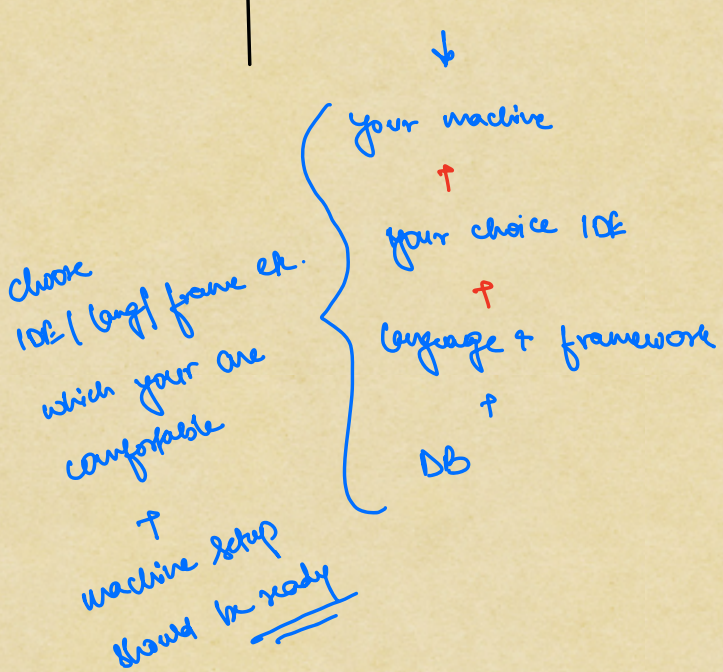
1) LLD interviews
2) Design a pen

⇒ Types of LLD interviews:-

THEORETICAL	DESIGN	MACHINE CODING
⇒ Old tech / service based companies ex ⇒ Oracle, WITCH, TP, Morganstanley	⇒ Big tech ⇒ Amazon, Walmart, Atlassian, Google etc.	⇒ new age tech + startups CRED, Sedar, Flipkart, Zepto, Swiggy
⇒ Text knowledge	⇒ problem solving	⇒ problem solving + implement
⇒ Theory ques, OOP, Java, syntax etc.	⇒ 1 Ques ⇒ Design	⇒ 1 PRD ↓ contains all requirements
⇒ 45 mins	⇒ class diagram, schema design, design pattern, how & why	⇒ design the system & code ↓ E2E running code
⇒ no coding (max to max syntax check)	⇒ no to very minm coding ⇒ pseudo code	⇒ 120 mins ↓ 100 mins (design + code)
	⇒ 60 mins	

PRD ⇒ Product Requirement Doc

DESIGN 60 ⇒ 50	MACHINE CODING 120 ⇒ 100
⇒ Given a single line problem statement	⇒ Detailed document containing all requirements
• Gathering detailed requirements (10)	• Clarify requirements (20)
• Class diagram (15)	• Class diagrams
• Schema design (10)	• Schema design
• Write some code to showcase the classes & their relationship (10)	• Complete end to end working code to satisfy all requirements (60 min)
• follow ups	• follow ups



⇒ Scaler

- ⇒ 1 live problem statement
- ⇒ gathering reqn
- ⇒ clarifying reqn
- ⇒ class diagram
- ⇒ schema design
- ⇒ E2E coding

↳ 4 projects

- ⇒ TicTacToe ⇒ Java
- ⇒ Parkinglot
- ⇒ BookMyShow ⇒ Java + SpringBoot + SQL
- ⇒ Splitwise

⇒ DESIGN A PEN:-

Step ⇒ 0 ⇒ Get an overview

⇒ align your mind to the direction of the interviewer

You know about the system

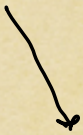
you dont know about the system

⇒ Share your ideas and thoughts

⇒ simply ask for a

with the interviewer

detailed description of what needs to be built



⇒ what exactly needs to be developed

⇒ do I need to persist the data?
(In-memory or DB)

⇒ how will the user's interact?
(command line, hardcoded, REST APIs)

Step 1 **Gather and clarify Requirements**

- Suggest ideas with rationale and ask if we should support that feature
- 3-5 core features
- Try to visualise and drive the flow
- features that we are supporting
  - → future scenarios
  - → edge cases etc.

⇒ PEN

i) Any physical entity that has write behaviour is a pen

ii) Types of pen ⇒ ball, gel, fountain

iii) Refillable ⇒ do they have refills

- Yes ⟶ No ⇒ Fountain Pen
- No ⇒ No ⇒ Use N throw Pen
- Yes ⇒ Yes ⇒ Ball / Gel pens

iv) Pen model name & brand name

v) Pen attributes ⇒ price & dimensions & color

vi) Pen can be closed via cap, click or roll

vii) Nib radius can be diff.

