Agenda (UML Diagrams)

1. Use case diagrams
2. Class diagrams.

UML diagrams.

Communication : How to communicate your technical knowledge.

On a daily basis...

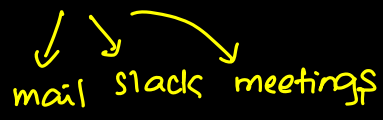
(i) clients
(ii) Tech leads /architects
(iii) EMs
(iv) PMs

Next class.

-> Requirement from users perspectives.
-> Approval of design doc. ↘ TODAY
-> career progressions
-> Requirement

Common ways to communicate?

1. Words



"Problems in communicating in words?"

1. misunderstanding / Ambiguity
2. explaining complex things get difficult.

What's the fix?

A _____ is worth 1000 words? -- PICTURE.

② PICTURE

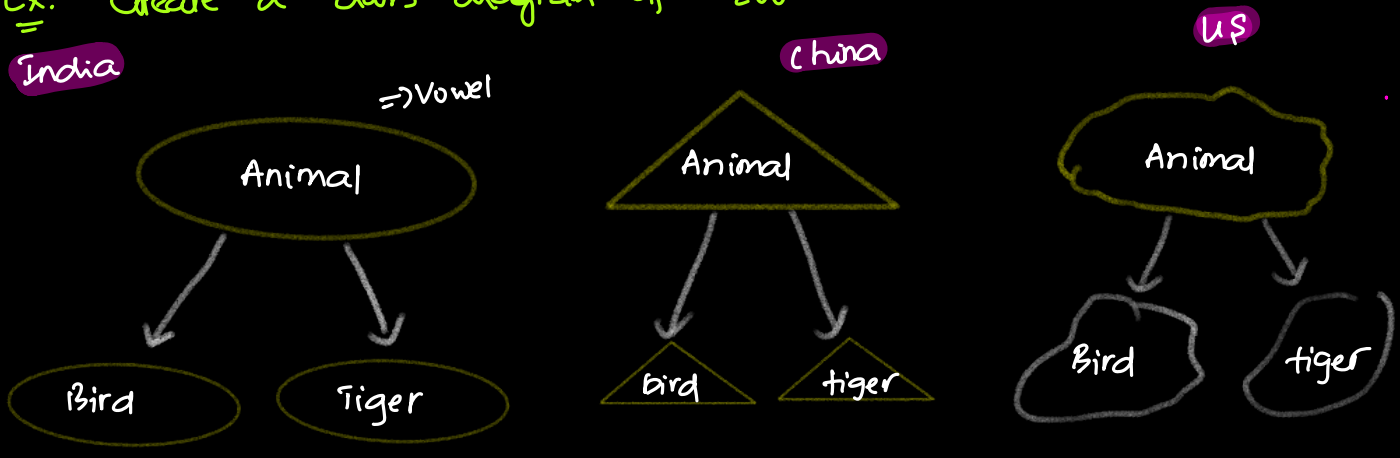
Benefits

1. less ambiguity
2. complex things, its easy to relate them
3. visualize.

Problems

"No standarization".

Ex: Create a class diagram of 'zoo'



Follow standards → UML.

UML: unified modelling language

↳ standardisation of how to represent diff SWE concepts using diagrams.

Type 1

Structural

↳ how code is structured

classes, methods, attributes, interfaces etc

Type 2

Behavioural

↳ how system behaves.

use case / feature

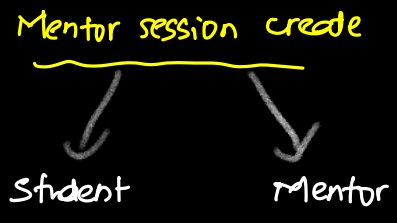
Structural	Behavioural
1. class diagram	1. activity diagram
2. package diagram	2. use case diagram
3. object diagram	3. sequence diagram.
4. component diagram.	
↳ HLW.	

