

Object Oriented Methods with UML



A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Lecture -4

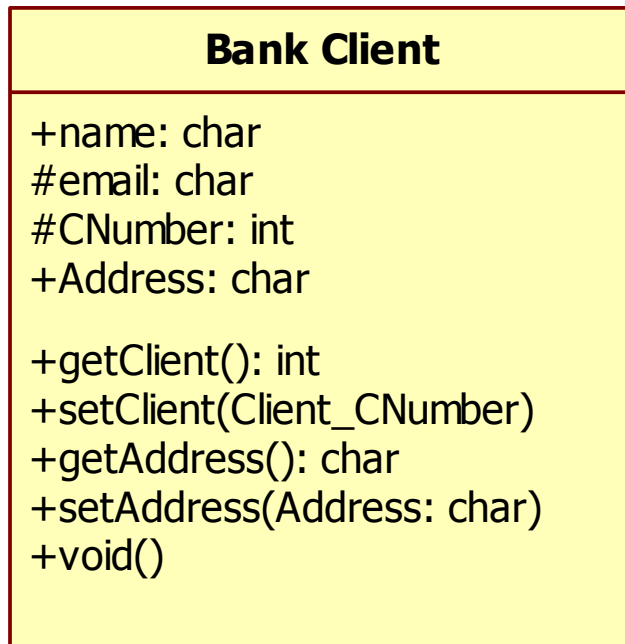
■ Topics

- Class diagram with sample code
- Interaction diagram
 - Sequence Diagram
 - Collaboration Diagram

Class Diagram with sample code



-



```
class Bank Client {  
public:  
char name;  
char Address;  
int getClient();  
void setClient(int Client_CNumber);  
char getAddress();  
void setAddress(char Address);  
void void();  
protected:  
char email;  
int CNumber;  
};
```

Interaction Diagrams



- Assist in understanding how a system (i.e., a use case) actually works
- Verify that a use case description can be supported by the existing classes
- Identify responsibilities/operations and assign them to classes
- To capture dynamic behavior of a system.
- To describe the message flow in the system.

Types of Interaction diagram



- **Sequence Diagram**

- ❖ Emphasizes on time sequence of messages

- **Collaboration Diagram**

- ❖ Emphasizes on the structural organization of the objects that send and receive messages.



Sequence Diagram



- Used to show how objects interact in a given situation

Two major components

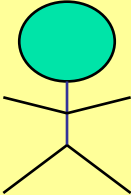





1)Active objects

2)Communications between these active objects

-Messages sent between the active objects



Notations used in Sequence Diagram

AN ACTOR	
AN OBJECT	
A LIFELINE	
A FOCUS OF CONTROL	
A MESSAGE	
OBJECT DESTRUCTION	

Active objects



- Any objects that play a role in the system participate by sending and/or receiving messages
- Placed across the top of the diagram

Can be:

- An actor (from the use case diagram)
- Object/class (from the class diagram) within the system

Communications between Active Objects



A . P . U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Messages

- ✓ Used to illustrate communication between different active objects of a sequence diagram

Used when an object needs

- to activate a process of a different object
- to give information to another object

Types of Messages

- ✓ Synchronous (flow interrupt until the message has completed)



- ✓ Asynchronous (don't wait for response)



- ✓ Flat (no distinction between synn/async)



