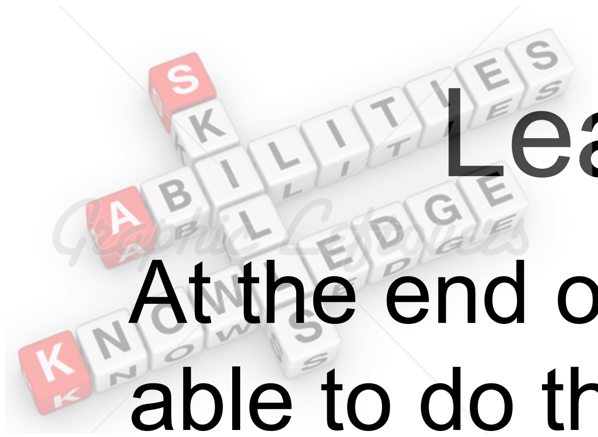
50.003 Elements of Software Construction Lecture 4

UML Sequence Diagrams 1

Scope

1. What is object-oriented analysis (OOA) ?
2. What is object-oriented design (OOD) ?
3. Realizing use cases through object-oriented analysis and design (OOAD)
4. What is a sequence diagram?
5. How to develop **sequence diagrams (or interaction diagrams)** as the main software model in analysis to understand how the new intended system is expected to perform



Learning Outcomes

At the end of this session, you should be able to do the following:

1. Explain the role of OOA & OOD in the SDLC context
2. Explain what use case realization means
3. Realize use cases through OOA & OOD
4. Describe UML sequence diagrams
5. Develop sequence diagrams.

Purpose of (Object Oriented) Analysis and Design (OOAD)

- The **<u>bridge</u>** between a user's requirements and the programming for the new intended system
  - "<u>Blueprints", or design models</u>, are necessary to build systems
- OOAD is an **<u>adaptive approach</u>** to development
  - Requirements and design are done <u>incrementally</u> within an iteration
  - A complete set of designs may not be developed at one time at the very first attempt.

Purpose of (Object Oriented) Analysis and Design (OOAD)

- The line between OOA and OOD is fuzzy and fluid
- Models created in OOA are <u>refined</u> and <u>extended</u> in OOD to produce systems design

What is Object Oriented Analysis (OOA)?

- OOA – is the process of <u>analysing</u> a task or a problem domain to develop conceptual models that can then be used to develop solutions to complete the task.
- Conceptual models:
  a) a set of use cases,
  b) one or more UML domain class diagrams,
  c) a number of interaction diagrams.
  d) It may also include some kind of user interface mock-up.

What is Object Oriented Analysis (OOA)?

- IN OOA, we:
  a) try to understand WHAT the problem is.
  b) capture software requirements in the use case model which consists of a use case diagram and detail description of each use case.
  c) capture “things” users deal with and assign responsibilities to these “things” to fulfil the requirements using sequence diagram

What is Object Oriented Design (OOD)?

- In OOD, implementation constraints are applied to the conceptual models to derive a design or solution model.
- Constraints could be:
  - imposed by the chosen architecture but also
  - any non-functional, technological or environmental constraints,
    - such as transaction throughput, response time, run-time platform, development environment, or those inherent in the programming language.

What is Object Oriented Design (OOD)?

- IN OOD, we:
  - focus on HOW to solve the problem.
  - convert the high-level responsibilities into actual operations and update the details in the (design/solution) class diagram.
  - build a set of detailed object-oriented design model (call solution or design model) needed for the final implementation (i.e. coding) by the programmers.

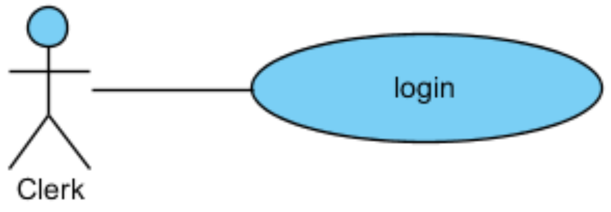
Use case realization

- To realise a use case is to make a use case executable
- To realise a use case, we adopt OOAD techniques to further develop the use cases
- Using OOAD we will understand what a use case is expected to do and to find a logical solution that is proper before coding

System Sequence Diagram (SSD)

