Intro to LLD

LLD 1

→ CN5

AND

Intro to OOP

⟸ → Case Study (LLD - 7)

Agenda

→ What is LLD

→ Why learn LLD

↳ Interview Prep

↳ Career

20%

DSA+

> 70%

→ LLD Classes overview

→ # of classes

→ Curriculum

→ Takeaway

→ Assignments

Intro to OOP

Key terms related to OOP

→ Class

→ Object

→ Structural v/s Procedural v/s OOP

What is LLD

LLD: Low Level Design

High Level Design

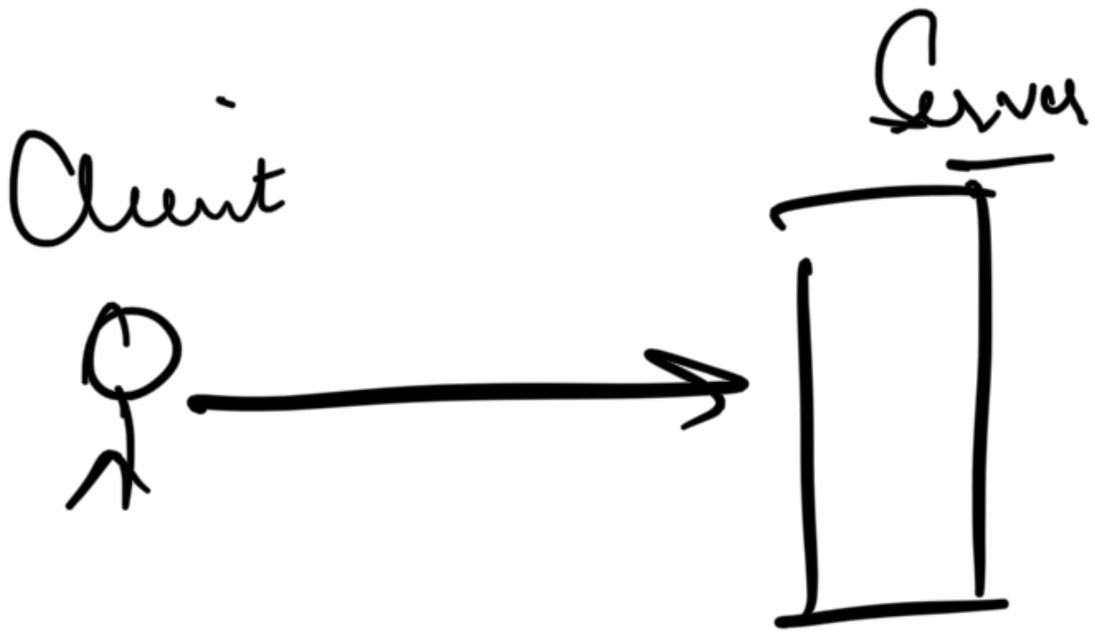
→ Overview

→ Bird's Eye

→ Not going in depth

OR

Software Systems


Client
Server


Less RAM
Slow CPU
Fast I/O
DB
Less RAM
Faster
App
More RAM
Fast CPU
Cache

Diff infra layers that work together to implement all the desired features of a software system at desired efficiency

Low Level Design : Going Deep into the code details of entire software system

⇒ Classes

→ Projects

→ Package

→ Metro

→ PM

→ BL

→ CEO

→ Manager

→ Interview

→ Gathering Requirement

→ Clarify Requirement

→ Design (Documentation
↳ design docs)

↳ Use Case Diagram

↳ Class Diagram

↳ Schema Design

→ Code ⇐

→ Testing =

ASSUME

ASS+U+ME

Why learn CD

≤ 12% writing code

Reading Code = Understanding code

88%

Result

→ Work ...

→ Meeting = Gather Req / Clarify Rel

→ Think / Analysis → Design

→ Cl...

→ S... Proble...