Behavioural diagram

Use-case diagram:

1. Diff use cases / functionalities supported by our SLW.
2. Who uses the functionalities?

ex:



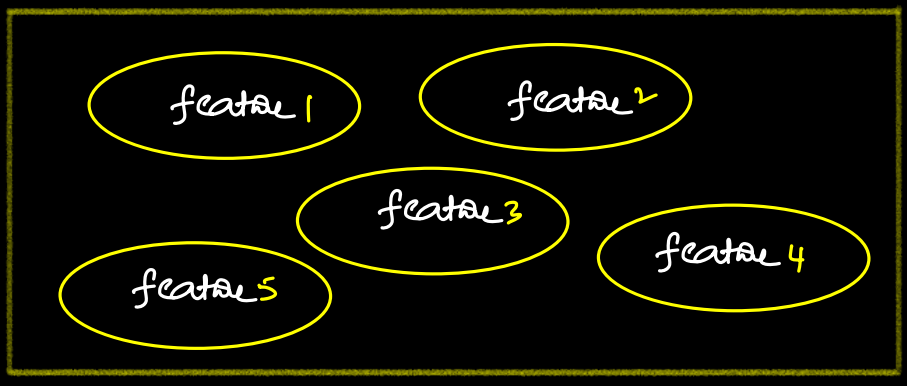
ex: login

Everyone.

How to create use-case diagram? (Just 5 key words)

① System Boundary [Represents scope of my SW]

Every feature that is supported by my SW will be inside this boundary.



Payment happening on a 3rd party is not scope of my SW.

② Use Case

→ functionality / feature / action.

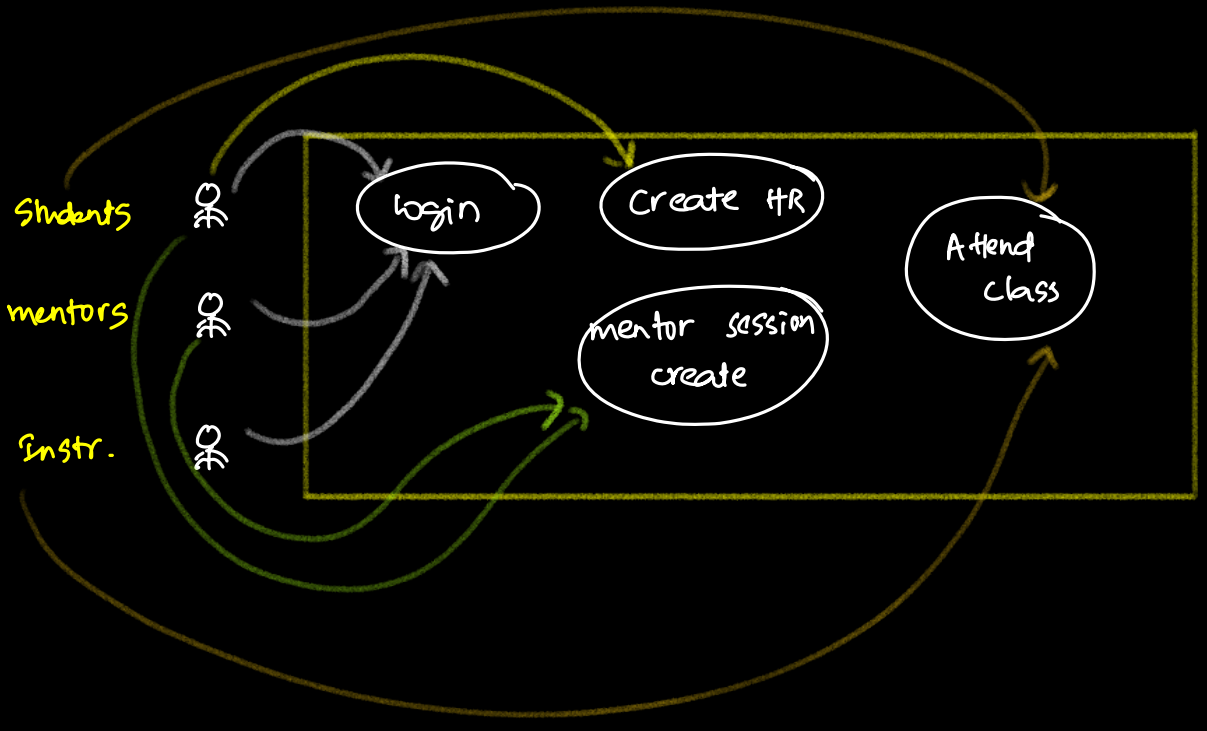
→ Should always be VERB.

