NANYANG POLYTECHNIC

IT2755
Software Engineering

Sequence Diagrams

Instructional Learning Outcomes

After this session you will be able to :

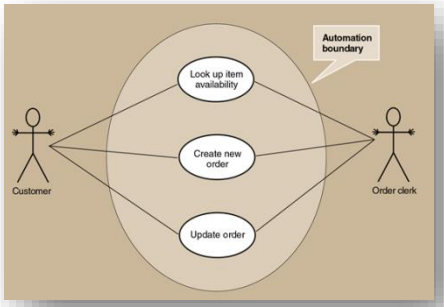
1. Describe the differences between Analysis and Design
2. Explain the purpose of a sequence diagram
3. Explain the role of each tier in the 3-tier software architecture
4. Explain the advantages and disadvantages of the 3-tier architecture
5. Assign responsibilities to the appropriate analysis objects in a sequence diagram

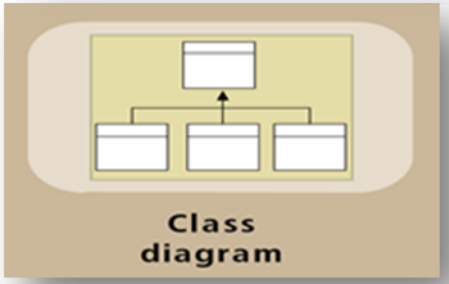
Object Oriented Analysis (OOA)

- In O-O Analysis we try to understand **WHAT** the problem is.
  - What happens in the current system?
  - What is required in the new system?
  - Seek to understand the organization and investigate its requirements
  - Models produced in analysis show WHAT is in the system and those parts are related to one another

Models developed during the OOA

- Use case diagrams and descriptions ✔
- Problem domain class diagram ✔
- State chart
- Activity diagrams





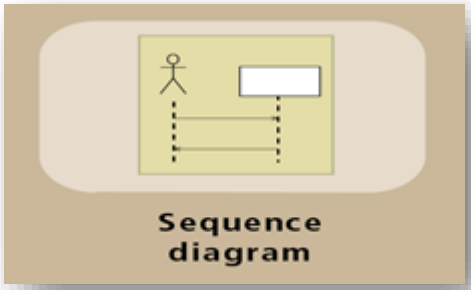
promote understanding

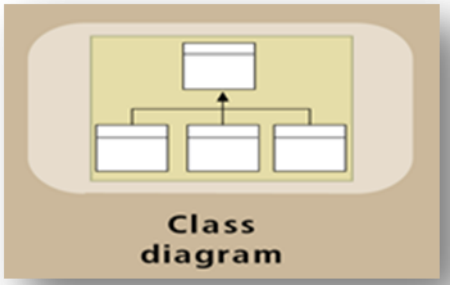
Object Oriented Design (OOD)

- In O-O Design, we focus on **HOW** to solve the problem. (to build the solution system)
  - Produce a best solution that
    - meets the requirements specified
    - Satisfies specified constraints
  - Prepare all the software classes needed for the final implementation (i.e. coding)
  - Models produced show how the various parts of the system will work together - “blueprints”

Models developed during the OOD

- Sequence diagram ✔
- Design class diagram ✔
- Package diagram ✔
- Network diagram
- Database schema





move project closer to implementation

System Design

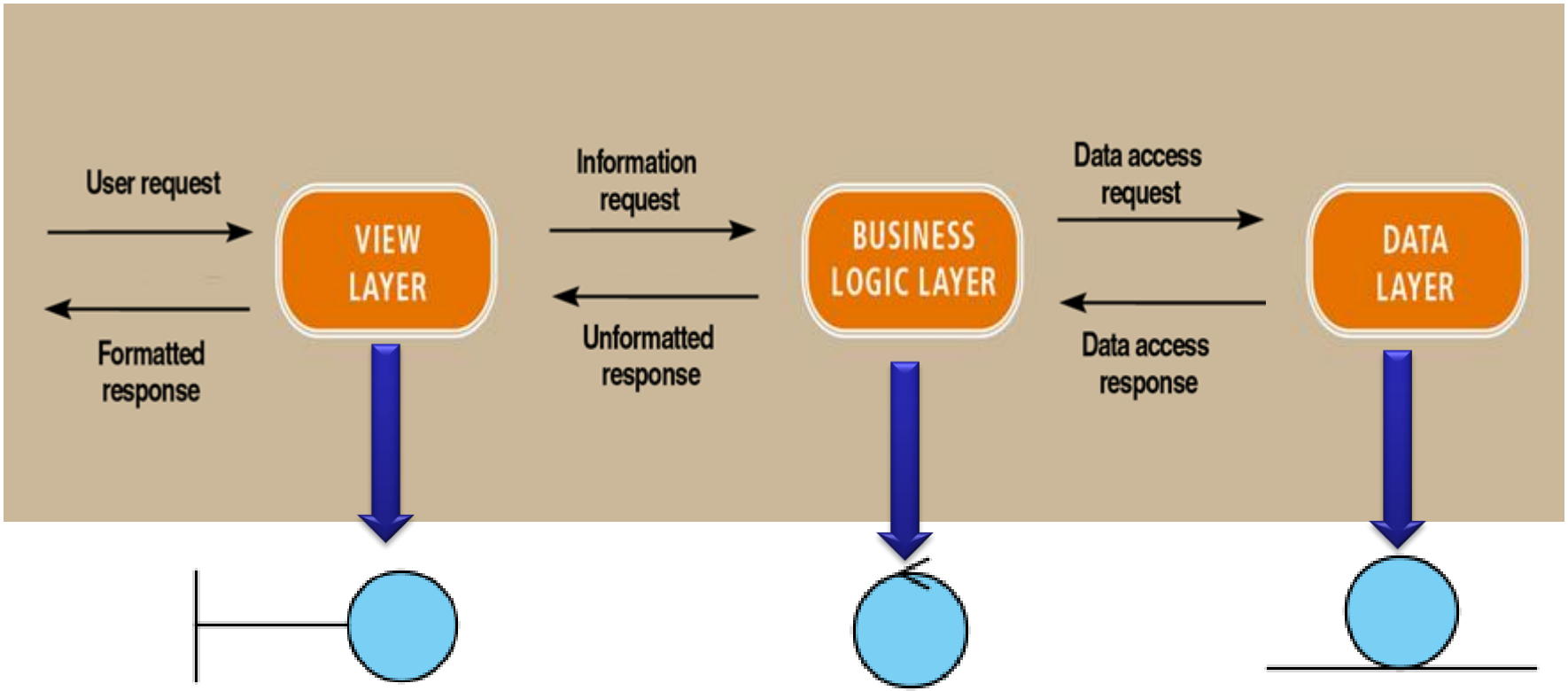
- 2 Levels:
  - High
    - System architecture
      - Hardware, network, and system software infrastructure
  - Low ✔
    - Software architecture
    - Software design for a use case
  - Higher-level activities contains and interacts with many lower-level activities

