CS5002NP

Software Engineering

Recap

- Unified Modeling Language

- Use case diagram, and use case description
    - Written from the user’s point of view
    - Avoid describing the internal aspects

Agendas

- Sequence Diagram
- Sequence Diagram and its Notations

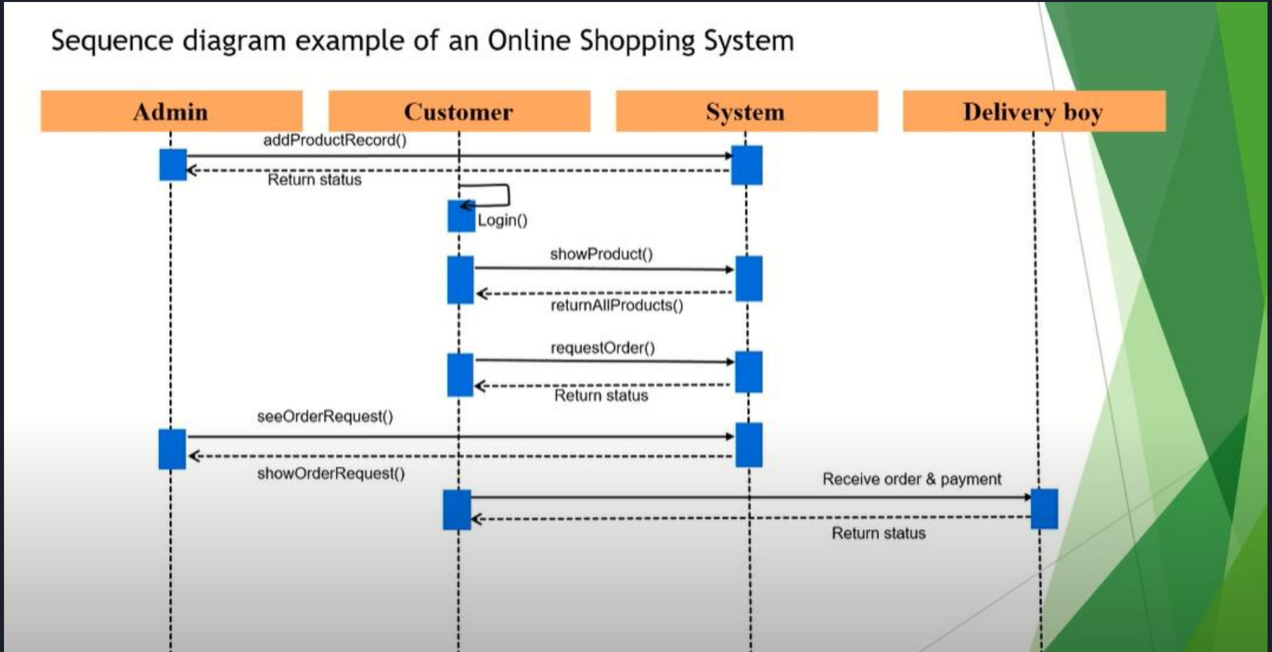
Sequence Diagram

- Describes **how** and in **what order** a group of objects work together
- **Event diagram** or **event scenario**
- **Visualise** and **validate** various runtime scenarios
- **Understand requirements** for a new system or to **document** an existing process

Sequence Diagram - Purpose

- Interaction between objects in sequential manner

Sequence Diagram example



Sequence Diagram - Purpose

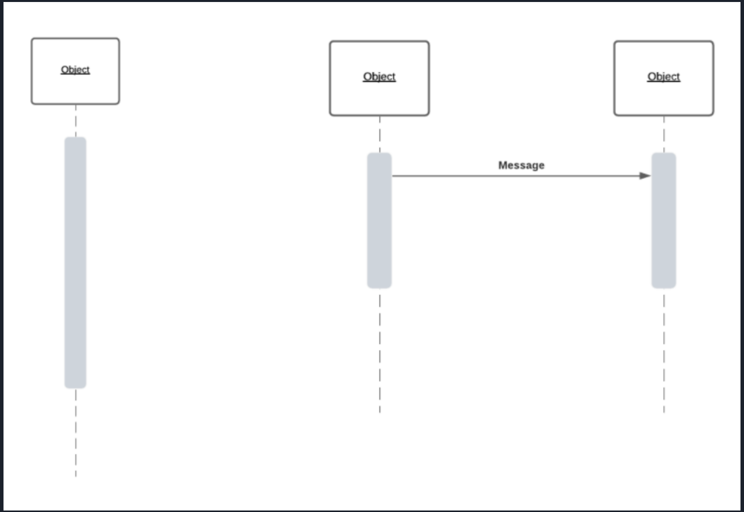
- Non-technical person to **explain the flow**
- Document **future system flow**
- **Forge out** the system's object interaction

Every USE CASE has an INDIVIDUAL SEQUENCE DIAGRAM

Sequence Diagram - Notations

- Activation Bar
    - Rectangle placed on the lifeline
    - Length represents the duration

Sequence Diagram - Notations



Sequence Diagram - Notations

- Messages
  - Arrow from the **caller** to the **receiver**
  - Flows in **any direction**
  - Comes with a description called message signature
    - **message_name(argument): return_type**

