LLD 1

Intro to OOP

and

Why LLD

1 day

DSA ⇐ inf. of n^n ≈ 10y

CS Fundamentals

LLD – 1.5 Months ≈ 16

HLD – 16 Classes (1.5 Months)

Project Building ≈ 1.5 Month ←

Naman Bhalla

Naman@Scaler.com

+91-9996203771

What is LLD
+
Why LLD

+

Introduction
to

Why LLD

OOP

→ Structural, Procedural, P-

→ What is OOP Pro

→ Key Terms related to OOP

→ Case Study

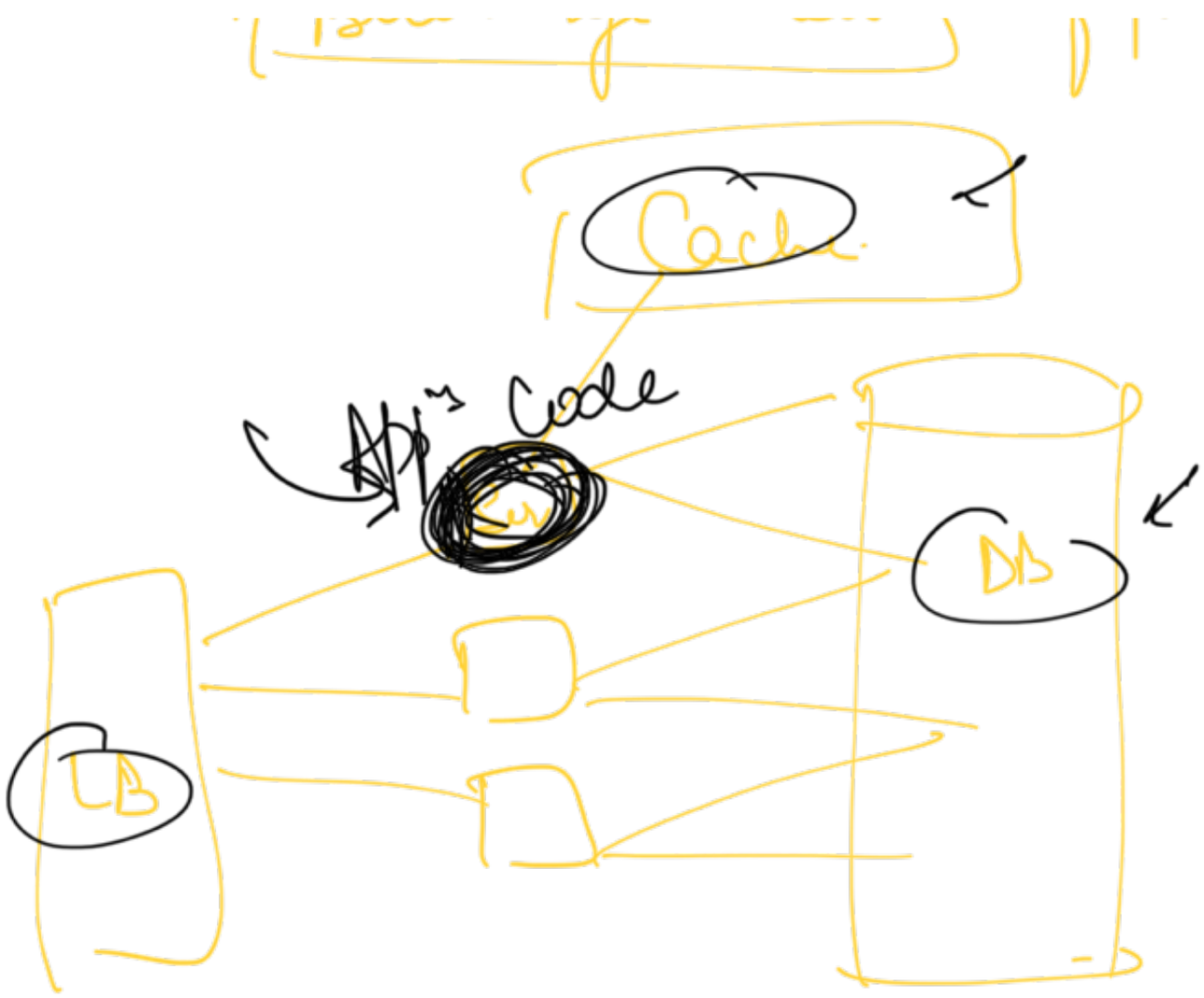
What is LLD and why learn LLD

LLD: low level Design

HLD: High level Design

↳ Big Picture

↳ Bird's Eye View of your software project



Game

Book My Show

Ludo

Chess

LLD

Low Level Design

→ goes into detail of implementation of app's code / code that implements

the software system

- Abstract HLD
- LLD
- Implementation

adapter

- Understand Requirement
- Classes
- DB Schema
- Modules

Web App's

↳ APIs

→ F ~

→ Services

→ APIs

Command line App

frontend

⇒ You write | GOOD SOFTWARE |

→ LLD 101 + Intro to OOP

⇒ Design Principles ← Write code that is loosely coup

Rules / fundamentals that are goi

⇒ 3 classes: Design Patterns ← Real implication of

→ Established SOL to common Software design

→ Schema Design and UML Diagrams

→ LLD / Machine coding Round
How to approach LLD problem

→ HLD
→ Dist Sys
→ Microservs
Monolith
→ Concurrency

→ 7 Classes on Real Systems

→ TicTacToe
→ Snake & Ladder
→ BMS

+ Code

→ Splitwise

→ Mailchimp

→ Parking lot

→ Pen

→ Payment Gpay

(Command Line 4/5 of each of

Current SDE1 : LLD & Machine Coding Interview

FAANG : SDE2+ : LLD / HLD Interview

It helps you perform better job

Interview + Career

Asg

1. Submit Design / Design Doc
2. Implement a service
3. MCQ

TAs

+

Peer review

CH

① Java

② Draw.io

③ Jetbrains | Intellij | Py Cha

Intellij. Ultimate

6 Month

PyCharm

UML Diag
→ Class Dia

NodeTs
React
Spring boot

→ Activity Di.

