Software Re-Engineering

Lecture: 06

Part-II

Dr. Imran Ali
School of Computing- Software Engineering
FAST-National University of Computer & Emerging Sciences, Karachi

Sequence [Todays Agenda]

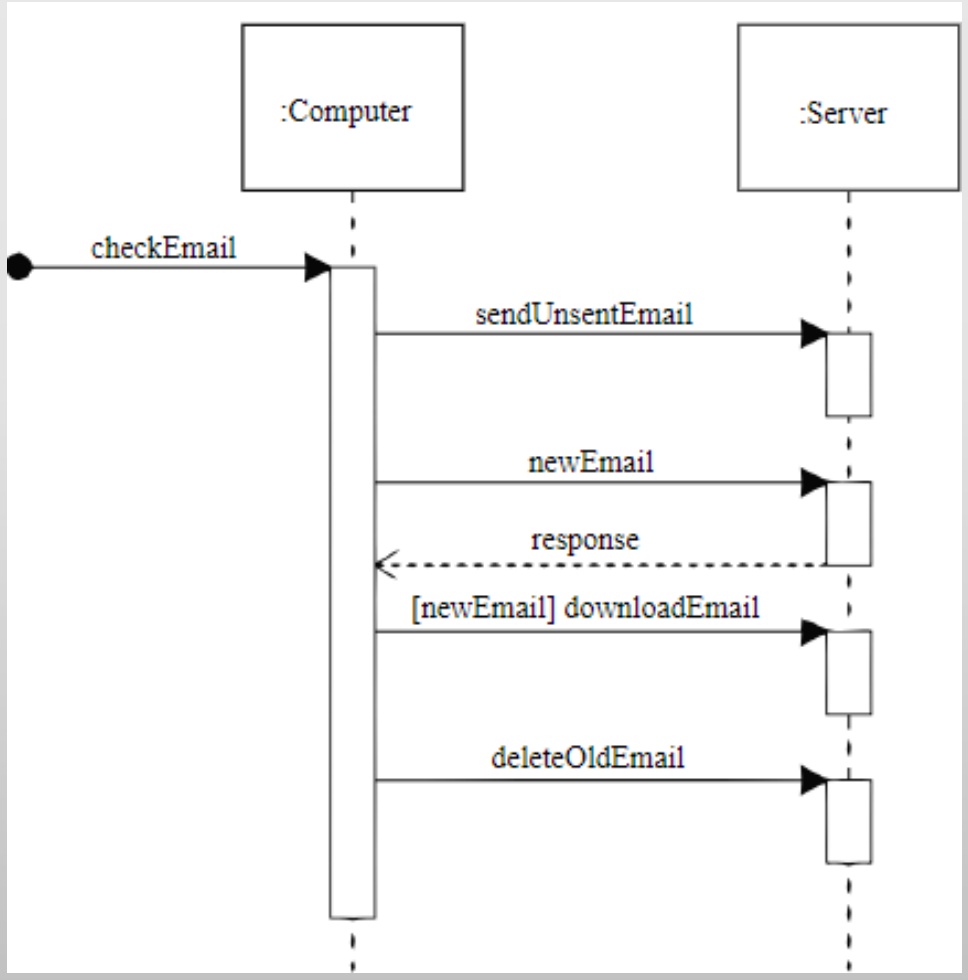
Content of Lecture

- Sequence Diagram
- System Sequence Diagram
- SSD vs SD
- Examples

Sequence Diagram

- A sequence diagram is the most commonly used interaction diagram.
- An interaction diagram is used to show interactive behavior of system.
- A sequence diagram represents the interaction between the objects in a sequential order i.e. the order in which these interactions occur.
- We can also use the terms event diagrams or event scenarios to refer to a sequence diagram.
- Sequence diagrams describe how and in what order the objects in a system function.
- These diagrams are widely used in businesses and by software developers to document and understand requirements for new and legacy systems.

Sequence Diagram-Example



Sequence Diagram

- Represent the object interactions in a given scenario identified for a given Use Case
- Specify messages passed between objects using horizontal arrows including messages to/from external actors
- Time increases from Top to bottom

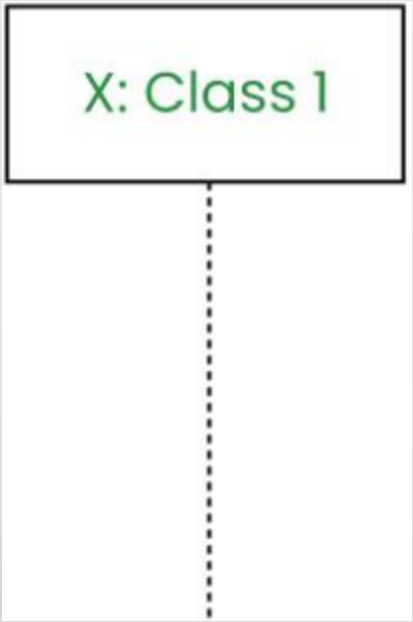
Sequence Diagram-Notations

- Actors
- Lifeline
- Messages
- Synchronous Messages
- Asynchronous Messages
- Create Message
- Delete Message
- Self Message
- Reply Message
- Found Message
- Lost Message
- Guards
- Activation Bar
- Alternative

Actors

- An actor represents a type of role where it interacts with the system and its objects.
- Actors are used to represent various roles including human users and other external subjects.
- We can have multiple actors in a sequence diagram.
- Represented by stick person.

