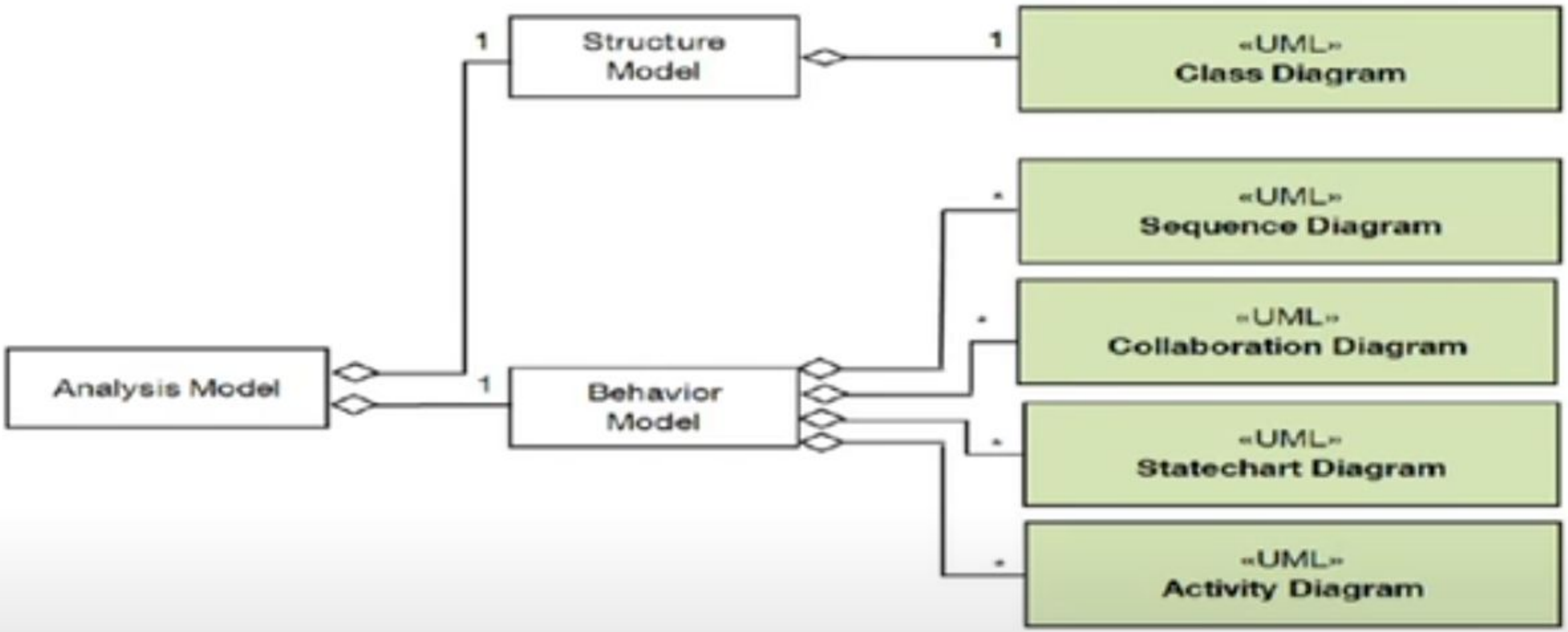
Object-Oriented Analysis and Design using JAVA

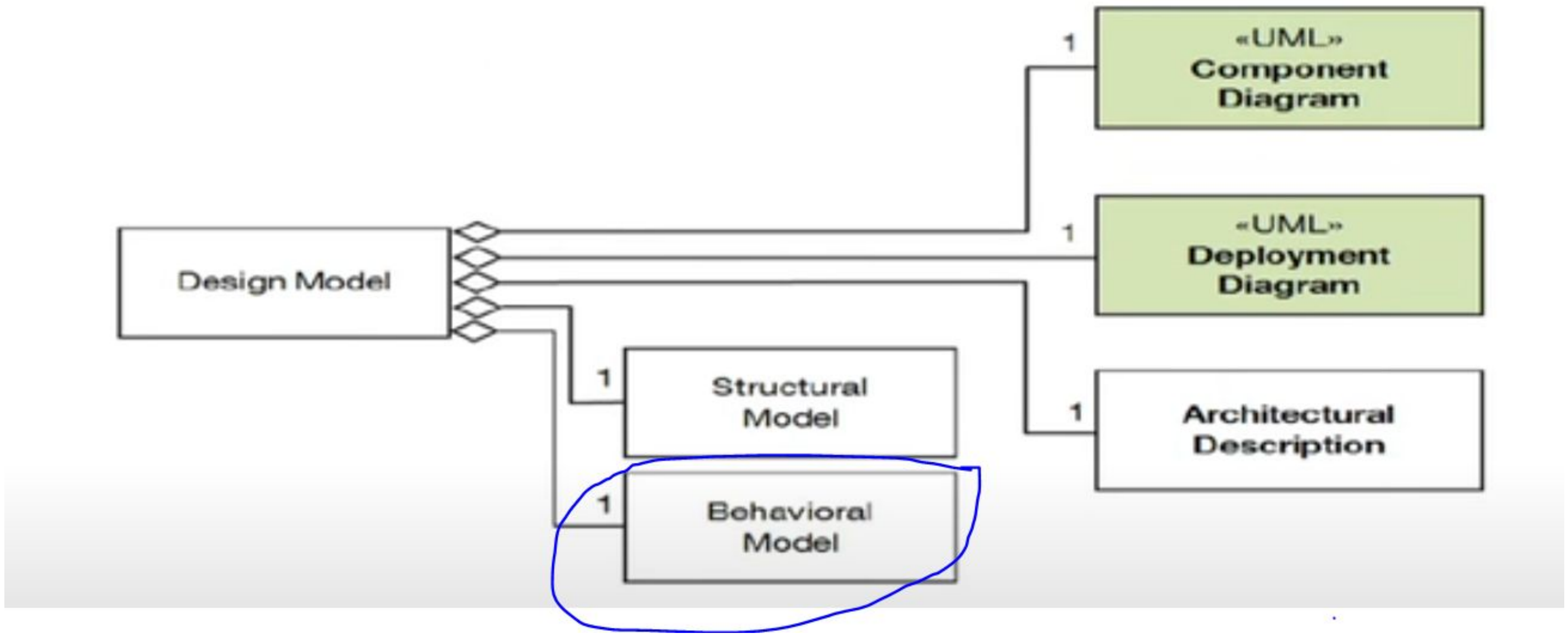
B.Tech (CSE/IT) 5th SEM
2021-2022

Lecture-18 Sequence diagram

Class diagram of SDLC



Class diagram of SDLC



Introduction

Sequence Diagrams –A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.

