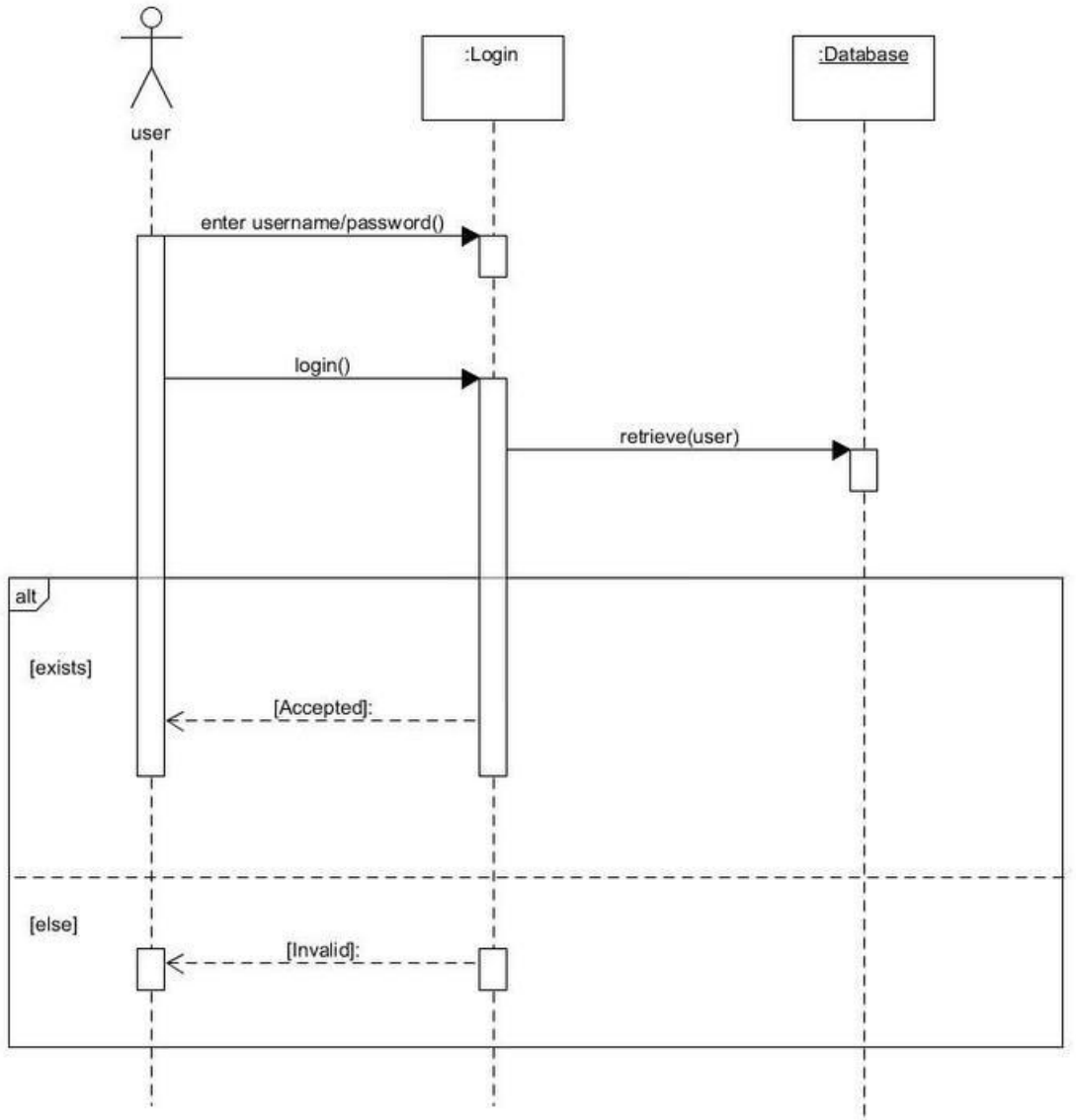
Sequence Diagrams



SWEN-261
Introduction to Software Engineering

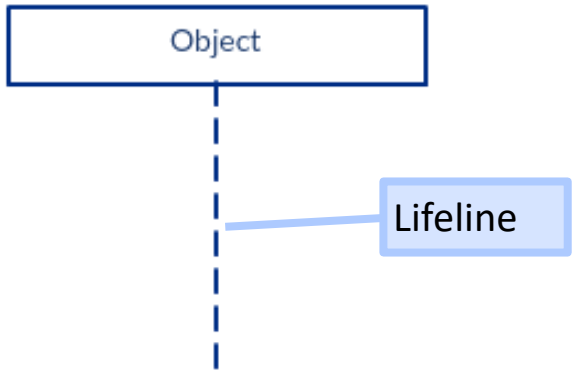
Department of Software Engineering
Rochester Institute of Technology

Software Engineering
Rochester Institute of Technology

The sequence diagram is a basic tool for modeling dynamic interactions between software entities.

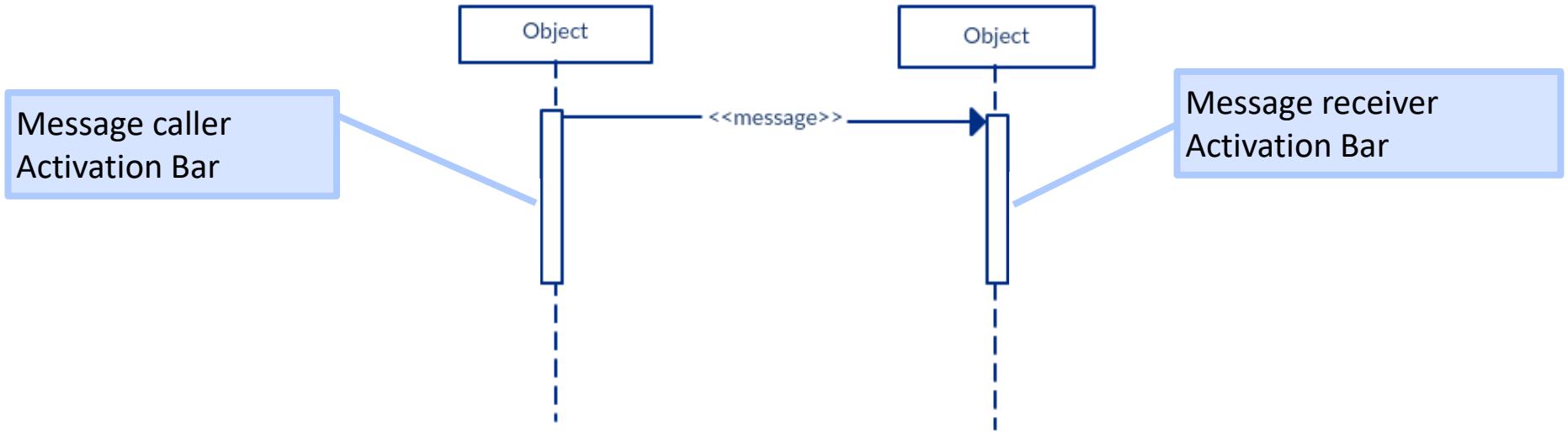
- They are used to model the flow of logic within your system
- Can be used at various levels of abstraction including:
  - ***Business workflow***
  - ***User story feature flow***
  - ***Object-level interactions*** ← We will look at this level.
- At any abstraction level, sequence diagrams captures the high-level information but not every detail
- The notation is simple to grasp
  - ***Time progresses top to bottom***
  - ***Operations generally flow left to right***
  - ***Show method calls with parameters***
  - ***Show return values when important***
  - ***Can show creation and deletion of objects***

