Agenda

Start @ 9:05 pm

1) UML Diagrams

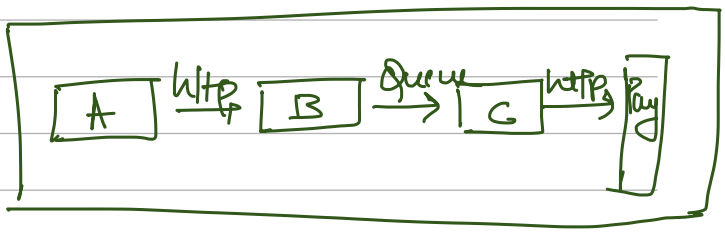
1) S/W Engg Managers ⇒ Req. | Code | Design | Approval
2) Client ⇒ Req. | Business use case
3) Architect ⇒ Design | Approvals
4) Product Managers ⇒ Req | Business use case

⇒ Ways of communicate

Words

↳ It is not easy to explain the complex system in words
→ Misunderstanding of requirements

eg



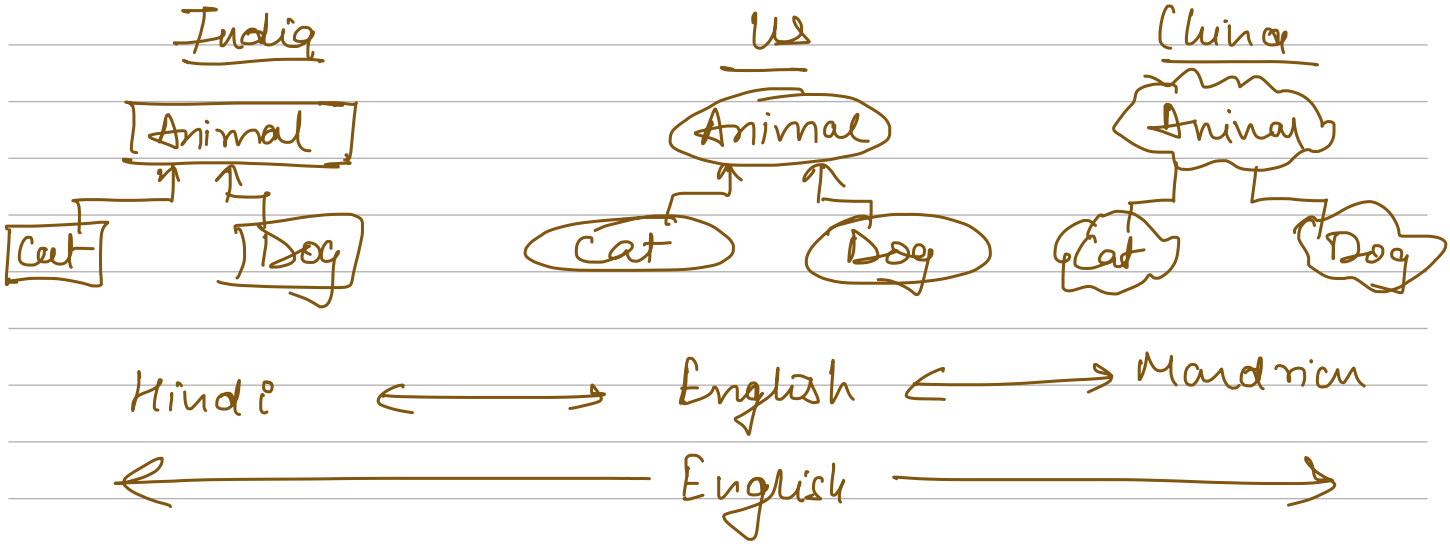
A picture is worth 1000 words

↓

Image | Flow Chart | Diagrams

→ Less ambigous
→ Easier to visualize the complex system
→ Easier to explain & understand.

eg. Amazon



Hindi ⟷ English ⟷ Mandrin

⟵ English ⟶

UML (Unified Modelling Language)

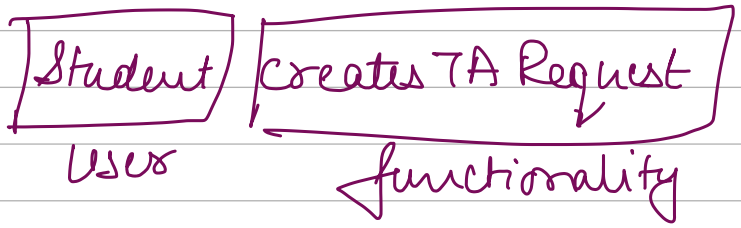
↳ Standard way of representing SWE concepts in form of diagrams