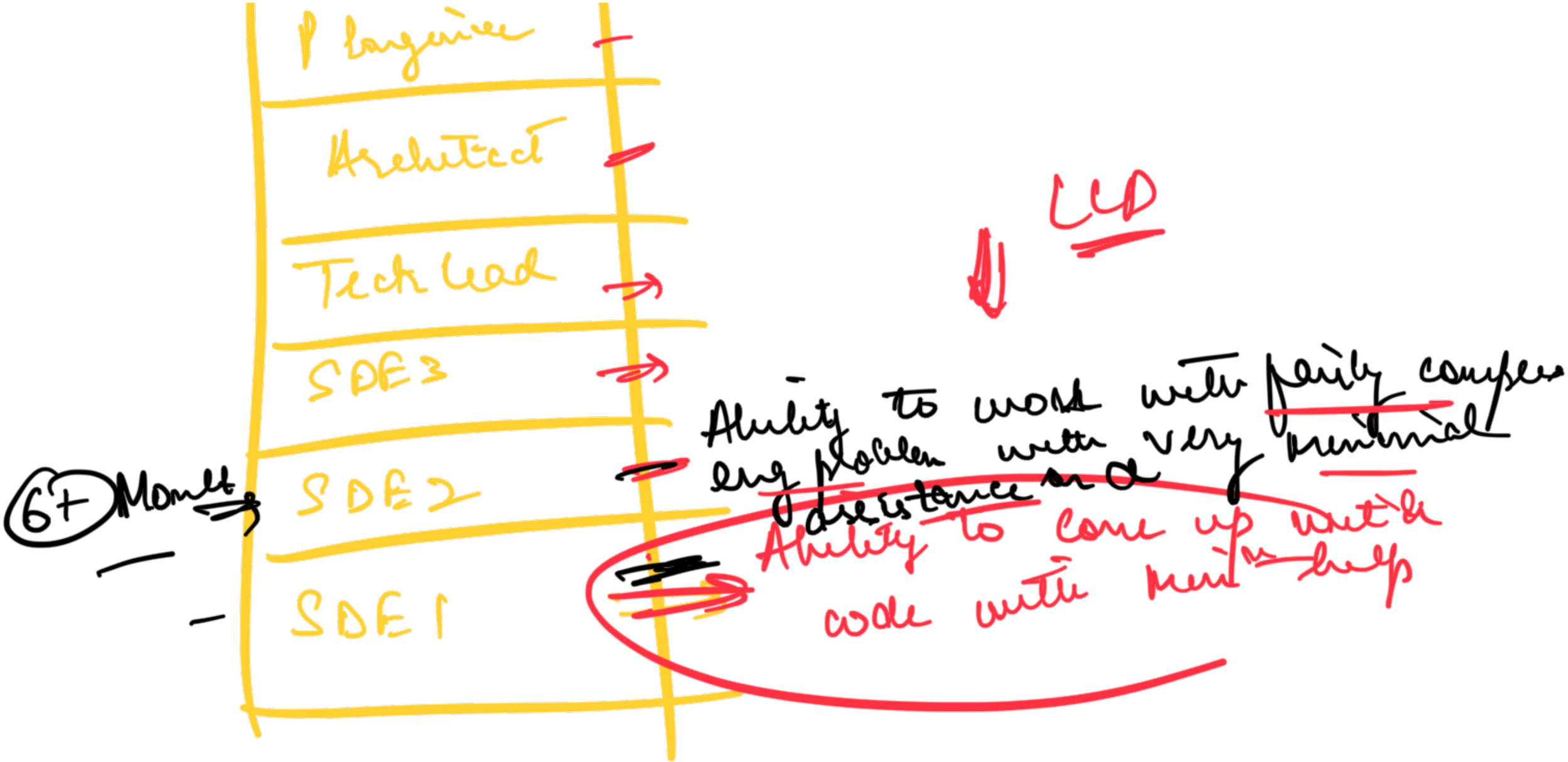
→ Testing — Easily Tested

→ Bug Fix

→ Debugging → Read Code & Understand

Engineering Ladder

HoE



Interviews

① Startups

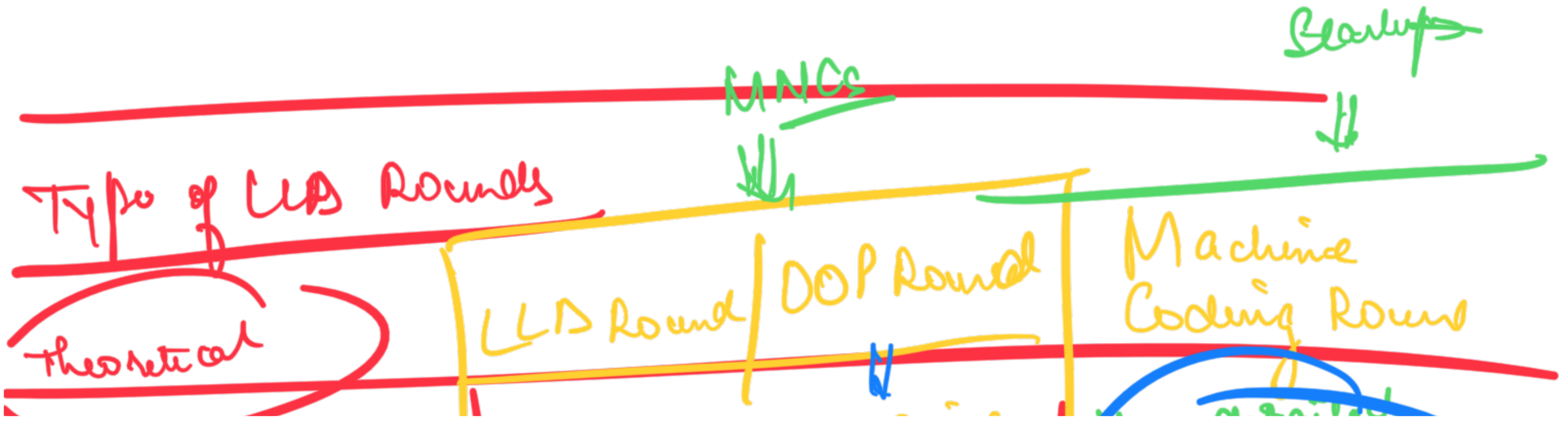
SDE1: atleast 1 LLD/ MC Round

Cred, Fk, Myntra, Inm

② MNCs

SDE 1: AirBNB, Coinbase, Slack, Uber

SDE 2: Every Compan



asked q² like
what is Cric
factors
adapt

↳ DP

↳ DP

↳ OOP Ten

UML
Diag

draw.io

① Very abstract single
line problem state-
ment

Design Snake &
ladder

→ Gather Req
(interest)
(ideas)
ask edge cases

→ Clarify Req

→ Use case Diagram

→ Class Diagram

→ Schema Design

Very detailed
problem
statement

X

Write SDE working

No code

write a single class

code.

Testing Skills (Unit Test)

Concurrency

Design BookMy Show

LLD Classes Structure (16 — 18 Classes)

21

① Intro to LLD and OOP

OOP

② OOP

* ③ SOLID principles

④ Creational DP

⑤ Creational DP

⑥ Structural DP

⑦ Behavioural DP

] * DP

* DP

⑧ UML Diagrams & Schema] → 1

⑨ How to approach LLD Problems

↳ Design a Lift

10 — 16 Case Study

Chess

Games

- Pen
- TicTacToe
- Snake & ladder
- Real Cricket
- Parking lot
- BMS
- Splitwise
- MailChimp