Software Architecture Design

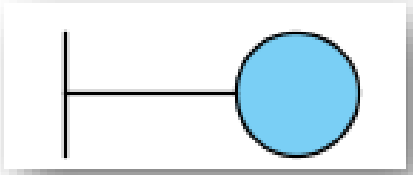
- Software Architecture
  - refers to the “big picture” structural aspects of an information system.
  - Defines a framework within which use cases are realized
- Two important aspects:
  - Division of software into classes
  - Distribution of classes across processing locations and specific computers

3-Tiers/Layers Software Architecture

Divides application software into 3 “independent” layers



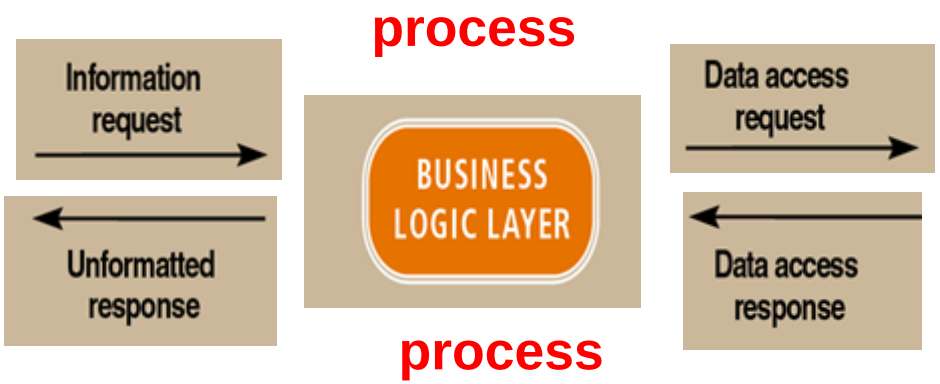
View layer (boundary object) responsibilities

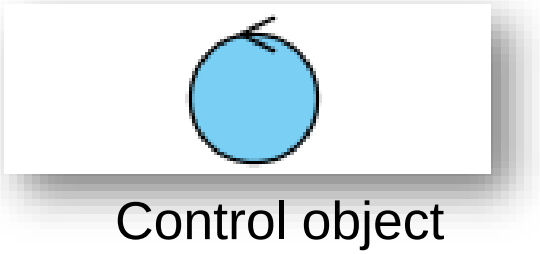


Boundary object

- Accepts and validates submission from user
- Relay service requests to business logic layer
- Receives unformatted response from business logic layer
- Format and displays responses to the user

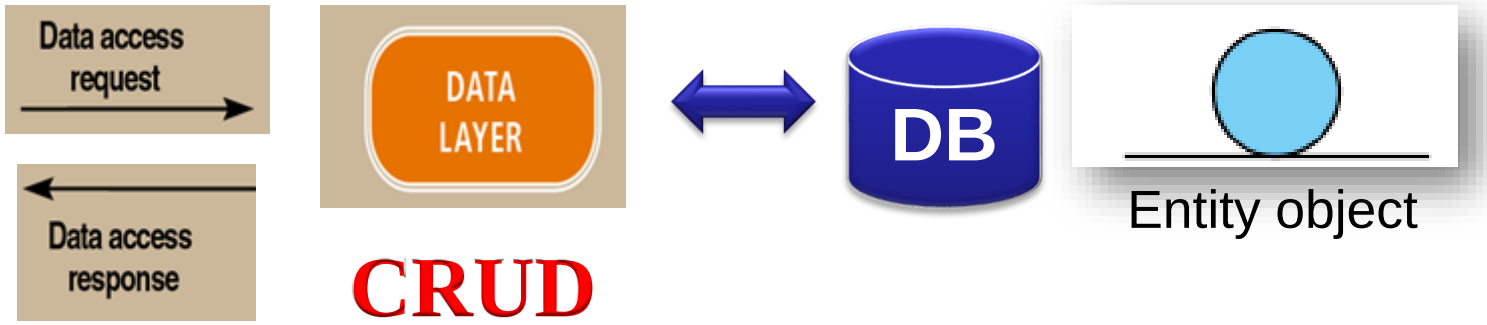
Business Logic layer (control object) responsibilities





