LLD - Design a pen

① Design discussions

↳ Abstract Problem statement

② Machine Coding

↳ Implementation

→ APIs

→ Flipkart

① Requirements

→ Real world

↳ Pen

↳ Book

→ Designing life lac lor

Expectations

① Structure the problem

② Clarification

— Remove all doubts

① ①

— Class diagram

— relationships

— entities

- cardinality

- use case diagram

---

③ Implementation

① Structurace

② SOLID

③ Design P.Hens

---

① Requirements LLD

② Use case (Entities , use cases)

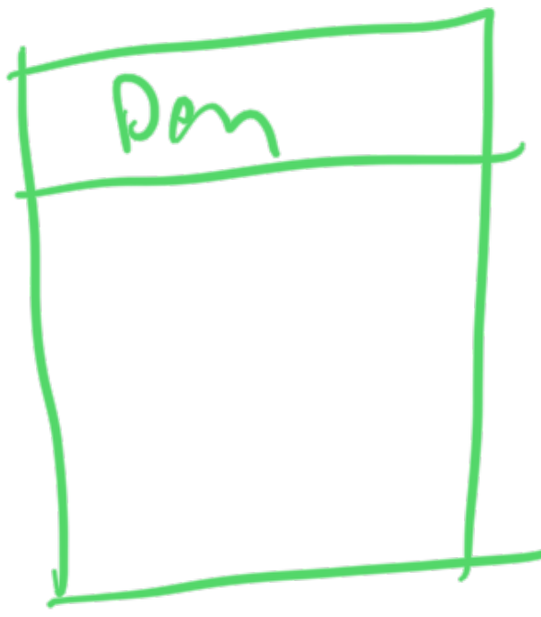
③ Class Diagram

Important — Functionalities

→ Core APIs

Pen → Google → SDE-1
→ Amazon

① Give an OOP design for a pen



→ Requirements (Functional)

↳ Current Requirements

↳ Future requirements

↳ (illegible)

↳ Behaviour

Questions

1. What are the types of a pen?
2. Can you play pen fighten?
3. Does a pen have a cup?

4. What is a pen!
5. Is it reusable?
6. Can you refill a pen!
7. Is it a digital pen?

---

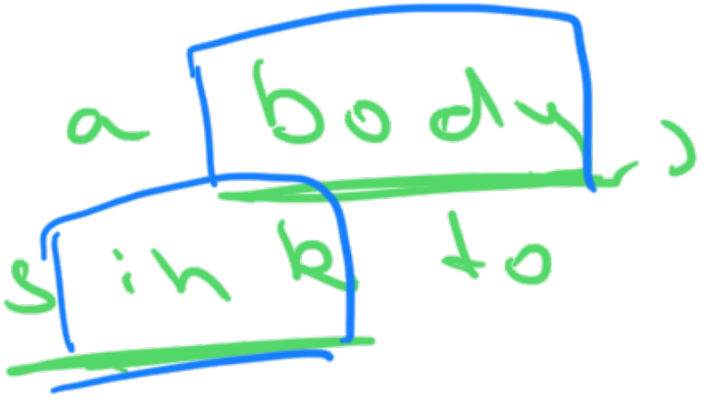
8. Where can we write?
   paper, Slate
9. Price of the pen

(10) [illegible]

---

(9) Design a pen with write functionality

(8) Anything with a body which contains ink to write is a pen



(7) → Ball point
→ [illegible] pen

→ Get pl...