- It focuses on the message interchange between a number of lifelines.
- It depicts the inter-object **behaviour** of a system, ordered by time. (order of events ex. email)

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 by businessmen and software developers to document and understand requirements for new and existing systems.

Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
+

Introduction

- Conveys this information along the horizontal and vertical dimensions:
  - Vertical dimension
    - Top down, the time sequence of messages/calls as they occur
  - Horizontal dimension
    - Left to right, the object instances that the messages are sent to.

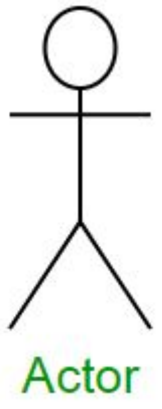
Emphasis on time ordering!

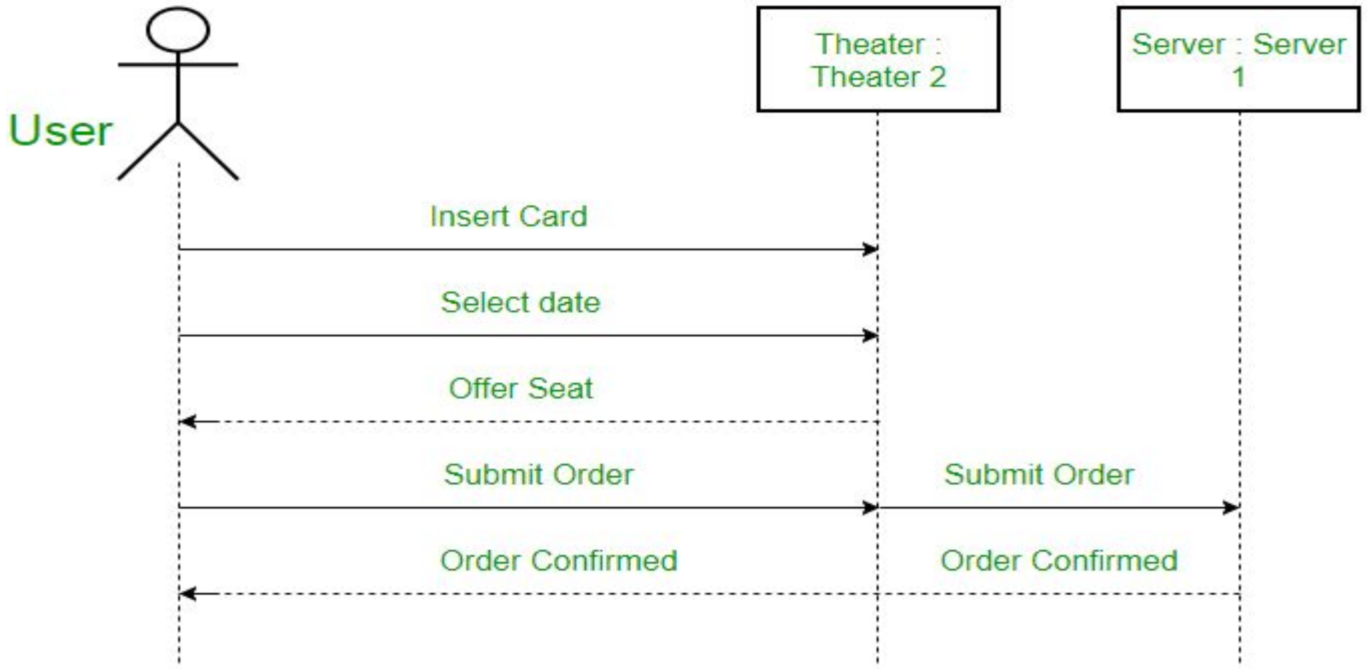
Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

+

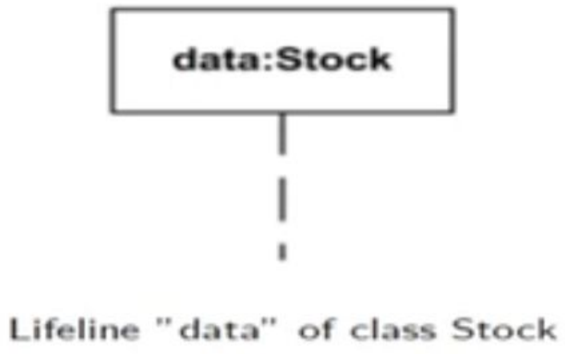
Sequence Diagram Notations

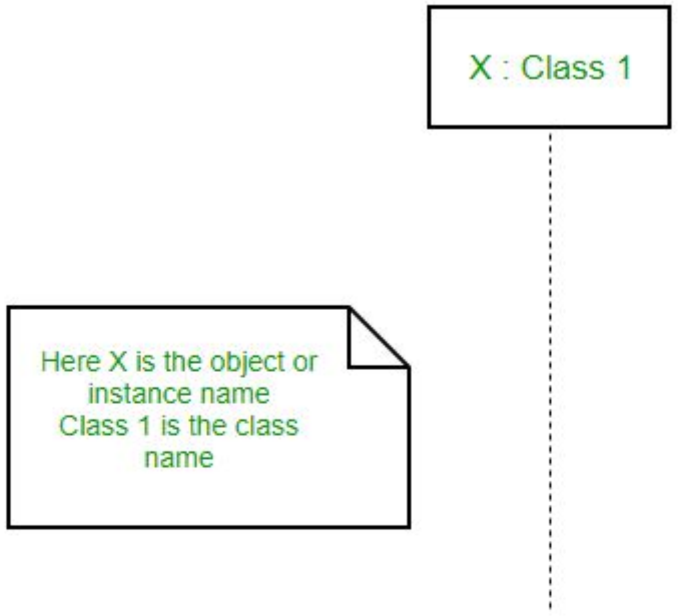
Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system. It is represented using a stick person notation. For example – Here the user in seat reservation system is shown as an actor where it exists outside the system and is not a part of the system