- ✓ Return (control flow has returned to the caller) **optional**



Object Creation

An object named 'student',
its type is not specified

`student`

An anonymous instance
of type Student

`:Student`

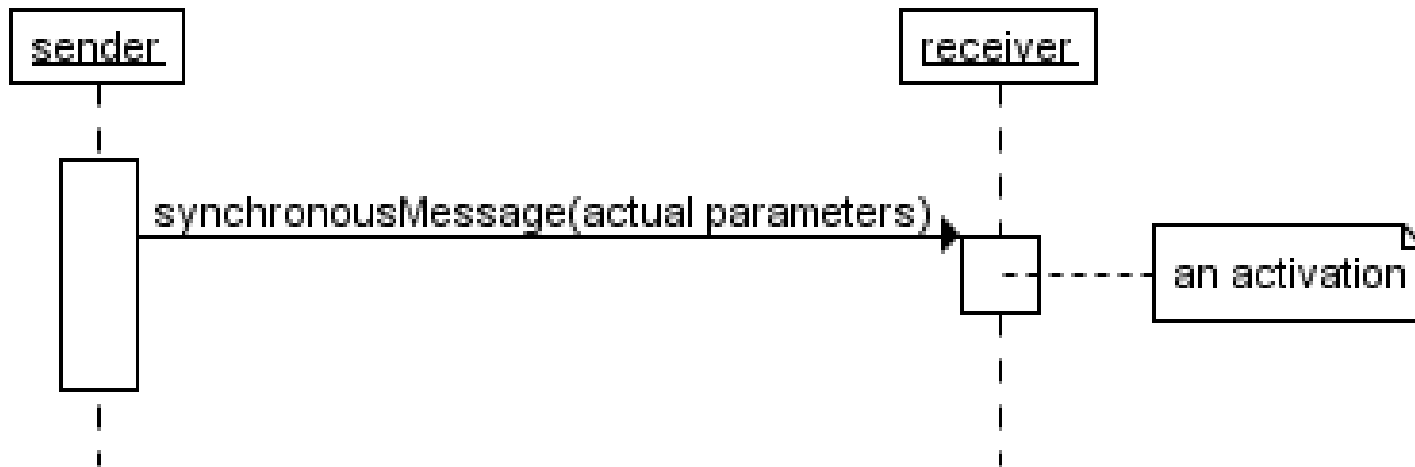
An instance of type
Student, named 's'

`s.Student`

synchronous message

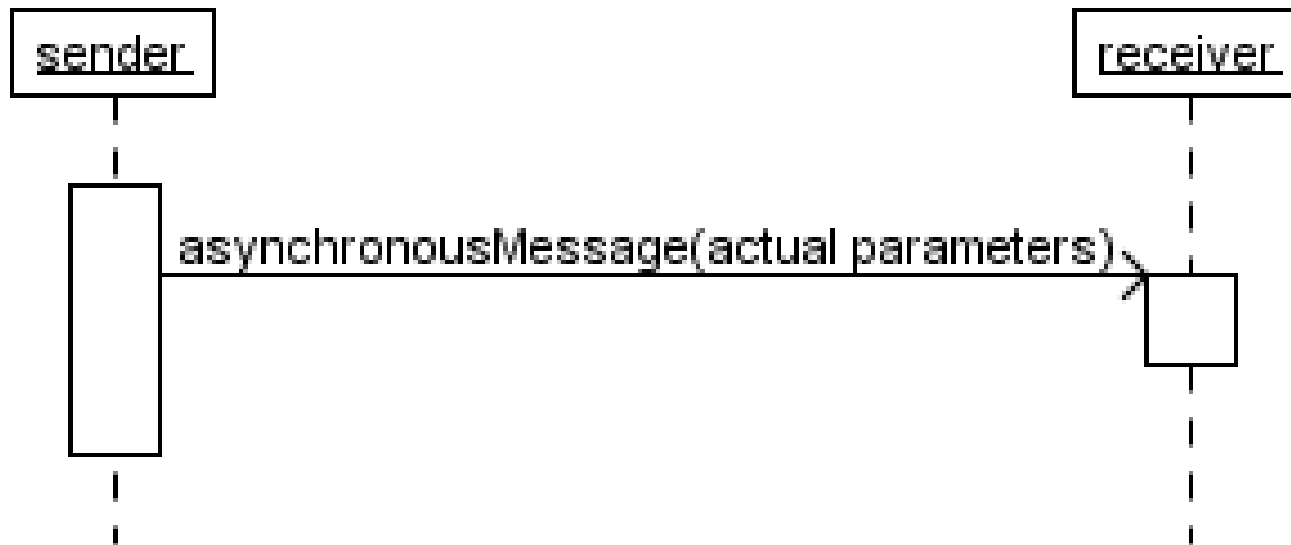


- A synchronous message is used when the sender waits until the receiver has finished processing the message, only then does the caller continue (i.e. a blocking call).