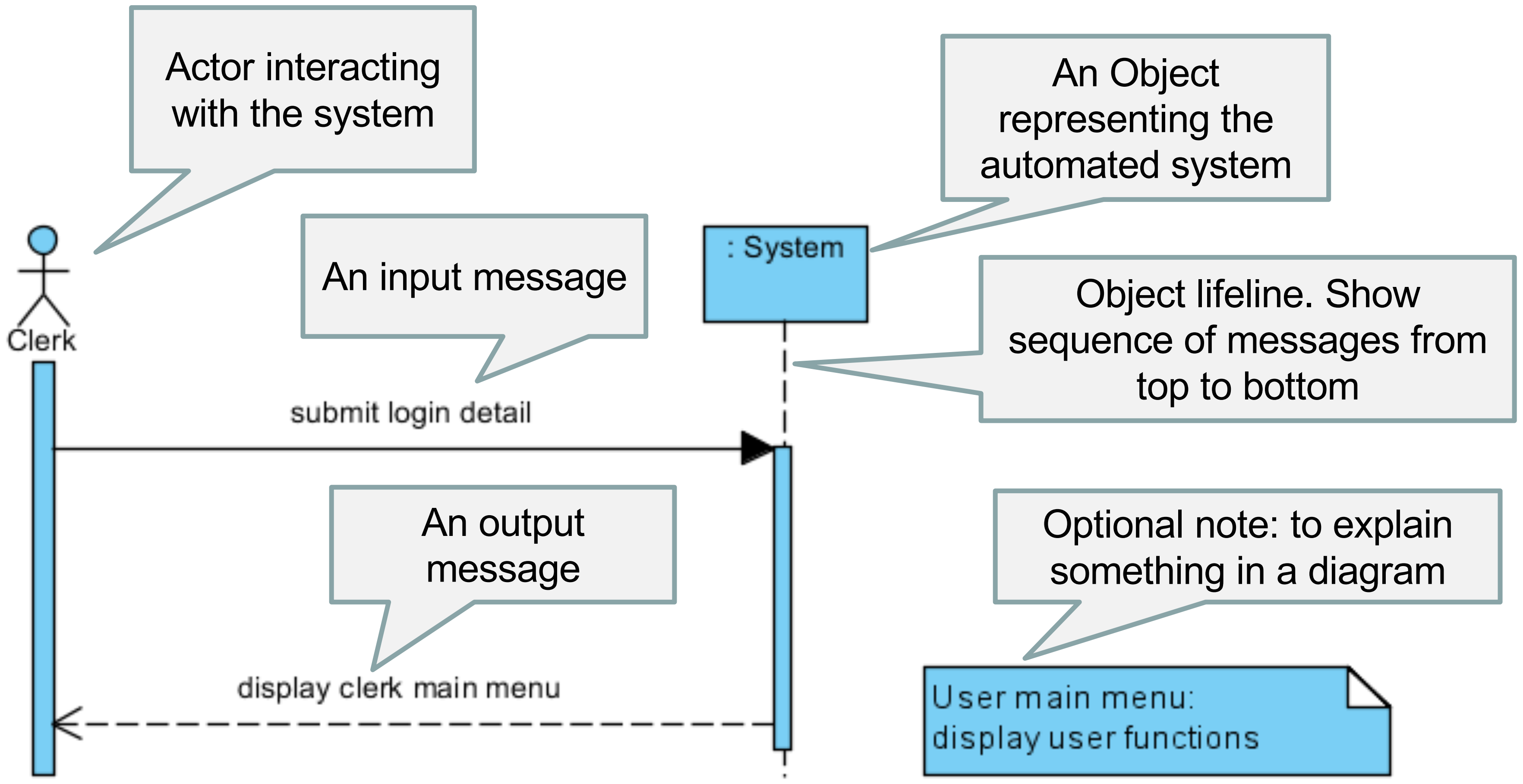
- A type of interaction diagram
- Describes the flow of information
- Identifies interaction between <u>actors</u> and <u>system</u>
- Message oriented

