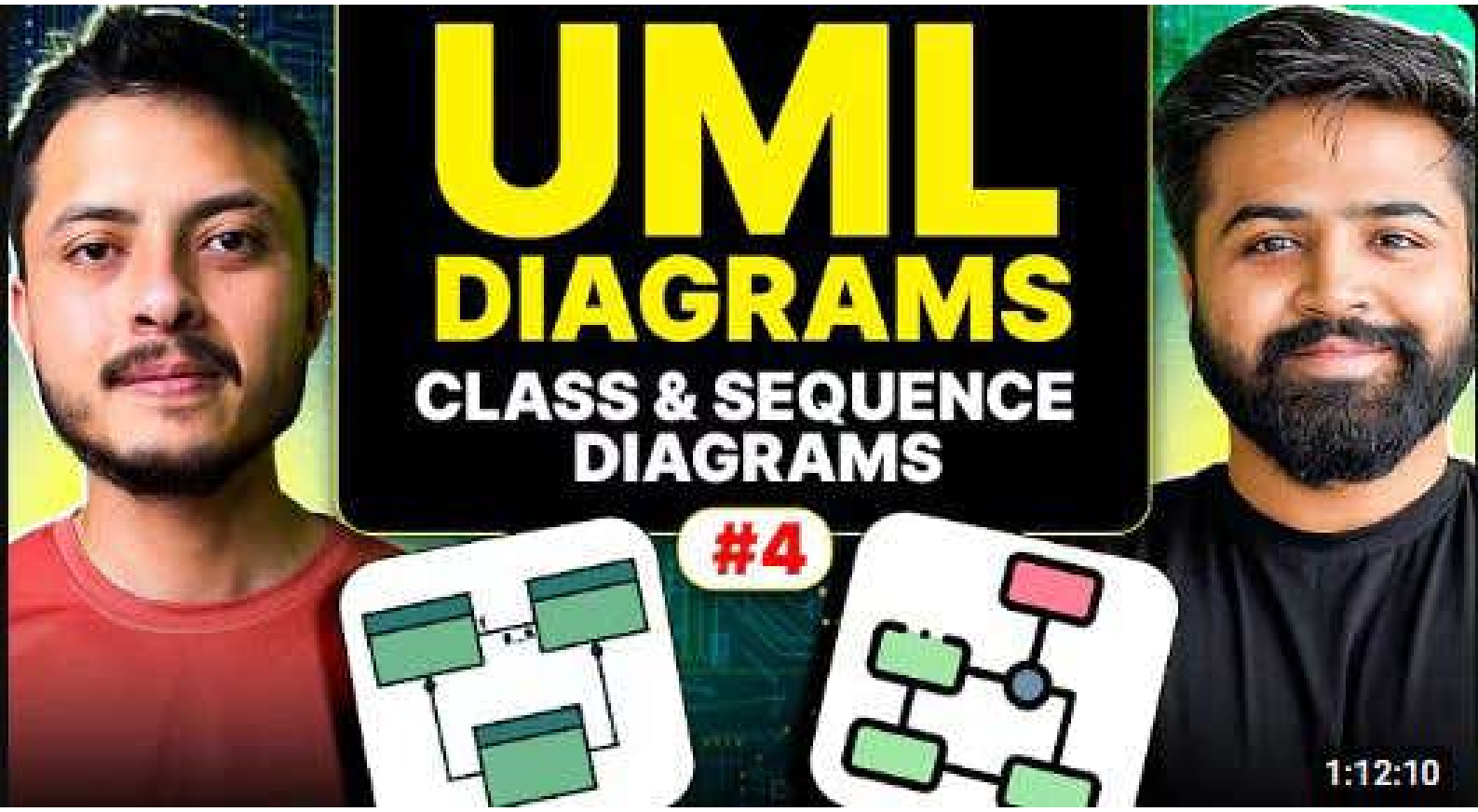

UML
DIAGRAMS
CLASS & SEQUENCE
DIAGRAMS
#4
1:12:10

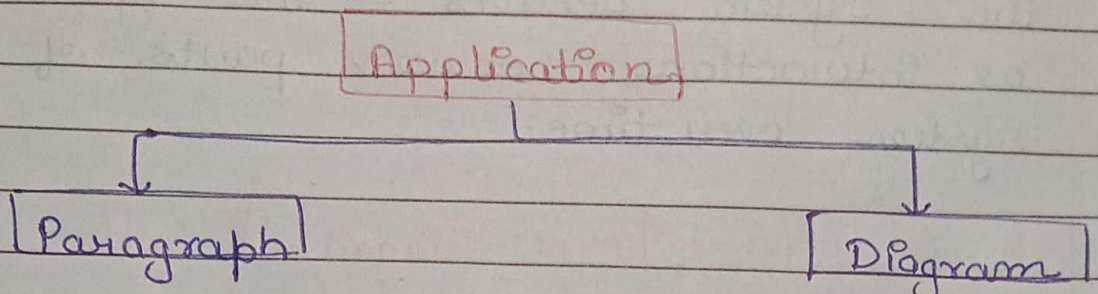
May
15/05/2025
Thursday

DAY-4
(Lecture-4)
UML Diagram | Class & Sequence

It's you vs you

Date
Page

* Introduction To UML Diagram



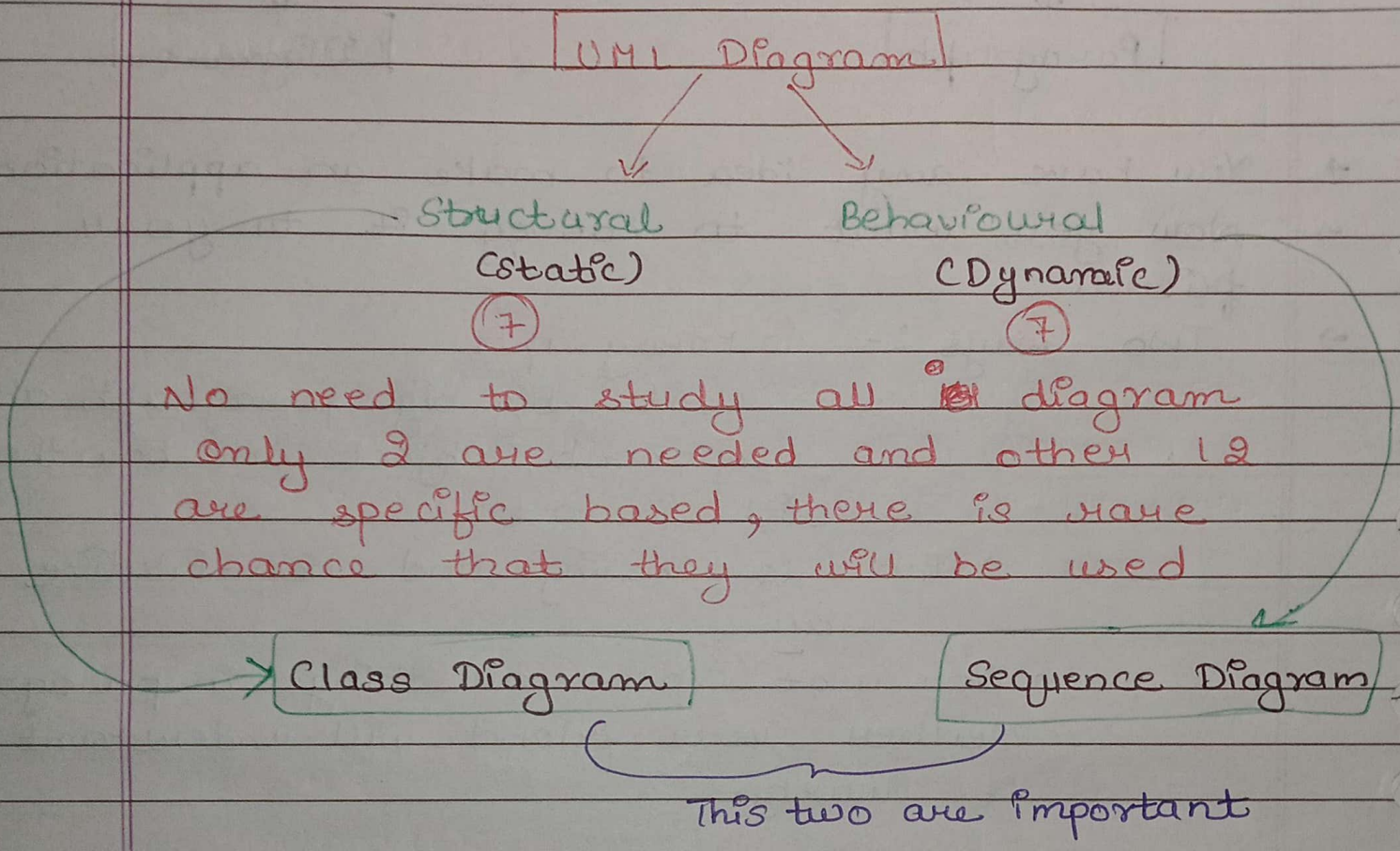
→ You have any idea to make an application
→ Now you want to convey it to your friend.
→ Two ways :- i) Paragraph
→ Write big big paragraph about component, object and how they will interact
→ This is boring and non-intutive way
→ You can't express all things in paragraph Neither your friend will understant it clearly.

