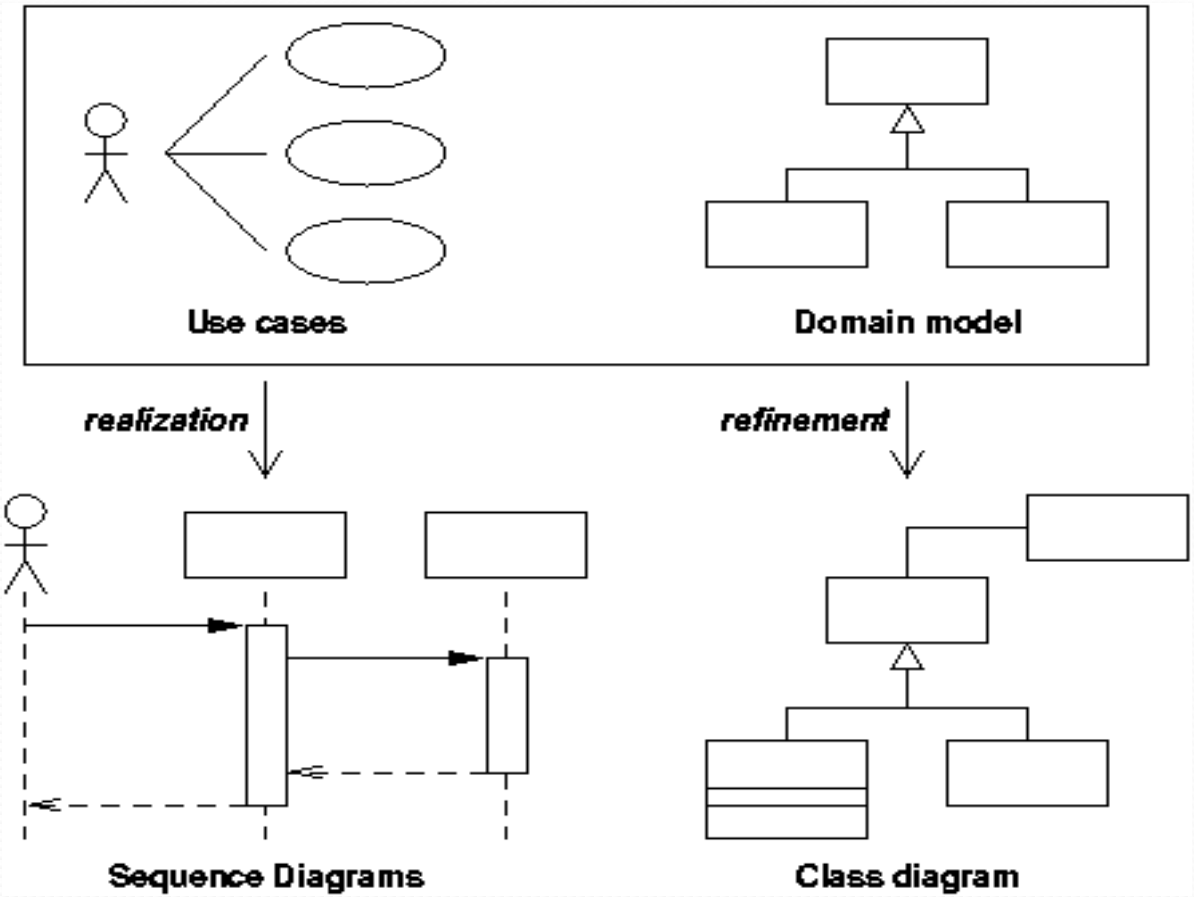
CT503/871 OOD: Lecture 4

- Lecture 4 Overview:
  - Systems Analysis: Interaction
  - UML Interaction Model:
    - UML Sequence Diagrams
    - UML Communication Diagram
  - UML Interaction Model Example: ATM

Analysis Model

- Goal of analysis is to specify the problem fully
- UML Systems analysis is concerned with:
  - Problem domain objects, modeled in the Class Model (Static), known as the Domain Model
  - The application interaction, modeled in the interaction and state models (dynamic), known as the Application Model
- This lecture considers the Interaction Model

Realisation & Refinement



Realisation & Refinement

- Use case realisations indicate how the functionality will be supported by the system
- Use case realisation: process of refinement
- Realisations are documented in UML interaction diagrams
- This causes the domain model to be refined into a more implementation-oriented class diagram

UML Interaction Model

- Objects interact by passing messages
- These messages define the system's behaviour
- The interaction model presents a simple, graphical representation of interactions between objects
- How objects interact to perform a task, discovery of relationships and associated operations

Objects and Responsibilities

- Object purpose: responsibility to do what we need
- Object's responsibilities = services it needs to provide (via operations)
- Responsibilities include two key items: knowledge and actions
- Services are provided by one object to another upon request: Client-server relationship

