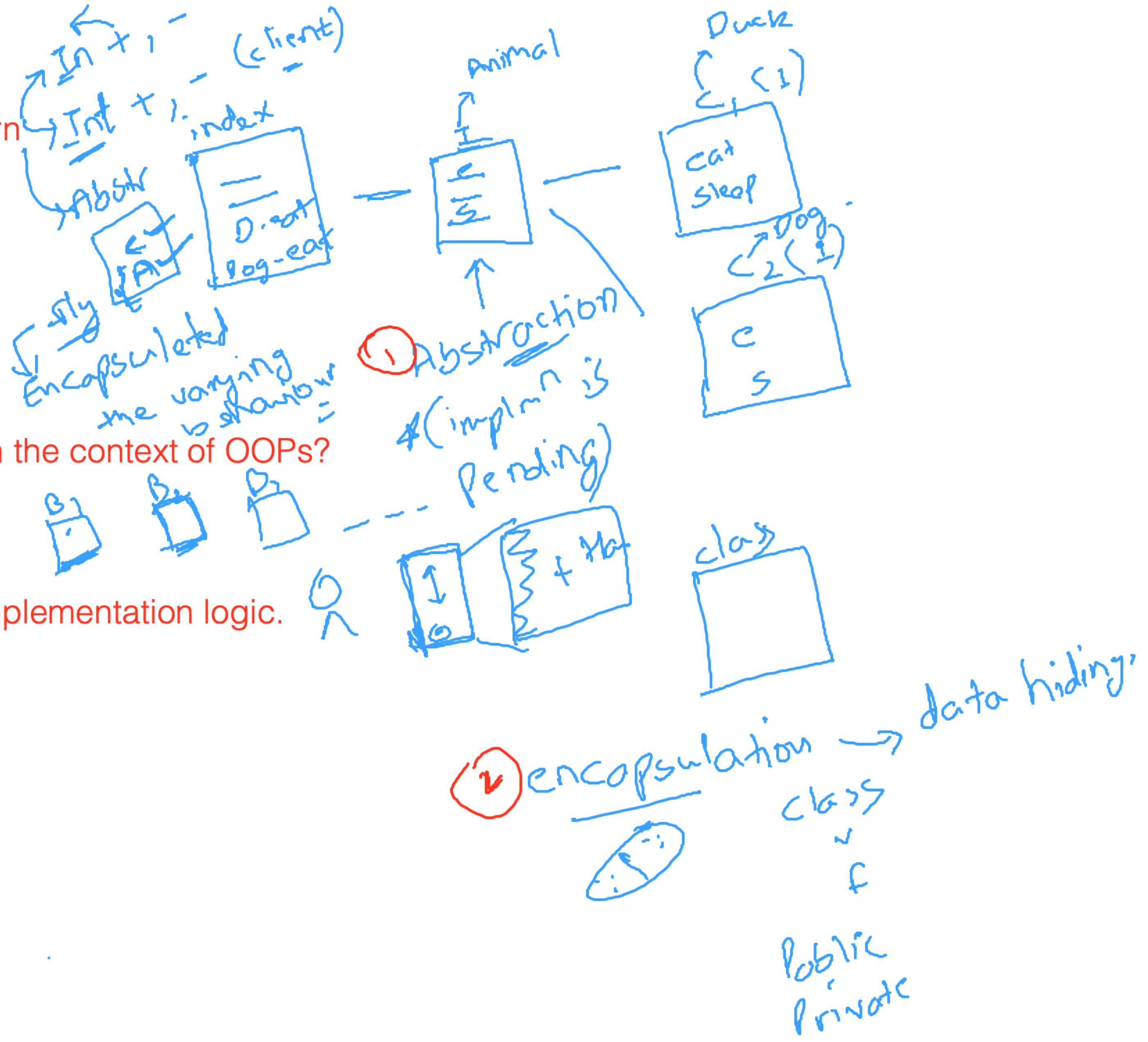
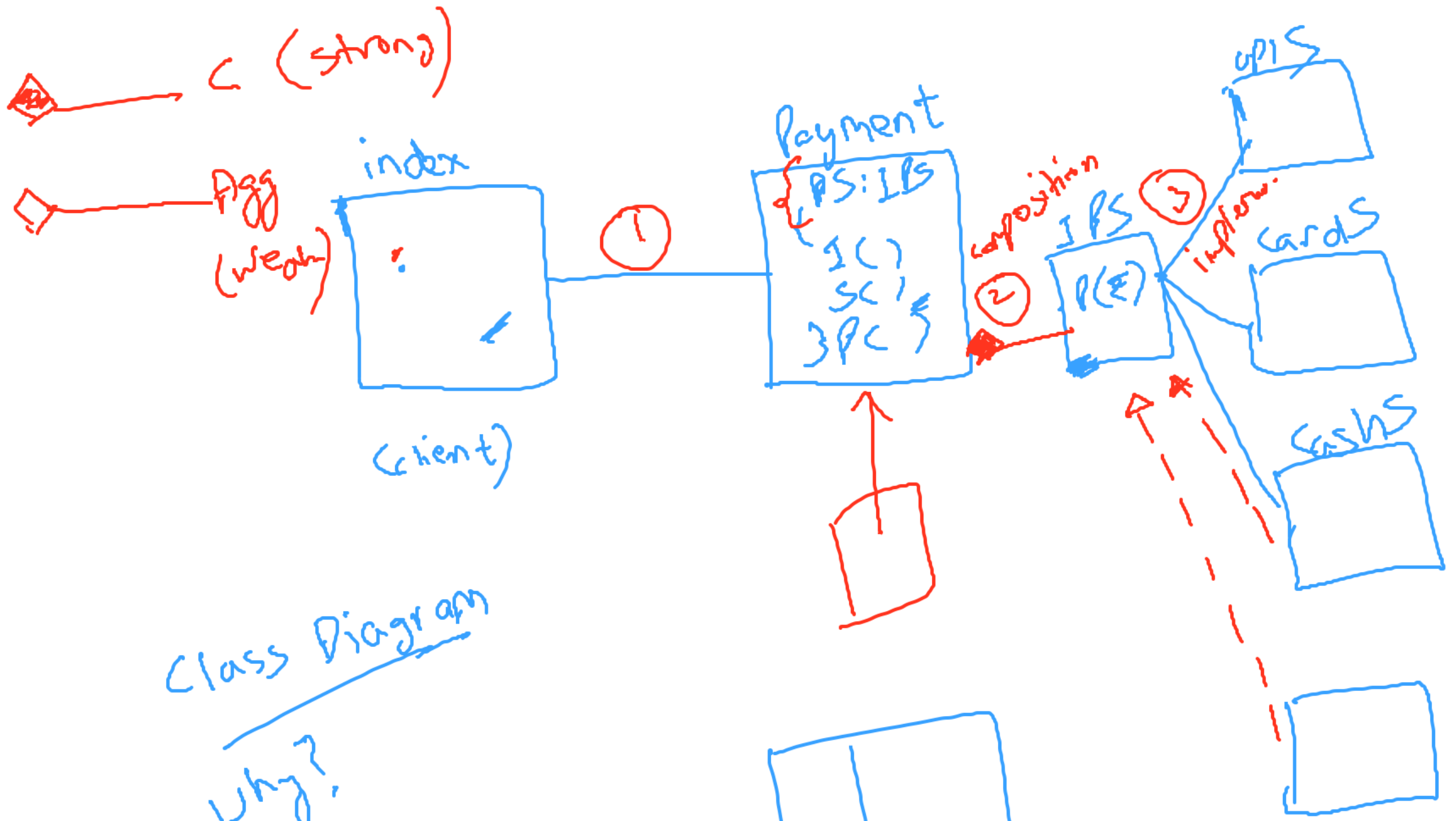
Last Session : Strategy Design Pattern
- Problem Statement
- Abstract class vs class
  - Why AC can't have objects?
- Abstract class vs interface
- Application of Strategy
- Explain Strategy Design Pattern
- Code level
- How we can achieve Abstraction, in the context of OOPs?
  - Abstract class
  - Interface
- What is Abstraction?
  - user doesn't need to know the implementation logic.
  - Smartphones, APIs, etc.



