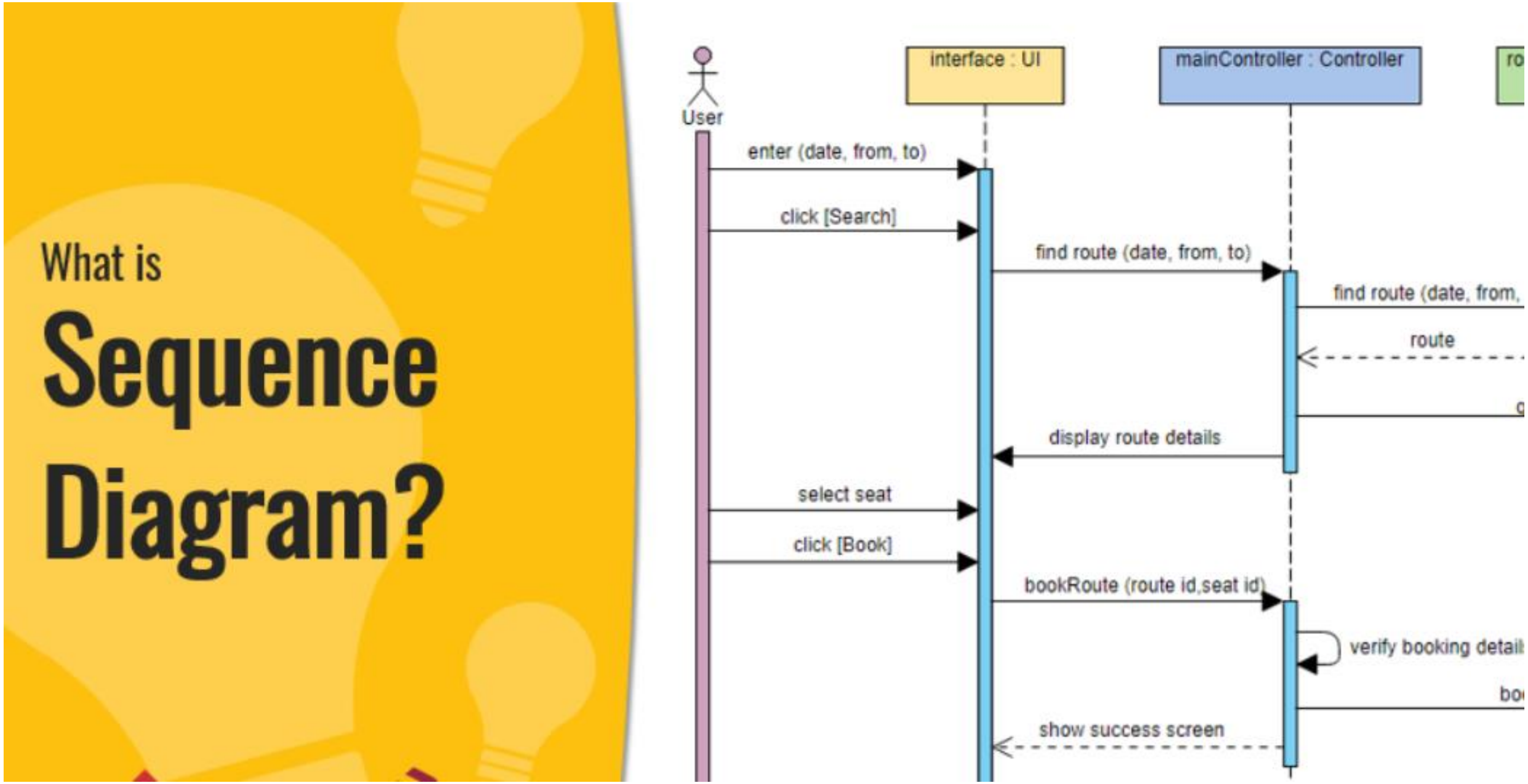
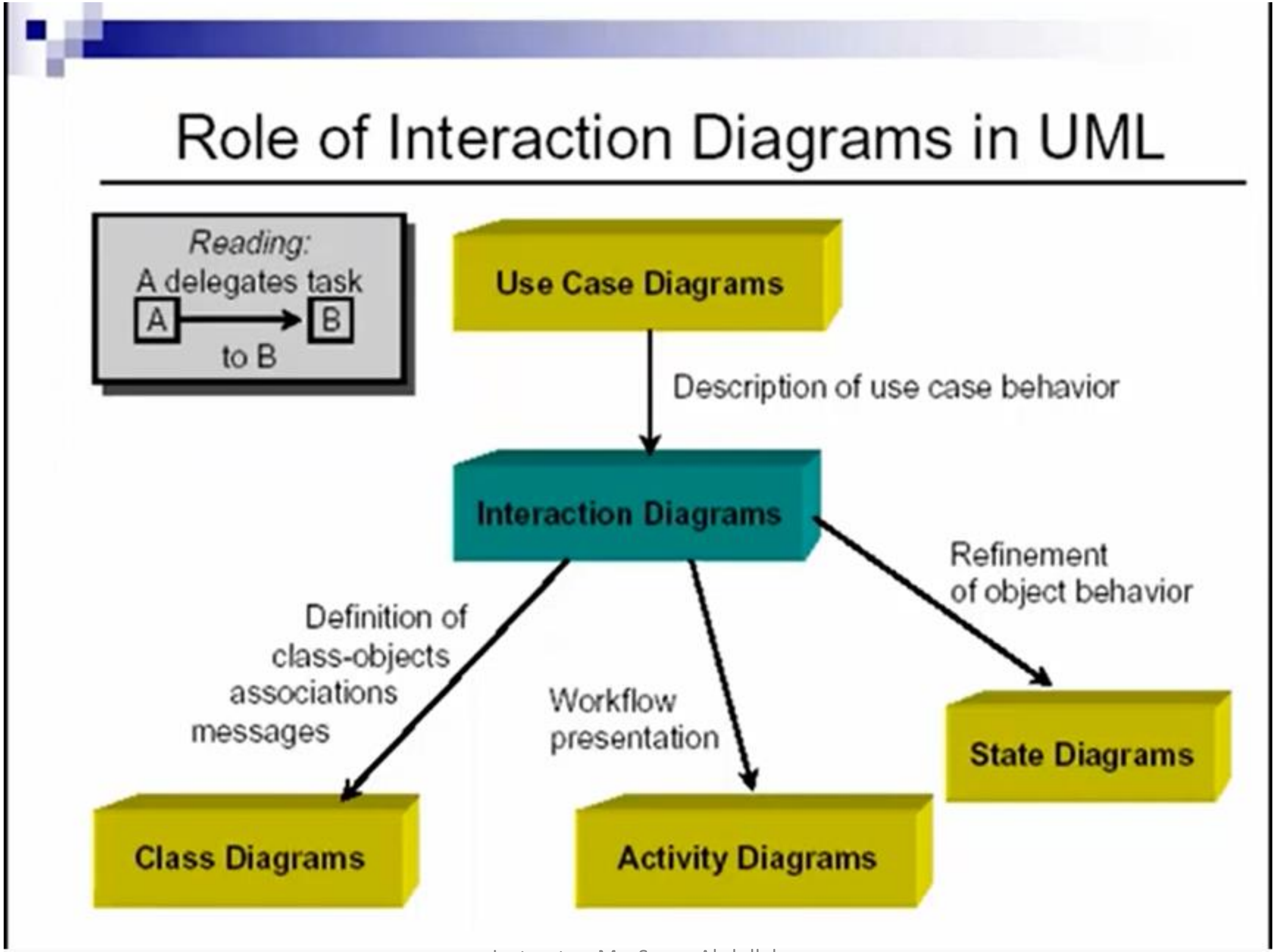

What is
Sequence
Diagram?
User
interface : UI
mainController : Controller
ro
enter (date, from, to)
click [Search]
find route (date, from, to)
find route (date, from,
route
display route details
select seat
click [Book]
bookRoute (route id,seat id)
verify booking detail
bo
show success screen