Ball pen writes differently when comp. to a Gel pen

→ Maaben

→ Pencil

Refilling

Each pen should have a refill functionality

7. Ball pen refilling behaviour is different to gel pen refilling

8. Other pen can or cannot have different refilling behaviours

---

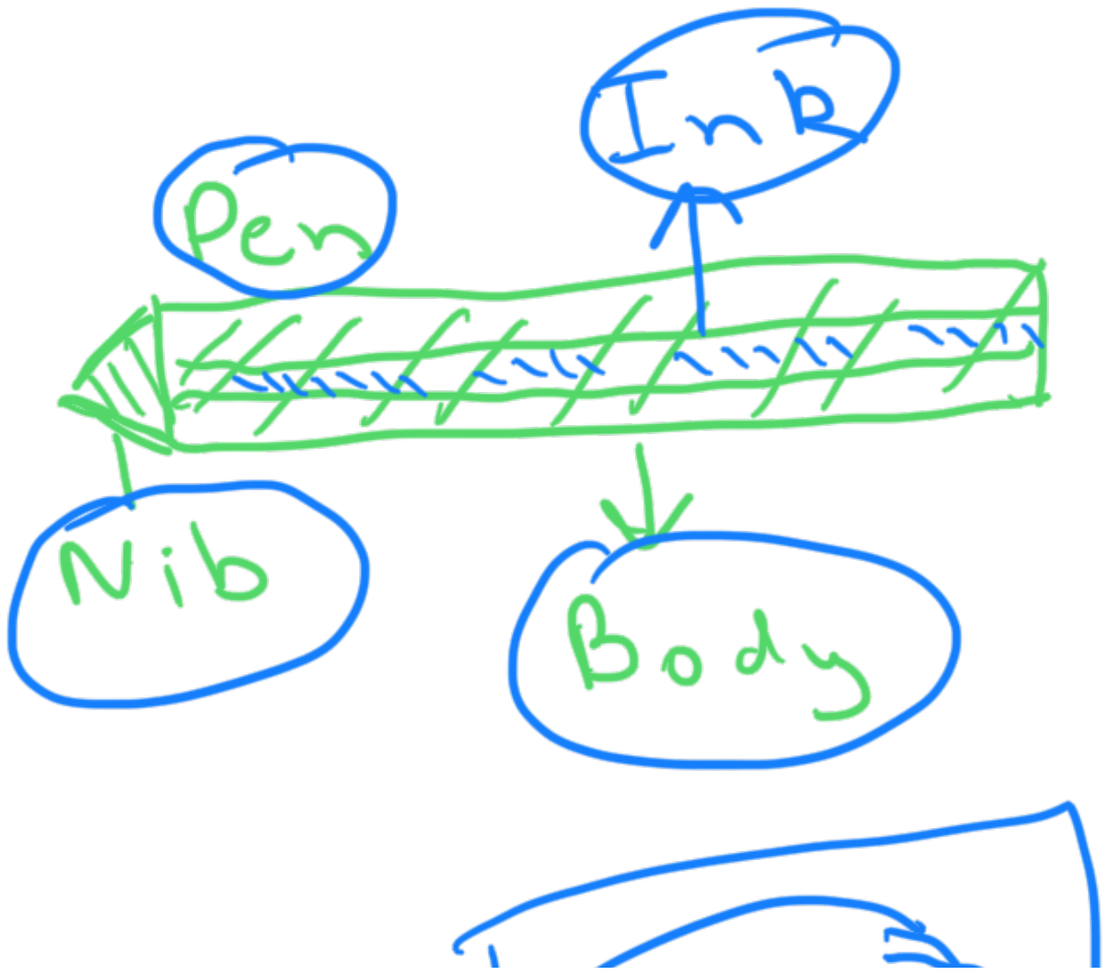
1. Entities → behaviours

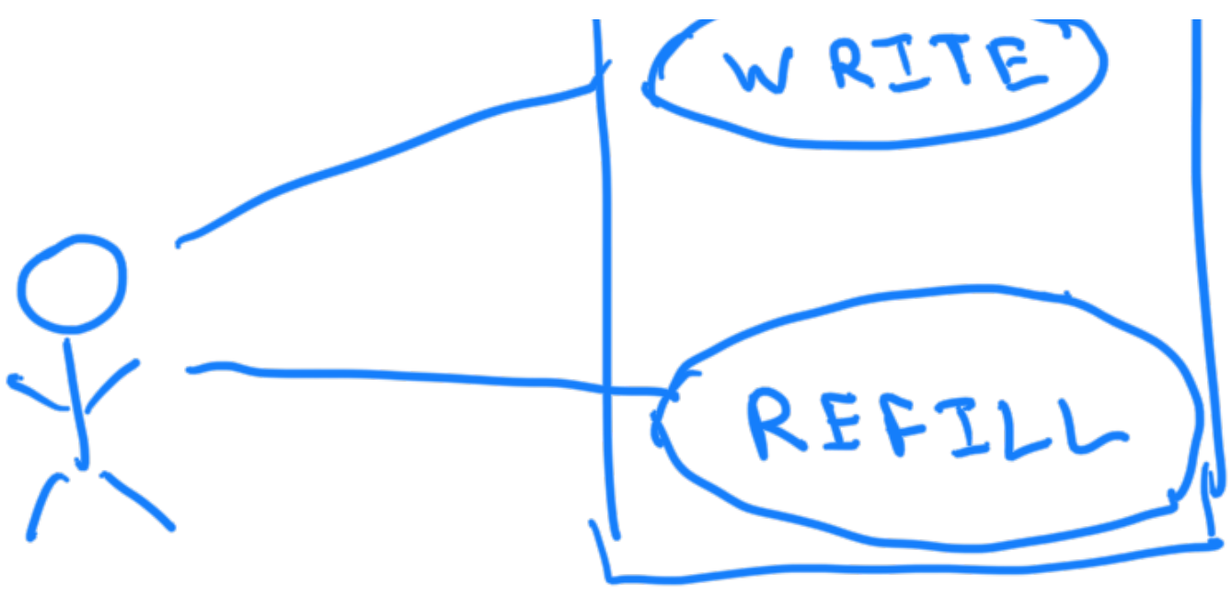
→ nouns

→ Visualisation

↳ use case

↳ data flow (DFD)