UML

Structural	Behavioral
⇒ Structure of codebase	⇒ functionalities of the system ⇒ How system is interacting
1) Class diagram	1) Use case diagram
2) Package diagram	2) Sequence diagram
3) Object diagram	3) Activity diagram
4) Component diagram	

Use case diagram

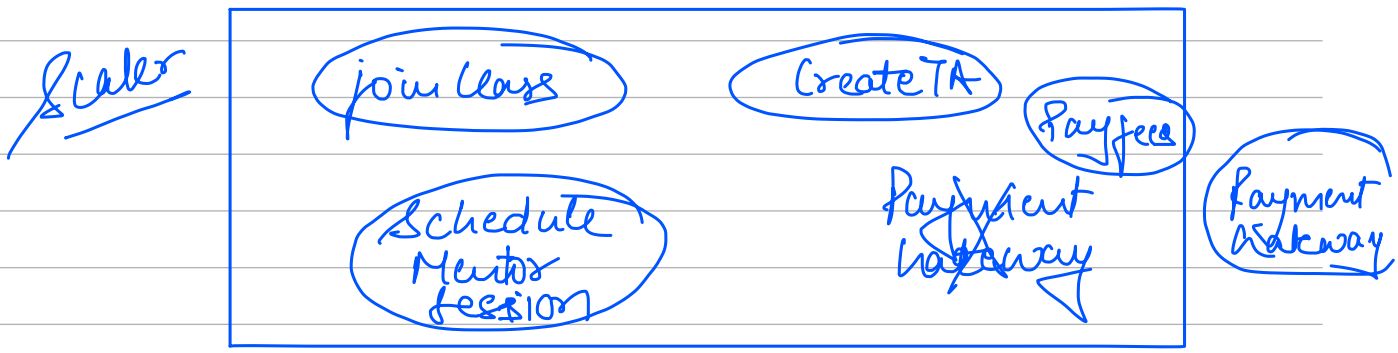
1) Behavioural
2) deals with actions / functions
3) what the functionalities are supported by our system
4) Who is the user of that function



#5 Keywords

1) System boundary

1) Represents the scope of the system



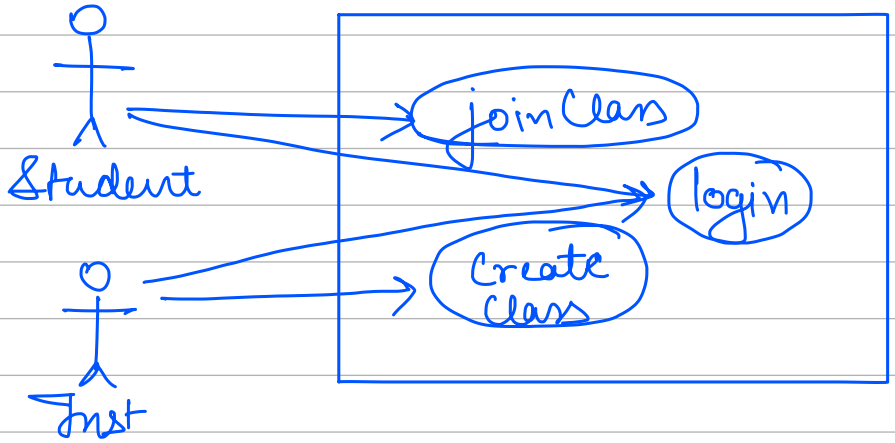
2) It doesn't include 3rd party functionalities

2) Use case ⇒ Verb

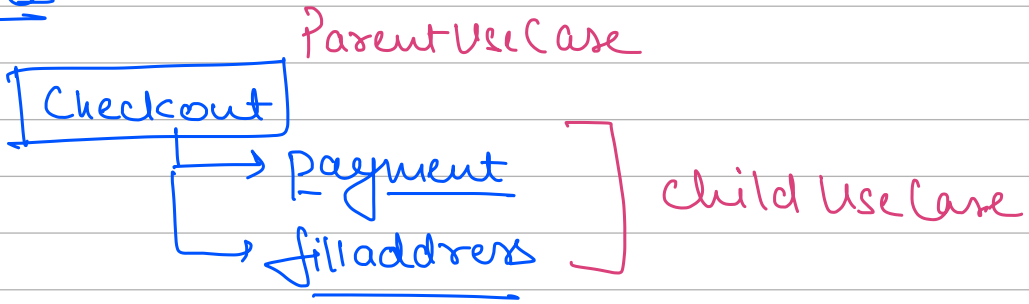
1) Behavior / functionality
2) Represented by Oval shape