Role of Interaction Diagrams in UML

Reading:
A delegates task
A → B
to B

Use Case Diagrams

Description of use case behavior

Interaction Diagrams

Definition of
class-objects
associations
messages

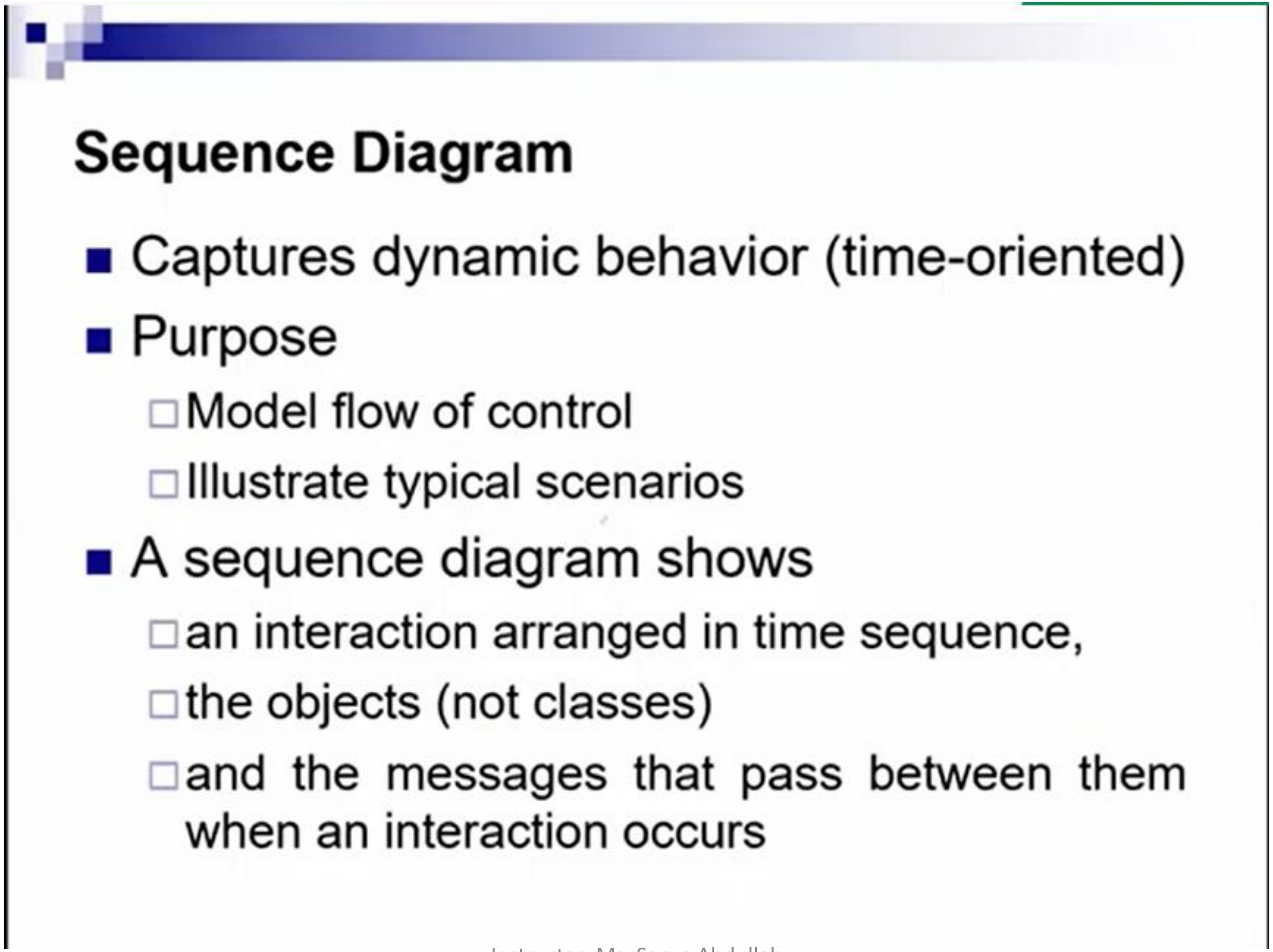
Workflow
presentation

Refinement
of object behavior

Class Diagrams

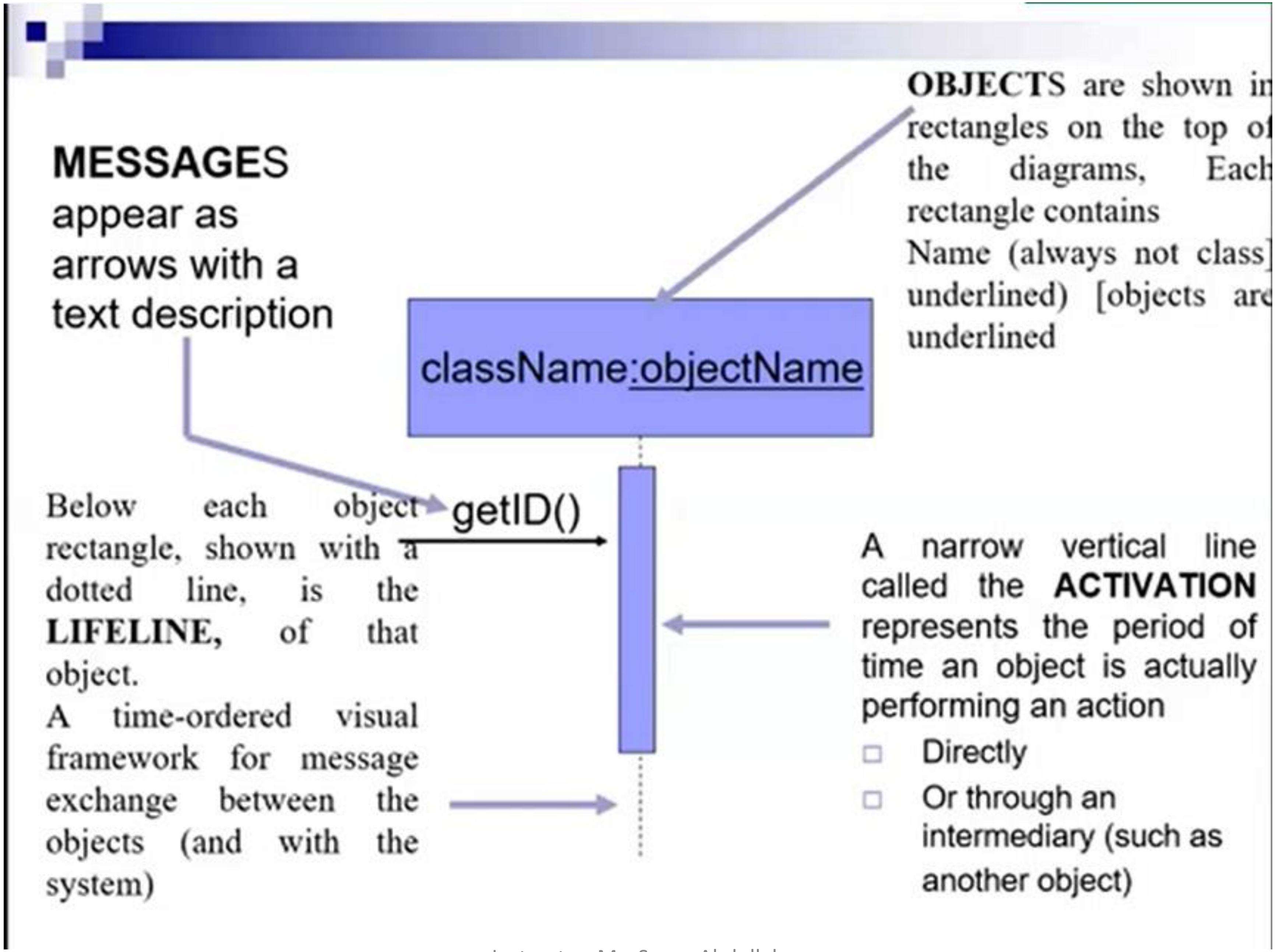
Activity Diagrams

State Diagrams



Sequence Diagram

- Captures dynamic behavior (time-oriented)
- Purpose
  - □ Model flow of control
  - □ Illustrate typical scenarios
- A sequence diagram shows
  - □ an interaction arranged in time sequence,
  - □ the objects (not classes)
  - □ and the messages that pass between them when an interaction occurs



MESSAGES appear as arrows with a text description

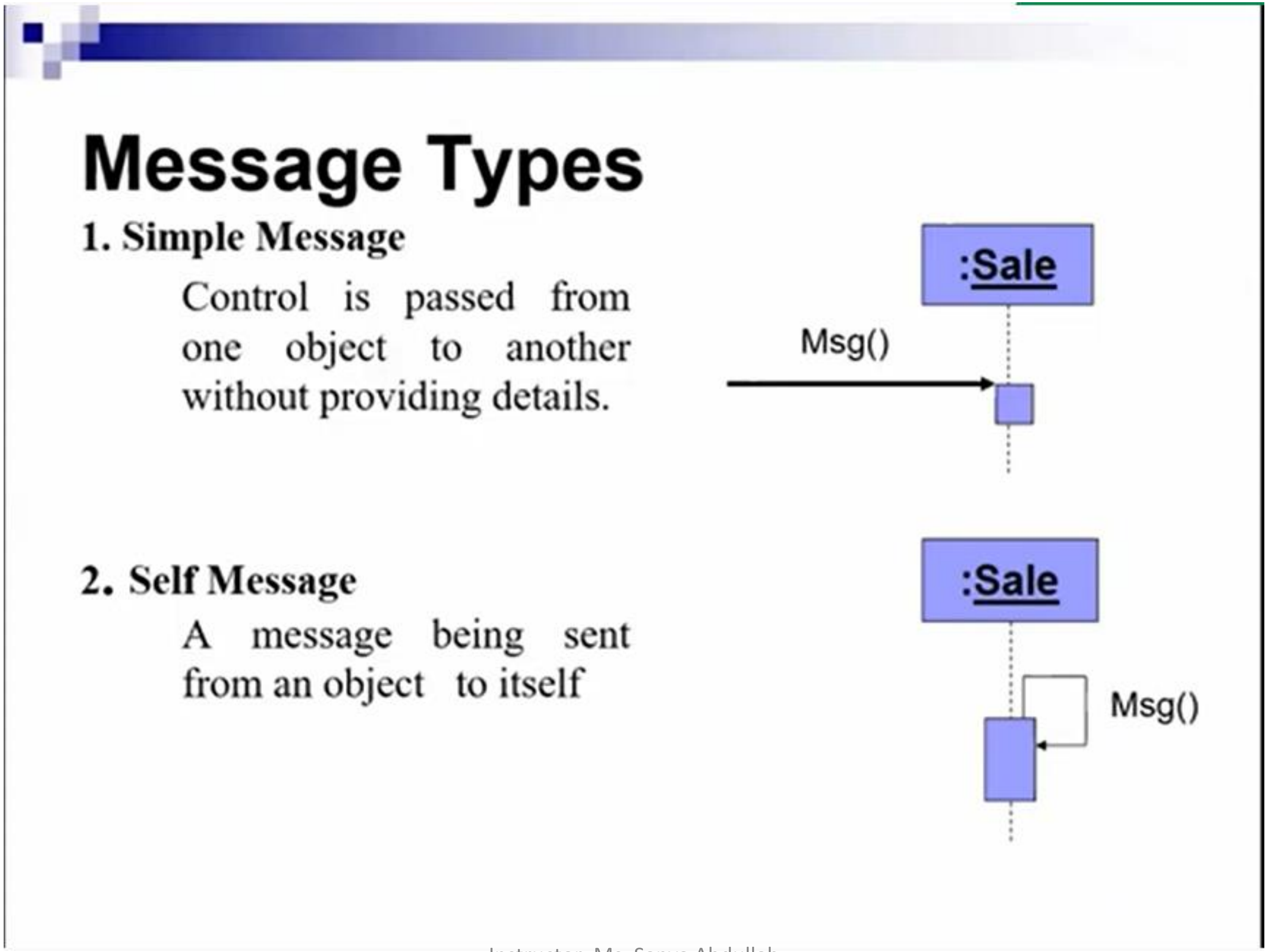
OBJECTS are shown in rectangles on the top of the diagrams, Each rectangle contains Name (always not class] underlined) [objects are underlined

Below each object rectangle, shown with a dotted line, is the **LIFELINE,** of that object.

A time-ordered visual framework for message exchange between the objects (and with the system)

A narrow vertical line called the **ACTIVATION** represents the period of time an object is actually performing an action

- □ Directly
- □ Or through an intermediary (such as another object)



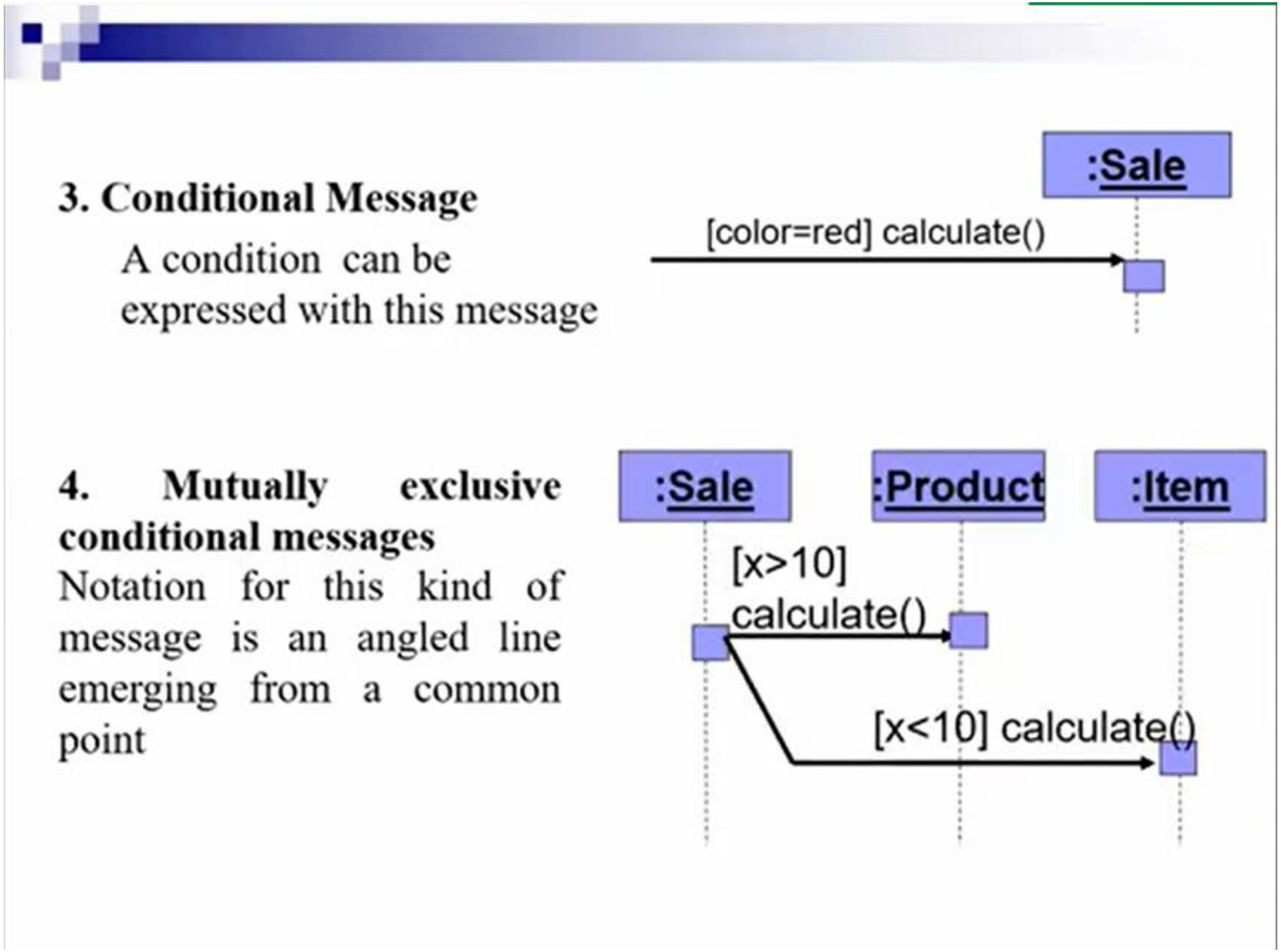
Message Types

1. Simple Message

Control is passed from one object to another without providing details.

