

## Today's Agenda :-

- 1) Types of LLD Interviews
- 2) How to approach LLD Interviews
- 3) Design a Pen.

No  
LLD  
Round { Google }  
          { Meta }

## Types of LLD Interviews :-

Theoretical  
Design  
Machine Coding }

Theoretical :- (45 min)

Concepts

Q & A

explanations

→ Questions based on OOPs, design Objective

patterns & principles & language concepts,  
etc.

→ Knowledge based on LLD

→ Older tech companies

Goldman Sachs, Wells Fargo.

(Oracle, JPMorgan, BNY Mellon, etc)

## Design Round (Most common) (45 min)

Problem statement to design some system

→ Case study based discussions.

→ Amazon | Microsoft | Walmart | Paypal | Uber | Atlassian | etc.

→ Basic API Design + Some basic code might be expect.

↳ 45 min to 1 hr

Machine Coding ::

(2 hr - 3 hr)

Case Study

Flipkart

PhonePe

Meesha

Myntra

Swiggy

1<sup>st</sup> round

as

MC.

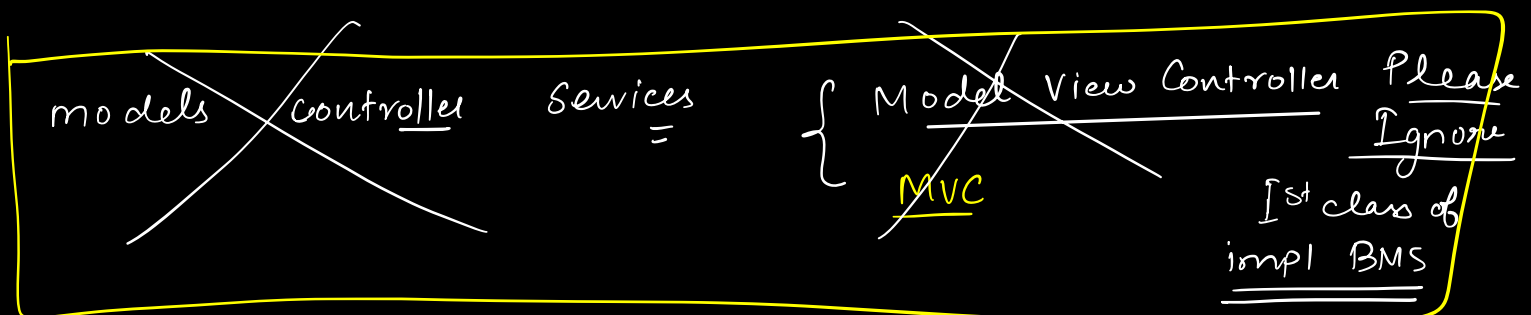
Design Snake  
Ladder game

UIX

↳ Design + Working Code

↳ Write end to end working code for the given set of requirements.

↳ Almost every tech startup.



# Design

1 line problem statement

→ Case study

- 1) Gather Requirements
- 2) Class Diagram ✓
- 3) Schema Design ✓
- 4) Basic API Design & code structure
- 5) Follow ups

