

# Behavioral Modeling Using UML (Interaction Diagrams)

## Sequence Diagram

# Content

- Explain **behavioral modeling** in UML
- Describe purpose and components of **sequence diagrams**
- Main characteristics of sequence diagram
  - **Frame**
  - **Actors**
  - **Objects/Stereo types/Lifeline/Activation Bar**
  - **Messages (Asynchronous/Synchronous/Self)**
  - **Fragments (Ref/Loop/Opt/Alt)**
- Construct sequence diagram for real world scenario
- **Evaluate the correctness** of interaction diagrams

# Lesson Learning Outcomes

**By the end of this lecture, students will be able to:**

- Explain behavioral modeling in UML.
- Describe the purpose and key components of sequence diagrams.
- Differentiate message types and interaction fragments.
- Draw sequence diagrams for real-world scenarios.
- Evaluate the correctness of sequence diagrams.

# How would you explain a conversation sequence with a friend?

- Time Line?
- Messages?
- Flow charts?



# What is a Behavioral Model?

- Behavioral model shows how the **system behaves** in **response to events**
- We model behavior of a software using **Interaction diagrams**
- Two main types in UML interaction diagrams:
  - **Sequence Diagrams** (time-ordered interactions)
  - **Communication Diagrams** (Collaborative interactions)

# Online Banking System

## Use case scenario

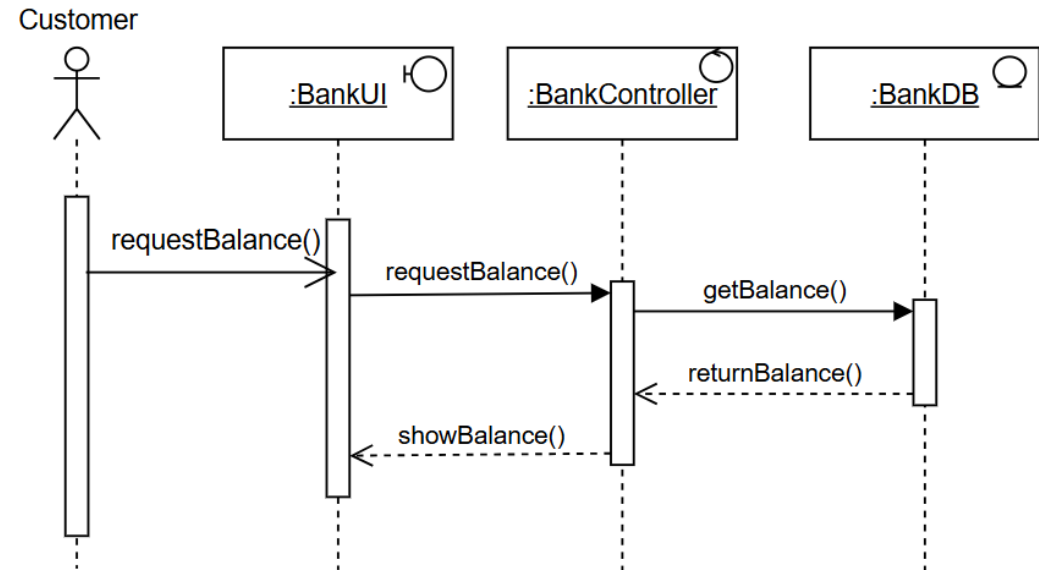
**Use Case Name:** *Check Balance*

**Actor:** Customer

**Scenario (Main Flow):**

- Customer requests account balance.
- System retrieves balance from the database.
- System shows balance to the customer.

## Interaction Diagram

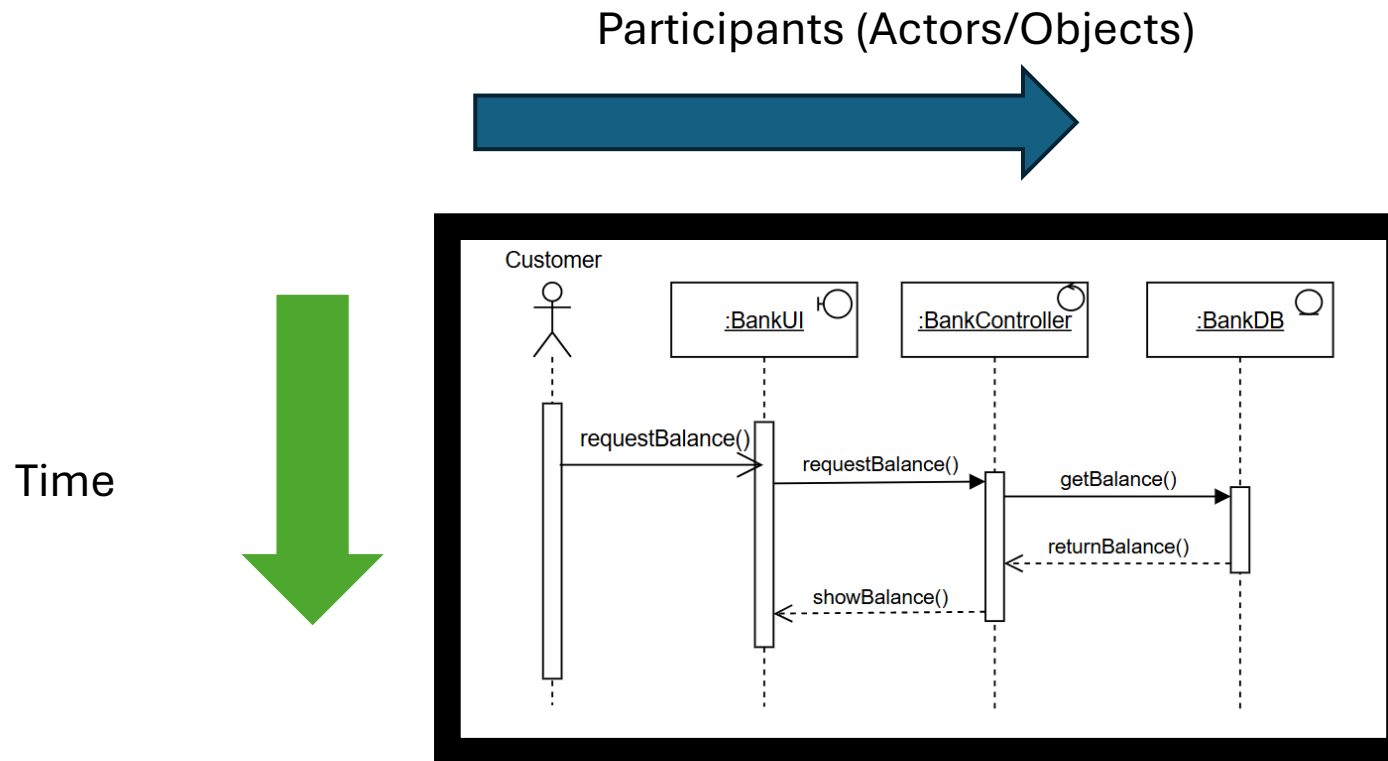


# Why Interaction Diagram?

- Understand **object collaborations**
- Provide **time-sequenced view** of interactions
- Aid requirement **validation & design** refinement
- Bridge **use cases to design**