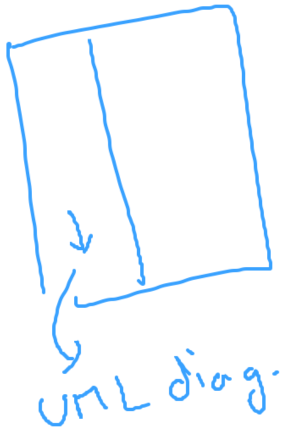
Class Daigram
Coupling

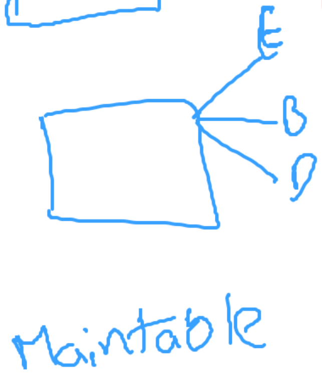


Class Diagram

1. Why?
2. Rules
3. Expectation

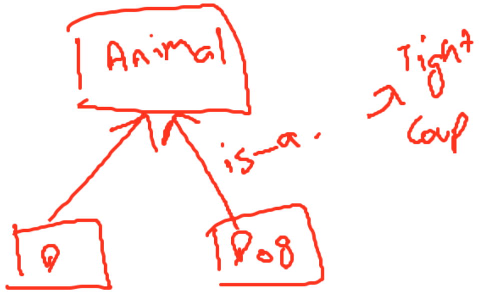
draw.io





C v I
↓
Coupling

① I



② Composition (has-a)
↳ Loose coupling

has-a >> is-a

LC TC

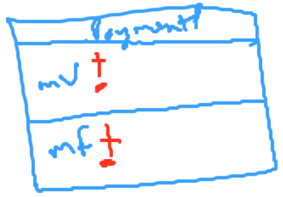
modularity ←

inhe → extends

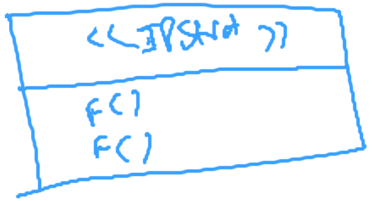
int → implements

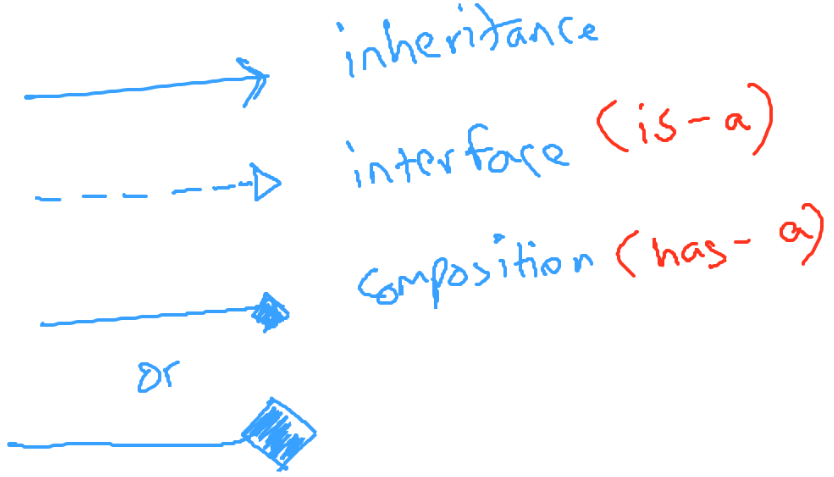
Strategy Design Pattern
- Encapsulate the varying behaviour
- eg Duck (flying)
- Encapsulated ? FlyB, NoFlyB, SonicFly, SuperSonicFly
- Encapsulation : Class (mV, mF)
- What is SD? Having Family of Algo aiming to achieve one goal, same thing in different way
- class Diagram for Stratedy Design Pattern

Rules of Class Diagram



class





+ Public

- Private

# Protected

Observer Pattern

- Problem Statement :
- Solution : Subject-Object, Publisher-Subscriber (Pub-Sub), Notification System
- What is Observer Pattern? Publisher-Subscriber model