Sequence Diagram Notations - Lifeline



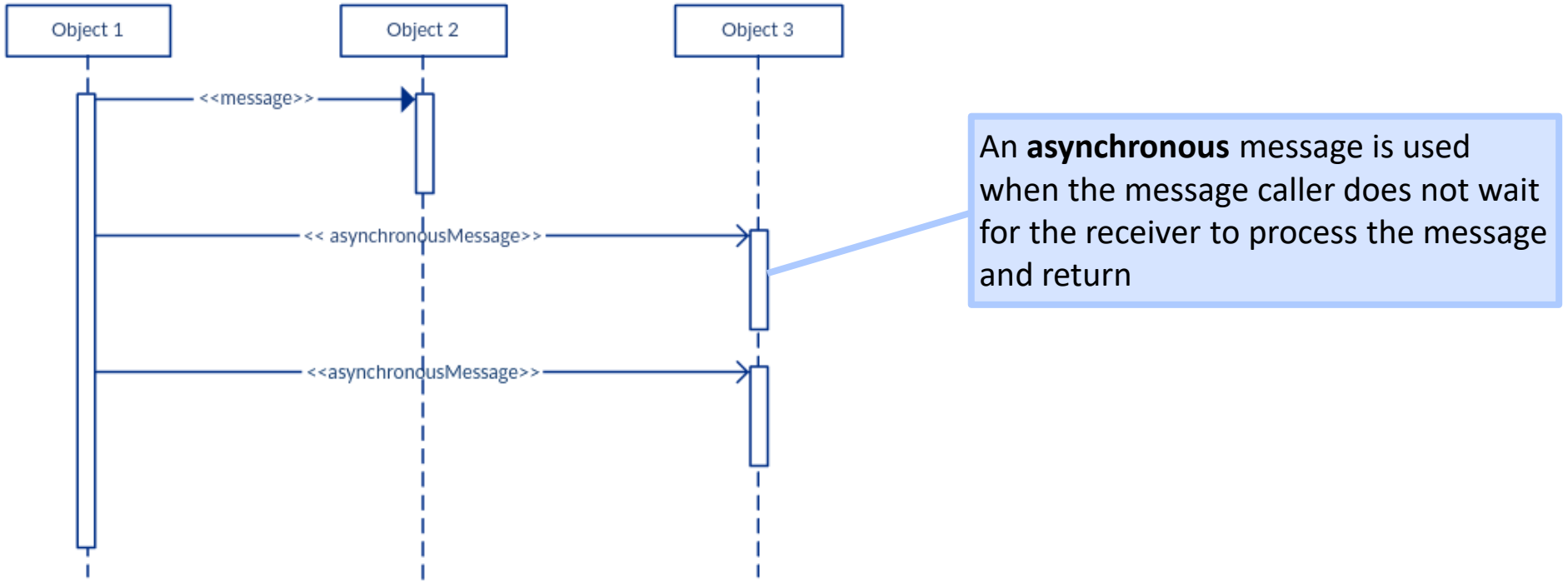
- A lifeline represents an individual participant in the Interaction.
- They represent the different objects or parts that interact with each other in the system during the sequence.
- A sequence diagram is made up of several of these lifeline notations that should be arranged horizontally across the top of the diagram
- No two lifeline notations should overlap each other

Sequence Diagram Notations – Activation Bars



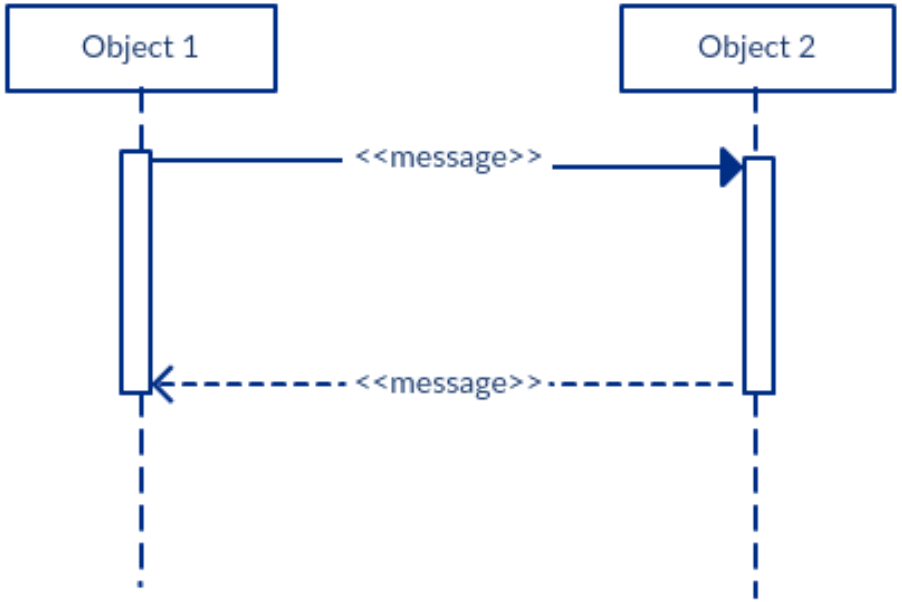
- Activation bar is the box placed on the lifeline.
  - ***It is used to indicate that an object is active (or instantiated) during an interaction between two objects.***
  - ***The length of the rectangle indicates the duration of the objects staying active***
- An interaction between two objects occurs when one object sends a message to another.