# Sequence Diagram

- Interaction diagram which emphasizes on **time ordering of messages**

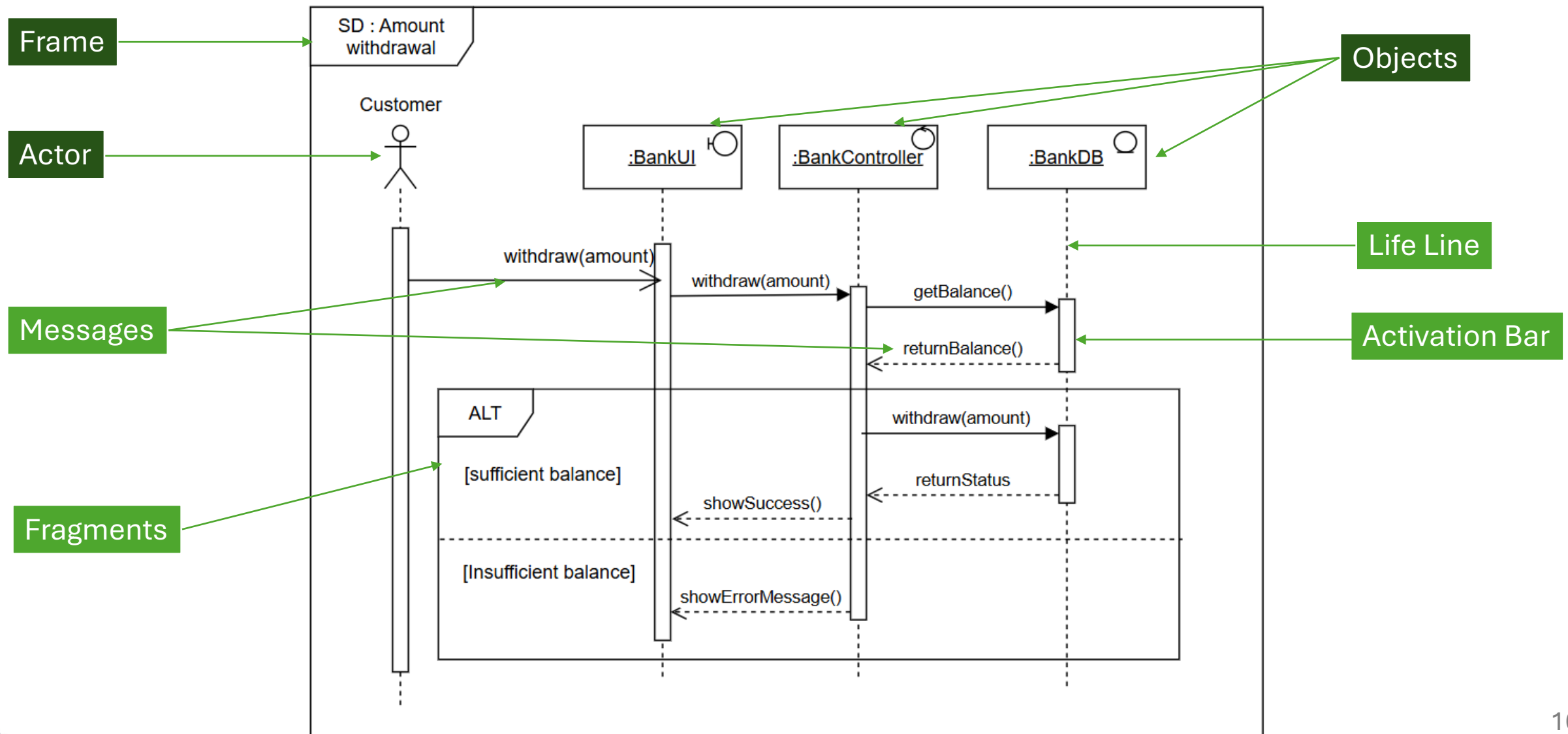




# Main Characteristics of Sequence Diagram

- **Frame** : Unit of observable exchange of information between different objects
- **Participants**
  - Actors
  - Objects
- **Lifelines & Activation Bars**
- **Messages**
- **Fragments**

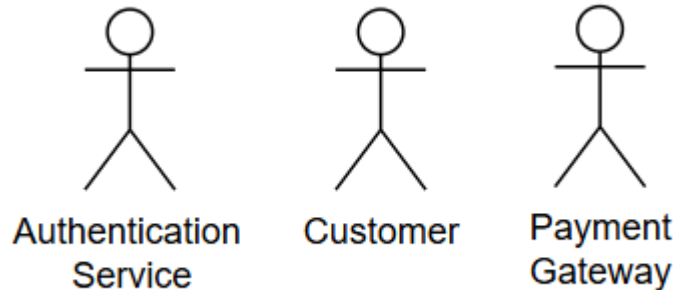
# Main Characteristics of Sequence Diagram



# Participants:

**Actor** : External human users/external HW/ services who send or receive messages

Examples:



**Object** : Instance of a class which interact with each other and actors. There are two types of objects.

1. Named object
2. Anonymous object

Object name : Class name

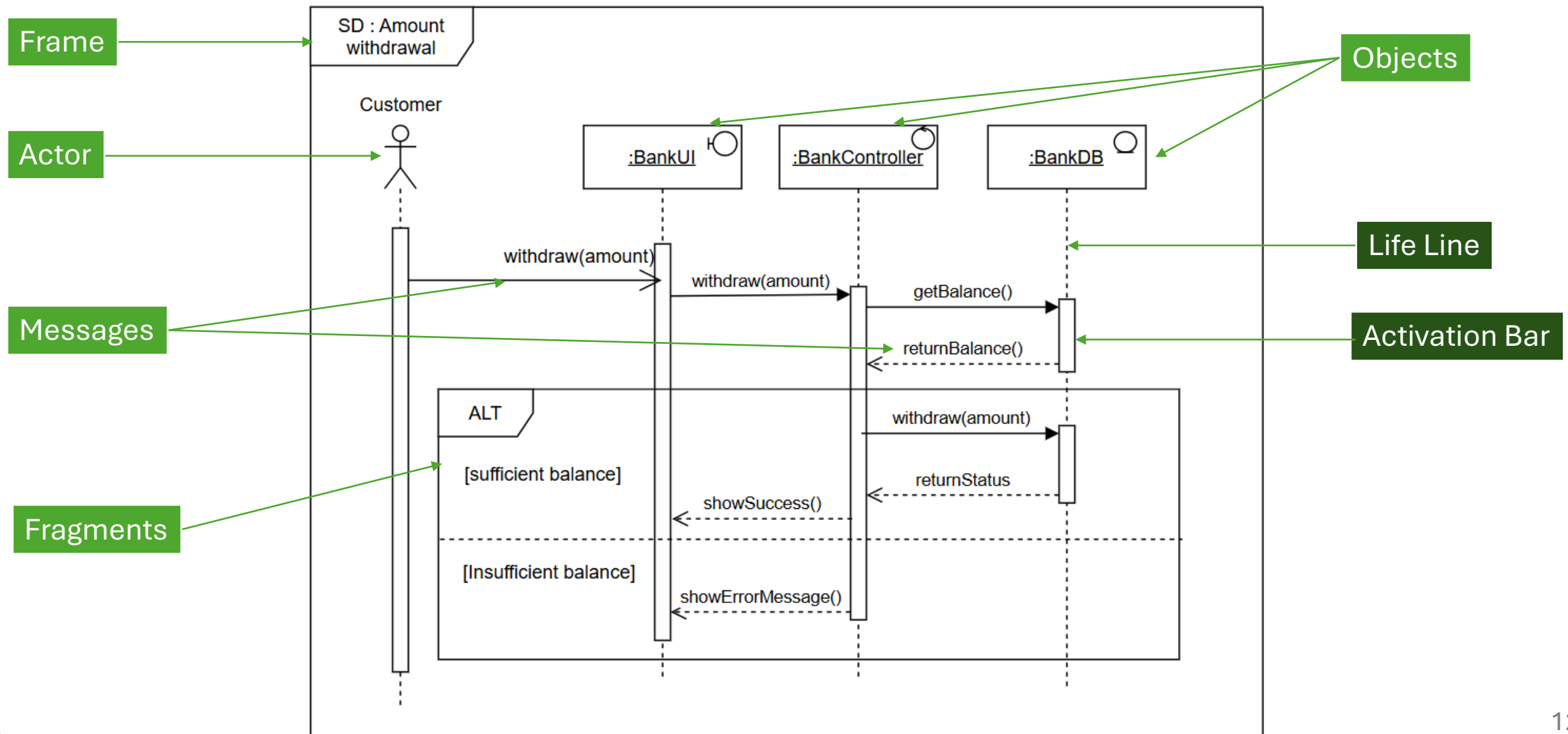
: Class name

Examples:

AndrewCart: ShoppingCart

: Order

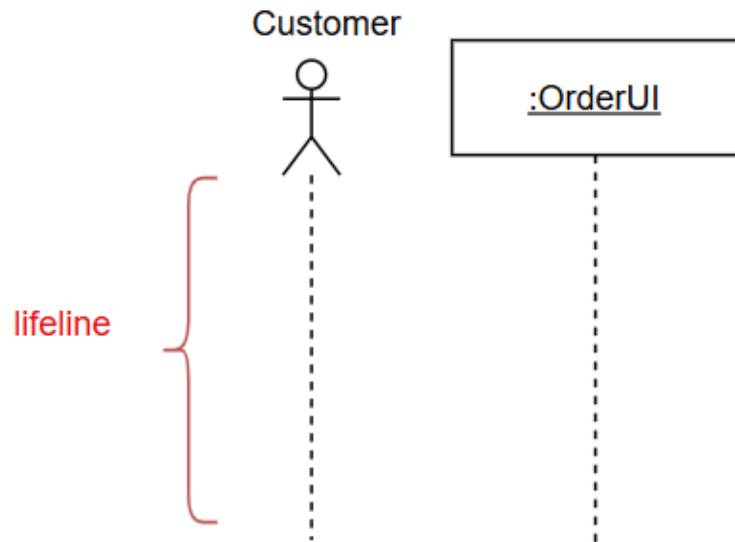
# Main Characteristics of Sequence Diagram



# Lifeline and Activation Bar:

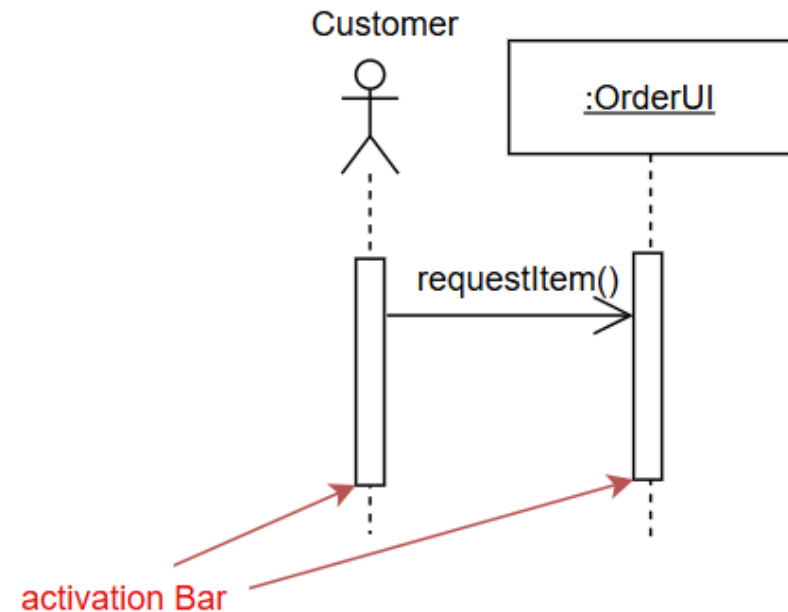
**Lifelines** : Each participant has corresponding lifeline. Vertical dotted line representing the time that an object exist

Examples:

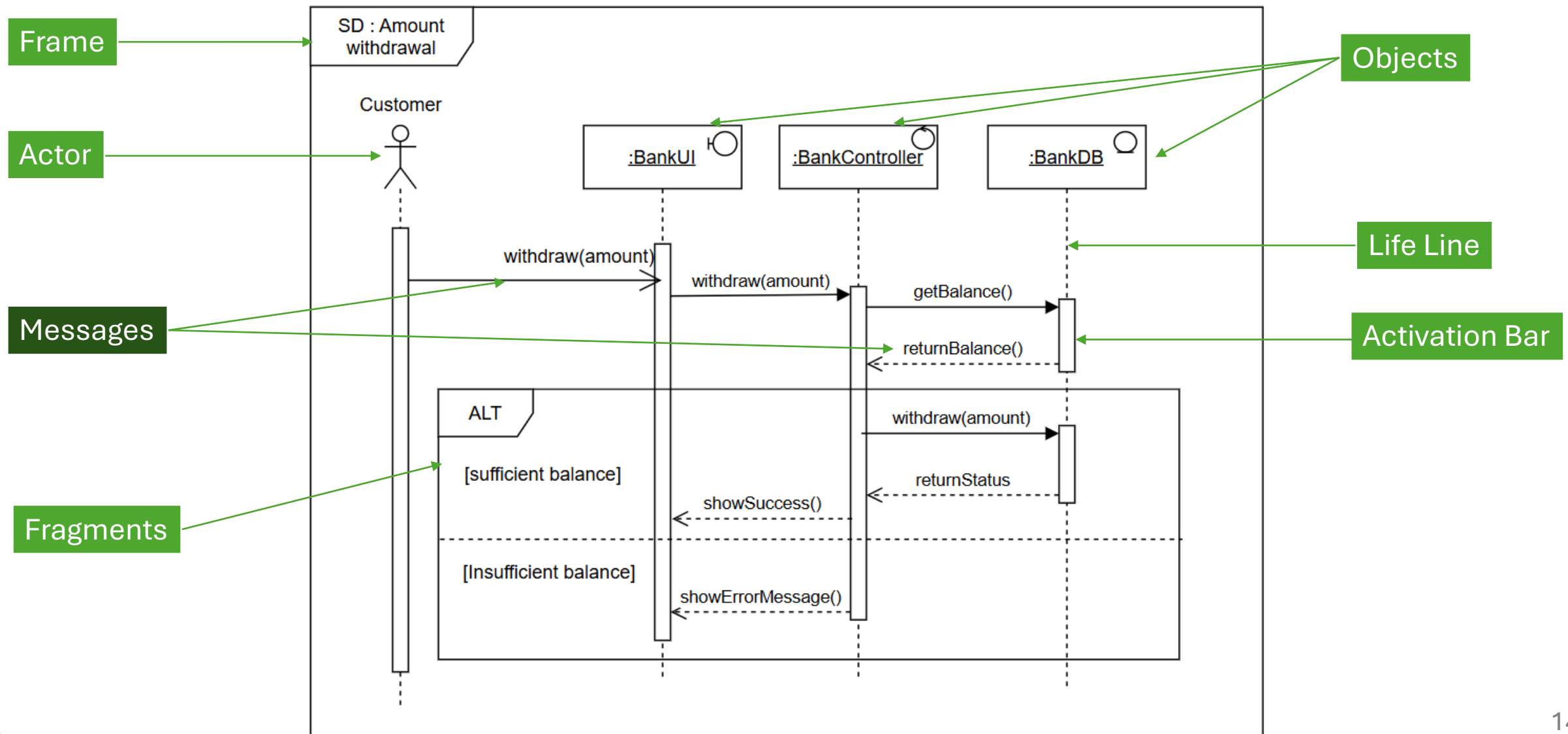


**Activation Bar** : participants activeness is visible through the activation bar

Example:



# Main Characteristics of Sequence Diagram



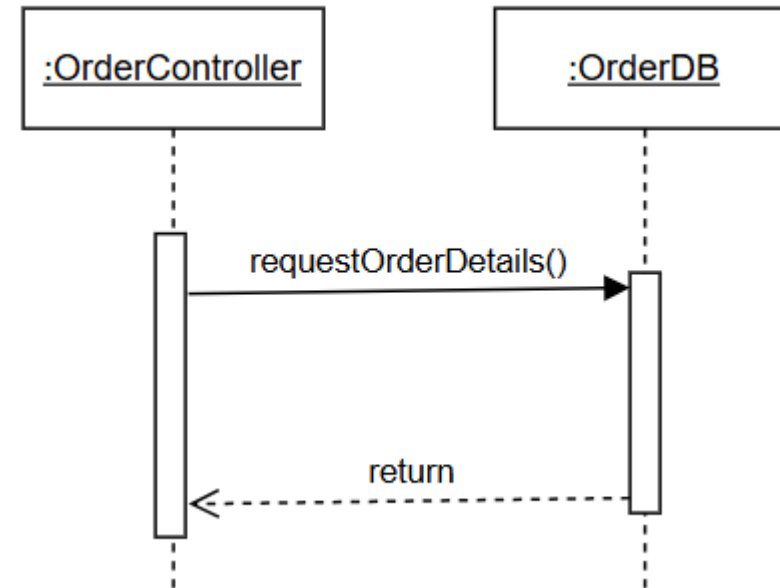
# Messages

Messages are used to illustrate **communication between different active objects** of a sequence diagram

- Asynchronous messages
- Synchronous messages
- Return messages
- Self message

## Message Passing



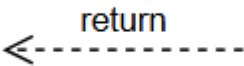
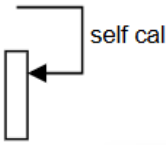
Invocation of a method in one object, by another method that belongs to a different object.