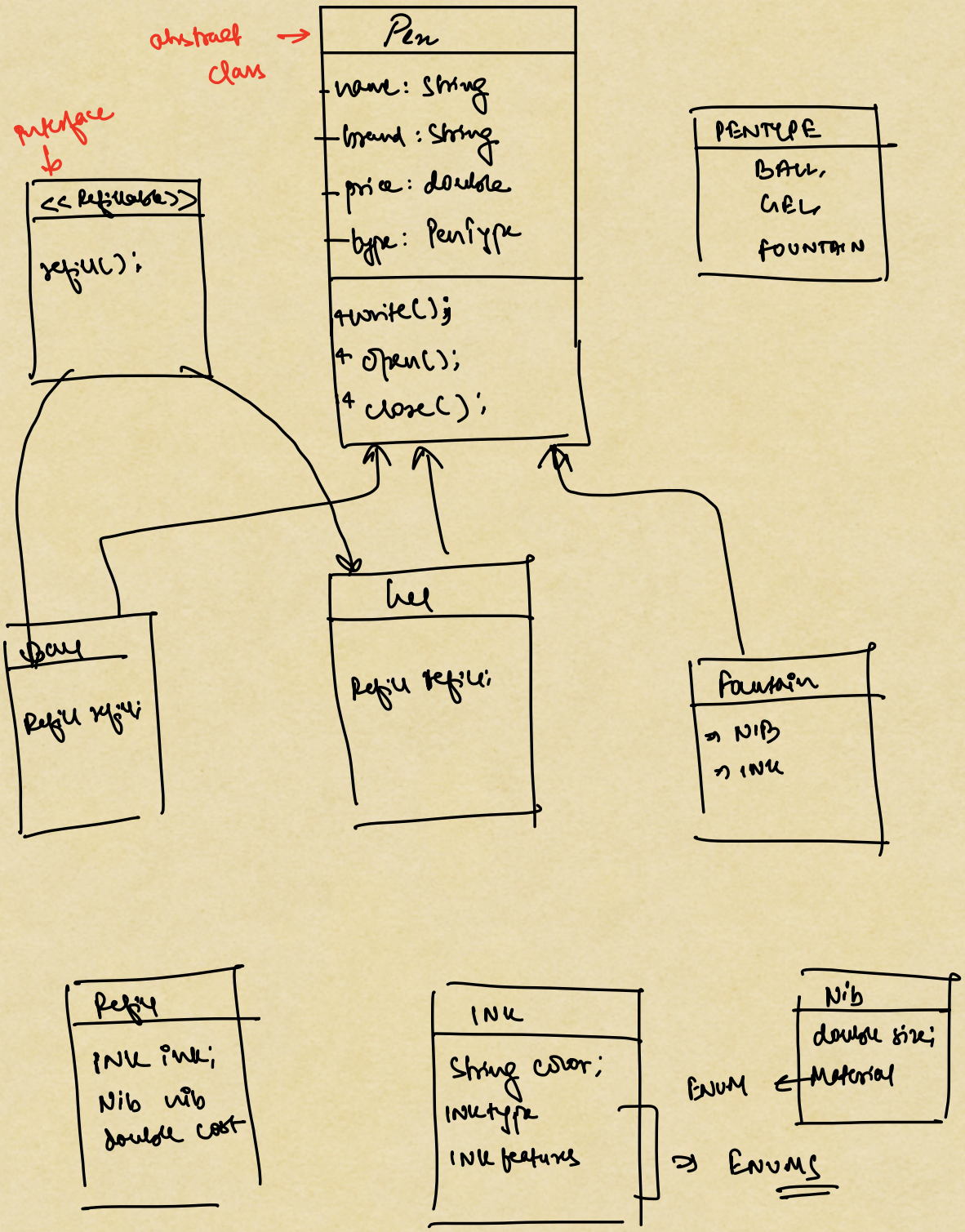
viii) All pens dont have nibs (pen ⇒ attached nibs
⇒ refill with nibs)

[
- x) Write
- x) Refill
- x) Close / Open
]

Step2 ⇒ Class diagram

i) By visualization of requirements (User journey)

ii) finding nouns in the requirements


abstract class
Pen
- name: String
- brand: String
- price: double
- type: PenType
+ write();
+ open();
+ close();
interface
<<Refillable>>
refill();
PENTYPE
BALL,
GEL,
FOUNTAIN
Ball
Refill refill;
Gel
Refill refill;
Fountain
→ NIB
→ INK
Refill
INK ink;
Nib nib
double cost
INK
String color;
INKtype
INK features
ENUM
Nib
double size;
Material
→ ENUMS

#HW

Create UML diagram ⇒ PEN

↓

(name)

SOLID ⇒ 1 & 2

+

UML

⇒ Revision

↓

Look at Pen class diagram ⇒ Ask Qns.

↓

UML diagram ⇒ Pen classes

Tomorrow

EOD