UML Diagram

⊛ What is UML Diagram ?

→ Diagramatic explaination of components inside an application and their interactions.

⊛ Types of UML Diagrams

Structural (Static)

Ye batate hai app me konse konse components honge and how these components r connected to each other

Behaivioural (Dynamic)

Ye batate hai ki apne app me jo components hai vo aapas me interact kaise karenge

→ There are 7 structural UML diagrams & 7 Behavioural UML diagrams.

Total = 14

But we just have to learn "2" [all LLD is covered in this]

→ We will learn.

① Class Diagram → Structural UML

② Sequence Diagram → Behavioural UML

Class Diagram

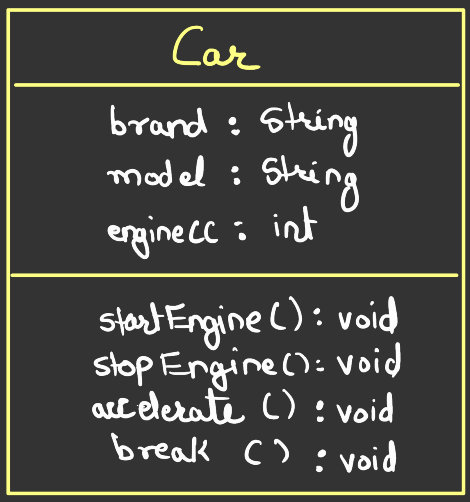
→ Tells about classes & their connection with each other.

① Class Structure ② Association/Connection

Class Representation :→

```
class Car {
    String brand;
    String model;
    int engineCC;
    void startEngine();
    void stopEngine();
    void accelerate();
    void break();
}
```

→ Private
→ Protected
→ Public



} class name
class variables
class methods/ behaiviours

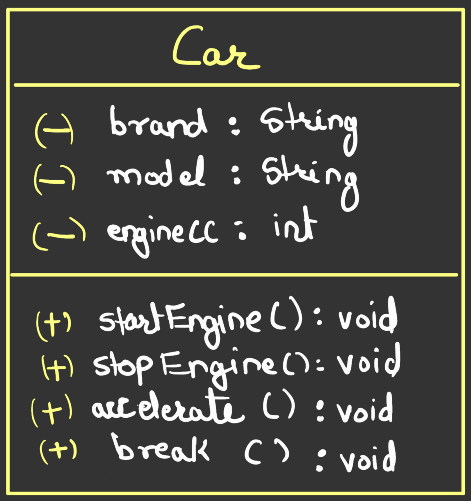
Access Modifiers → Public, Protected, default, Private

Access Modifiers

Modifier	Class	Package	Subclass	Global
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

→ (+)
→ (#)
→ (+)
→ (−)

default methods are implicitly public



} class name

class variables

class methods/ behaiviours

Now, there are Two Types of classes.

1. Abstract
2. Concrete

Abstract

<<abstract>>

Car
(-) brand : String (-) model : String (-) engineCC : int
(+) startEngine() : void (+) stopEngine() : void (+) accelerate() : void (+) break() : void