Sequence Diagram Notations – Message Arrows



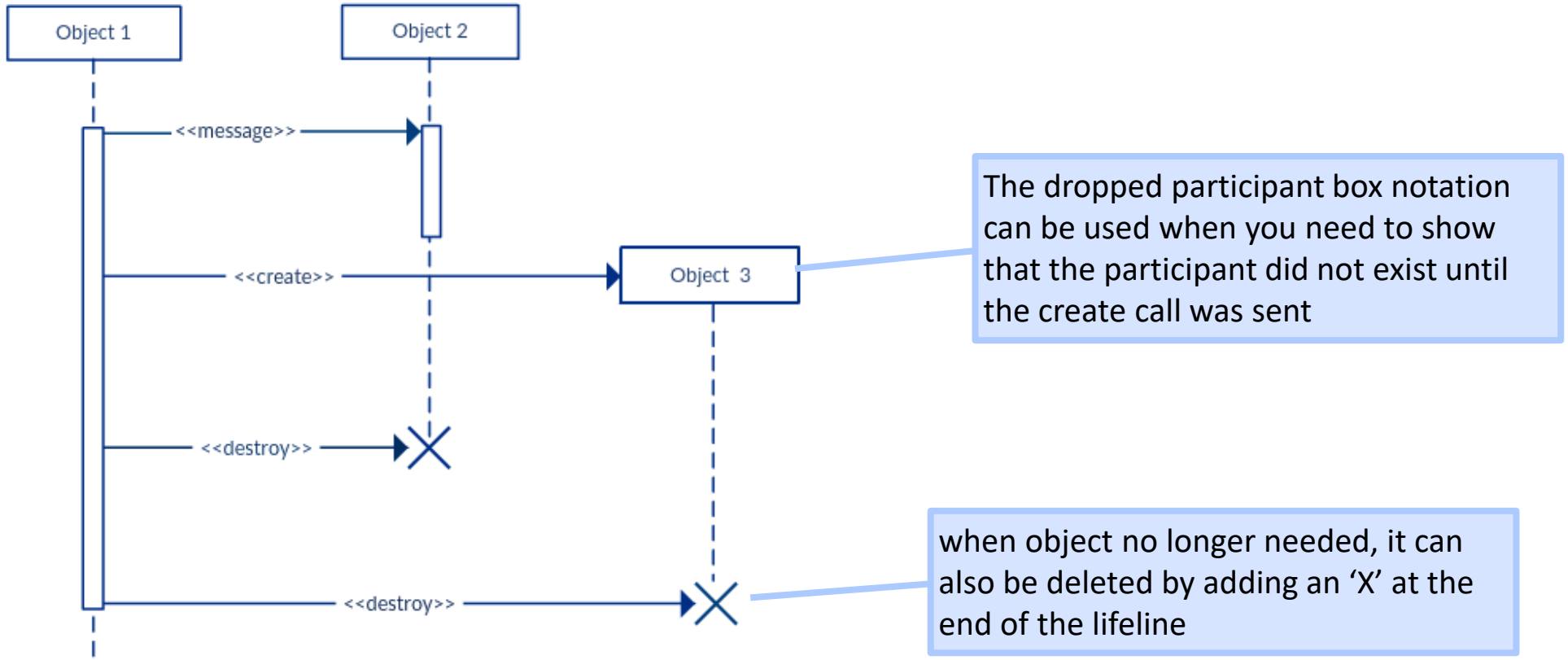
- A message can flow in any direction; from left to right, right to left or back to the Message Caller itself
  - ***While you can describe the message being sent from one object to the other on the arrow, with different arrowheads you can indicate the type of message being sent or received.***