- Receives service request from the view layer
- Process & Relay data access request to the data layer
- Receives & process data access response from the data layer
- Return unformatted response to view layer

Data layer (entity objects) responsibilities



- Receives data access requests (insert, retrieve, update, delete) from the business logic layer
- Carry out the data access requests (insert, retrieve, update, delete) in the database
- Return an appropriate data access response to the business logic layer

Advantages of the 3 Layer Architecture

- Separation of concerns.
  - Each tier has designated and specific purpose
- Promotes code reuse
  - All business logic can be defined once within the business layer and then shared by any number of components within the presentation layer
- Easy to implement changes
  - Changes in the contents of any one of the tiers (layers) can be made without having to make any or much corresponding changes in any of the other tiers

Advantages of the 3 Layer Architecture

- Enables parallel development of the different tiers of the application.
- Improved data integrity
  - The business logic layer can ensures that only valid data is allowed to be updated in the database
- Improved security
  - The presentation layer/client does not have direct access to the database
  - The database structure is hidden from the caller.

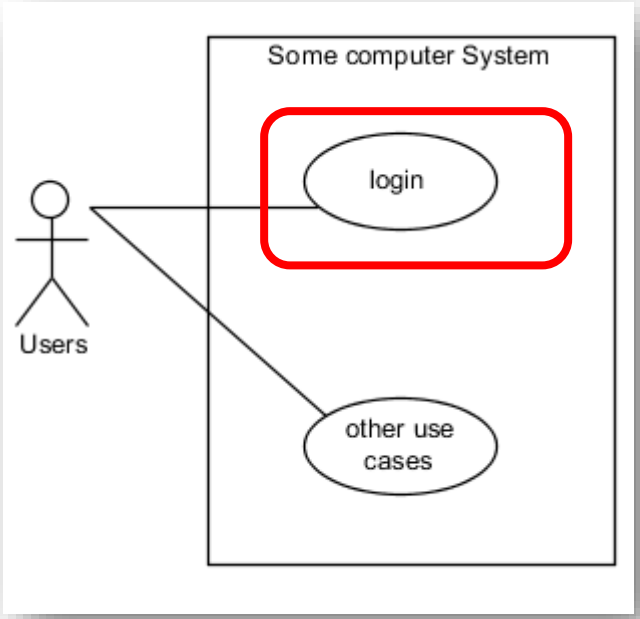
Disadvantages of the 3 Layer Architecture

- More complex structure
- Communication between the tiers may moderately affect performance

Use Case Realization

- The design of the software that implements each use case to make them executable
- Focus on the **dynamic interactions** required to perform specific processing tasks
- Ie: Need to identify the work to be carried out by each object & the coordination of work in order to accomplish the goal of a use case.
- We need a sequence diagram to specify the above details.