Concrete

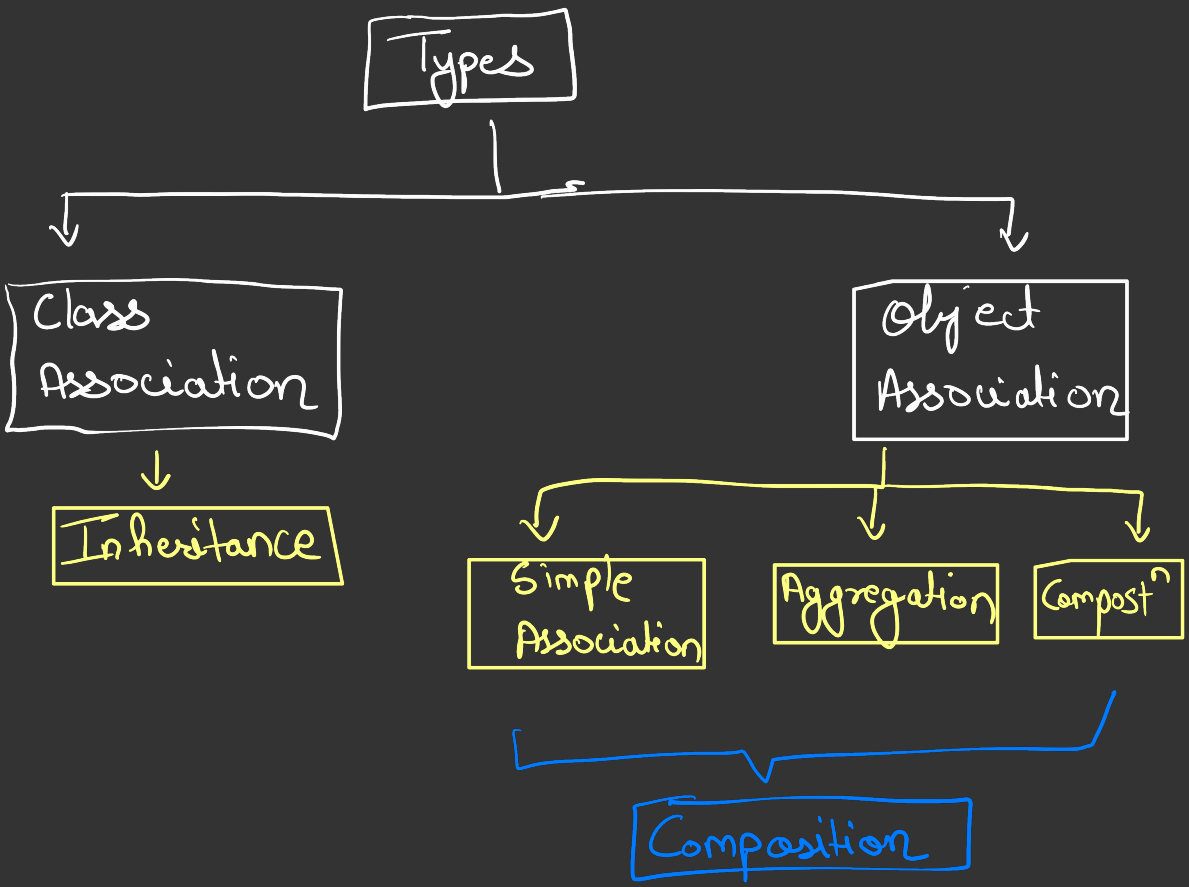
Car
(-) brand : String (-) model : String (-) engineCC : int
(+) startEngine() : void (+) stopEngine() : void (+) accelerate() : void (+) break() : void

Summary

<<abstract>>

ClassName
(+) var1 : datatype (-) var2 : datatype
(+) m1() : datatype (+) m2() : datatype

⓯ Associations



These are theoritically 3 different but Programatically there is only one way to represent them.

Hence we say "Composition" only -

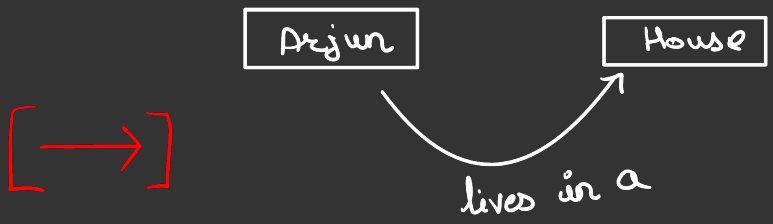
⊙ **Inheritance** ⇝ (IS-A) relationship [—▷]

```
class A {
    m1();
}

class B extends A {
    m2();
}

main() {
    B b = new B();
    b.m1();
    b.m2();
}
```

⊙ **Composition** ⇝ (HAS-A) relationship

> Jab bhi kisi relationship me "has-a" dal sake then it is Composition

⊙ Simple Association



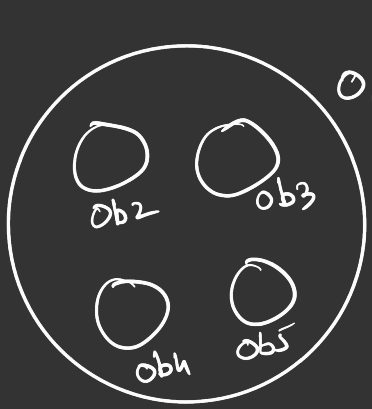
[→]

⊙ Aggregation

⊙ Now Sofa, Bed & Chair related to Room?