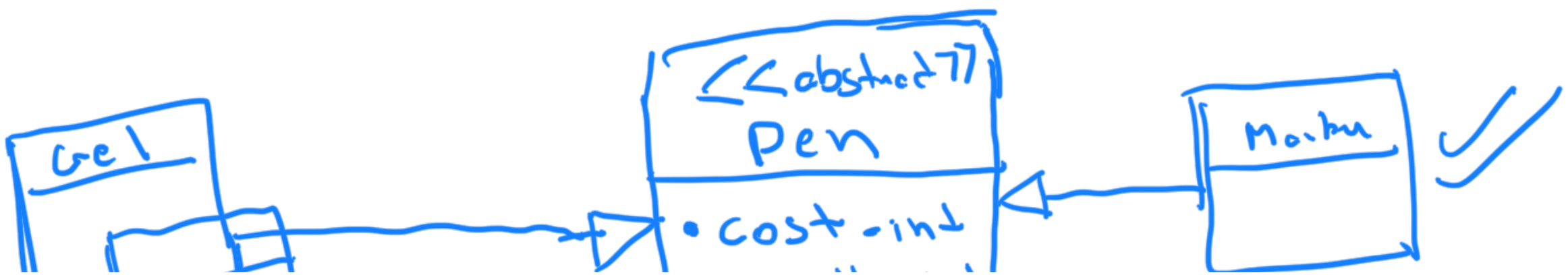
2/3

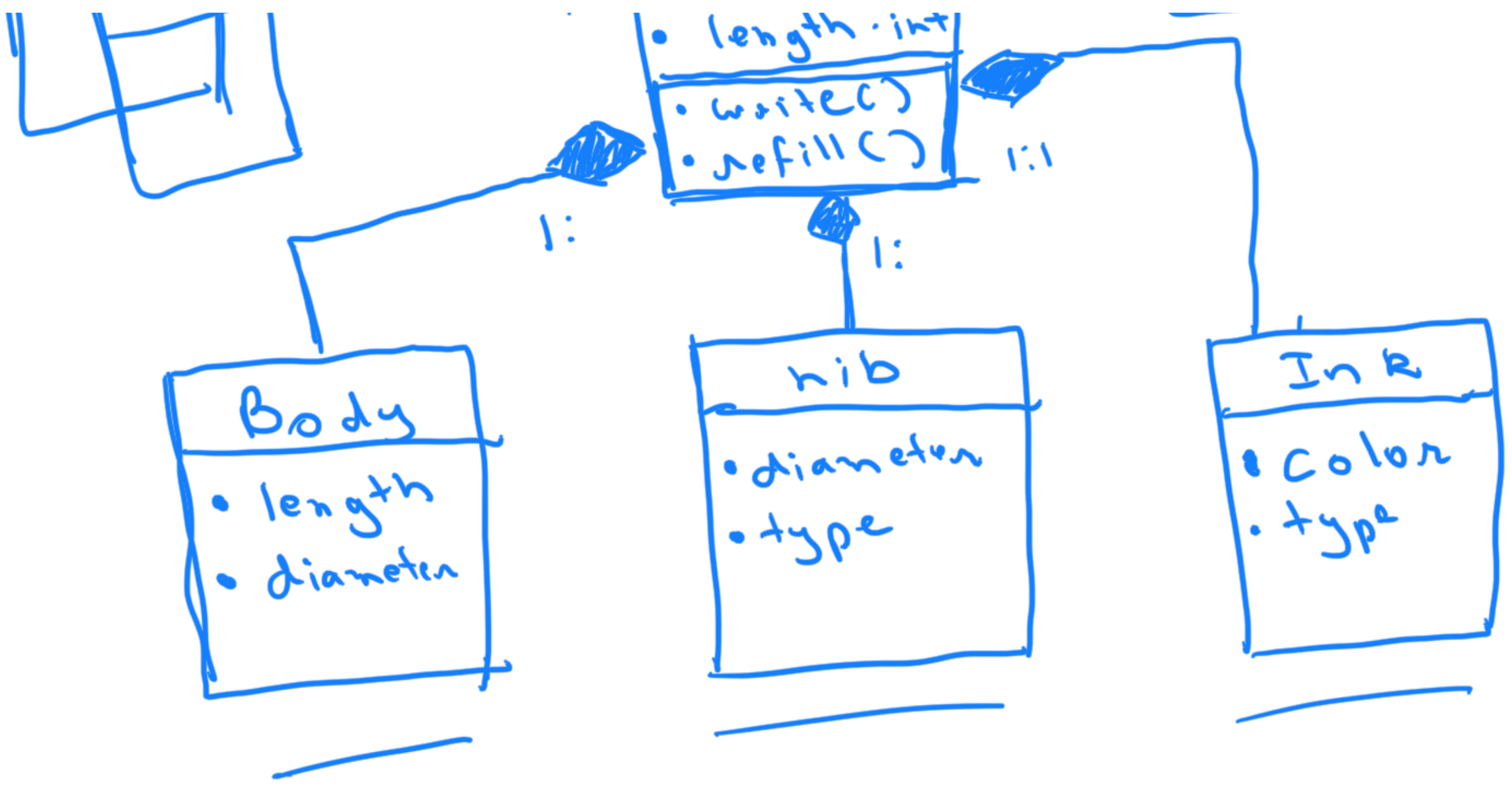
CLASS Diagram

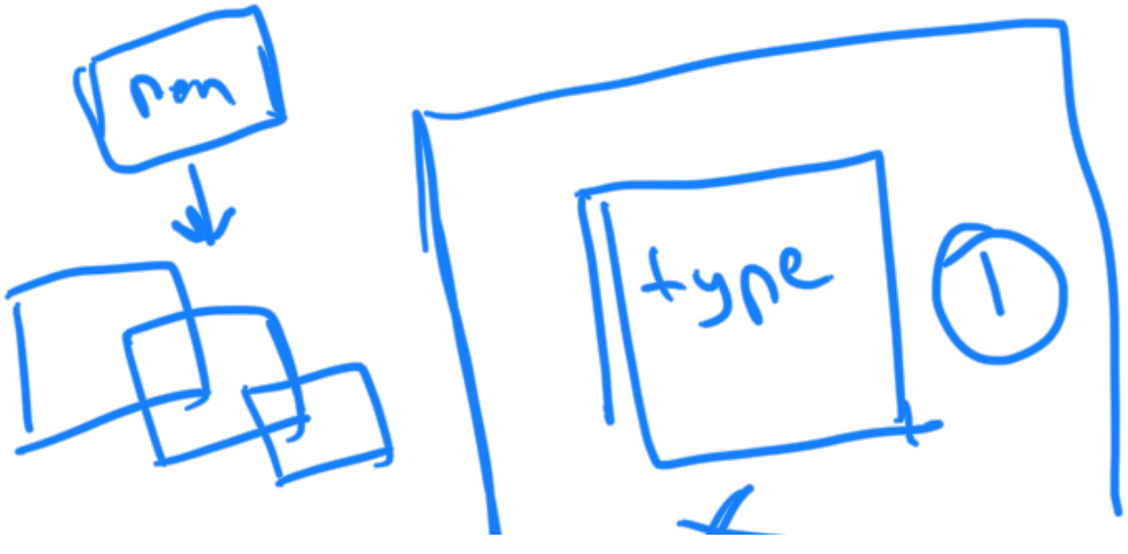
No assumptions

→ E-R

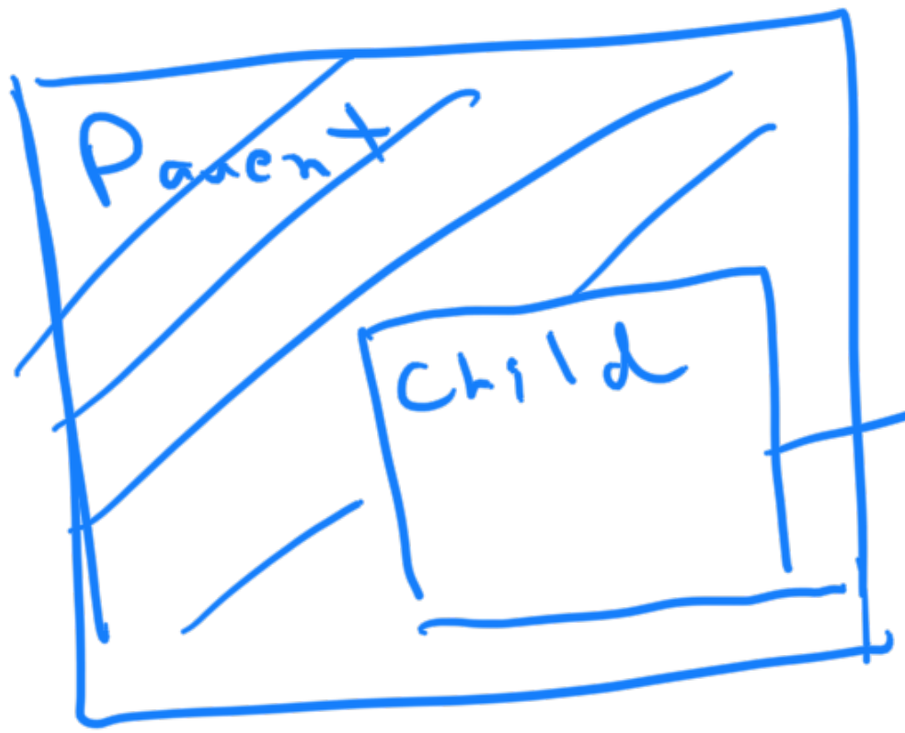
1. Relationship
2. Cardinality








```
if (pen == "rel")
    { ... } else
    { ... }
