Agenda

1. What is LLD?
2. Why is LLD important
3. LLD module at Scaler
4. Intro to OOPS

Ways of working:

1. Lectures start at 9:05 PM
2. Use public chat for doubts
3. While teaching no doubts, will take up questions after a topic is covered.
4. Please join WA group.

Intro

Bhavik Dand

6+ YOE

VESIT, Mumbai University → Comp Engg.

MAQ S/W (1.25 Yrs)

Paytm (2-3)

Clevertap (1.25 Yrs)

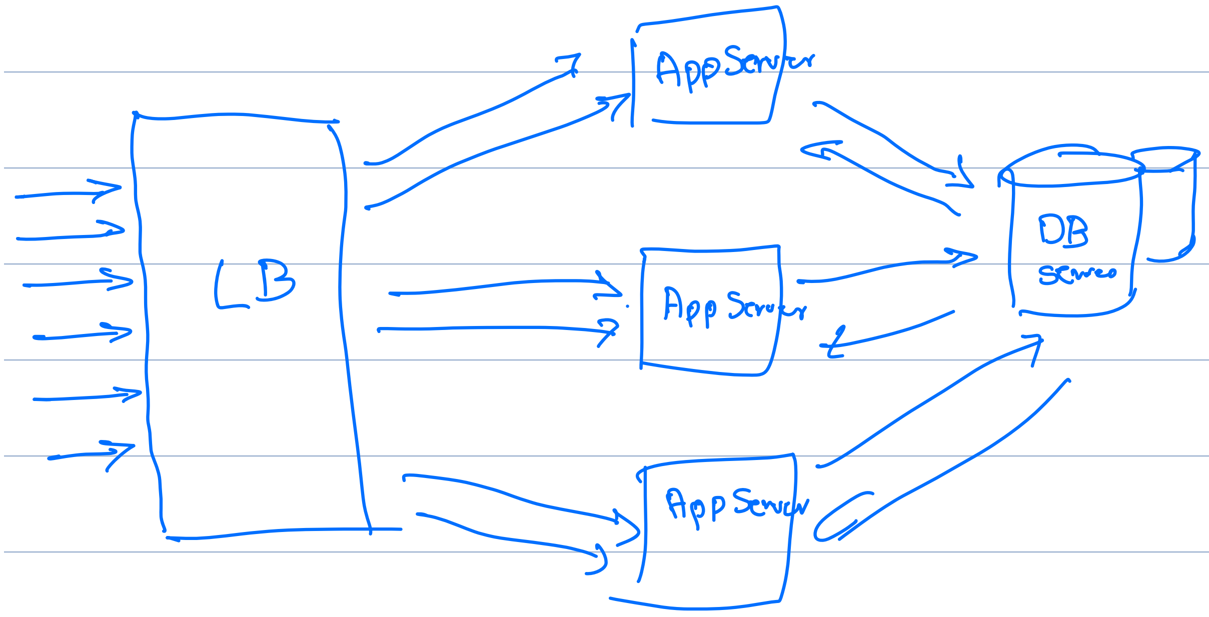
Dunzo - SDE2 (1 Yr)

Scaler (1yrs) Akto.io (2 Yrs)

1] What is LLD? Low Level Design

HLD: High Level Design

- Birds eye view of system.
- How different components / systems are connected to each other.



Essentially all the different components are just servers with different configurations running diff software.

<u>LLD</u> : How is the code structured? How are different modules inside our codebase interacting with each other?

2] Why is LLD important?

Startups : SDE 1 Atleast 1 LLD round
FAANG : Generally ask OO Design question but not Full fleged LLD rounds.

Day - to - Day : ✓ ① Write new code
✓ ② Review code
2 ✓ ③ Debug issues
✓ ④ Requirement gathering

88% of time of engineers is spent doing activities related to LLD.

Goals of LLD:

① Maintainable : We should be able to fix issues efficiently
② Extensible : It should be easy for us to add new features.

<u>LLD</u> <u>module</u> <u>at</u> <u>Scaler</u>

① Advanced Programming Concepts (LLD 1)
② Design Principles & Patterns (LLD 2)
③ Project Building (LLD 3)

① APC

✓ OOPS (4)

✓ Concurrency (4) ***

Collections & Generics (1)

Exception, Annotation & Reflection (1)

Contest - 1 → Mock Interview **

② Design P & P :

First 2 lectures : SOLID Principle

Next 7 classes : Design Patterns

$$\begin{tikzcd}
 & \text{Design Patterns} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Creational} & \text{structural} & \text{Behavioral}
\end{tikzcd}$$

Contest - 2

③ Project Building :

How to approach Design Problems?