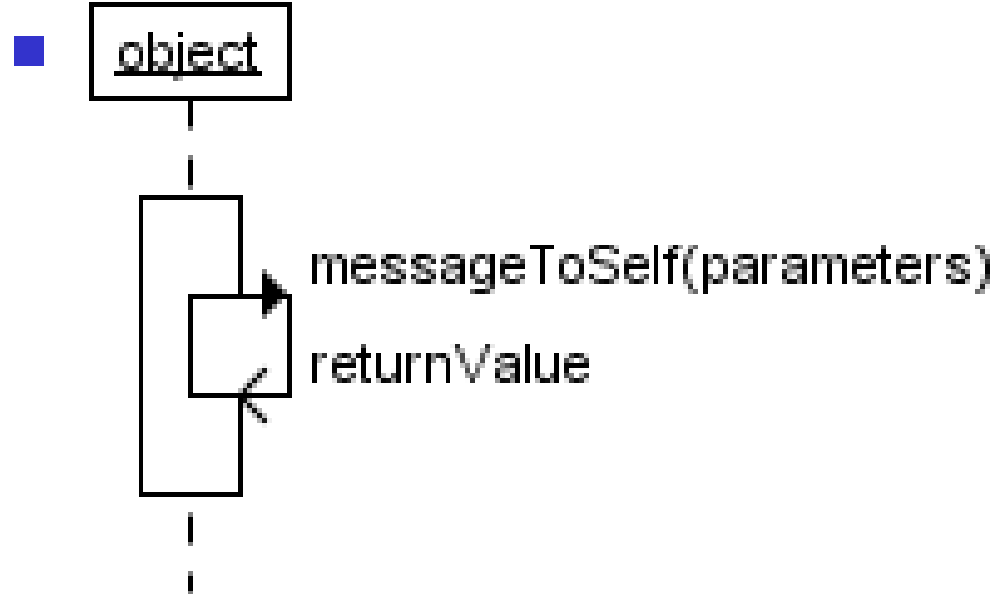
Asynchronous messages

- With an asynchronous message, the sender does not wait for the receiver to finish processing the message, it continues immediately.
- Messages sent to a receiver in another process or calls that start a new thread are examples of asynchronous messages.