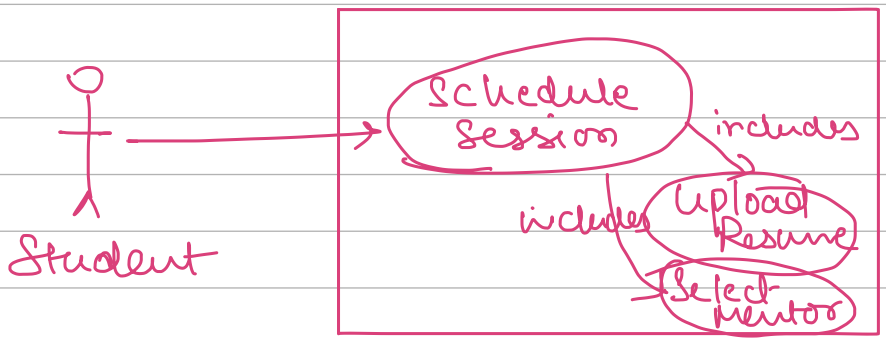
3) Actor ⇒ NOUN

⇒ User of behaviours
⇒ Stick diagram



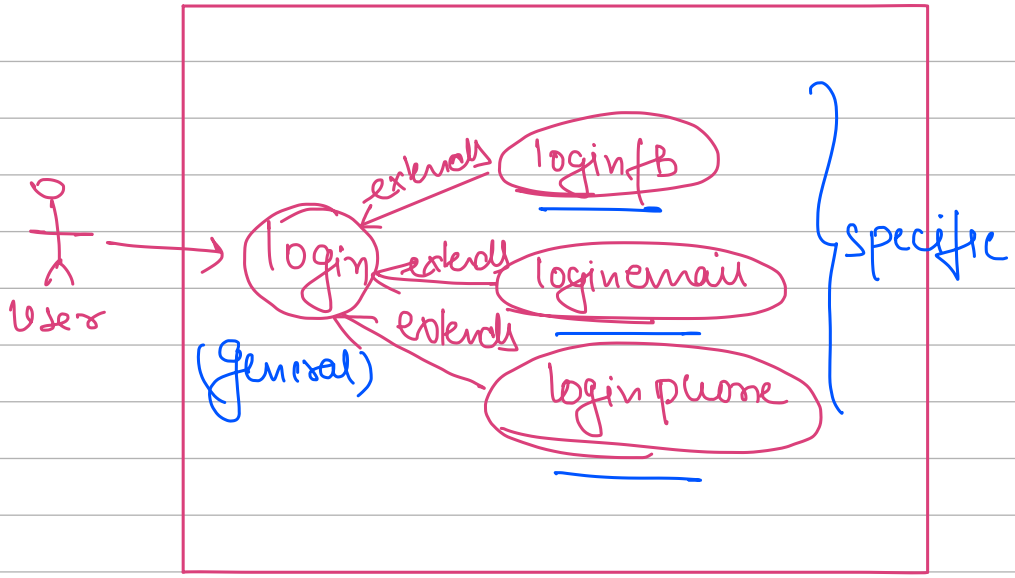
4) Includes

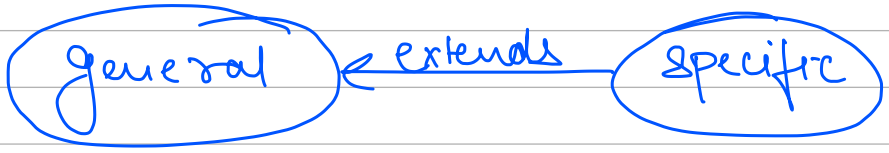




5) Extends

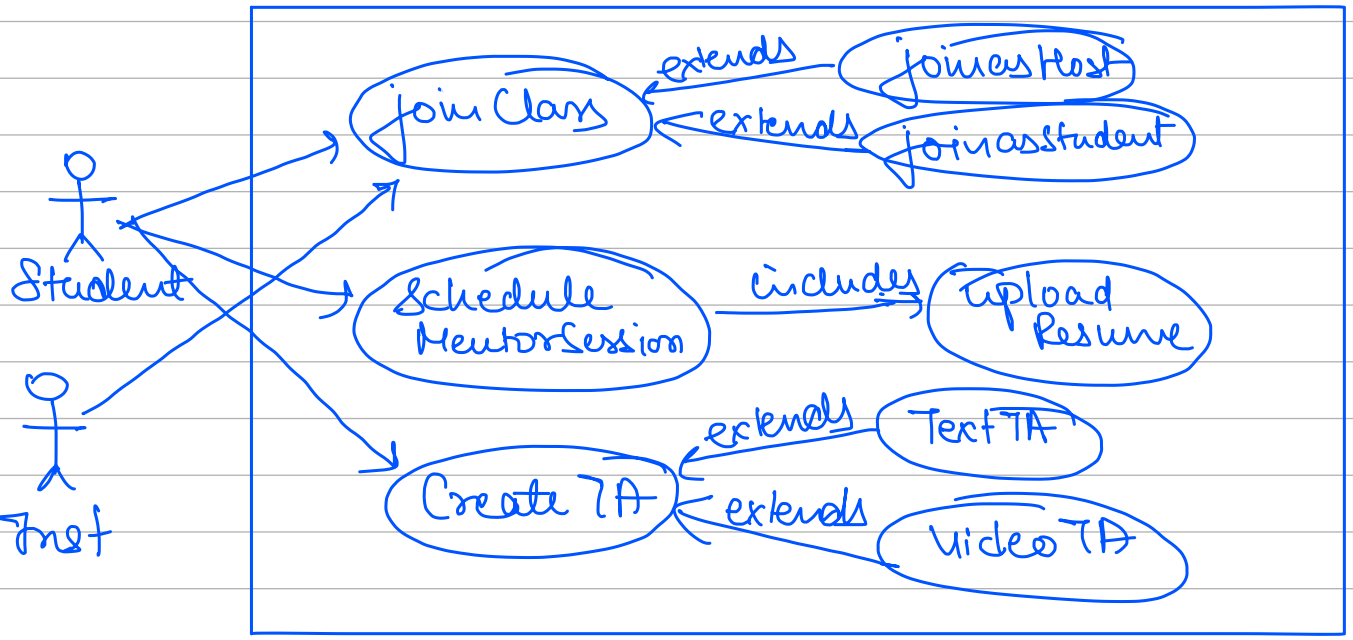