→ Represented by an oval.

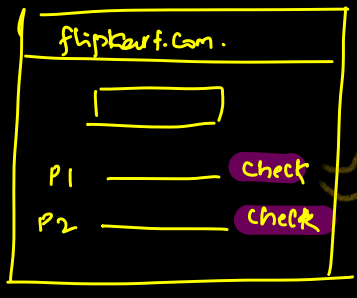
ex:

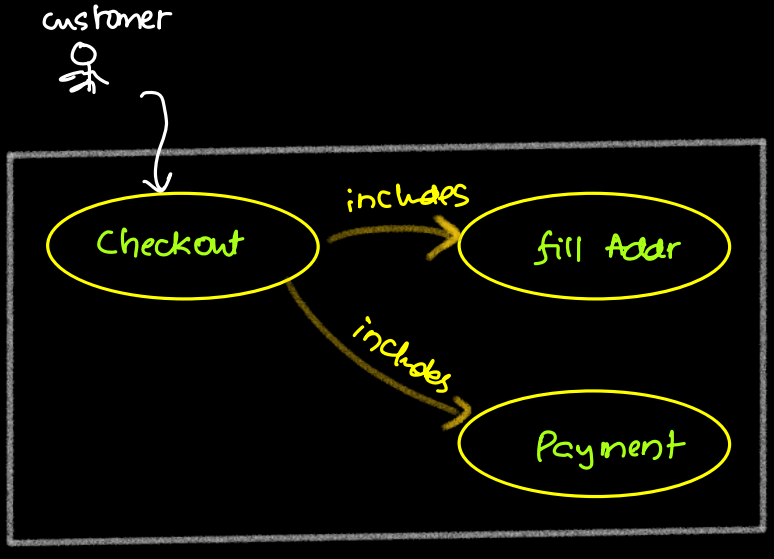
③ Actor

1. People who use our functionalities.
2. Noun.
3. Stick diagram.
4. Arrows indicates the rel^n b/w actor & use-case.



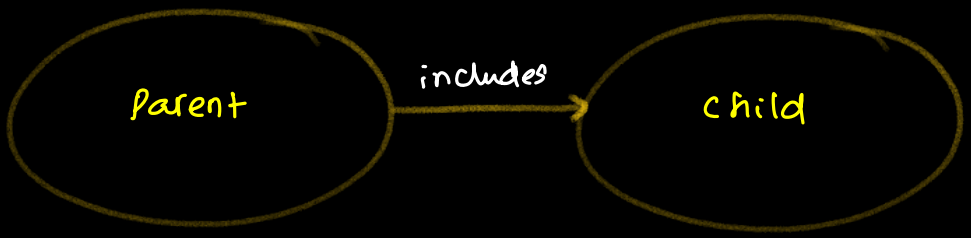
④ Includes





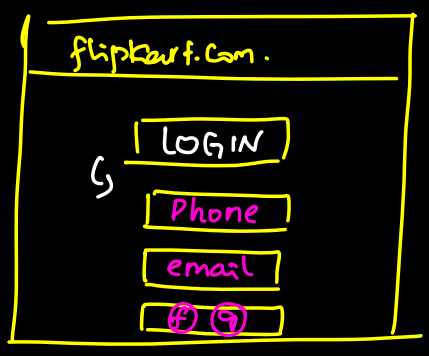
Think of every feature like a fun^n.