When **OrderController** object sends a message to **OrderDB**,

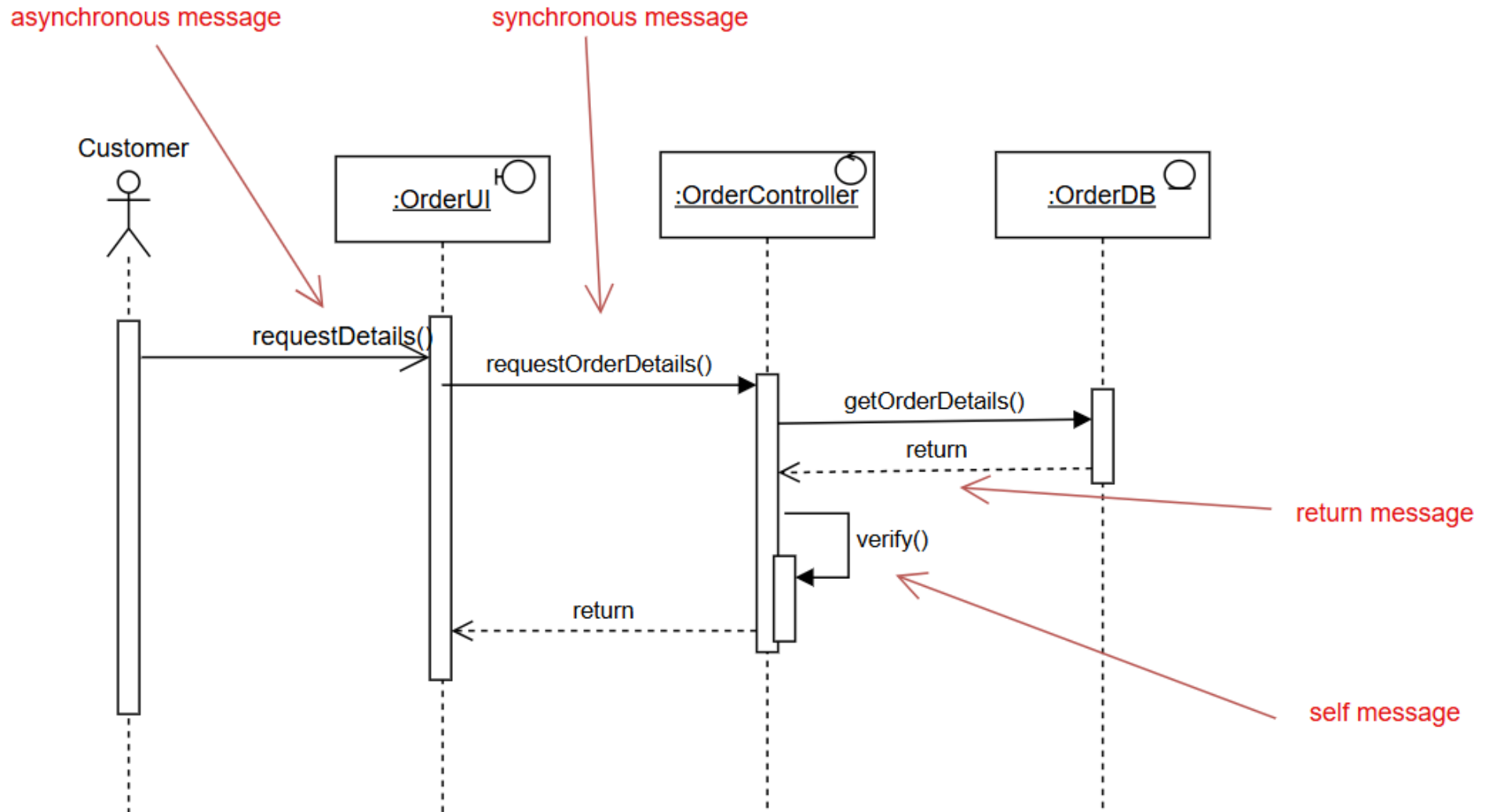
it requests OrderDB to execute one of its own methods (**requestOrderDetails()**)

# Messages

	Asynchronous Message	Synchronous Message	Return Message	Self Message
Description	The <b>sender does not wait for the receiver to finish</b> processing the message ( for any return values), it continues immediately	when the sender <b>waits</b> until the receiver has finished processing the message.	Usually shows the <b>return value</b> (e.g., balance, success, error).	When one method calling another method <b>belonging to the same object.</b> (message to the same lifeline)
Symbol				



# Messages

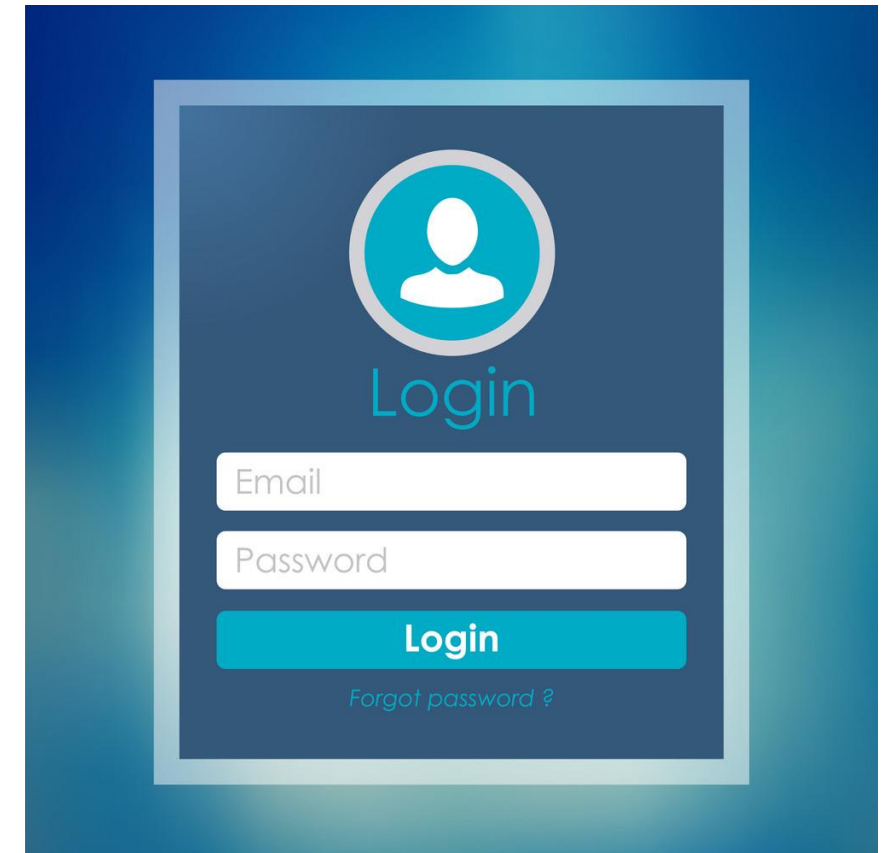


# Activity 1:

Draw a basic sequence diagram for the following scenario




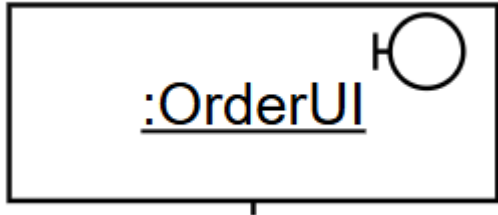
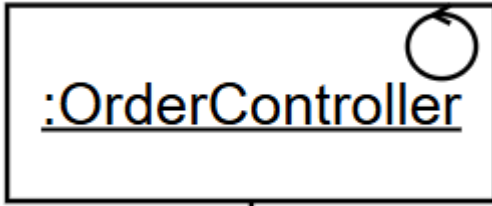
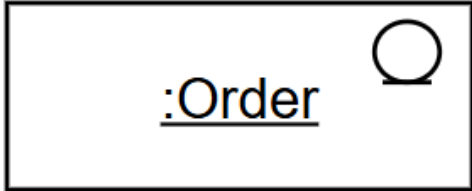
Scenario: User logs into an online banking system

1. Actor: **User**
2. Objects: **LoginPage, AuthService, Database**
3. Steps:
  - **Enter credentials(username, password)**
  - **Validate**
  - **Response**