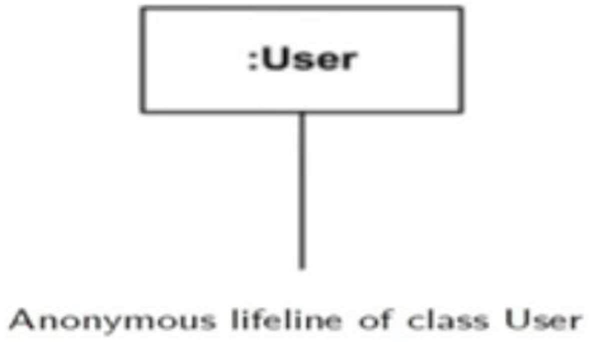


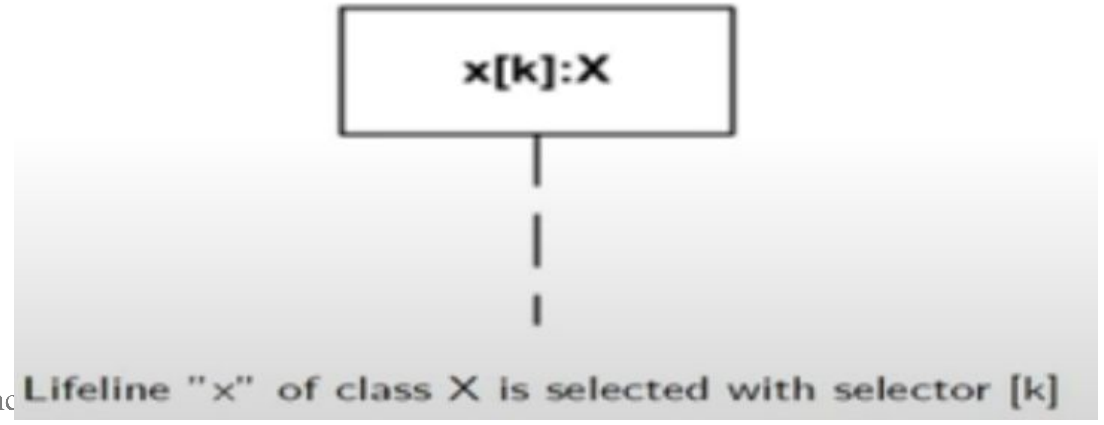


Lifelines – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name : Class Name




X : Class 1
Here X is the object or instance name
Class 1 is the class name





Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequenc

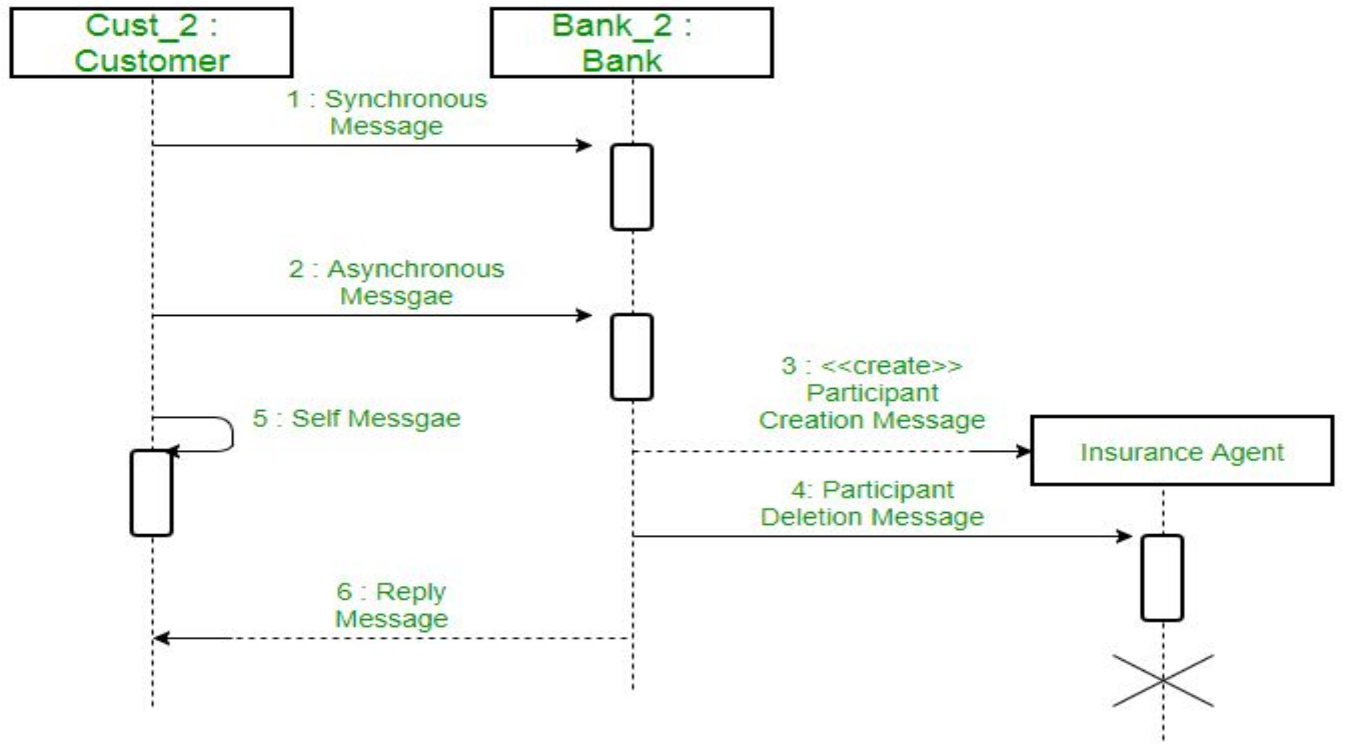
Messages – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

There are two major types of messages in Sequence Diagram:

- Message by Action Type - a message reflect either an operation call or start of an execution or sending/receiving of a single
    - Synchronous call
    - Asynchronous call/ signal
    - create
    - delete
    - reply

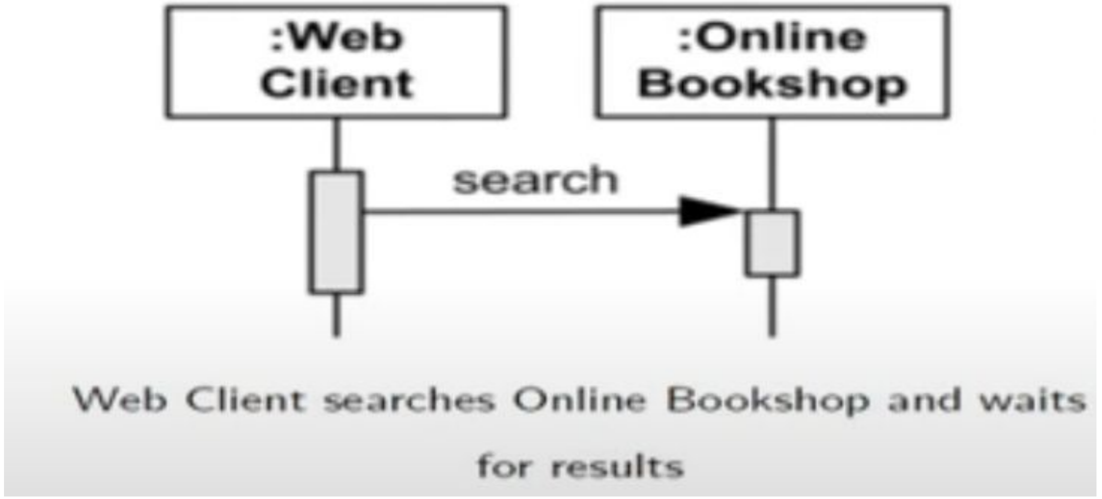
- Message by Presence of Events - a message depends upon whether messages send event and receive events are present