Eg:



login
Use case

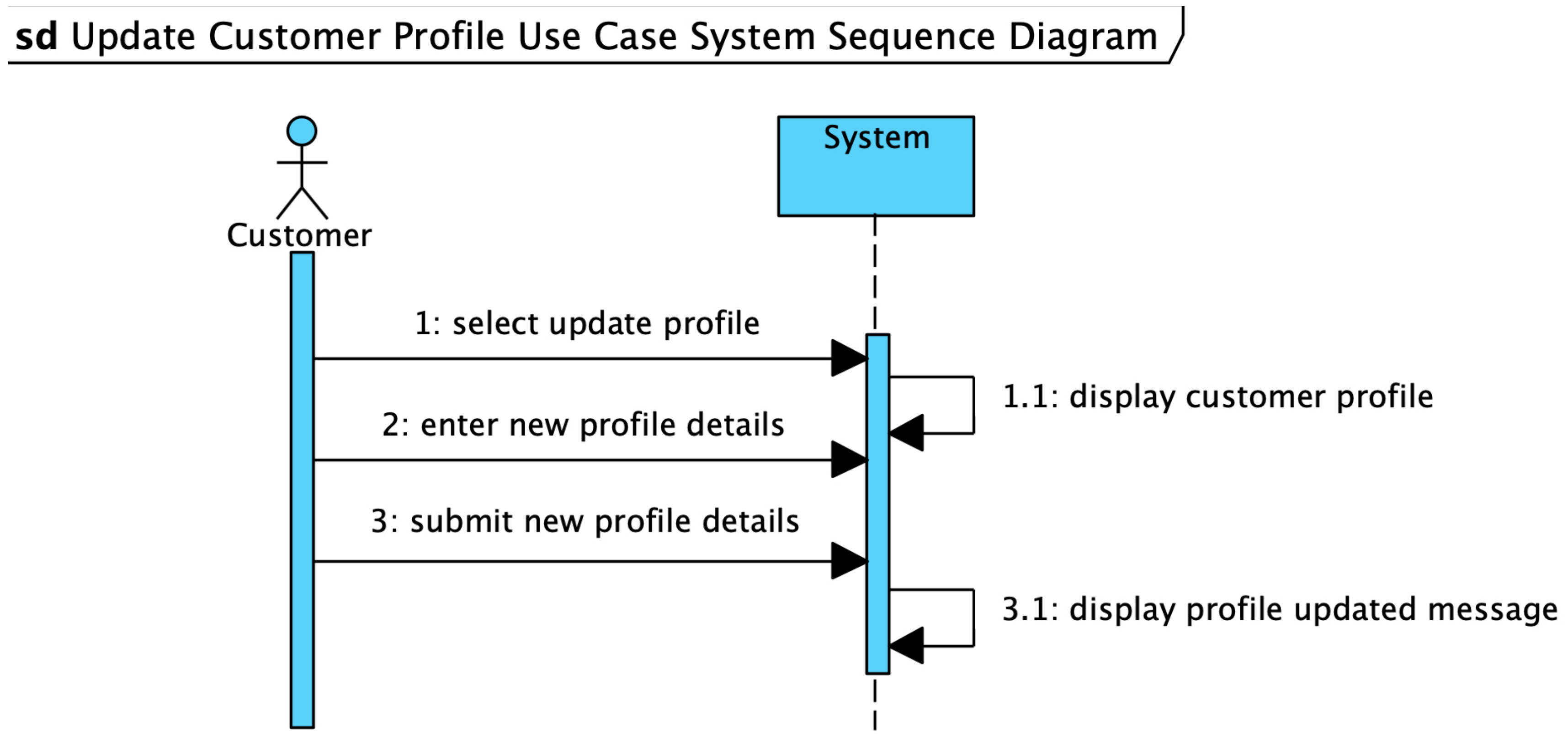
System Sequence Diagram (SSD)



Actor interacting with the system
An Object representing the automated system
An input message
: System
Clerk
Object lifeline. Show sequence of messages from top to bottom
submit login detail
An output message
Optional note: to explain something in a diagram
display clerk main menu
User main menu:
display user functions