→ Use Case Dia

Testing
Debuggi

∴ 24 Patterns De

↳ 12 design pat

How much time of your day you spend codin

1-2 hr ≈ 25%

12% actually coding

40 LPA

4 Lak + 36 Lak

Debugging

Thinking → logic

Meeting → Planning

KTS → teach ah

Understand Requ

Reading Code

Deploy

draw.io

Notes

→ Tech Doc

Scaler Academy

Code Review

Documentation

Testing

→ Distributed Cache Design

→ PayTM / GPay Design

HLD

Reading Code v/s Writing Code

Good Software

Readable Code

Understandable Code

① Readable / Understandable

② Testable → Modular

③ Maintainable → ensuring the code keeps

④ Extensible → Cater to New requ

Scalability

3 month

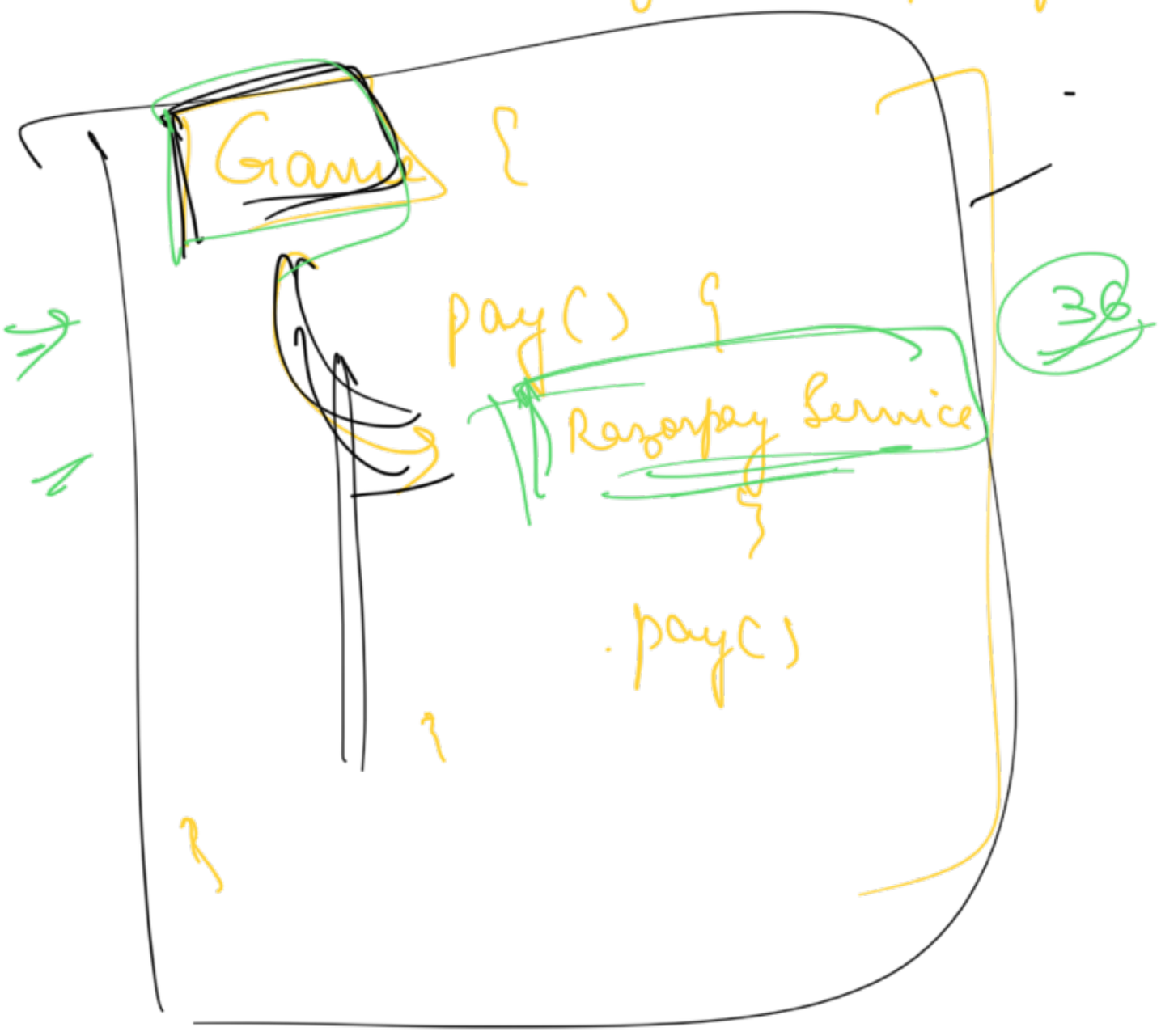
Principle of good software

① Easy to reuse codes

Should not have some logic in 2 places

Why is Code reuse diff

① Tight Coupling



↳ Pay U Money

(2) Hard coded Thing

```
areaOf Circle(Radius) {

    return 3.14 * radius * radius

}
```