Design Use Case Realisation – Login use case example

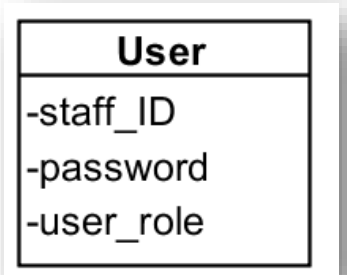


Use case

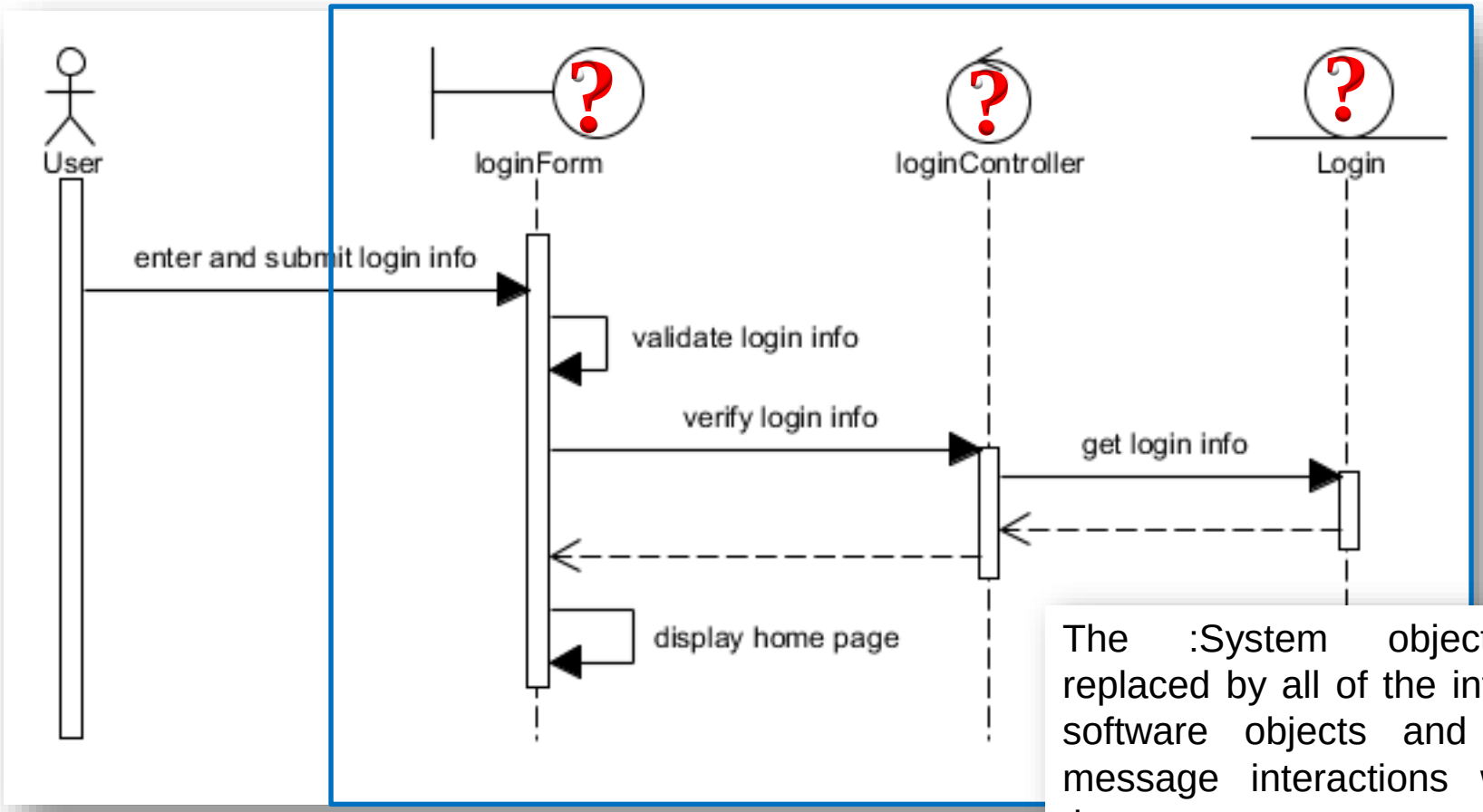
Use Case ID:	ACC_UC_1
Use Case Name:	login
Created By:	Joe Blogs
Date Created:	1-1-2012
Description:	This use case allows user to login into the system to access the relevant functions according to the user's role. The various user roles are staff, admin staff, system administrator, manager and department head. To login to the system, all users have to enter their unique staff id which is their NRIC number and a valid current password. The users have a maximum of 3 attempts to login after which their account are locked and they will have to contact the system administrator to unlock their account upon successful login the system will display the relevant user's home page.
Primary Actor:	User
Secondary Actor:	None
Preconditions:	1. User has to have a valid account
Postconditions:	1. The system displays the relevant homepage
Main Flow:	1. The user enters the staffID and password 2. The user submits the staffID and password 3. The system validates the staffID and password 4. The system verifies the staffID and password 5. The system displays the user's homepage 6. The use case ends