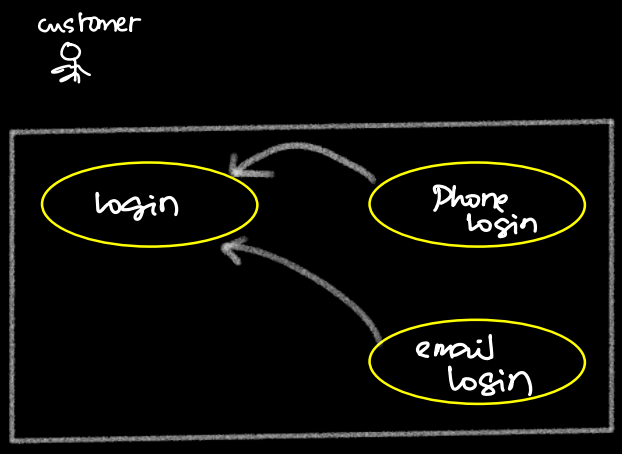
```
void checkout() {
    // fillAddress()
    // makePayment()
}
```

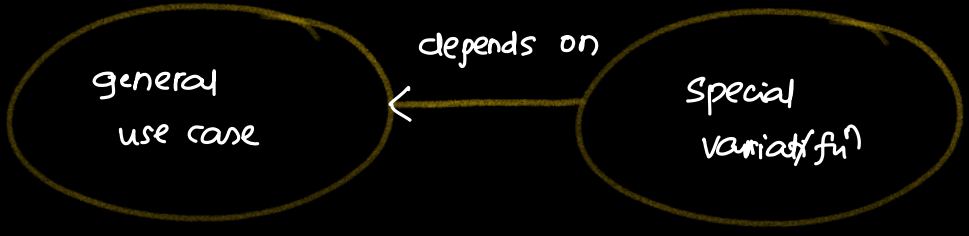


5. Extends

If one feature depends on other variants.