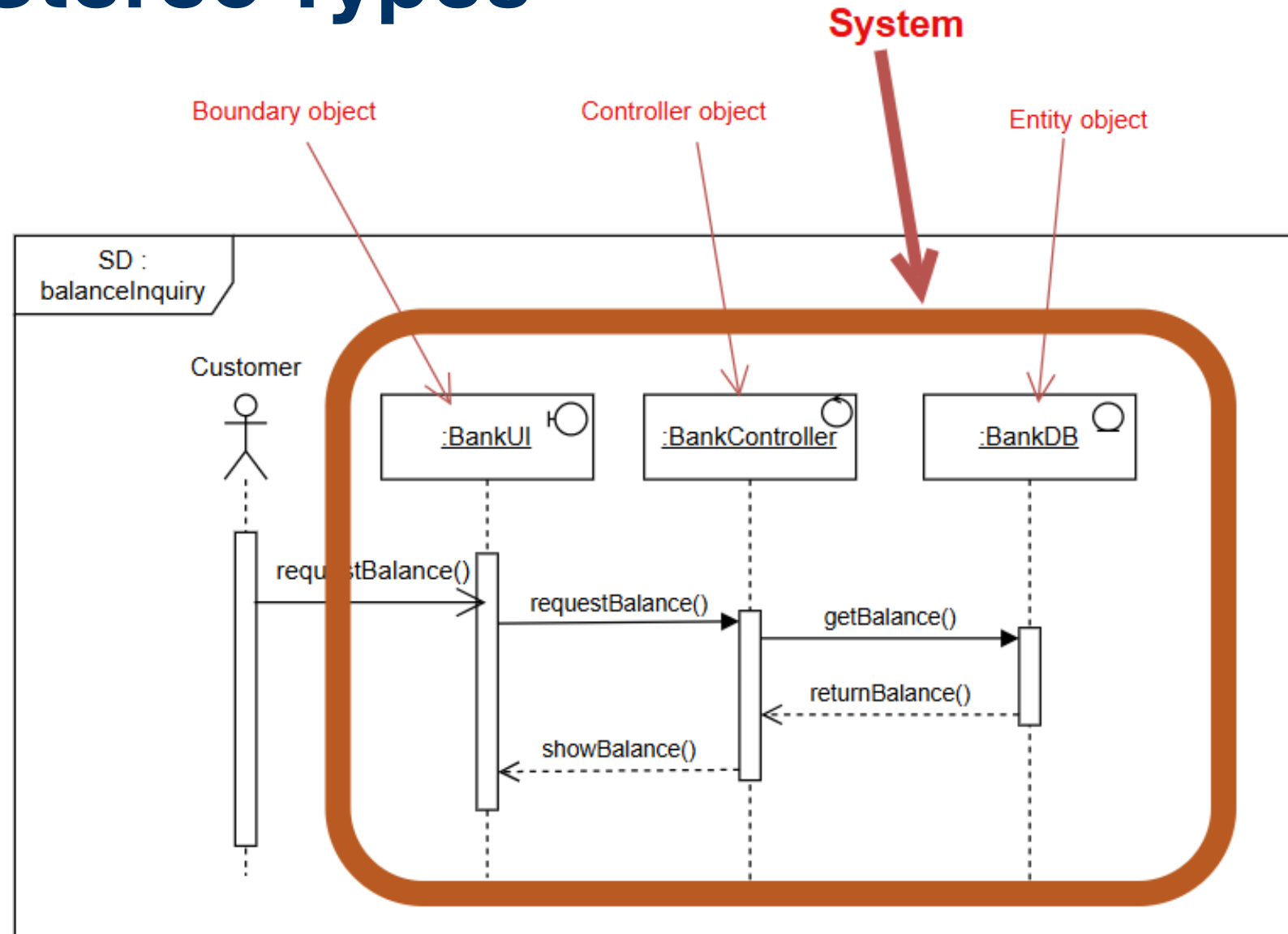


# Objects : Stereo Types

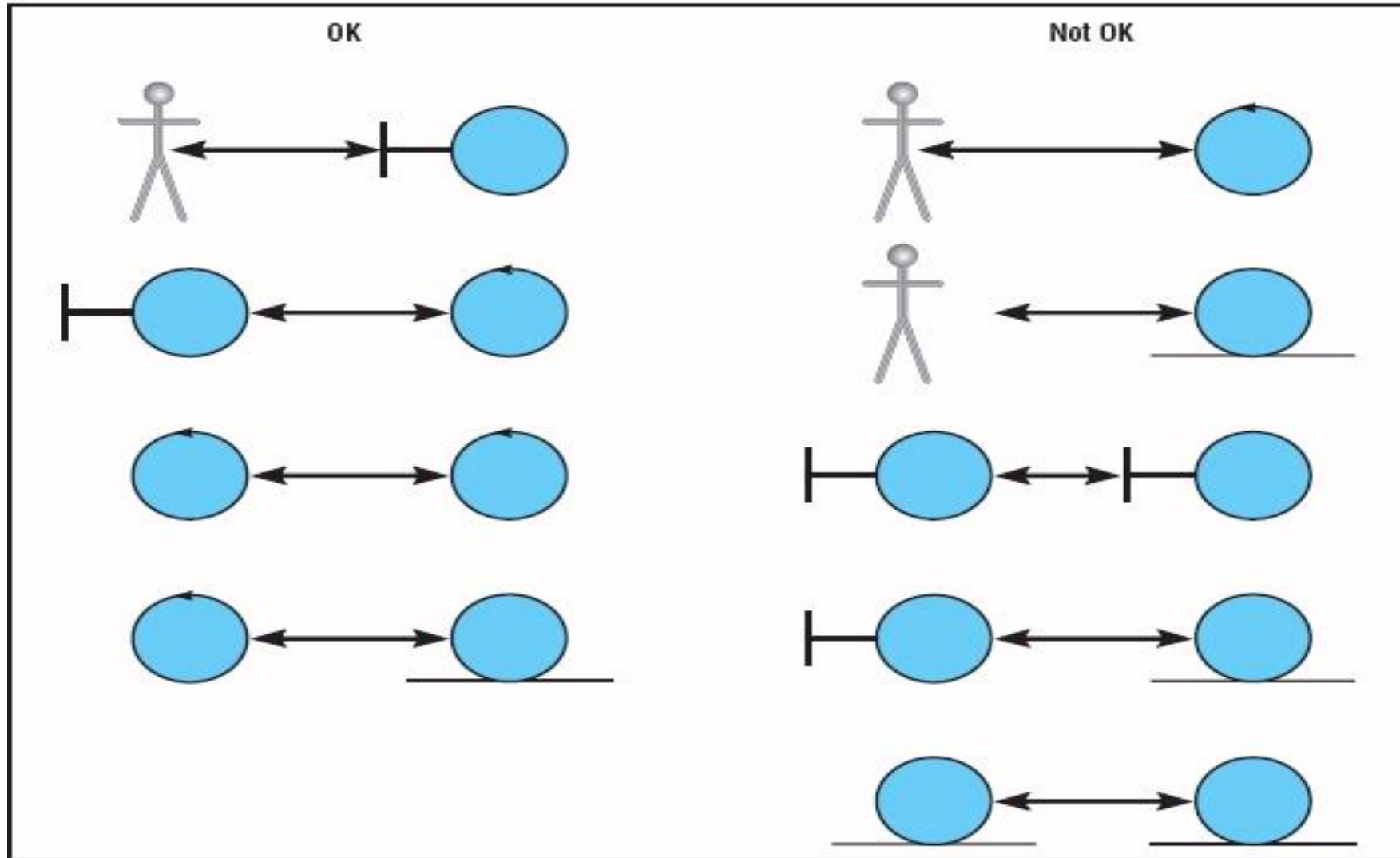
Stereotypes differentiate the **roles** that classes / objects can play.

	Boundary Objects	Controller Objects	Entity Objects
Description	Objects in the <b>boundary of the system which the actors act upon</b>	Objects which <b>control and coordinate</b> other objects.	Objects which <b>represent information</b> and behavior in the application domain
Symbol			
Example			

# Objects : Stereo Types



# Rules of Stereo Types



# Activity 2:

Draw a basic sequence diagram for given “Inquire Book status” by identifying stereo types.