Sequence Diagram - Notations

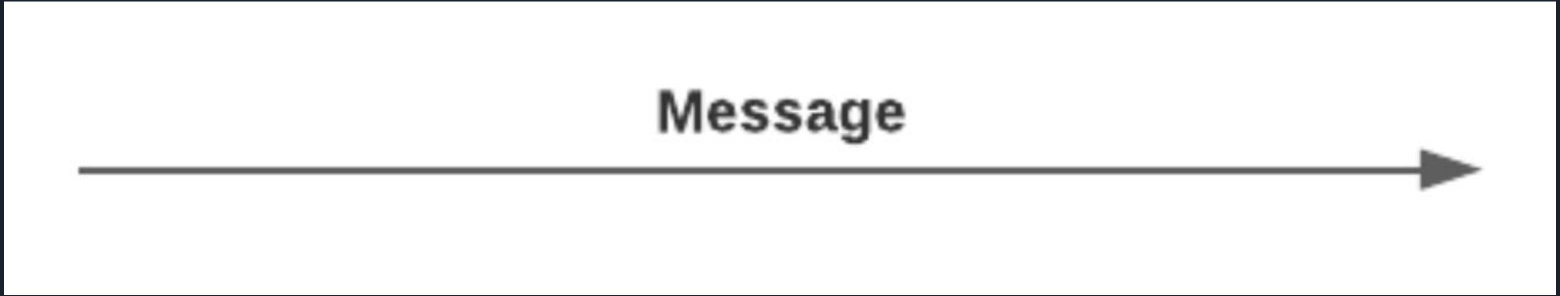
- Messages - Types
    - Synchronous Message
    - Asynchronous Message
    - Return Message
    - Creation Message
    - Destructive Message
    - Reflexive Message

Sequence Diagram - Notations

- Synchronous Message
    - Waits for the receiver to process the message
    - Represented as a straight line with a solid arrowhead
    - Regular message call

Sequence Diagram - Notations

- Synchronous Message

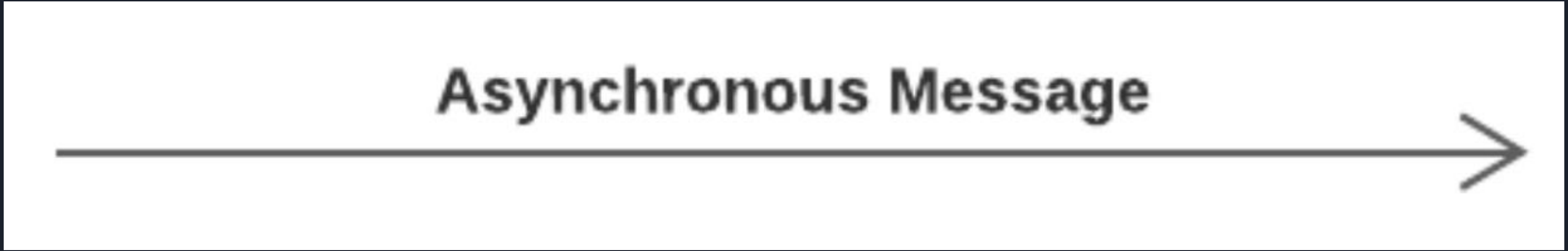


Sequence Diagram - Notations

- Asynchronous Message
  - Caller does not wait for the receiver to process the message
  - Represented as a straight line with a line arrow
  - Does not have a reply

Sequence Diagram - Notations

- Asynchronous Message

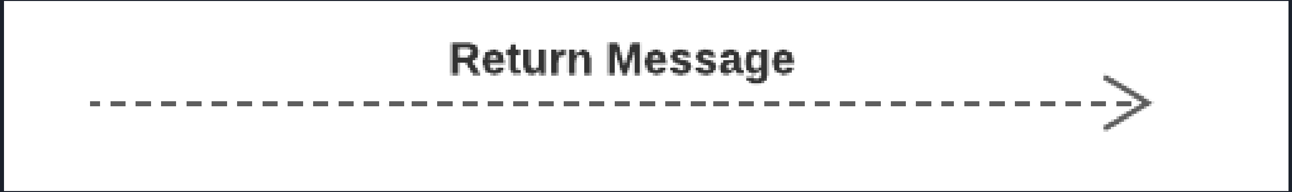


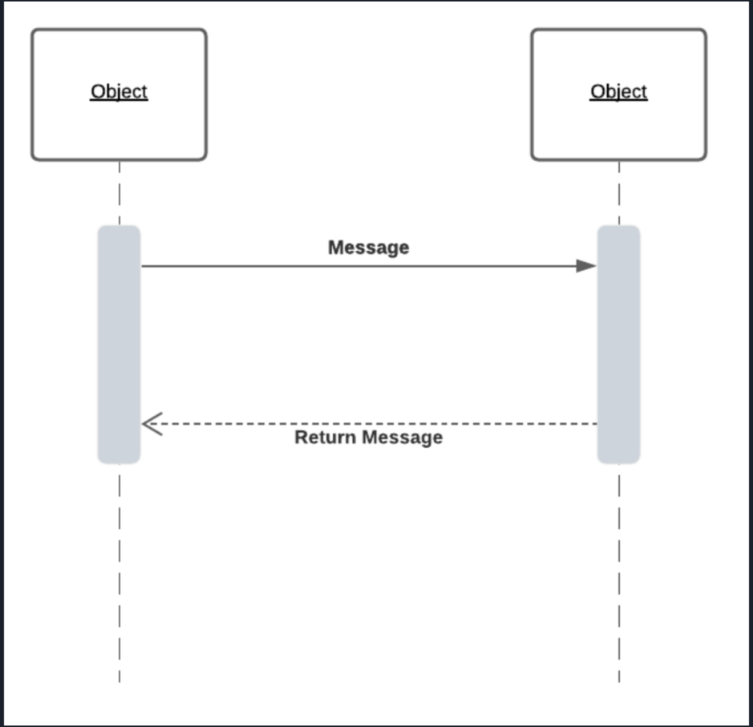
Sequence Diagram - Notations

- Return Message
    - Done processing the message
    - Represented as a dotted line with a line arrow
    - Are optional notations