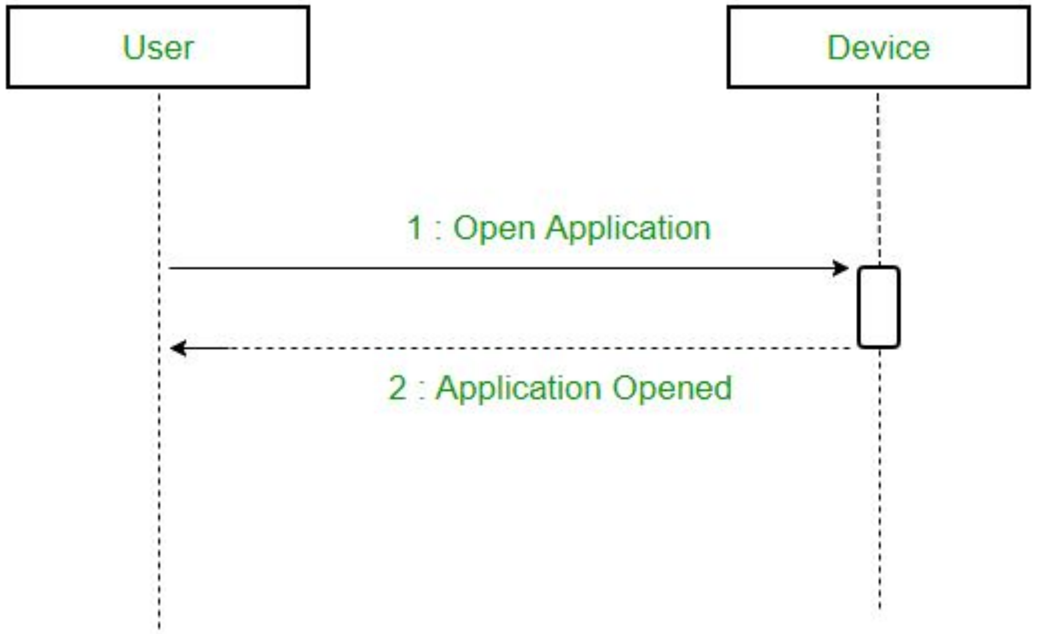
Message by Action type



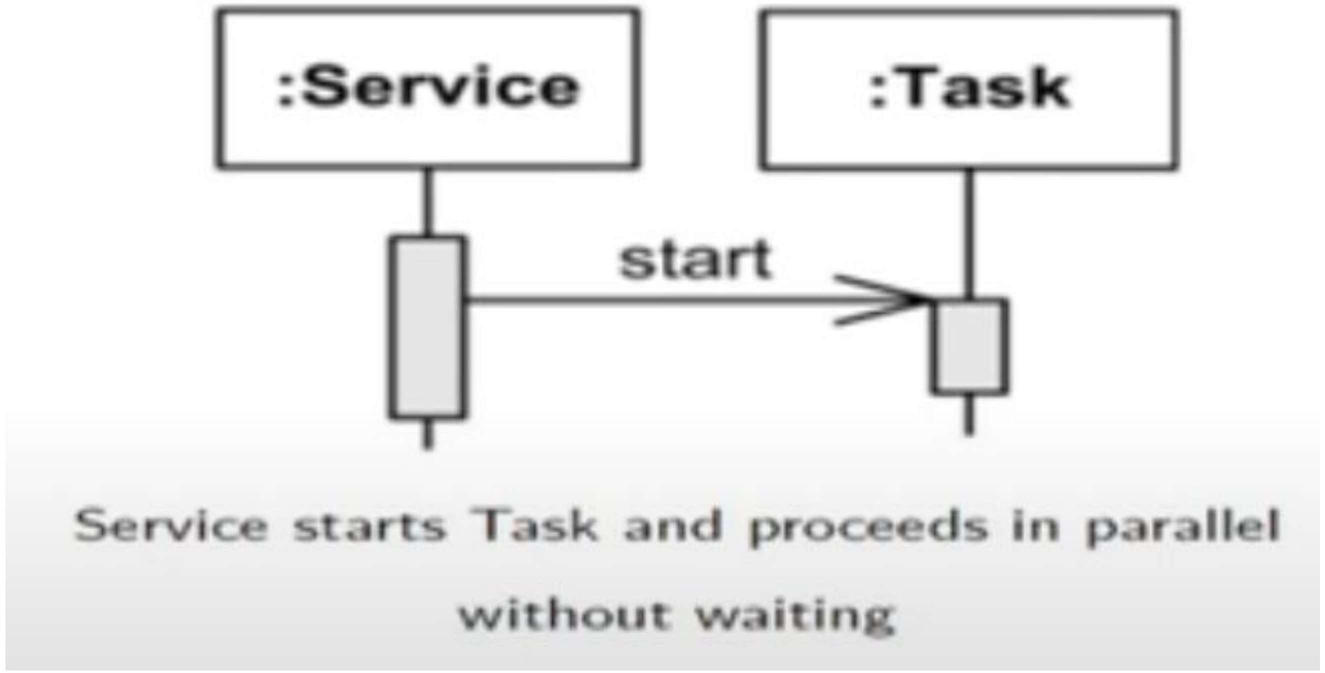
Message by Action type

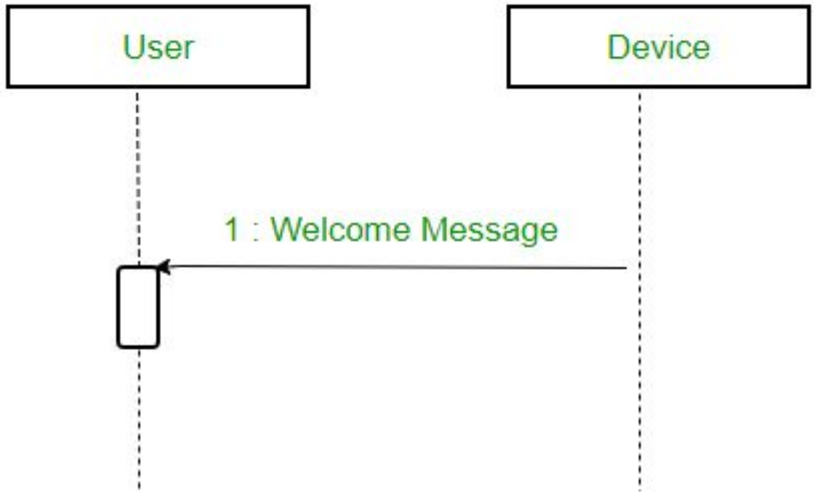
Synchronous messages – A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message. The caller continues only when it knows that the receiver has processed the previous message i.e. it receives a reply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





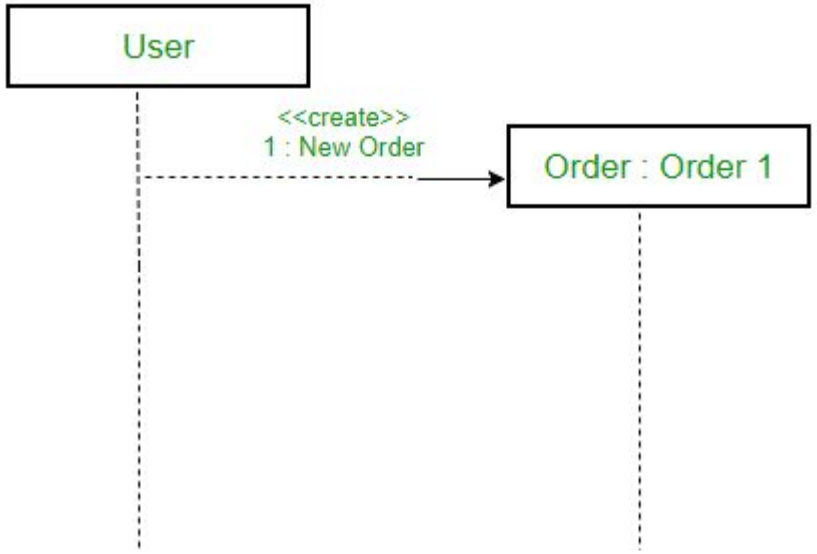
Asynchronous Messages – An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message.


:Service
:Task
start
Service starts Task and proceeds in parallel without waiting



Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

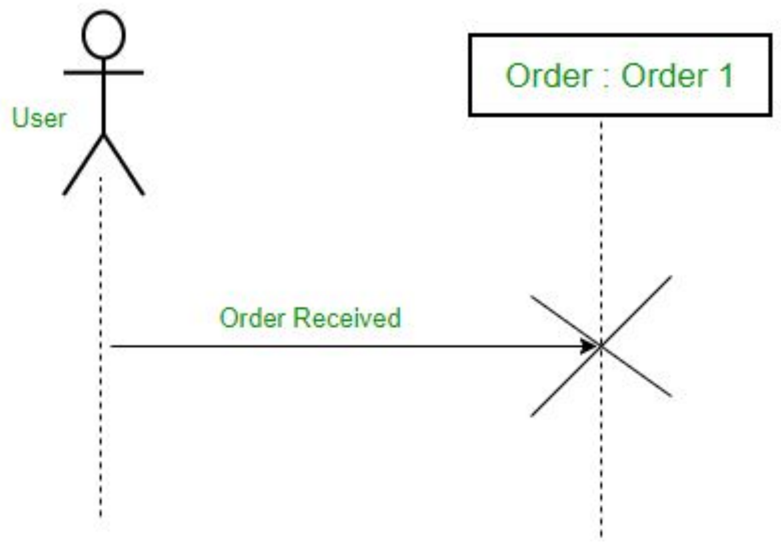