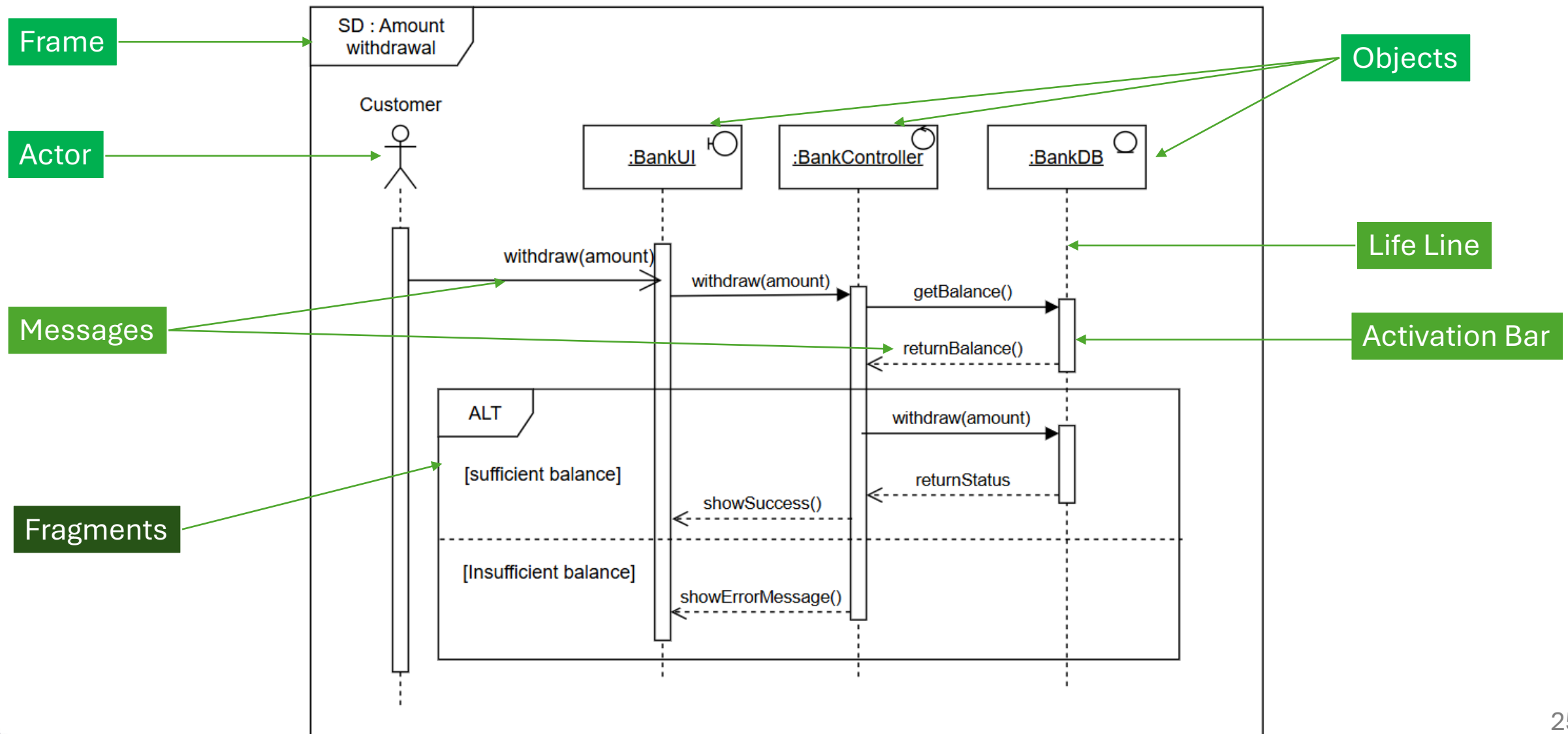
**Brief Description:** borrower inquires about availability of a book.

**Actor:** Library Member

**Flow of Events:**

1. Library member clicks “SearchBook” option in the **Library UI** by giving the Book ID.
2. Then **LibraryController** class will call SearchBook method in the **Book** class.
3. Then the system will send book status to the library member.
4. Next, the library member click reserve option in the UI and the LibraryController will call ReserveBook function of Book class.
5. Finally, Book class will send the reserve status to the library member.

# Main Characteristics of Sequence Diagram



# Interaction Fragments



# Interaction Fragments (TAGS)

Fragments helps to show more complex interactions in sequence diagram mainly **logic and control**

- REF - reference
- ALT - alternative
- OPT - optional
- LOOP
- PAR – parallel

These interaction fragments and operators greatly enhance the ability of sequence diagram.

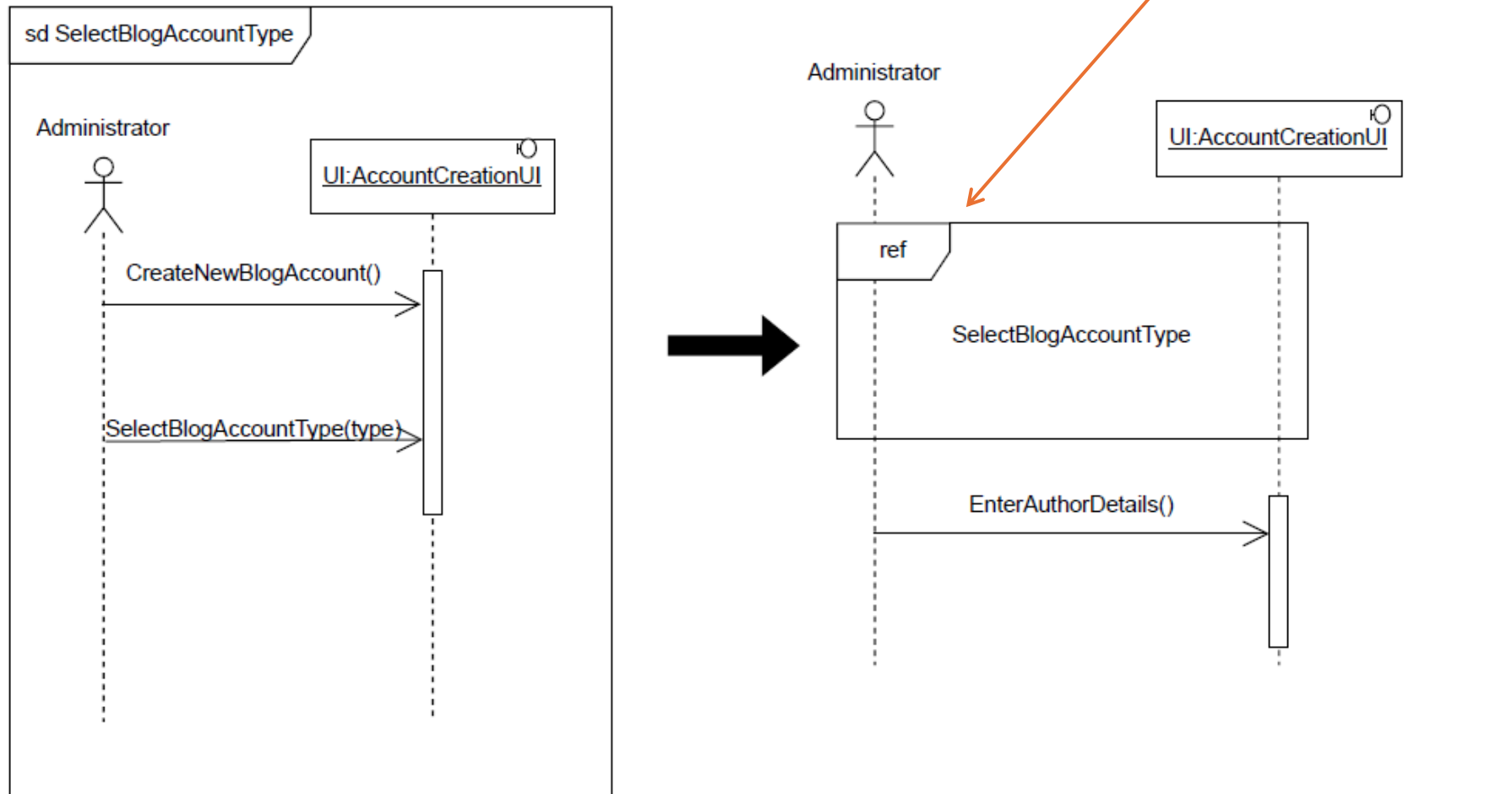
# Reference Fragment (REF Tag)

Reference to another interaction diagram.

Helps you to manage a large diagram by splitting, and potentially reusing, a collection of interactions



# Reference Fragment (REF Tag)



# Activity 3 :REF tag

Draw a sequence diagram for the following scenario

Customer needs to logging to the system to purchase an e-ticket for cinema.