→ Parts of Room.

(#) Aggregation relationship me ek object hota hai jisme alag alag chote objects hote hai.



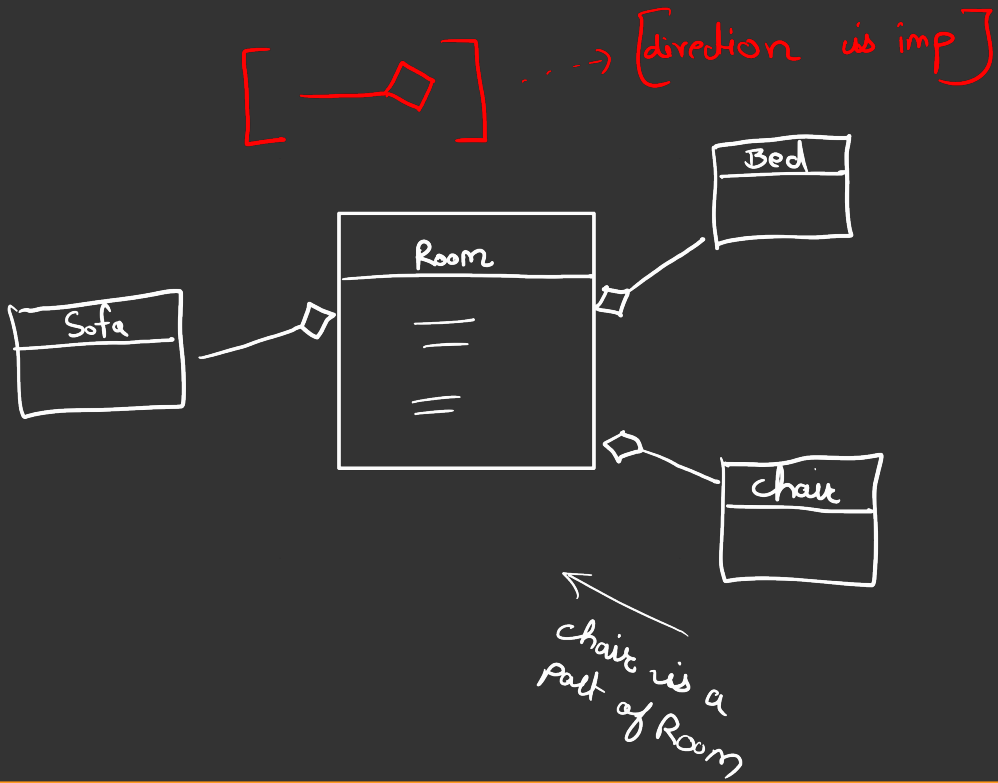
Here Room is Aggregator / Container

Aggregation is stronger than simple relationship but weaker that 3rd type composition.

* What is stronger?