Components of a SSD

Update Customer Profile Use Case SSD



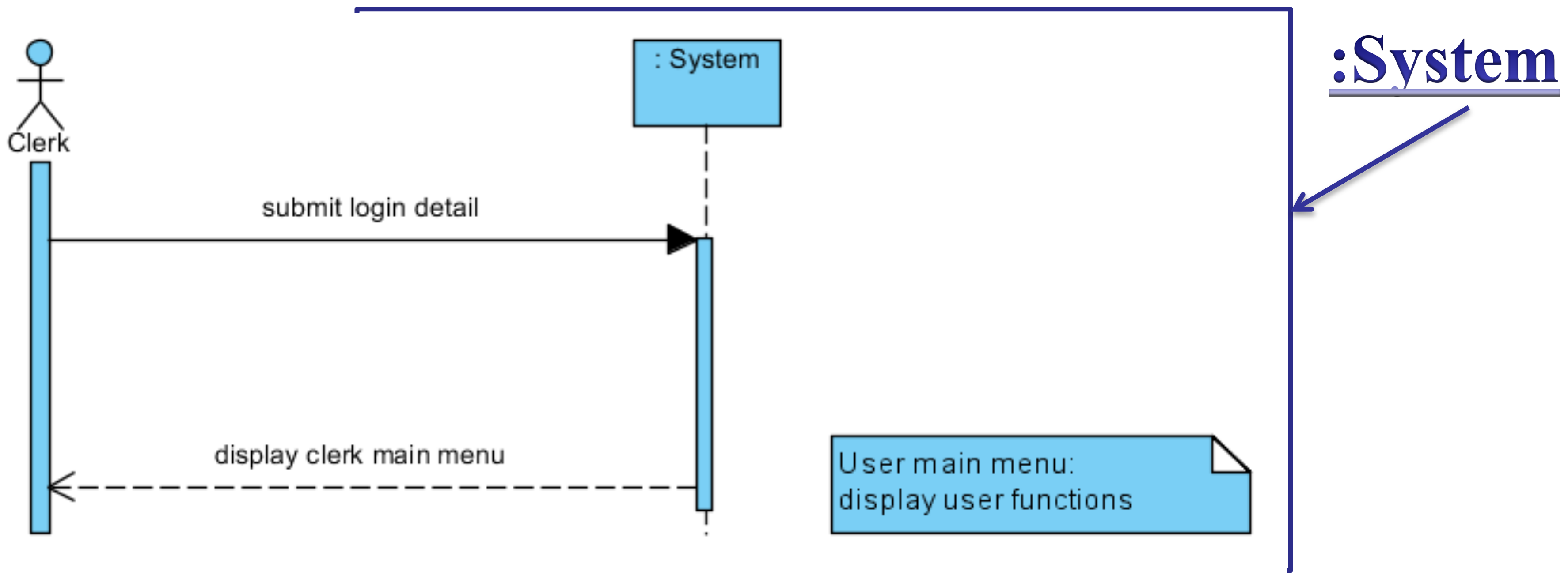
sd Update Customer Profile Use Case System Sequence Diagram
Customer
System
1: select update profile
1.1: display customer profile
2: enter new profile details
3: submit new profile details
3.1: display profile updated message

System Sequence Diagram (SSD)

- Gives a very high-level view of the interactions between the system (which is treated like a black box) and the external world represented by actors
- For developers, this is not good enough
- More details about the behind-scene is required to develop the intended system
- A more detailed sequence diagram for each use case is required – detailed sequence diagram

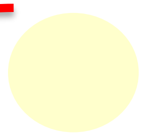
(Detailed) Sequence Diagrams (SD)

- A detailed sequence diagram uses all of the same elements as an SSD
- The :System object is replaced by internal objects and messages within the system