Successfully logged customers, request movie details via **MovieUI** and the request will pass to the **MovieManeger**.

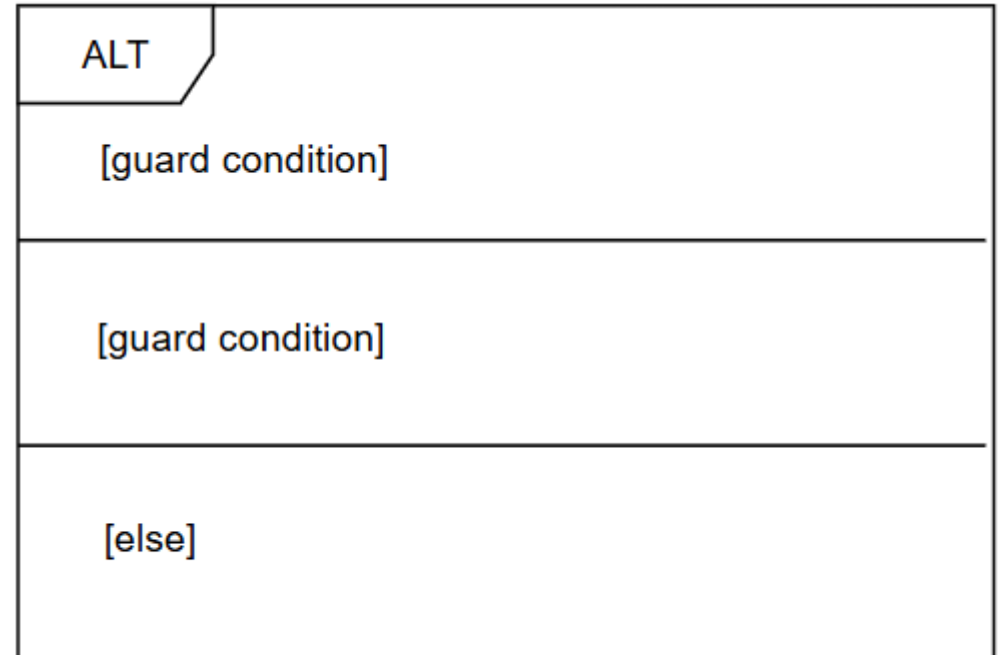
Then MovieManager request the details from the **MovieDB** and pass to the customer.

Then customer can request to book with MovieID and MovieManager generate the ticket and send to the customer. Then, the MovieManager update the seat count via the **SeatInfo** class.

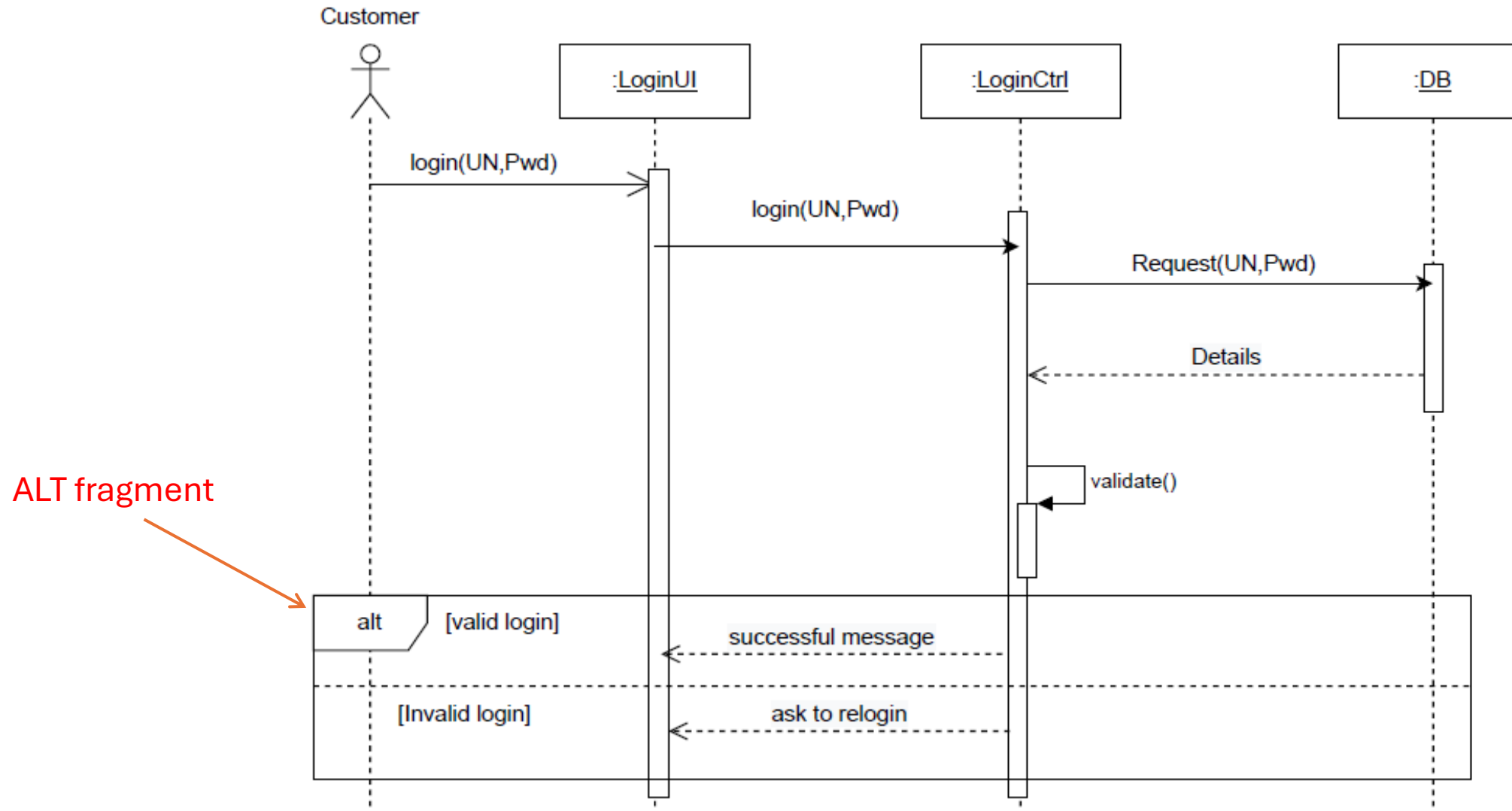
**Assume that the logging process already designed and you have to use it in your diagram.**

# Alternative Fragment (ALT Tag)

- Fragment with **two or more alternatives** to execute with **guard conditions**
- One time ,one of the alternatives will be true or else there will be else clause to execute