Amazon 1 hour

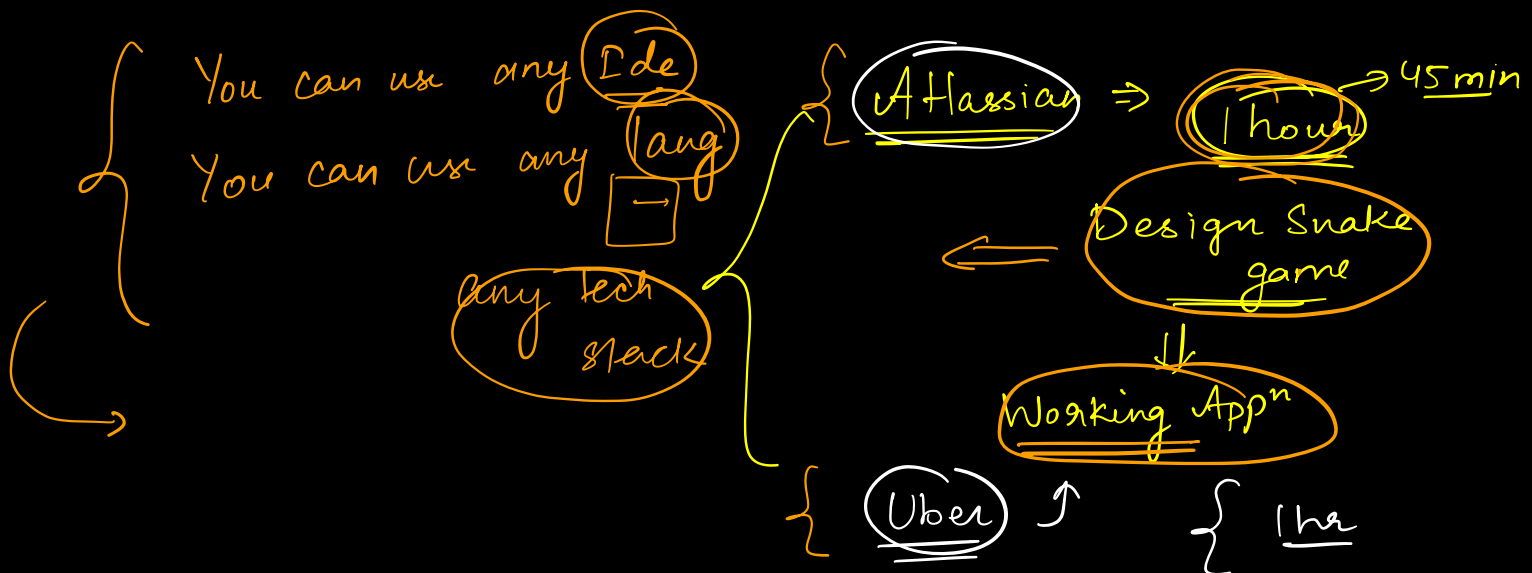
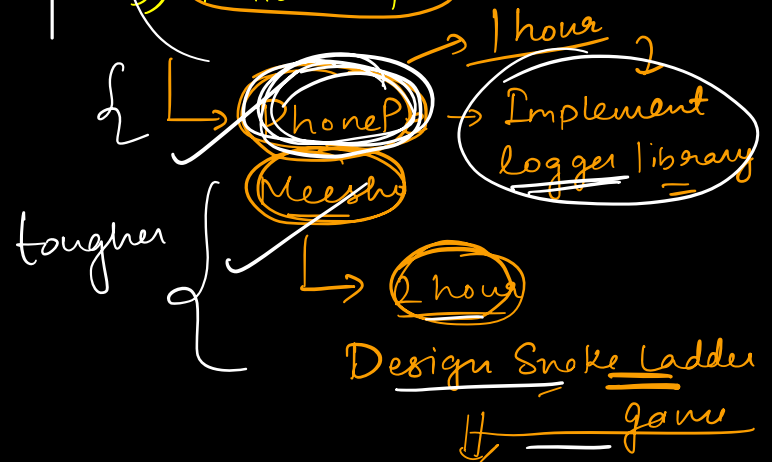
# Machine Coding

2-3 hrs

3hr

Design + Working Code

- 1) Gather Requirements
- 2) Class Diagram
- 3) Schema Design
- 4) Write end to end working code
- 5) Follow ups



At Scale,

1 line  
problem statement

OR

given a set of requirements

easier

Gather requirement yourself

you need  
what to  
focus on

1 line problem statement ✓

Gather requirement · (Gather what to focus on)

↳ v. Imp (Core MVP)

