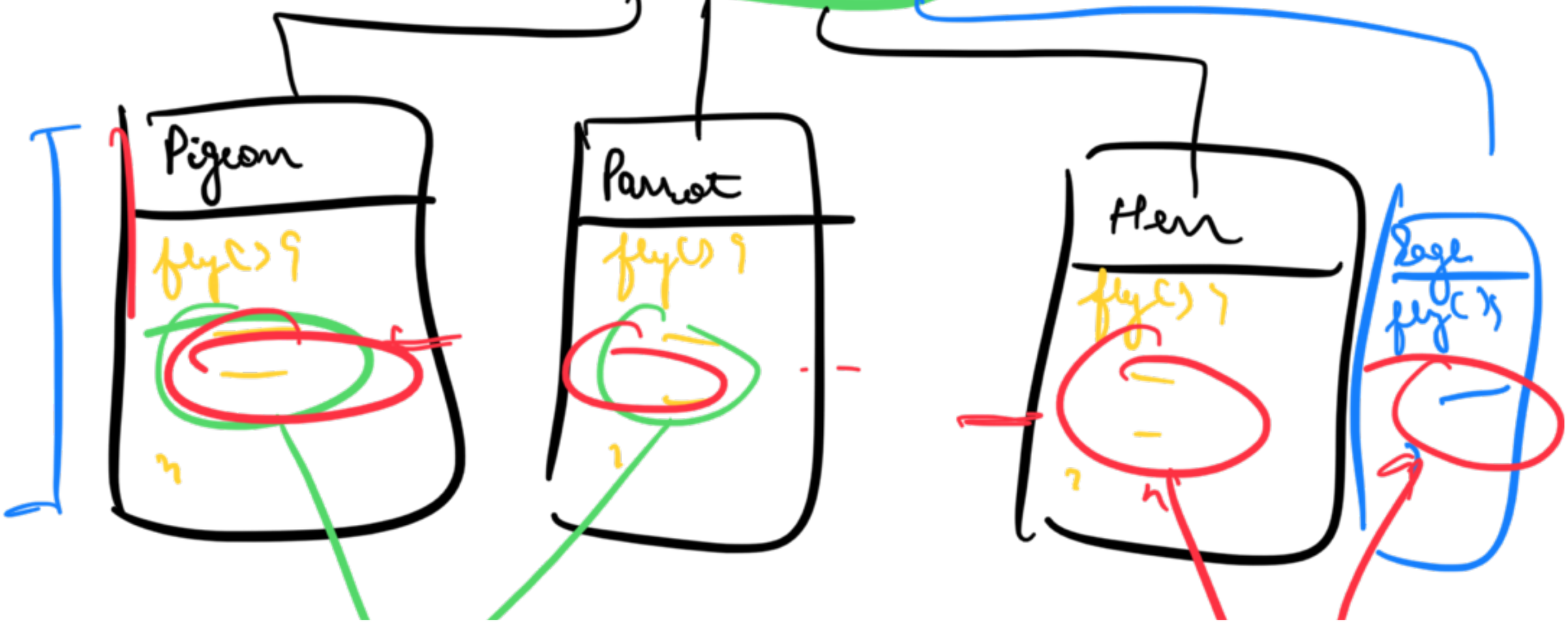
→ Child can override (change)

Any behaviour of the parent




Bird
color
Name
NO Of Eyes
Bird()
fly()
walk()
eat()


SRP
O/C


Pigeon
fly()
Parrot
fly()
Hen
fly()
Eage
fly()

Code Duplication

Assignment

How can we solve this code duplicate issue.

→ 22nd April ⇒ Q Doc

SRP

100r. SRP is

Design is subjective / compromise /

→ What might be enough for you might not be for other

→ Dont over engineer

→ Head first DP

→ Effective Java