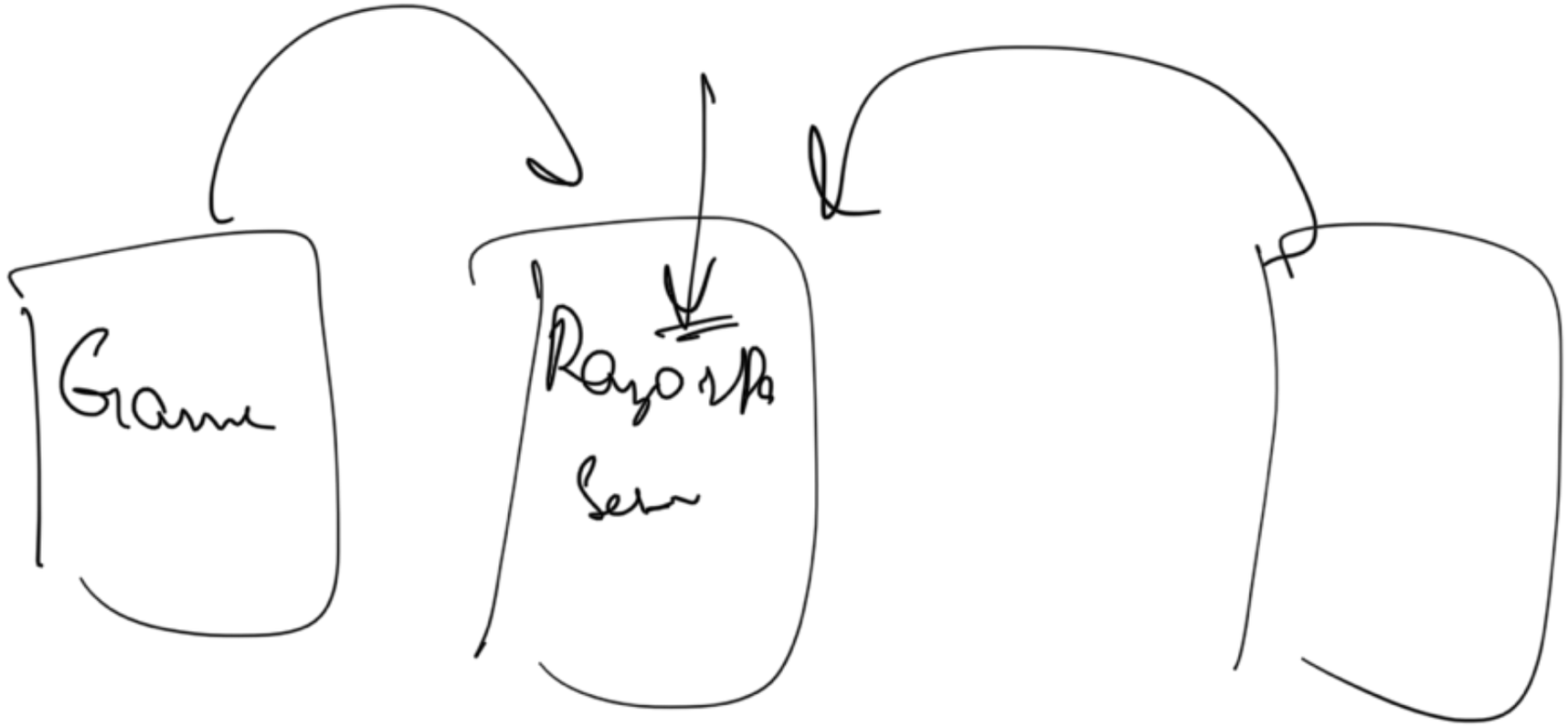
② Extensible

Requirements will keep on chan
→ iOS

③ Maintainable

~~Tight~~ Coupling = Flagging

SOLID




Game
Another


Sound
Dependency
Inver

already using design pattern / principles

to communicate with fellow progr

→ facto

→ Sw

→ Identify problems befor they occ

LLD in Swiggy

Paytm → DP