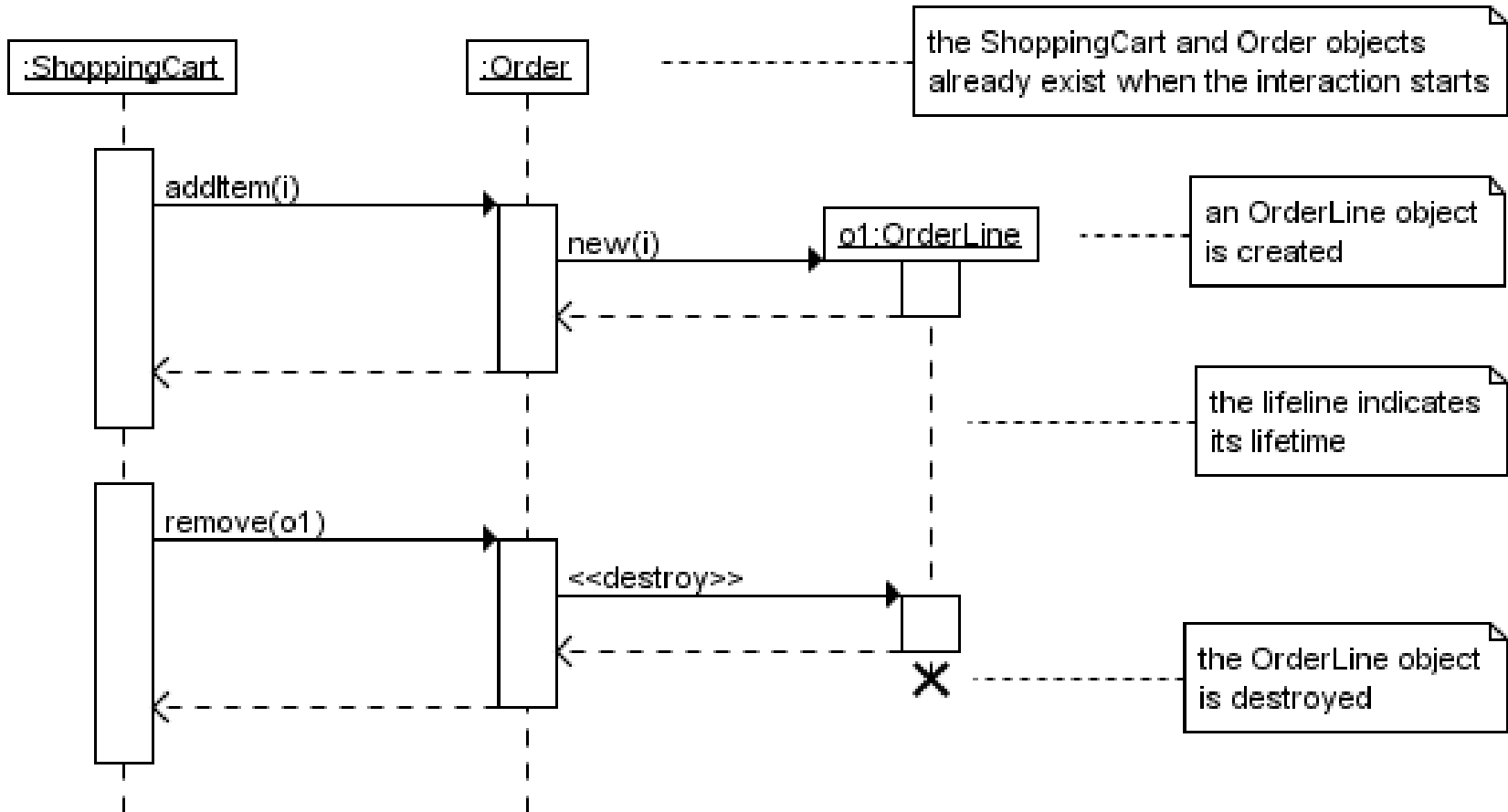
Message to self

- A message that an object sends itself can be shown as follows :