Lifeline

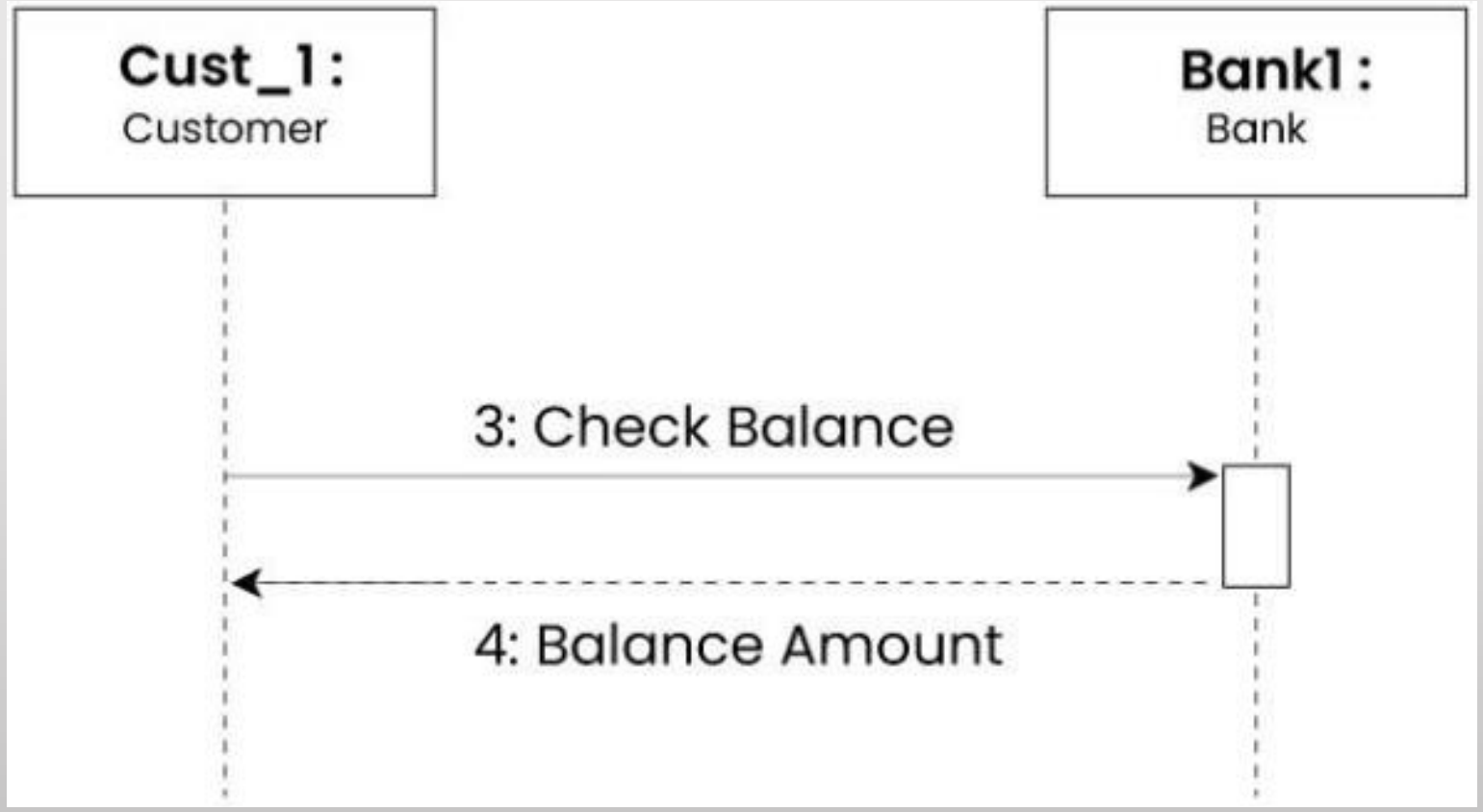


- A lifeline is a named element which represents an individual participant in a sequence diagram.
- Each instance in a sequence diagram is represented by a lifeline.
- Lifeline elements are located at the top in a
- sequence diagram.
- The standard in UML for naming a lifeline is given as:

Instance Name: Class Name

- In the given representation X is the object and Class 1 is Class name
- Lifeline is displayed in a rectangle called head with its name and type.
- The head is located on top of a vertical dashed line (referred to as the stem) as given here.

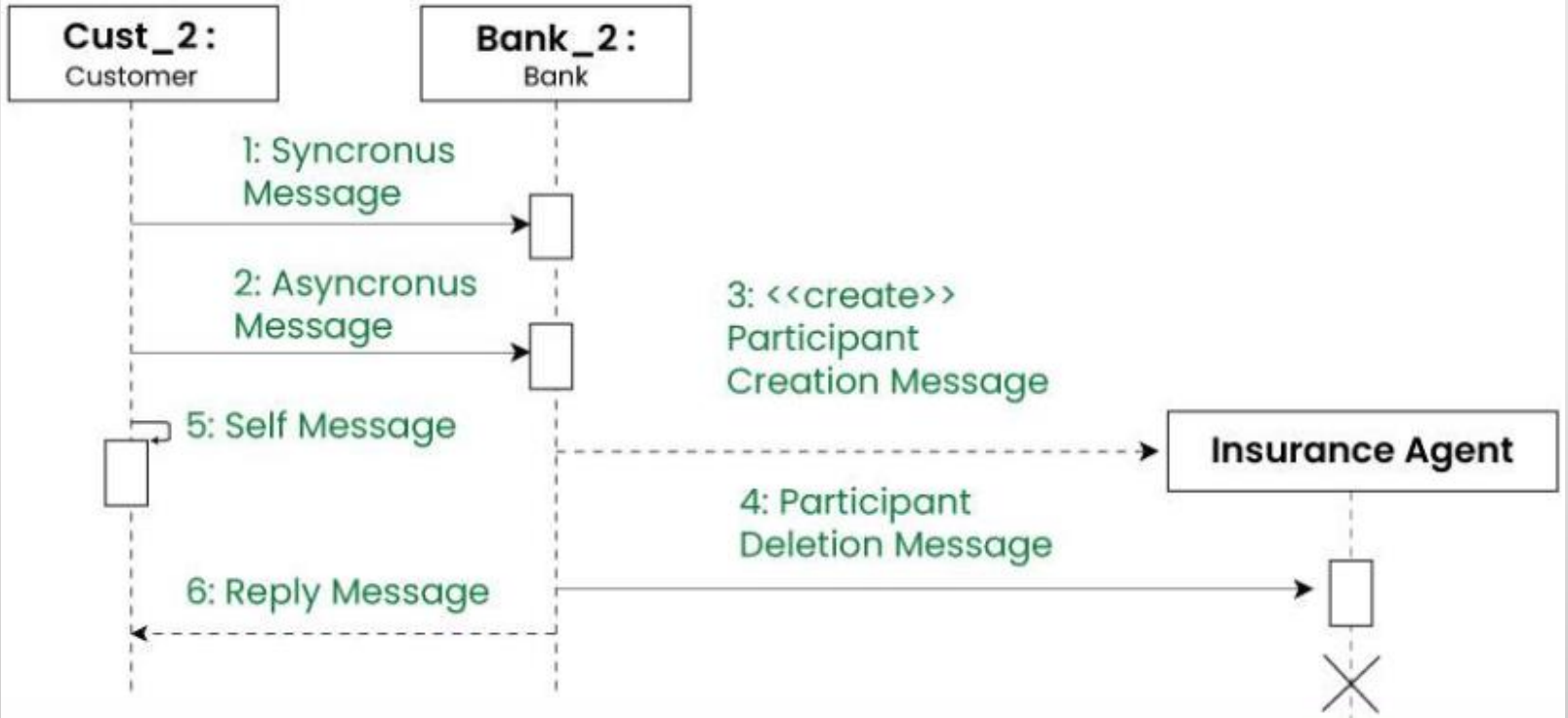
Lifelines- Example



Messages

- Communication between objects is represented by using messages.
- The messages appear in a sequential order on the lifeline.
- We represent messages using arrows.
- Lifelines and messages form the core of a sequence diagram.