2. Self Message

A message being sent from an object to itself



3. Conditional Message

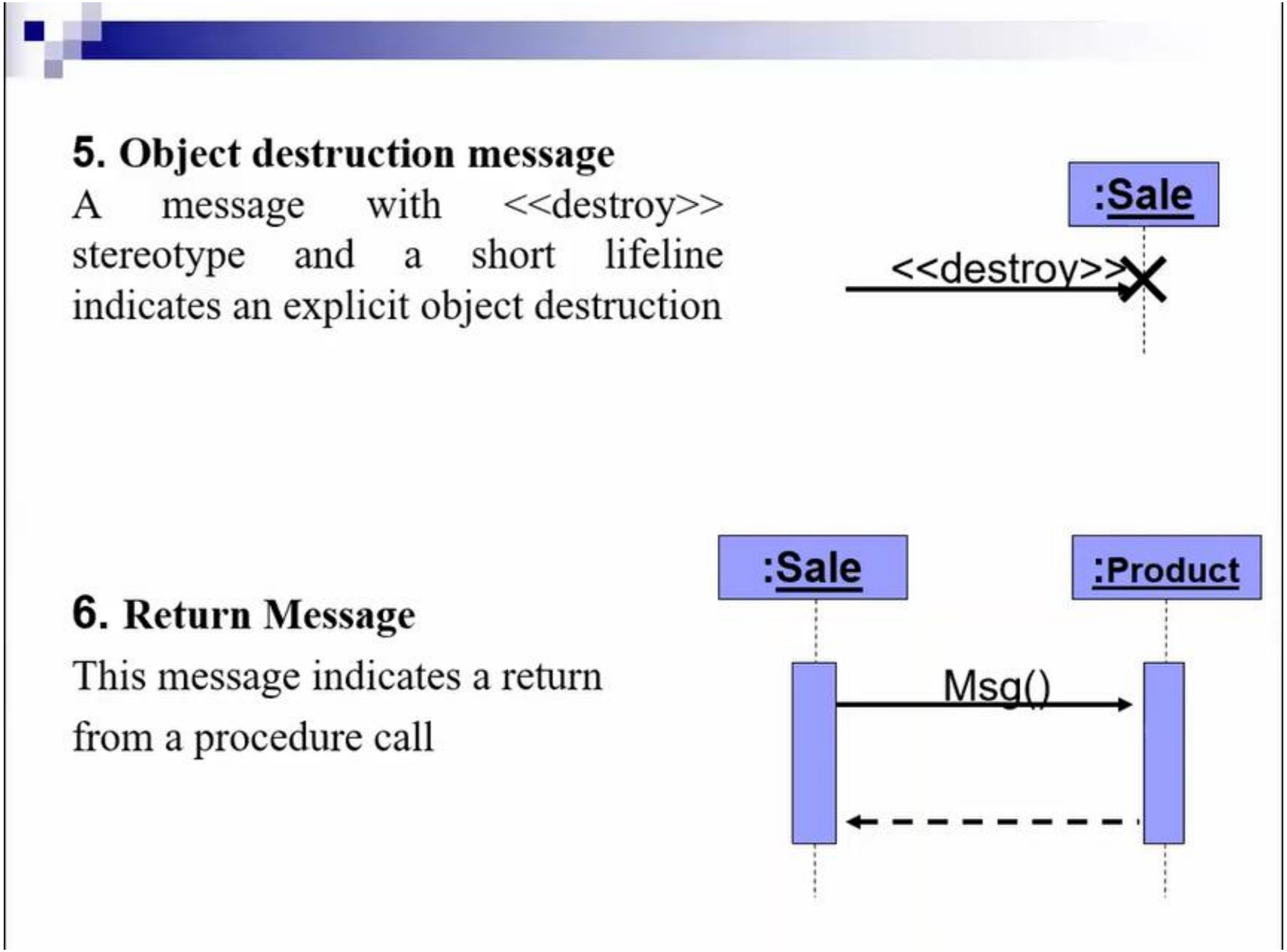
A condition can be expressed with this message

:Sale
[color=red] calculate()

4. Mutually exclusive conditional messages

Notation for this kind of message is an angled line emerging from a common point

:Sale
:Product
:Item
[x>10] calculate()
[x<10] calculate()

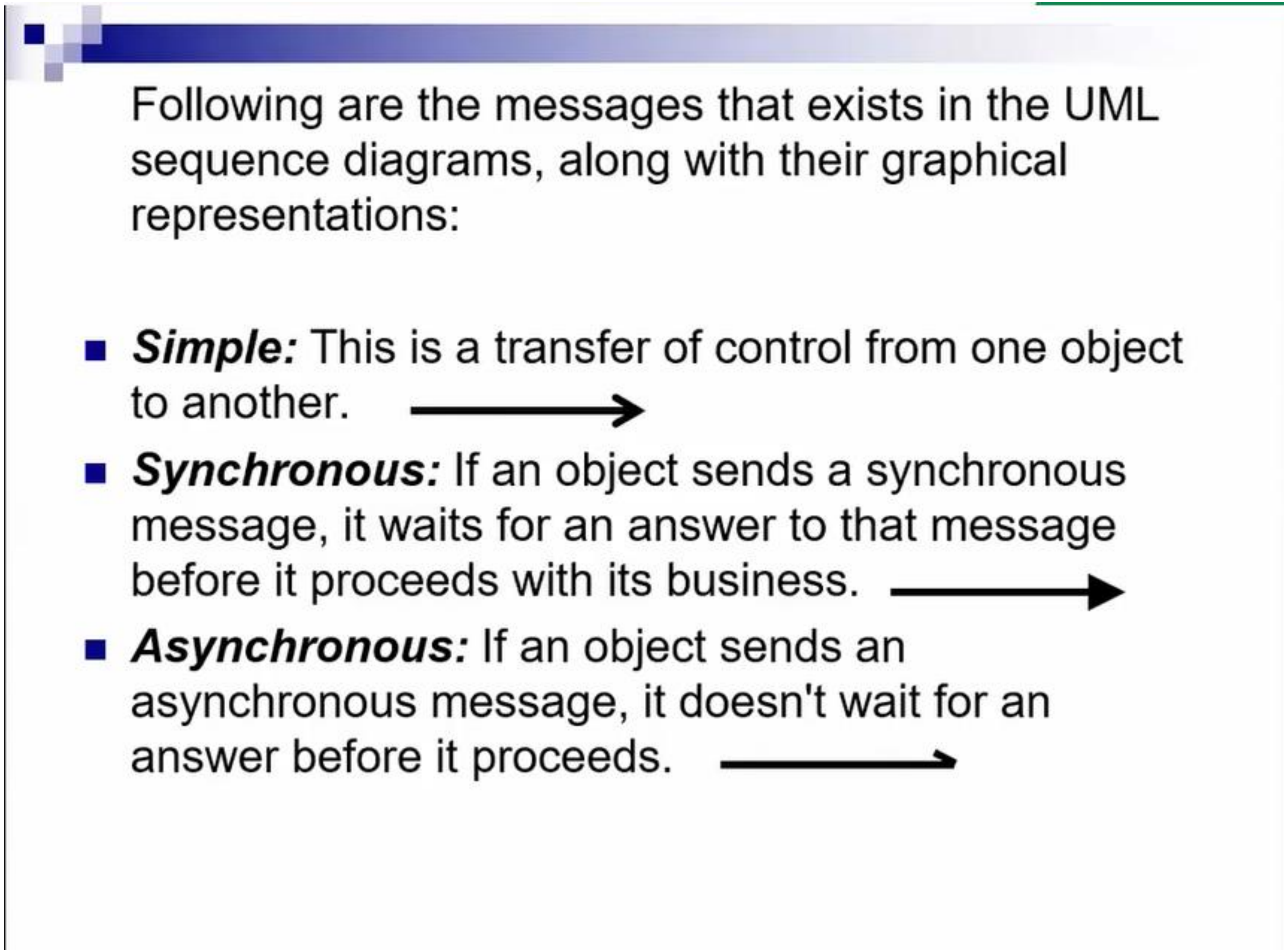


5. Object destruction message

A message with <<destroy>> stereotype and a short lifeline indicates an explicit object destruction