Tic Tac Toe

Parking Lot

BMS

Google Calendar

Pending Project Module

Contest - 3

Intro to OOPs (Java)

Any living or non-living for which we want to store info.

Entities are at the core,

Entities are made up attributes

& behaviours (method / function / procedure)

Scaler → Learners, Mentors, Batches, TA,

Instructors, Classes, Assignments,

Contest.

Represent student Entity

```
            class Student {
                int age;
attributes {    String name;
                float psp;

                void changeBatch(){..} ;
behaviours {    void joinClass(){..} ;
                void coursePause(){..}:
            }
```

4 main concepts: Abstraction
Encapsulation
Polymorphism
Inheritance.

1 Principle : Abstraction
3 Pillars : Polymorphism
Encapsulation
Inheritance

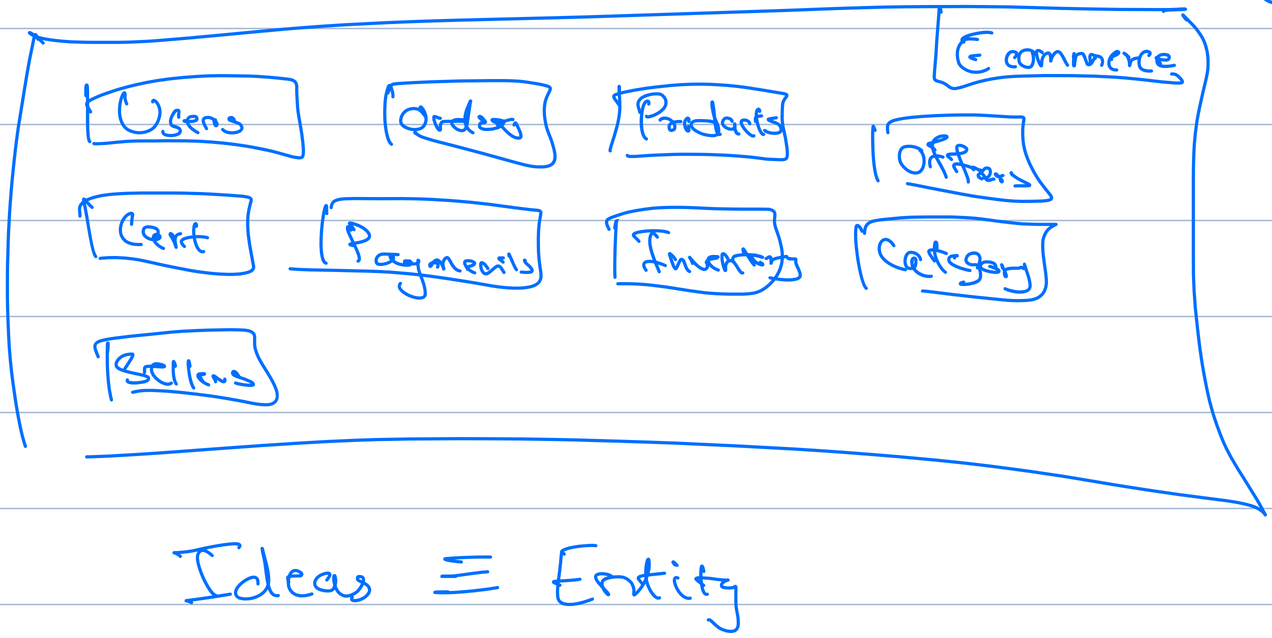
Principle : rule / siddhant

Pillar : support

Abstraction

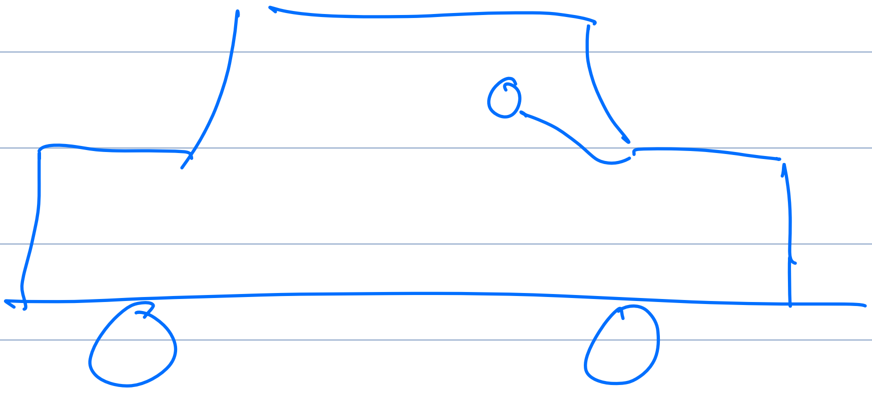
Abstract: idea ; something i.e. not in existence.