Engineering Systems

- Distributed Cache

Reder

Evaluation

3 Peer Review

+

1 TA Review

Score

Assignment

= ① MCQ → Auto → Submit a design image

② Design

UML

③ Code → Few missing Classes Implement M

Java

78%

draw.io

Submit

↓

Test Cases

↓

Score

Submit

Bridge Resource

Java Spring Boot

OS/Real

1 MOm

Backend

Pellet

Full

1.5 1.5 1 1
LLD + HLD + DB + F

→ Head First Design Patte

→ Effective Java

→ Refactoring. Jussi

10:35 PM

Intro to OOP

Build intuition of OOP

Types of Prog languages

1. Procedural
2. OOP
3. functional

©

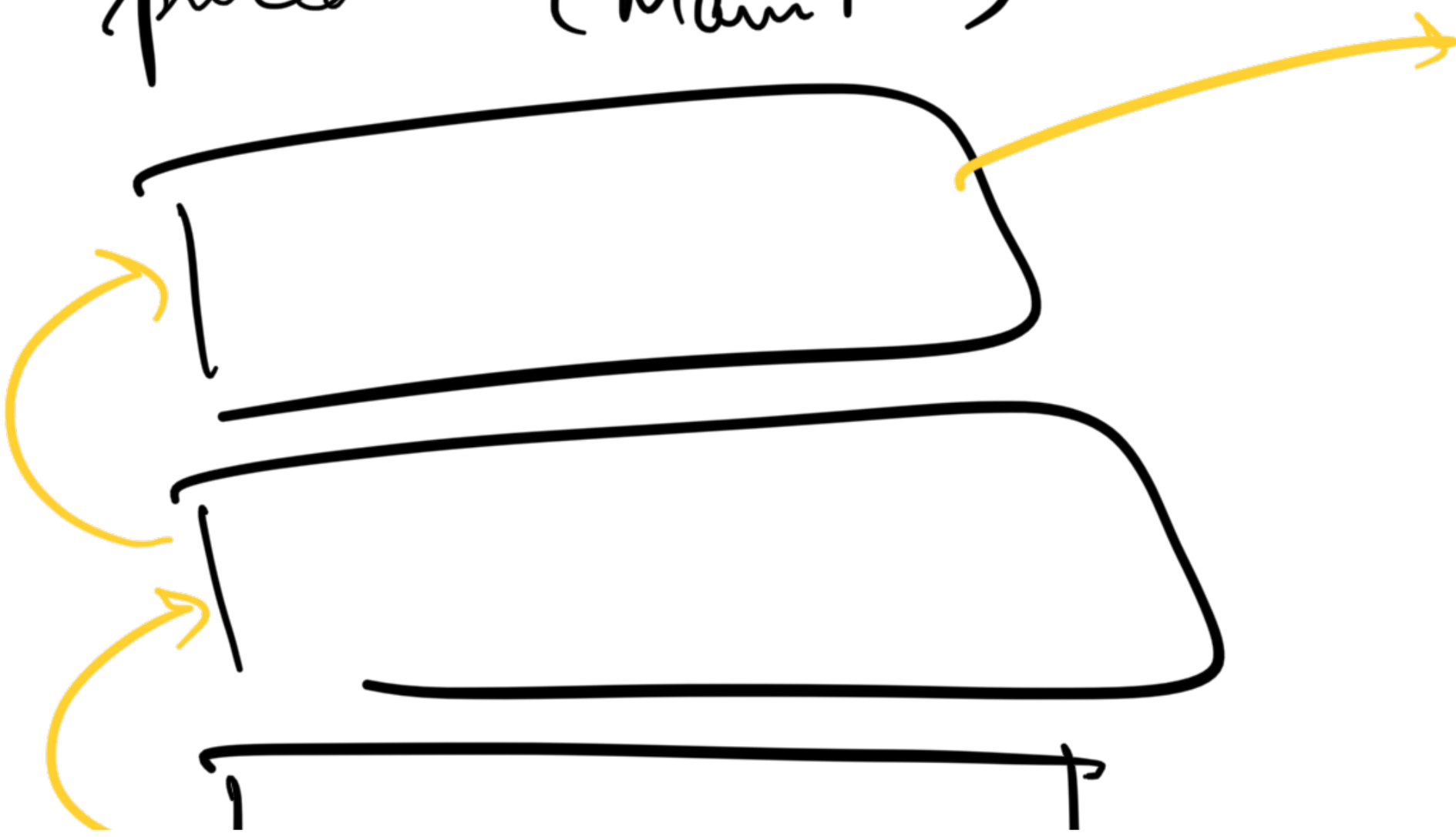
Procedural PL :

Procedure

```
def abc(s):
```

Procedure: group of instructions that work together by taking some input to perform a specific action

program is a set of procedures running in a desired order where execution of a program starts from a specific procedure. (Main Fn)