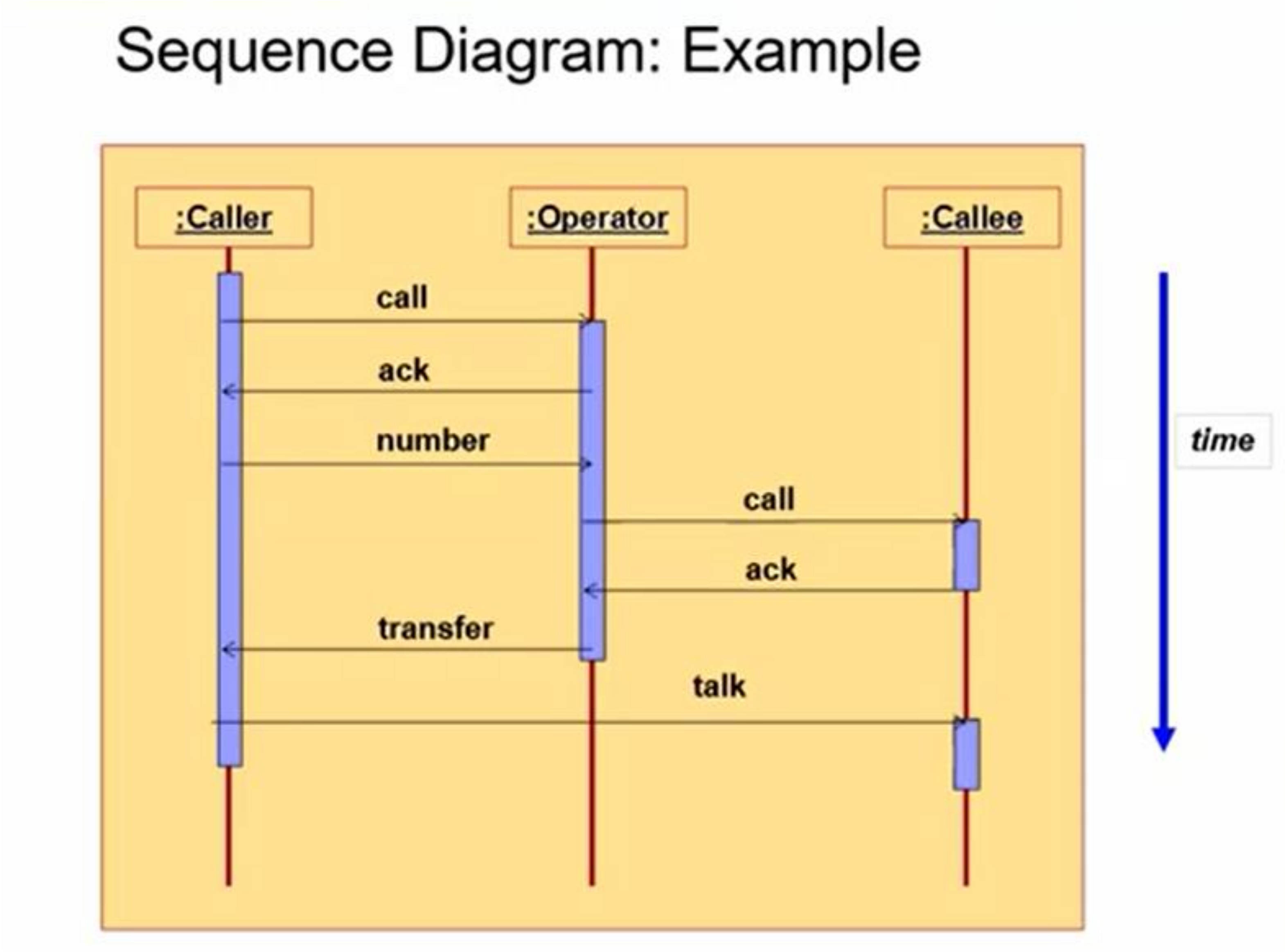
6. Return Message

This message indicates a return from a procedure call



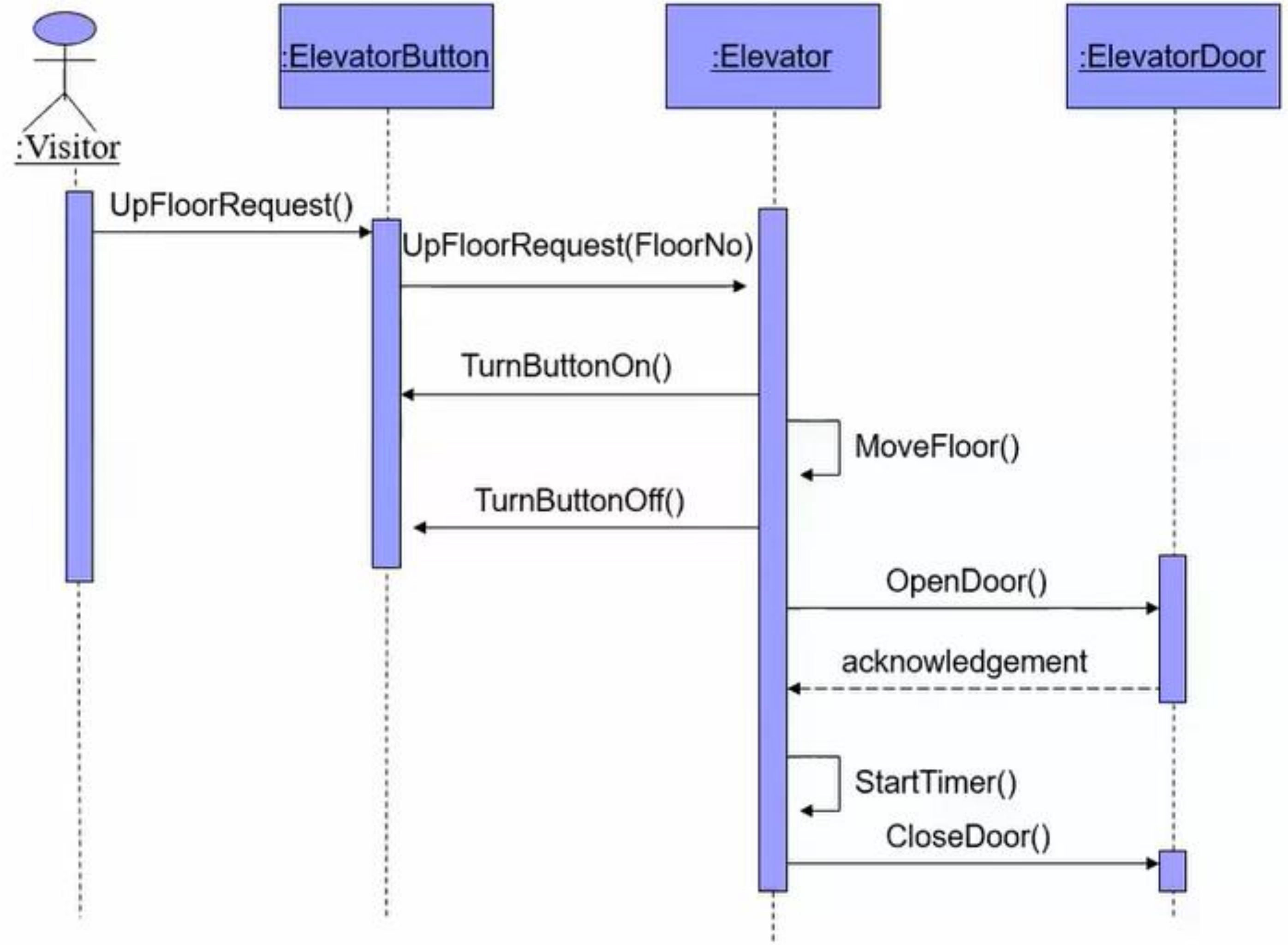
Following are the messages that exists in the UML sequence diagrams, along with their graphical representations:

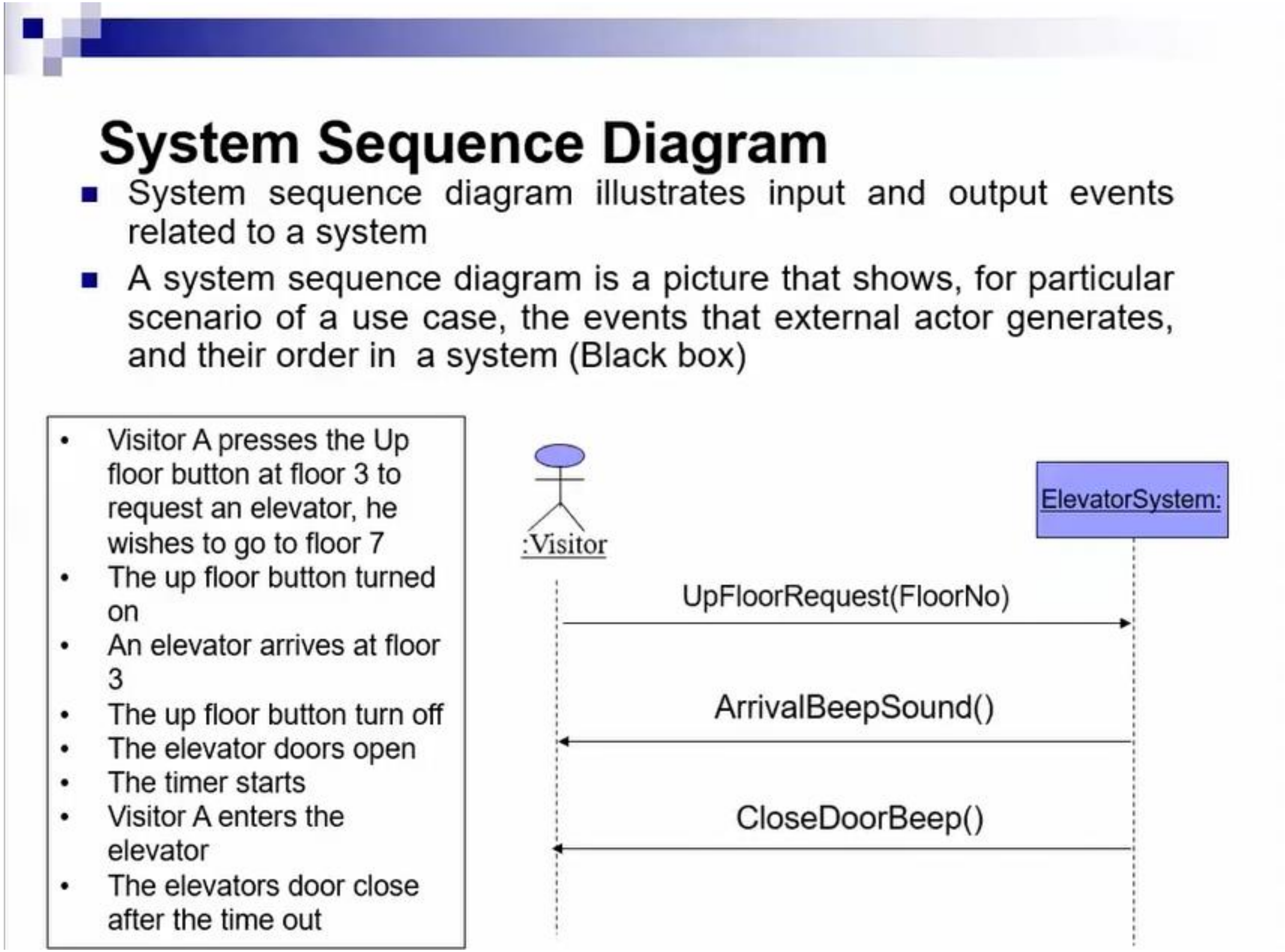
- ***Simple:*** This is a transfer of control from one object to another. →
- ***Synchronous:*** If an object sends a synchronous message, it waits for an answer to that message before it proceeds with its business. →
- ***Asynchronous:*** If an object sends an asynchronous message, it doesn't wait for an answer before it proceeds. →



Sequence Diagram: Example

<<UC-1>>Start Elevator Sequence Diagram





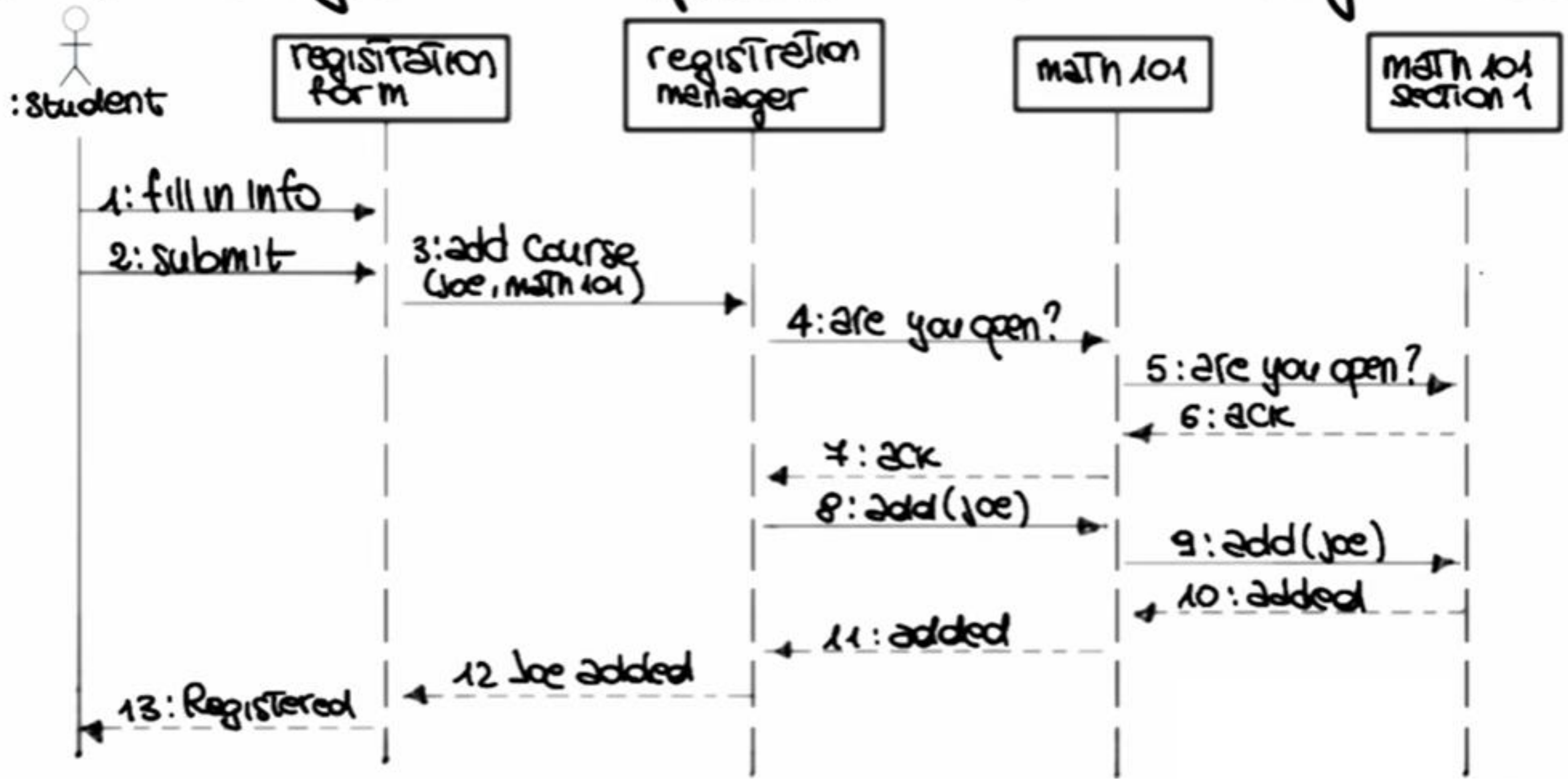
System Sequence Diagram

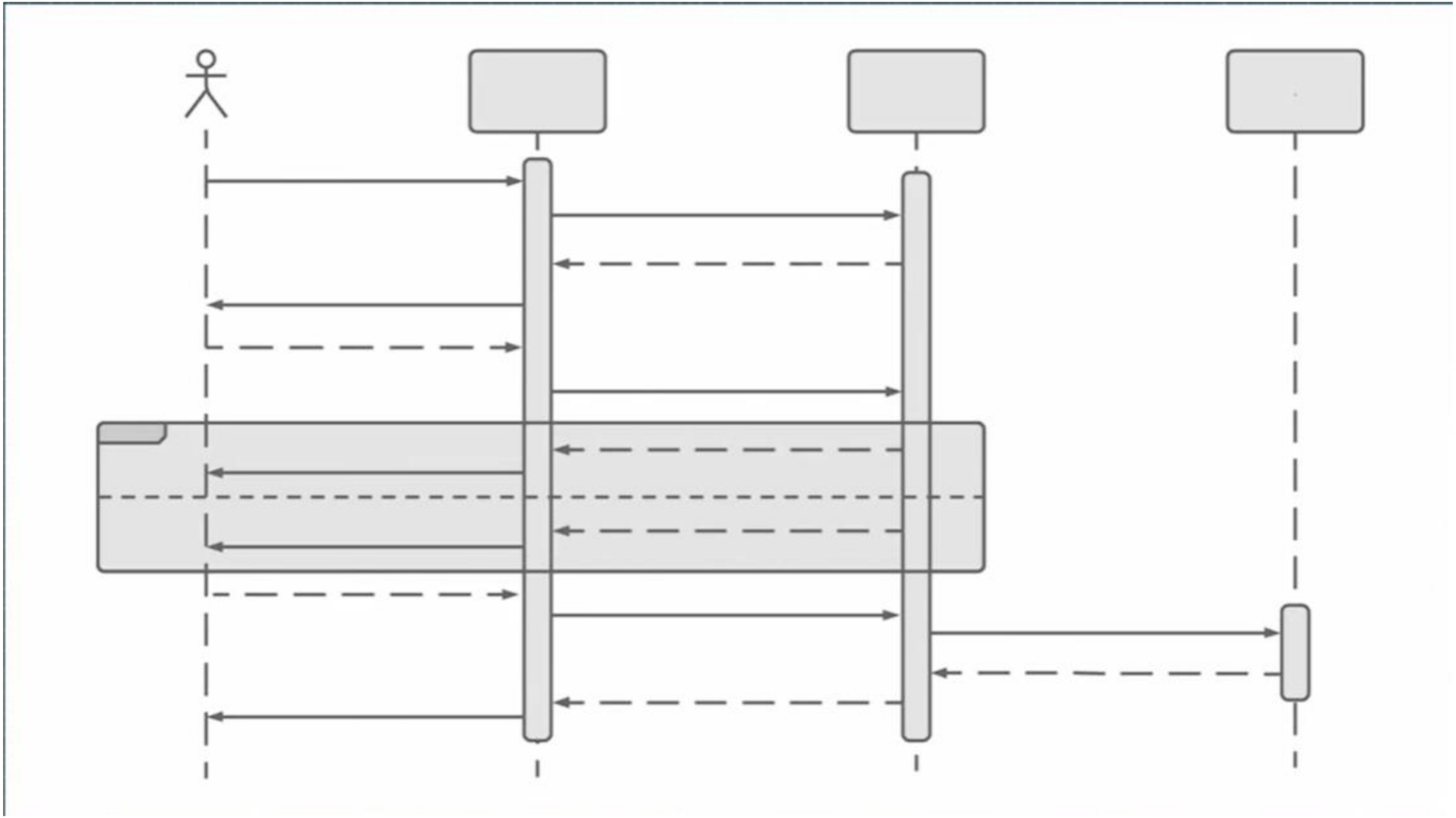
- System sequence diagram illustrates input and output events related to a system
- A system sequence diagram is a picture that shows, for particular scenario of a use case, the events that external actor generates, and their order in a system (Black box)