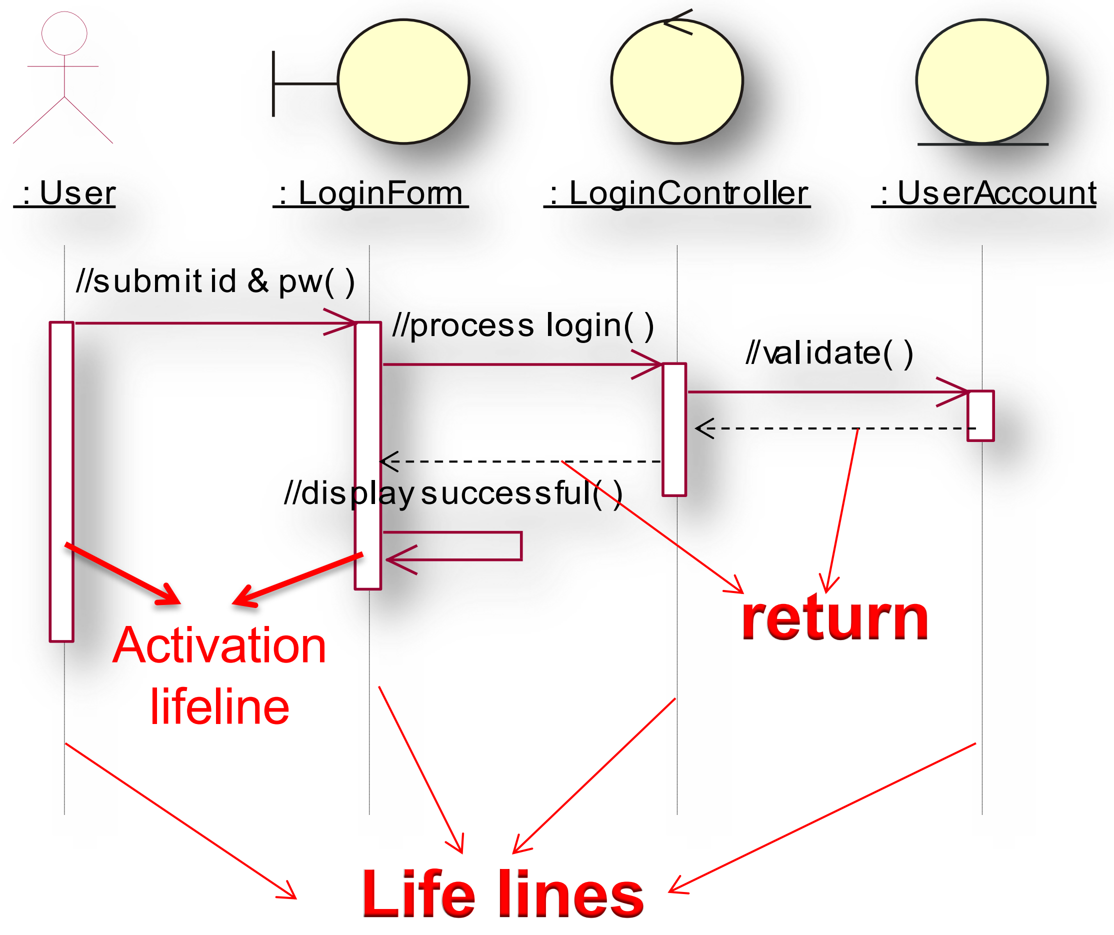


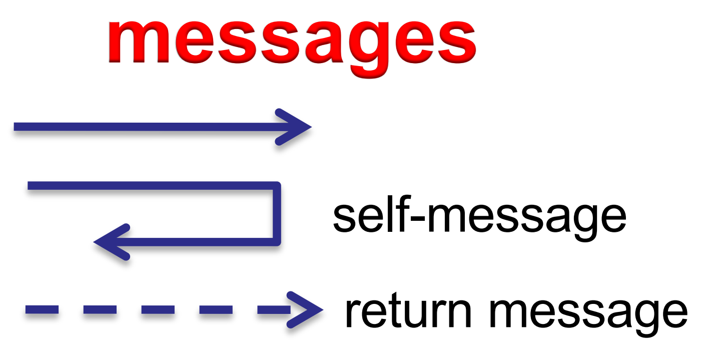
Sequence Diagram Notations

analysis objects

Iconic representation		Class template with stereotype	Purpose
 LoginAccount	or	<<**entity**>> LoginAccount	Stores/handles the data about login id and password Data Layer
LoginController	or	<<**control**>> LoginController	handles the process of validating the login Business Logic Layer
LoginForm	or	<<**boundary**>> LoginForm	allows the users the means to enter id and password Presentation Layer

Sequence Diagram Notations





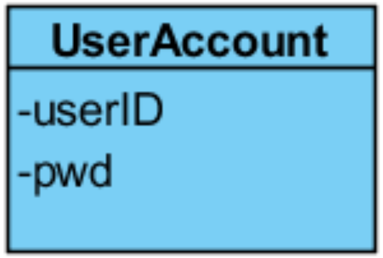
This is the notation for message exchanged among the objects.

These messages are identified from the flow of events in the use case description.

By indicating these **messages**, we are basically specifying the **responsibilities of these objects.**

Software Models required for Sequence Diagram

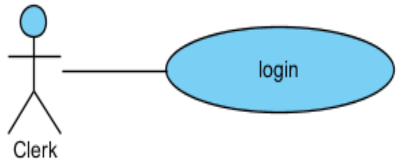
Domain model



- Use to decide which domain class(es) should be included in a particular sequence diagram as entity objects (known as participating classes)
- The class attributes can help to decide the participating classes:
  - Decide which attributes of a class where process data needed can be found.
  - Decide which attributes of a class where new output data to be created should be stored.
    - to verify login, data from the UserAccount attributes userID and pwd are need
    - To create a new account, the new userID and pwd data has to be stored in a new UserAccount userID and pwd attributes.

