Agenda:

Rahul
↳ +91-8847433272
↳ Slack
↳ rahul.grover_1@scaler.com

1. What is LLD?
2. Why is LLD important?
3. LLD module structure at Scaler.
4. Intro to OOPs → Programming Paradigms
   → Principles and Pillars
   → Classes and objects

   Github →

5. Demo → Intellij →
   ↳ Github →

Note:
1. Please try to join by 9:05 pm
2. Questions → PUBLIC chat
3. Doubts after every topic finishes
4. Please join W.A group.

[2 hrs 15-20 mins]
↑

What is LLD ? → Low-Level Design.

Opposite : High Level Design

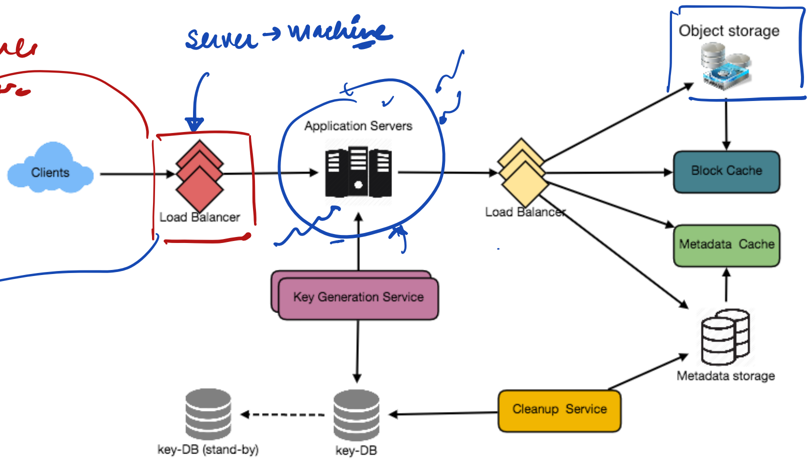
→ End to End Architecture
→ Birds eye view
→ How the systems are interacting with each other.

DB Server → X

software on a machine

n. Open source software

Server → machine



CPU
RAM?

Disk storage

Essentially all different components are just servers with diff configuration.

LLD → CODE ⇝ How is code structure?
↳ Different modules organised
↳ Codebase is interacting with each other.

Why is LLD important?

DSA

Dunzo → 2 LLD rounds

Interviews:

Startups → SDE1, SDE2 ...

FAANG → SDE1 ⇝ DSA
⇝ OOPS, SQL
SDE2 ⇝ SDE3, TL ↳ Design

Design → HLD, LLD

Day to Day Activities:

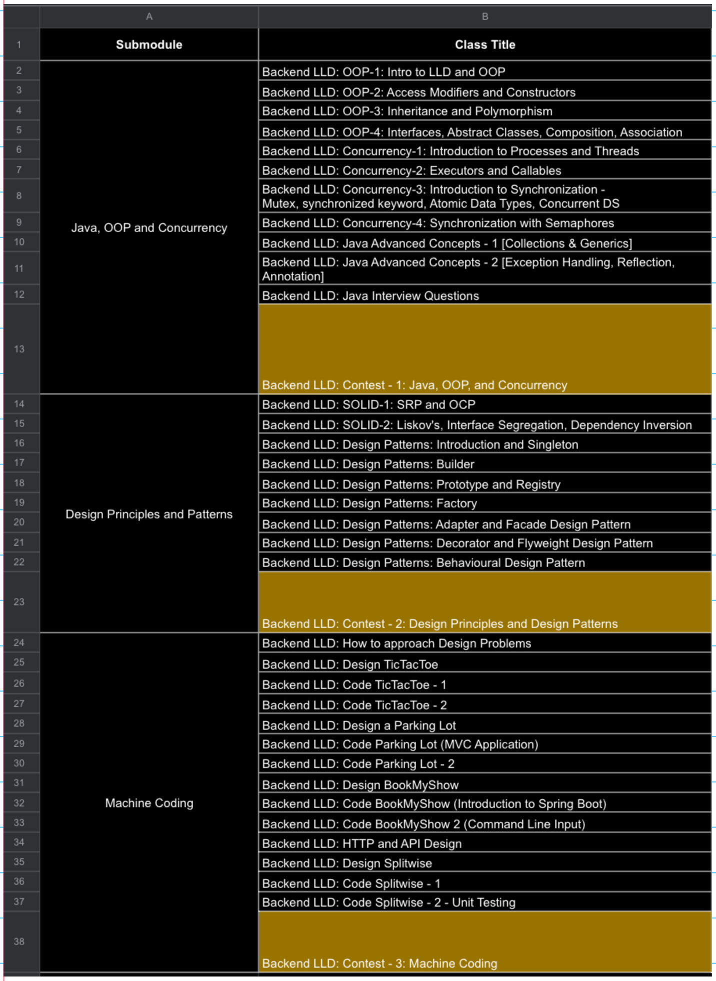
Office:
- Stand up ⇝
- Design Do. ⇝ Designs →
- Coding → ✓
- Meetups
- Code Reviews ⇝ Reading code
- Coffee → is
- Production Issue ⇝ Reading code and logs
- Planning
- Unit Testing ⇝