↳ NO Best Desig

↳ Bad des

↳ take decisions and
justify them

Bread for SMum 10:25 PM

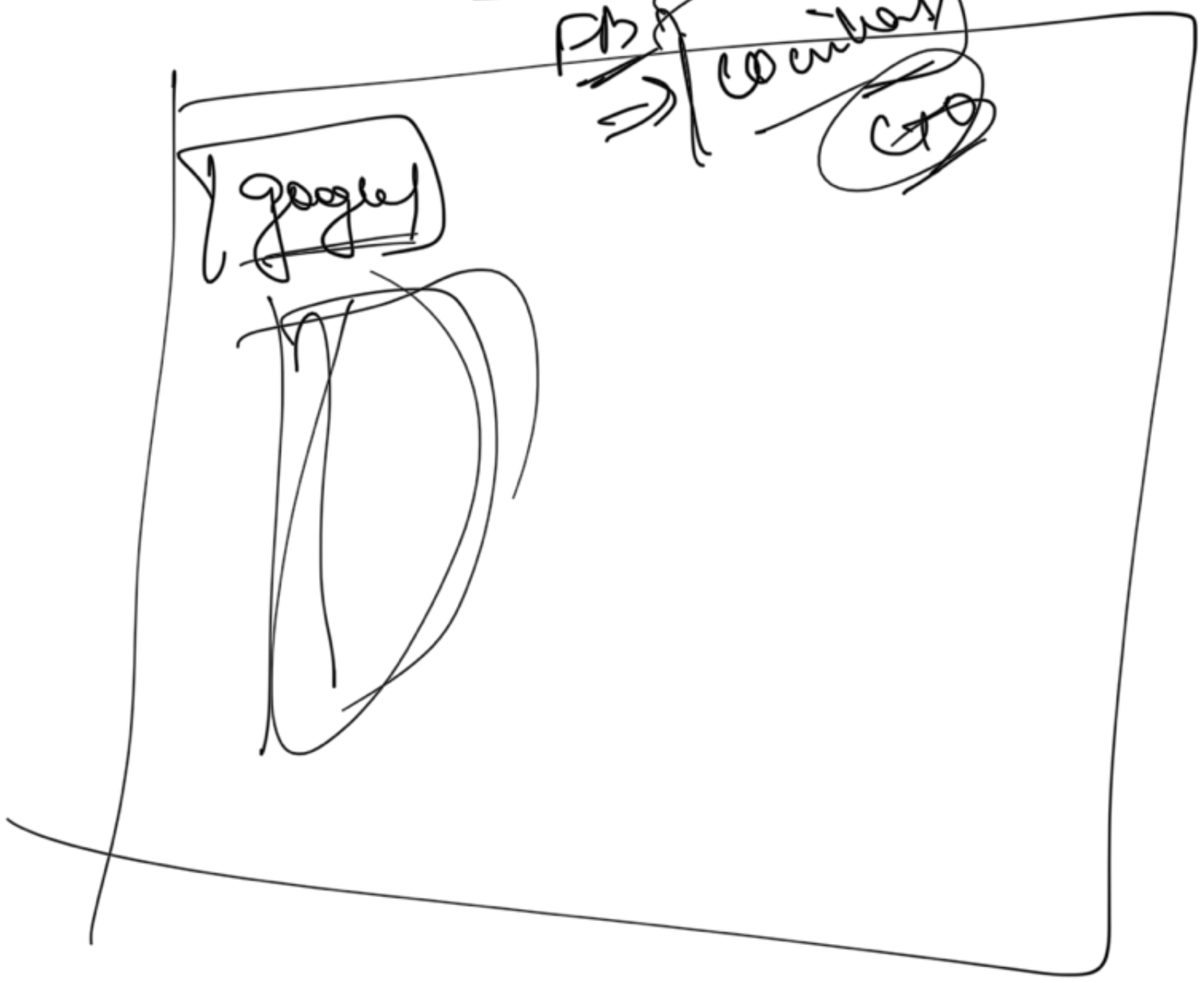
OOP

≈ 11:15PM → Next Class

Schema Des.
VM

@ Naman

Monolothic Codula

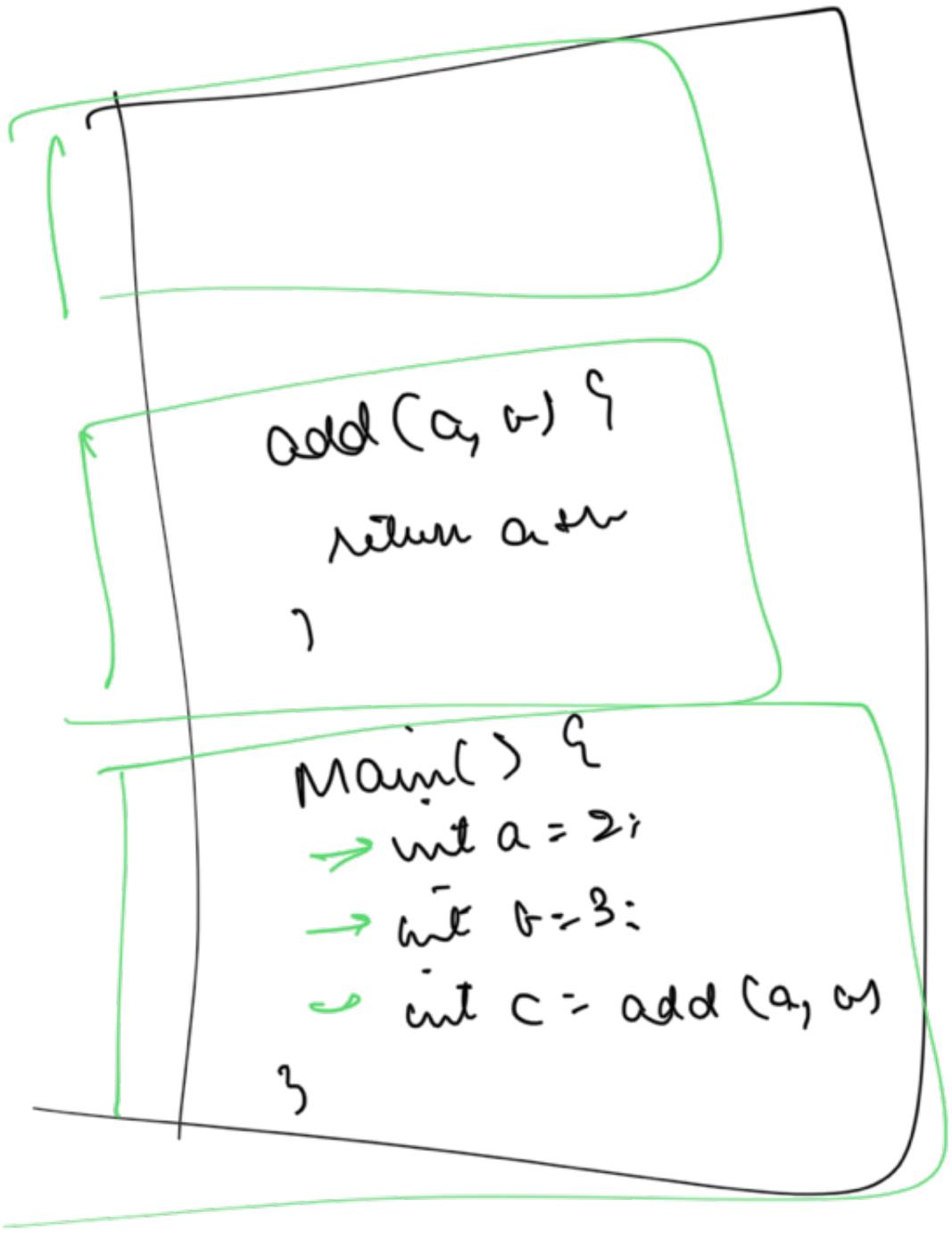


=16 bit Cpu

HPT Multimedia

Intro to OOP

function / Method



Program is running by procedure.