Minimal  
Viable  
Product

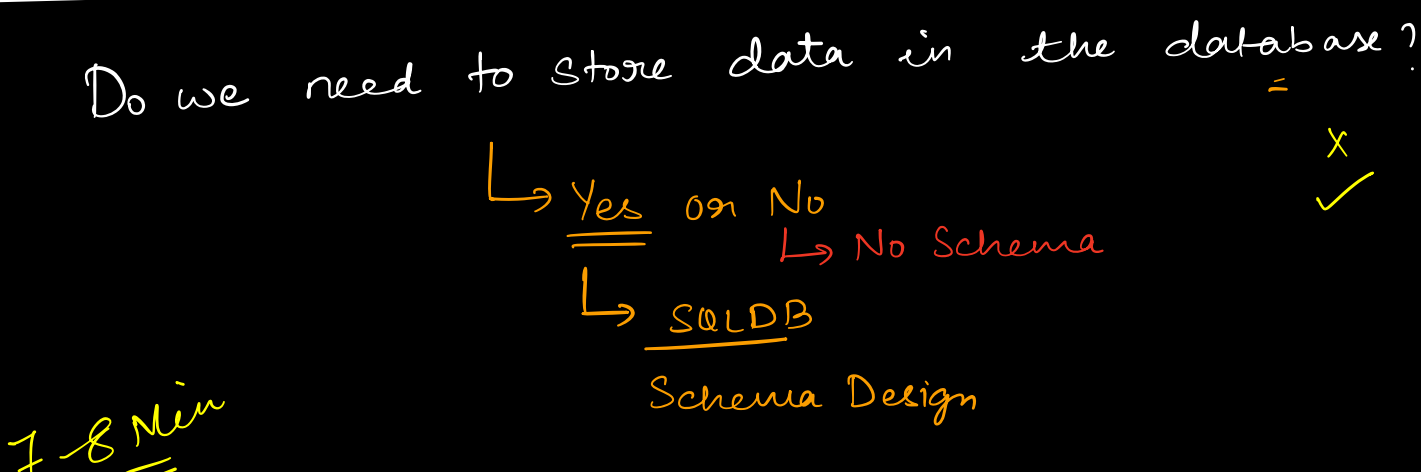