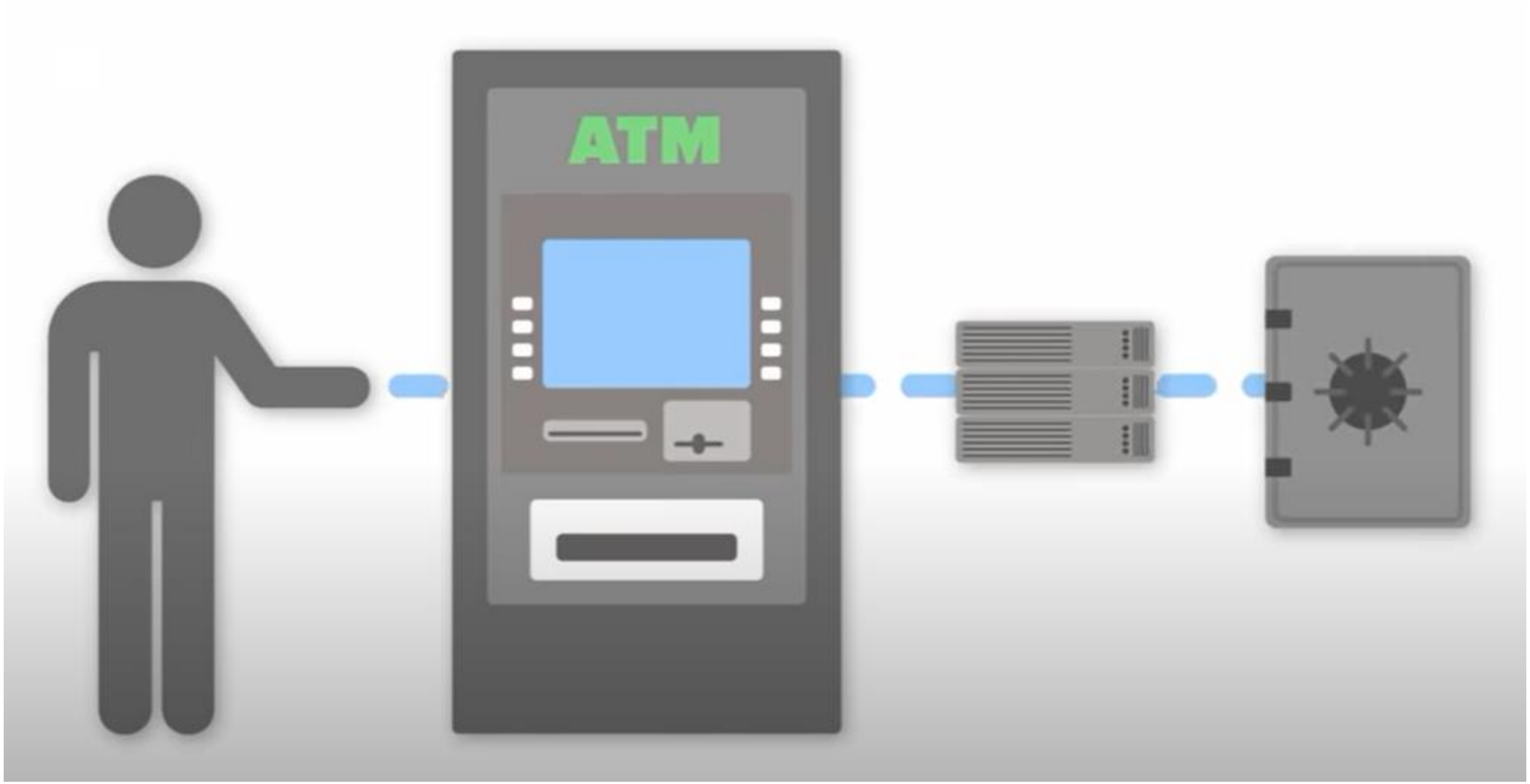
- Visitor A presses the Up floor button at floor 3 to request an elevator, he wishes to go to floor 7
- The up floor button turned on
- An elevator arrives at floor 3
- The up floor button turn off
- The elevator doors open
- The timer starts
- Visitor A enters the elevator
- The elevators door close after the time out

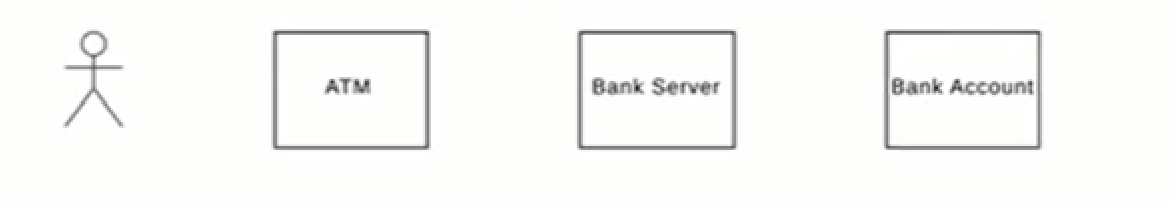
SEQUENCE DIAGRAM

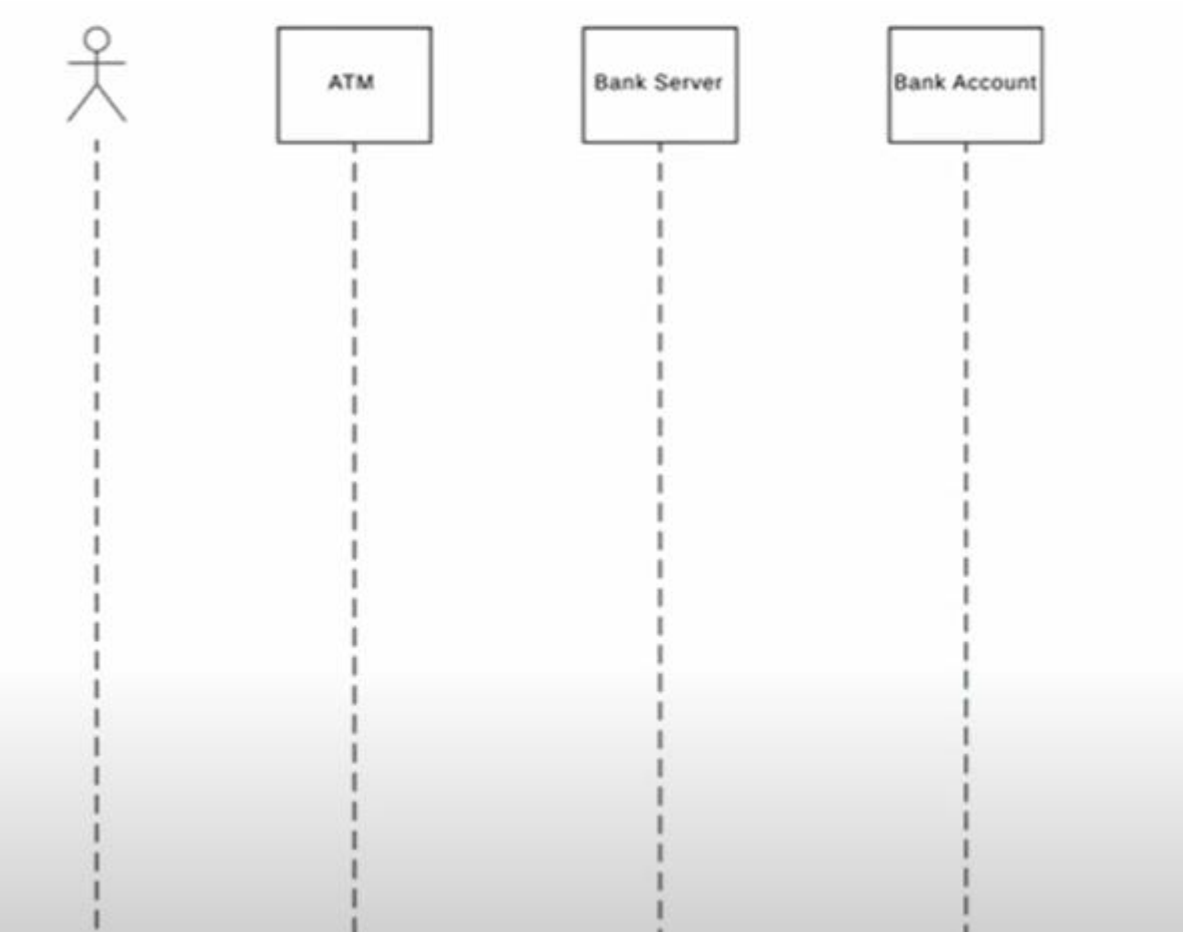
Interaction diagram that emphasizes the time ordering of messages