```



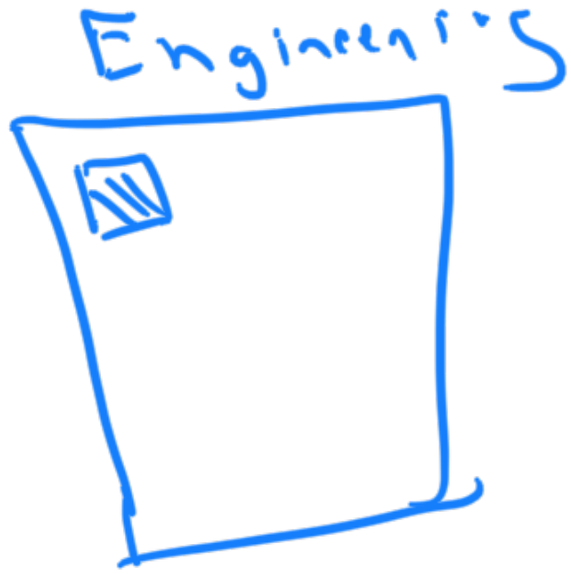
→ destroyed?

yes → composition

no → aggregation

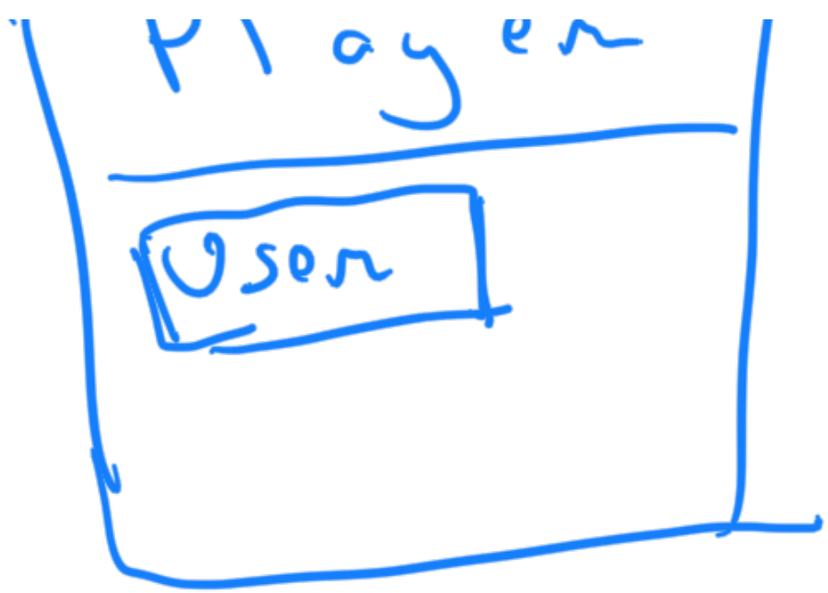
→ composition

Pen #2



٩-١٠

١٠: ١١



A

```
{
    GelPen ext.Pen
    write() {
        ...
    }
    ...
    nofill() {
        ...
    }
}
```