Sequence Diagram Notations – Return message



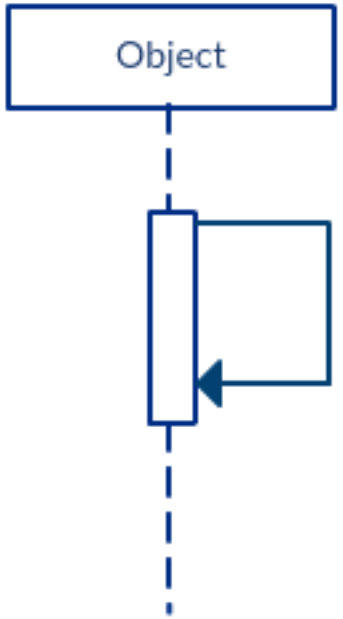
- A return message is used to indicate that the message receiver is done processing the message and is returning control over to the message caller
  - ***Return messages are optional notation pieces, for an activation bar that is triggered by a synchronous message always implies a return message.***

Sequence Diagram Notations – Object creation and deletion



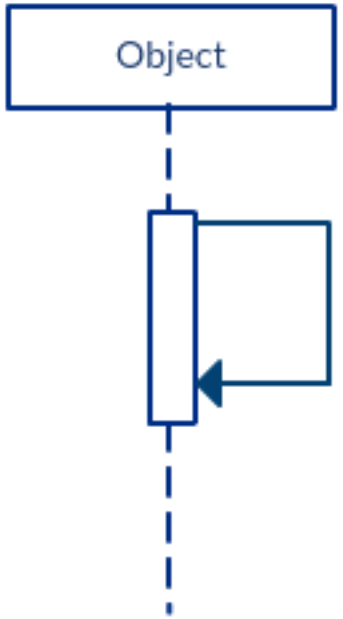
- Objects do not necessarily live for the entire duration of the sequence of events
  - ***Objects or participants can be created according to the message that is being sent.***

Sequence Diagram Notations – Self message



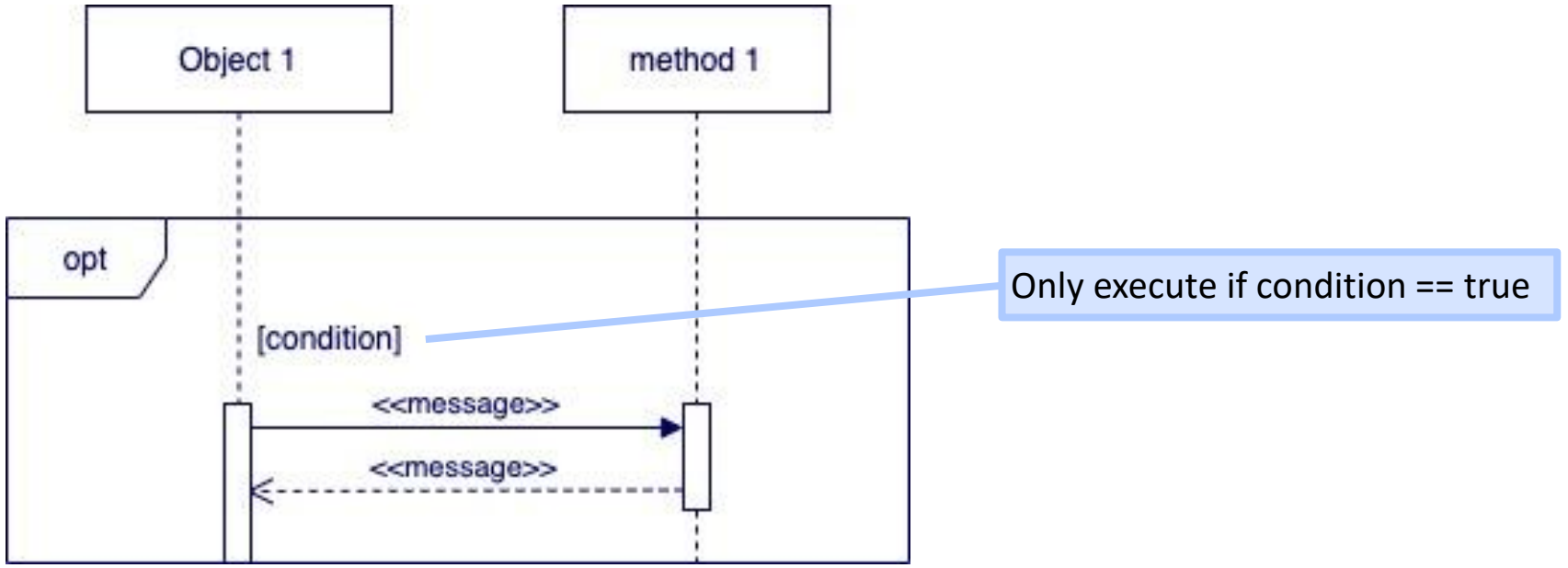
- A self message is a message that an object sends to itself.
  - ***It is a message that represents the invocation of message of the same lifeline.***
  - ***A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object.***