Procedural Programming

→ Multiple Procedures

→ Execution starts from 1 procedure

Sequential

Sequence

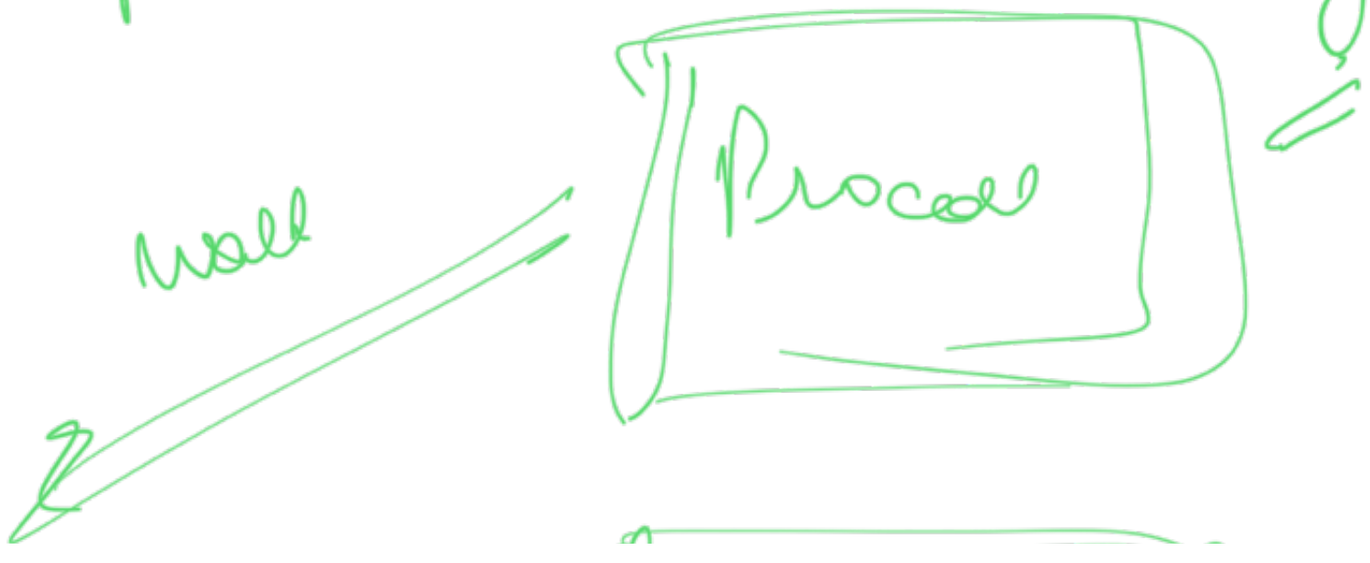
Code execution happens

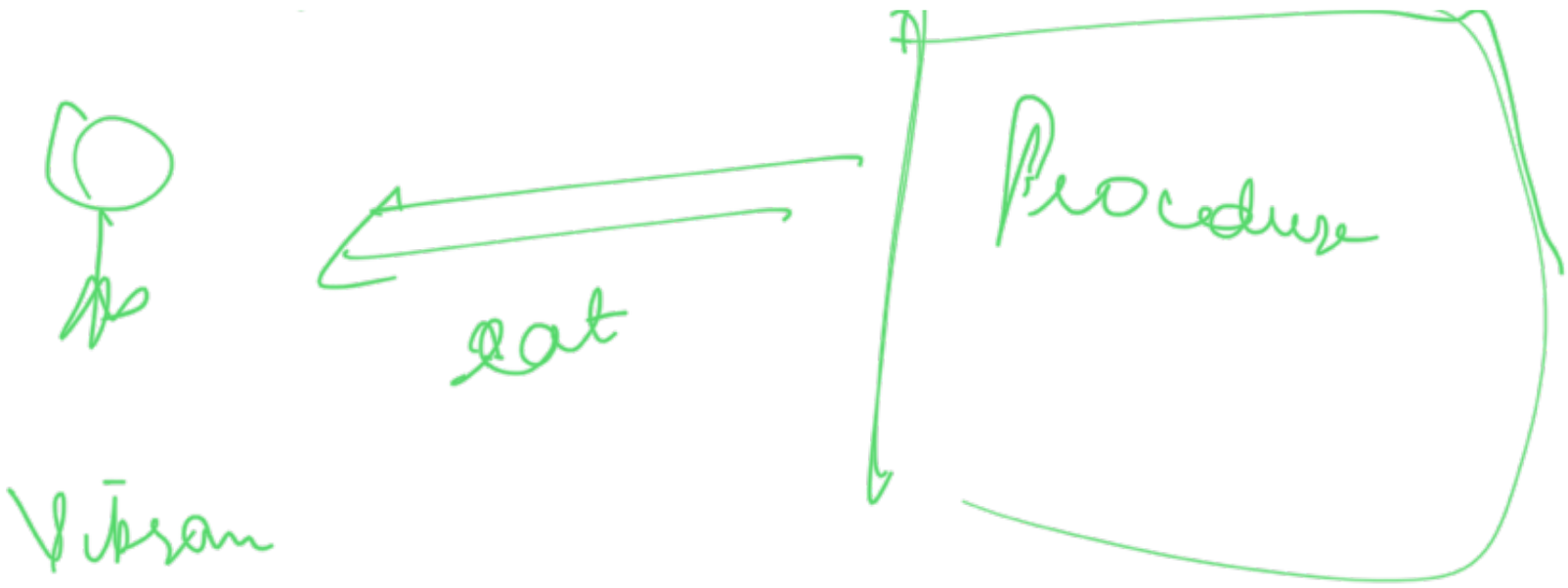
line by line

In procedural prog

we have data

we have procedures that do something to the data





Vikram Walke()

{

vikram.eat();

}

data

procedures

eat()

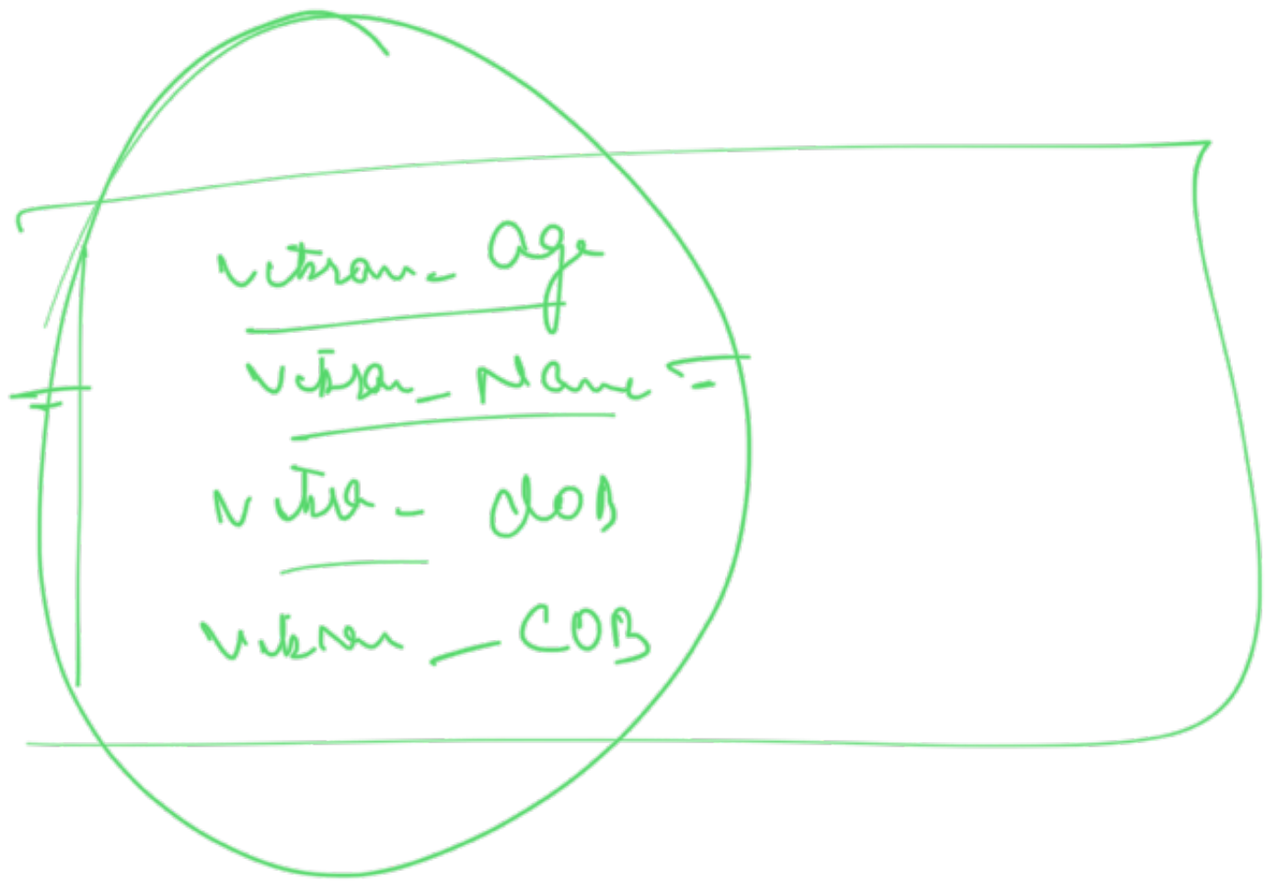
walk()

on land

Object Oriented Prog

→ Objects Know best about what all they can do.