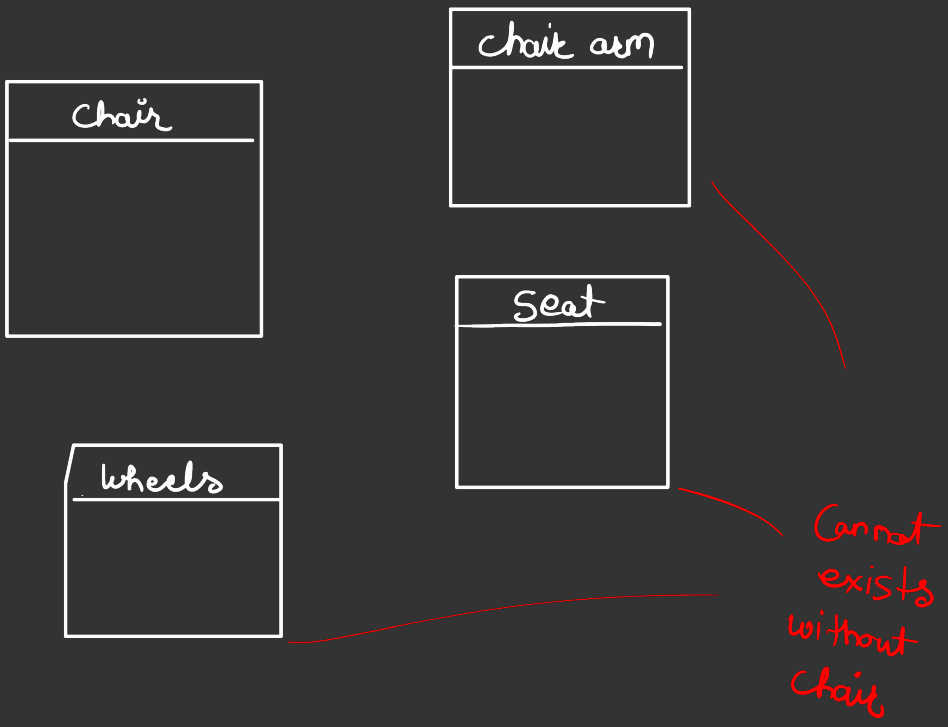
→ More complex relationship.

* Representation

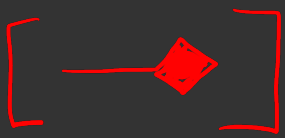


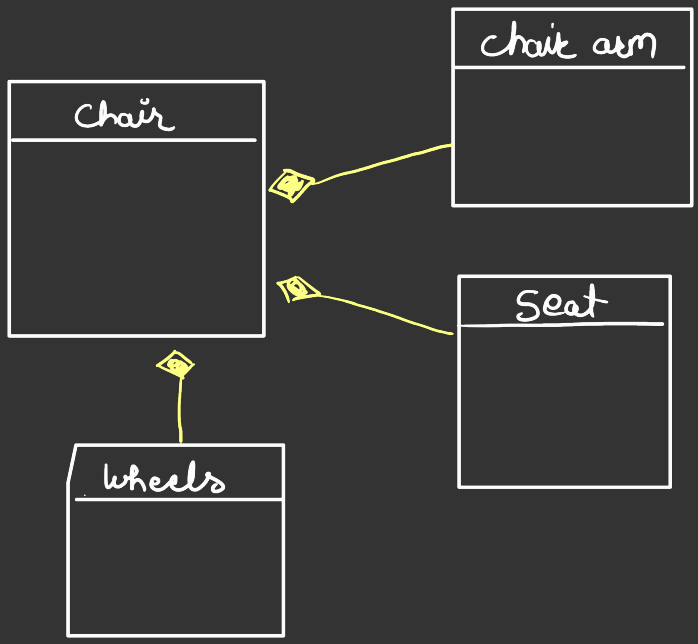
Note :- Here Sofa, Bed, Chair are part of Room but they can exist without Room as an entity.

Composition :→ [Strongest relⁿ betⁿ two objects]



Represented By :→





⊙ Code Representation of Composition → Programatic Repres^n

```
class A {
    m1();
}
```

```
main () {
    B b = new B();
    b.m2();
    b.a.m1();
}
```

⇒ VV Imp

```
class B {
    A a;

    B() {
        a = new A();
    }
    m2();
}
```

Exercise

Q Take a Car with 3 properties. There can be two types of Cars

- Manual Car
- Electric Car

Manual Car has changeGear() fun^c but not in electric Car.

Electric Car has chargeBattery() fun^c but not in manual Car.