Sequence Diagram Notations – Self message



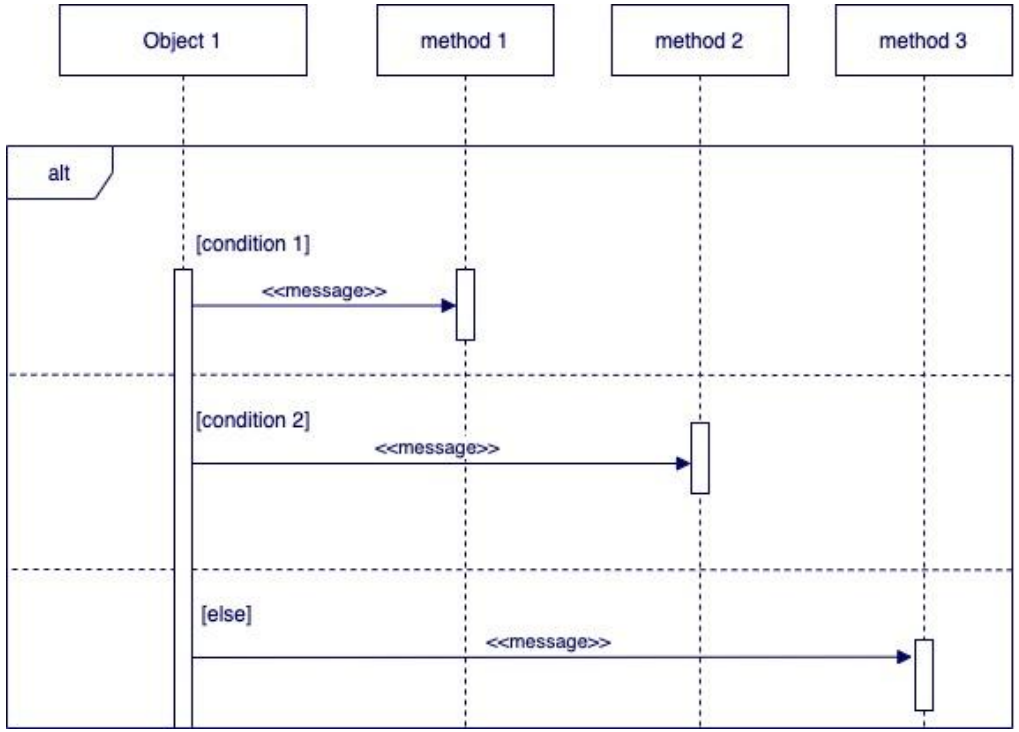
- A self message is a message that an object sends to itself.
  - ***It is a message that represents the invocation of message of the same lifeline.***
  - ***A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object.***

Sequence Diagram Notations – option fragment



- The option (opt) fragment is used to indicate a sequence that will only occur under a certain condition, otherwise, the sequence won't occur.
  - ***It models the "if then" statement.***