ii) Diagram : → Intutive way
→ Easily understandable

UML Diagram :- Diagramatic way to express component, object and how they will interact

Types of UML Diagram : i) Structural Diagram
It shows what a system is made of - the static parts like classes, objects, and how they are connected.

ii) Behavioral Diagram: It shows how the system behaves — the dynamic actions or interactions between parts of the system over time.



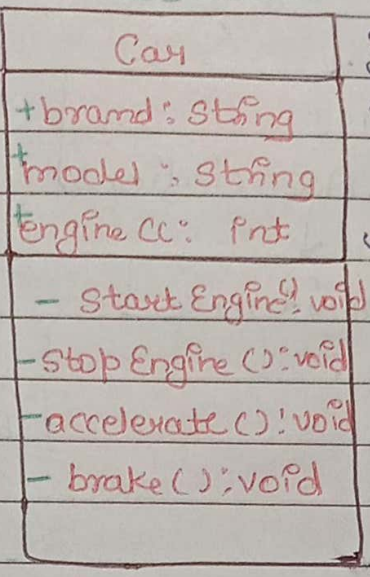
* Class Diagram: i) Class structure
ii) Association / Connection
↳ How different classes will be connected with each other

Some methods are virtual and inherited Class should define it.

Representation of Class

```
Class Car {                          Private
    String brand;
    String model;
    int engineCC;
                                     Public
    startEngine();
    StopEngine();
    accelerate();
    brake();
}
```

void



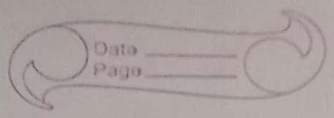
How Access modifier will be represented ?

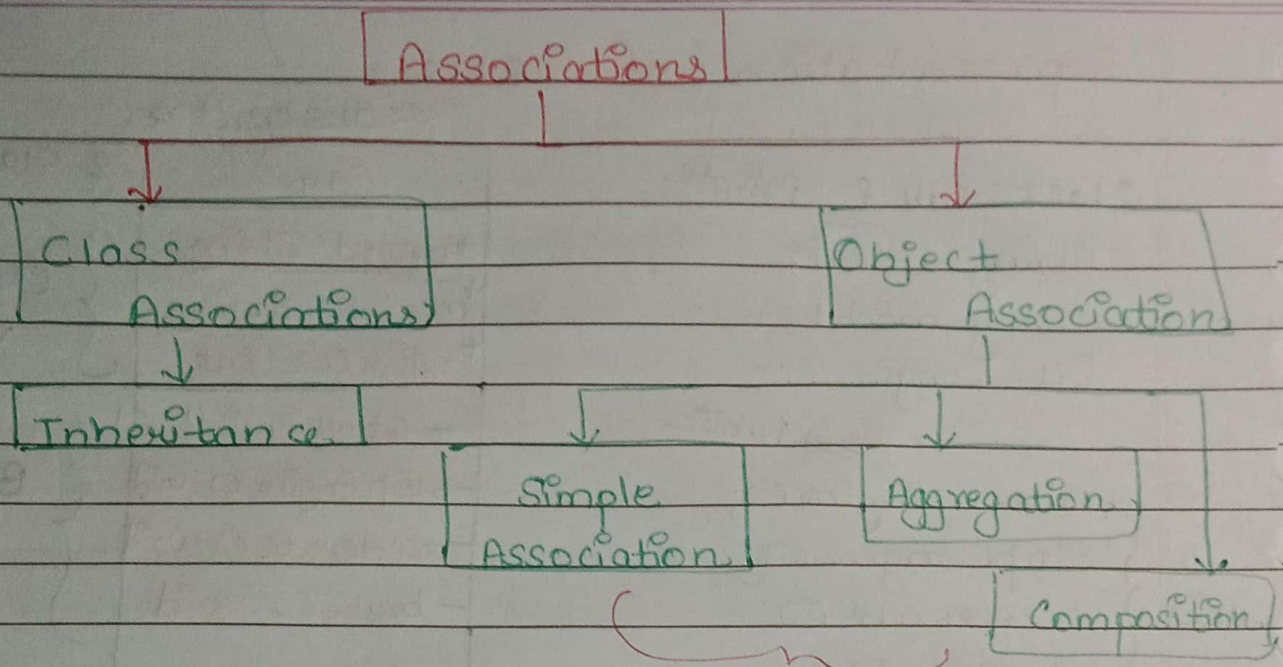
	Within Class	From Child Class	Outside the Class	
Public	✓	✓	✓	→ (+)
Protected	✓	✓	X	→ (#)
Private	✓	X	X	→ (-)