Collaborations

- A collaboration takes place when an object has a responsibility it cannot handle on its own
- **Collaboration**: request from a **Client** to a **Server** for a service to help it fulfil its responsibilities
- Interactions are defined by adding messages to collaborations

Interactions

- An **interaction** is a unit of behaviour of a classifier (use case in this context)
- Create one or more interactions (diagrams) to demonstrate how behaviour is realised
- An **interaction** defines the message passing between lifelines (e.g. objects) within the context of a collaboration to achieve a particular behaviour
- Uncover more of the operations and attributes of analysis classes through interaction analysis

UML Interaction Diagrams

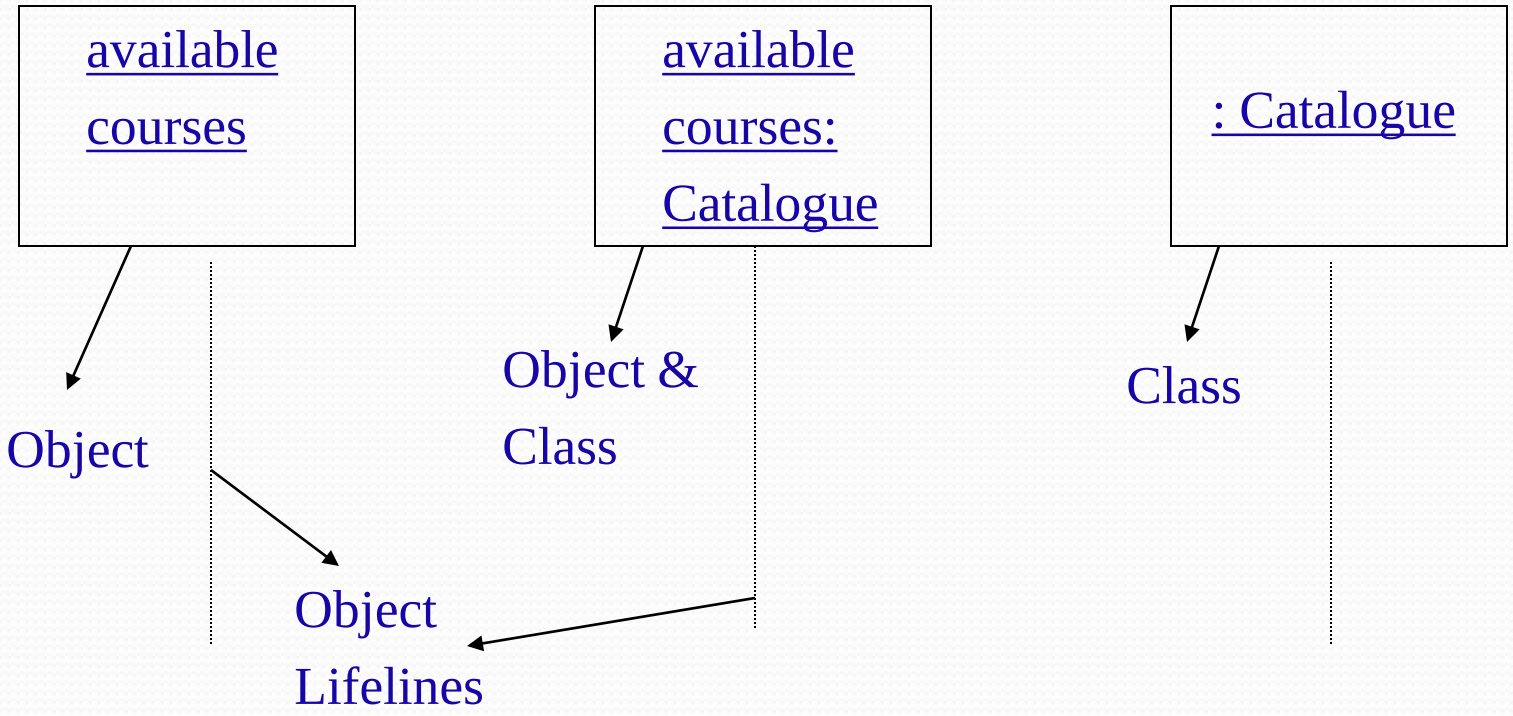
- Shows the behaviour of several objects within a single use case: ***use case realisation***
- Shows collaboration among objects; not precise behaviour
- UML defines two types of diagram for showing interactions (different views):
  - Sequence diagrams: most useful and used
  - Communication diagrams

1. UML Sequence Diagram

- Shows objects participating in and sequence of messages exchanged during the interaction
- Shows an interaction between lifelines (e.g. objects) arranged in a time sequence
- A sequence diagram contains:
  - **Objects** with their lifelines
  - **Messages** exchanged in time-ordered sequence
  - **Focus of control**: period of time during which an object is performing an action
  - **Scripts**: explanatory notes placed down the lhs

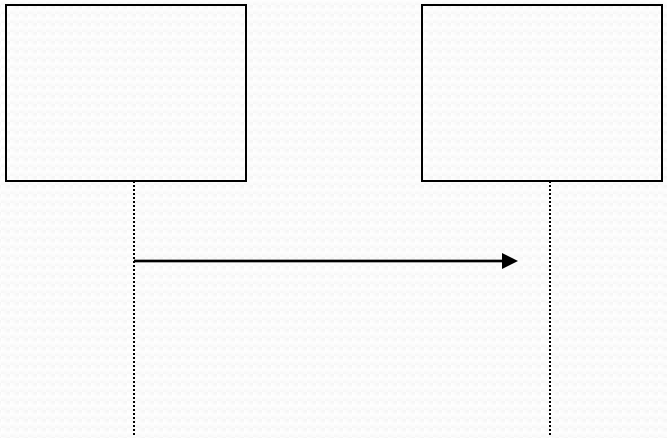
Sequence Diagram Notation

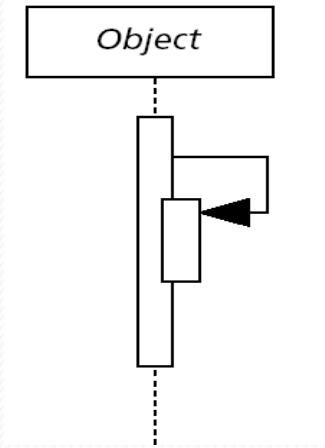
- Shows object interactions in time sequence



Sequence Diagram Messages

- Messages are shown as arrows directed from the vertical line of **client** object to the **supplier** object
- **Labeled** with message
- Vertical position indicates message **time ordering**
- Messages can also be reflexive: self-delegation



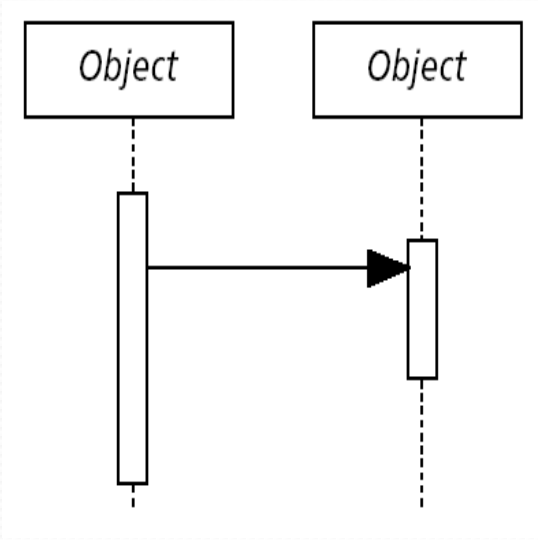


UML Sequence Diagram

- In procedural interactions, exactly one object is computing at any one time
- An object starts to compute when it receives a message: it has a **live activation**
- Sending a message **passes control** to the receiver of the message
- Messages can be **guarded** by a **condition**: only sent if the condition is satisfied
- Condition is shown in front of the message in square brackets

Focus of Control

- The relative time that the flow of control is focused in an object: i.e. the time an object is processing or directing messages
- Think of control as a token that gets passed as a message along the links and back again
- To keep track of the stack of objects and focus of control, messages are numbered



Conditions: Alternatives

- *Conditions* can be added to messages to show the situations when they are sent
- *Conditions* also distinguish between alternative message sequences
- Sequence diagrams provide the notation to show these alternative flows
- This is a complex notation: separate diagrams are clearer

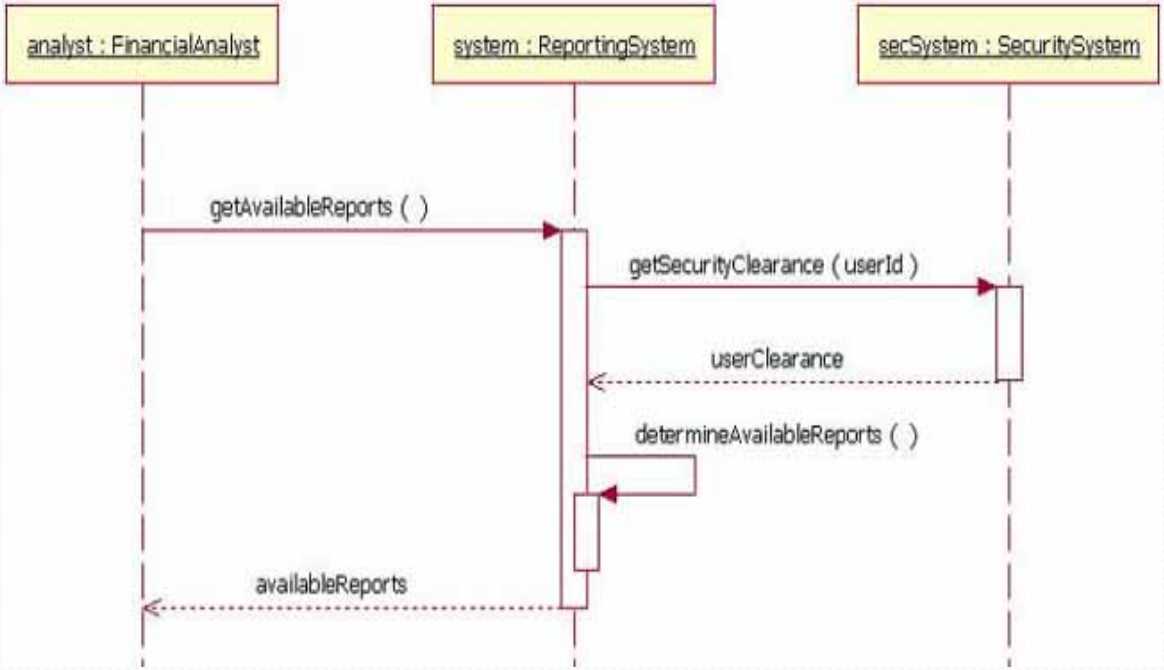
Messages / Operations

- The message signature consists of:
  - Preceding conditions
  - Message name
  - Optional argument list
  - Return class

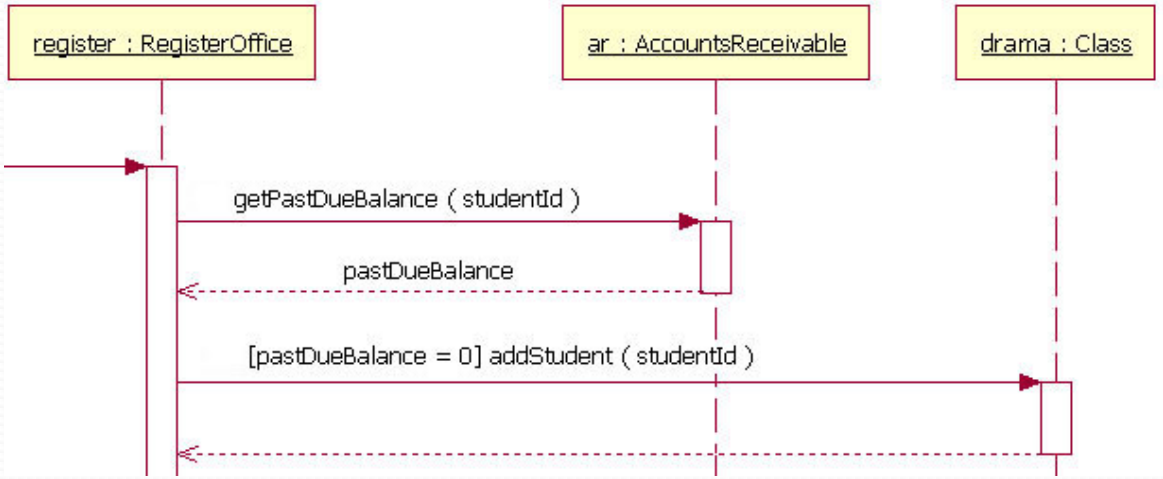
- Example:

  [Cond1] Course.getPrerequisite (): CourseList

Sequence Diagram Example

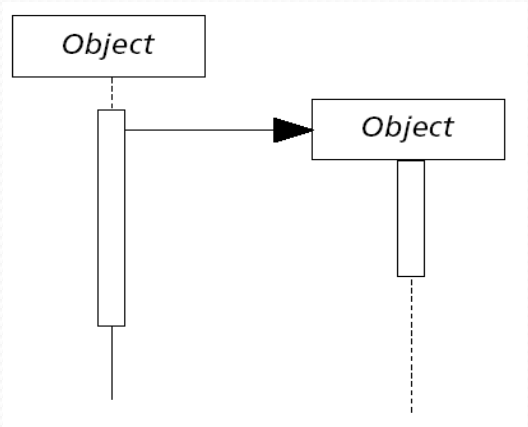


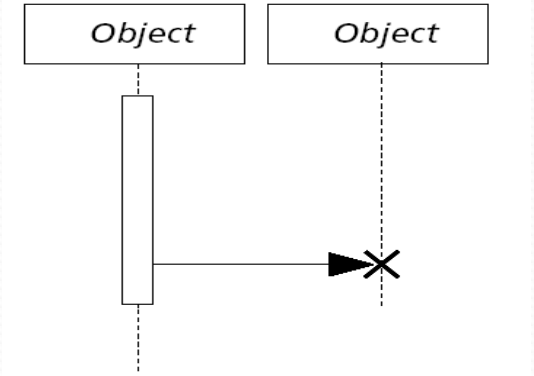
SD Example with Condition



Object Construction & Destruction

- Object creation is shown with the construction arrow (dashed) going to the new object symbol which is shown at the point of creation
- Object destruction is denoted by an X at the end of the object lifeline on receipt of the delete message

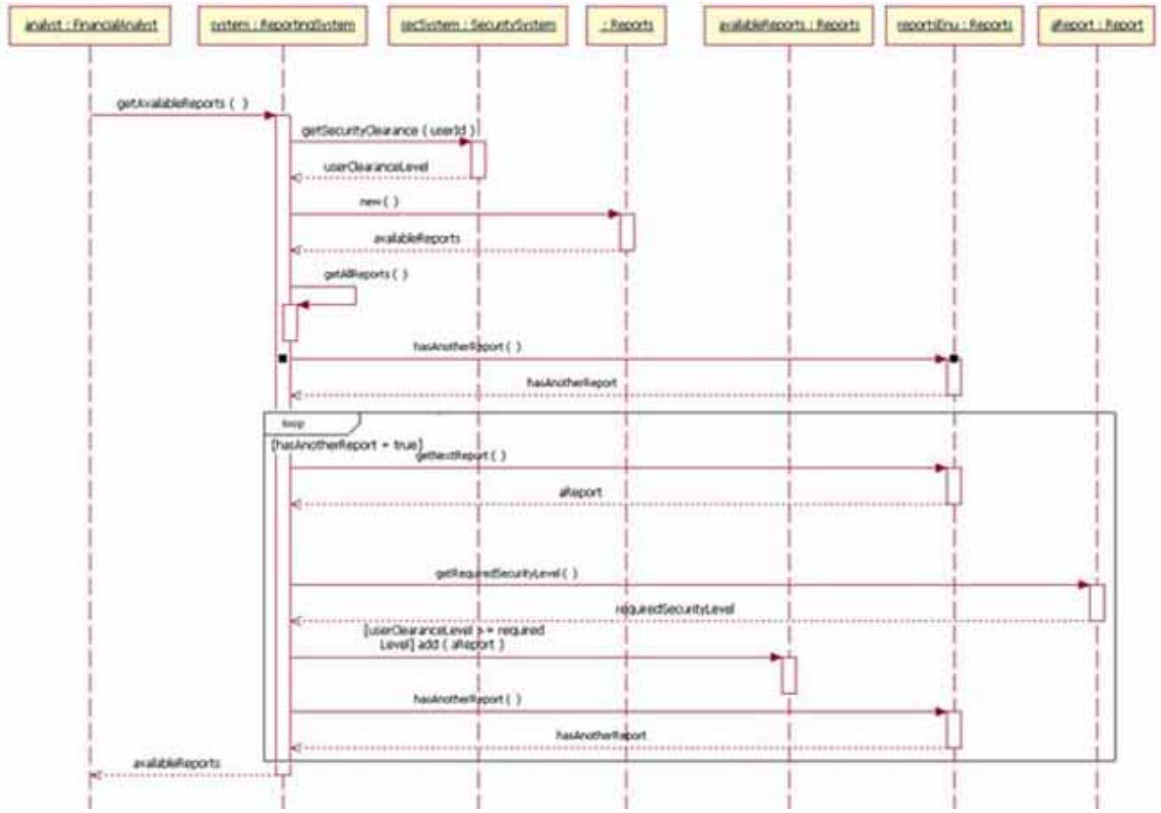




Iteration

- Iteration is shown by enclosing the repeated messages inside a frame with the heading *loop*
- This interaction is known as a ***combined fragment***
- The keyword loop is an example of an *interaction operator*
- The conditions for ending an iteration may be shown beside the frame's heading: *interaction constraints*

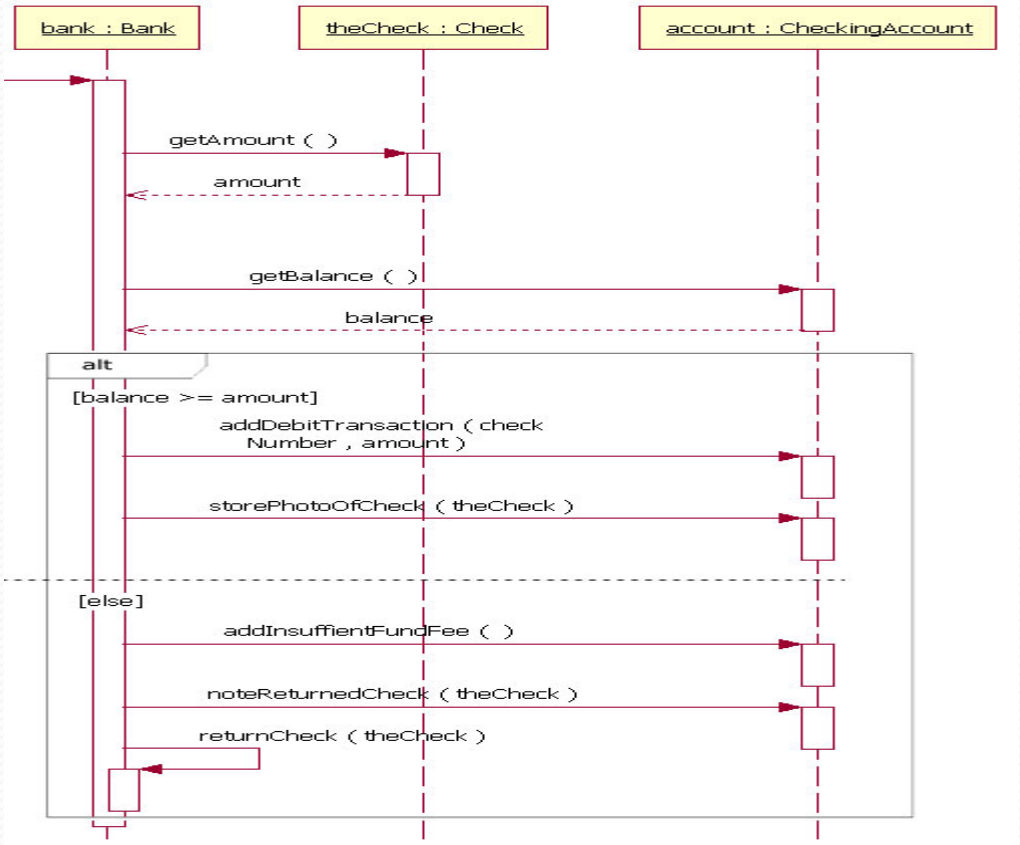
SD Example: Iteration



Branching

- Interactions considered so far have only one execution path
- Some interactions will have two or more alternative execution pathways
- This branching is shown in a ***combined fragment*** named with the keyword *alt*
- Branching constructs typically correspond to decision points in the use case

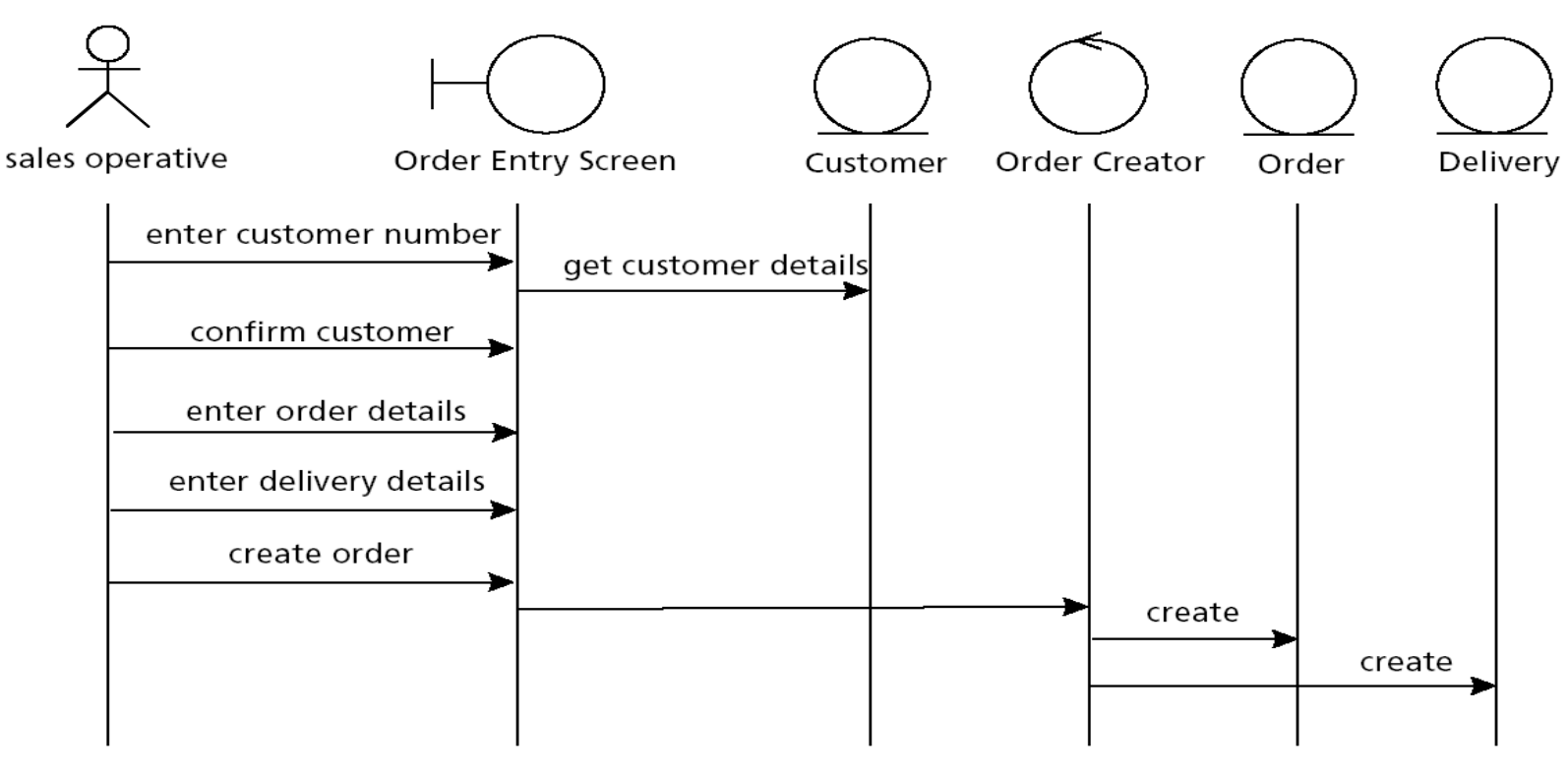
SD Example: Branching



Use Case Realisation

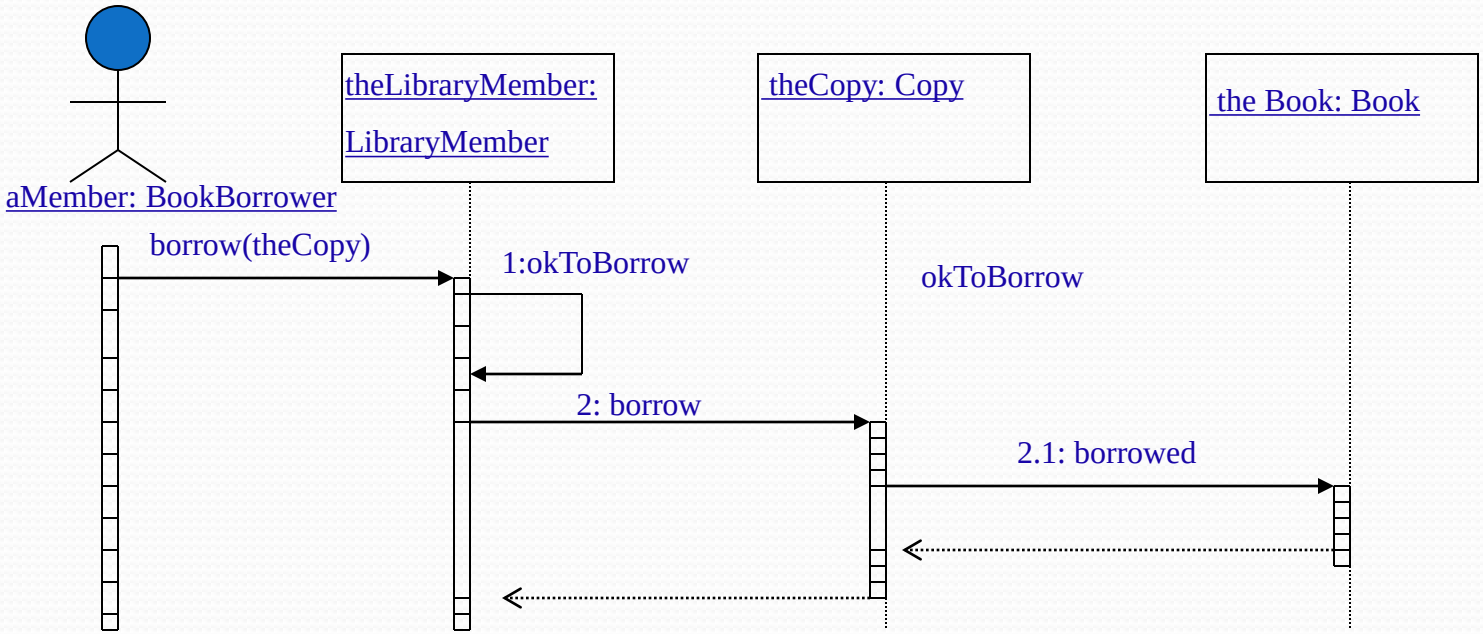
- Most use cases imply at least one *boundary* object that manages the dialogue between the actor and the system
- A *control* object manages the overall object communication during the use case
- In the next slide the diagram shows the actor, *sales operative*, interacting with the boundary object *orderEntryScreen*
- A later part of the interaction involves the control object *OrderCreator*

Sequence Diagram Example



UML SD: Library

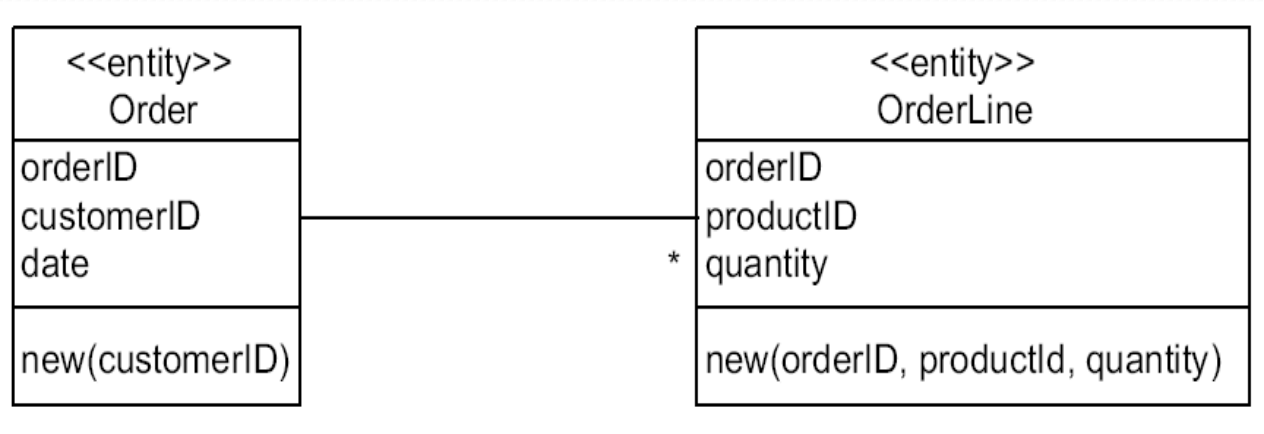
- UML uses interaction diagrams to show how messages pass between system objects to carry out some task



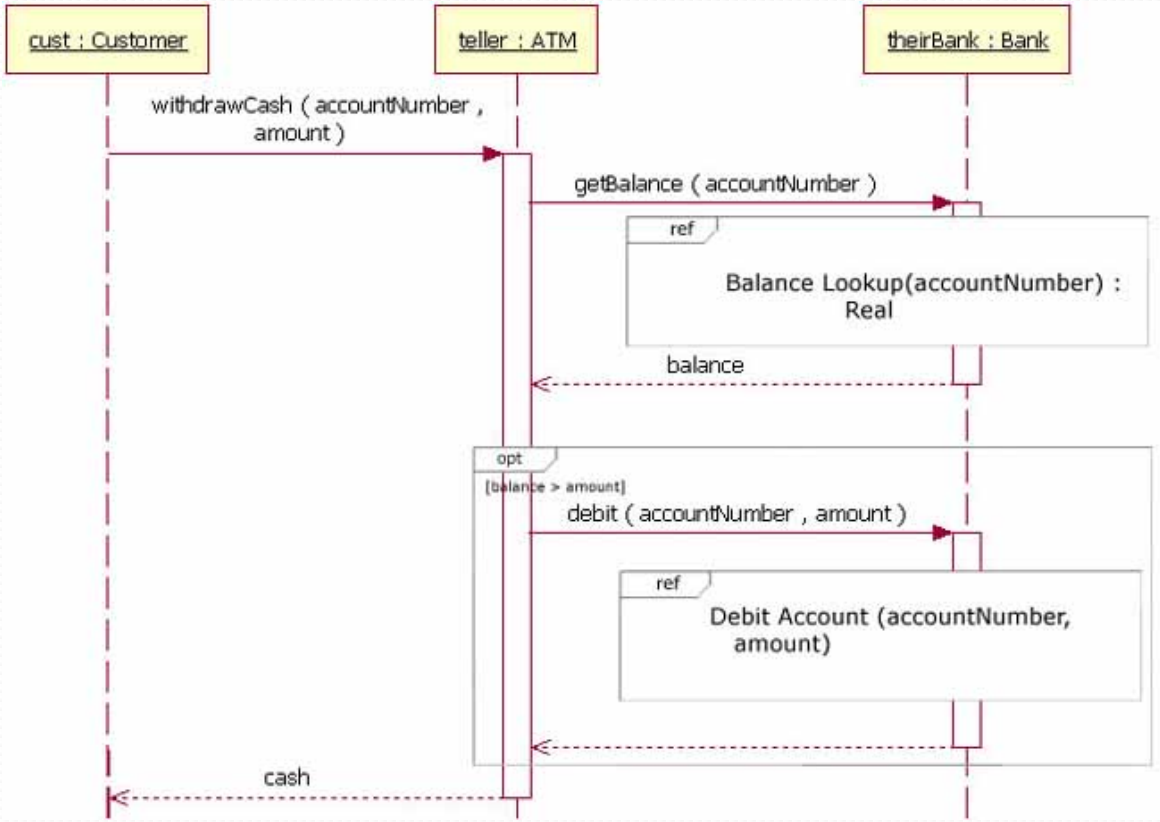
Order Example

- The following represents basic order processing:
  - User creates a new order: UI object sends a “new” message to an Order object
  - Order lines are created: Order object then sends “new” messages to each OrderLine on the Order
  - Stock levels are checked: Each Order Line object checks the given Stock Item object for quantity
    - If this check returns “true”, the Order Line removes the appropriate quantity of Stock Item from Stock
    - Otherwise, the quantity of Stock Item has fallen below the reorder level, and Stock Item requests a new delivery

Order Classes



Nesting SD's


cust : Customer
teller : ATM
theirBank : Bank
withdrawCash (accountNumber , amount)
getBalance (accountNumber)
ref
Balance Lookup(accountNumber) : Real
balance
opt
[balance > amount]
debit (accountNumber , amount)
ref
Debit Account (accountNumber, amount)
cash