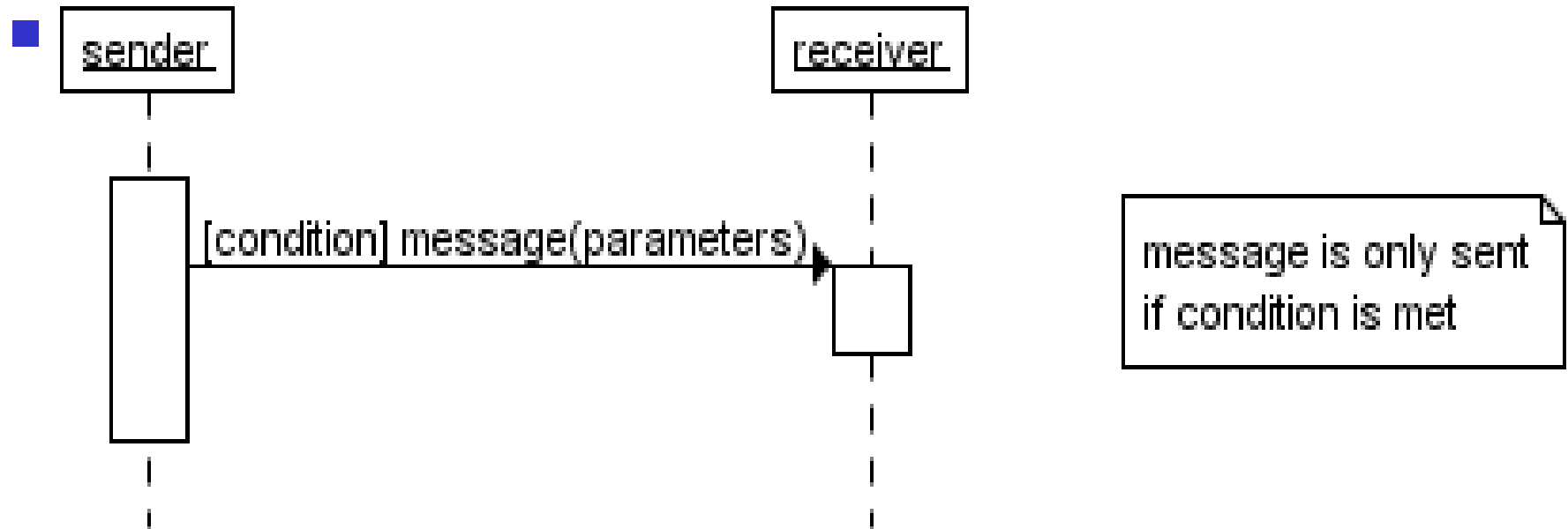
a message to self
and its return value

Creation and destruction



Conditional interaction

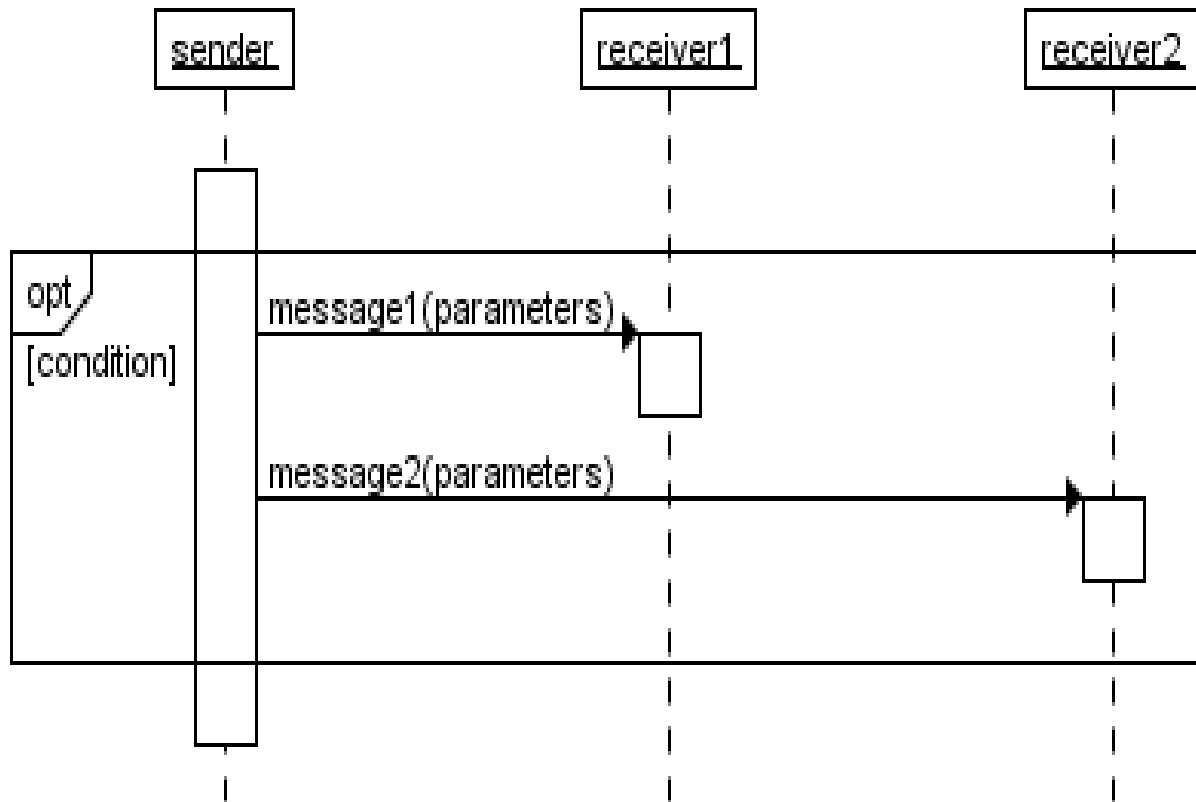
- A message can include a guard, which signifies that the message is only sent if a certain condition is met. The guard is simply that condition between brackets.



Combining several messages (If –condition)