Use case description

Domain Class diagram



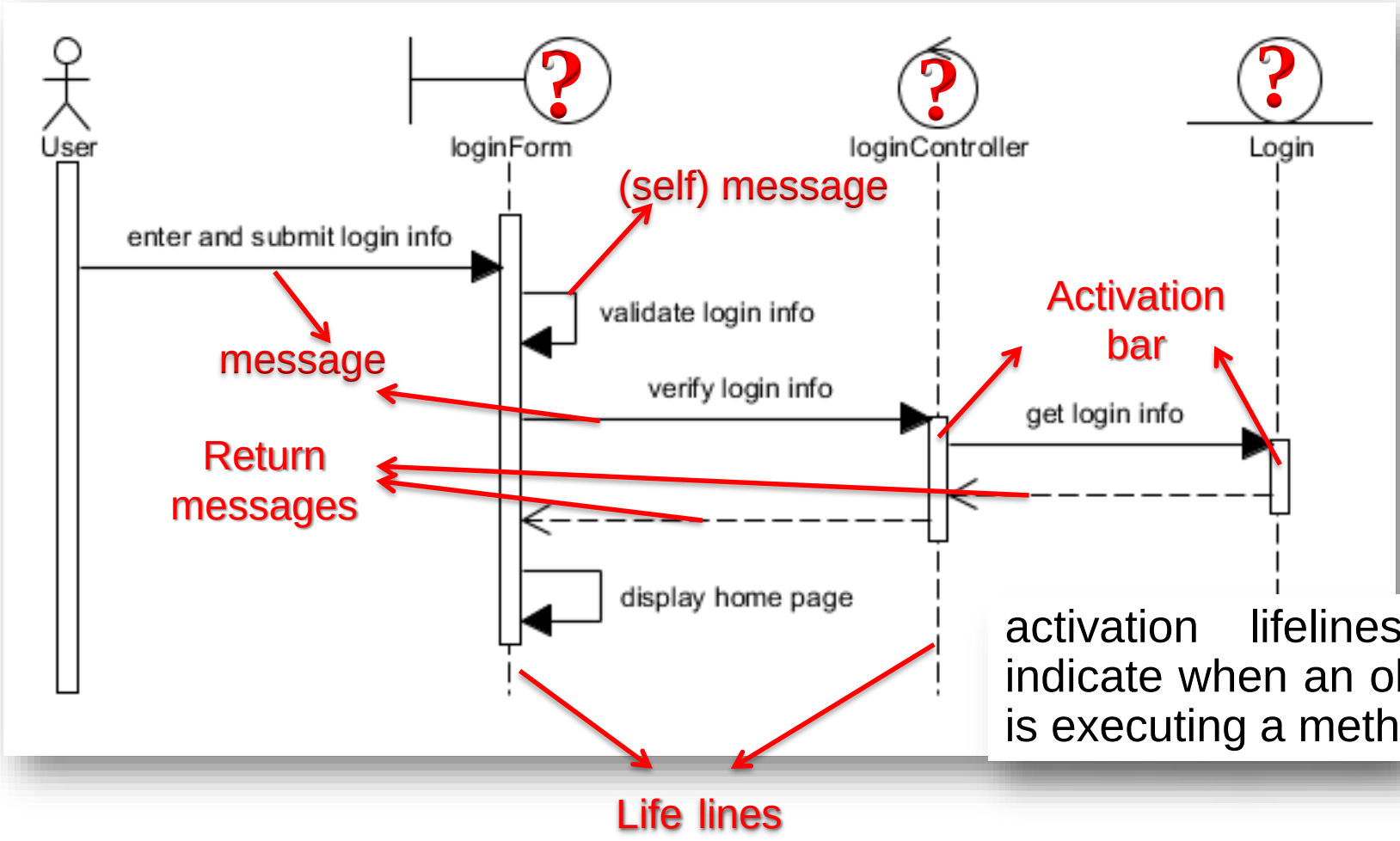
Detail Sequence Diagram



The :System object is replaced by all of the internal software objects and their message interactions within the system

Sequence Diagram

(Detail) Sequence Diagram



activation lifelines to indicate when an object is executing a method

(Detail) Sequence Diagram

3 types of analysis objects:

UML Notation	Analysis object Name	UML Class with stereotype << >>	3 – tier architecture	Purpose

Elements of a Sequence diagram

- Only one boundary object
  - Normally just append the word "form" to the use case name
- Only one control object
  - Normally just append the word "controller" to the use case name
- One or more entities objects
  - Identified from the domain class diagram

Interactions between analysis objects in a Sequence diagram

- Actor can only interacts (i.e. send messages to) with the boundary object
- Boundary object can only interacts (i.e. send messages to) with controller object
- Control object can only interacts (i.e. send messages to) with entity objects
  - Control object can only return to boundary object
- Entity objects can only interact (i.e. send messages to) with other entity objects
  - Entity objects can only return to other entity objects or control object

Case Study – rent video use case

Basic Flow :

- The staff enters and submit member ID
- The system validates member ID
- The system retrieves and displays member details
- The staff enters and submit movie ID
- The system verifies member's loan limit
- Repeat step 4 to 5 for all movie
- The system creates rental and rental item records
- The system displays a successful rental message.
- The staff acknowledges the message.

Alternate Flow

2a. Invalid member

i. The system prompt "Invalid Member" error message

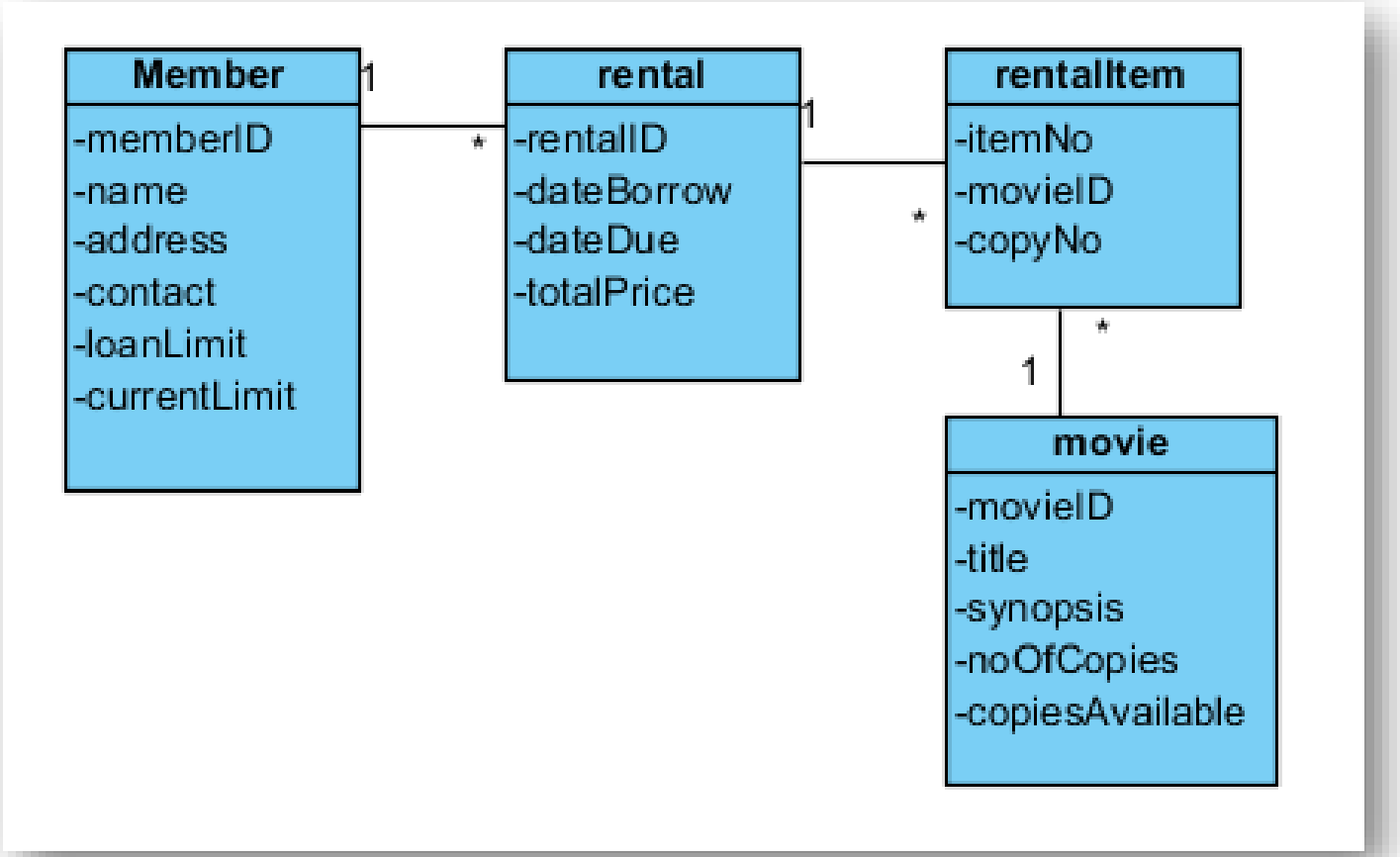
ii. Use case ends

5a. Loan Limit exceeded

i. The system will prompt "Loan Limit Exceeded" error message

ii. Use case ends.