Sequence Diagram - Notations

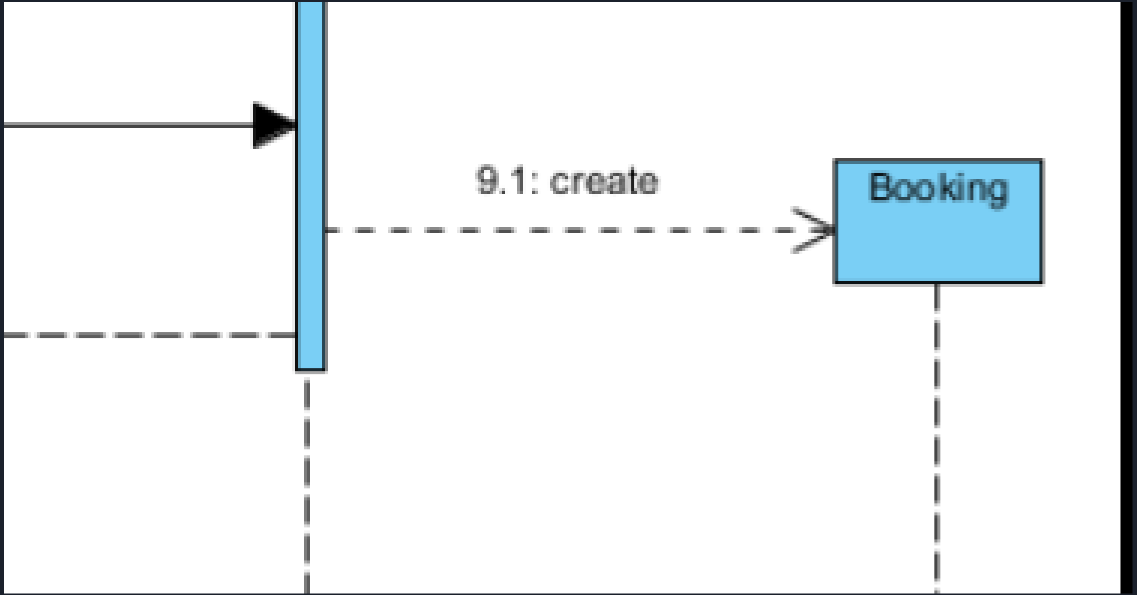
- Return Message




Object
Object
Message
Return Message

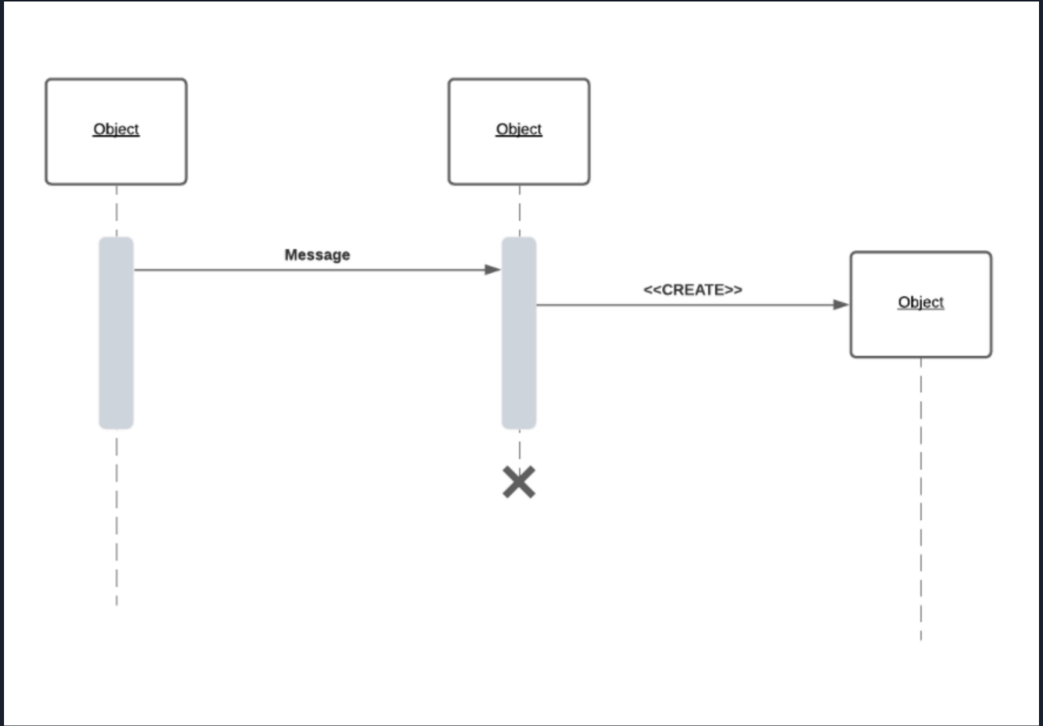
Sequence Diagram - Notations

- Creation Message
    - Objects or participants can be created according to the message that is being sent


9.1: create
Booking

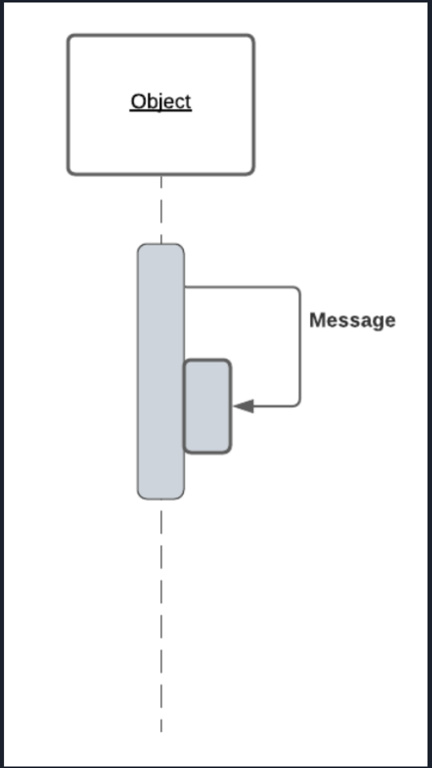
Sequence Diagram - Notations

- Destructive Message
    - Participant no longer needed
    - Adding a ‘X’ mark at the end of the lifeline