Symbols to represent modifiers

Only abstract will be written above the class structure diagram. No need to write concrete

Associations : One class is depend on other class. How it will interact with each other.





Composition

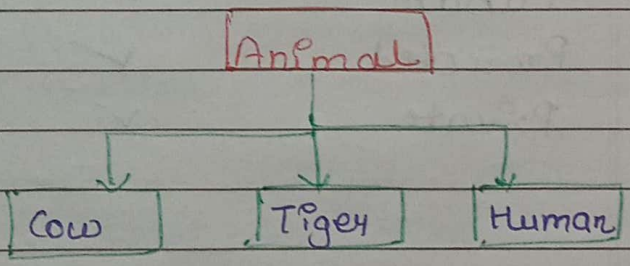
(Combine all known as composition because there is only one way to represent them. Theoretically they are different)

i) Inheritance $\xrightarrow{\text{Known as}}$ (is-a) relationship

```
Class A {
    method1();
}

Class B: PublicA {
    method2();
}

main() {
  B *b = new B();
    b → method1();
    b → method2();
}
```



Cow (is-a) Animal

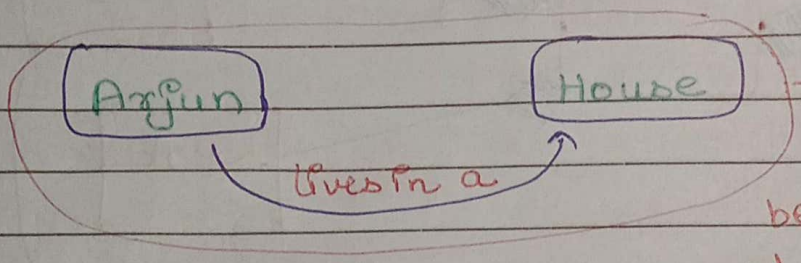
Representation

(→ (is-a))

Inheritance relationship

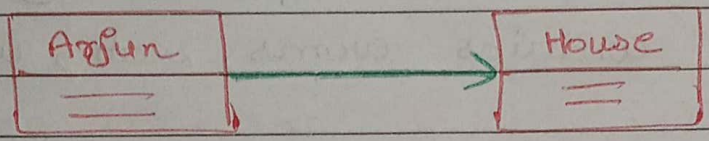
ii) Composition: (has-a) relationship

- Simple Association: Weakest and simple form of relationship



There is no any other relationship between arjun and house. This is simple relationship not a complex one