① Abstraction asks us to envision a complex system in terms of ideas/ entities



Ideas ≡ Entity

② Hiding complex details:



Driver of car doesn't care about how the wheels turn on turning steering wheel.

```
Class EmailHelper {
    void sendEmail ( string content, string from,
                     string to ) {
        :
        :
    }
}

Class Student {
    void changeBatch (        ) {
        :
        EmailHelper eh = new EmailHelper()
            eh. sendEmail ( ↑ , ↑ , student.email) ;
                                    ↑
}
```

Abstraction Summary:

① Represent complex s/w systems in term of ideas / entities

② Hiding details of complex implementation.

2] Encapsulation :

Capsule : ① holds medicine together

② protects the medicine from outside.

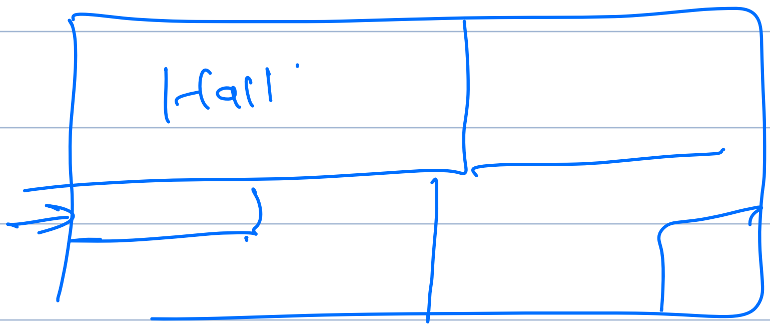
Encapsulation in OOPS :

① Holds attributes & behaviours together $\Rightarrow$ via class

② Protects attrs & behaviours from outside world $\Rightarrow$ access modifier

Terms in OOPS :

① Class : Blueprint of an entity.



Blueprint of house.

Class occupies no memory.

② Objects : Depending upon class, you can Create instances of that class.

Objects occupy space.

Access Modifiers

Access modifiers defines who accessible is a member of a class.

$$\begin{tikzcd} & \text{member} \arrow[dl] \arrow[dr] & \\ \text{methods} & & \text{attributes} \end{tikzcd}$$

```
class Student {
Attribute → access_modifier data_type attr_name;
Method    → access_modifier return_data_type
                method_name ( . . . ) {

            }
}
```

4 access modifiers in Java:

1. private
2. default
3. protected
4. public

1. private : then that member is only accessible within the class. ~~No~~ No other code outside the class can access that member.

2. default:
   To use this, don't write anything before the member declaration.
   Any class within package can access this member.

③ protected

Any member which is using protected access modifier can be accessed via its child class.

same package ✓ diff package. ✓

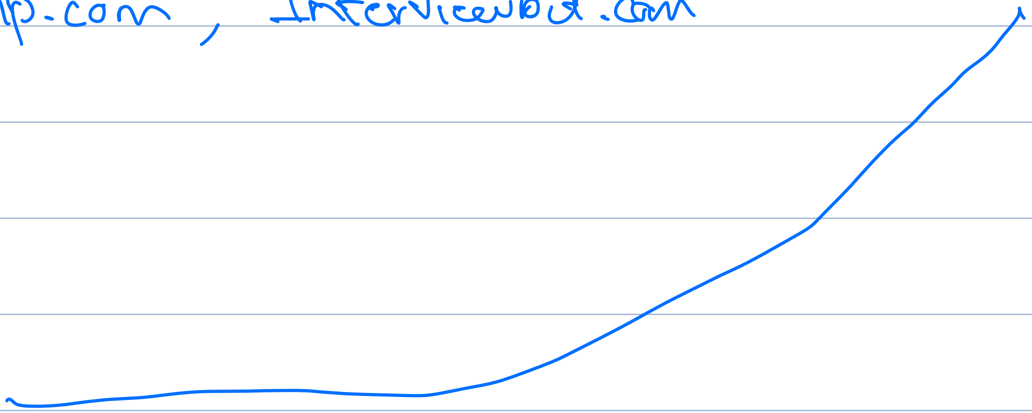
④ Public : most lenient access modifier.

a public member can be accessed from anywhere

same class ✓ same package ✓ diff package ✓

Access modifier	Same Class	diff class same package	child class diff package	diff class diff package
private	✓	✗	✗	✗
default	✓	✓	✗	✗
protected	✓	✓	✓	✗
public	✓	✓	✓	✓

Pramp.com , Interviewbit.com



SQL Profiler

↳ In depth analysis

↳ Which queries slowest queries?

↳ Which queries are not utilizing index?