=> How to approach design problems

-> Types of LLD interviews :-

Theoretical	Design	Machine Coding
* Old tech companies, Banks, Consulting firms * Oracle, Kotak, Morgan Stanley, PwC, Deloitte ---- * Test knowledge on syntax, OOPs, language etc.	* Tech companies * Vague problem statement, asked you to design it * No code * Class diagrams	* End to end code * No need for diagrams * Properly understood problem statement -> 2 hours

4 projects

i) Tic Tac Toe => started with machine coding

ii) Parking lot => Layered architecture (models, repository, services, controllers)

iii) BookMyShow =>

1) Spring Boot with DB (JPA)

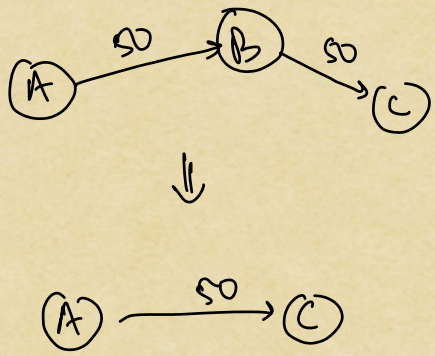
2) 1 person gets 1 seat [concurrency, isolation levels]

↳ iv) Spitwise ⇒

i) SpringBoot

ii) Unit testing (?) ← project module

iii) write the algorithm to minimize transactions



How we will follow the machine coding

i) 1 live PS

ii) Clarify requirements

iii) Requirement gathering

iv) Class diagrams

v) Schema Design

[vi) E2E working code

1 lecture

2-3 lectures

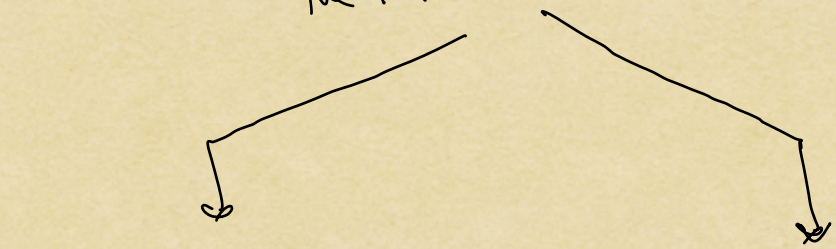
prepare you for both design + machine coding

→ Design a Pen

Step 0 → Get an overview

↓

aligning yourself to the thought process of the interviewer.



* You already knows about the system

→ explain your understanding to the interviewer so that both are aligned in same thought process.

* You don't know about the system

→ ask the interviewer to give you an overview of the project/product

Q1. What exactly do you want me to design?

Ans → Entities or, E2E real world s/w design

Q2. Do I have to persist data?

Ans → DB or in-memory

↓

Schema Design?

Q3 How does the user interact with the system?

Ans ⇒ REST API

Command line

Hard Code

Step 1 : Gather and clarify requirements :-

→ Suggest ideas with rationale.

→ 5-8 core features.

→ Try to visualize as a user and suggest ideas

→ Jot down points

→ For every feature, think about edge cases and future scope for changes.

⇒ Requirement gathering for Pen :-

i) Any physical entity that can write is a pen

ii) Supporting pen ⇒ Ball Pen., Gel Pen and Fountain Pen.

iii) Gel and Ball pen will have refill and ink out.
Refills will contain ink

iv) Diff inks will have diff colors

v) For every pen we will have price, type, name, brand, color

vi) Some pen will have nibs

vii) Pen not having nibs, will have refills and refills may contain nibs

viii) Nibs will have diff sizes

ix) Pens can be closed by click, refill or cap.

x) Pen can write

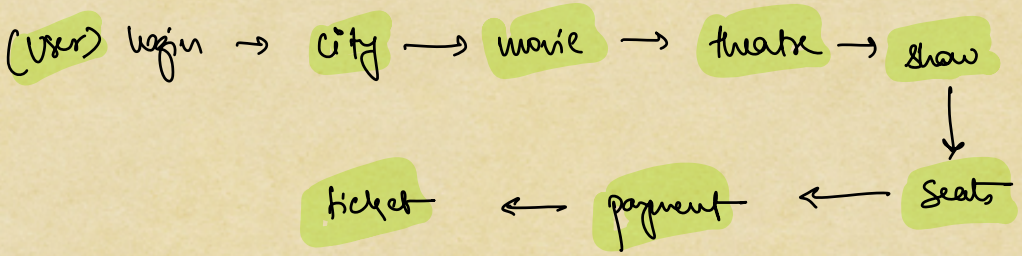
xi) Pen can open/close

xii) Pen can be refilled.