Object
Object
Message
<<CREATE>>
Object

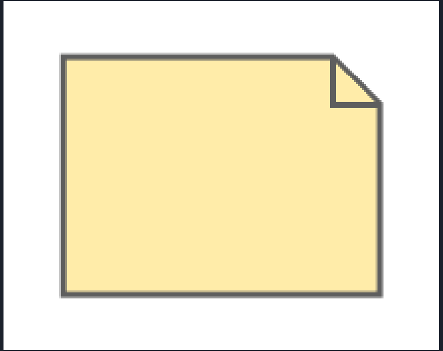
Sequence Diagram - Notations

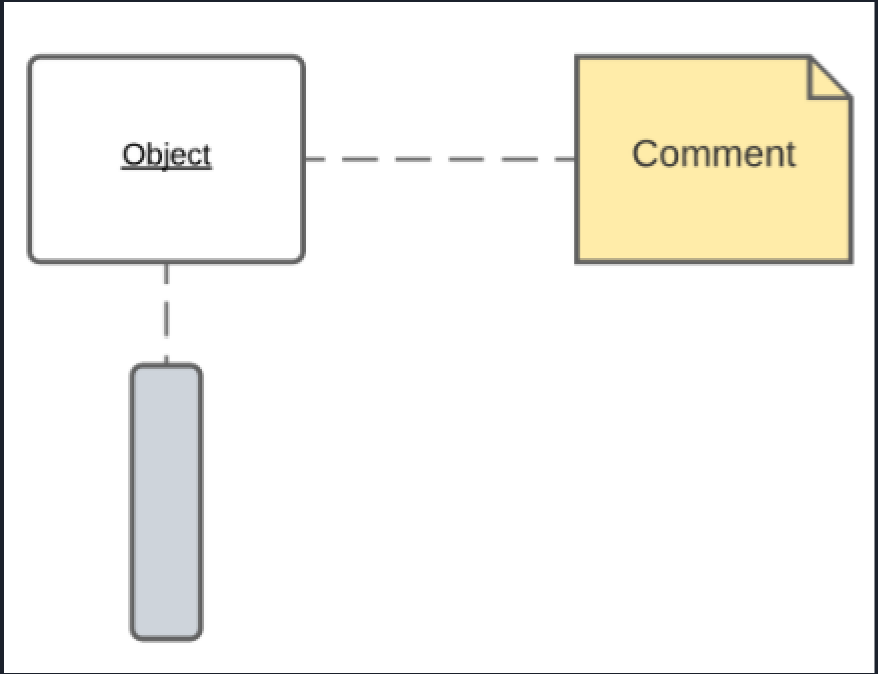
- Reflexive Message
    - Object sends a message to itself



Sequence Diagram - Notations

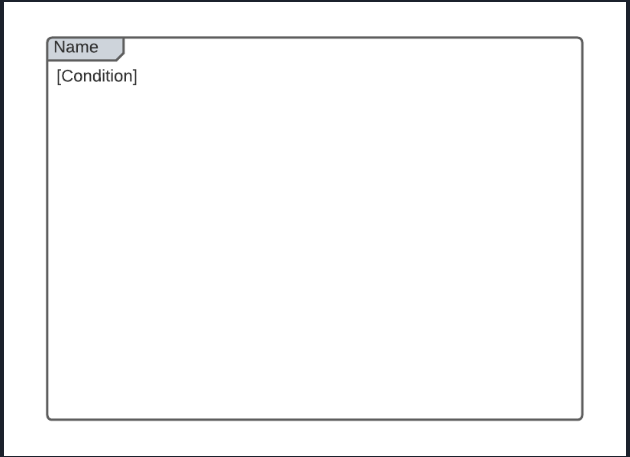
- Note
    - Represented as a rectangle with a folded-over corner
    - Not carry semantic force, contain information useful to modeler
    - Linked to object with a dashed line