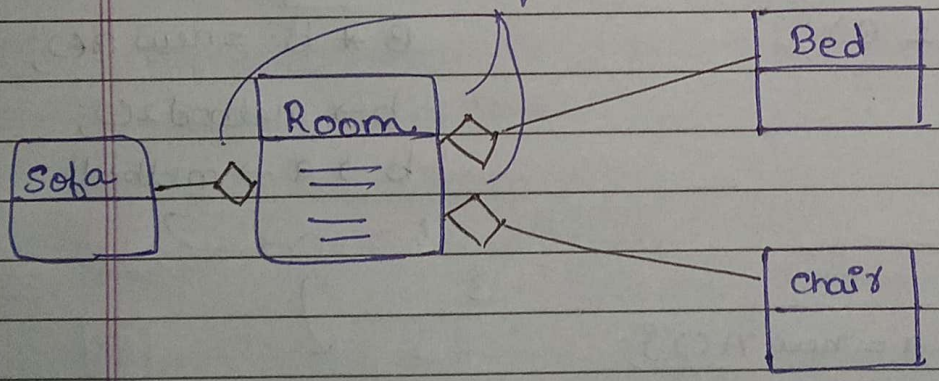
Representation
[→]

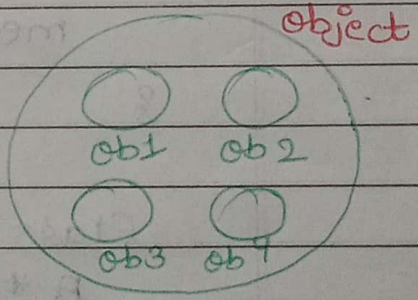


Arjun has-a house

- Aggregation:

we represent it towards the parent object because they are part of it

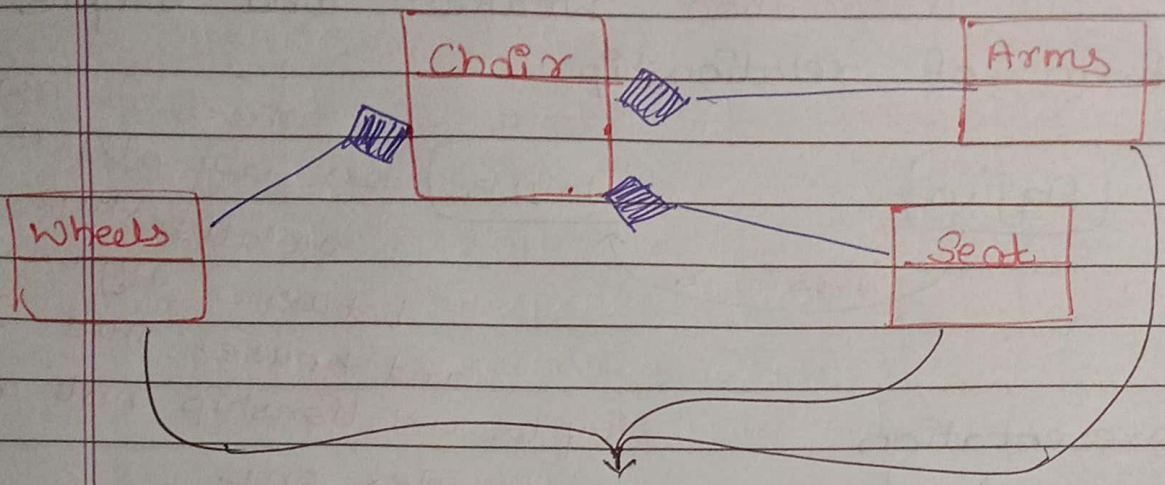




There is one parent object in which small object are present.

Example: In a Room, Bed, chair, sofa all are present inside the room
Sofa, Bed, chair can exist individually

- **Composition**: Strongest form of relationship



They can't exist independently because Chair contains arms, seat, wheels

Programmatic Way to Represent Composition

```
Class A {

    method1 ();

}

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

    method2();

}
```

```
main() {

    B * b = new B();
    b → method2();
    b → a → method1
}
```

We can't directly call method1 from class B. First we access class A object through B object then call method1

Composition is used very much in LLD.

Conclusion of Class Diagram

→ It depend on how we think for composition association.

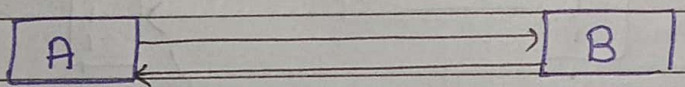
→ If we take restaurants example then

→ If we think menu can exist without restaurants then it is aggregation.

→ If menu can't exist without restaurants then it is aggregation.

