L-4

UML Diagram

Application → Paragraph
Application → Diagram

Type

UML Diagram

→ Structural (static)
⇒ What will be Structure of the Application.
↓
(7) Diagram

→ (Dynamic) Behavioural.
⇒ How Components Interact with each other.
↓
(7) Diagram

→ 12 Diagram is not @ Important becoz the depends on use cases.
⇒ But 2 Diagrams are Important.

① Class Diagram ② Sequence Diagram

Class Diagram :- How classes are connected & which type of classes will have in our application.

① Class Structure.
② Association / connection. → No need to write for concrete class

```
class car {        Public :
    string brand;
    string model;
    int enginCC;
    Private :
    void {
        Startengine();
        Stopeengine();
        accelerate();
    }
    break;
}
```

<<abstract>>

car
+ brand : String
+ model : String
+ enginecc : int
- Startengine() : void
- Stopengine() : void
- break() : void

{ Class Name
{ characters / variables
{ Behaviour / method

Access modifier.

UML Diagram.

Access modifier

	within class	from child class	Outside the class.	
① Public	✓	✓	✓	+
② Protected	✓	✓	X	#
③ private	✓	X	X	-
				~

Overview

<<abstract>>

ClassName.
(+) Var1 : datatype
(-) Var2 : datatype
(+) Method1 : datatype
(+) Method2 : datatype

Associations.

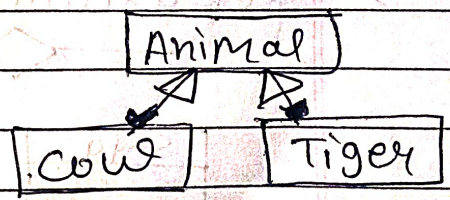
- Class Association
  - Inheritance.
- Object Association.
  - Simple Association
  - Aggregation
  - Composition

Composition

Beaz only one way is available to define in programme.

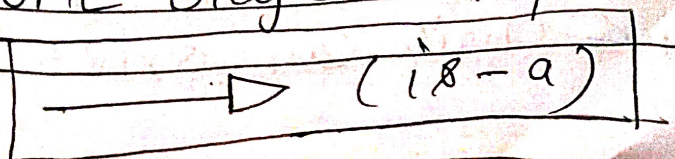
Inheritance :- (is-a) Relationship.

```
Class A {
    method1 ();
};
Class B : public A {
    method2 ();
};
main() {
    B *b = new B();
    b-> method1();
    b-> method2();
}
```