Object
Comment

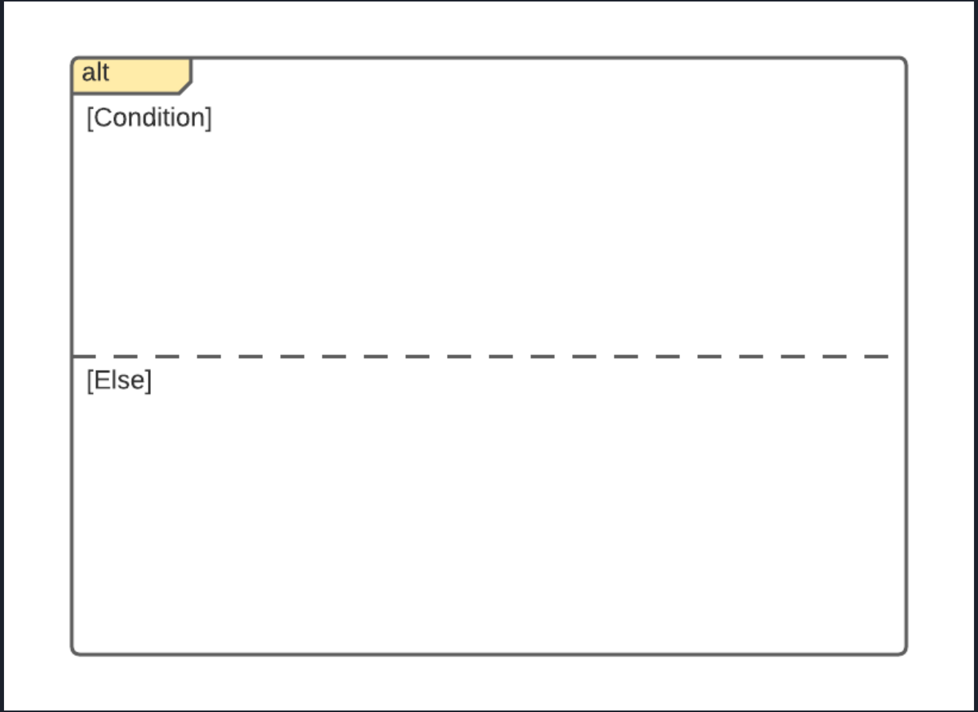
Sequence Diagram - Notations

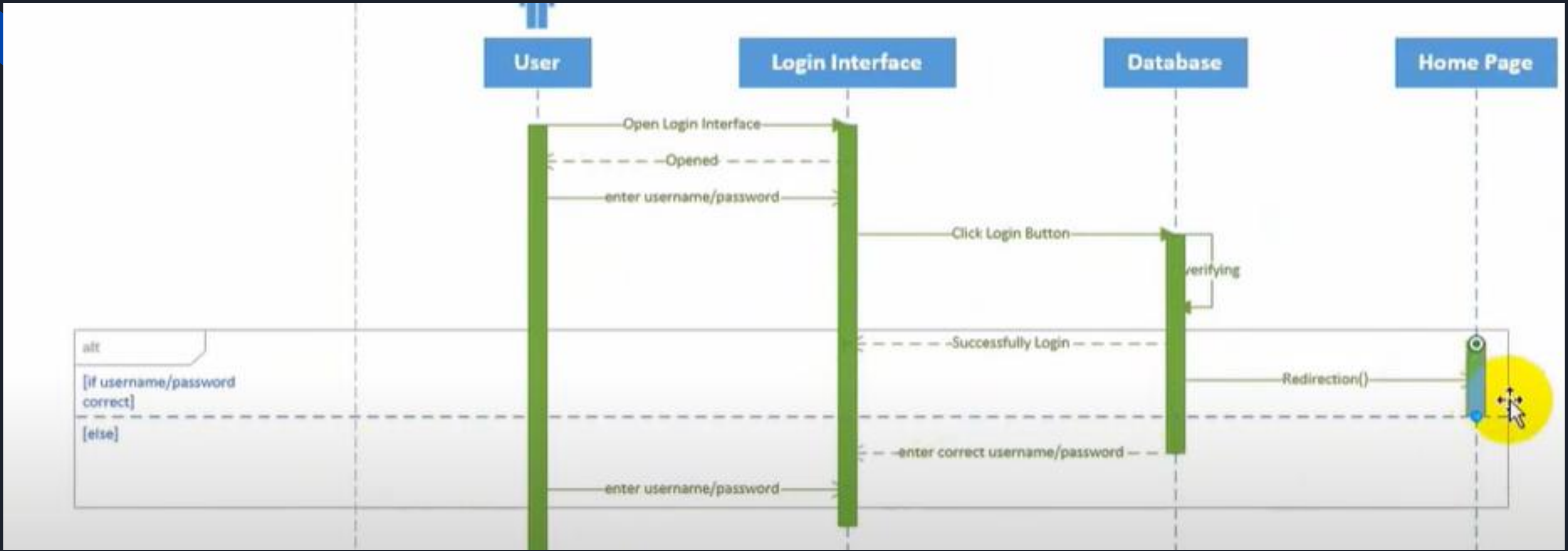
- Sequence Fragment
  - Represented as a box, with the fragment name of the top left
  - Types of fragments
    - Alternatives
    - Options
    - Loops



Sequence Diagram - Notations

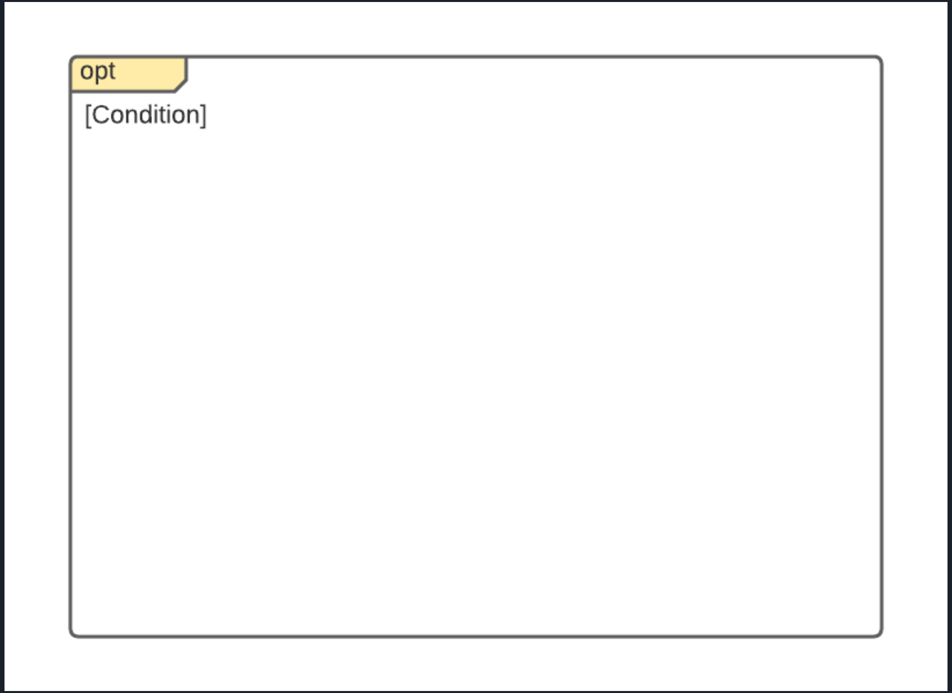
- Alternative Fragment
  - Choice needs to be made between two or more message sequences.
  - “If then else” logic
  - Represented by a rectangle and specified by ‘alt’ inside the fragment operator