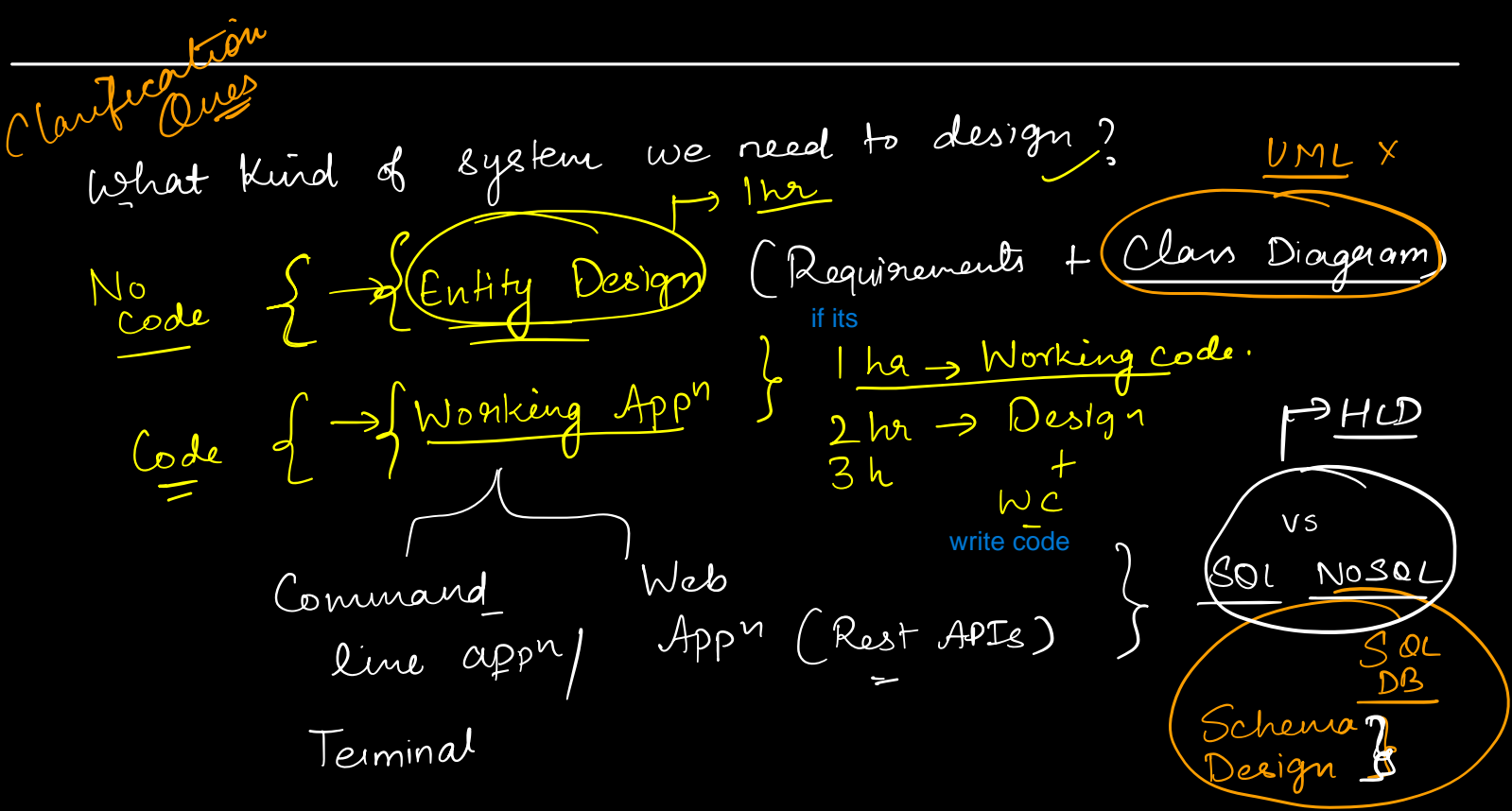
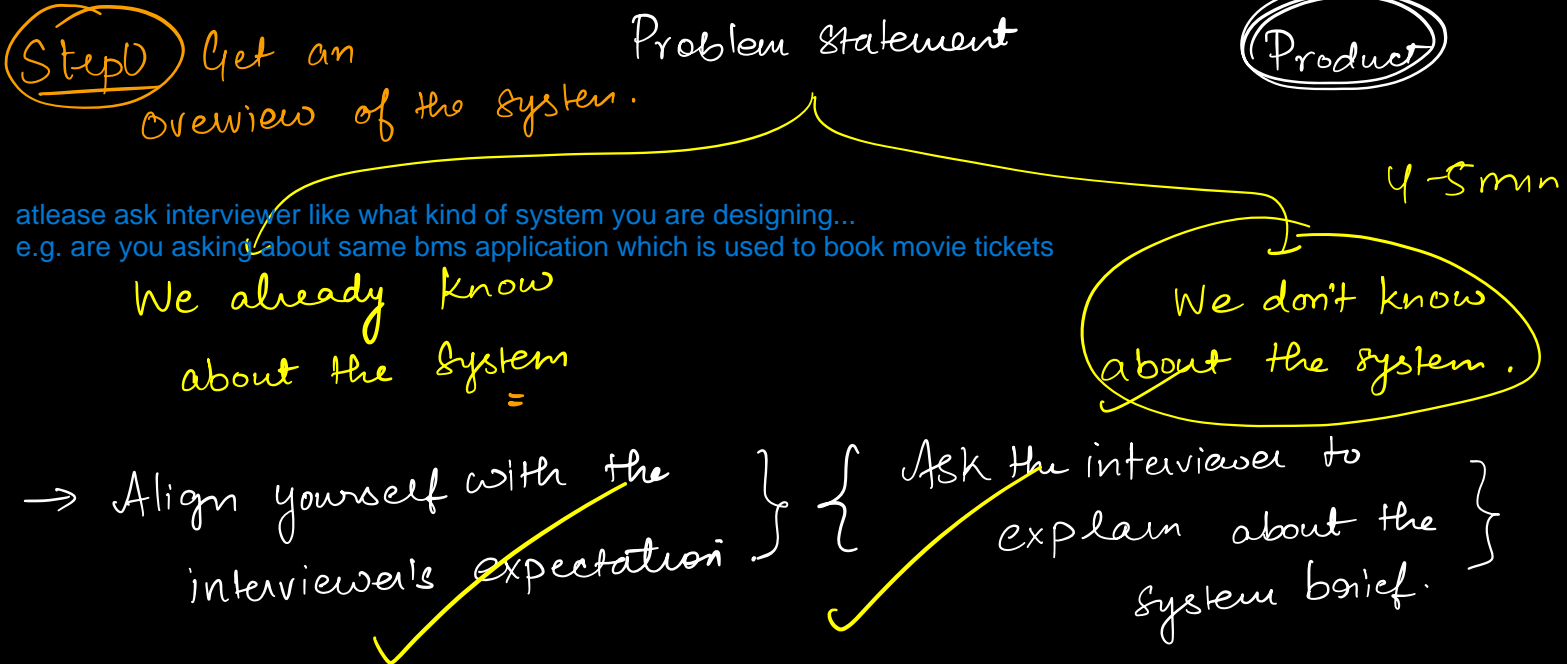
VO