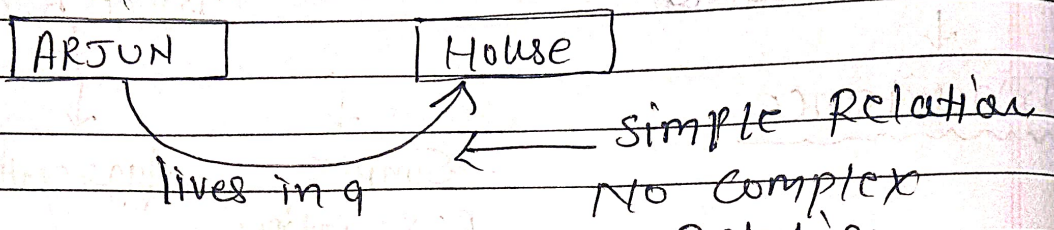
Cow is-a Animal

UML Diagram Representation of Inheritance



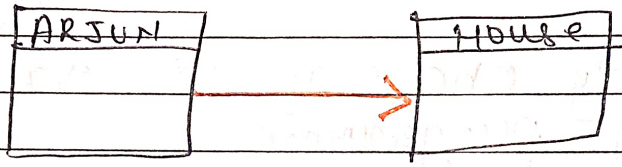
Composition :- (has-a) Relationship

→ Simple Association



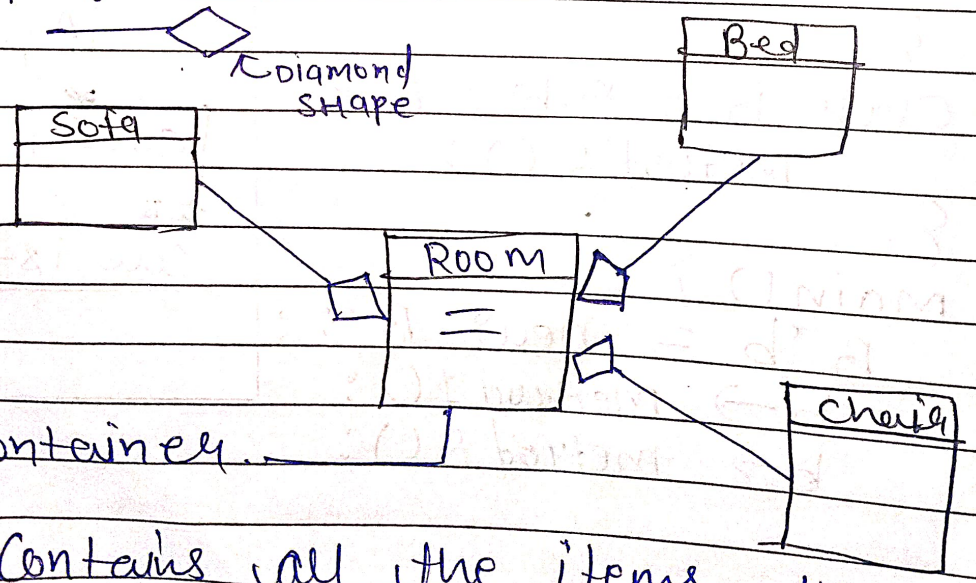
Representation.

→



→ Aggregation

→ Representation.

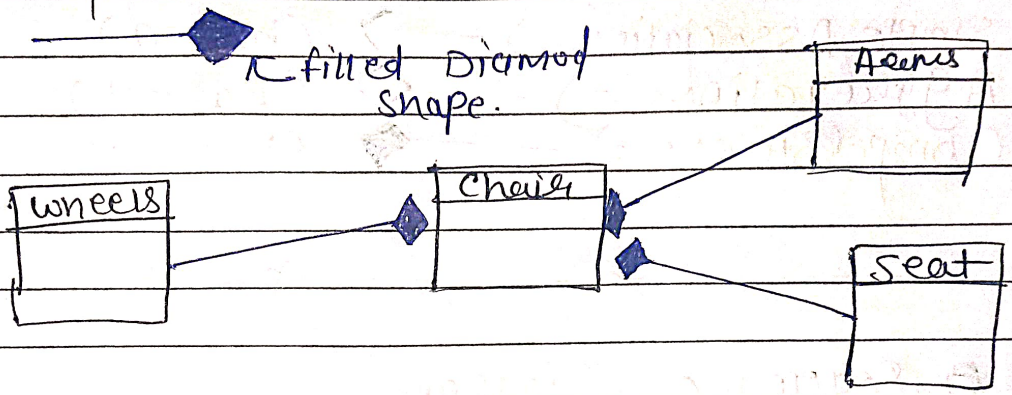


& it Contains all the items like Sofa, Bed & Chair within it. But Sofa, Bed & Chair Can exist without Room.

→ Composition

→ Representation.

"has-a"



→ Here. wheels, ARMS & seat are parts of Chair & they can't be exist without chair.