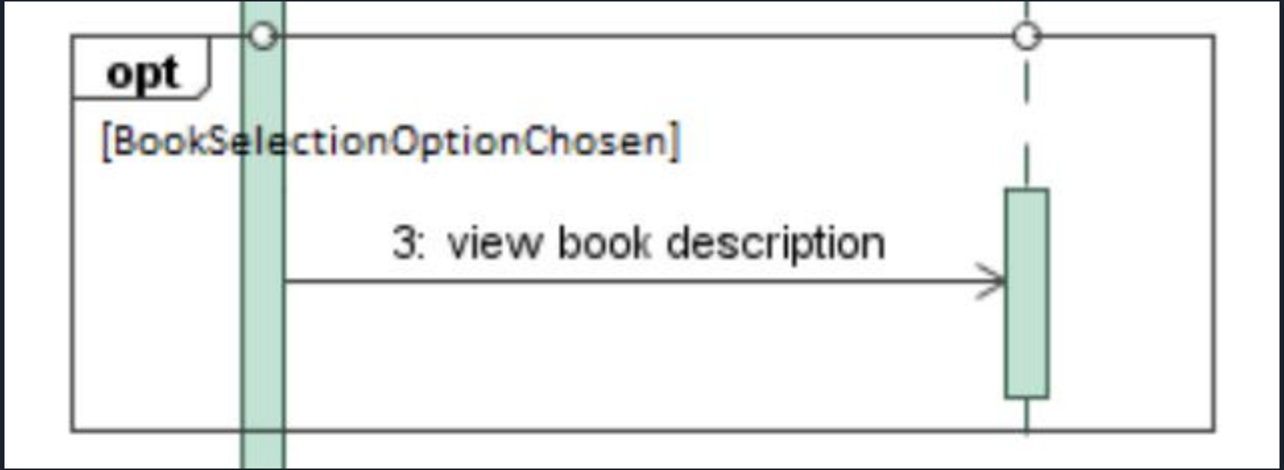

alt
[Condition]
[Else]


User
Login Interface
Database
Home Page
Open Login Interface
Opened
enter username/password
Click Login Button
verifying
alt
[if username/password correct]
Successfully Login
Redirection()
[else]
enter correct username/password
enter username/password

Sequence Diagram - Notations

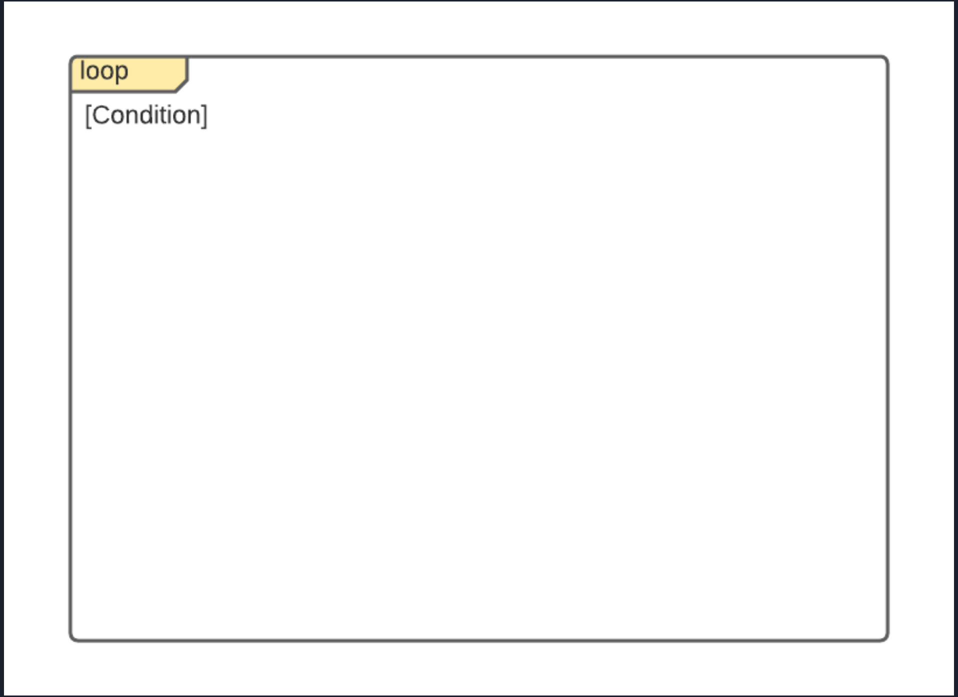
- Option Fragment
    - Indicate a sequence that will occur only under a certain condition
    - Rectangle frame where ‘opt is placed inside the fragment operator
    - Is not divided into two or more operands