If one feature has multiple variants





C.U.

- Scaler as system
- 3 Use Cases
- 2 Actors
- includes & extends



Class diagram

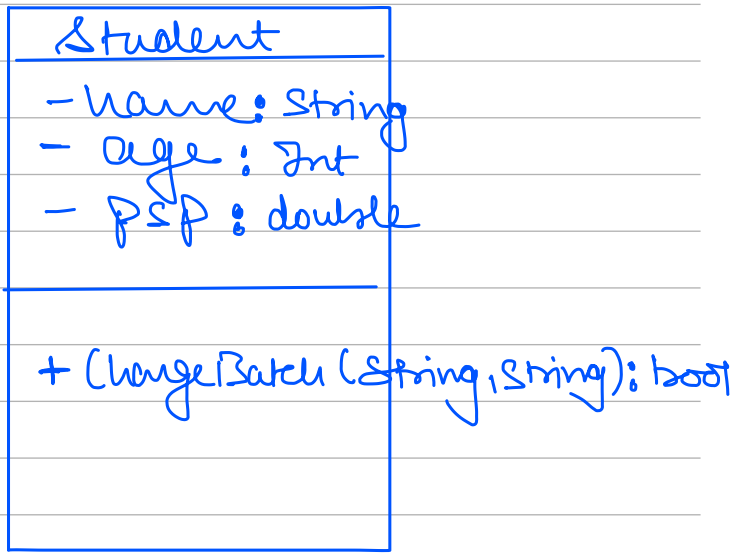
→ Represent different entities in the s/w system