Goals of LLD:

1. Maintainable → Bug, Readable, Reliable
2. Extensible → Adding a feature

[How much code you had to change or had to add]

LLD Course Structure @ Scaler :



	A	B
1	**Submodule**	**Class Title**
2	Java, OOP and Concurrency	Backend LLD: OOP-1: Intro to LLD and OOP
3		Backend LLD: OOP-2: Access Modifiers and Constructors
4		Backend LLD: OOP-3: Inheritance and Polymorphism
5		Backend LLD: OOP-4: Interfaces, Abstract Classes, Composition, Association
6		Backend LLD: Concurrency-1: Introduction to Processes and Threads
7		Backend LLD: Concurrency-2: Executors and Callables
8		Backend LLD: Concurrency-3: Introduction to Synchronization - Mutex, synchronized keyword, Atomic Data Types, Concurrent DS
9		Backend LLD: Concurrency-4: Synchronization with Semaphores
10		Backend LLD: Java Advanced Concepts - 1 [Collections & Generics]
11		Backend LLD: Java Advanced Concepts - 2 [Exception Handling, Reflection, Annotation]
12		Backend LLD: Java Interview Questions
13		Backend LLD: Contest - 1: Java, OOP, and Concurrency
14	Design Principles and Patterns	Backend LLD: SOLID-1: SRP and OCP
15		Backend LLD: SOLID-2: Liskov's, Interface Segregation, Dependency Inversion
16		Backend LLD: Design Patterns: Introduction and Singleton
17		Backend LLD: Design Patterns: Builder
18		Backend LLD: Design Patterns: Prototype and Registry
19		Backend LLD: Design Patterns: Factory
20		Backend LLD: Design Patterns: Adapter and Facade Design Pattern
21		Backend LLD: Design Patterns: Decorator and Flyweight Design Pattern
22		Backend LLD: Design Patterns: Behavioural Design Pattern
23		Backend LLD: Contest - 2: Design Principles and Design Patterns
24	Machine Coding	Backend LLD: How to approach Design Problems
25		Backend LLD: Design TicTacToe
26		Backend LLD: Code TicTacToe - 1
27		Backend LLD: Code TicTacToe - 2
28		Backend LLD: Design a Parking Lot
29		Backend LLD: Code Parking Lot (MVC Application)
30		Backend LLD: Code Parking Lot - 2
31		Backend LLD: Design BookMyShow
32		Backend LLD: Code BookMyShow (Introduction to Spring Boot)
33		Backend LLD: Code BookMyShow 2 (Command Line Input)
34		Backend LLD: HTTP and API Design
35		Backend LLD: Design Splitwise
36		Backend LLD: Code Splitwise - 1
37		Backend LLD: Code Splitwise - 2 - Unit Testing
38		Backend LLD: Contest - 3: Machine Coding

Intro to OOPs

↳ Object-oriented Programming

[Programming Paradigms :]

- style of writing code
- structuring / organizing

Some common.

1. Imperative → assembly, BASIC
2. Procedural → C, Pascal
3. Functional → Haskell → functions → First class citizens
4. Declarative → SQL → Specify what you want ↓ But don't specify how
5. OOPs → Java

Make tea → Get water in Pan
Heat it
Sugar, chai patti

Prep the Pan( )

↳ Add ingredient( ) → { sugar, chai, patti

Wait and heat

Pour 'i'n lup

Make a hotel booking in procedural programmin

[email, phone, id, ...]

bool makeABooking (User user , Room room)

checkAvailability (Room)

debitAmount ( )

updateStorage (user , room)