Sequence Diagram Notations – alternative fragment



- The alternative (alt) fragment is used when a choice needs to be made between two or more message sequences.
  - ***It models the "if then else" logic.***
- There can be as many alternative paths as are needed
  - ***If more alternatives are needed, all you must do is add an operand to the rectangle with that sequence's guard and messages.***

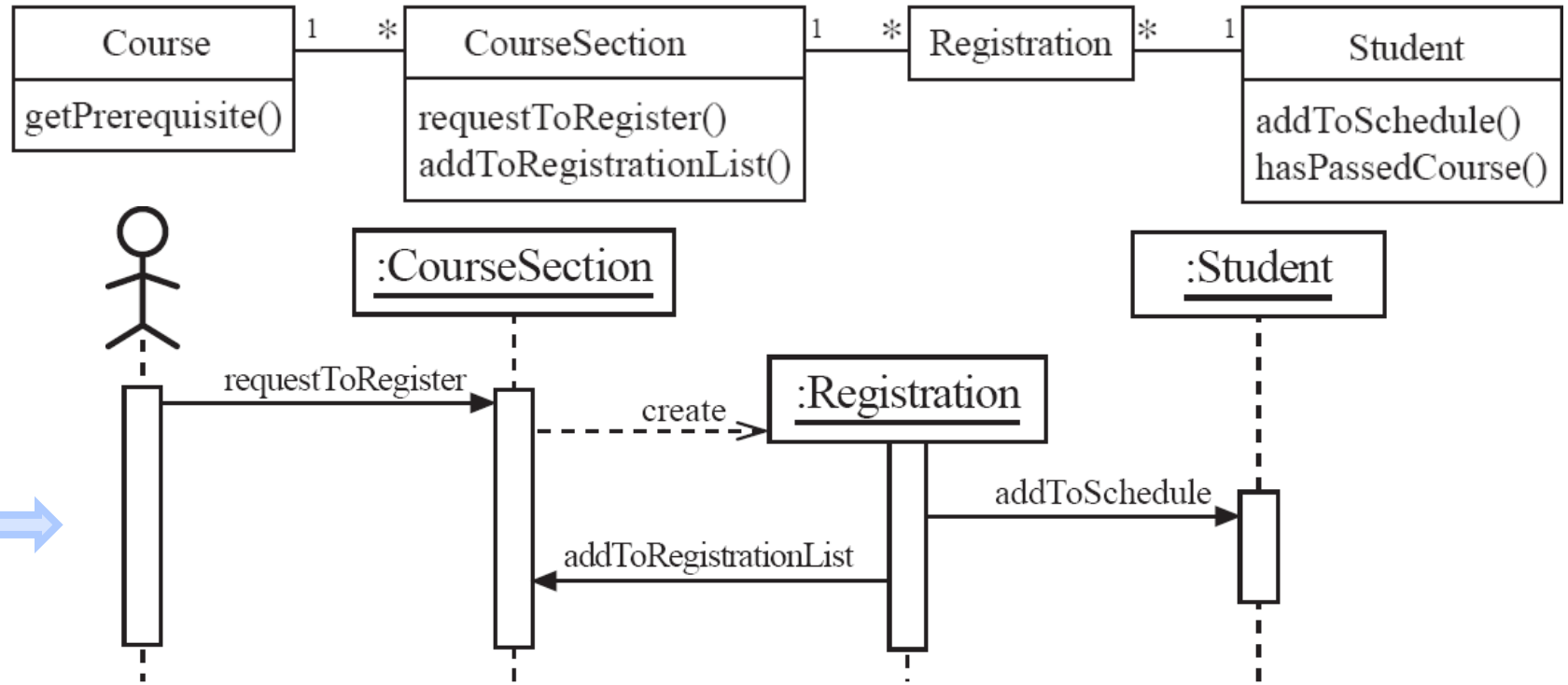
Manage complex interactions with sequence fragments.

- The type of fragment is determined by the fragment operator.
- You can use fragments to describe several control and logic structures in a compact and concise manner.

Fragment Operator	Description
opt	Defines condition to a single call - the call will execute only if the supplied condition is true. Equivalent to an alt with only one trace.
alt	Divides fragment into groups and defines condition for each group - only the one whose condition is true will execute
par	Defines that the calls within the fragment run in parallel.
loop	Defines that the calls within the fragment run in a loop.
region	Defines that the calls within the fragment reside in a critical section, i.e. the fragment can have only one thread executing it at once.