✳ Sequence Diagram

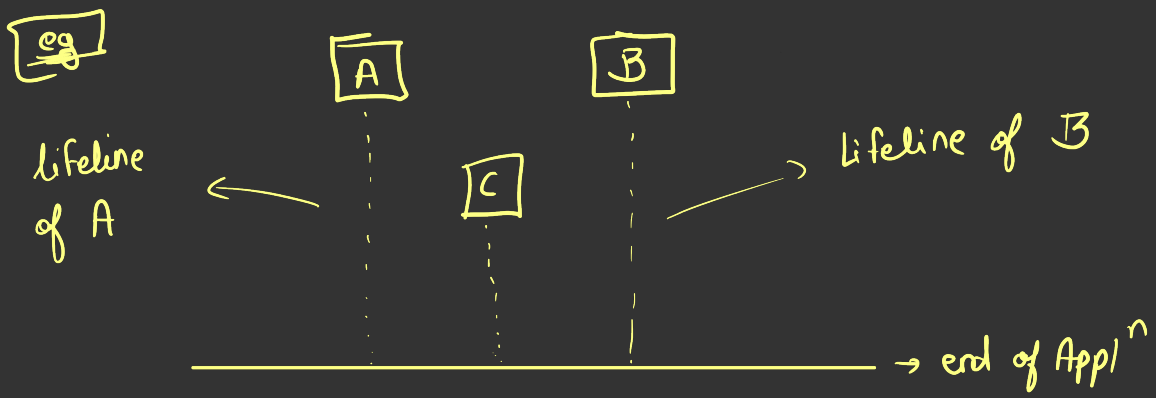
⊙ Representation

---

① Class

---

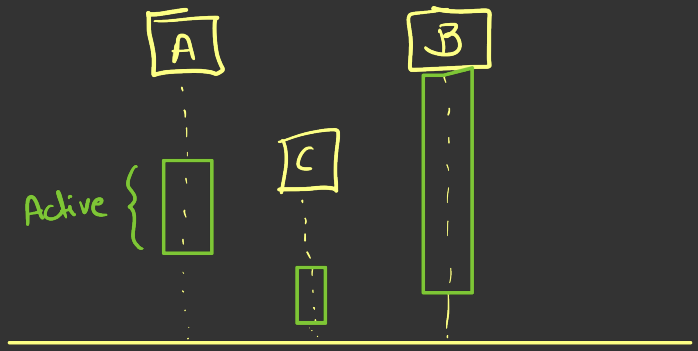
② Lifeline :→ Ye batata hai ki, koi bhi ek object/entity kab kab exist karegi humare pure application me.



Above diagram shows A & B will exists till end of application unlike C.

---

③ **Activation Bar** :→ Ye batata hai ki ek object kab tak active rahega app me.



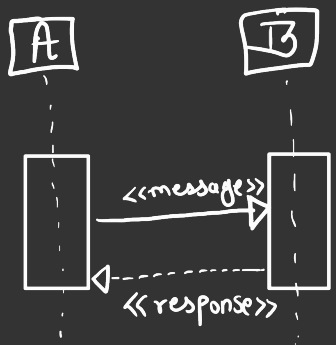
Active :→ Is state me object send or Receive karega message from different objects.

Inactive :→ Won't be able to send or receive message.

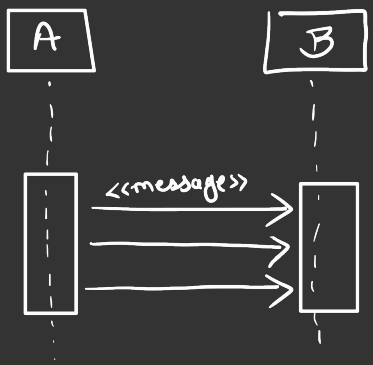
---

④ Messages

- Async :→ we send message w/o waiting for response
- Sync :→ we send message & wait for response