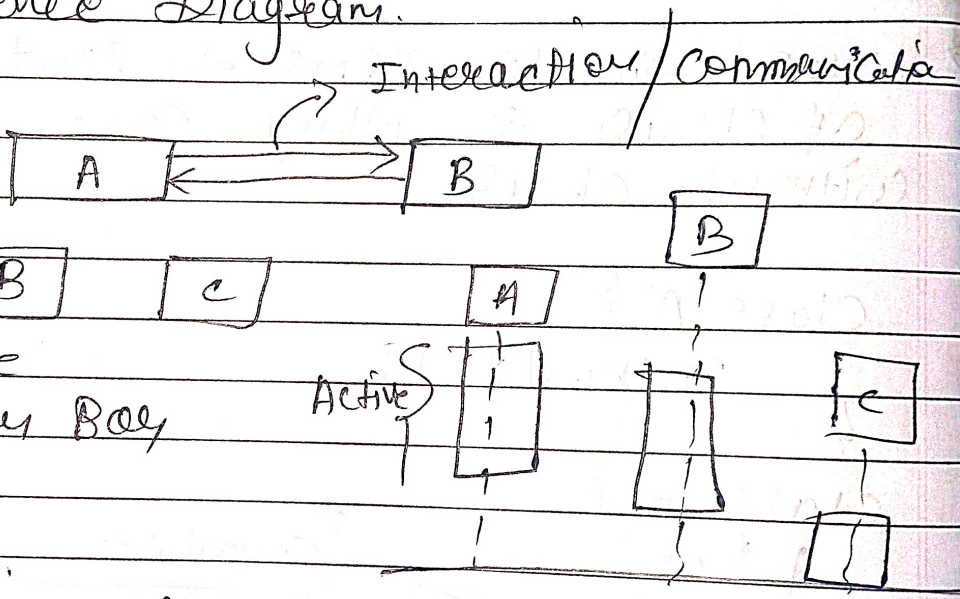
```
Class A {
    method1();
};
Class B {
  A *a;
  B() {
    a = new A();
  }
  method2();
};

Main() {
  B *b = new B();
  b→ method2();
  b→a→ method1(); ← Composition
}
```

constructor.

Representation Overview

1. Inheritance ⇒ ——▷ (is-a)
2. Simple Association ⇒ ——→ (has-a)
3. Aggregation ⇒ ——◇ (has-a)
4. Composition ⇒ ——◆ (has-a)

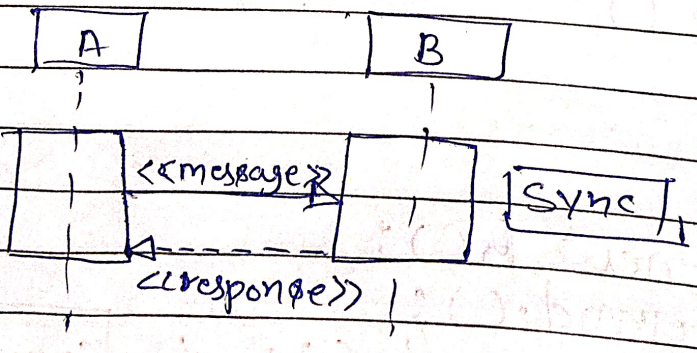
Sequence Diagram.