Software Models required for Sequence Diagram

Use case model



use case description

- Each use case has a corresponding detail sequence diagram
- Steps in the use case description are used to assign responsibilities to the various analysis objects in the sequence diagram
  - Each step in the use case description might be achieved by one or more messages sent to different analysis objects.

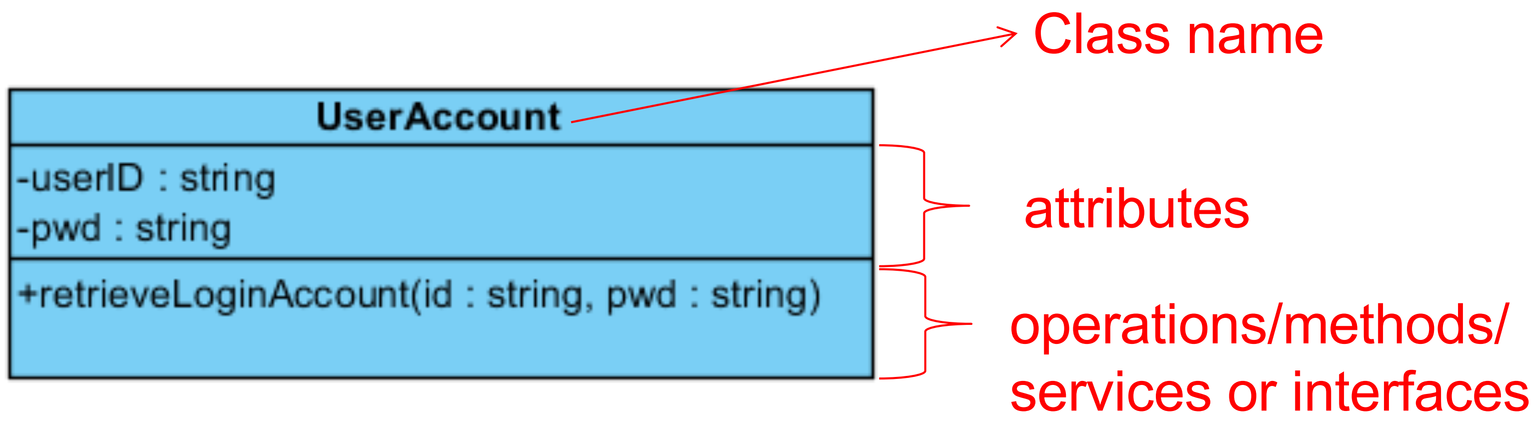
Object-Oriented Programs

- Consist of a set of software objects that cooperate and interact to accomplish a task
  - Each object has <u>program logic and data</u> encapsulated within it
  - Objects <u>send each other messages</u> to collaborate
- Two major areas of responsibilities
  - Knowing - Knowledge about its own data and about other classes with which it must collaborate to carry out use cases (attributes)
  - Doing -The activities an object carries out to assist in the execution of a use case (operations)

Purpose of Sequence Diagram

- Shows the interactions of different analysis objects (i.e. behavioral aspects of the system) when use cases execute.
- The interactions are depicted as messages
- messages are used to delegate responsibilities to the various analysis objects
- These responsibilities are used later to:
  - define the operations for the class of objects.
  - refine and extend the domain model into a Design/Solution Class model by defining all the operations

Design/Solution Class (Model)



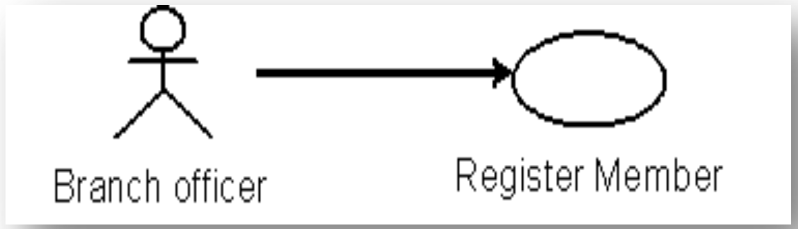
Class name
UserAccount
-userID : string
-pwd : string
+retrieveLoginAccount(id : string, pwd : string)
attributes
operations/methods/ services or interfaces