Register for Courses

Static feeds to => Dynamic model: SEQUENCE DIAGRAM UML 1.0



2) Dynamic Model

(aka Sequence Diagram)

1) Static Model

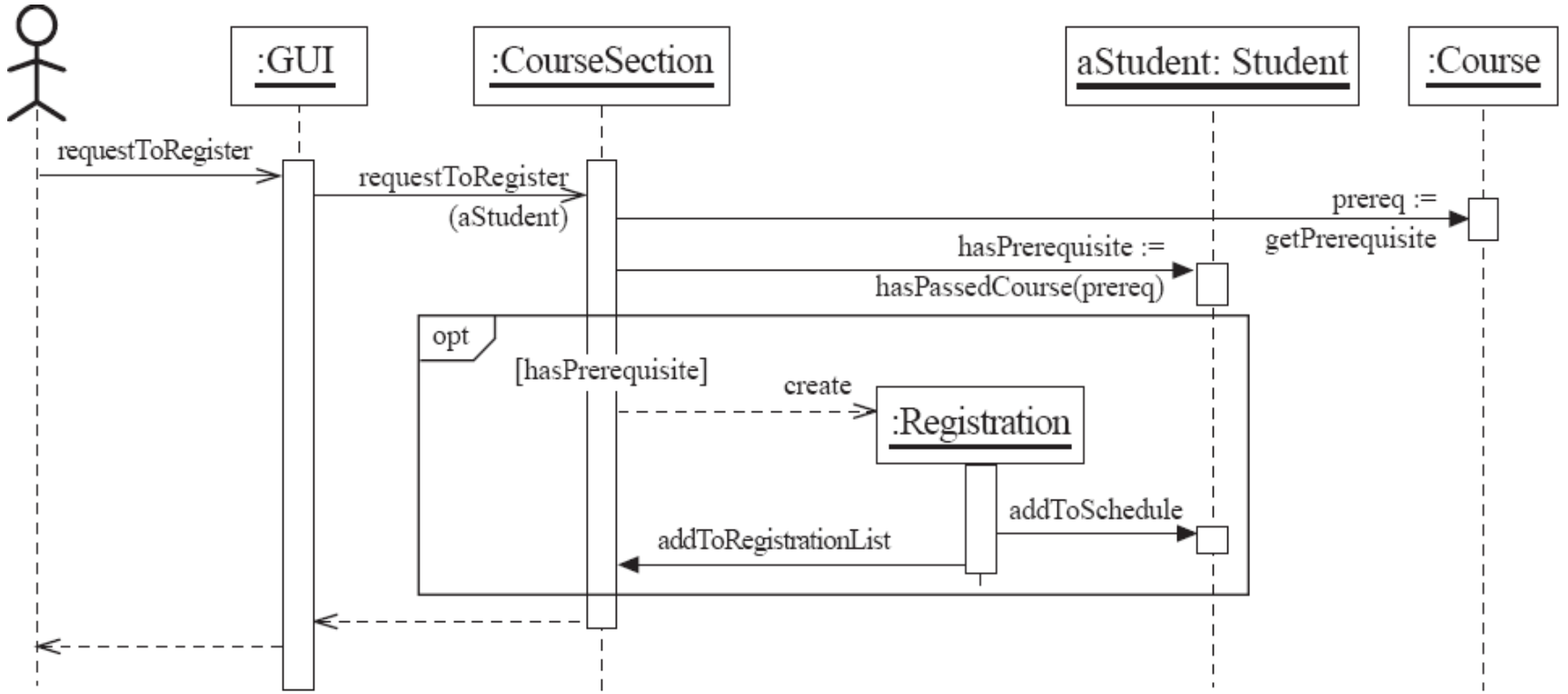
(aka Class Diagram)

Register for Courses

Same example more detailed SEQUENCE DIAGRAM UML 1.0

3) Dynamic Model

(refined)



These are the basic notations for sequence diagrams that you can use. Our notation ≈UML 2.0