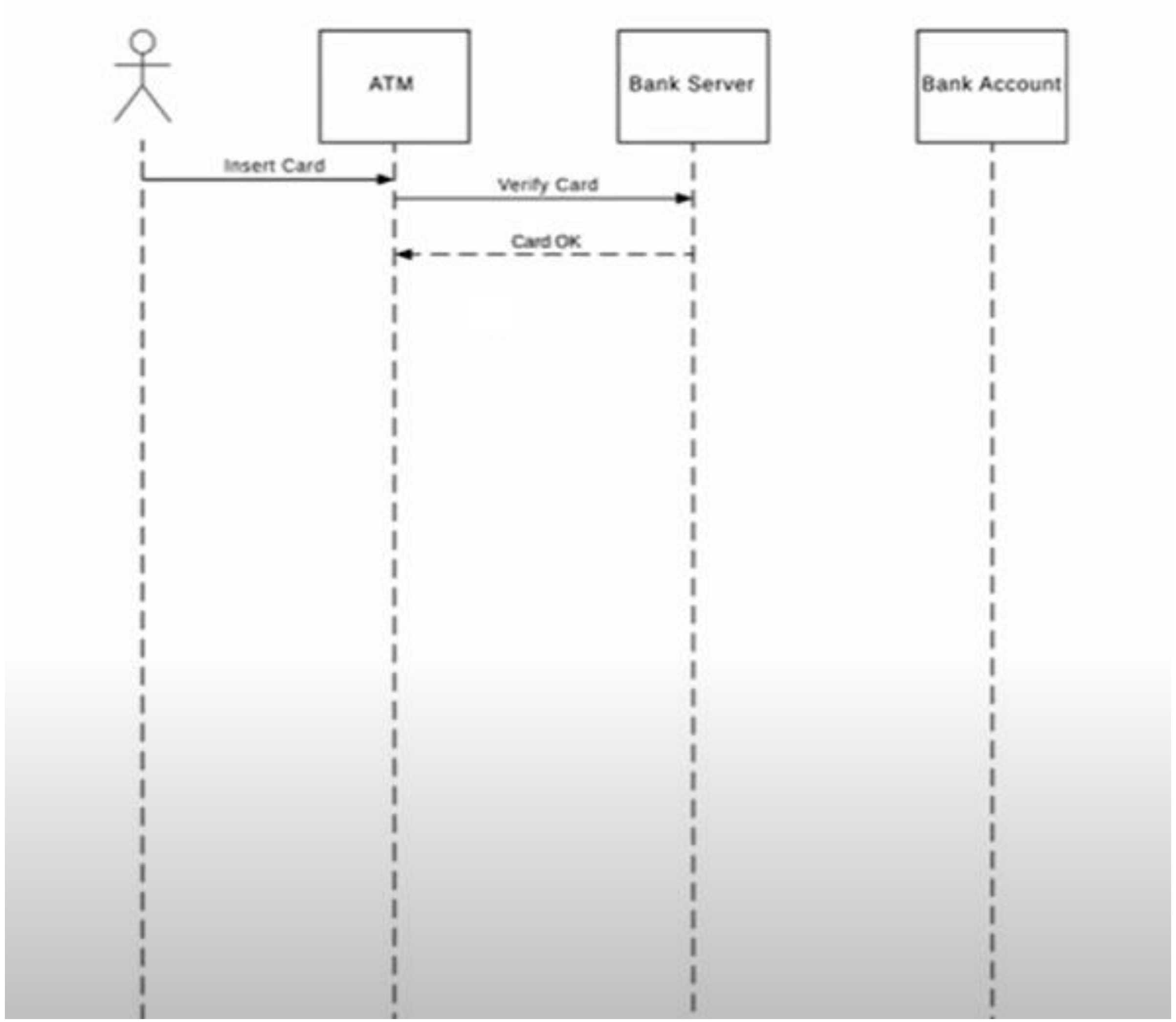




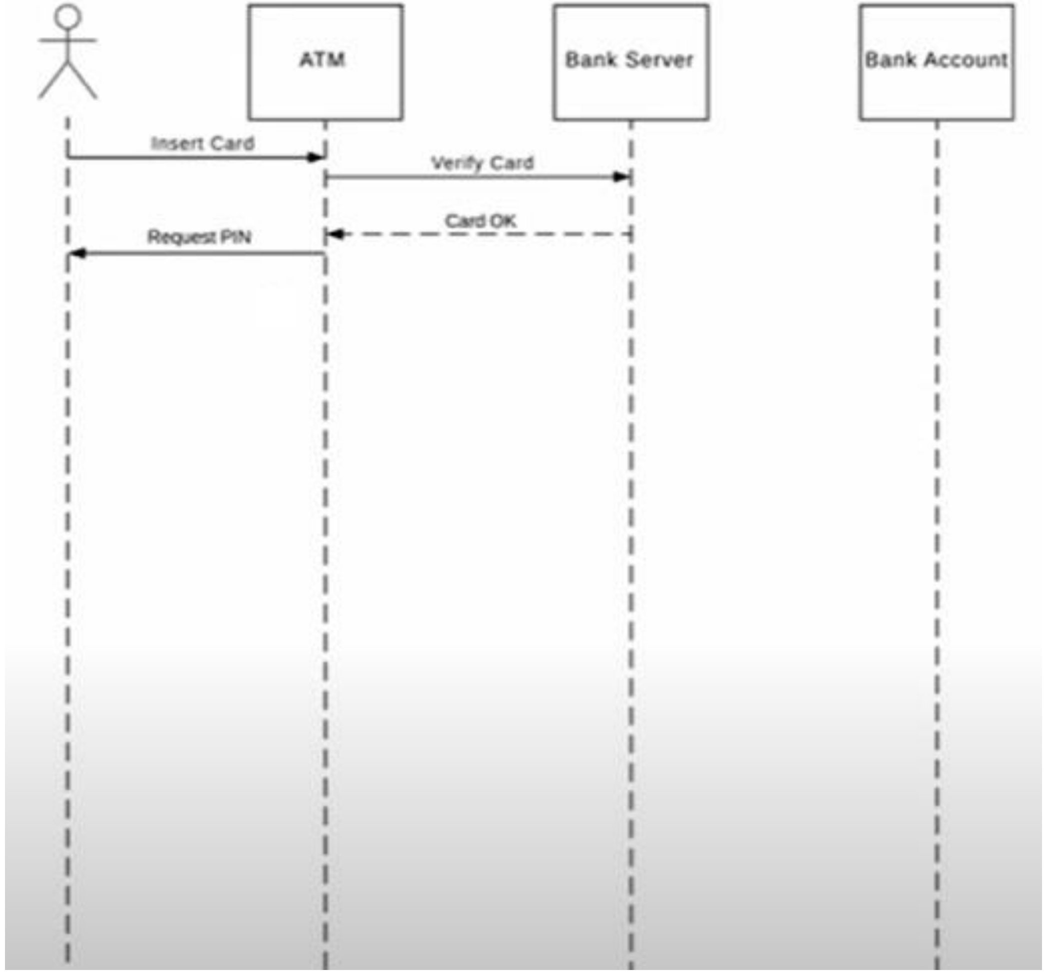

ATM

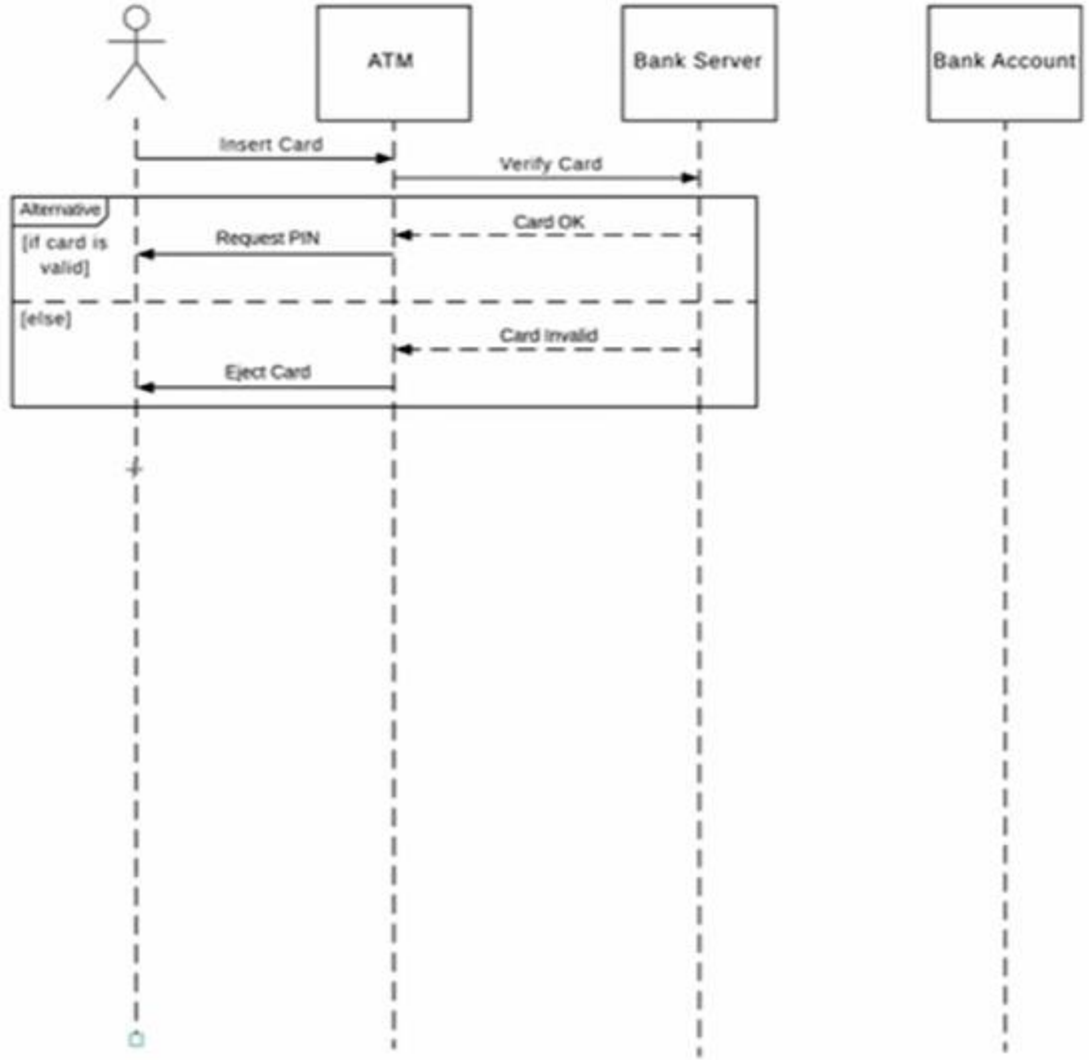

ATM
Bank Server
Bank Account

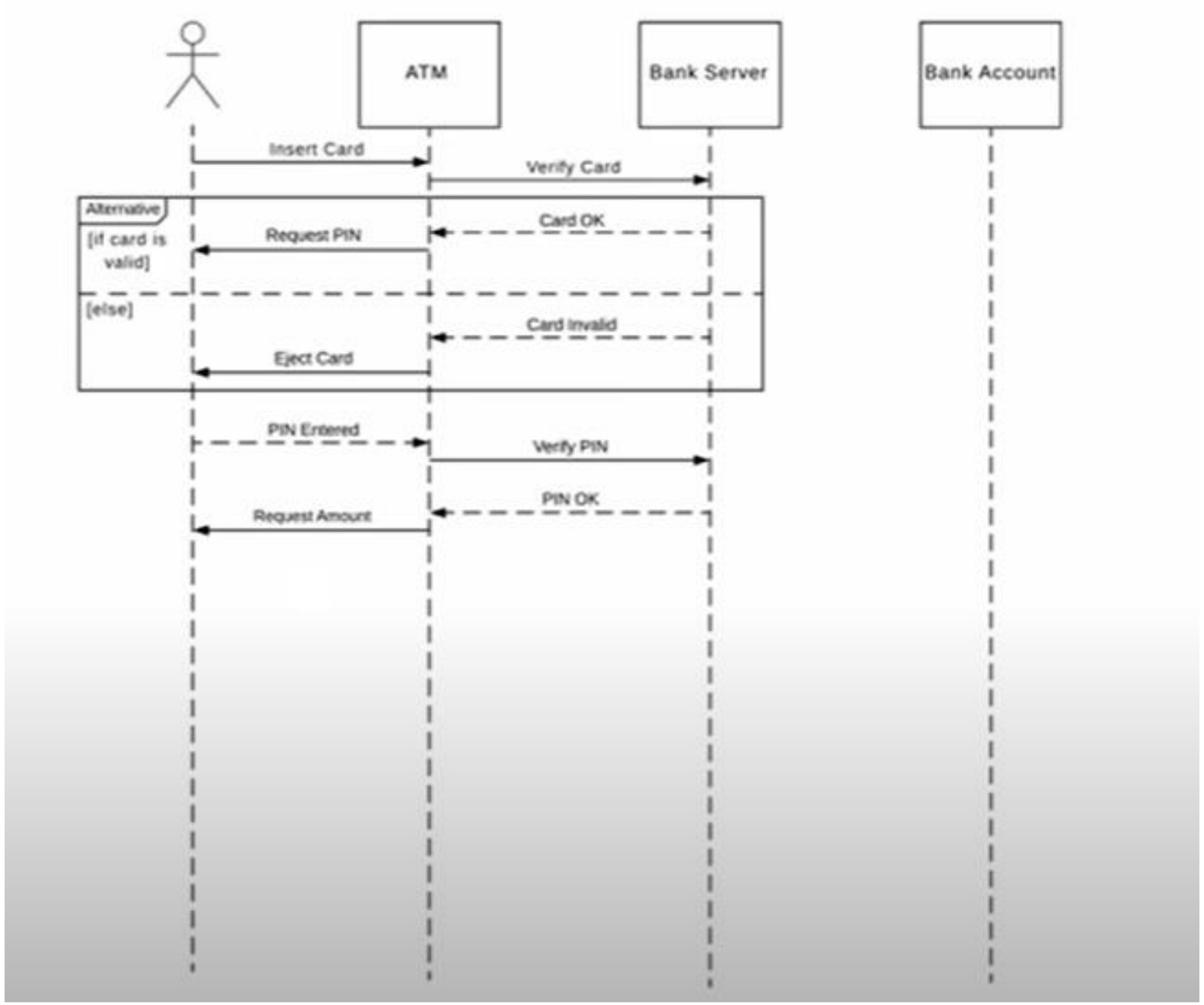

ATM
Bank Server
Bank Account

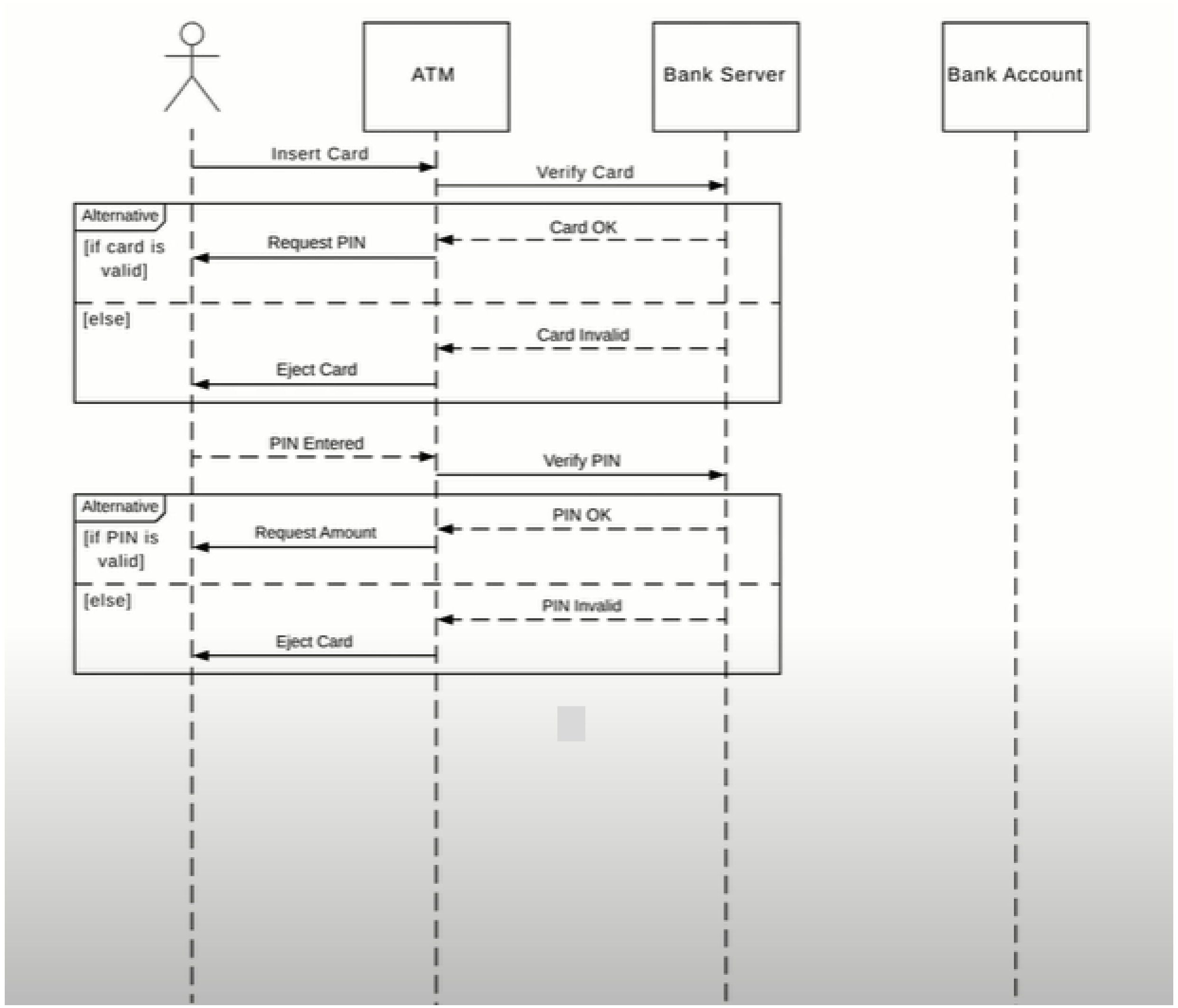


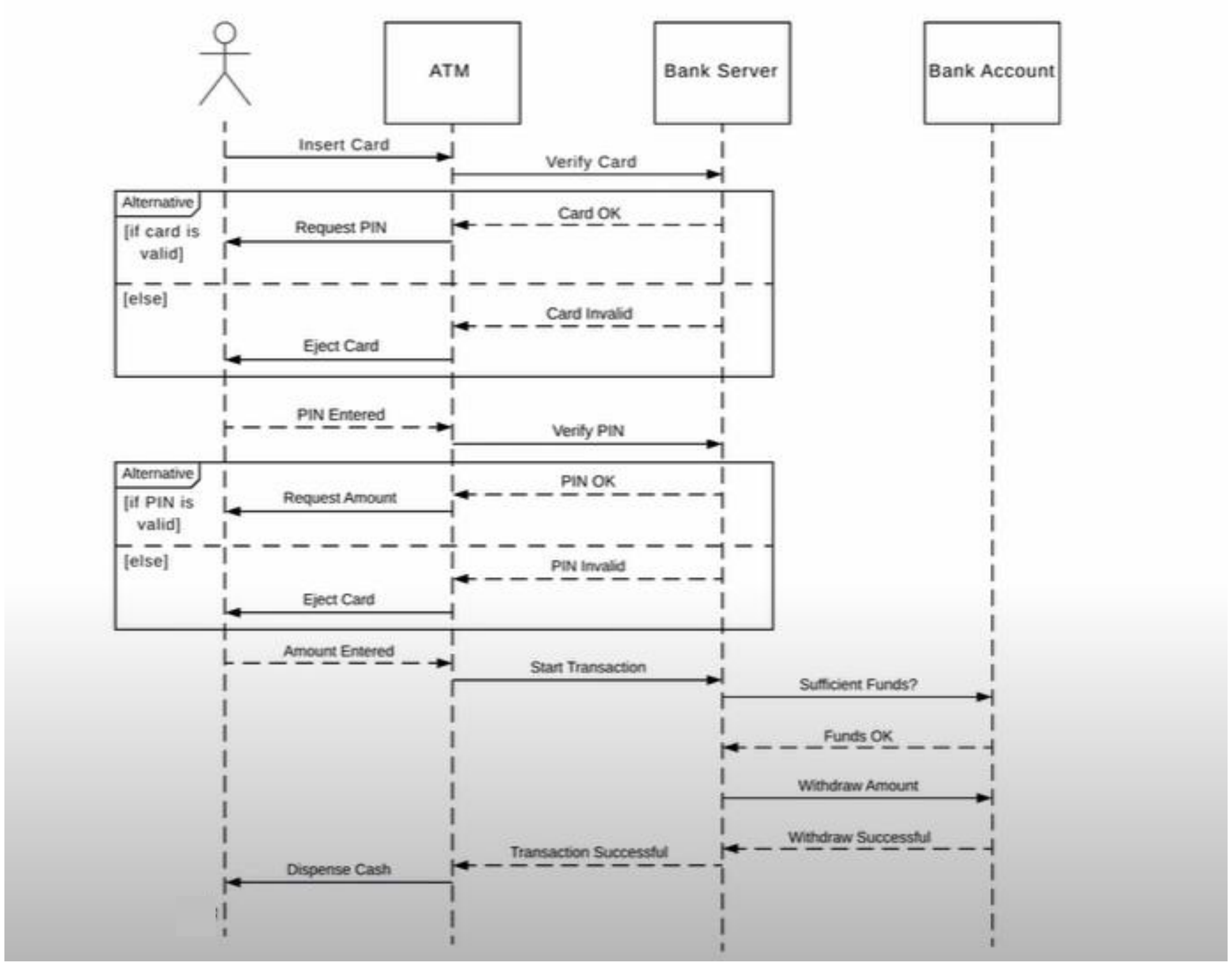
ATM
Bank Server