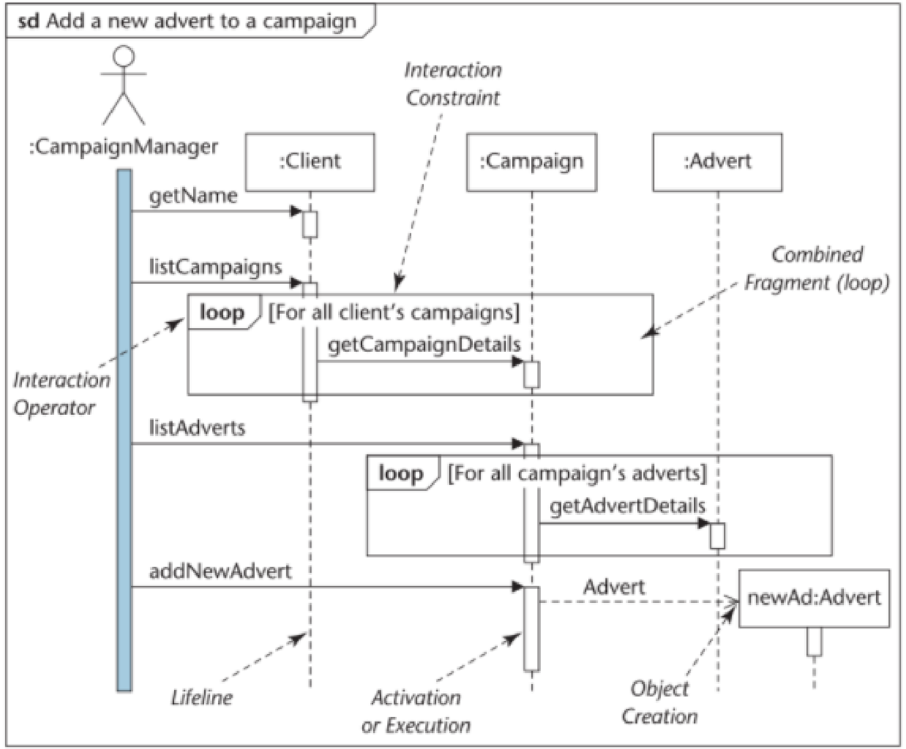

opt
[Condition]


opt
[BookSelectionOptionChosen]
3: view book description

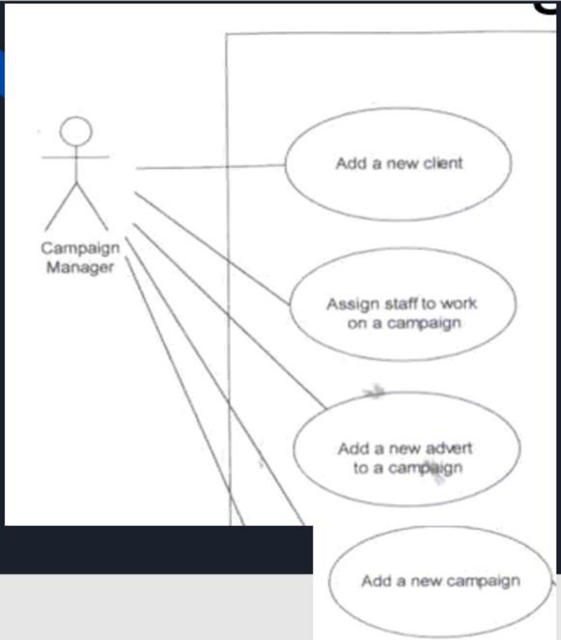
Sequence Diagram - Notations

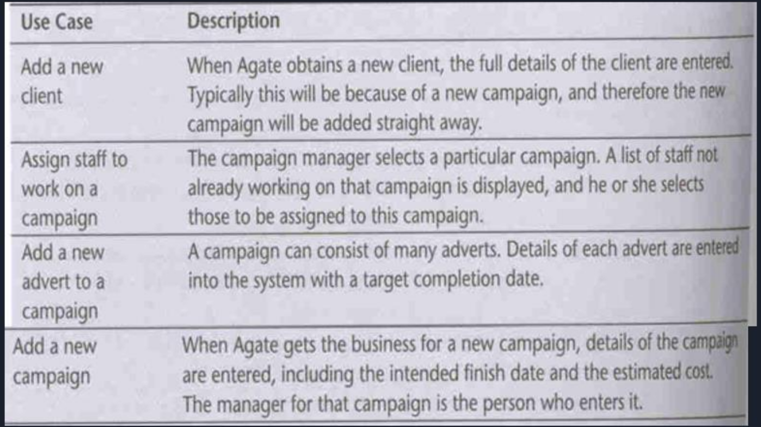
- Loop Fragment
  - Rectangle frame where ‘loop’ is placed inside the fragment operator
  - Tests boolean, minimum iterations and maximum iterations


loop
[Condition]