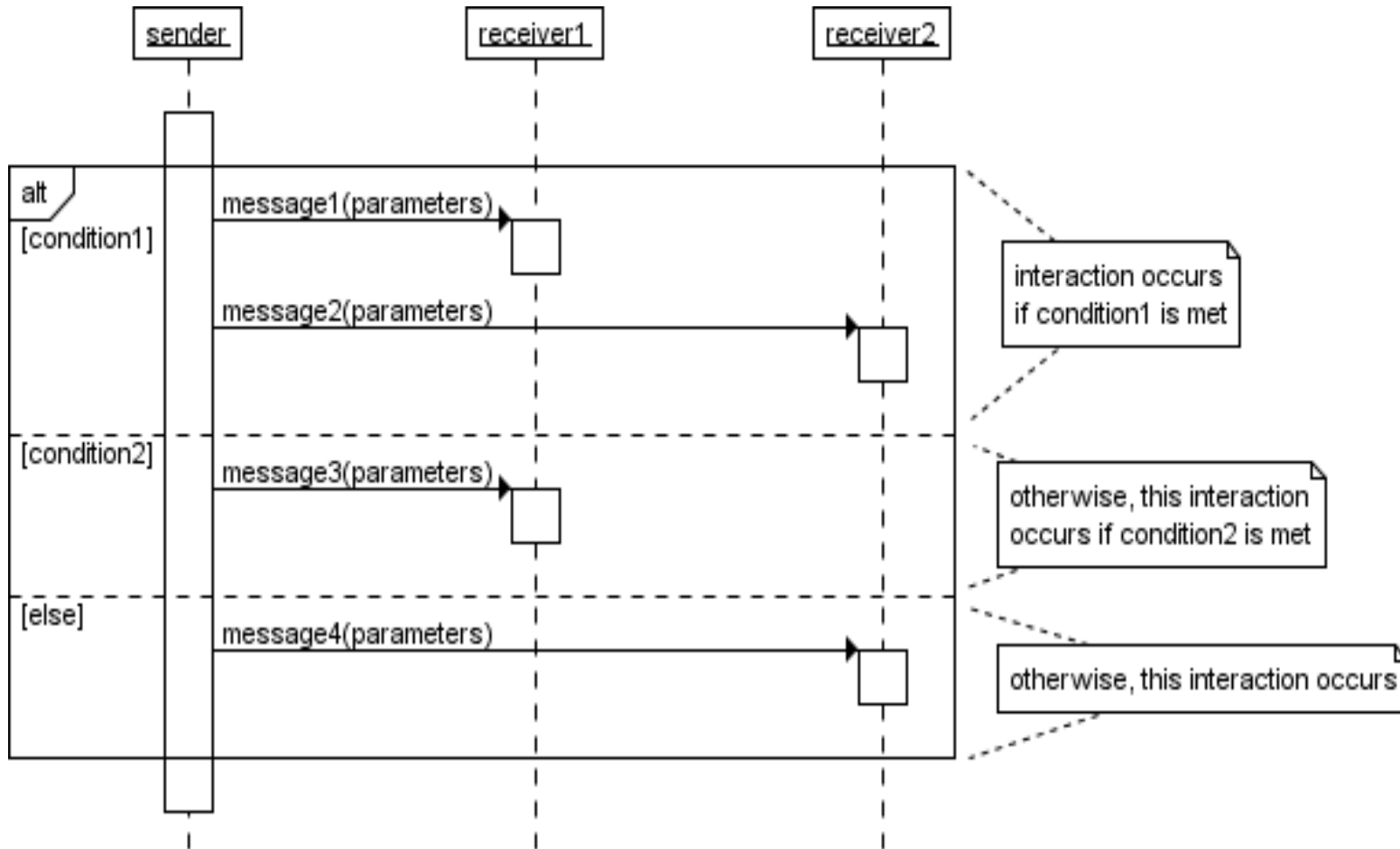


A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

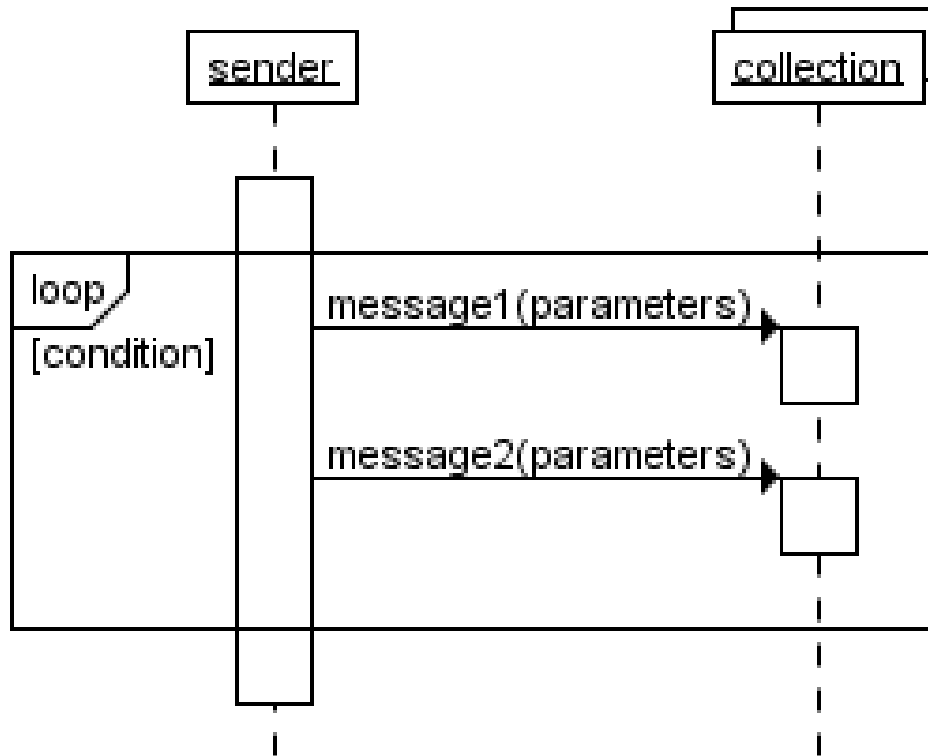


If condition is met,
both messages are sent

Combining several messages (If-else condition)



Repeated Interaction (Loops)