Create message –Create message is sent to a lifeline to instantiate itself as a new object in the sequence diagram. There are situations when a particular message call requires the creation of an object. It is represented with a dotted arrow and create word labelled on it to specify that it is the create Message symbol.
For example – The creation of a new order on a e-commerce website would require a new object of Order class to be created.



Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

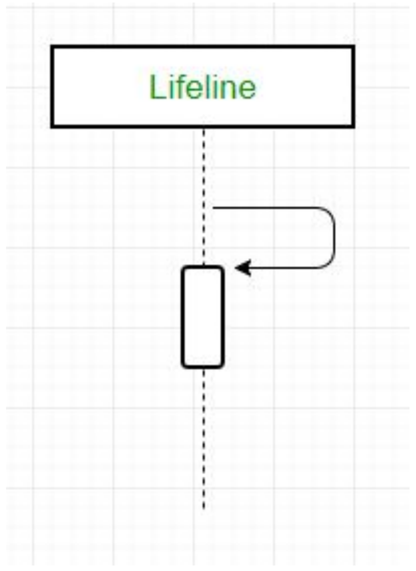
Delete Message –Delete Message is sent to delete another lifeline/object. When an object is deallocated memory or is destroyed within the system we use the Delete Message symbol. It destroys the occurrence of the object in the system. It is represented by an arrow terminating with a x.

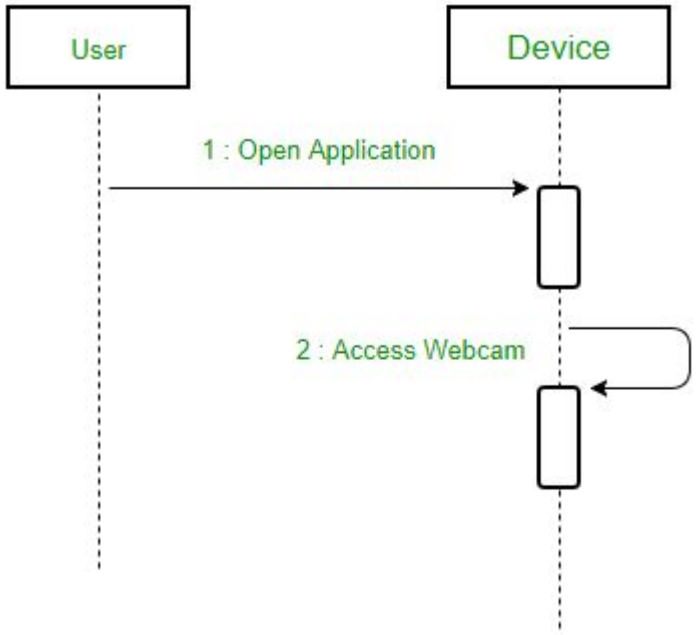
For example – In the scenario below when the order is received by the user, the object of order class can be destroyed.