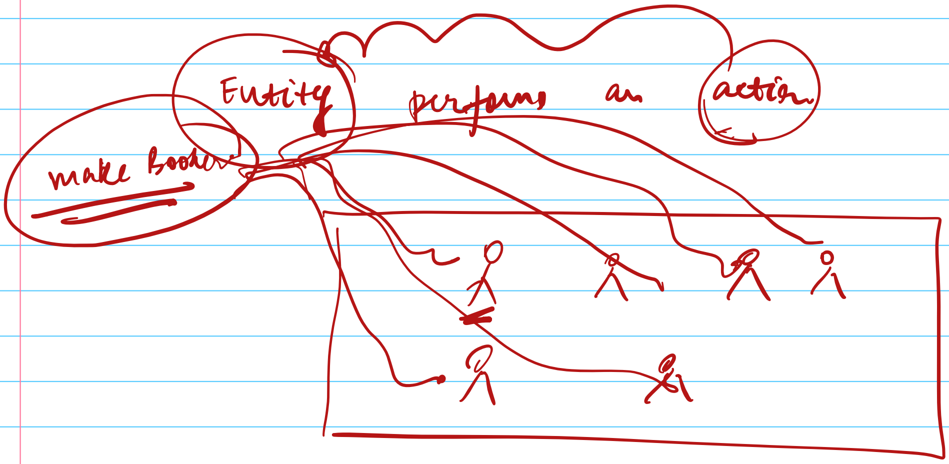
[Price
id
Beds]

→ [Performing an action on user]

→ We have to pass all info about that user

Data of a user is not safe

Privacy Issues



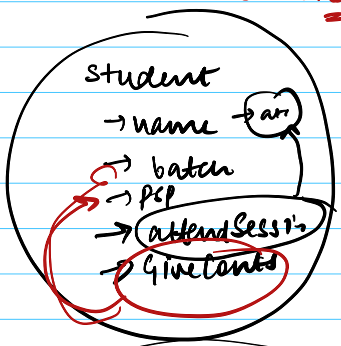
In procedural programming, it gets difficult to model real world systems

[5 mins break]