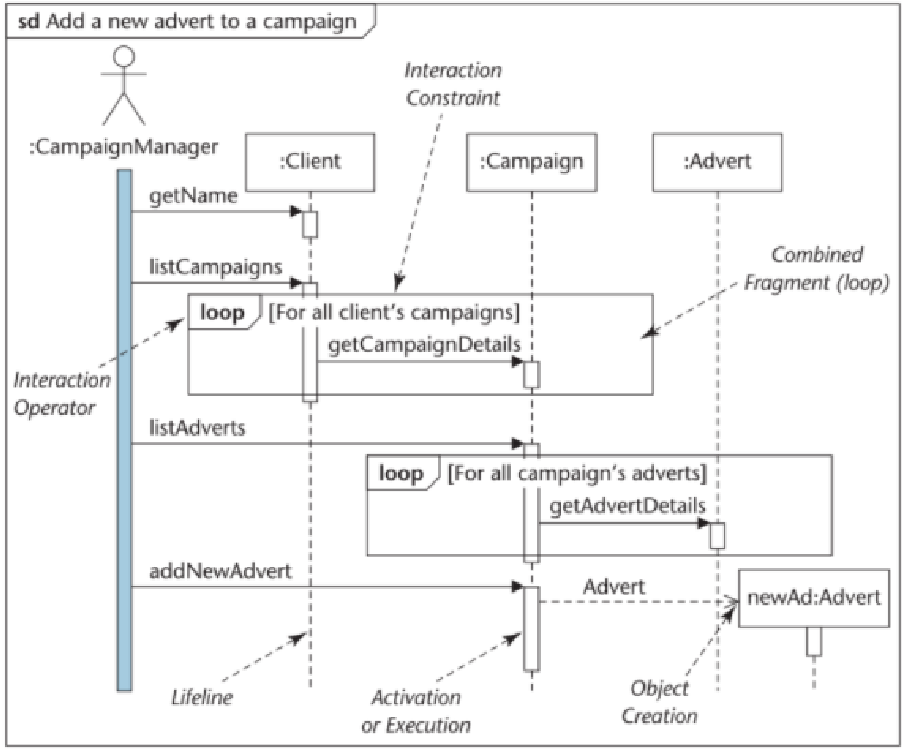

sd Add a new advert to a campaign
Interaction Constraint
:CampaignManager
:Client
:Campaign
:Advert
getName
listCampaigns
Combined Fragment (loop)
loop [For all client's campaigns]
getCampaignDetails
Interaction Operator
listAdverts
loop [For all campaign's adverts]
getAdvertDetails
addNewAdvert
Advert
newAd:Advert
Lifeline
Activation or Execution
Object Creation

Sequence Diagram Creation. Let's see an example of a small scenario !!!


Campaign Manager
Add a new client
Assign staff to work on a campaign
Add a new advert to a campaign
Add a new campaign



Use Case	Description
Add a new client	When Agate obtains a new client, the full details of the client are entered. Typically this will be because of a new campaign, and therefore the new campaign will be added straight away.
Assign staff to work on a campaign	The campaign manager selects a particular campaign. A list of staff not already working on that campaign is displayed, and he or she selects those to be assigned to this campaign.
Add a new advert to a campaign	A campaign can consist of many adverts. Details of each advert are entered into the system with a target completion date.
Add a new campaign	When Agate gets the business for a new campaign, details of the campaign are entered, including the intended finish date and the estimated cost. The manager for that campaign is the person who enters it.

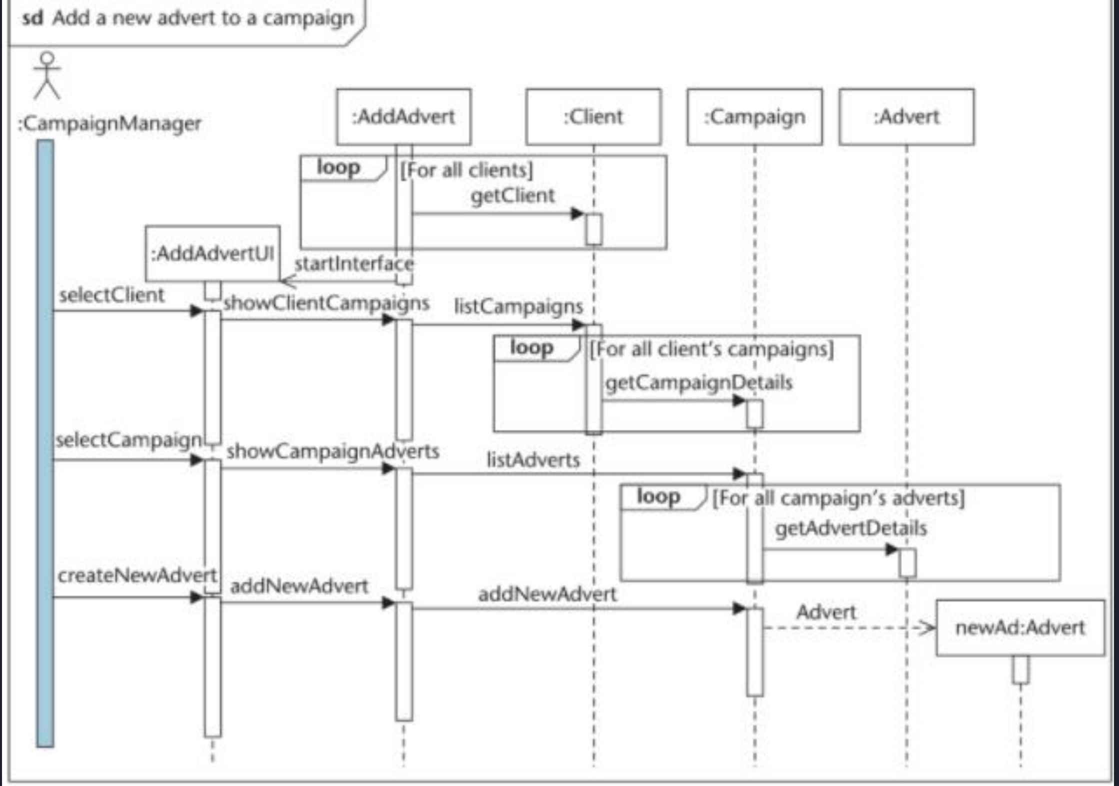

sd Add a new advert to a campaign
Interaction Constraint
:CampaignManager
:Client
:Campaign
:Advert
getName
listCampaigns
Combined Fragment (loop)
loop [For all client's campaigns]
getCampaignDetails
Interaction Operator
listAdverts
loop [For all campaign's adverts]
getAdvertDetails
addNewAdvert
Advert
newAd:Advert
Lifeline
Activation or Execution
Object Creation

Boundary Object and Control Object

- Boundary Object manages the dialogue between actor and the system.
- Control Object manages the object interaction.

Note:

- In most of the cases , control object is named same as the use case and boundary object is named as the use case name with addition of UI at last.


sd Add a new advert to a campaign
:CampaignManager
:AddAdvert
:Client
:Campaign
:Advert
loop [For all clients]
getClient
:AddAdvertUI
startInterface
selectClient
showClientCampaigns
listCampaigns
loop [For all client's campaigns]
getCampaignDetails
selectCampaign
showCampaignAdverts
listAdverts
loop [For all campaign's adverts]
getAdvertDetails
createNewAdvert
addNewAdvert
addNewAdvert
Advert
newAd:Advert

THANK YOU!