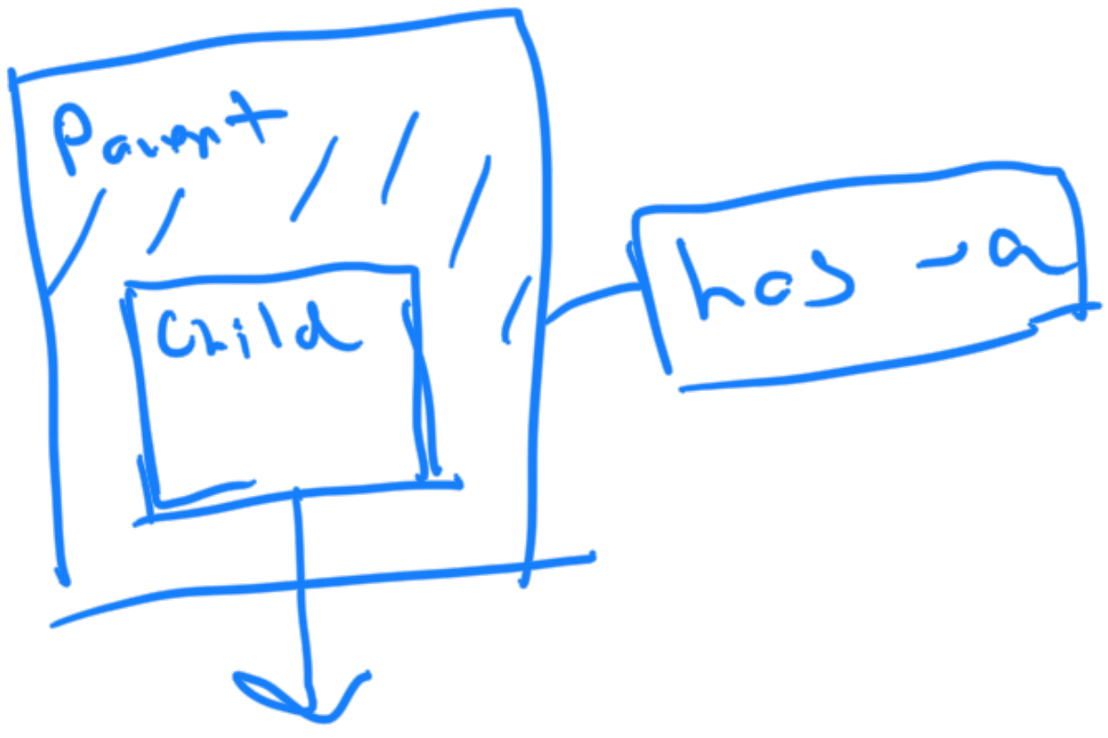
10:51

Association vs Inheritance

has-a

is

Composition vs Aggregation



if child is destroyed