Class Diagram ✓

Schema Design ✓ (when you're persisting data in the db)

5 min

First 2-3 Min ✓



## Step 1 Gather Requirements

↳ Try to collect 5-8 core features of the system.

↳ Try to find out edge cases

first 10 min ✓

## Design

### Step 2

Class Diagram

↳ Classes | Interfaces

↳ Rel<sup>n</sup> b/w classes

↳ Design patterns.

+  
25 min

+

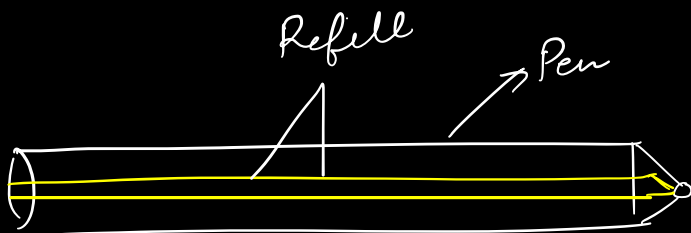
### Step 3

Schema Design.

+  
follow ups

10 Min

↳ 45 min



Visualize the  
Physical  
Structure.

## Design a Pen :- (10 Min upper bound)

### Requirements:-

- 1) Any physical entity that can write is a Pen.
- 2) Different types of pens :- Ball Pen, Gel Pen, Fountain Pen, Pencil, etc.
- 3) Ball & Gel Pen will have a refill, fountain pen doesn't have a refill.
- 4) Refill will have Ink. Fountain pen directly will have ink.
- 5) Ink can be of diff colors.
- 6) Refill will have a nib. Nib can be of different radius
- 7) Fountain pen will directly have a nib.
- 8) Some pens can be refilled & some pens can't.
- 9) Pens can be opened via cap or via button

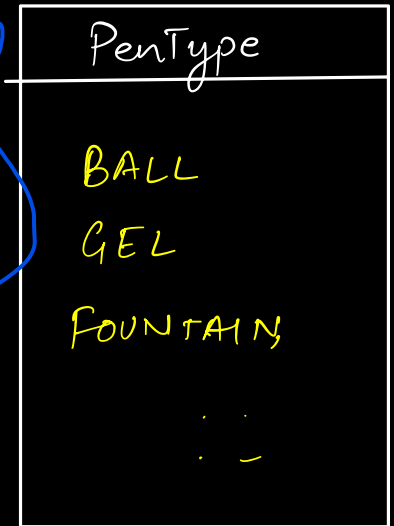
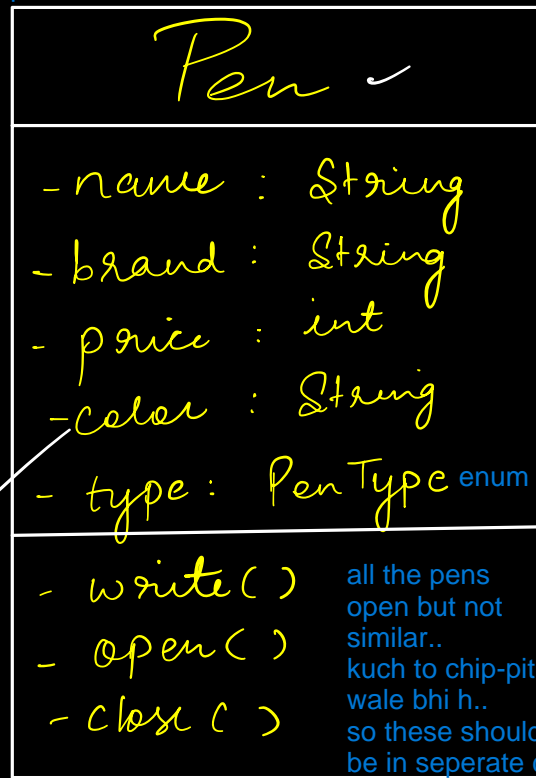