Bank Account
Insert Card
Verify Card
Card OK

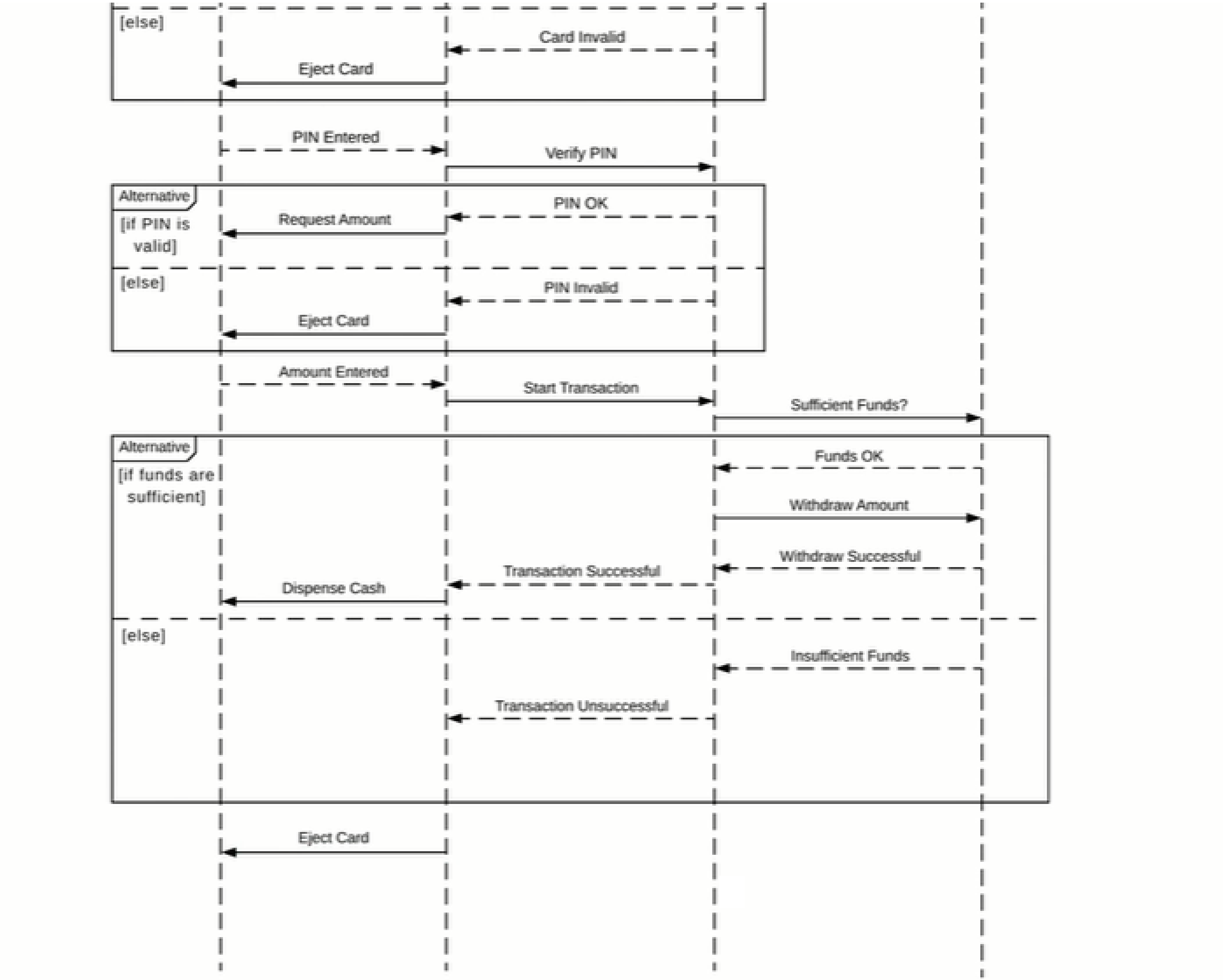

ATM
Bank Server
Bank Account
Insert Card
Verify Card
Card OK
Request PIN

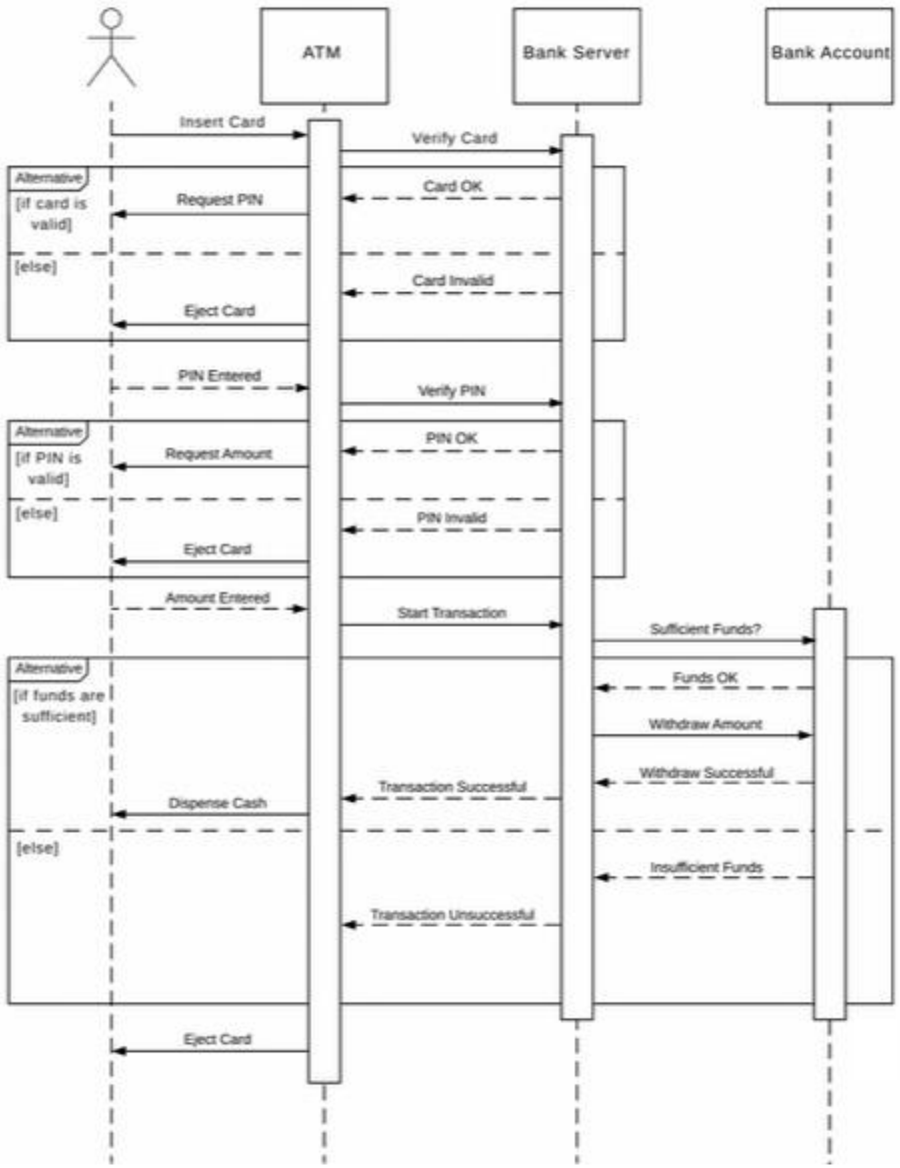

ATM
Bank Server
Bank Account
Insert Card
Verify Card
Alternative
[if card is valid]
Card OK
Request PIN
[else]
Card Invalid
Eject Card