- Class
- Interface
- Enum
- Abstract Class

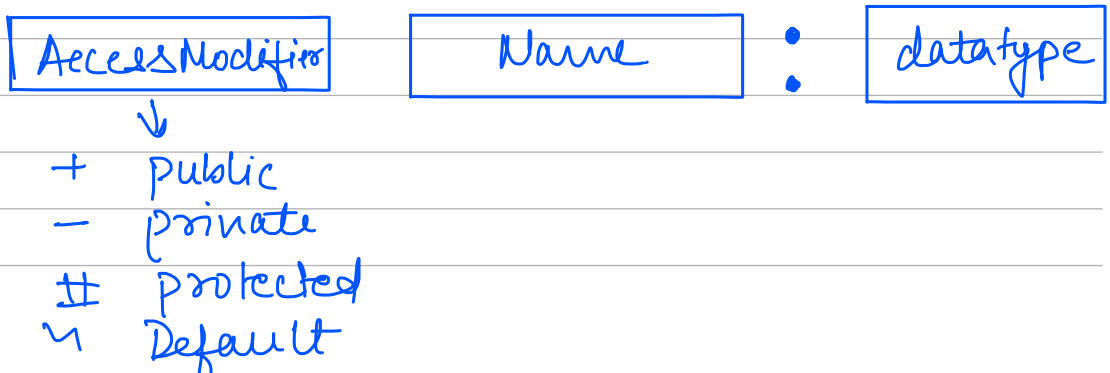
→ Also represent the relation b/w the entities

- is-a
  - extending a class (inheritance)
  - implementation of interface
- has-a
  - having an attribute of another class

Class



Attributes



Method

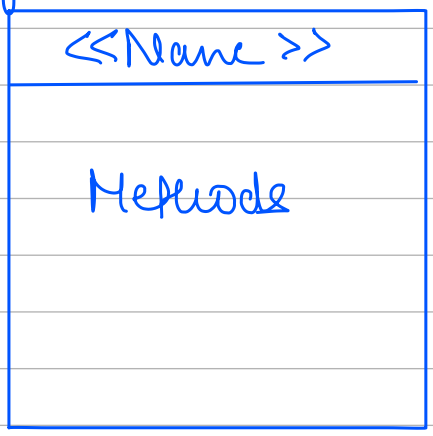
bool changeBatch(String oldbatch, String newbatch)