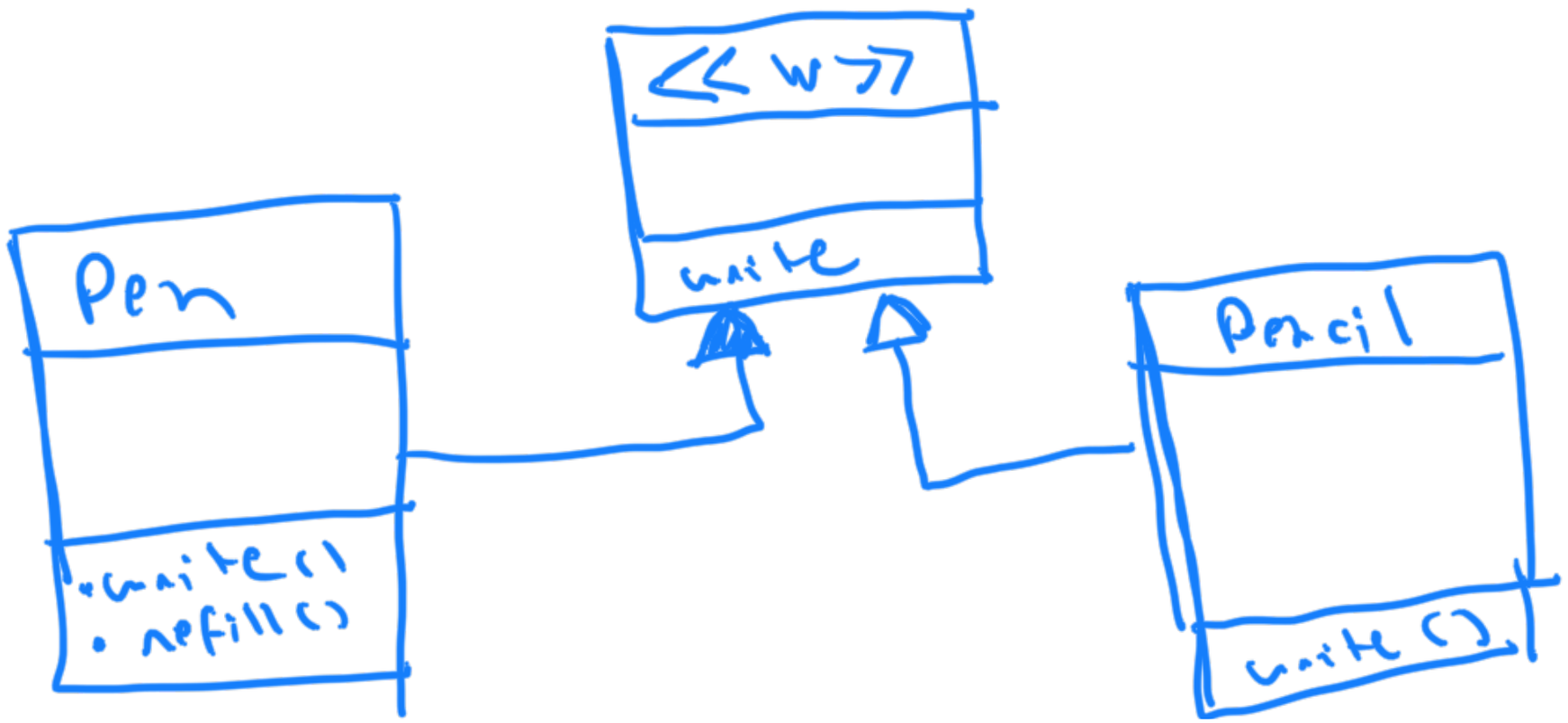
if

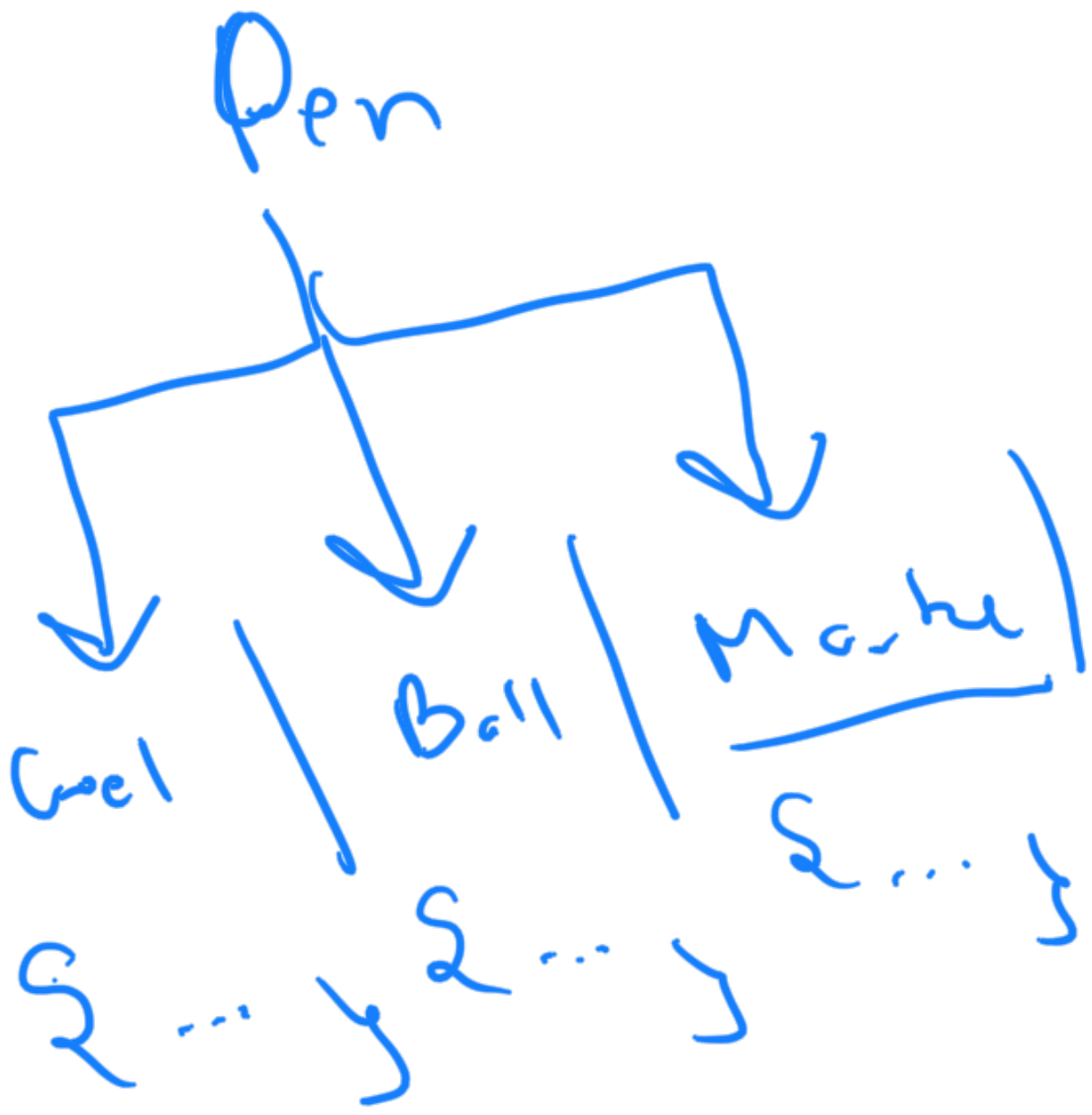
↳ composition

---

if child is not destroyed