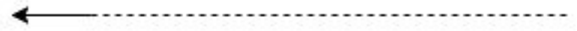
Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

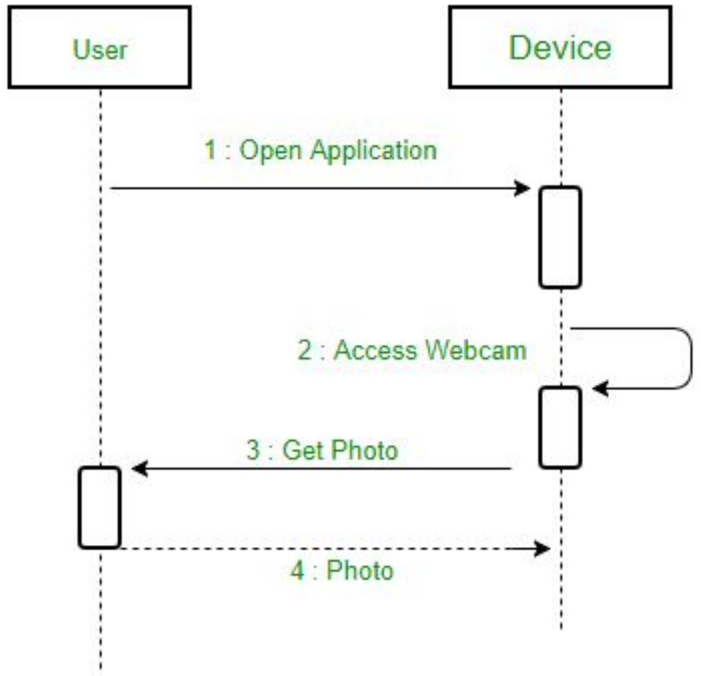
Self Message – Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages and are represented with a U shaped arrow.





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 arrowhead with a dotted line. The interaction moves forward only when a reply message is sent by the receiver.





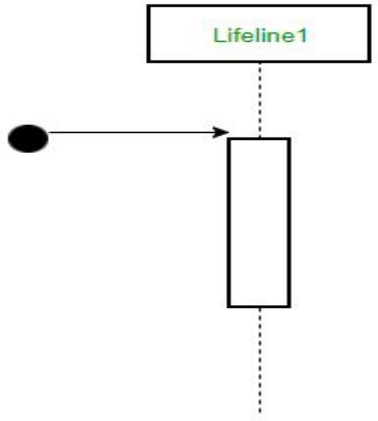
For example – Consider the scenario where the device requests a photo from the user. Here the message which shows the photo being sent is a reply message.

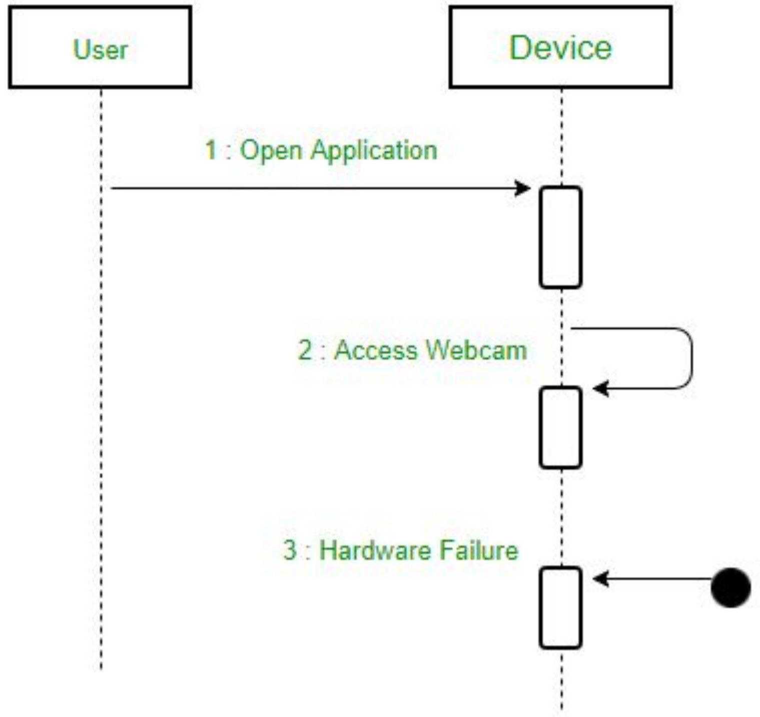
Message by presence of event

- Complete Message
  - Semantic: <SentEvent, ReceiveEvent>
  - both sent and received events are present
- Lost message (just send event)
- Found message (just receive event)
- Unknown message (default) - both absent (should not appear in SD)

Message by presence of event

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. For example: Consider the scenario of a hardware failure.