Sync Message

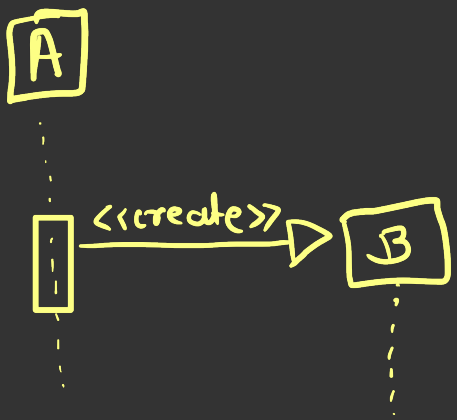


Async Message

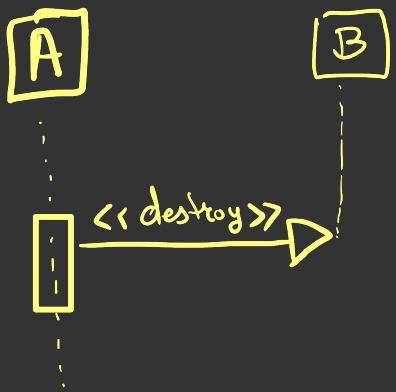
Note :→ Message & Response should be sent or received within Activation Bar.

⑤ Types of Message

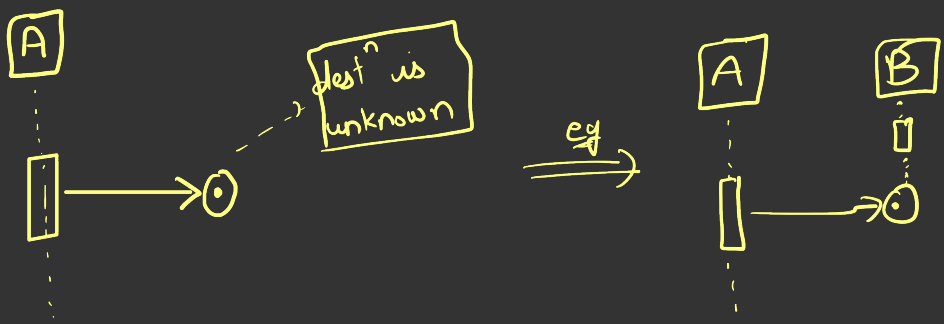
① Create Message :→ Message to create a Component



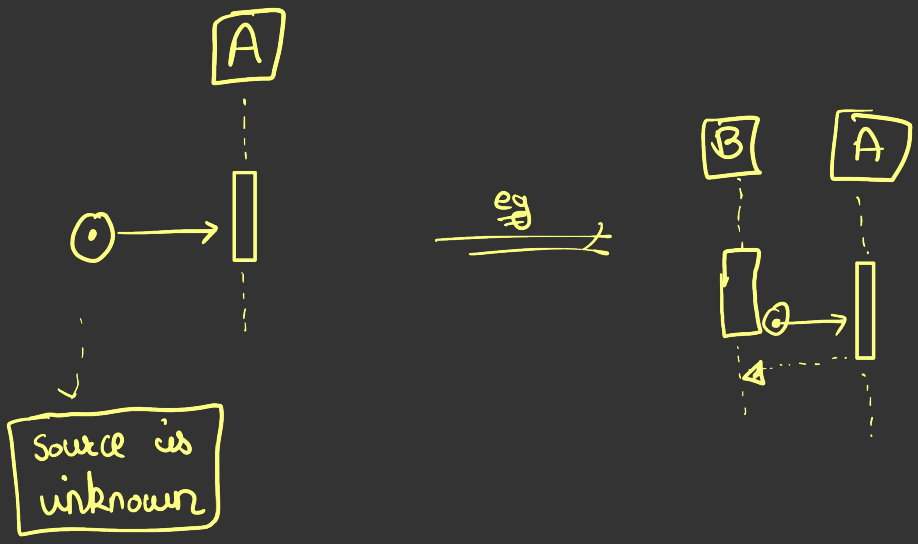
② Destroy Message :→ Message to destroy a Component.