Messages- Example



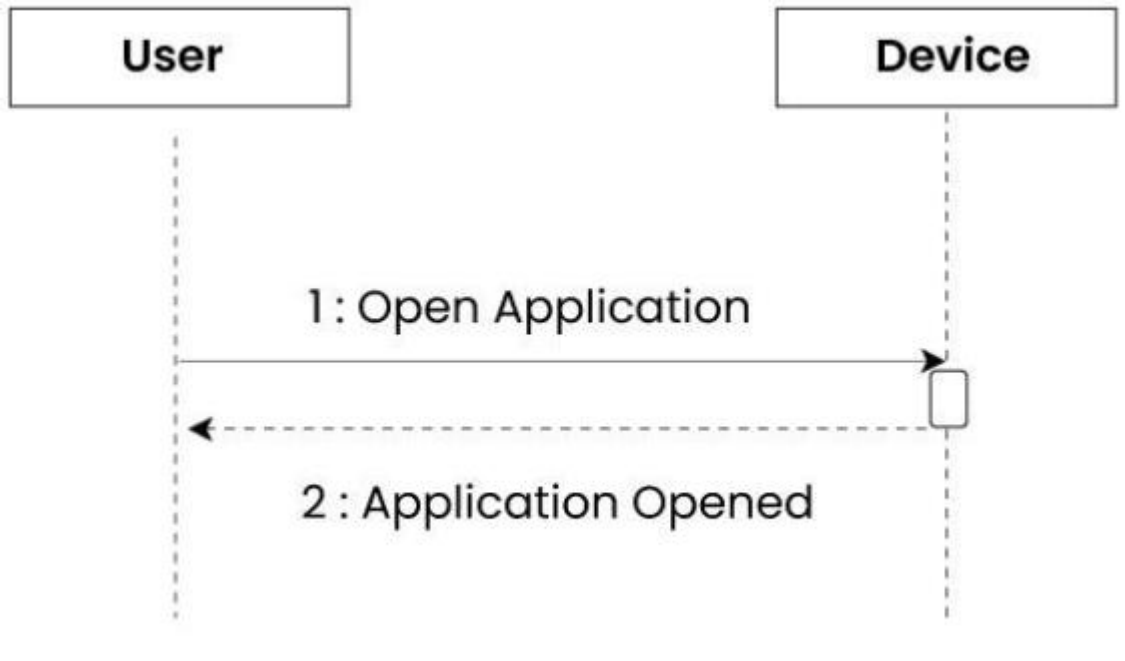
Messages

- Messages are classified into the following categories:
- Synchronous Messages
- Asynchronous Messages

Synchronous Message

- Synchronous message waits for a reply before the interaction can move forward.
- The sender waits until the receiver has completed processing of message.
- The caller continues only when it knows that the receiver has processed previous message i.e. it receives a reply message.
- A large number of calls in object oriented programming are synchronous.
- We use a solid arrow head to represent a synchronous message.

Synchronous Message- Example



Asynchronous Message

- Asynchronous message does not wait for a reply from the receiver.
- The interaction moves forward irrespective of the receiver processing previous message or not.
- We use a lined arrow head to represent an asynchronous message.

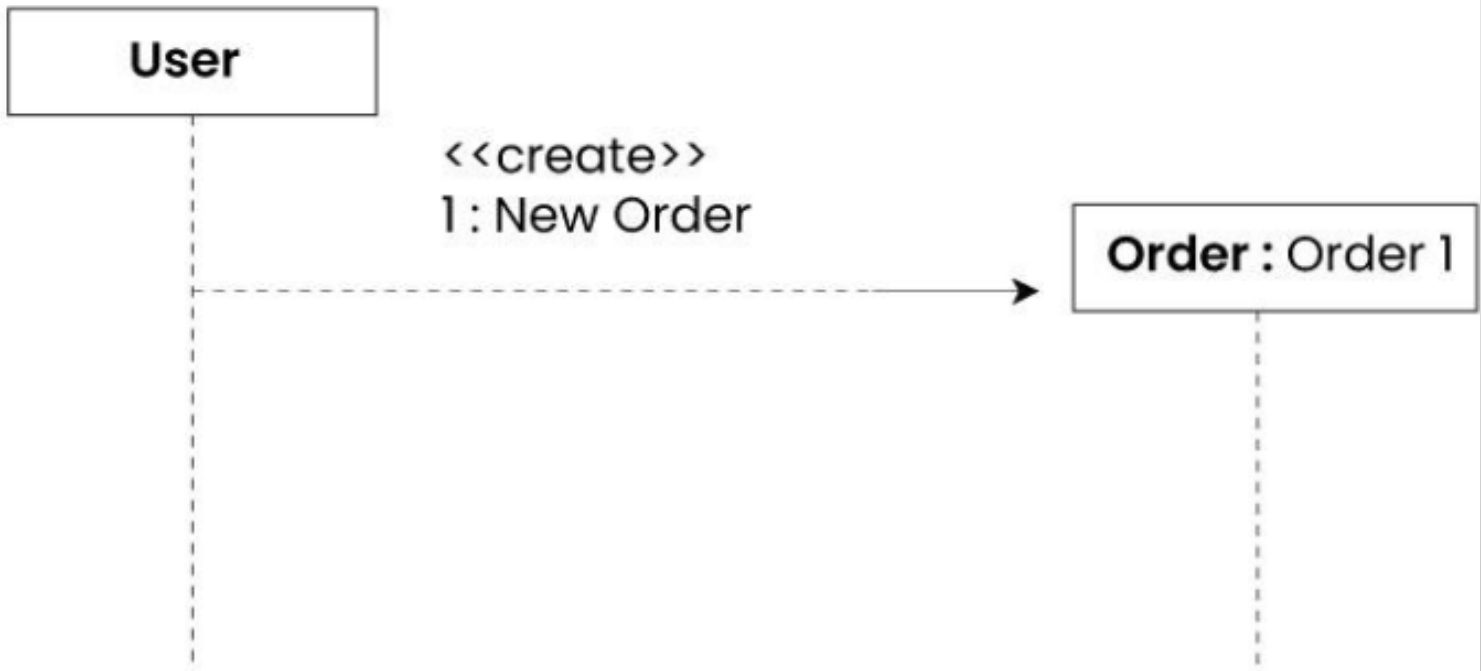
Asynchronous Message- Example

Create Message

- Create message is used to instantiate a new object in sequence diagram
- Represented with dotted arrow and create word labelled on it to specify that it is Create Message symbol

Create Message- Example

- The creation of a new order on a e-commerce website would require a new object of Order class to be created.