Case Study – rent video use case



Domain Model

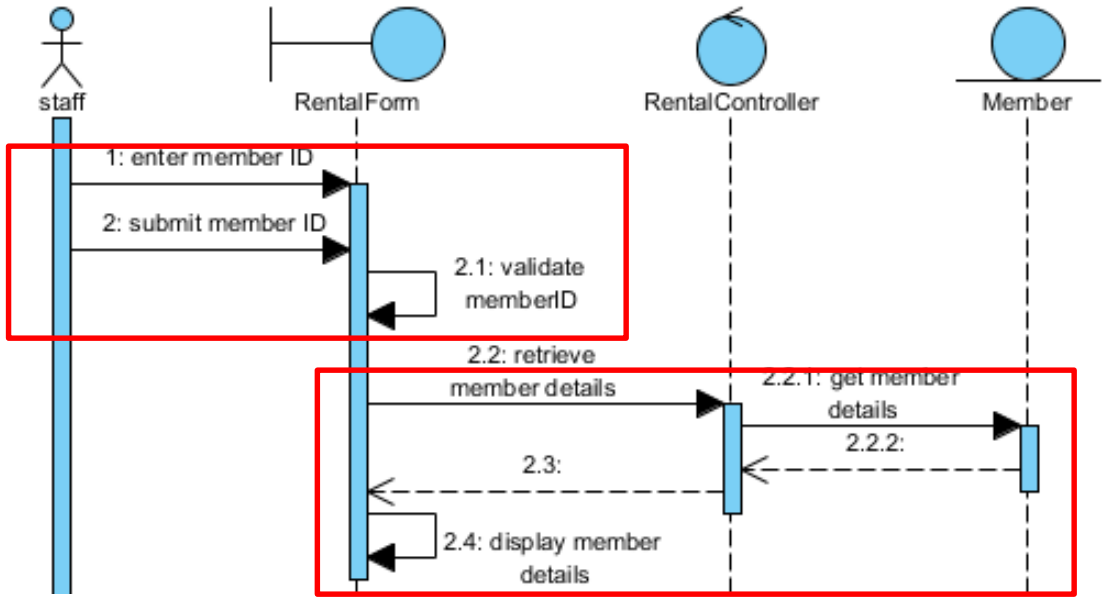
Case Study – rent video use case Analysis objects

- Rent video boundary object – RentalForm
- Rent video control object – RentalController
- Rental video entities:
  - Member – memberID, loanlimit and currentLimit
  - Movie - MovieID,
  - Rental – new rental details
  - RentalItem – new rental items details

Case Study – rent video use case

Basic Flow :

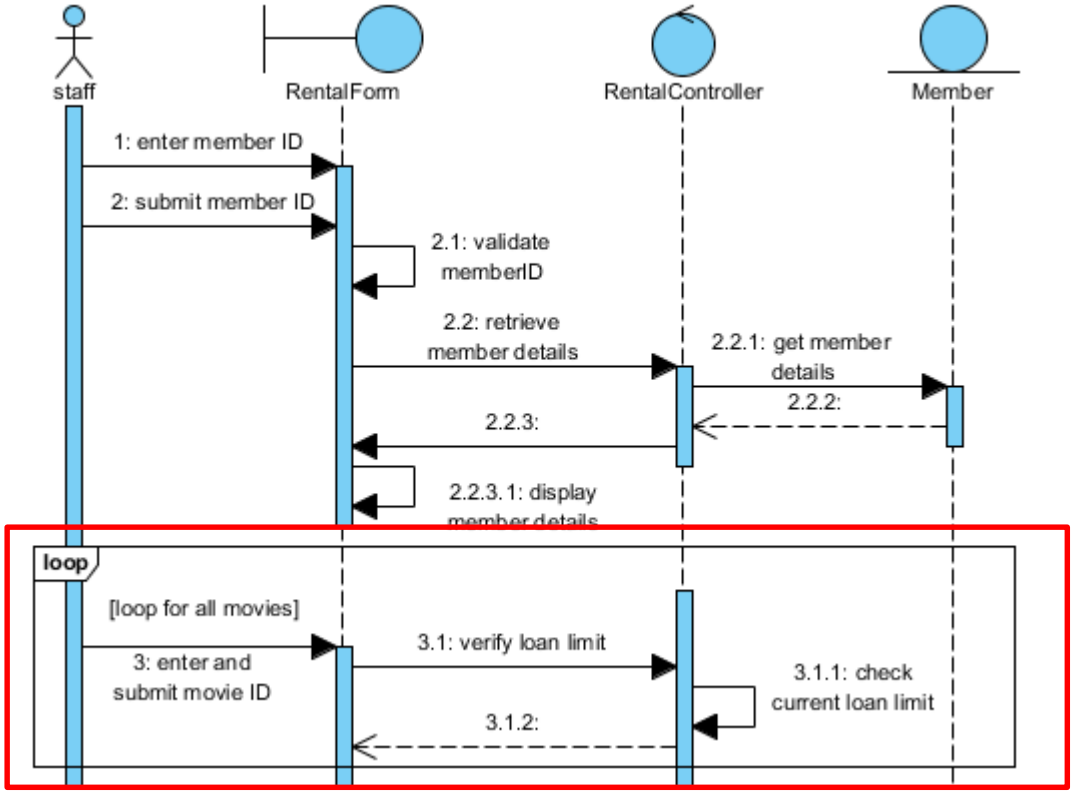
- The staff enters and submit member ID
- The system validates member ID
- The system retrieves and displays member details