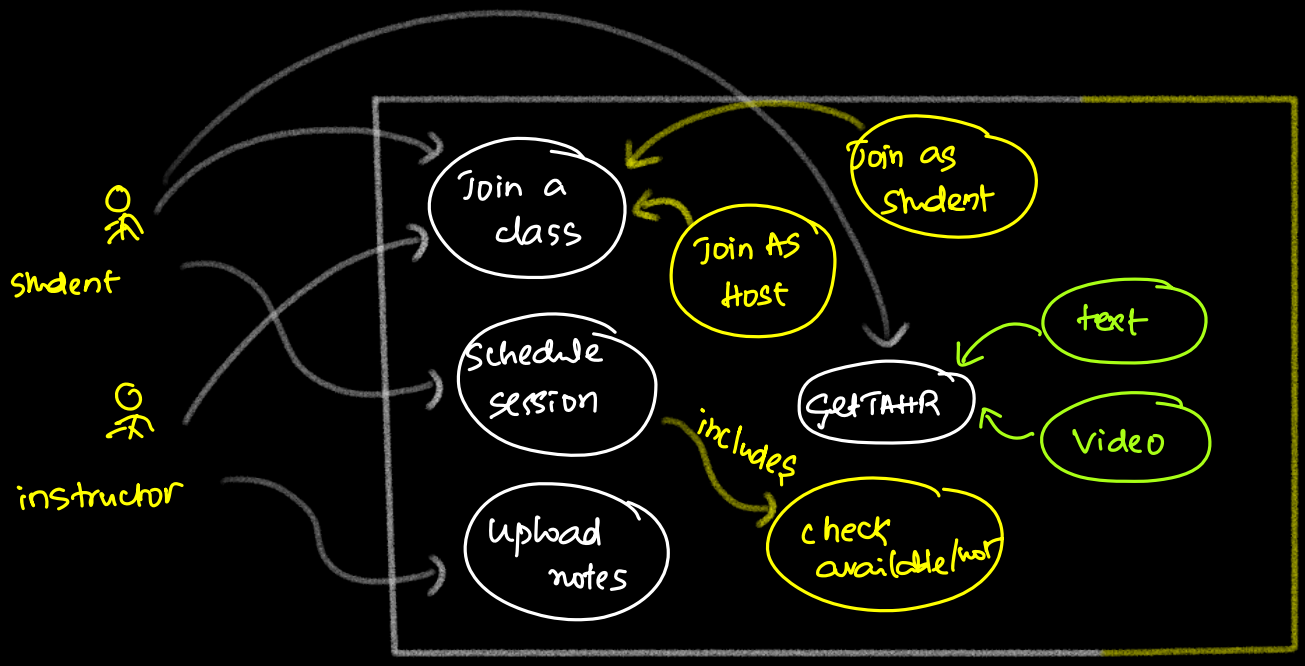


Q. Use case diagram for scaler software system.

=

1. 5 use cases, atleast
2. 2 Actors, atleast
3. 1 use case involves another one, atleast
4. 1 use case should've special use case // variants, atleast

// choose something
Other than login



Structural diagram

Class diagrams

1. Representing different entities in slw slm.

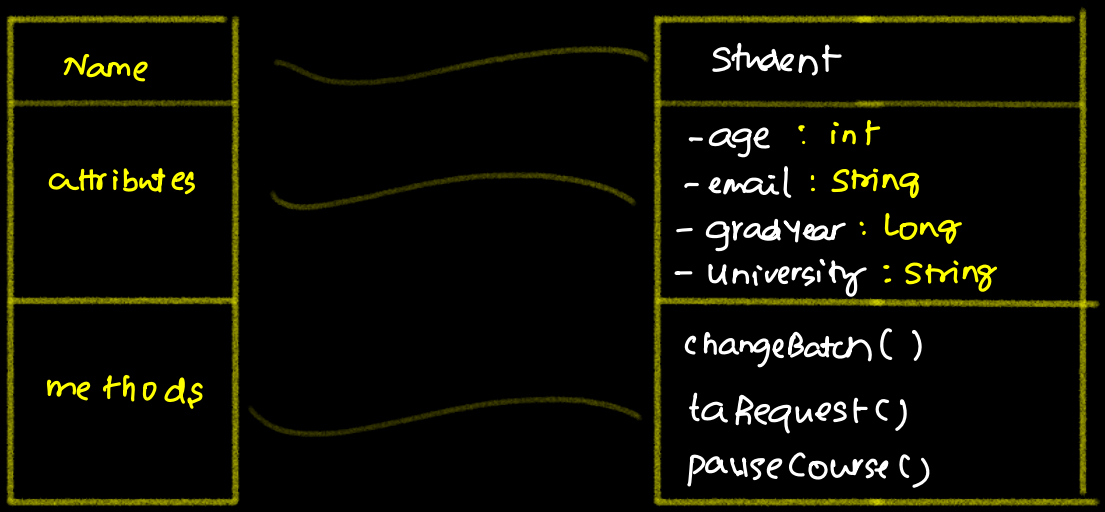
   - class
   - abstract
   - interfaces
   - enums