⇒ Object should contain all its attributes



Virat

age

Name

dob

bob

eat() {}

dance() {}

walk() {}

drink() {}

OOP

Objet

details

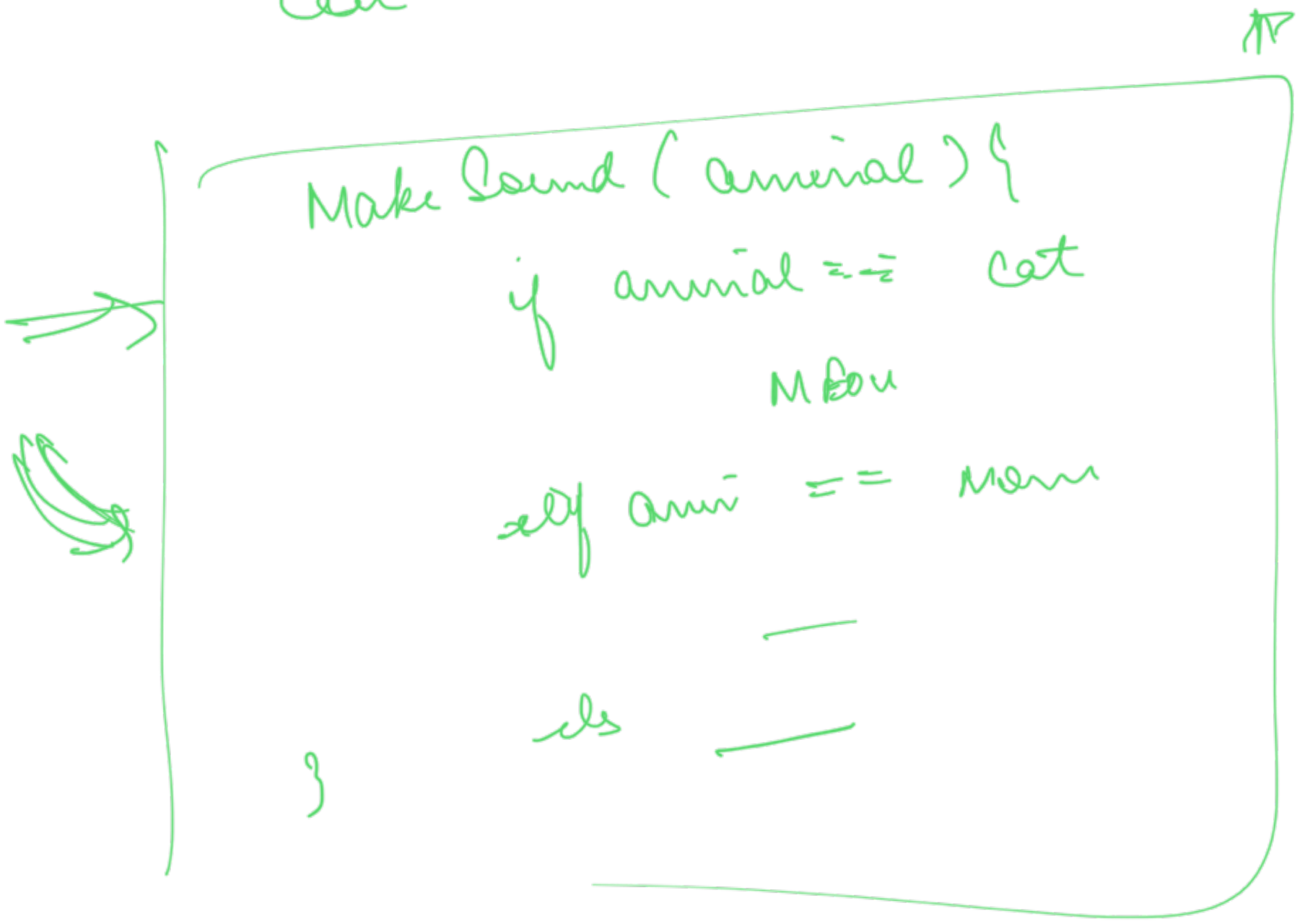
Beham

Zoo

tiger

Monkey

cat



eat (animal) {

if animal == cat

Animal

eat()

make Sound()

f^n

Procedural ==> f^n Prog

No

f^n are treated as

Query bar

first word entu obj

(pass f^n as parameter to f^n)

f^n are treated as objects/ data

f^n as param

assign f^n to var

Zoo

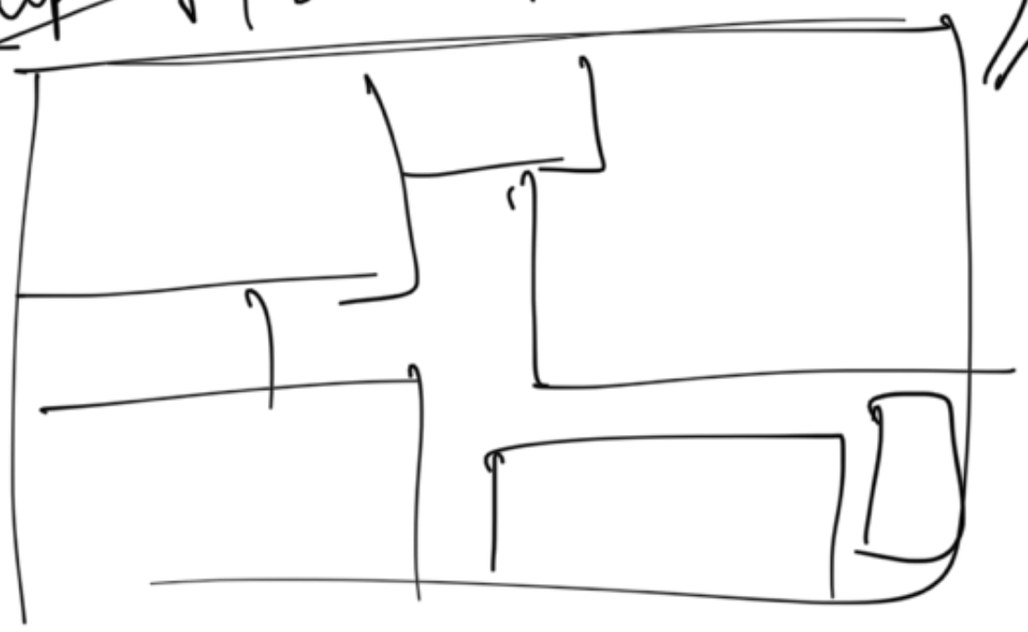
Animals

CLASS : Blueprint of a real entity

↳ Skeleton

Animal

Blueprint | Sketch | Ch



Blue color

Attributes / Properties / Fields

- Name
- NOOf Feet
- Color
- Size
- weight

Members of Class

eat()

run()