whenever we feel like an entity is not complete itself better to create abstract class instead of interface.. but whenever we feel these are behaviours only and these are not gonna change better to go with interface

all the things in this class are not concrete so

: abstract / Interface

Behaviours... and refill can not behaviour.. because all pens do not have refills  
e.g. all birds can not fly

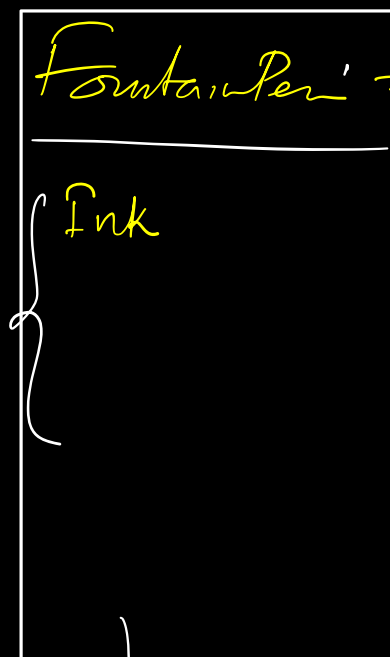
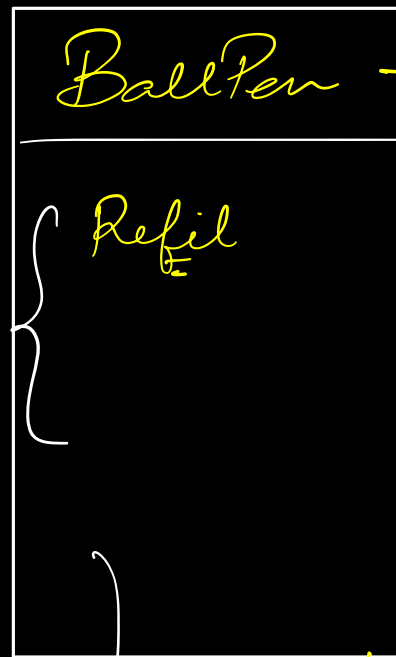
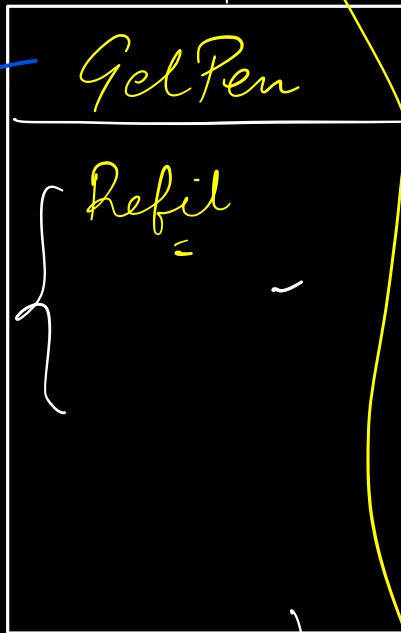


all the pens open but not similar..  
kuch to chip-pit wale bhi h..  
so these should be in seperate classes.. as well as write and close