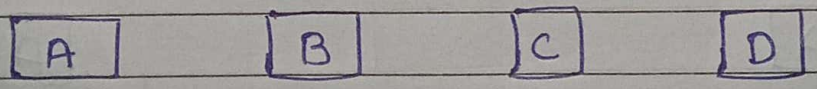
→ There is only thin line difference. There is no right and wrong way to draw diagrams we can draw it in any way.

* Sequence Diagram : (It will not ask in interview but it use in some specific use case)

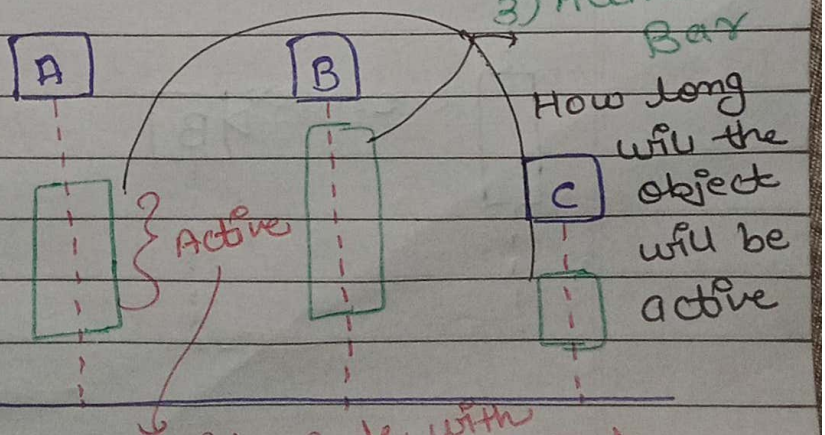


Communication and interaction b/w two objects.

1) Class Representation



2) Lifeline
It tell from where the object will be start and how long it will be exist



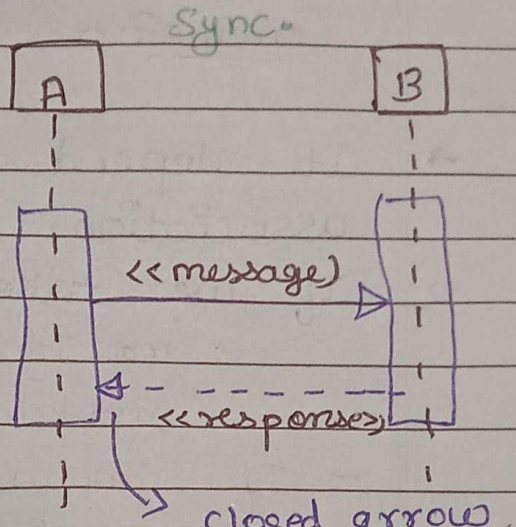
Messages

Async.

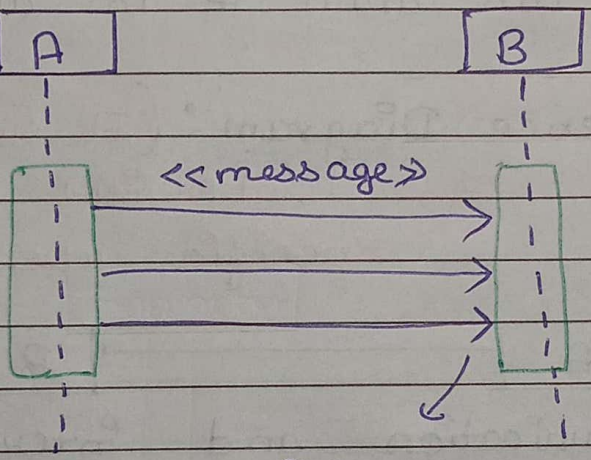
We send msgs one by one and not waiting for response

Sync.

first send msg then wait for response. After receiving response we will again send msg

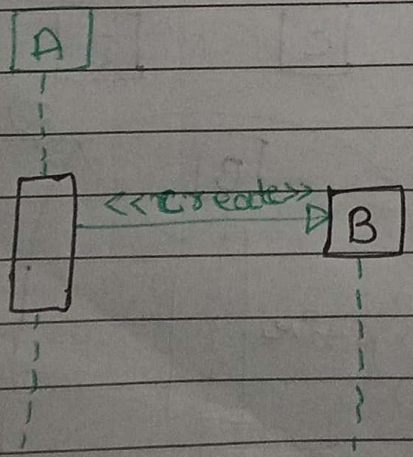


Async.