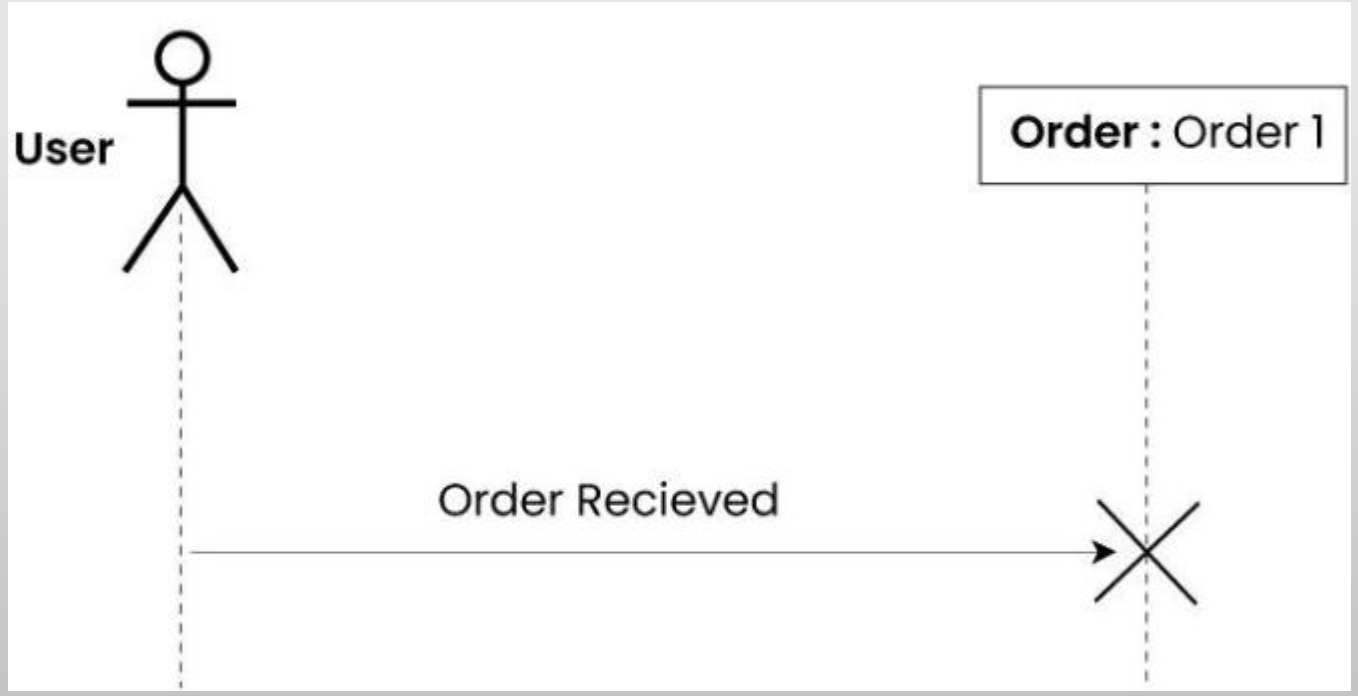


Delete Message

- Delete Message is used to delete an object.
- When an object is deallocated memory or is destroyed within the system we use the Delete Message symbol.
- It destroys the occurrence of the object in the system.
- It is represented by an arrow terminating with a x.

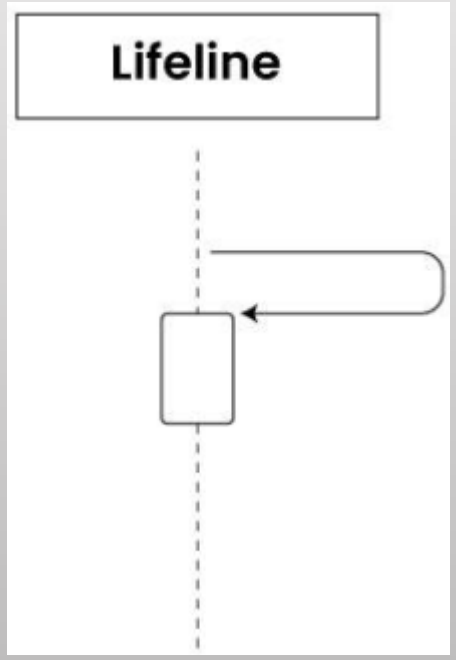
Delete Message- Example

- In the scenario below when the order is received by the user, the object of order class can be destroyed.



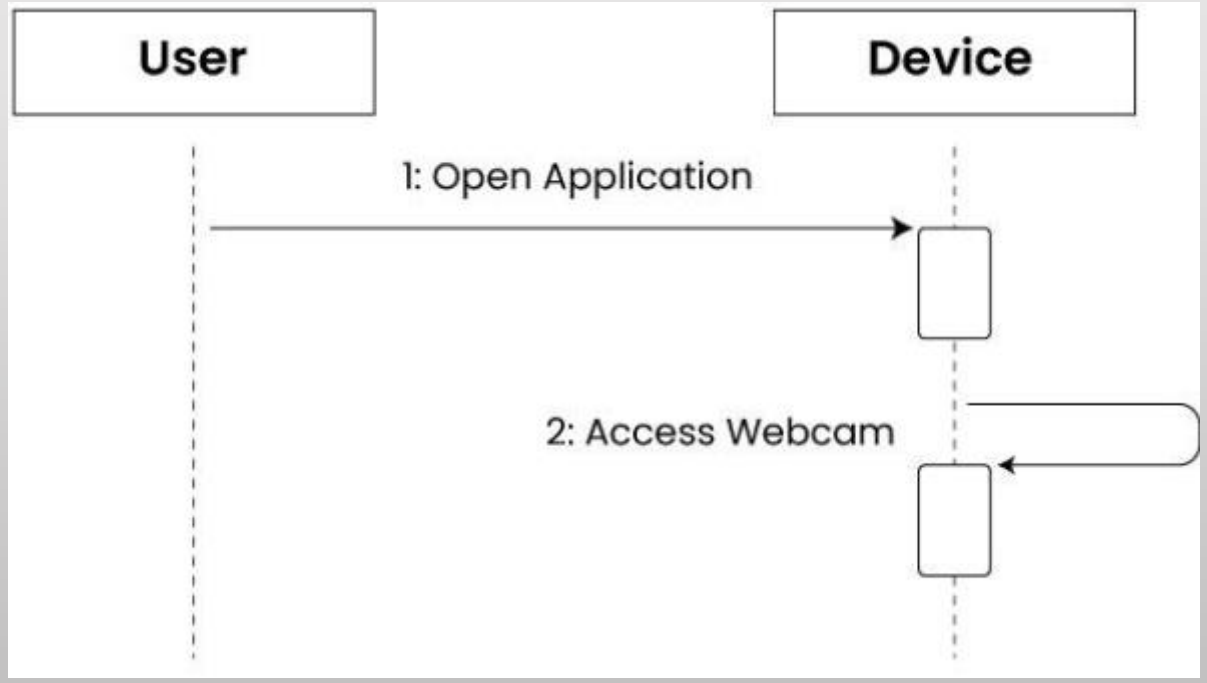
Self Message

- Certain scenarios might arise where the object needs to send a message to itself.
- Such messages are called Self Messages and are represented with a U shaped arrow.



