=> LLD :- things to core

=> Structure of LLD module

=> Intro to OOPs

---

* As per the task and time distribution of a developers => these are the major pillars :-

- → Read and understand code
- → Req. gathering
- → add more features / functionality of code
- → fix and maintain the code.

→ LLD helps to make our code :-

i) more readable and understandable

ii) req. gathering and task breakdown + estimations.

iii) Extensible

iv) Maintainable

extensible → easy to add more features and functionality

maintainable → easy to debug and fix issues.

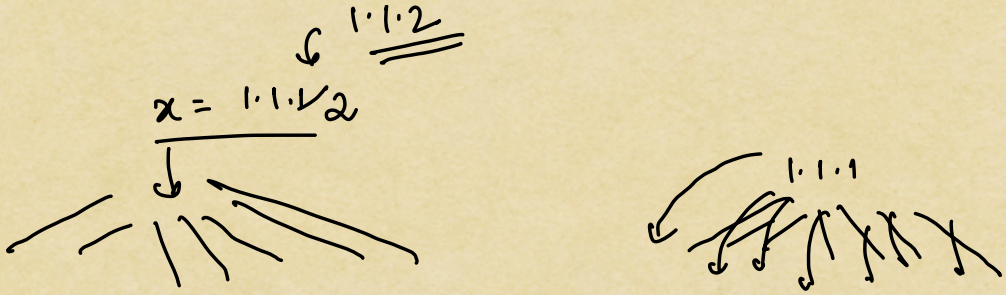
↓

easy to maintain the system in its current state.

maintainability

↓

ex => log4J issue ⇒ vulnerability




1) ⇒ fundamentals

- OOPs
  - → Abstraction
  - → Encapsulation
  - → Polymorphism
  - → Inheritance
- → Interfaces, Abstract class, (diff)
  - anonymous class
  - lambda
  - Streams
  - inner classes
  - pass by val vs. pass by ref.

- Design Patterns
  - → Creational (5)
    - Object creation
  - → Structural (3)
    - Code structuring
  - → Behavioural (2)
    - Behaviour of the code

- UML diagram
- Machine coding

plain Java ⇐
- → Tictactoe
- → Parking lot

Java + SpringBoot ⇐
- → BookMyShow
- → Splitwise

- → Simple problem statement
- → breakdown into multiple requirements
- → design → class diagram
  - → schema design
- → code implementation E2E
- → test and run

## ⇒ INTRO TO OOPS :

### Programming - paradigm :-

H.W explore

i) Procedural ⟶ C

ii) Object Oriented ⟶ Java, C++, Python, Typescript

iii) functional ⟶ Scala, Haskell

iv) Reactive ⟶ Java

v) Declarative ⟶ SQL, HTML

* **Procedural :**

procedure ] ⇒ set of instruction / function

Old age name for functions / methods

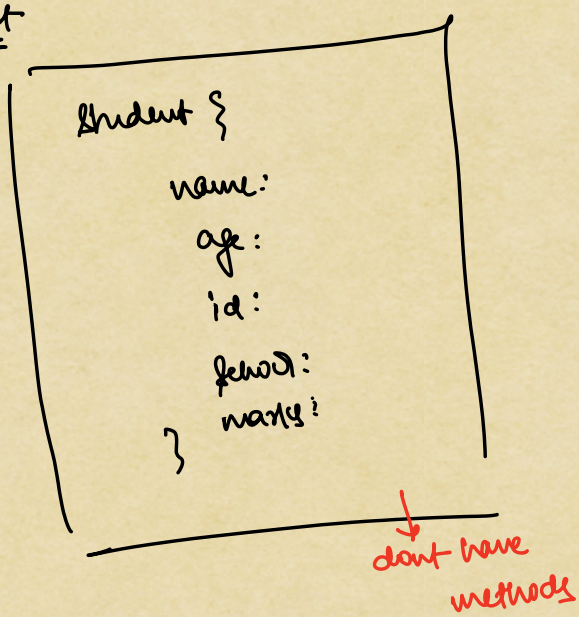
⇒ organise the code base into a bunch of procedures.

⇒ each of them might call other procedures internally

⇒ starts execution from a particular procedure

eg ⇒ (main())

store data => struct




struct and functions / procedures are separate

=> procedures are acted upon structs

action is being done on the subject

∴ subject + verb => subject does the verb

→ Sandeep is teaching ] real world thing

IRL => someone is doing something

OOPs follow IRL

[object] → data
→ action it can perform

print Student data:

procedural

```
void print (Student s){
print (s.name);
print (s.id);
}
```

```
Student{
    id:
    name;
}
```

OOPs

```
Student {
→ id
→ name
public void print(){
    print(id)
    print(name)
}
}
sObj. print();
```

subject executing the work.

⇒ Object oriented programming is easier to correlate with real life.

⇒ OOPs

⇒ how many pillars of OOPs ⇒ 3

⇒ principle ⇒ 1

1 principle + 3 pillars ✓ ✓

principle ⇒ fundamental foundations

pillars ⇒ support to always follow

⇒ principle ⇒ ABSTRACTION

⇒ pillars ⇒ Inheritance, Polymorphism, Encapsulation

⇒ **ABSTRACTION**

hiding any extra info.

or. showing bare minimum details.