↳ Aggregation



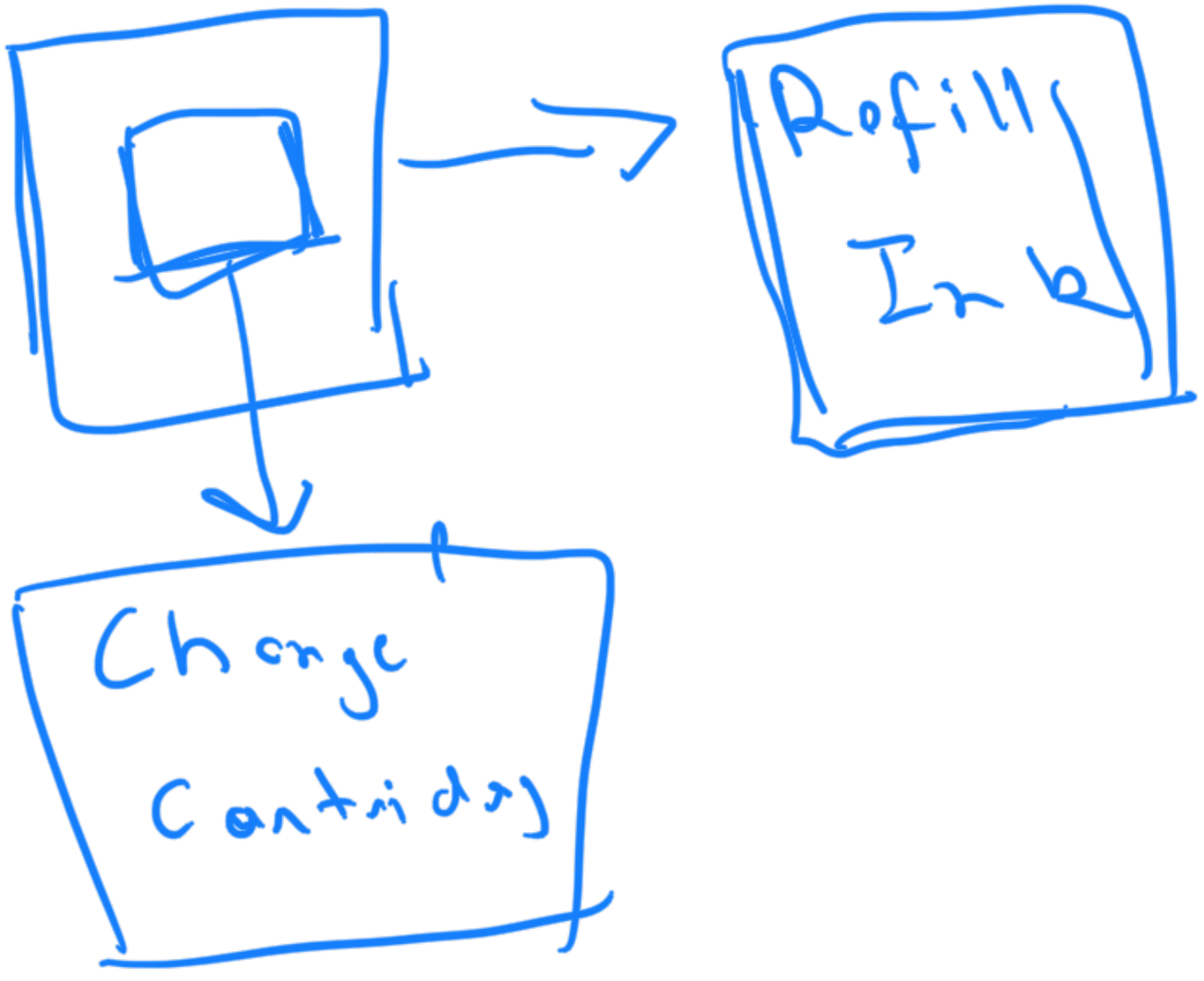


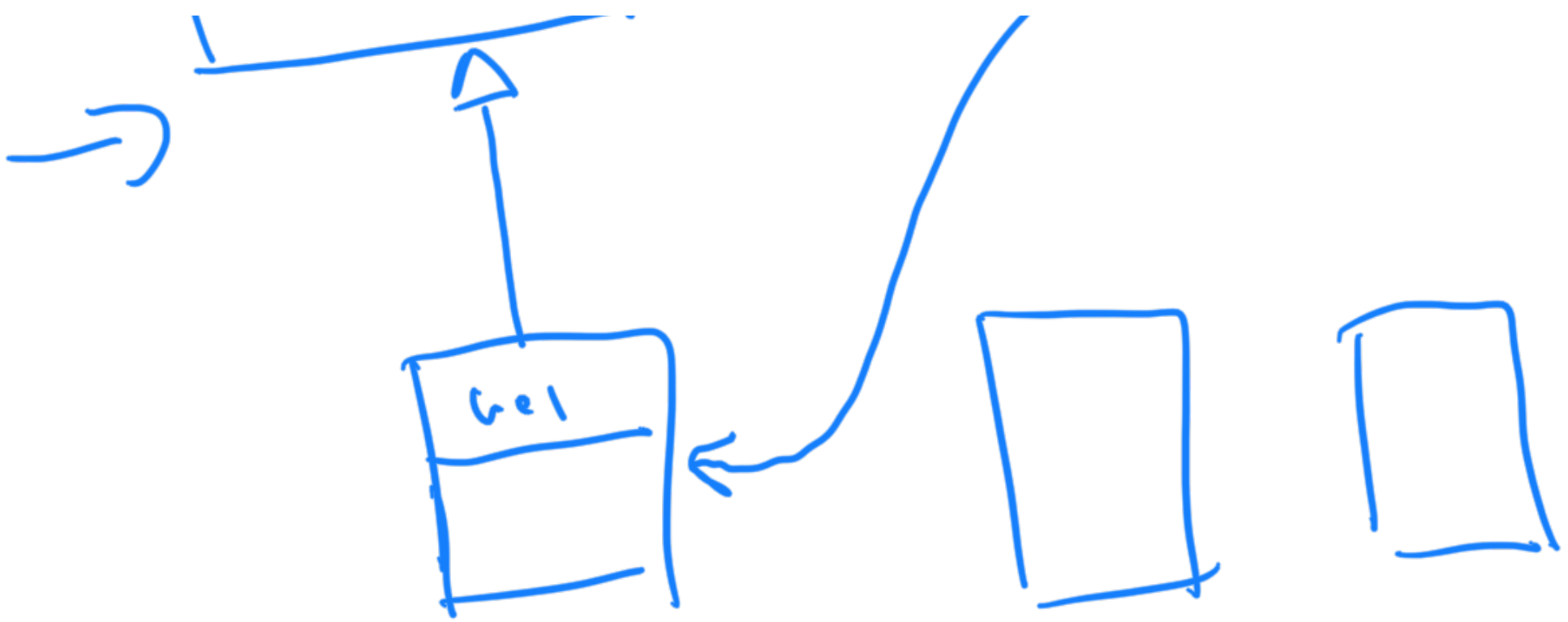
→ Create Interface [Refillable]