OOP →

Entities / Objects at the core

→ Properties / attributes
→ Behaviours

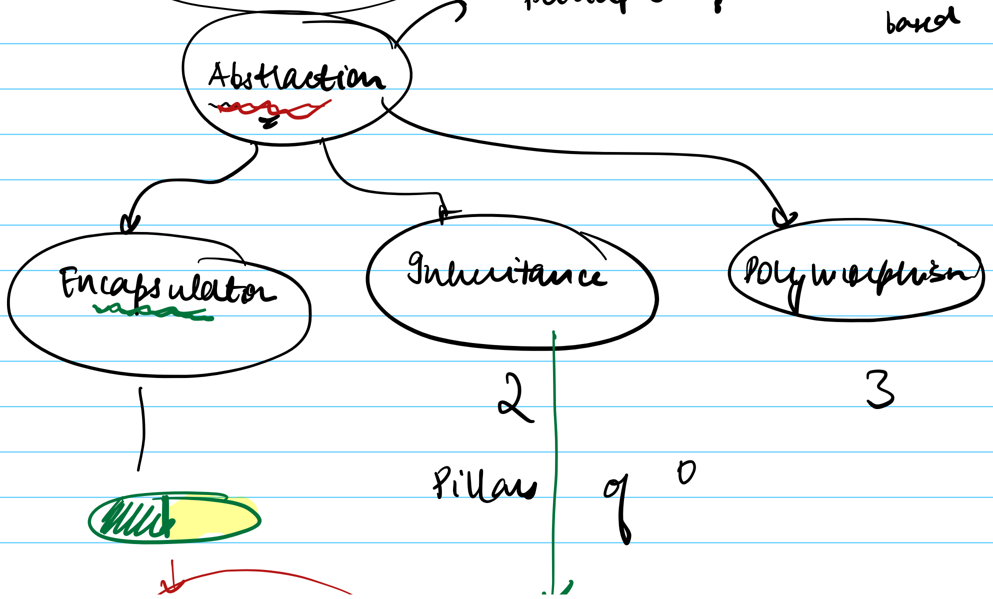


4 main concepts

③, ④

Java → Complete Reference

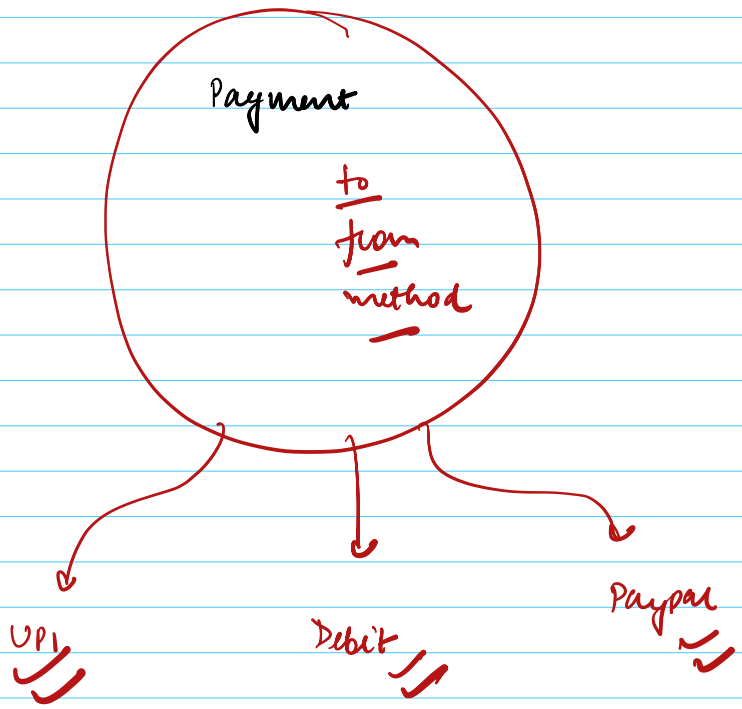
Principle of which OOPs based

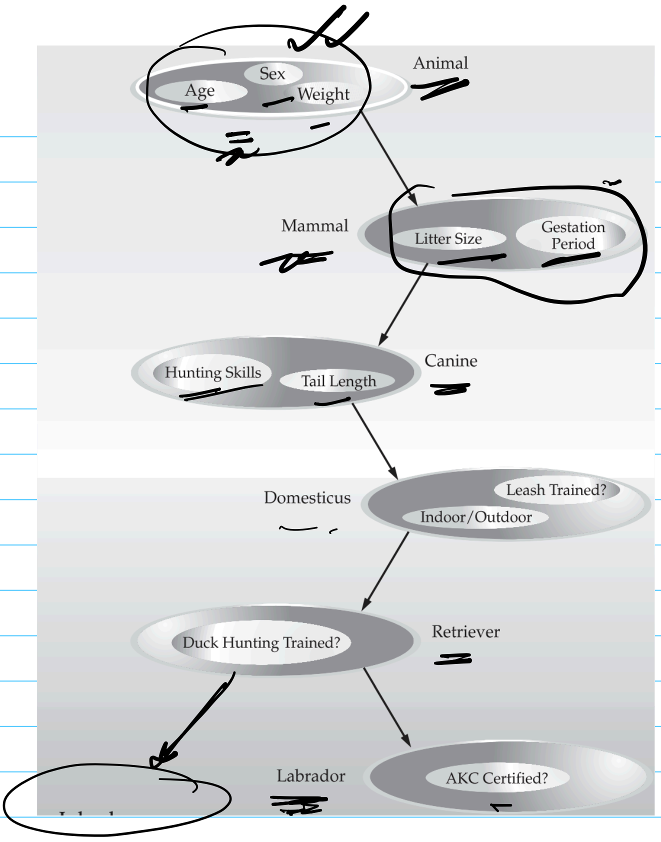


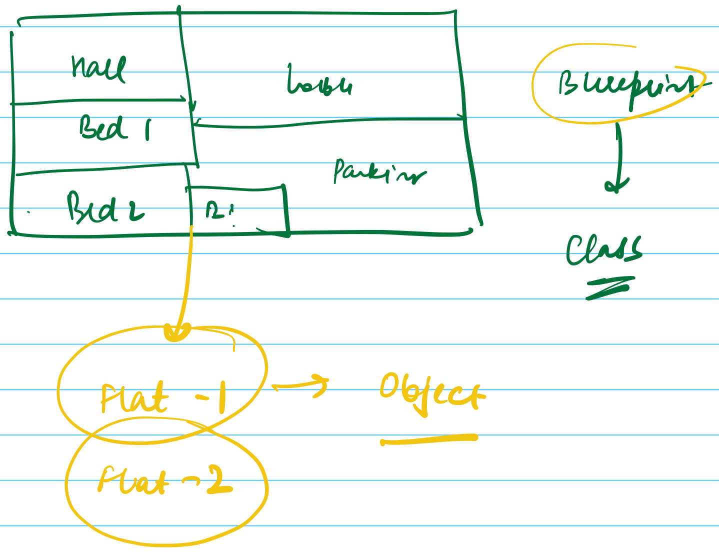
Pillars of O

1. Hiding
2. Holding it together

Classes
- Attributes / Properties
- Behaviours




Animal
Sex
Age
Weight
Mammal
Litter Size
Gestation Period
Canine
Hunting Skills
Tail Length
Domesticus
Leash Trained?
Indoor/Outdoor
Retriever
Duck Hunting Trained?
Labrador
AKC Certified?


hall
Lobby
Bed 1
Parking
Bed 2
B:
Blueprint
Class
Flat -1
Object
Flat -2