Self Message- Example

- Consider a scenario where the device wants to access its webcam.
- Such a scenario is represented using a self message.

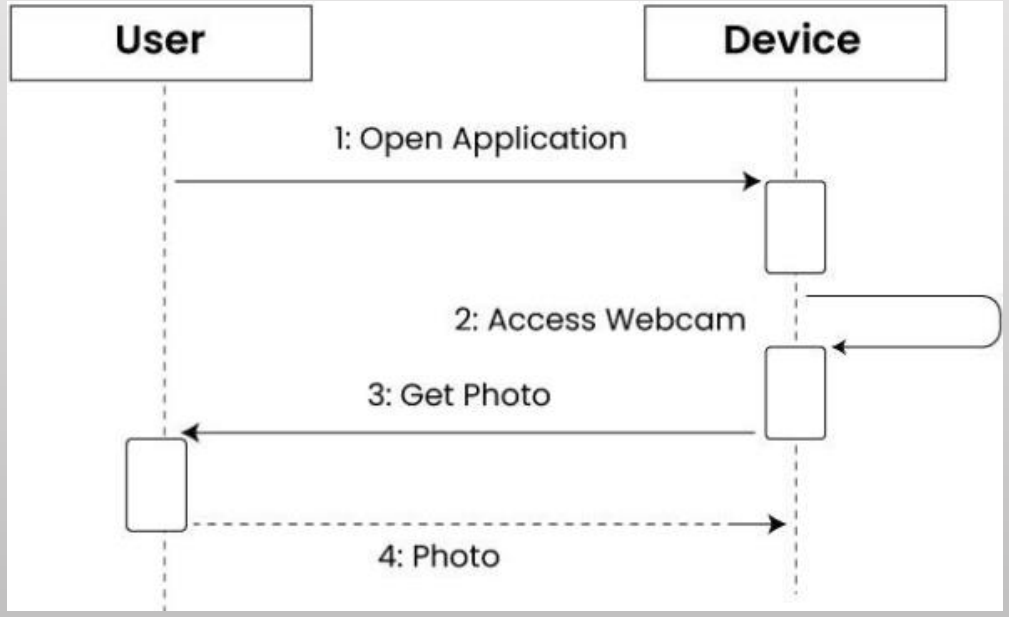


Reply Message

- Reply messages are used to show the message being sent from the receiver to the sender.
- We represent a return/reply message using an open arrow head with a dotted line.
- The interaction moves forward only when a reply message is sent by the receiver.

Reply Message- Example

- Consider the scenario where the device requests a photo from the user.
- Here the message which shows the photo being sent is a reply message.

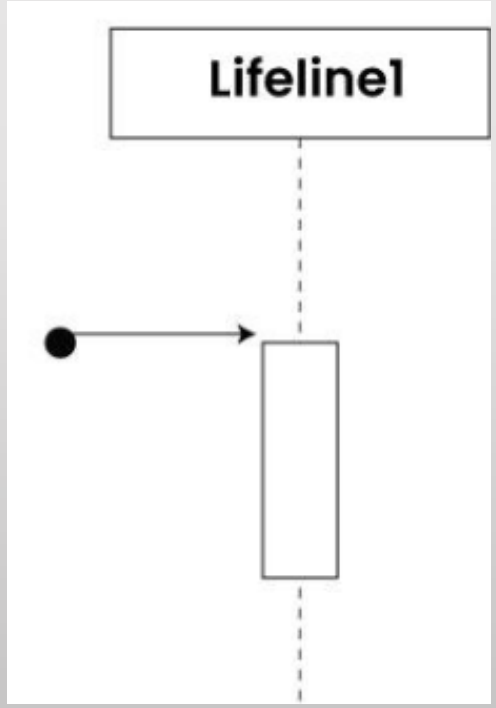


Found Message

- A Found message is used to represent a scenario where an unknown source sends the message.
- It is represented using an arrow directed towards a lifeline from an end point.