ATM
Bank Server
Bank Account
Insert Card
Verify Card
Alternative
[if card is valid]
Card OK
Request PIN
[else]
Card Invalid
Eject Card
PIN Entered
Verify PIN
PIN OK
Request Amount


ATM
Bank Server
Bank Account
Insert Card
Verify Card
Alternative
[if card is valid]
Card OK
Request PIN
[else]
Card Invalid
Eject Card
PIN Entered
Verify PIN
Alternative
[if PIN is valid]
PIN OK
Request Amount
[else]
PIN Invalid
Eject Card


ATM
Bank Server
Bank Account
Insert Card
Verify Card
Alternative
[if card is valid]
Card OK
Request PIN
[else]
Card Invalid
Eject Card
PIN Entered
Verify PIN
Alternative
[if PIN is valid]
PIN OK
Request Amount
[else]
PIN Invalid
Eject Card
Amount Entered
Start Transaction
Sufficient Funds?
Funds OK
Withdraw Amount
Withdraw Successful
Transaction Successful
Dispense Cash

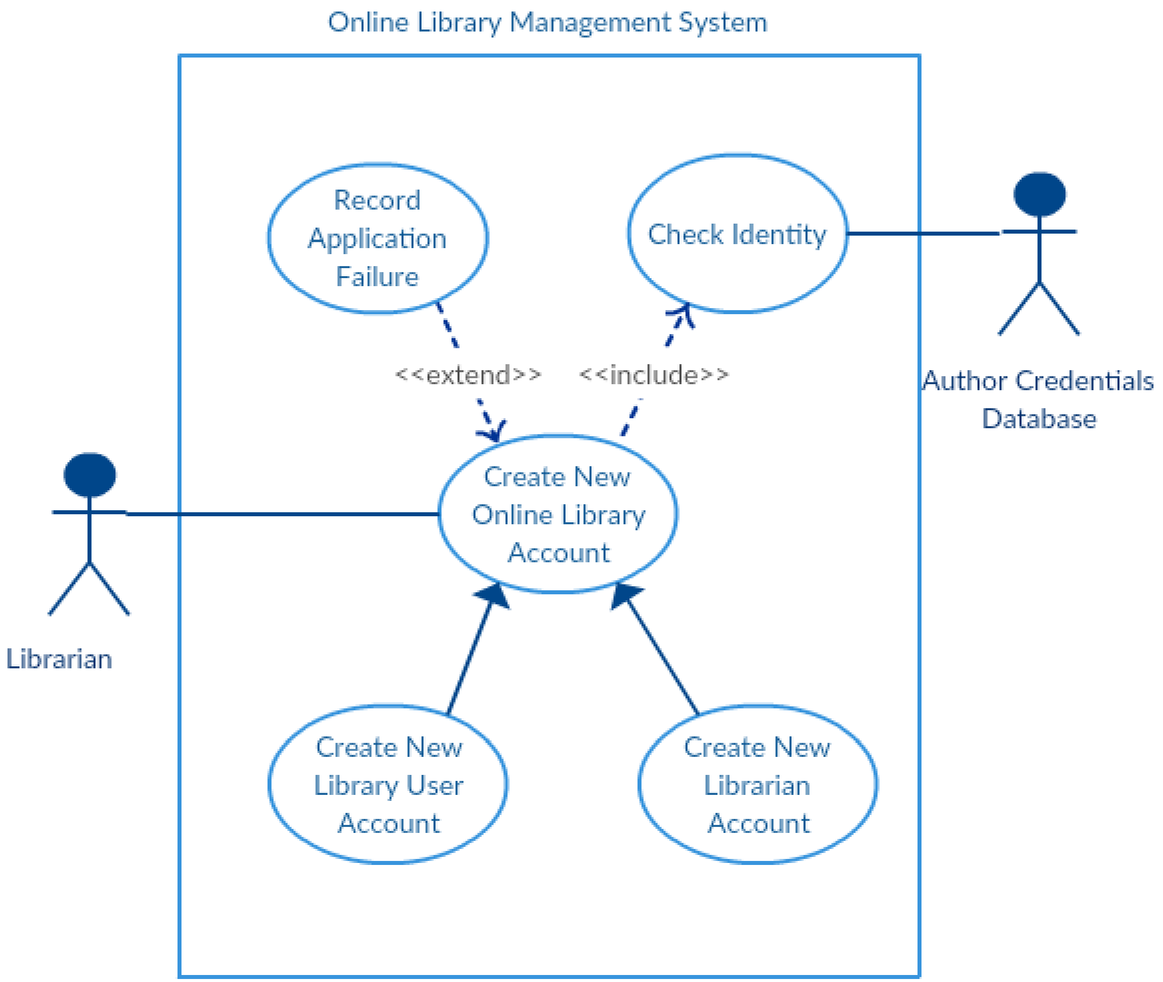

[else]
Card Invalid
Eject Card
PIN Entered
Verify PIN
Alternative
[if PIN is valid]
PIN OK
Request Amount
[else]
PIN Invalid
Eject Card
Amount Entered
Start Transaction
Sufficient Funds?
Alternative
[if funds are sufficient]
Funds OK
Withdraw Amount
Withdraw Successful
Transaction Successful
Dispense Cash
[else]
Insufficient Funds
Transaction Unsuccessful
Eject Card


ATM
Bank Server
Bank Account
Insert Card
Verify Card
Alternative
[if card is valid]
Card OK
Request PIN
[else]
Card Invalid
Eject Card
PIN Entered
Verify PIN
Alternative
[if PIN is valid]
PIN OK
Request Amount
[else]
PIN Invalid
Eject Card
Amount Entered
Start Transaction
Sufficient Funds?
Alternative
[if funds are sufficient]
Funds OK
Withdraw Amount
Withdraw Successful
Transaction Successful
Dispense Cash
[else]
Insufficient Funds
Transaction Unsuccessful
Eject Card

How to Draw a Sequence Diagram:

A sequence diagram represents the scenario or flow of events in one single use case.
The message flow of the sequence diagram is based on the narrative of the particular use case.

Then, before you start drawing the sequence diagram or decide what interactions should be included in it, you need to draw the use case diagram and ready a comprehensive description of what the particular use case does.


Online Library Management System
Record Application Failure
Check Identity
<<extend>>
<<include>>
Author Credentials Database
Create New Online Library Account
Librarian
Create New Library User Account
Create New Librarian Account

From the above use case diagram example of ‘Create New Online Library Account’, we will focus on the use case named ‘Create New User Account’ to draw our sequence diagram example.

Before drawing the sequence diagram, it’s necessary to identify the objects or actors that would be involved in creating a new user account. These would be;

- Librarian
- Online Library Management system
- User credentials database
- Email system

Once you identify the objects, it is then important to write a detailed description on what the use case does.

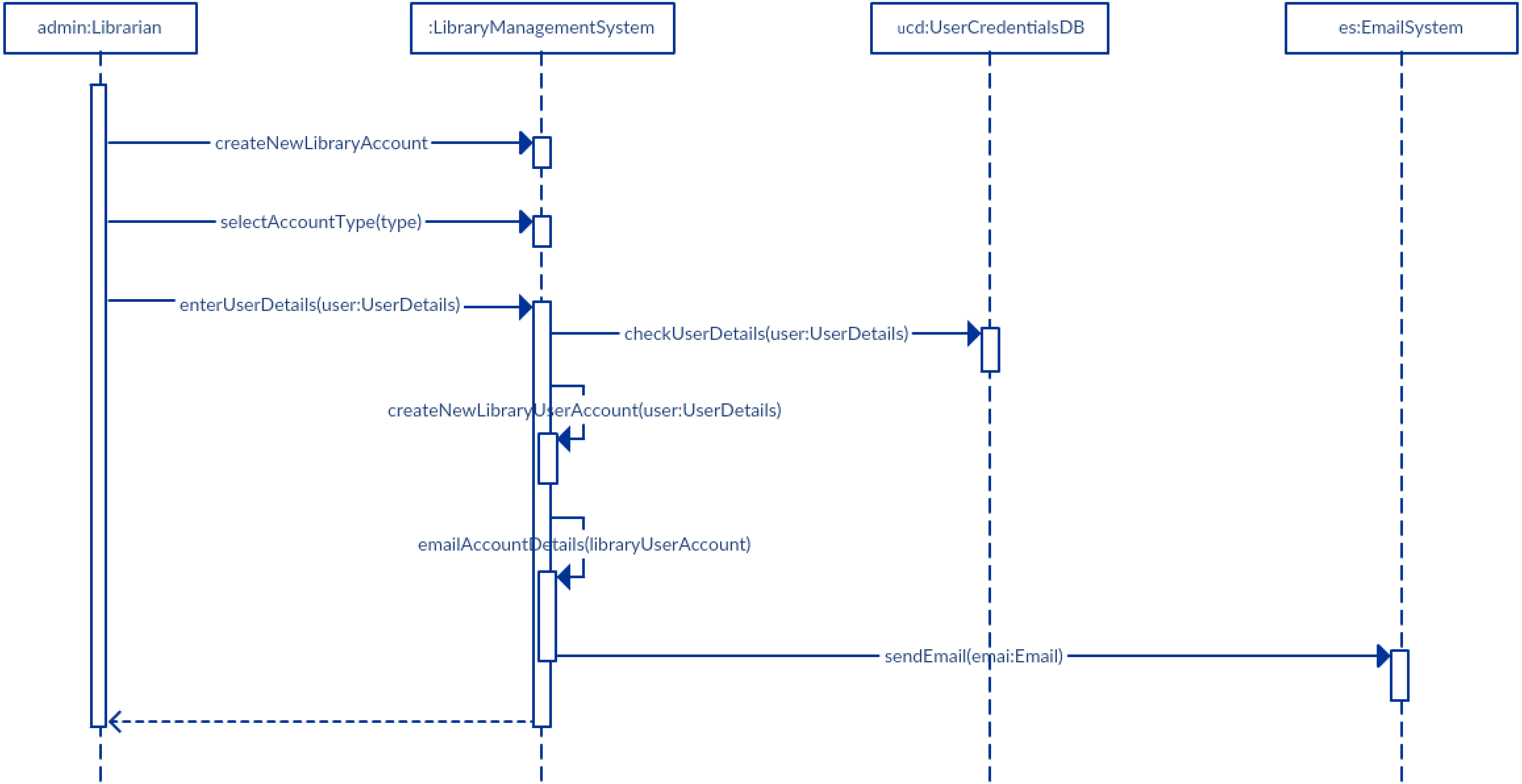
From this description, you can easily figure out the interactions (that should go in the sequence diagram) that would occur between the objects above, once the use case is executed.

Here are the steps that occur in the use case named ‘Create New Library User Account’.

- The librarian request the system to create a new online library account
- The librarian then selects the library user account type
- The librarian enters the user’s details
- The user’s details are checked using the user Credentials Database
- The new library user account is created
- A summary of the of the new account’s details are then emailed to the user

From each of these steps, you can easily specify what messages should be exchanged between the objects in the sequence diagram. Once it’s clear, you can go ahead and start drawing the sequence diagram.

The sequence diagram below shows how the objects in the online library management system interact with each other to perform the function ‘Create New Library User Account’.


admin:Librarian
:LibraryManagementSystem
ucd:UserCredentialsDB
es:EmailSystem
createNewLibraryAccount
selectAccountType(type)
enterUserDetails(user:UserDetails)
checkUserDetails(user:UserDetails)
createNewLibraryUserAccount(user:UserDetails)
emailAccountDetails(libraryUserAccount)
sendEmail(emai:Email)