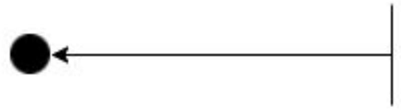


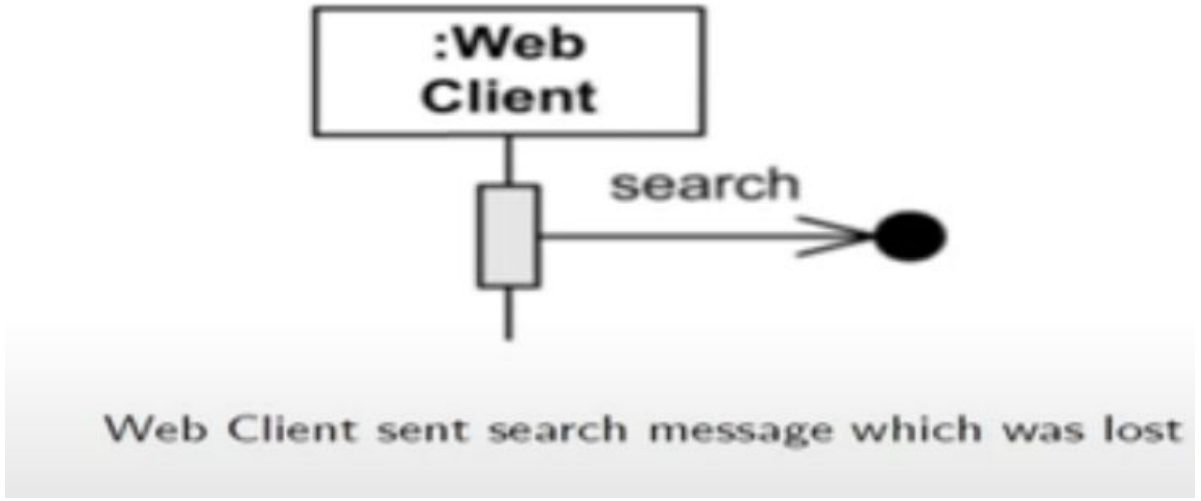


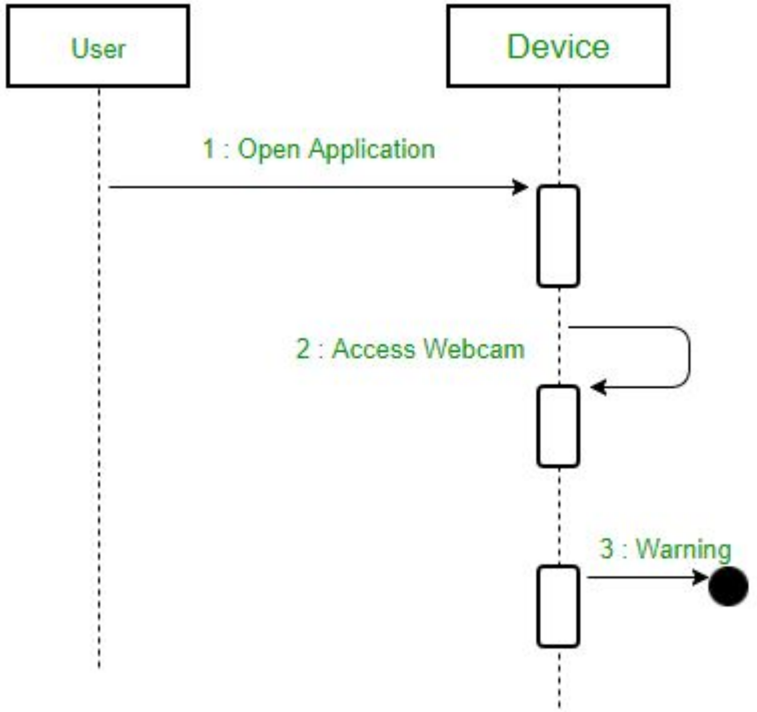
It can be due to multiple reasons and we are not certain as to what caused the hardware failure.

Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

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. For example: Consider a scenario where a warning is generated.