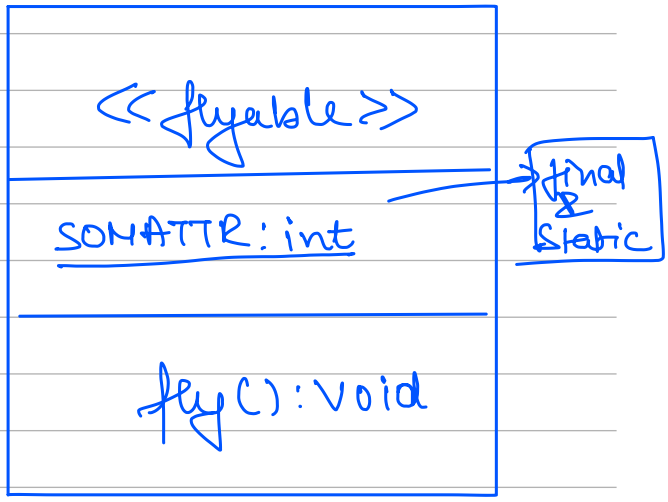
+ changebatch(String, String): bool

Static : Underline

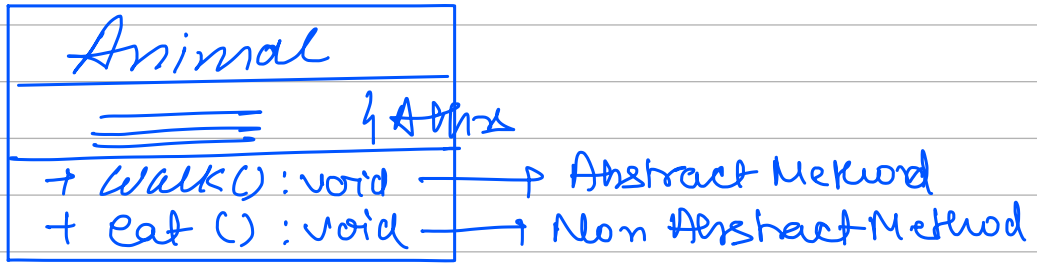
Final : CAPITAL LETTERS

Interfaces





Abstract Class => Italics write the name in italics

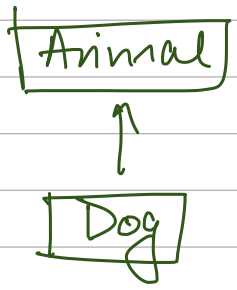


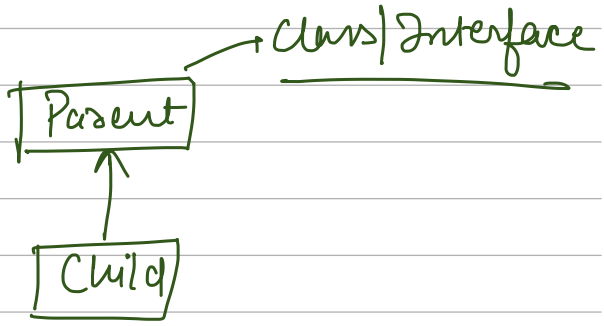
Enum

CAR_TYPE
SUV,
SEDAN,
Electric

How to represent relationships

1) Is-A → Inheritance
   → Interface implementation

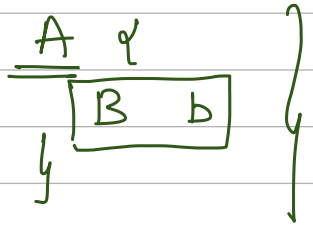




2) Has-A ⇒ have an attribute of other class

Association
→ Composition
→ Aggregation

eg



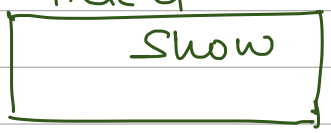
Association | Has-A

1) Aggregation

⇒ Collection

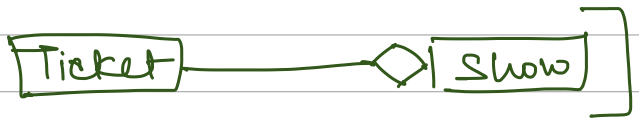
⇒ Rel^n b/w 2 entities A & B where A has an object of B & B can exist independently

This an aggregation WEAK Relationship

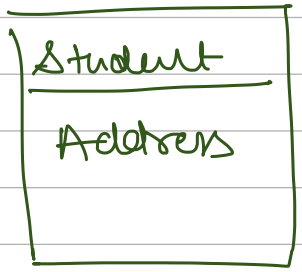


Ticket has an attribute of Show class

⇒ Show can exist without ticket



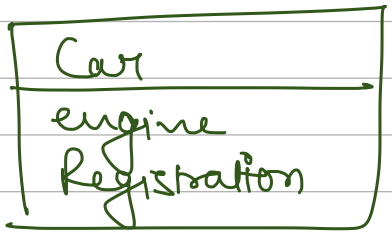
2) Composition

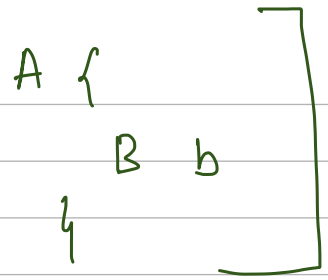


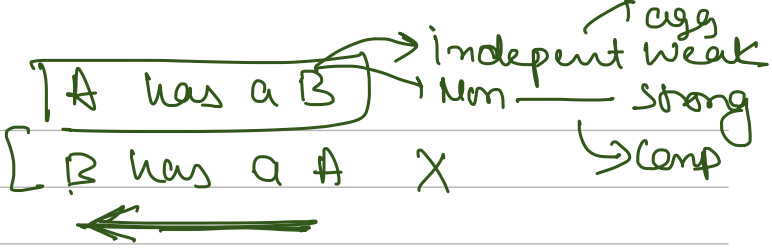
No independent existence of other object.

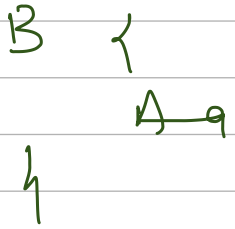
STRONG Relationship

eg




A {
B b
}


A has a B
B has a A
agg
independ
weak
Non
strong
comp
X


B {
A a
}