→

t=1

t=2





Strategy

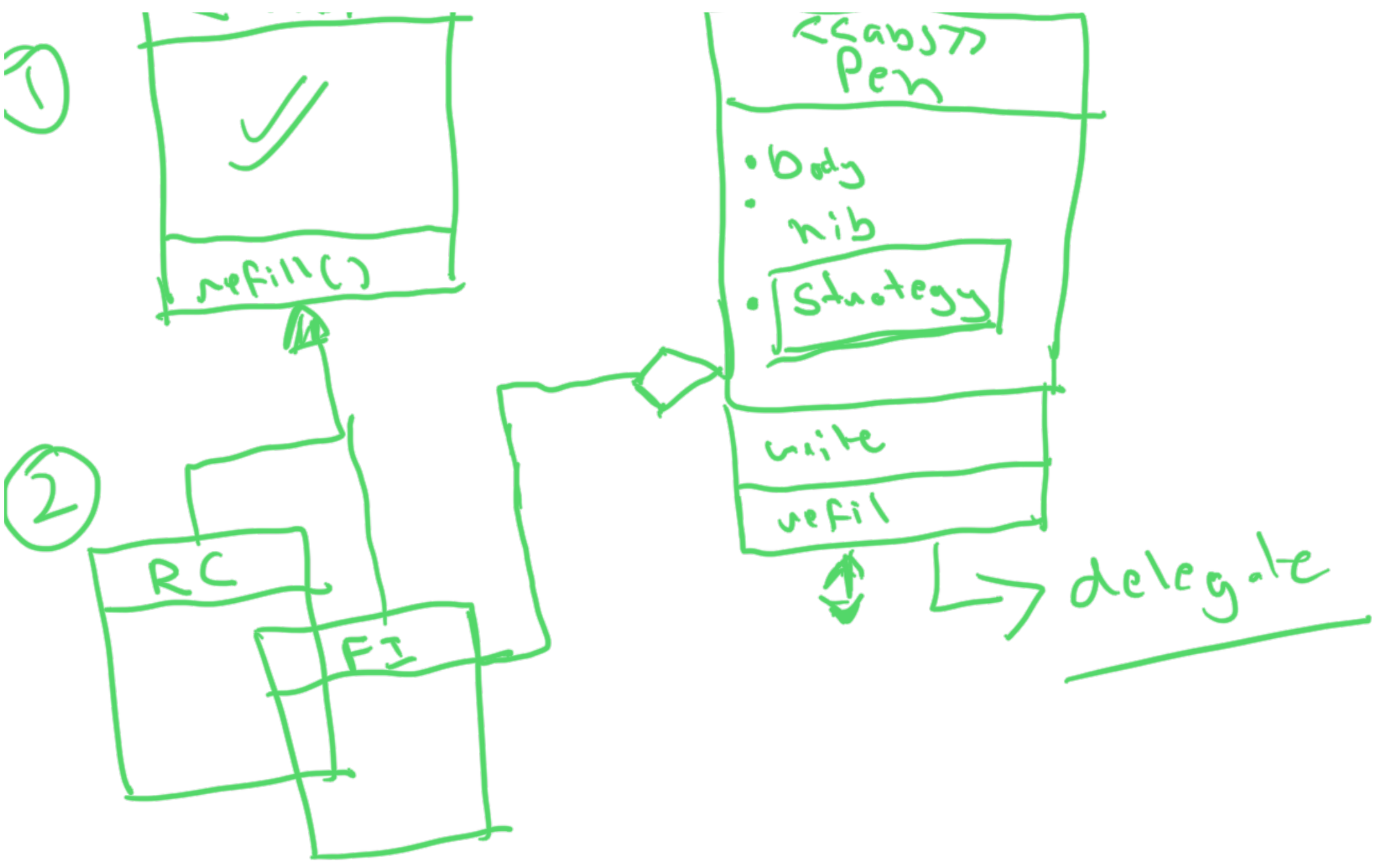
1. Interface (Strategy)
   - Refilling strategy

② Implement different types of strategy

→ Replace Refill()

→ Fill Ink()

③ Inject the strategy in the pen class


1
refill()
2
RC
FI
<<abstr
Pen
• body
• nib
• Strategy
write
refil
delegate

Code