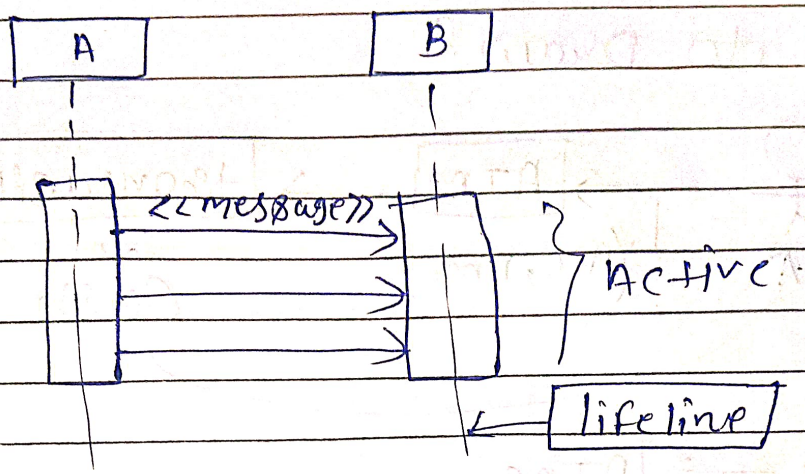
Message

Async

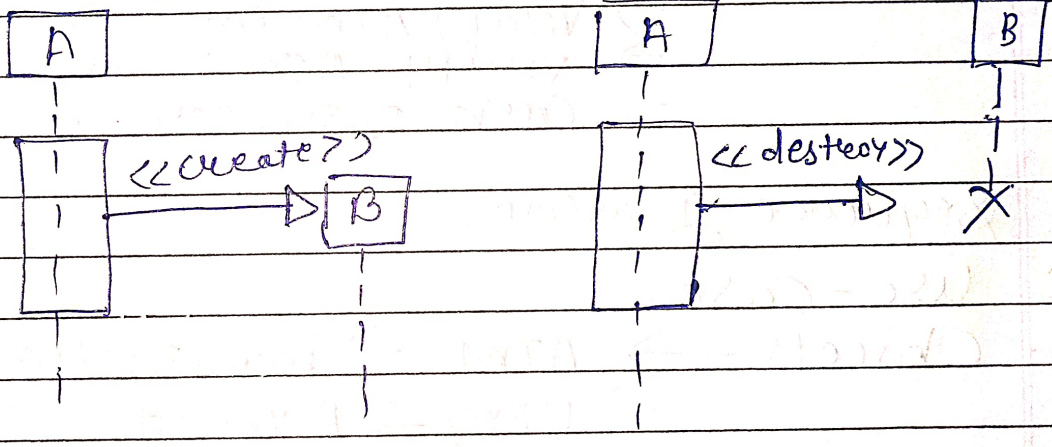
Sync → wait for response.



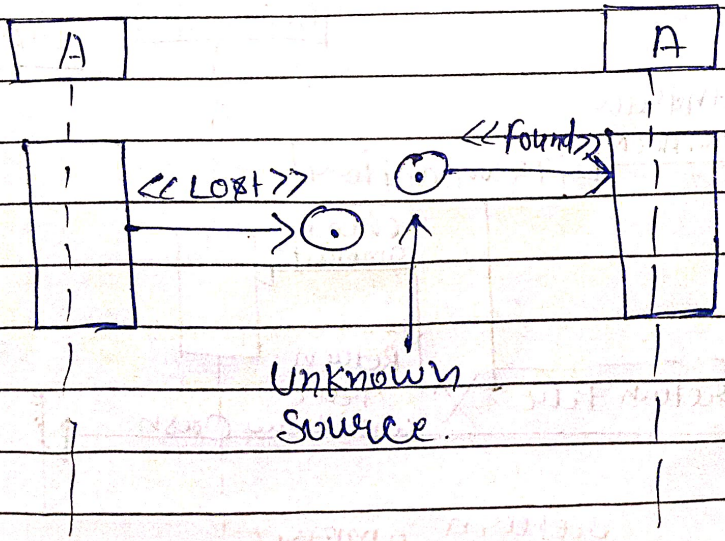
Async



Create message & Destroy message.



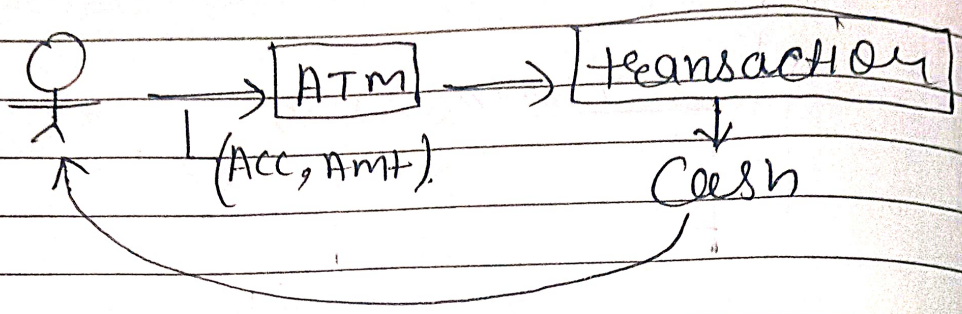
Lost message & found message.



① alt :- if-else

② Option :- if

How to Draw?



① User → ATM
↳ Account No. & Amount }.

② ATM → Transaction
↳ Verify ATM Pin
→ Verify ACC
→ Cash Dispenser }

Sequence Diagram.

① Use-Case.

② Objects → ATM → Transaction
→ User → Account
→ Cash Dispenser

③ Draw Sequence Diagram.