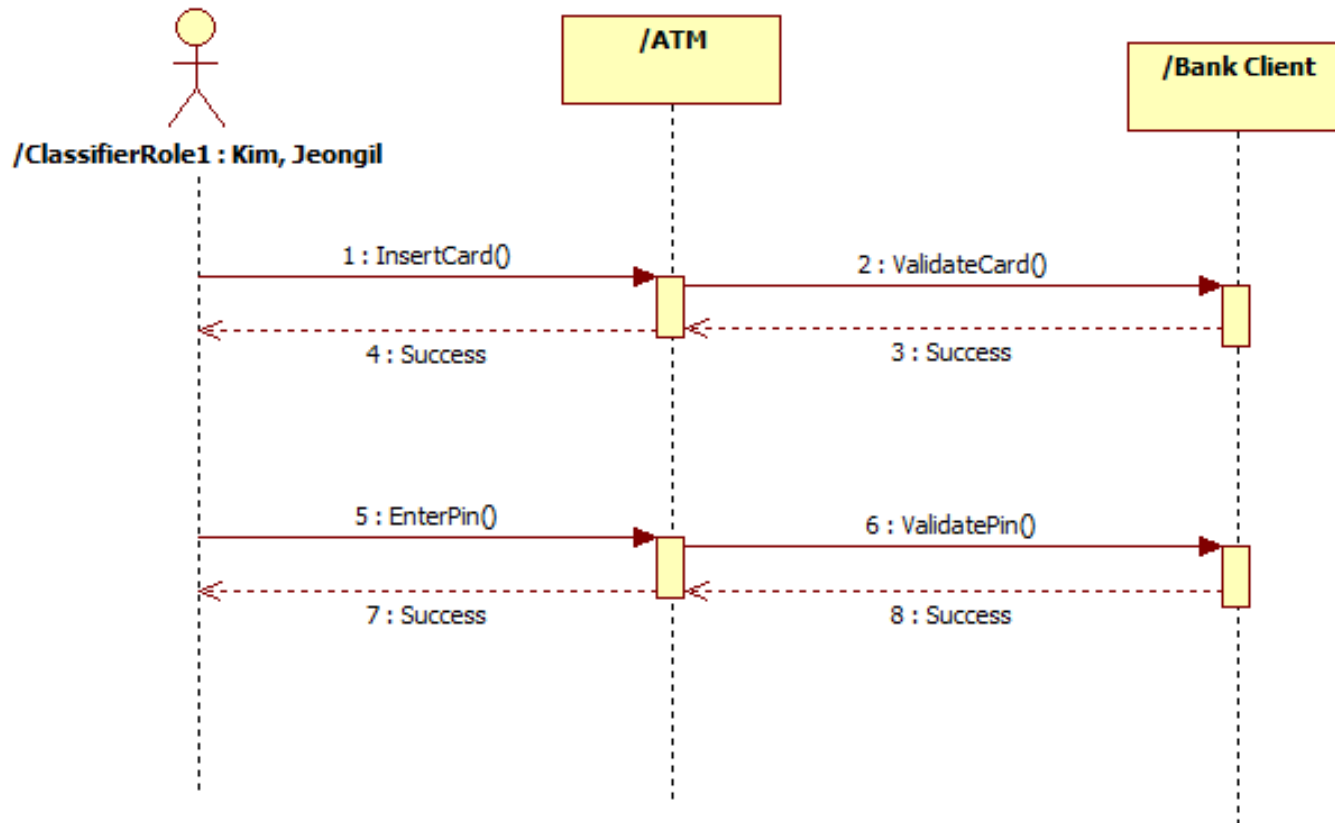
Both messages are sent as long as condition is met

Sequence Diagram

Use case:Login (Positive scenario)