③ Lost Message



④ Found Message



(#) How to draw Sequence Diagram ?

Use Case :- User goes to ATM to get Cash.

Ans :-

Step 1 :- Write down Flow of use case

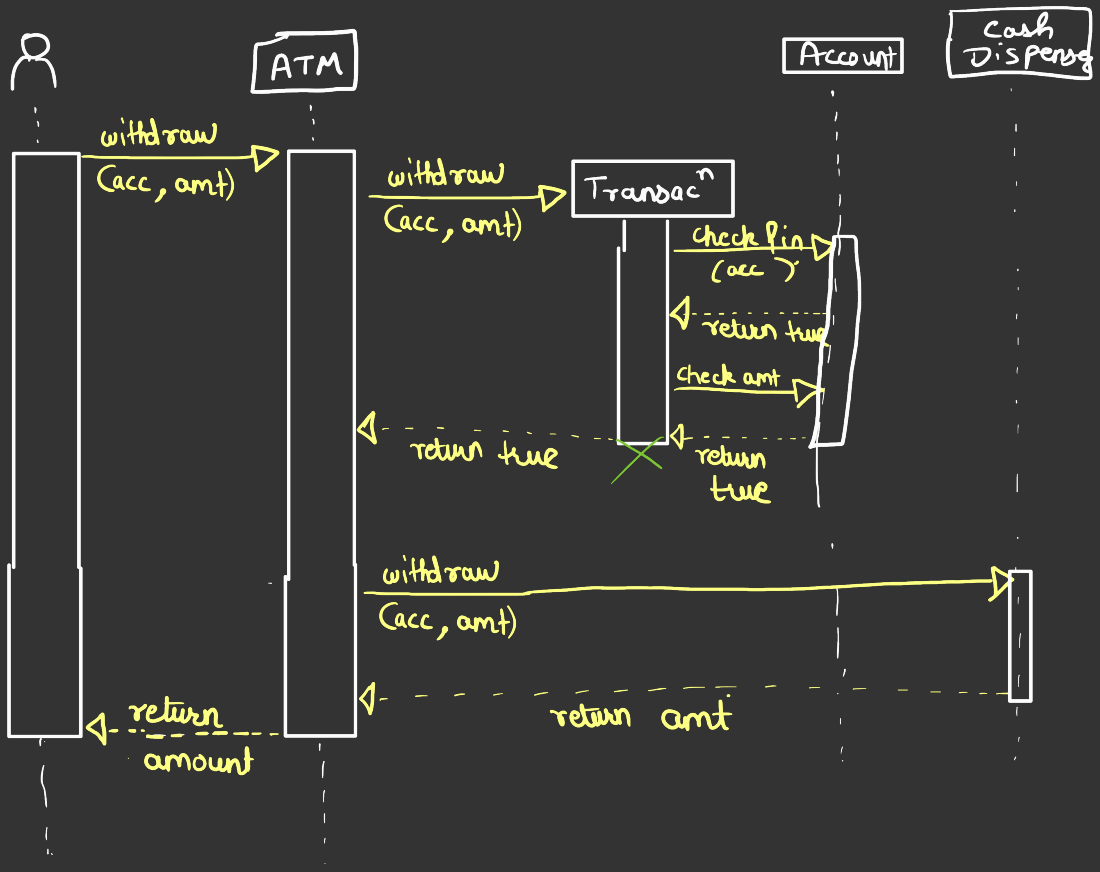
1. User goes to ATM with Acc no & Amt.
2. ATM creates a Transaction.
3. Transaction
   - 1. Verify Pin
   - 2. Verify Sufficient funds present?
   - 3. Invoke Cash dispenser which will give cash.
4. Cash Dispenser will give cash to User.

Step 2 :-

Identify objects :-

1. ATM
2. User
3. Transaction
4. Account
5. Cash Dispenser.

Step 3 :→ Draw Seq Diagram.



Few more sequence diagram terms before wrap-up.

① alt → Denotes (If-else)

② option → Denotes (if)

③ Loop : → for/while.