2. Representing relation blw the entities [slw slm].

   - → impl an interface } IS-A
   - → extend a class } IS-A
   - → having an attribute of another class. } HAS-A

class

Rectangle with 3 things

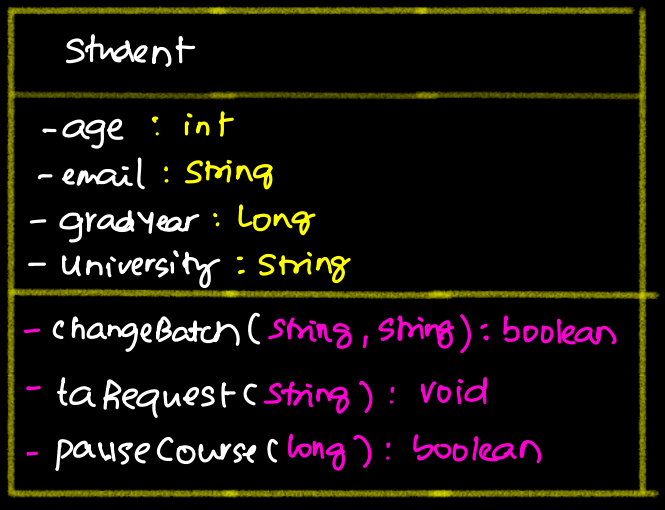


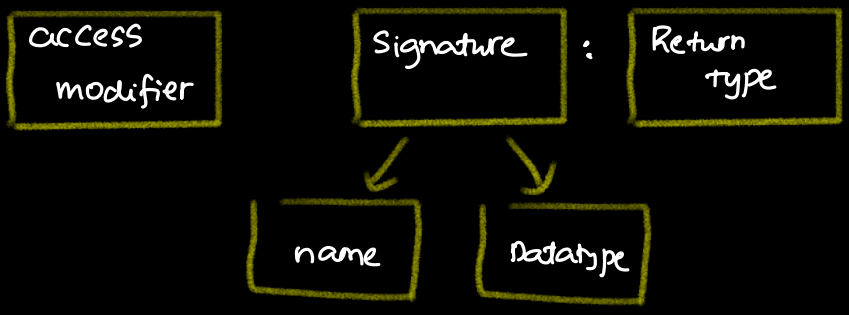
Attributes

- \+ public
- \- private
- # protected

Methods

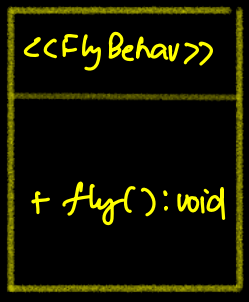
How to identify a method? Signature [name & parameters]





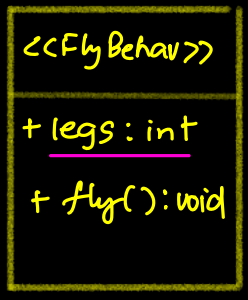
Interfaces

Rectangle with 2 things



how to represent static attributes?

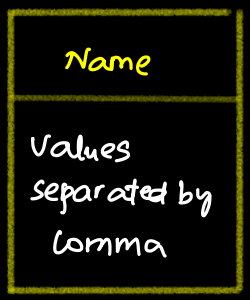
Whether the attribute / method is inside a class / interface / abstract simply put an underline below the field. to mark it static.

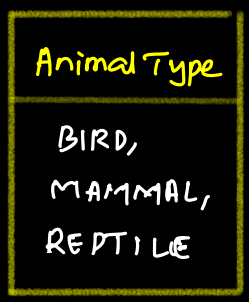


Abstract class

Just like class but only the name of the class is in italics

Enums





← ex.