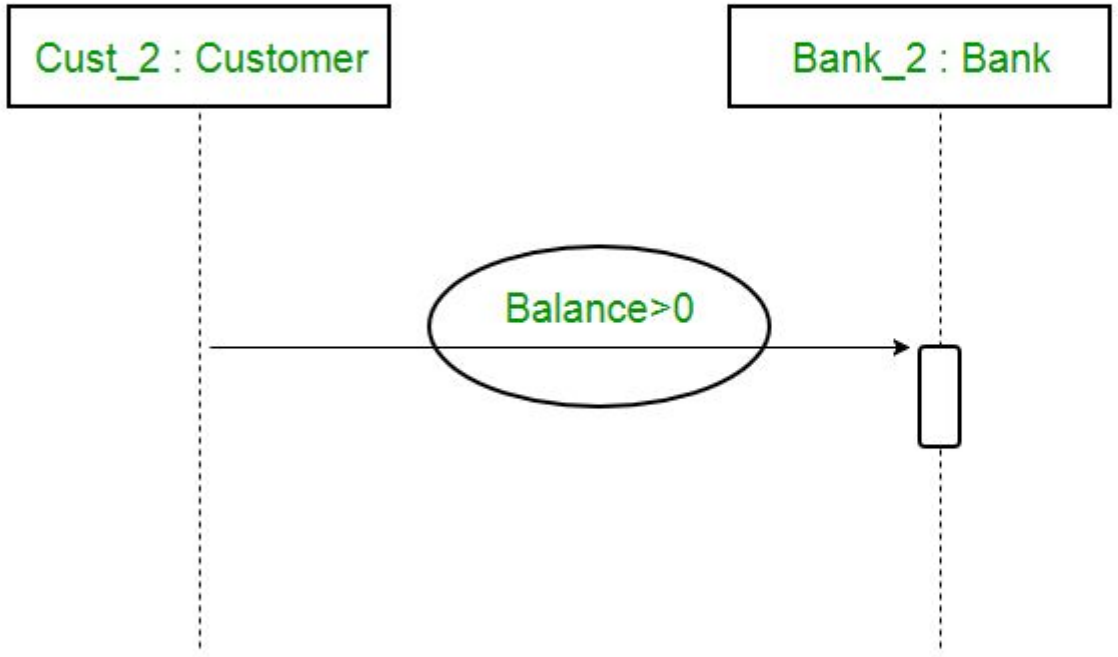






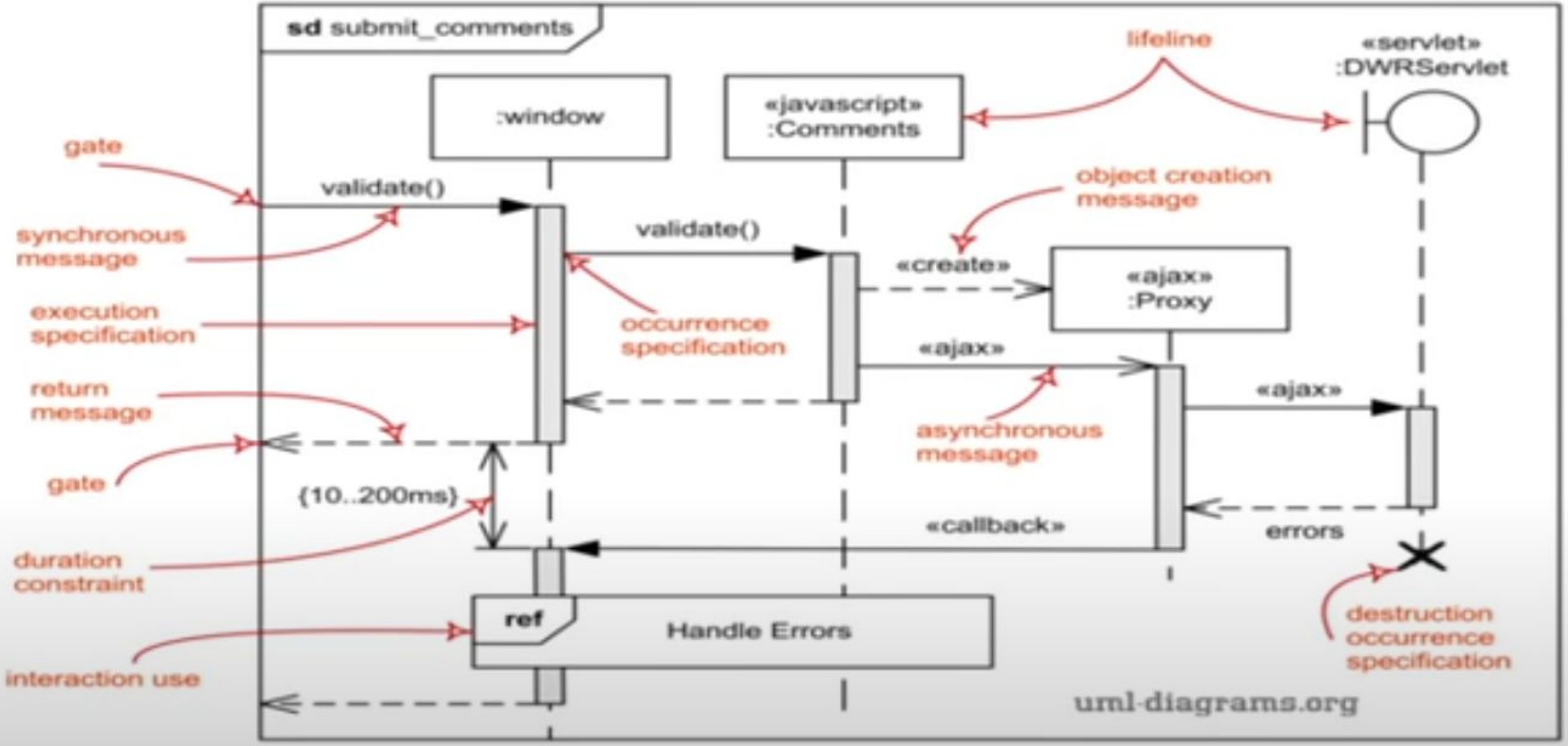
Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

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. For 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.

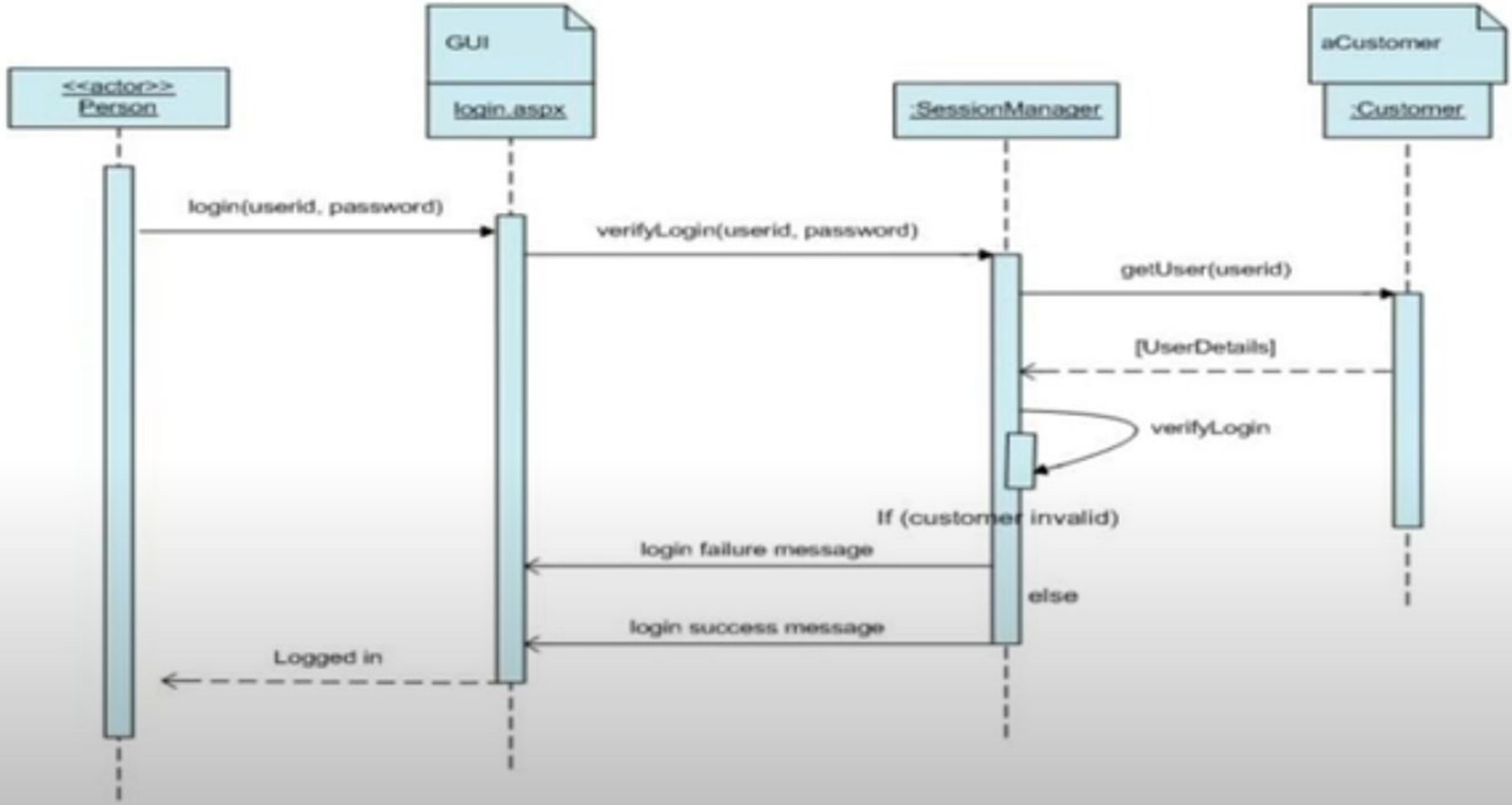


Reference: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

An annotated Sequence Diagram



Example: Login



Key references

Unified Modeling Language (UML): Complete Guide & Examples
-By James Rumbaugh, Ivar Jacobson, Grady Booch

https://www.guru99.com/uml-activity-diagram.html
https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

Thank You