- A design class template consists of : class name, its attributes and its operations (methods)
- Relationships among the classes e.g. using association and generalization
- The domain and design models show the definition of a class and how it is related to other classes (i.e. the structural aspect of the system)

Guidelines to construct a Sequence Diagram

1. Understand the flow of the events for each use case
   - How? – read your detailed use case description
2. Identify the **complete set of classes** that will be affected by each message
   - How? – identify the boundary, controller and entity class (this is a 3-tier system architecture)
3. Specify the responsibilities
   - How? Use the flow described in the use case description

Case Study – register member

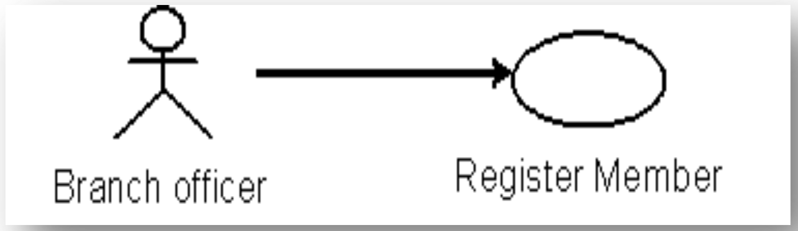


Consider the **Register Member** use case of the Video Rental System.

1. The branch officer enters and submits the customer details such as NRIC, name and contact information.
2. The system validates the submitted information.
3. The system creates a new member record with a unique id.
4. The system displays a “successful registration” message with the new member id.
5. The use case ends

Next -> Identify the classes needed (boundary, control and entity)

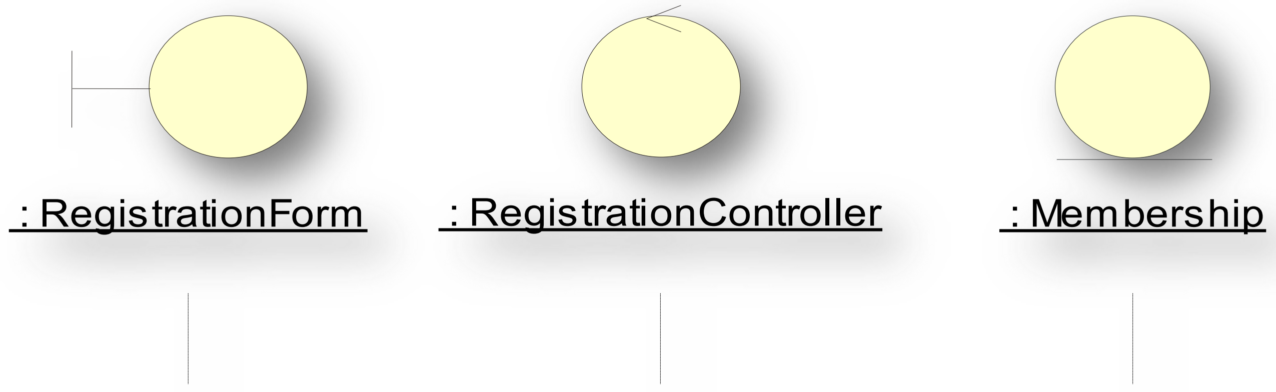
register member SD: Identifying analysis objects



1. There is one actor for this use case; hence, only **one boundary class** is needed.
   - The boundary class represents the user interface form.
2. Every use case should have a control class to coordinate the activities; hence, **a control class** is needed.
3. Since a member record will be created for the membership, the appropriate entity class would be **Membership** with the responsibility to create and to store the details.
   - **Suitable entity classes can be identified from your domain model.** (But not ALL are needed in a particular use case)

register member DSD : Identifying analysis objects

- Analysis classes identified are:
  - RegistrationForm **(boundary)**
  - RegistrationController **(control)**
  - Membership **(entity)**
- They are represented as follows:

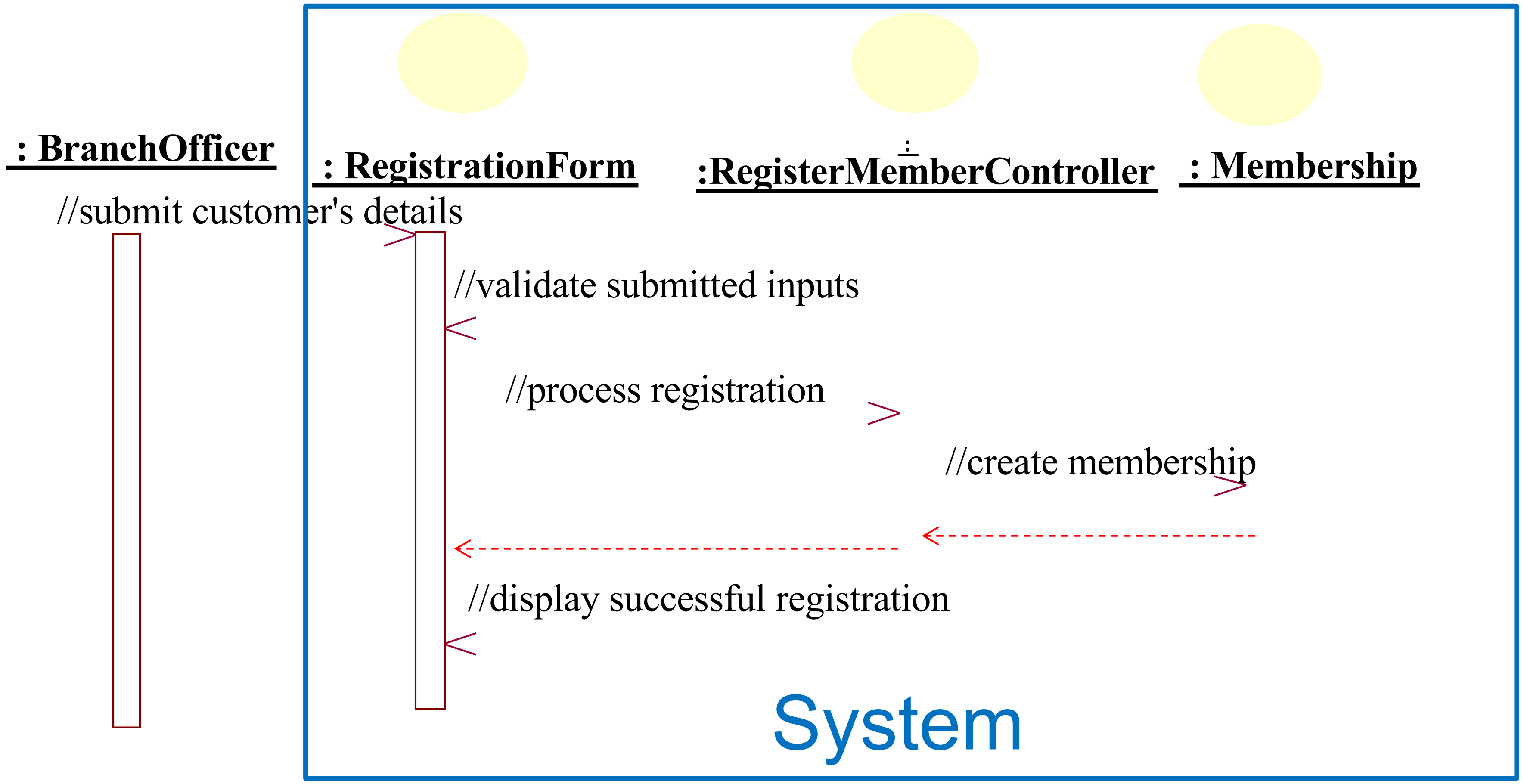


Next -> Identify the responsibilities

Visual Paradigm

- a suite of design, analysis, and management tools
- create a wide range of diagrams, including flowcharts, UML diagrams

register member SD : Assigning responsibilities to analysis objects


: BranchOfficer
: RegistrationForm
:RegisterMemberController
: Membership
//submit customer's details
//validate submitted inputs
//process registration
//create membership
//display successful registration
System

A sequence diagram showing how a branch officer can register memberships for a customer

Summary

- What is OOAD?
- Differences between OO Analysis & OO Design
- Software Models required to develop DSD
- Components of a Detail Sequence Diagram
  - 3 types of analysis objects: boundary, controller, entity
- Purpose of a DSD
  - Refine and extend the domain model into a solution model by identifying the responsibilities of analysis objects
- Guidelines to construct a more detailed Sequence Diagram
- Register Member use case

Cohort Exercise

Case Study

For each of the use case and use case descriptions you and your team had completed in week, identify all the analysis objects that will participate in the use cases sequence diagram. Explain why you had identified these analysis objects for the sequence diagram.