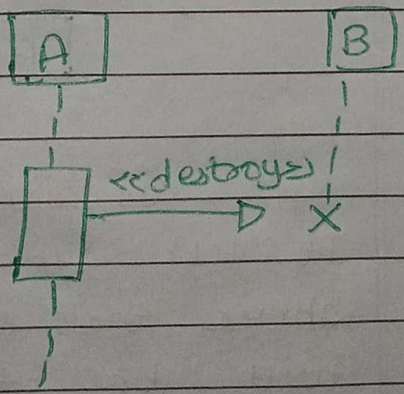


Open arrow to represent

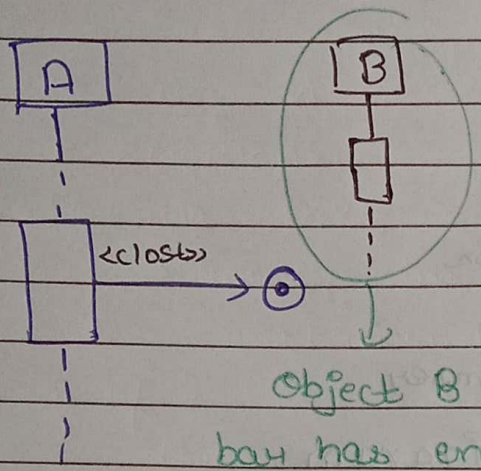
Create Message



Destroy Message

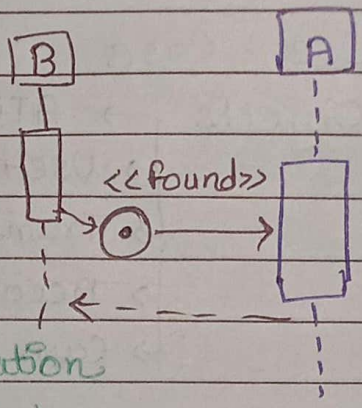


Lost Message



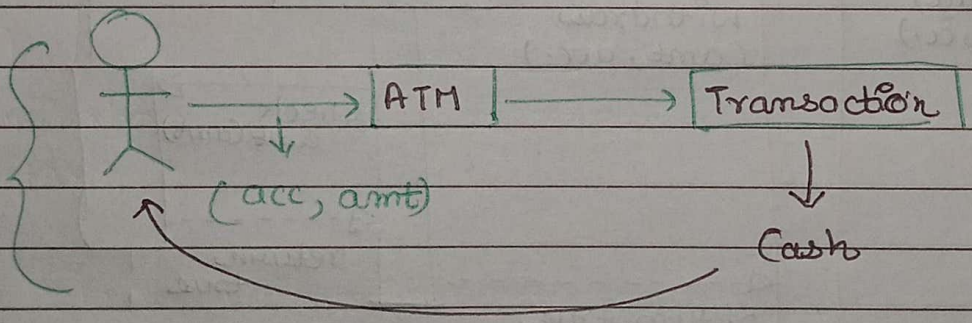
Object B activation bar has ended before receiving the message so message get lost because it never reach to destination.

Found Message



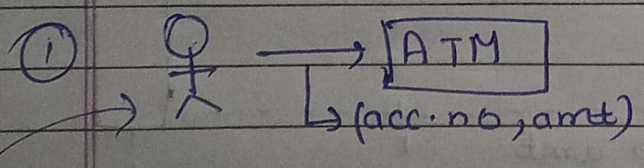
In which object A receive msg but from unknown source. When A wants to send respons but B is inactive so it is known as found message