Case Study – rent video use case

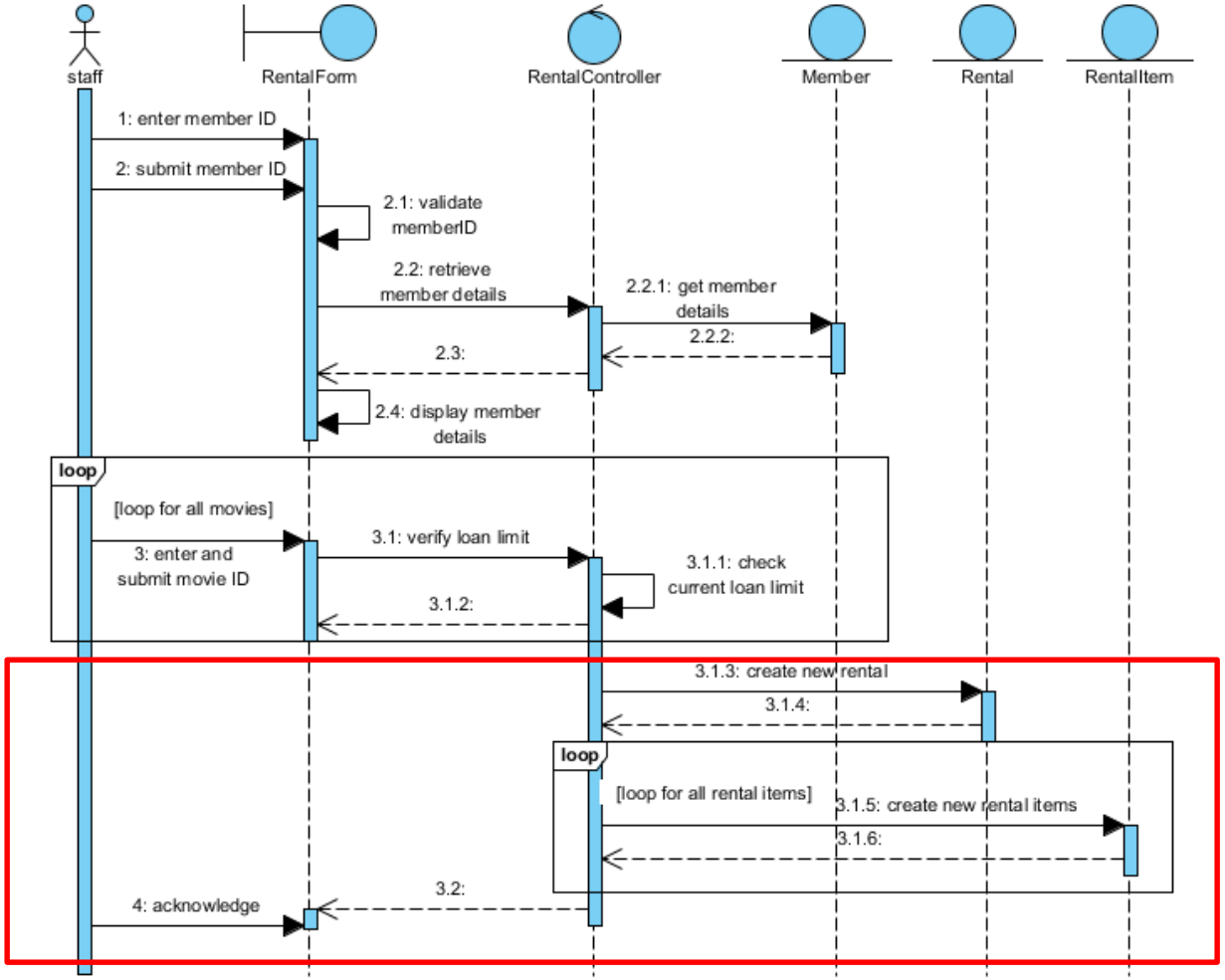
repetition

- The staff enters and submit movie ID
- The system verifies member's loan limit
- Repeat step 4 to 5 for all movie



Case Study – rent video use case

- The system creates rental and rental item records
- The system displays a successful rental message.
- The staff acknowledges the message.