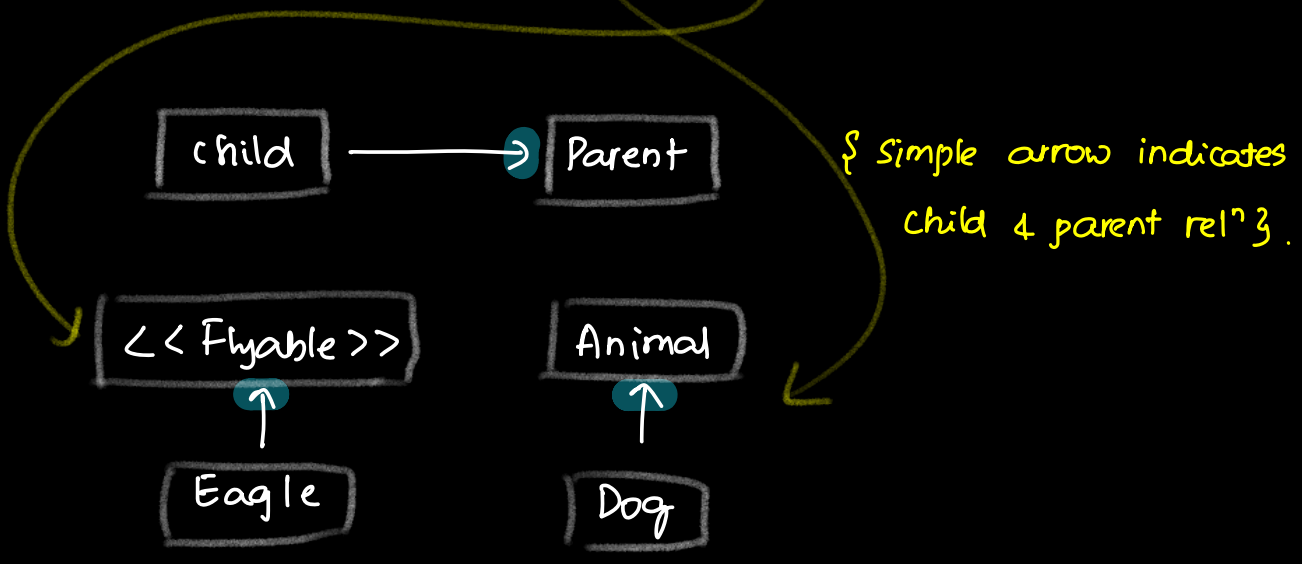
→ inner enums needn't be represented here.

Going back to rel^n blw entities

2 categories of rel^n:

① IS-A

Inheritance



② HAS-A

ASSOCIATION

↓

having an attribute of another class.

Two types of association:

1. composition
2. agregation

Aggregation

1. Collect / group / combine things
2. The other entity has independent existence

B should exist without A.

```
class Batch
    Instructor iId
    int bId
```

Instructor can exist without a Batch.

How to represent?

unfilled.

Composition

1. Create things
2. The other entity has no independendent existence

B cannot exist without A.

filled.

ex:

instructor ——— class

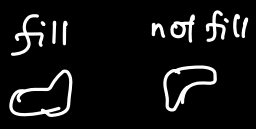
class ——— assignment

House ——— door.

Hence

Composition : Strong rel^n.

Aggregation : Weak rel^n.



More ex:

① Ticket ——◇—— show

↳ fill / not fill.

Ans: show can exist without Ticket.

② Scaler class ——◇—— Assignment

↳ fill / not fill.

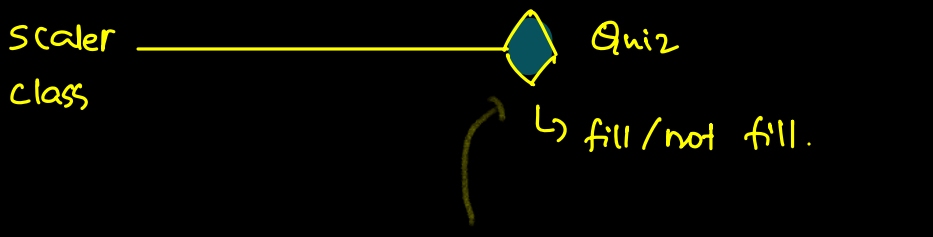
Ans: assgn can not exist without scaler class.

③ Theatre ——◇—— Movie

↳ fill / not fill.

Ans: movie can exist without theatre.

④



Ans: quiz cannot exist without class: