MUST

Wisdom & Virtue

MIRPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF SOFTWARE ENGINEERING

Software Design & Architecture

(Lecture # 2)

UML Use Case Modeling

Saba Zafar
(Lecturer)

Date: 7-11-2024

LECTURE CONTENTS

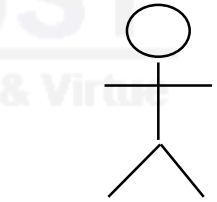
1. Notation used in Use Case Diagram
2. Extend Relation in Use Case Diagram
3. Include Relation in Use Case Diagram
4. Use Case Documentation

USE CASE DIAGRAM : NOTATIONS USED

- *System boundary*
- *Actors*
- *Use-cases*
- *Flow of information / stimulus*

ACTORS

- *An actor is an external agent that interacts with the system*
- *An actor stimulates the system with input events, or receives something from it*

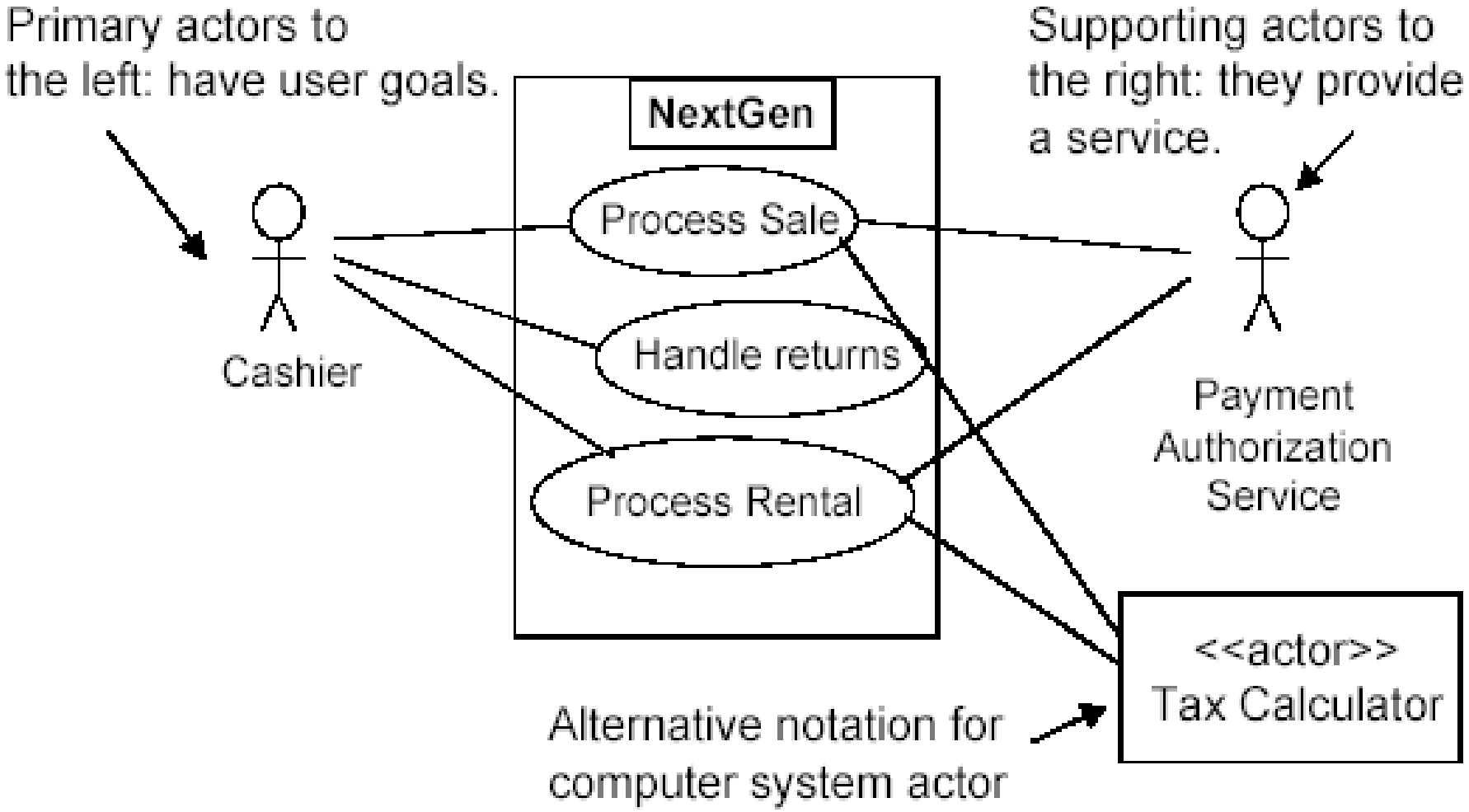


<actor's name>

USE CASE

- Describes a process from the user's point of view expressed in the user's language
- A collection of interactions between the system and actors
- A Use Case is an end-to-end process description that includes many steps or transactions

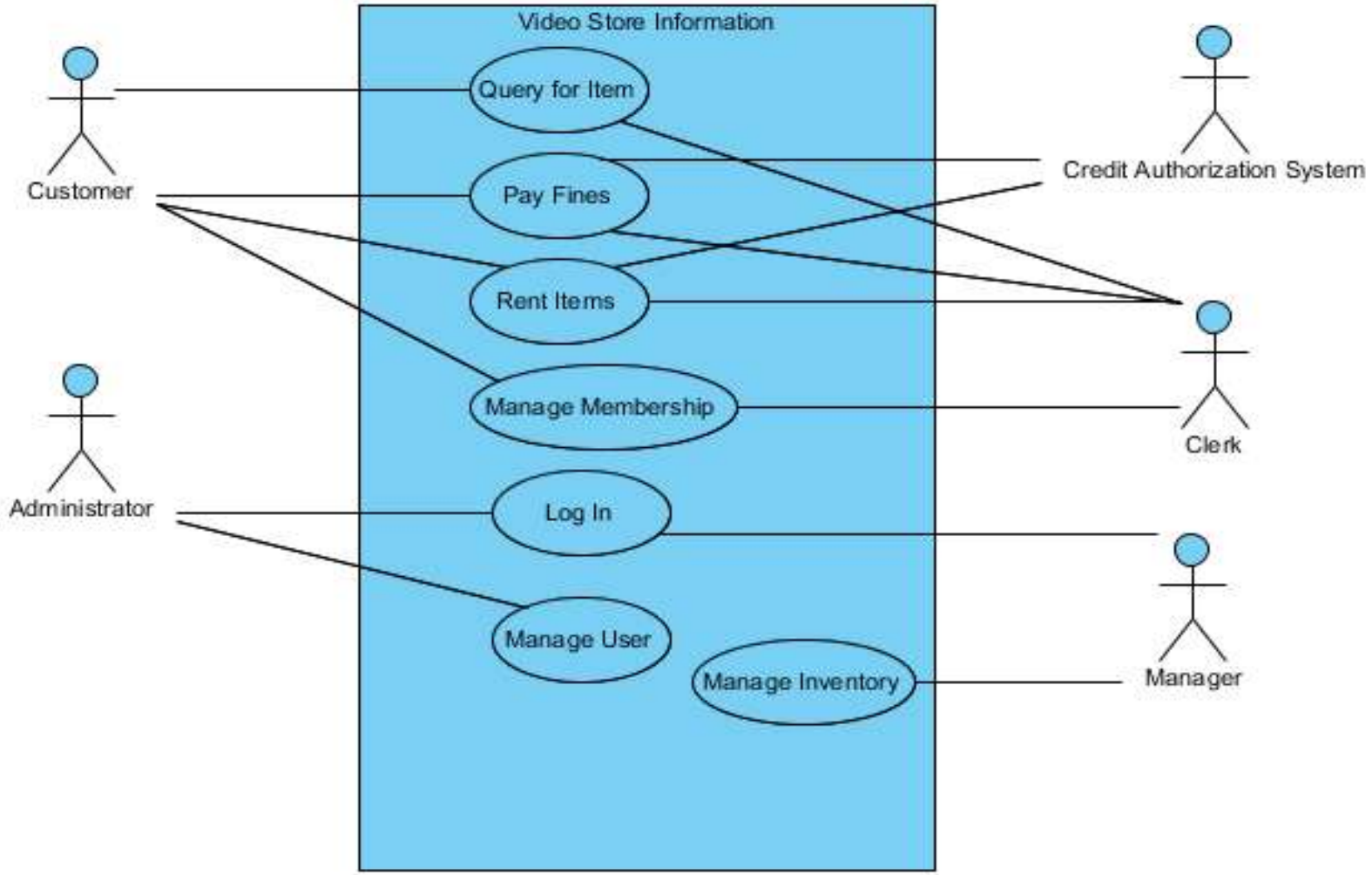
EXAMPLE: POS


Primary actors to
the left: have user goals.
Supporting actors to
the right: they provide
a service.
NextGen
Process Sale
Handle returns
Process Rental
Cashier
Payment
Authorization
Service
<<actor>>
Tax Calculator
Alternative notation for
computer system actor

HOW TO PROCEED?

- Choose the system boundary
- Identify primary actors
  - Those that have user goals fulfilled through using services of the system
- For each actor, identify their user goals
- Define use cases that satisfy user goals; name them according to their goal

EXAMPLE: VIDEO STORE



TERMS AND CONCEPTS

- **Actor:**
  - *Something with behavior, such as a person, computer system, or organization, e.g. a cashier.*
- **Scenario:**
  - *Specific sequence of actions and interactions between actors and the system under discussion, e.g. the scenario of successfully purchasing items with cash.*
- **Use case:**
  - *A collection of related success and failure scenarios that describe actors using a system to support a goal*

EXTEND RELATIONSHIP OF USE CASES

- ***Extend*** *relationship is shown as a dashed line with an open arrowhead directed from the extending use case to the extended (base) use case*
- *The* ***arrow*** *is labeled with the keyword «extend».*
- *If an alternative course is itself a stand-alone use case, you can* ***extend*** *the normal course by inserting that separate use case into the normal flow*
- *The "Search Vendor Catalogs" use case extends the "Request a Chemical" use case*
- *In addition, the chemical stockroom staff use the stand-alone "Search Vendor Catalogs" use case by itself*

Include Relationship

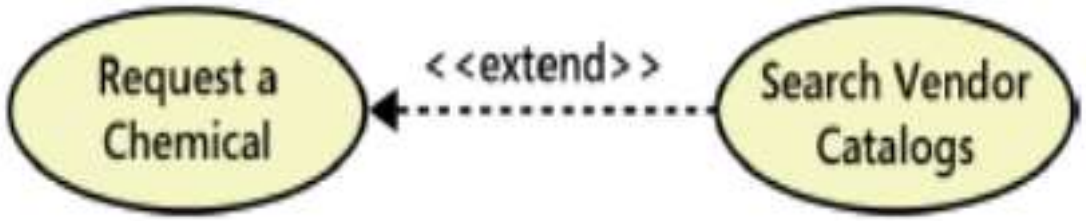
- Include Relationship:
- Denoted by a dashed arrow with an open arrowhead pointing to the included use case.
- It signifies that one use case (the base use case) incorporates the behavior of another use case (the included use case).
- The base use case includes the behavior defined in the included use case. In other words, the included use case is always executed whenever the base use case is executed.
- Typically used when one use case requires the behavior defined in another use case to complete its functionality.

Exclude Relationship

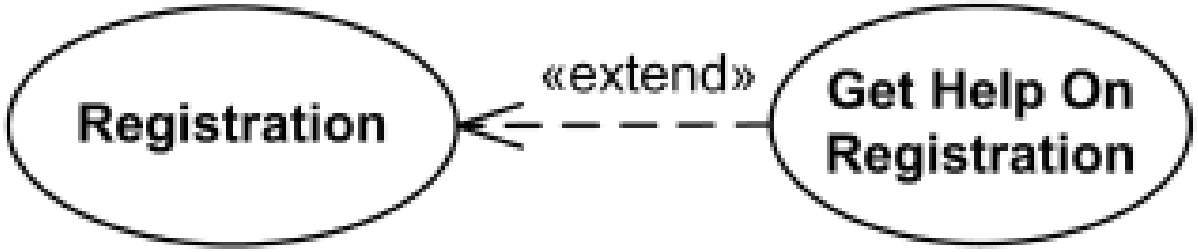
- Denoted by a dashed arrow with a crossbar (X) at the end pointing to the excluded use case.
- It signifies that one use case (the base use case) may choose not to execute the behavior of another use case (the excluded use case).
- The base use case may execute independently of the excluded use case, but under certain conditions, it may choose not to include or invoke the behavior defined in the excluded use case.
- Typically used when one use case represents an exceptional scenario that may or may not occur during the execution of another use case.

- Include Relationship:
- Symbol: Dashed arrow with an open arrowhead pointing to the included use case.
- Example: Base Use Case ----->> Included Use Case
- This symbolizes that the behavior of the base use case includes or incorporates the behavior of the included use case.
- Exclude Relationship:
- Symbol: Dashed arrow with a crossbar (X) at the end pointing to the excluded use case.
- Example: Base Use Case ----X Excluded Use Case
- This symbolizes that the base use case may choose not to execute the behavior defined in the excluded use case under certain conditions.

EXTEND RELATIONSHIP OF USE-CASES



Registration ***use case*** *with* ***extension points*** *Registration Help and User Agreement*

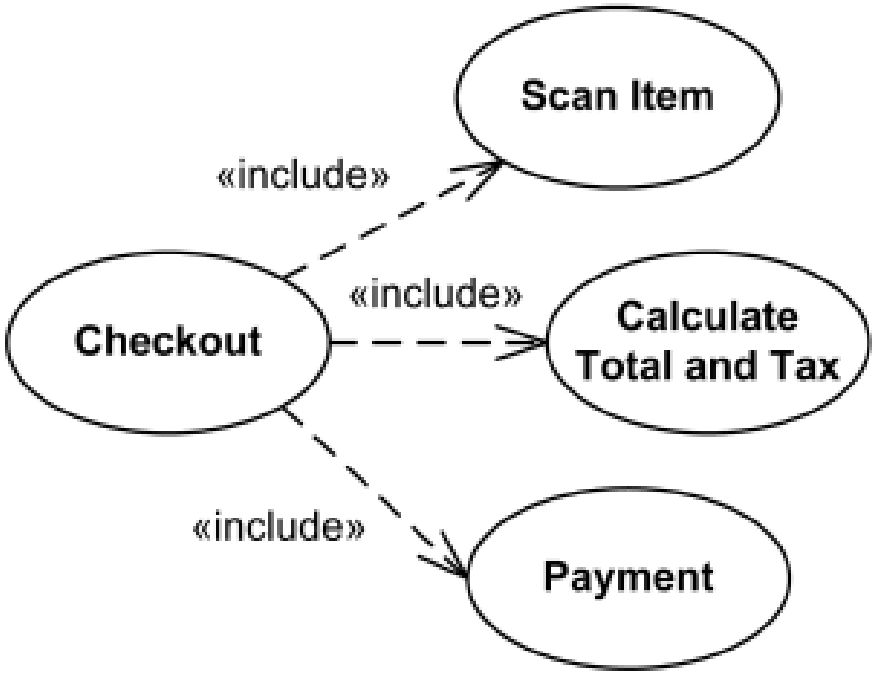


Registration *use case is complete and meaningful on its own.*
It could be extended with optional ***Get Help On Registration*** *use case.*

INCLUDE RELATIONSHIP OF USE-CASES

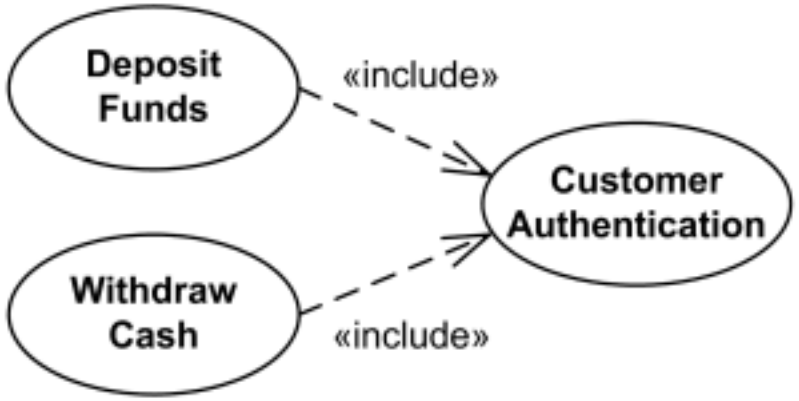
- The include relationship could be used:
  a. To simplify large use case by splitting it into several use cases
  b. To extract common parts of the behaviors of two or more use cases
- Sometimes several use cases share a common set of steps
- To avoid duplicating these steps in each such use case:
  - Define a separate use case that contains the shared functionality
  - Indicate that the other use cases *include* that sub use case

INCLUDE RELATIONSHIP OF USE-CASES



Checkout use case includes several use cases - Scan Item, Calculate Total and Tax, and Payment

INCLUDE RELATIONSHIP OF USE-CASES



Deposit Funds *and* ***Withdraw Cash*** *use cases include* ***Customer Authentication*** *use case.*

TASK 1

- In a scenario involving a **university library system**, the main goal is to streamline book lending and return processes, manage inventory, and support user accounts. There are three primary actors: **Students**, **Librarians**, and the **Library System** itself.
- **Students** can **search for books** using the system, **borrow books**, and **return books** upon completion. They can also **reserve books** that are currently checked out by others, receiving notifications when the book becomes available.
- **Librarians** have additional privileges: they can **add new books** to the system, **update book records** (such as availability, condition, or location), and **remove books** that are damaged or lost. They can also **manage student accounts** by verifying their eligibility to borrow or reserve books and handling any overdue penalties.
- The **Library System** automates certain tasks, like sending **overdue notifications** to students, **approving reservations** once books are available, and **tracking inventory** to help librarians manage stock.

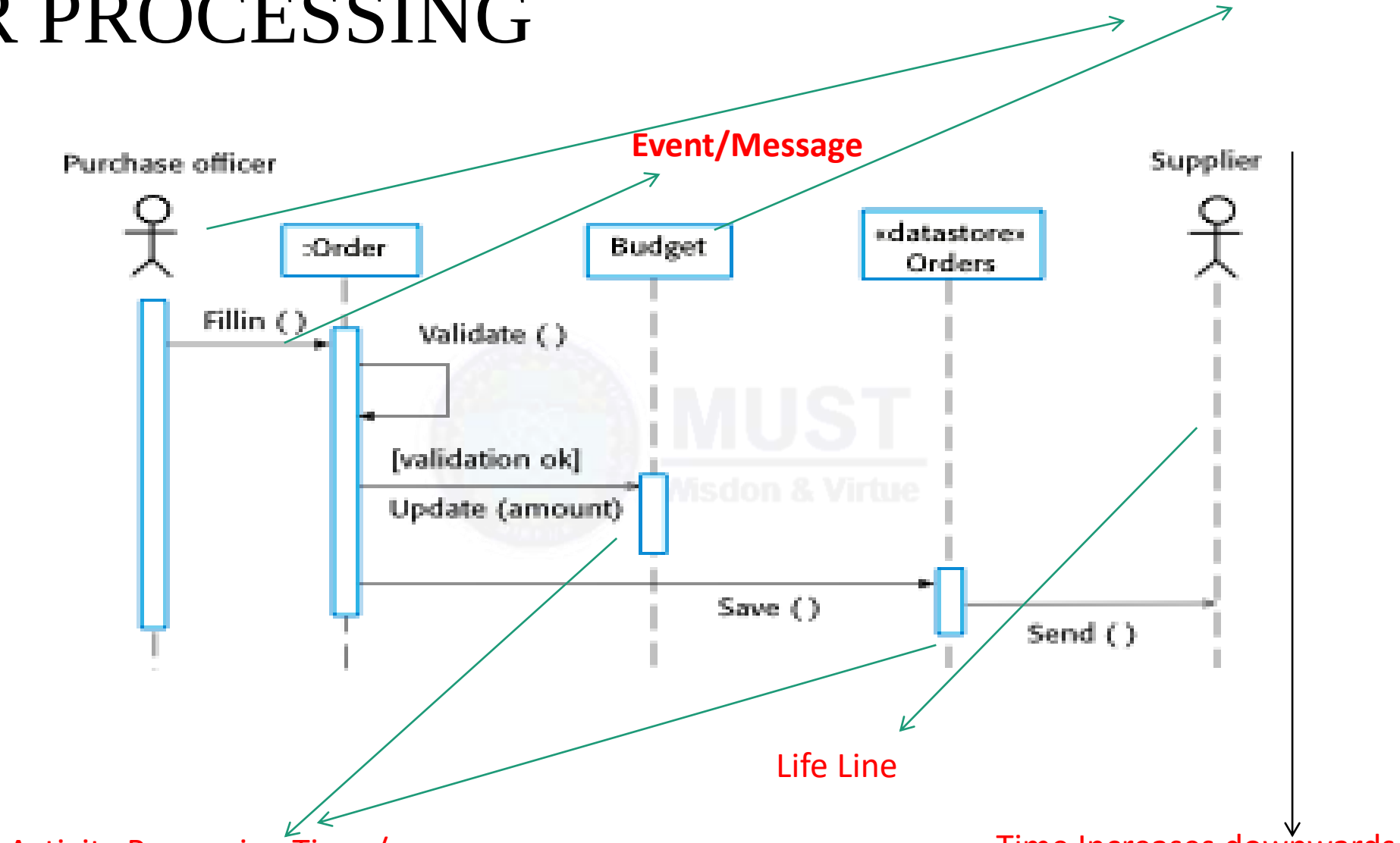
LECTURE CONTENTS

1. Sequence Diagram
2. Components of Sequence Diagram
3. Condition & Iteration in Sequence Diagram
4. Linking Sequence Diagrams

SEQUENCE DIAGRAM

- *Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case.*
- *Sequence diagrams are used to model the interactions between the actors and the objects within a system*
- *They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur, when a particular use case is executed*
- *The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these*
- *Interactions between objects are indicated by annotated arrows*
- *It shows dynamic communications between objects during execution of task*

ORDER PROCESSING



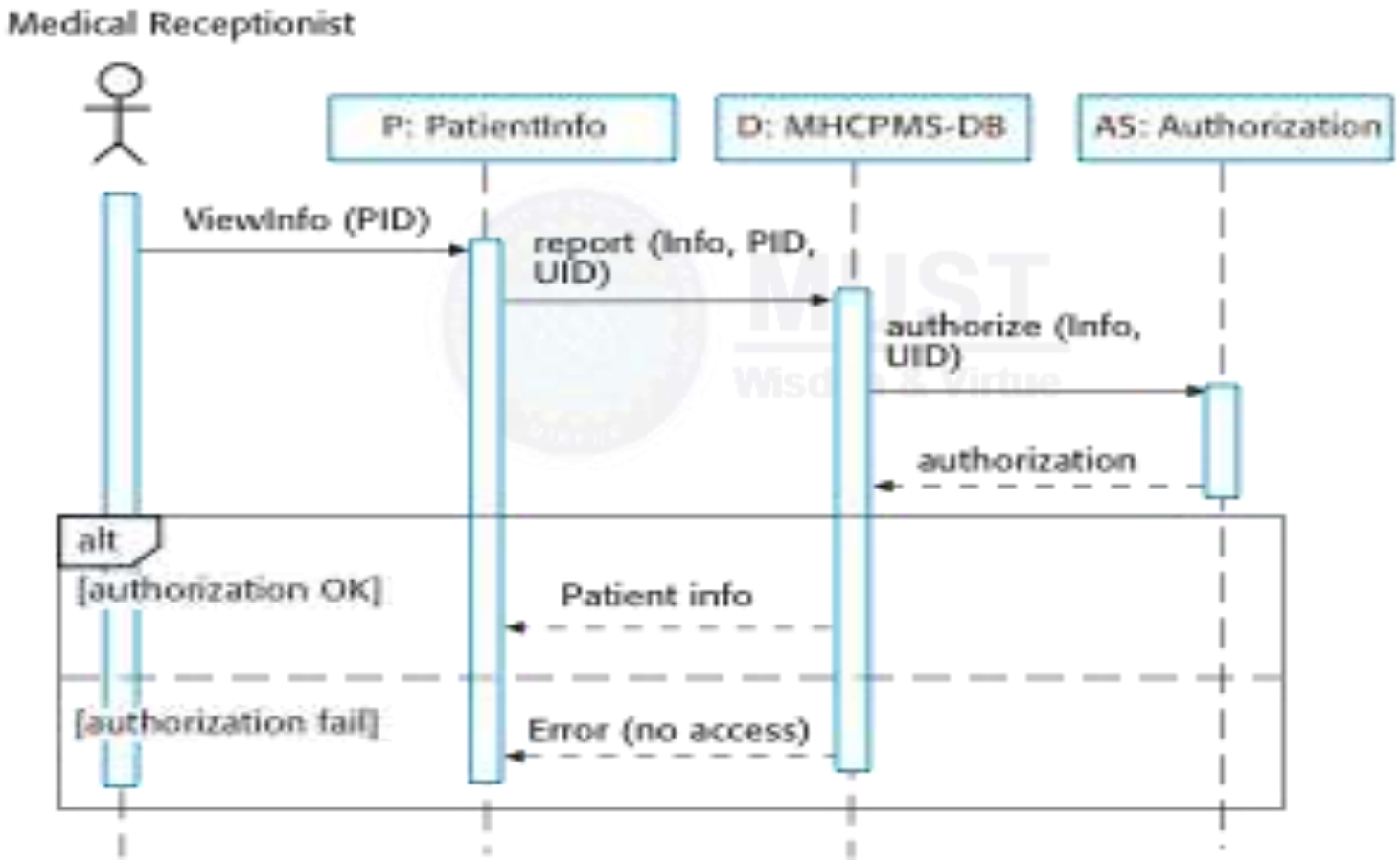
COMPONENTS OF SEQUENCE DIAGRAMS

- ***Classifiers***
  - *The boxes across the top of the diagram represent classifiers or their instances*
  - *Typically these show objects, classes, or actors (usually depicted as rectangles, although they can also be symbols)*
- ***Lifelines***
  - *The dashed lines hanging from the boxes are called object lifelines*
  - *It represents the life span of the object during the scenario being modeled*
  - *The long, thin boxes on the lifelines are method-invocation boxes (Activation Bar) that indicate processing is being performed by the target object/class to fulfill a message*

COMPONENTS OF SEQUENCE DIAGRAMS

- ***Modeling messages***
  - *Messages are indicated as labeled arrows*
  - *When the source and target of a message is an object or class, the label is the signature of the method invoked in response to the message*
  - *All parts except the message_name are optional*
  - *However, if either the source or target is a human actor, then the message may be labeled with brief text describing the information being communicated*

SEQUENCE DIAGRAM FOR VIEW PATIENT INFORMATION



SEQUENCE DIAGRAM OF "SET TIME"

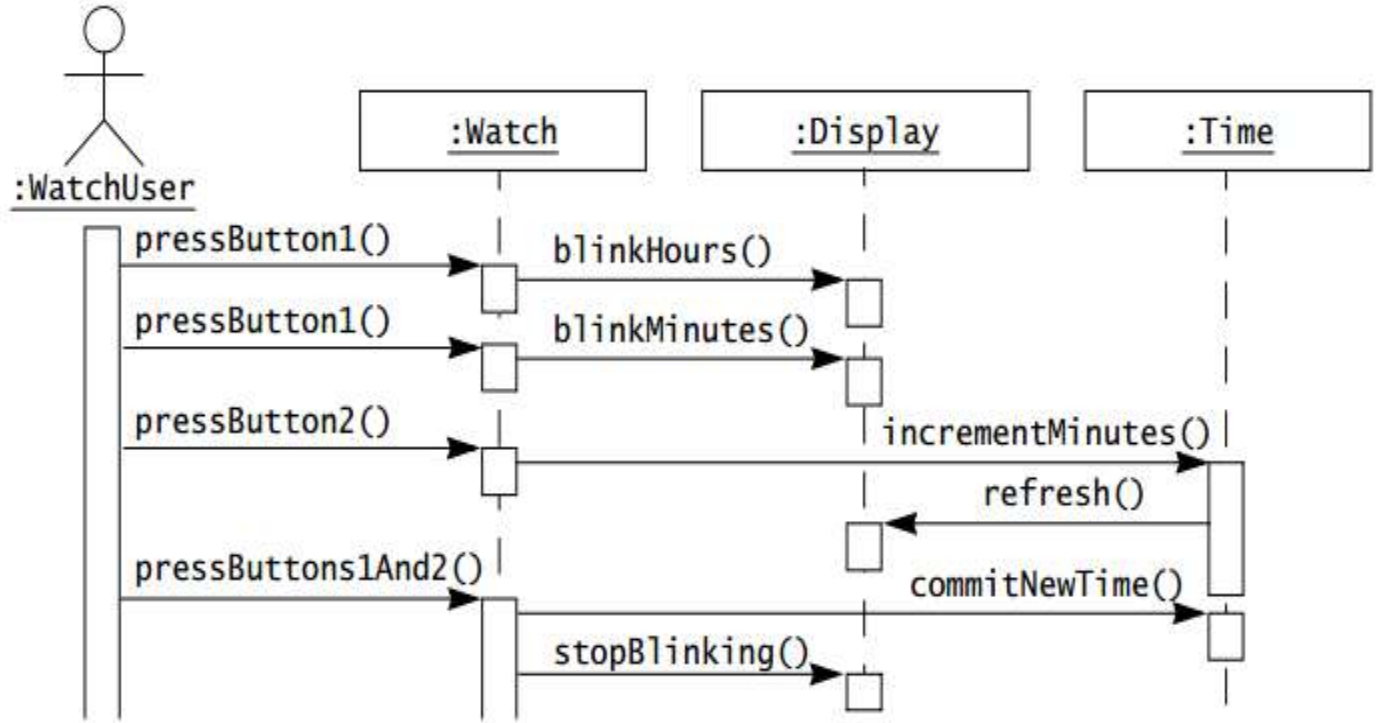
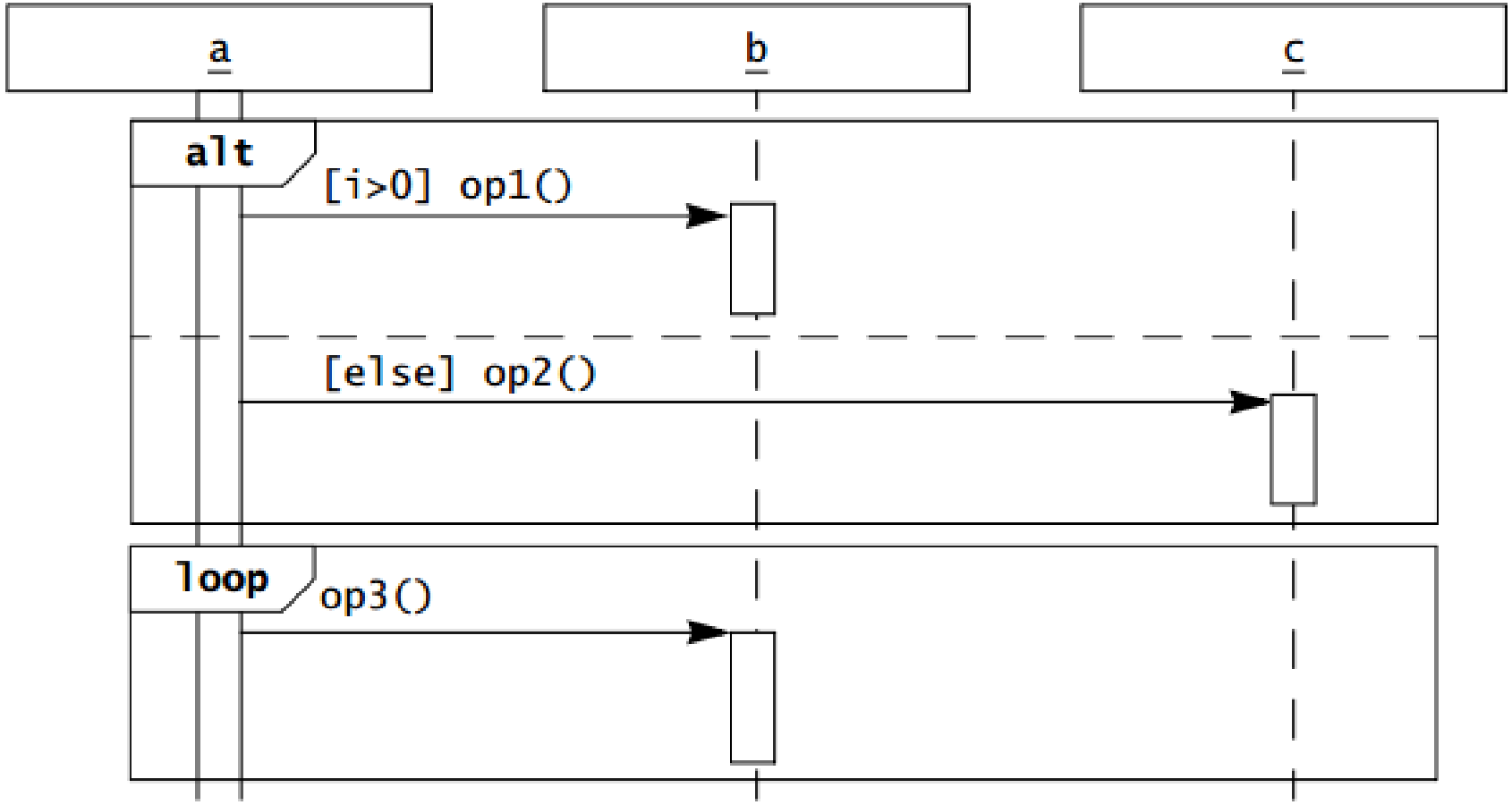


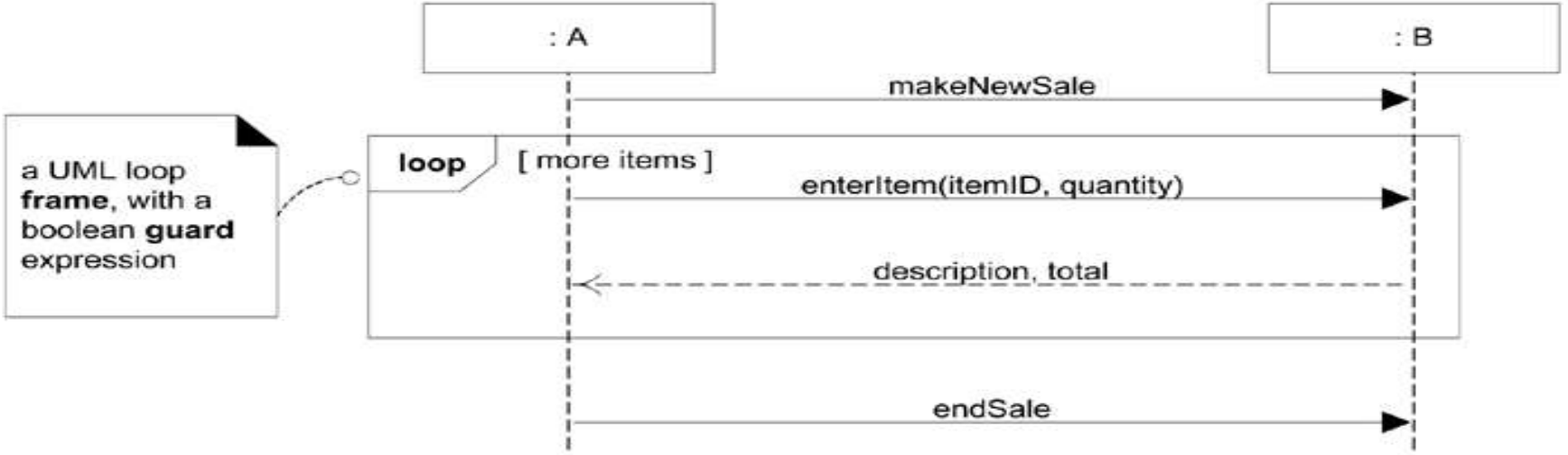
Figure 2-3 A UML sequence diagram for the `Watch`. The left-most column represents the timeline of the `WatchUser` actor who initiates the use case. The other columns represent the timeline of the objects that participate in this use case. Object names are underlined to denote that they are instances (as opposed to classes). Labeled arrows are stimuli that an actor or an object sends to other objects.

CONDITIONS AND ITERATION IN SEQUENCE DIAGRAM



SELECTION AND LOOPS

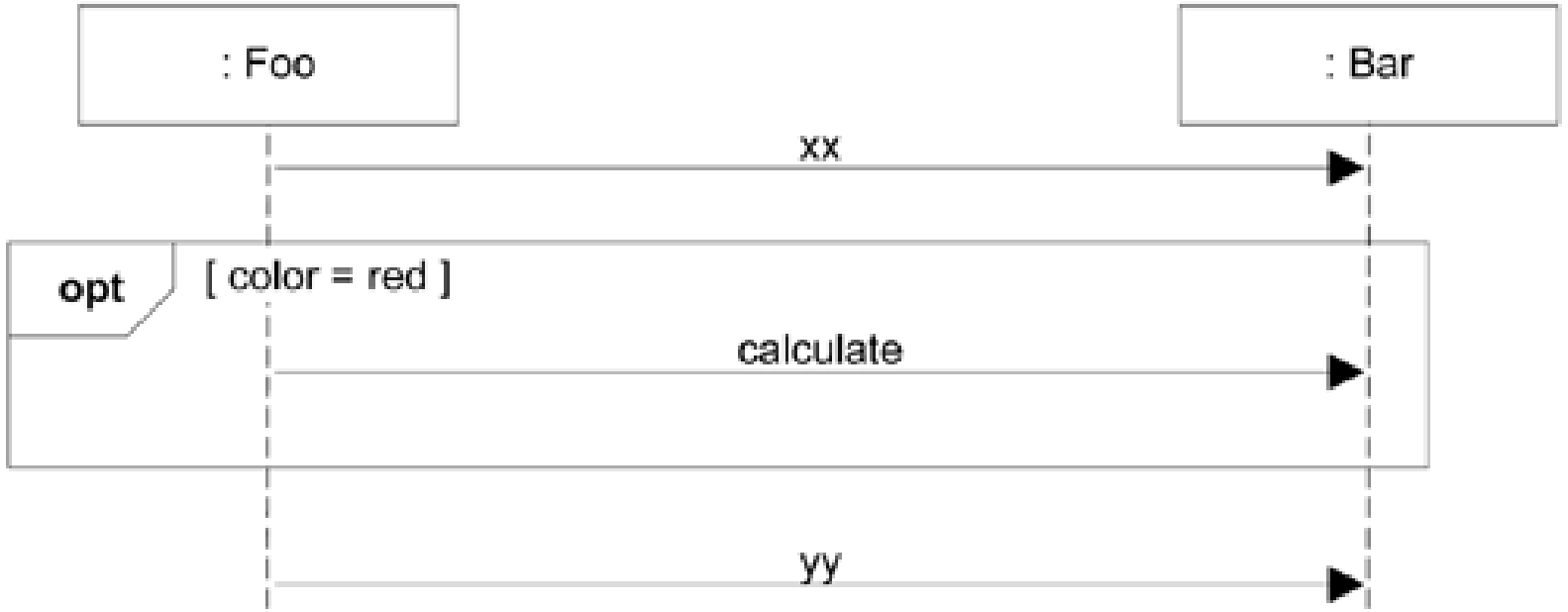
- ***Frame:*** *Box around part of a sequence diagram to indicate selection or loop*
  - ***if:*** *(opt) [condition]*
  - ***if/else:*** *(alt) [condition], separated by horizontal dashed line*
  - ***loop:*** *(loop) [condition or items to loop over]*



CONDITIONAL MESSAGES

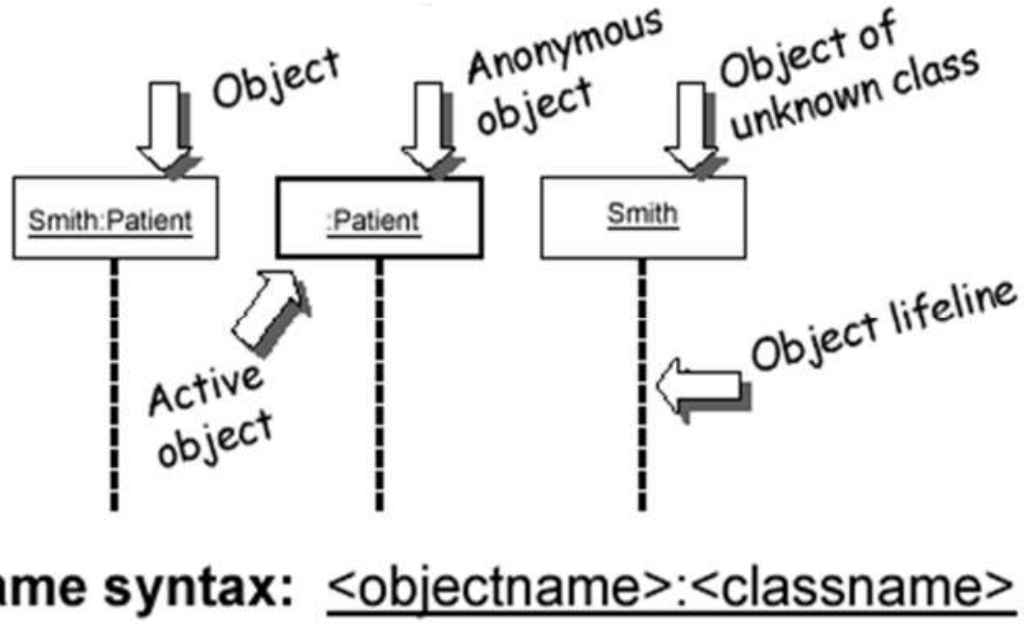
- *An OPT frame is placed around one or more messages. Notice that the guard is placed over the related lifeline*
- *Optional fragment that executes if guard (condition) is true.*

Figure 15.13. A conditional message.



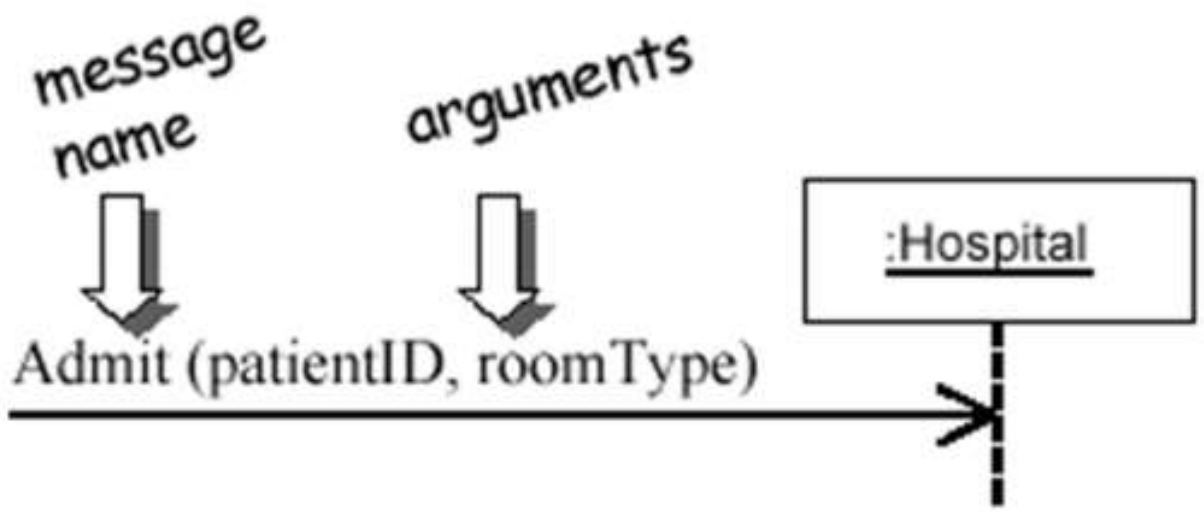
REPRESENTING OBJECTS

- ***An object:*** *A square with object type, optionally preceded by object name and colon*
  – write object's name if it clarifies the diagram
  – object's "life line" represented by dashed vertical line



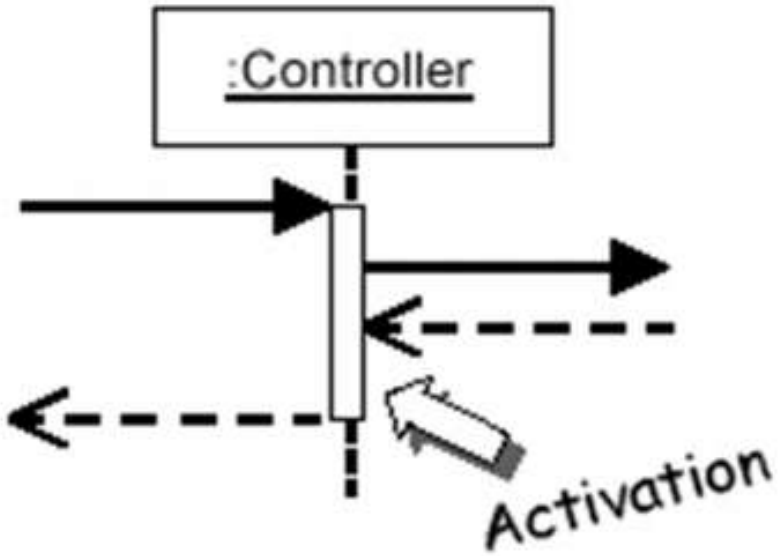
MESSAGES BETWEEN OBJECTS

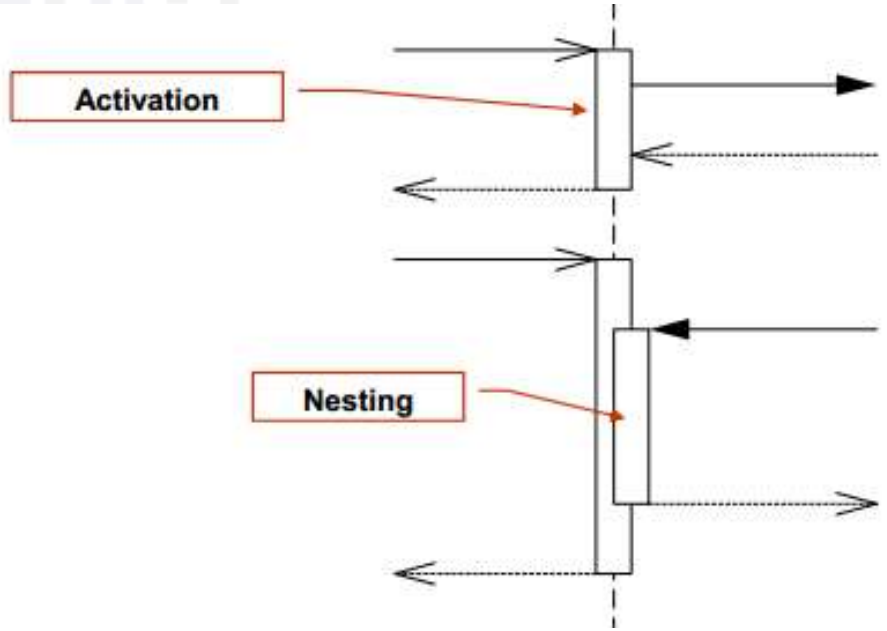
- *Message (method call): horizontal arrow to other object – write message name and arguments above arrow*



INDICATING METHOD CALLS

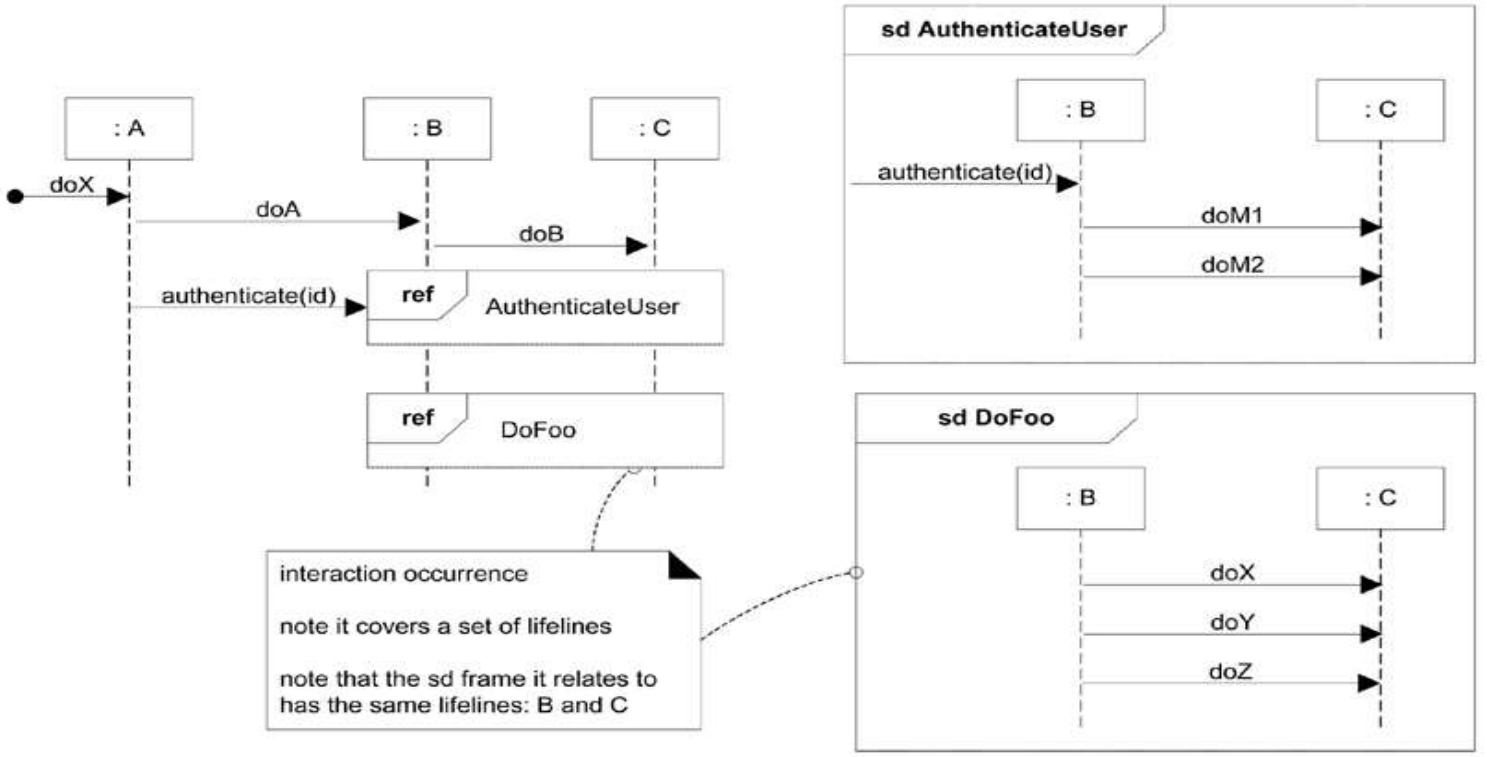
- ***Activation:*** *Thick box over object's life line*
  – Either: that object is running its code or it is on the stack waiting for another object's method
  – Nest to indicate self-calls and callbacks





LINKING SEQUENCE DIAGRAMS

- *If one sequence diagram is too large or refers to another diagram:*
- *It is useful, for example, when you want to simplify a diagram and factor out a portion into another diagram*
  *– a "**ref**" frame that names the other diagram*



THANKS