Case Study – rent video use case

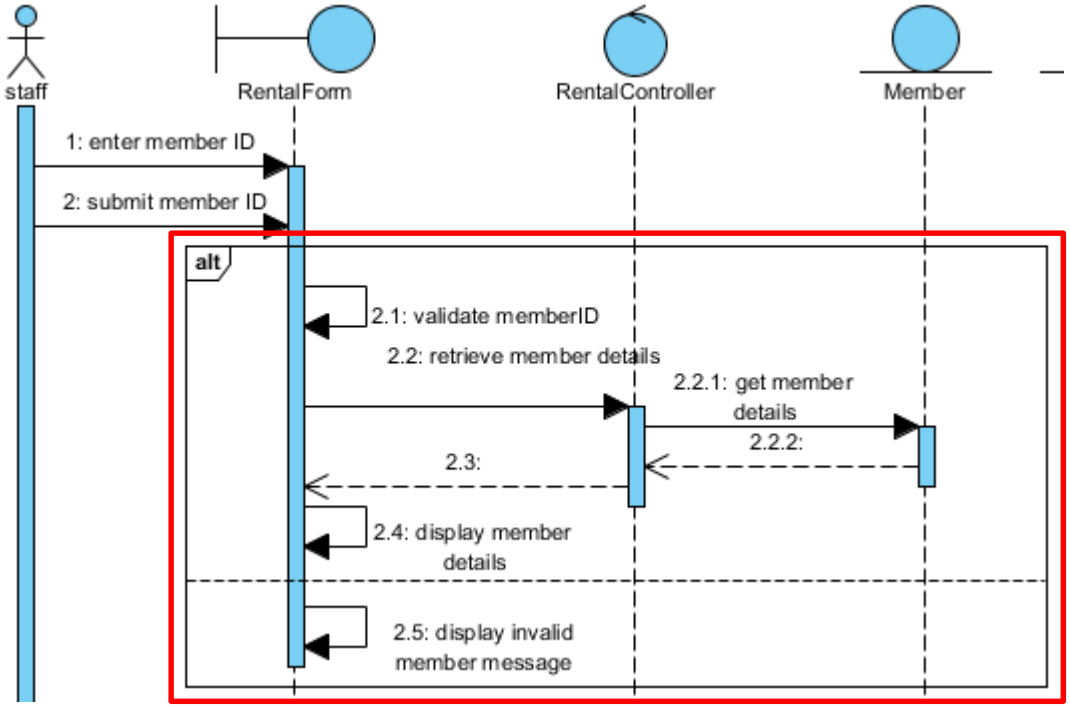
Alternate flow

Alternate Flow

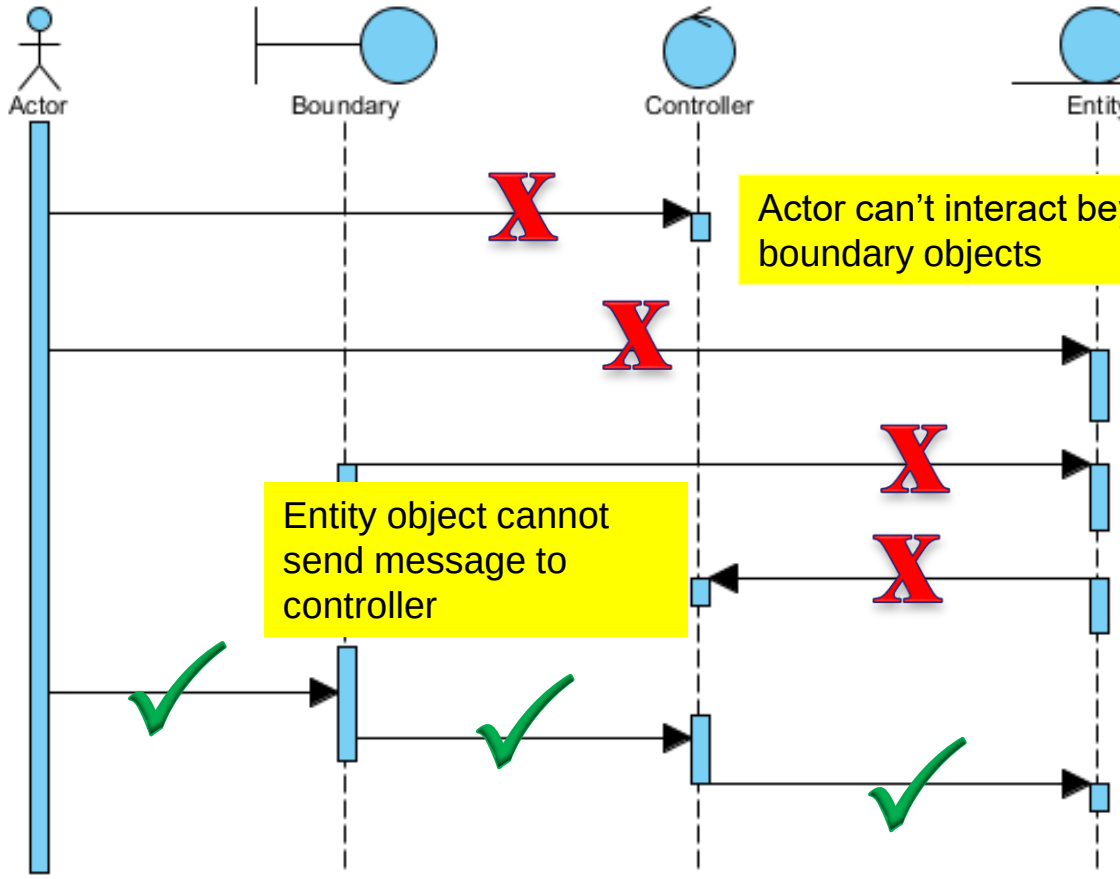
2a. Invalid member

i. The system prompt "Invalid Member" error message

ii. Use case ends



Common mistakes in constructing sequence diagrams



Summary

- 3 layer architecture
  - View layer
  - Business logic layer
  - Data layer
- Advantages and disadvantages of the 3 layer architecture
- Separation of concerns of the 3 layers
- Object interactions in a sequence diagram
- Constructing a sequence diagram
  - A step in the flow can be translated to a sequence of messages to different analysis objects
- Common mistakes in constructing a sequence diagram