How to Draw Sequence Diagram

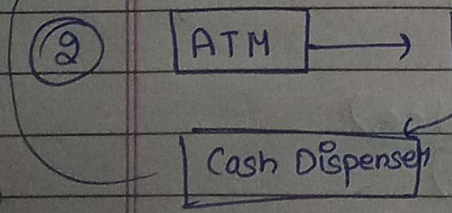


Scenario to draw Sequence Diagram

Use-Case



User go to ATM with account no., amount



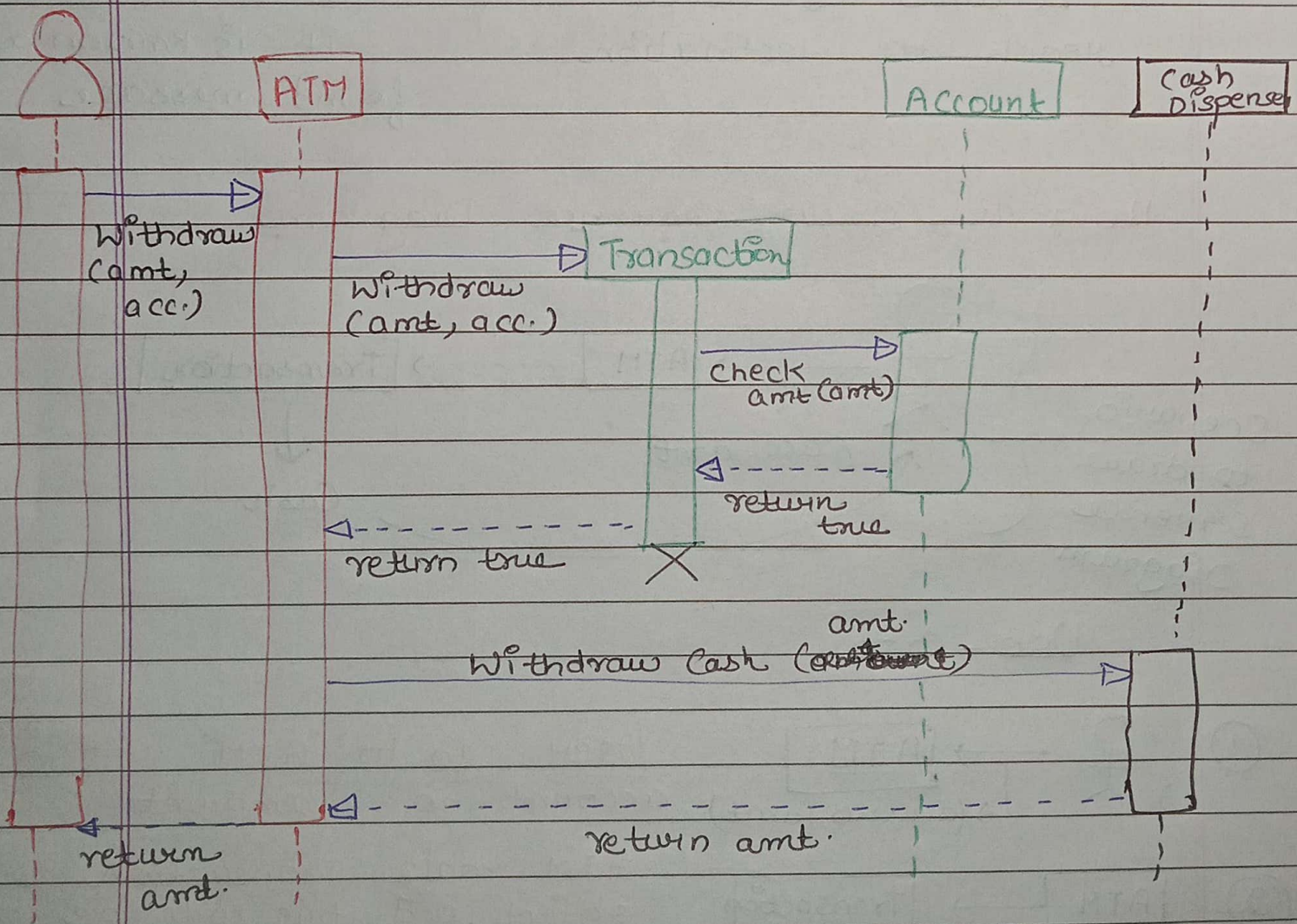
ATM make Transaction object and this will be unique for every transactions.

① Verify ATM Pin
② verify acc
③ Cash Dispenser

Transaction object verify details and send to Cash Dispenser C.D give cash to user

Sequence Diagram flow

1. Use - Case
2. Objects
   - ATM
   - User
   - Transaction
   - Account
   - Cash Dispenser
3. Draw sequence Diagram



Terms: alt : (if - else)
option : (if)
loop : for / while