walk

Methods / behaviours

Members = Fields + Methods

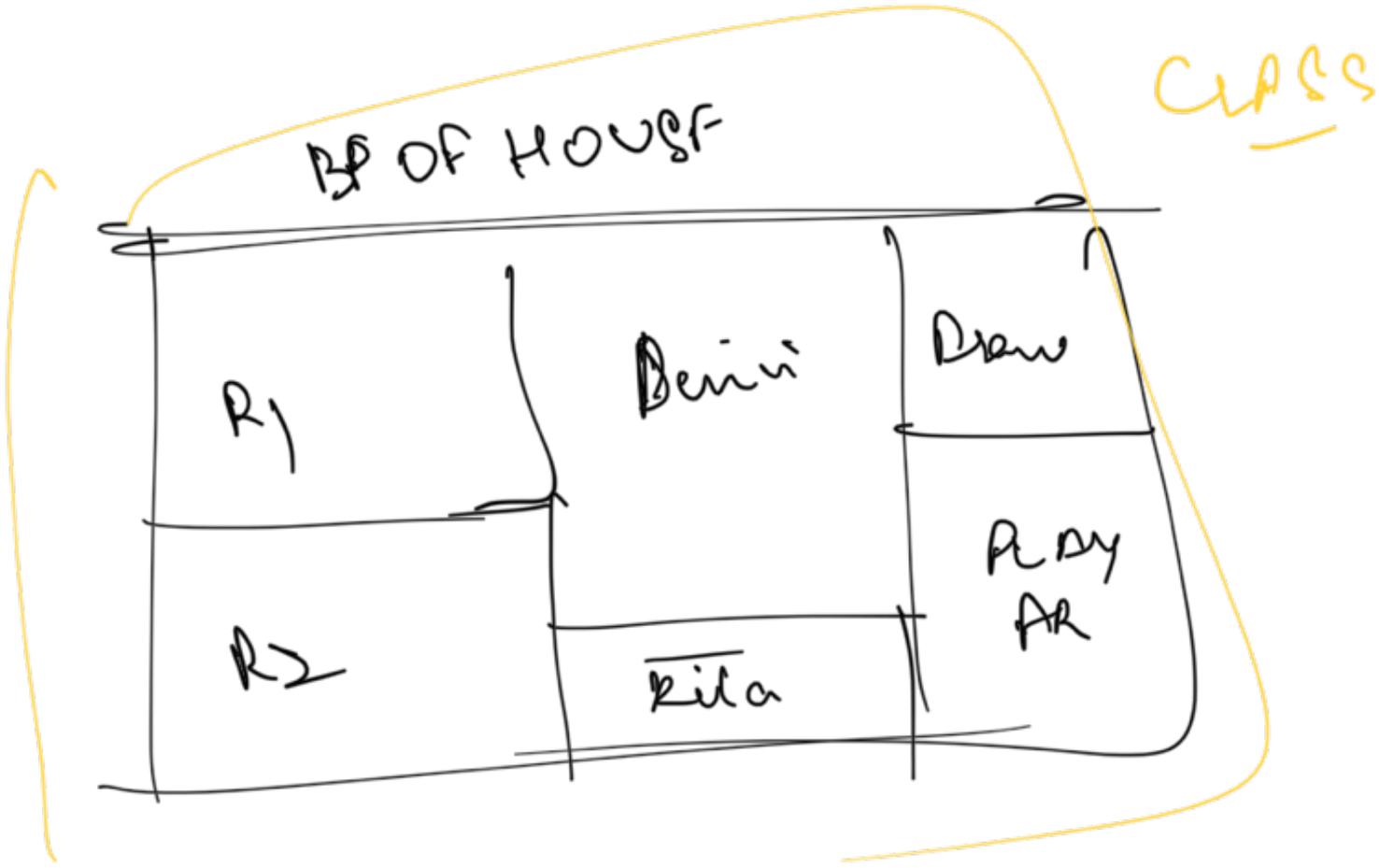
OBJECT

Real instance of a class

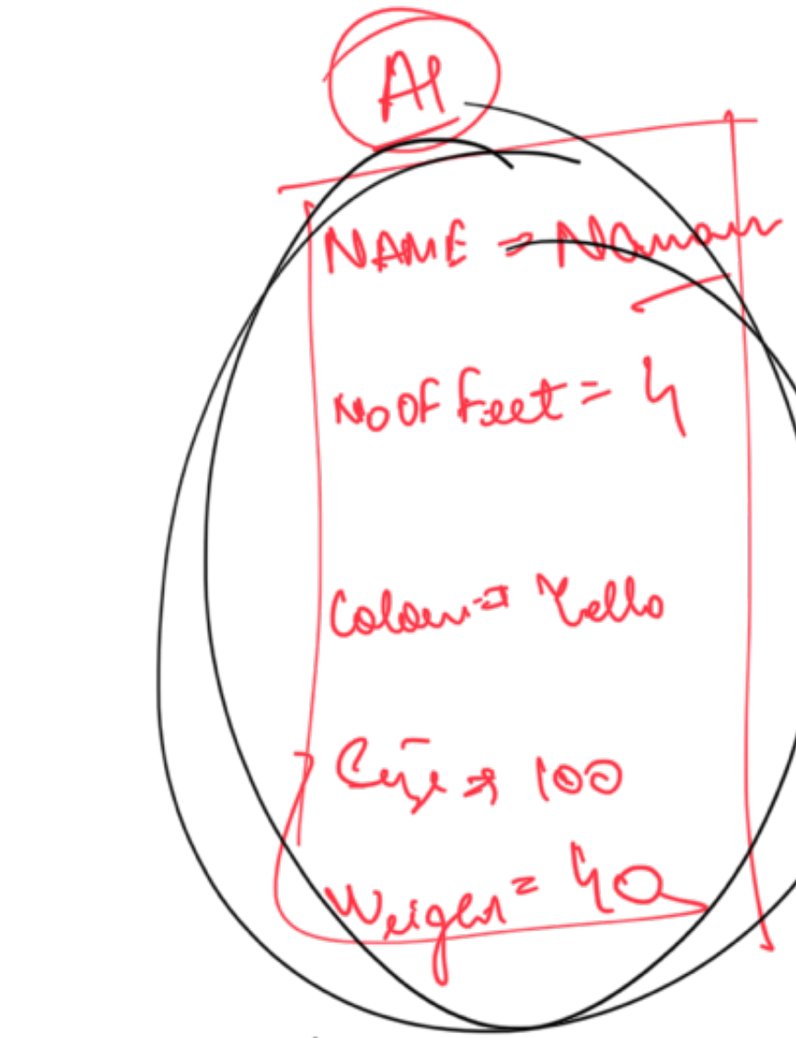
TAKE MEMORY

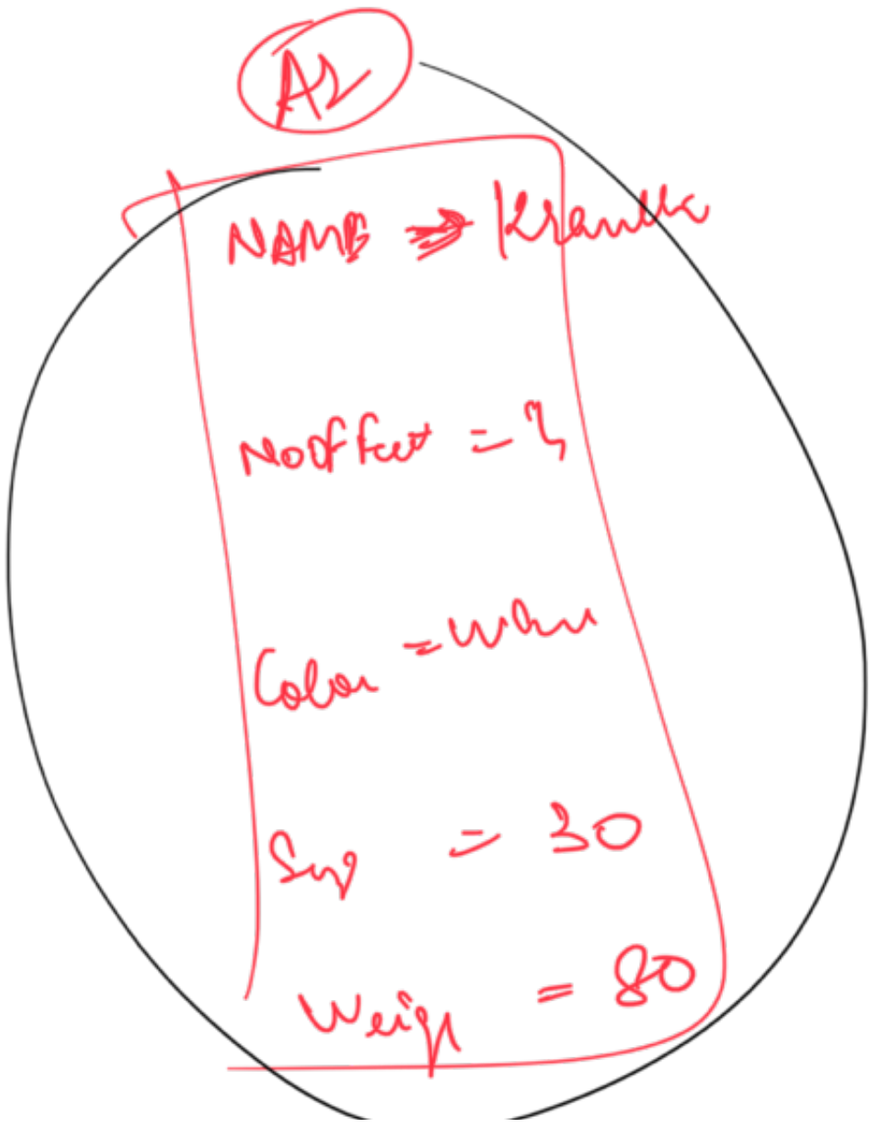
Animal c1 = New Animal()

Animal c2 = New Animal


CLASS
BP OF HOUSE
R1
Benin
Rita
Dave
Ajay
AR
R2

OBJECT


AI
NAME = Monu
No of feet = 4
Colour = Yello
Cost = 100
Weight = 40


AI
NAME = Kranthi
No of feet = 4