# Example: ALT Fragment

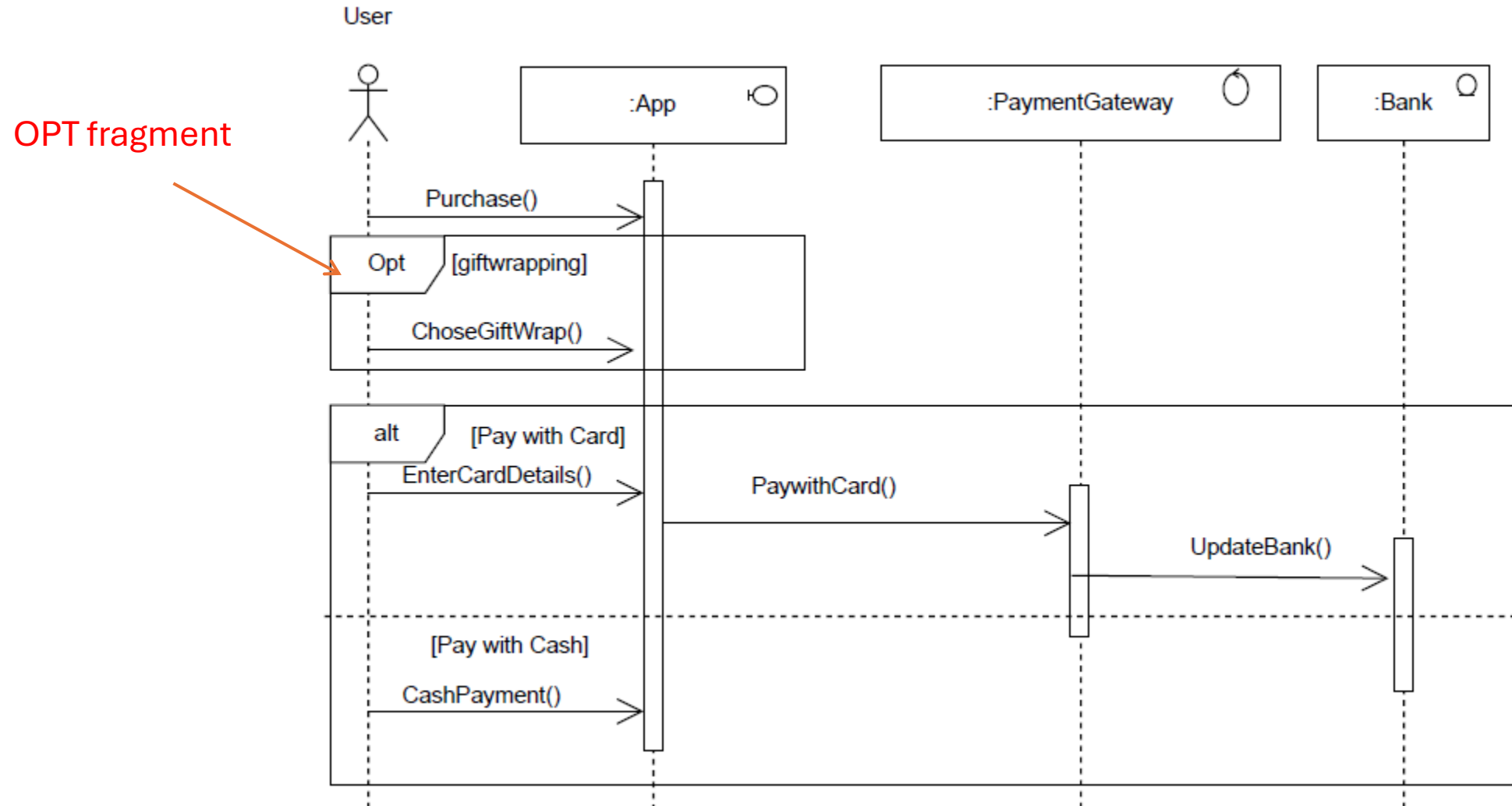


# Optional Fragment (OPT Tag)

- Handle a **single condition situation**. This looks like an alt that offers only one interaction.
- If the guard condition fails, the behavior is **simply skipped**



# Example: OPT Fragment





# Activity 04

“ABC” cinema has a eTicket booking system.

Customer can request a particular movie details via **MovieUI** and the request will pass to the **MovieManager**.

Then MovieManager request the details from the **MovieDB** and pass the results to the customer.

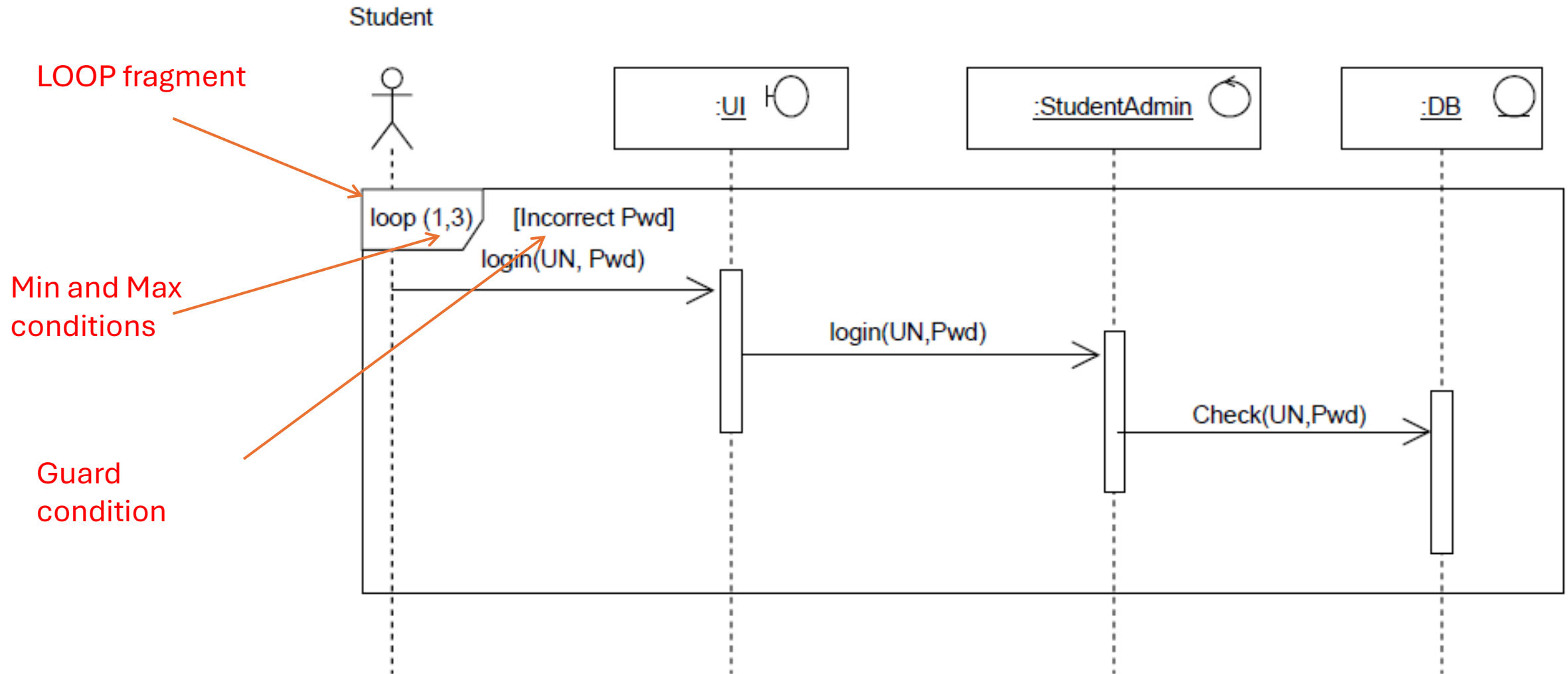
Only if the requested movie is available, the customer can request to book with MovieID and MovieManager generate the ticket and send to the customer.

Draw a sequence diagram to the above scenario.

# LOOP Fragment (Loop Tag)

- Represent a repetitive sequence with guard condition.
- The **guard condition** in a loop fragment can have two other special conditions tested against.
  - **minimum iterations (min int)**
  - **maximum iterations (max int).**
- When the guard condition become false, the loop ends.

# Example: Loop Fragment



# Activity 05

“ABC” cinema has a eTicket booking system. Customer can request a particular movie details via MovieUI and the request will pass to the MovieManeger.

Then MovieManager request the details from the MovieDB and pass the results to the customer.

Only if the requested movie is available, the customer can request to book with MovieID and MovieManager generate the ticket and send to the customer.

**In the same way, customer can make any number of requests for eTickets for different movies.**

# How Do you Evaluate Your Sequence Diagram?

## Checklist:

- Are all actors & objects included?
- Are messages well-ordered?
- Does it reflect real scenario?
- Is it readable and reusable?

# Errors and Consequences of Sequence Diagram?

- Missing messages → miscommunication
- Wrong logic in fragment → incorrect flow
- Leads to expensive redesign in later SDLC stages

# Learnings

- **Behavioral modeling** in UML
- Describe purpose and components of **sequence diagrams**
- Main characteristics of sequence diagram
  - **Frame**
  - **Actors**
  - **Objects/Stereo types/Lifeline/Activation Bar**
  - **Messages (Asynchronous/Synchronous/Self)**
  - **Fragments (Ref/Loop/Opt/Alt)**
- Construct sequence diagram for real world scenario
- **Evaluate the correctness** of interaction diagrams

# References

- Sommerville, Software Engineering, 10th edition, Pearson Education Limited, 2016
- Russ Miles, Kim Hamilton, Learning UML 2.0, O'Reilly Media, Inc, 2009
- IEEE Computer Society, SWEBOK: Guide to the Software Engineering Body of Knowledge, IEEE Computer Society Publications, 2014
- UML 2 Bible : Chapters 8 & 9
- Applying UML and Patterns by Craig Larman : Chapter 15
- TheElementsofUML2Style: Chapter 7



# Thank You