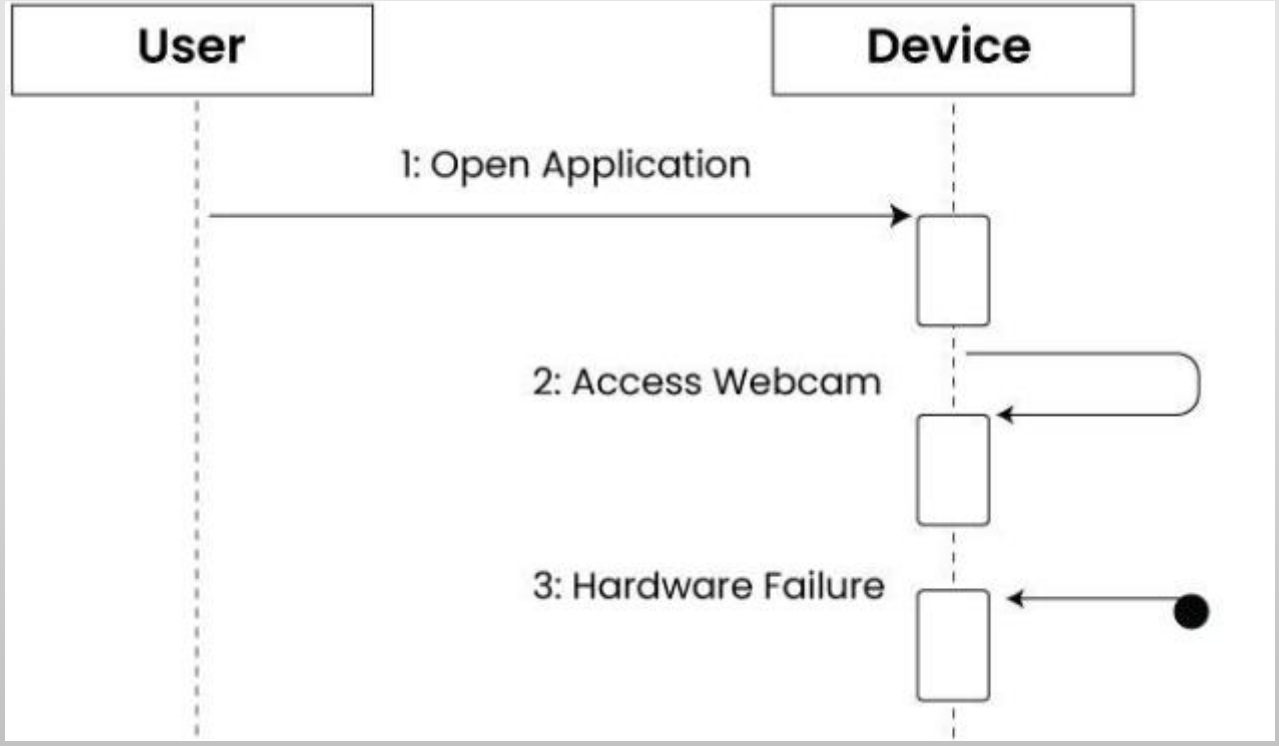
Found Message- Example

- Consider the scenario of a hardware failure.



Found Message- Example

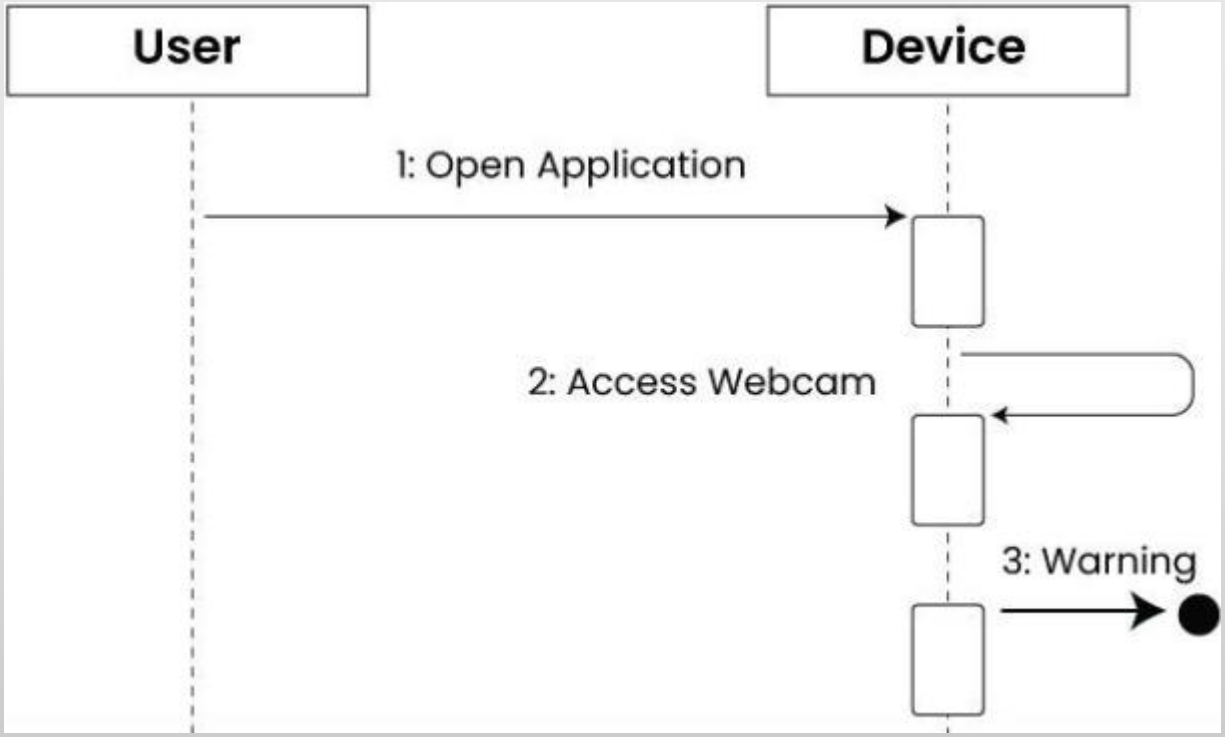
- It can be due to multiple reasons and we are not certain as to what caused the hardware failure.



Lost Message

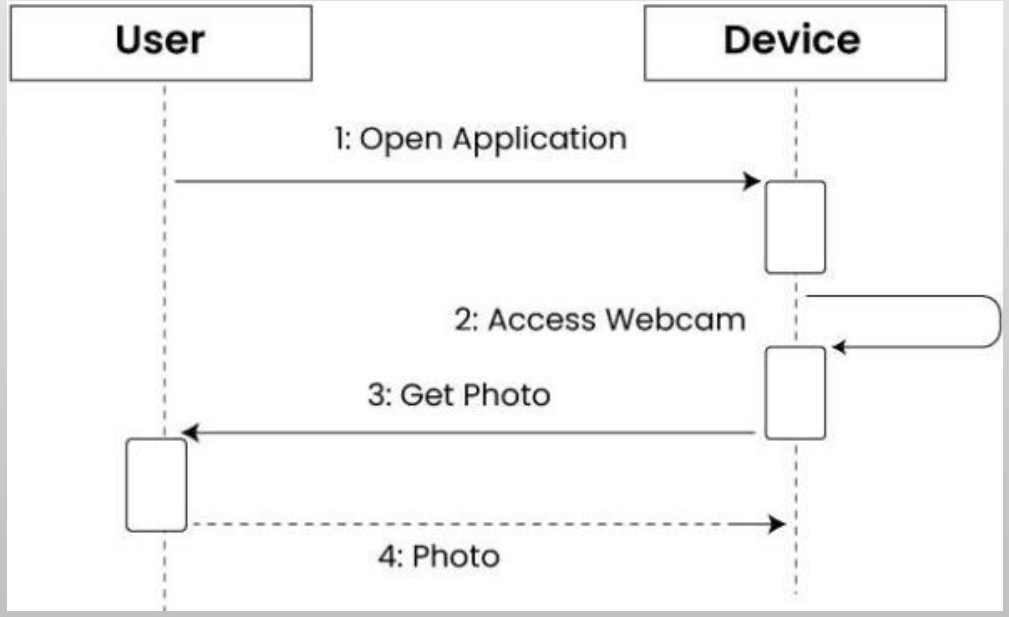
- A Lost message is used to represent a scenario where the recipient is not known to the system.
- It is represented using an arrow directed towards an end point from a lifeline.
- Consider a scenario where a warning is generated.