Colour = White
Cost = 30
Weight = 80

) Naman, 4, Yello, 100, 40)

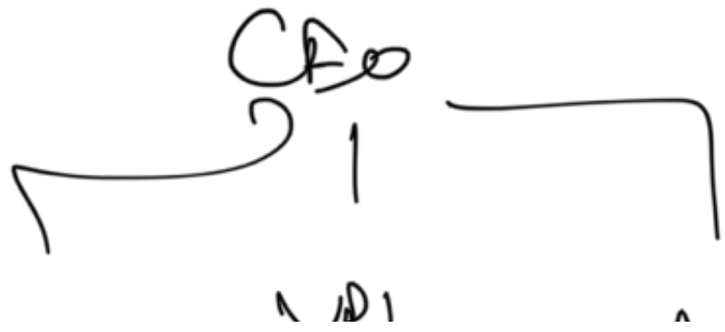
STATE of object

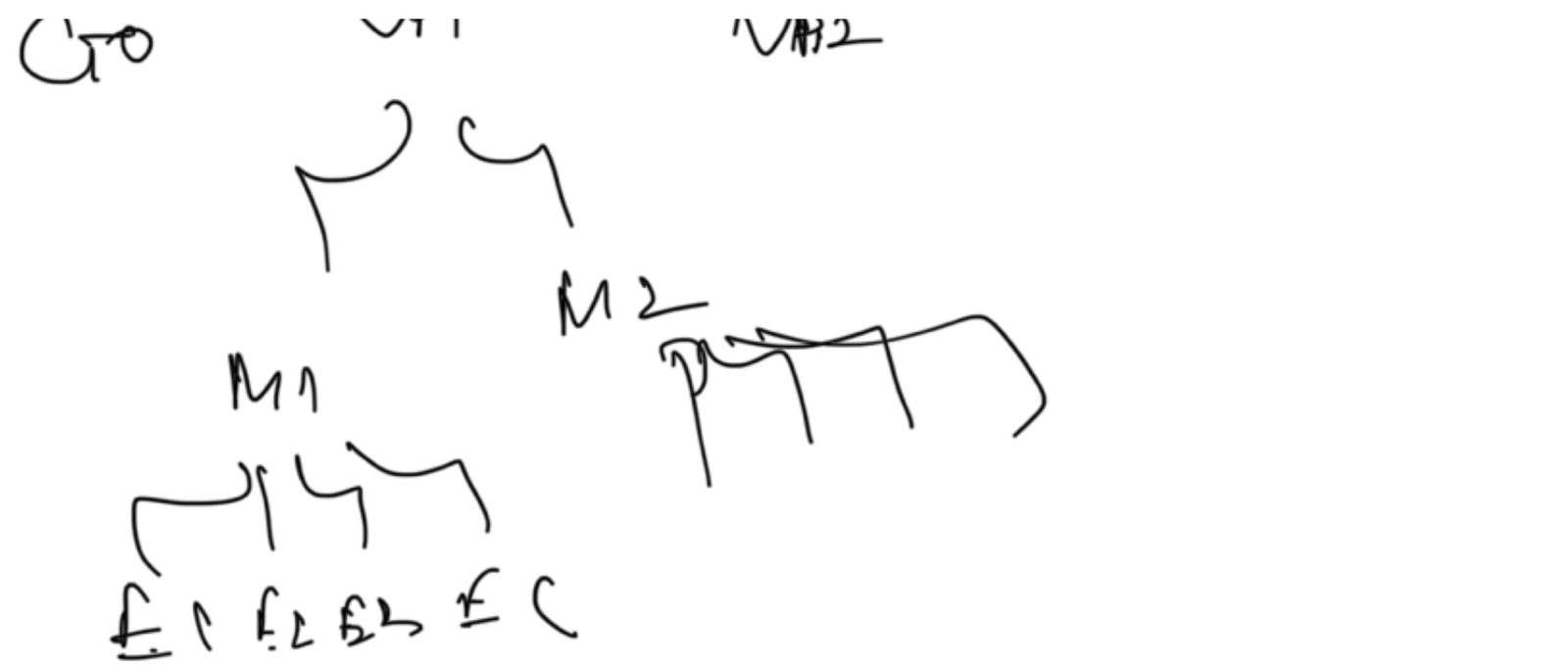
⇒ value of diff attributes of an object at a particular time

Al. Name = XYZ

) XYZ, 4, Yello, 100, 40)

Company





Parent Class
Superclass
Base Class

Animal

Attributes / behaviours

Inheritance

Child Class
Subclass

Mammal

Reptile

Birds


Llama
Dogs
Cat