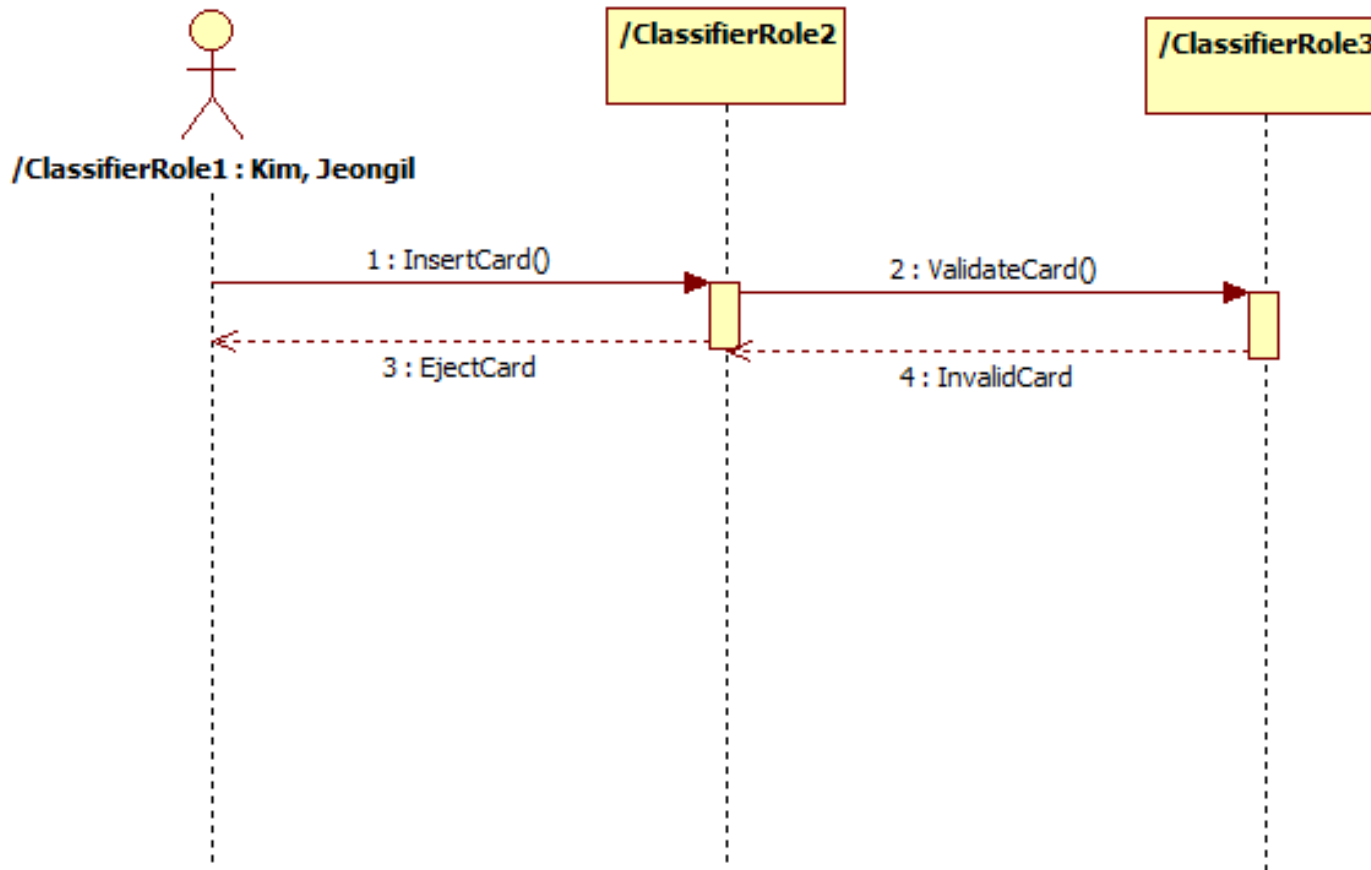
A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION





Sequence Diagram

Use case: Login (Negative scenario)



Steps for building sequence diagram



Identify the objects that may participate in the implementation of this use case by completing the supplied message table.

a) List candidate objects.

- Use case controller class
- Domain classes
- Database table classes
- Display screens or reports

Collaboration Diagram Elements



- There are three primary elements of a collaboration diagram:
 - Objects
 - Links
 - Messages

Collaboration Diagram

- Objects

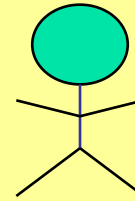
rectangle containing the object signature .

object signature:

- **object name : object Class**
- object name (optional) - starts with lowercase letter
- class name (mandatory) - starts with uppercase letter

Notations used in Collaboration Diagram

AN ACTOR



AN OBJECT



anObject:aClass

A MESSAGE



aMessage()

Association



Links



- The connecting lines drawn between objects are links
- They enable you to see the relationships between objects

Messages

- An interaction is implemented by a group of objects that collaborate by exchanging messages
- An asterisk (*) indicates that a message runs more than once
- Flow by numbers
 1. Enter Borrower ID
 - 1.1 CalcAmtCanBorrow
 - 1.1.1 <<create>>
 - 1.1.2 CalcBorrowerFines
 - 1.1.3 GetBorrowersCheckedOutMedia
 - 1.1.4 IsMediaOverdue
 - 1.1.5 Amt Can Borrow
 - 1.2 Display Invalid User Msg

Collaboration Diagram