Lost Message- Example



Reply Message- Example

- Consider the scenario where the device requests a photo from the user.
- Here the message which shows the photo being sent is a reply message.

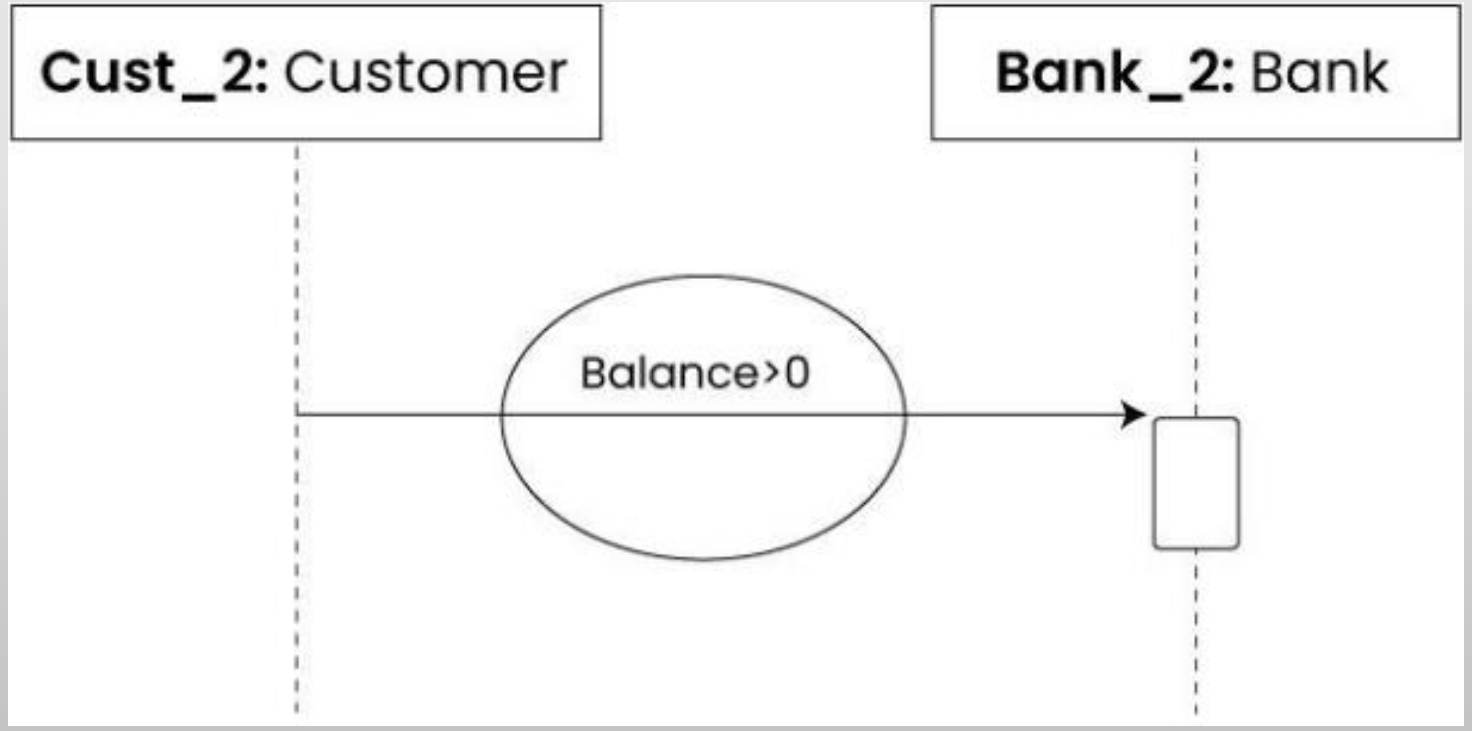


Guards

- To model conditions we use guards in UML.
- They are used when we need to restrict the flow of messages on the pretext of a condition being met.
- Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Guards- Example

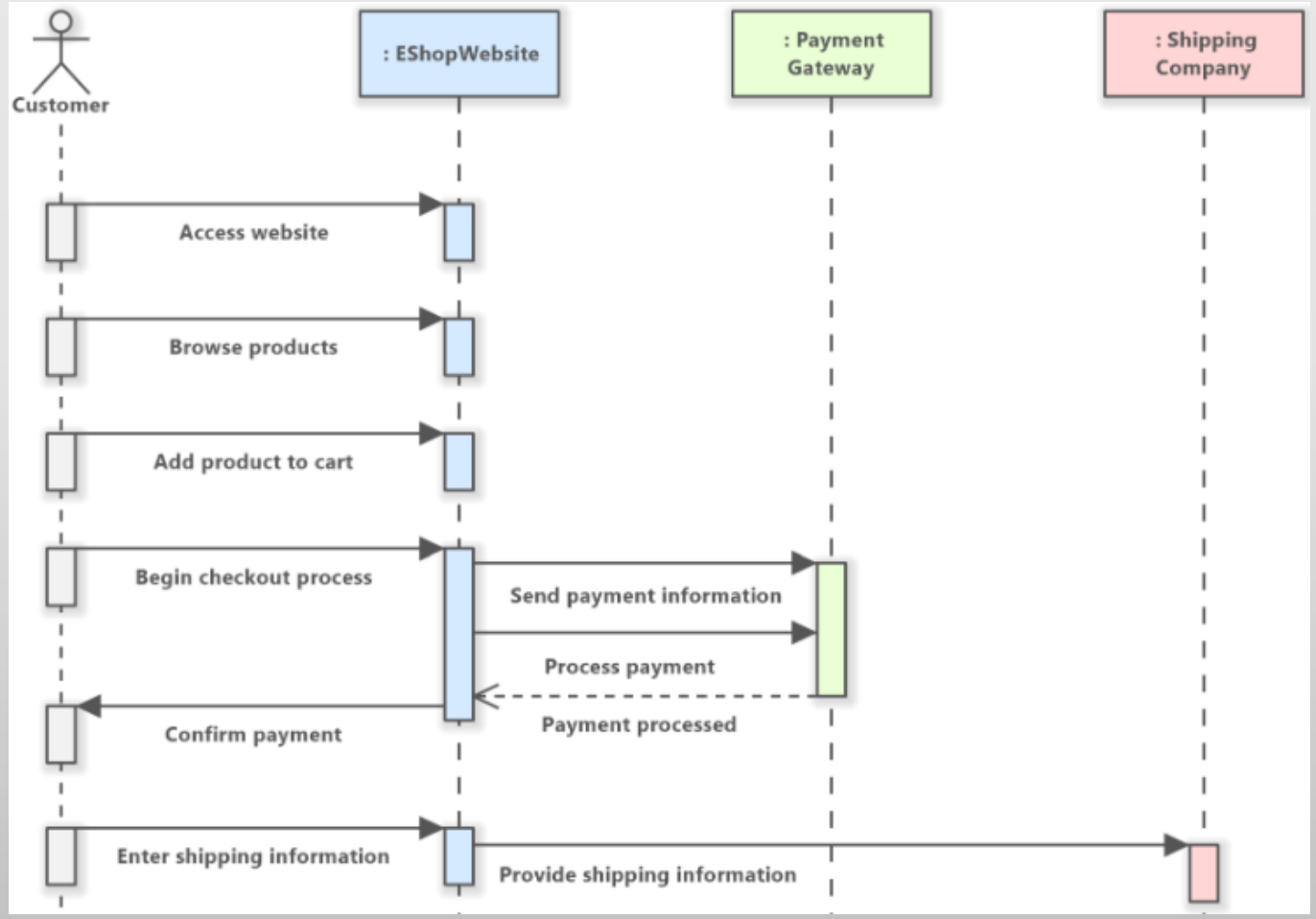
- In order to be able to withdraw cash, having a balance greater than zero is a condition that must be met as shown below.