Step2 :- Class diagram

We need to know about entities.

i) Visualize the user journey :-



ii) Nouns from requirements

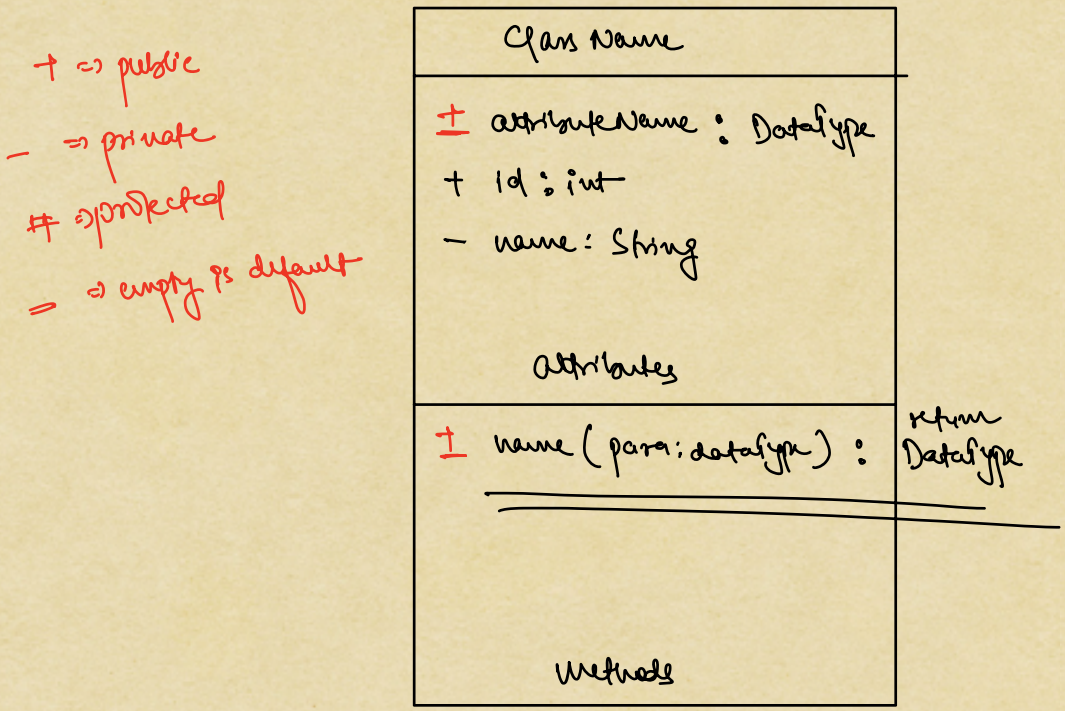
Entity → store some information

↓

real world Object

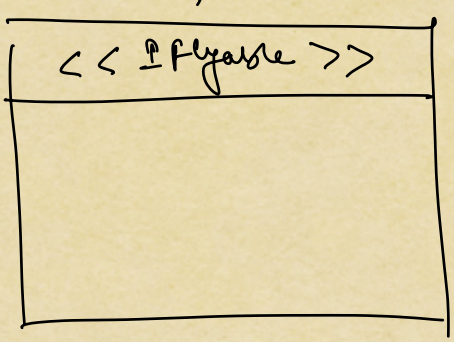
→ Pen class diagram:

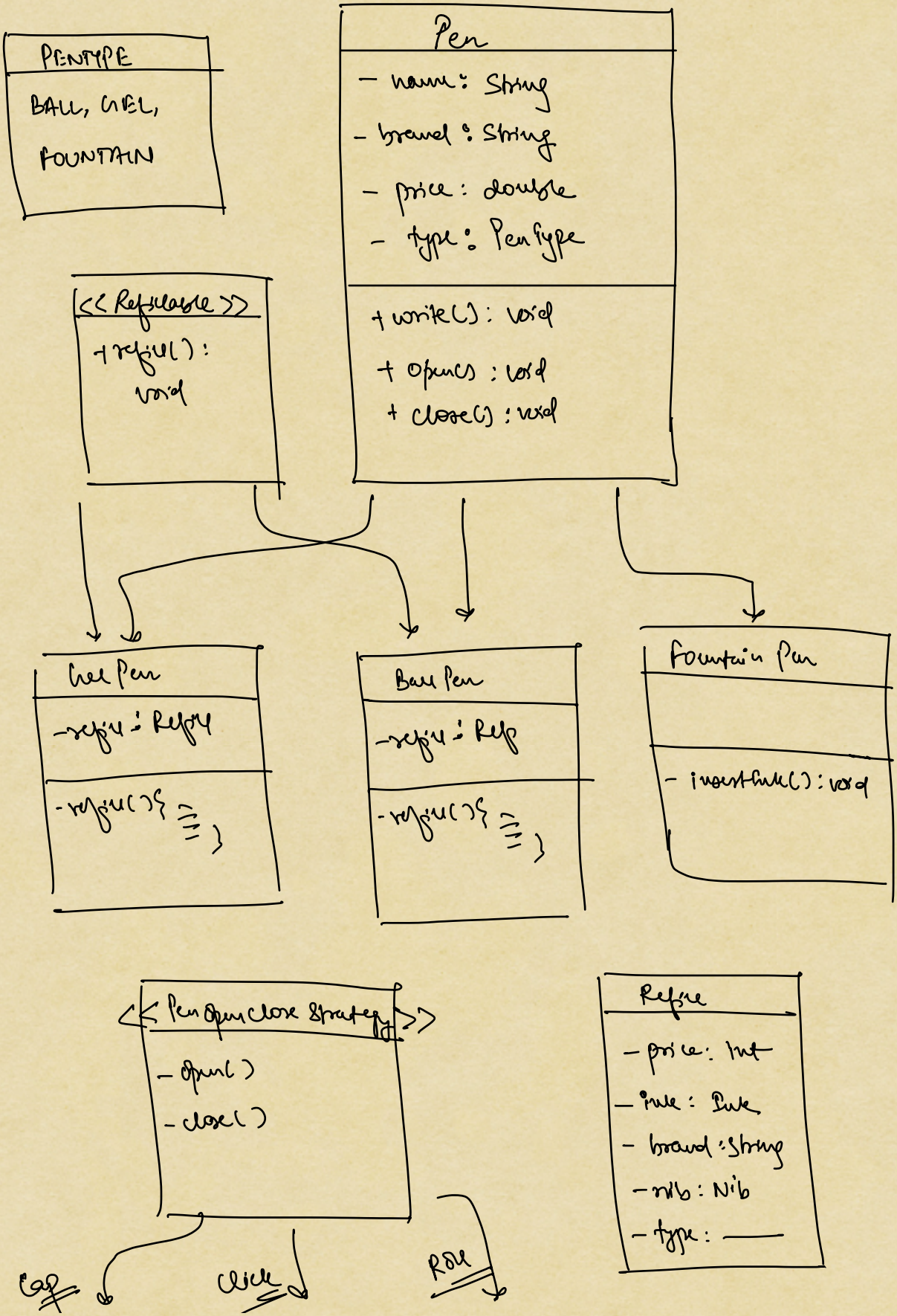
: Class diagram

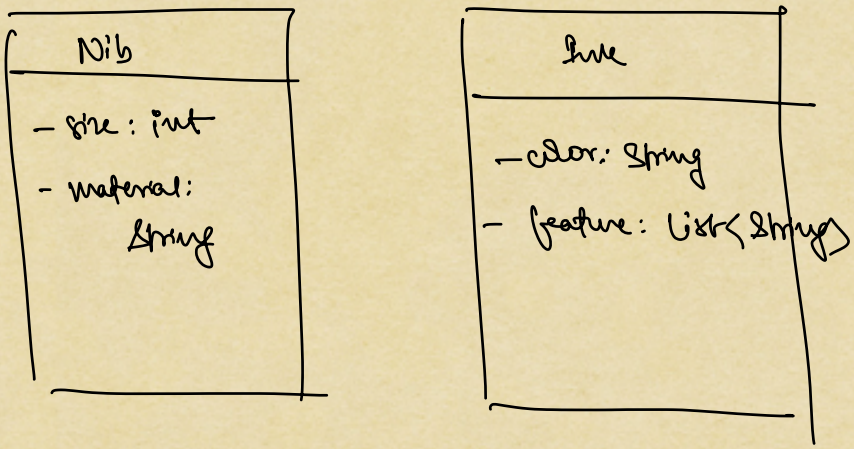


Static => italics

abstract => interface name / class name => italics




PENTYPE
BALL, GEL,
FOUNTAIN
Pen
- name: String
- brand: String
- price: double
- type: PenType
+ write(): void
+ open(): void
+ close(): void
<<Refillable>>
+ refill(): void
Gel Pen
- refill: Refill
- refill(){ }
Ball Pen
- refill: Refill
- refill(){ }
Fountain Pen
- insertInk(): void
<<PenOpenCloseStrategy>>
- open()
- close()
Cap
Click
Rotate
Refill
- price: int
- ink: Ink
- brand: String
- nib: Nib
- type: ____


Nib
- size : int
- material :
String
Shoe
- color : String
- feature : List<String>