<<Refillable>>

⇒ Categoricalisation of Pens

implement  
Italic so.. abstract classes

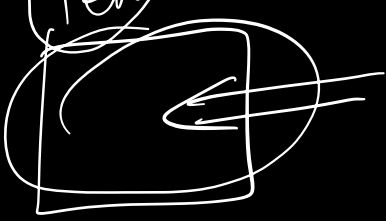


Different companies so different specifications

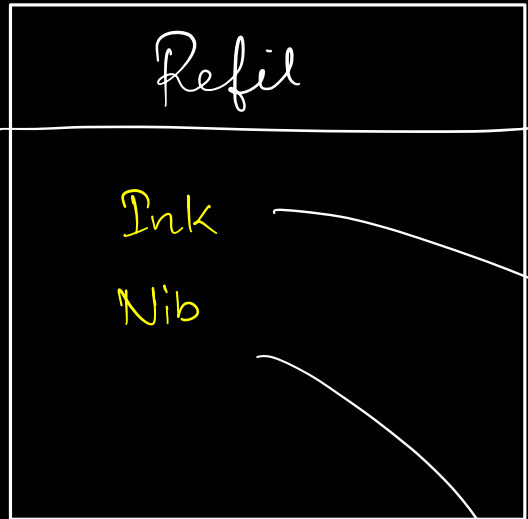
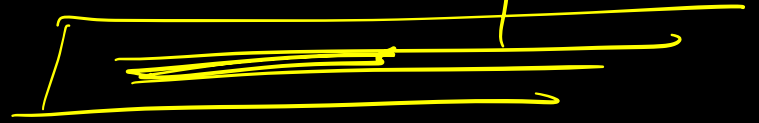
These will be concrete classes because those specification will be concrete and same

Concrete classes

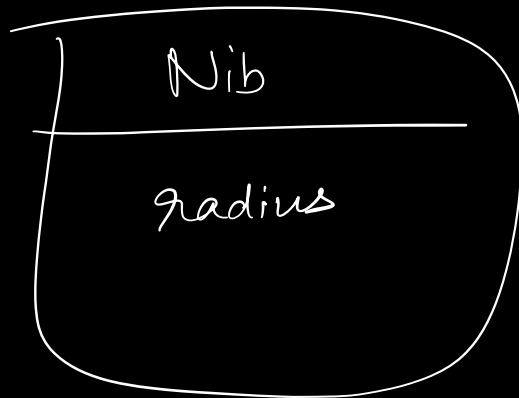
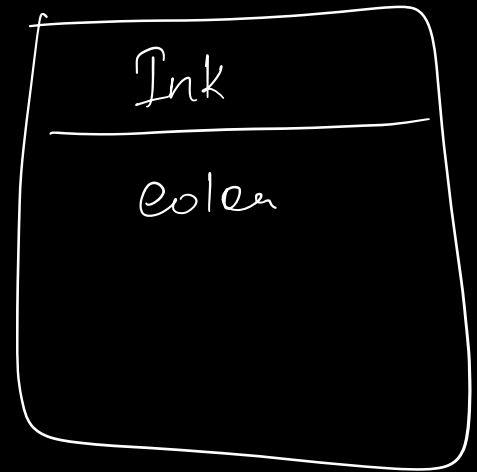
Pen ✓

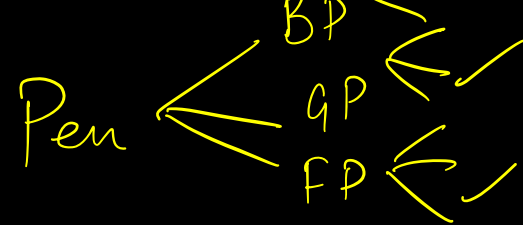


Refil



refill





Nib

Refil

Ink

Categorising

list of Gel Pens

list < GelPen >

⇓

only can put

printPrices OF All Gel Pens (list < GelPen >)

{ attr of Gel Pen ✓ }