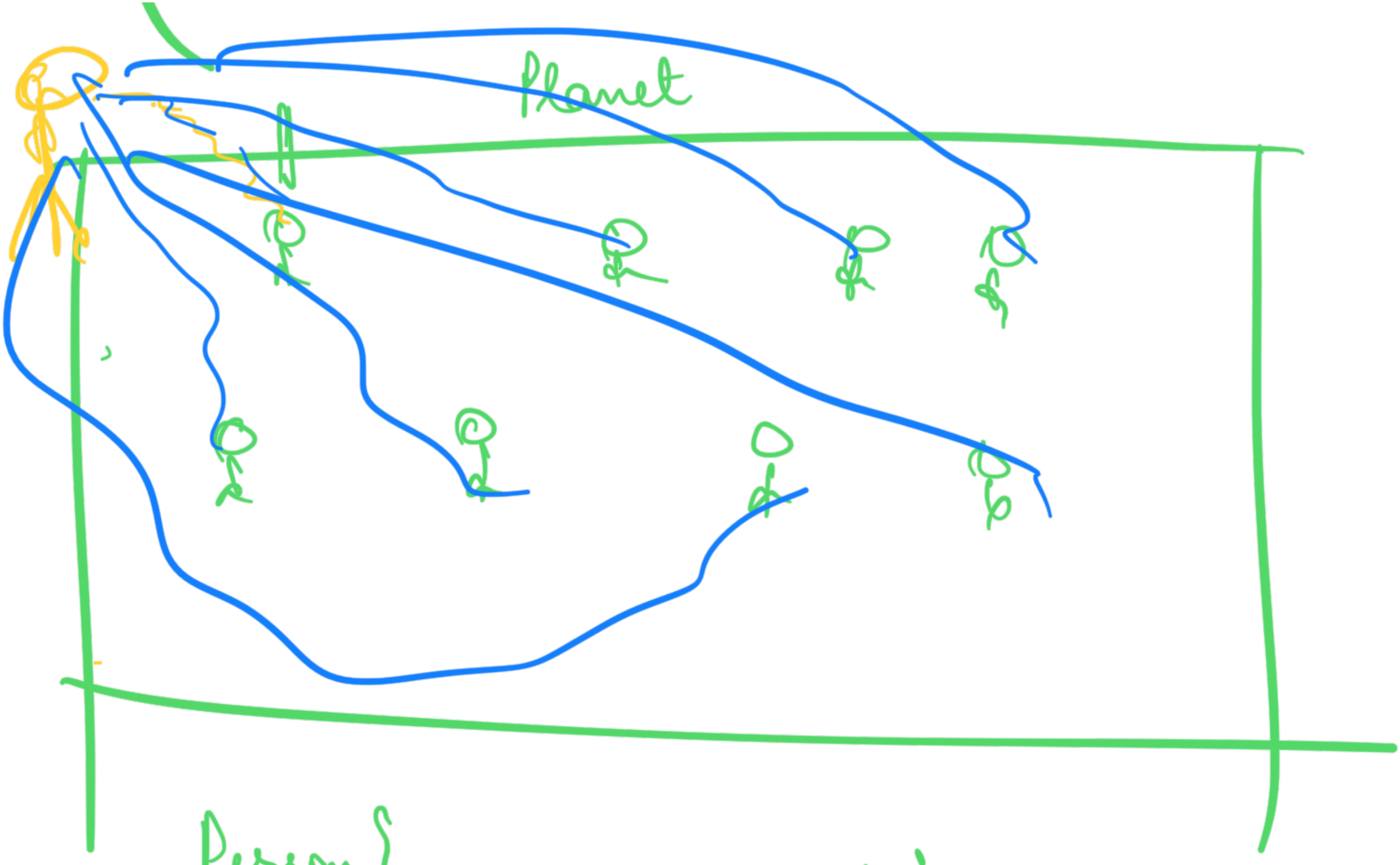
← Mani

Data Members : Set of attribut

. Is entity is nothing but a set of attributes

↳ Entities have No behaviour

⇒ If you have to perform an action on any entity, a specific procedure has to be called to do that


planet
Person 1
age
watu 1

Name

gender

)

Subject verb

Noun adj ~ Chi

entity performing action → ✓

OR

action being performed on an entity

Procedural

Puppet Holder ⇒ Procedure

Entities are a passive concept

Procedures are the central concept.

OOP

⇒ Entities are the central concept

⇒ Each entity now controls what all attributes it has

+

What all behaviour it can do

Person → in the Person class

Person -

fly Person (Person p) {

}

struct Person {

age
weigh
gender

} => Public

}

class Person {

OOD

Object Oriented Design

Procedure

Functional

Procedure ⇒ Procedures are the central entity

OOP ⇒ Objects/Entities are the central concept

Functional ⇒ Functions behave as full-fledged OOP

↳ pass fn as metho d parameter

↳ return a variable of fn type

Procedural + OOP + functional

Python

JS

Java

Struct {

}

main () {

}

1. HEAD FIRST DP
2. EFFECTIVE JAVA
3. REFACTORING.GURU