Activation Bars

- Activation bars are thin rectangles over an object's lifeline, beginning when the object becomes active and ending when it's no longer active.
- These show the length of an object's activity or processing time.
- In an online store system, an activation bar for the payment processor could represent the time it takes to process a payment.

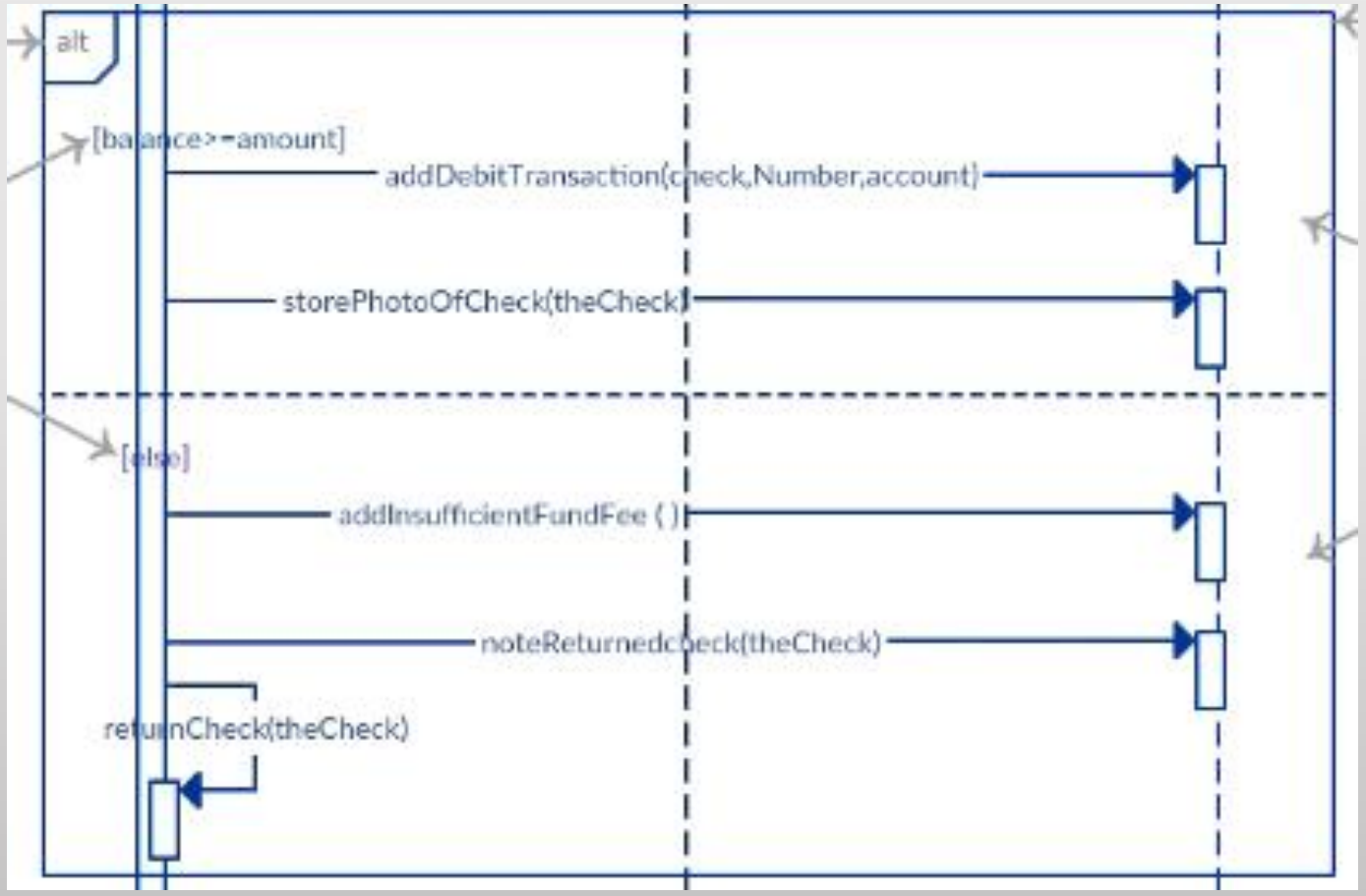
Activation Bars- Example



Alternative

- The alternative combination fragment is used when a choice needs to be made between two or more message sequences.
- It models the “if then else” logic.
- The alternative fragment is represented by a large rectangle or a frame; it is specified by mentioning ‘alt’ inside the frame’s name box.
- To show two or more alternatives, the larger rectangle is then divided into what is called interaction operands using a dashed line.

Alternative- Example



SSD vs Sequence Diagram

- A System Sequence Diagram (SSD) is an artifact that illustrates input and output events related to the system under discussion
- System Sequence Diagrams are typically associated with use case realization in the logical view of system development
- Sequence Diagrams (*Not System Sequence Diagrams*) display object interactions arranged in time sequence.

SSD- System Behavior

- System behavior is a description of what a system does, without explaining how it does it
- One part of that description is a System Sequence Diagram
- System behaves as “Black Box”
- Interior objects are not shown, as they would be on a Sequence Diagram.

System Sequence Diagram

- Use cases describe:
  - How actors interact with the software system,
  - Typical course of events that external actors generate, and
  - The order of the events
- When an actor interacts with the system, he generates system events, usually requesting some system operation to handle the event
  - For example, when a cashier enters an item's ID, the cashier is requesting the POS system to record that item's sale (the enterItem event). That event initiates an operation upon the system. The use case text implies the enterItem event, and the SSD makes it concrete and explicit.


Thank You!