EAST WEST UNIVERSITY

CSE347

Information System Analysis and Design

Nishat Tasnim Niloy

Lecturer

Department of Computer Science and Engineering

Faculty of Science and Engineering

Topic: 9

Sequence Diagram

UML sequence diagrams

- **sequence diagram**: an "interaction diagram" that models a single scenario executing in the system
  - perhaps 2nd most used UML diagram (behind class diagram)
- Often used to model the way a use case is realized through a sequence of messages between objects.
- relation of UML diagrams to other exercises:
  - CRC cards -> class diagram
  - use cases -> sequence diagrams

Purpose of a Sequence Diagram

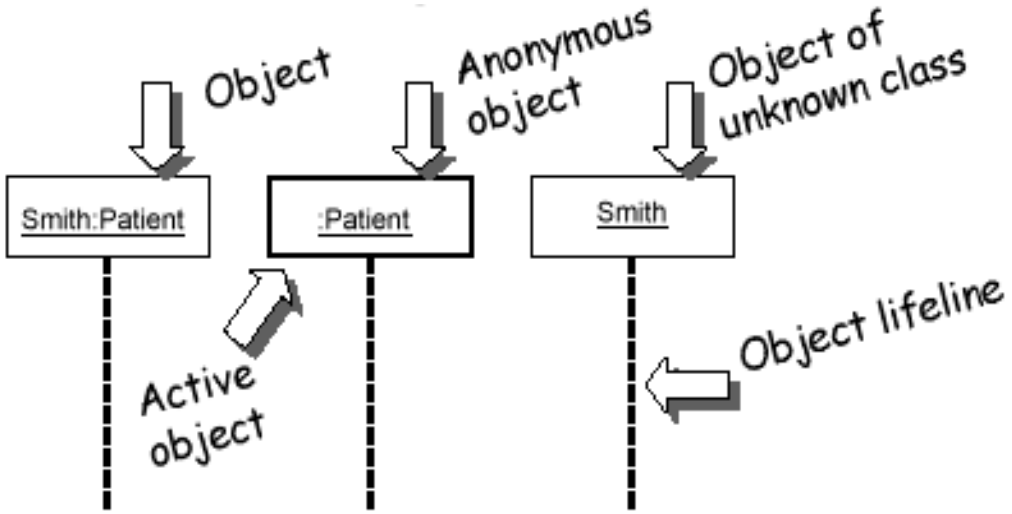
- Model interactions between objects
- Assist in understanding how a system (a use case) actually works
- Verify that a use case description can be supported by the existing classes
- Identify responsibilities/operations and assign them to classes

Key parts of a sequence diagram

- **Participant**: An object or entity that acts in the sequence diagram
  - Sequence diagram starts with an unattached "found message" arrow
- **Message**: communication between participant objects
- The axes in a sequence diagram:
  - horizontal: which object/participant is acting
  - vertical: time (down -> forward in time)

Representing objects

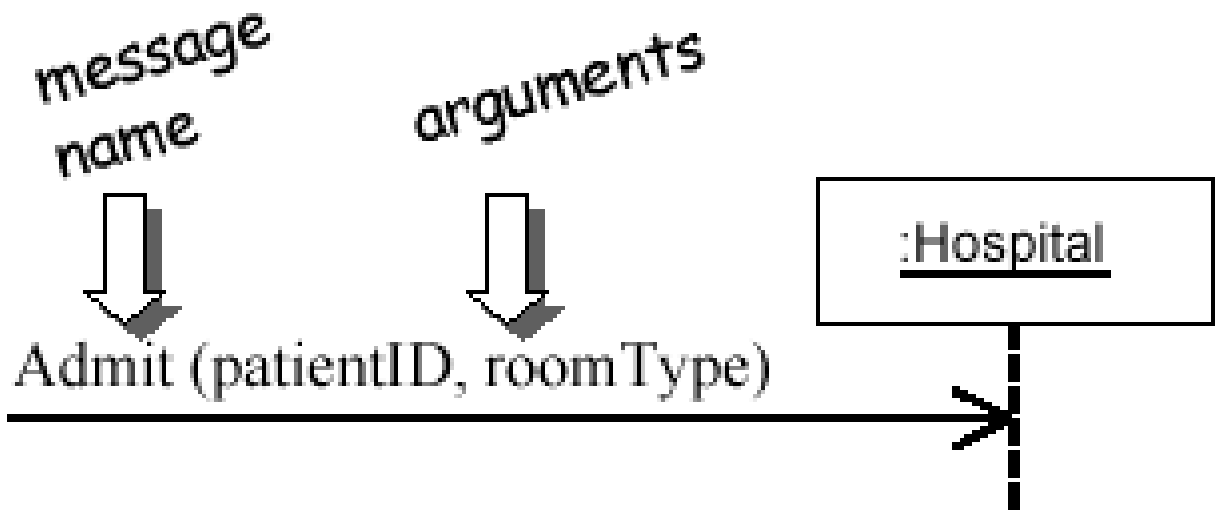
- Squares with object type, optionally preceded by object name and colon
  - write object's name if it clarifies the diagram
  - object's "life line" represented by dashed vertical line



Name syntax: <objectname>:<classname>

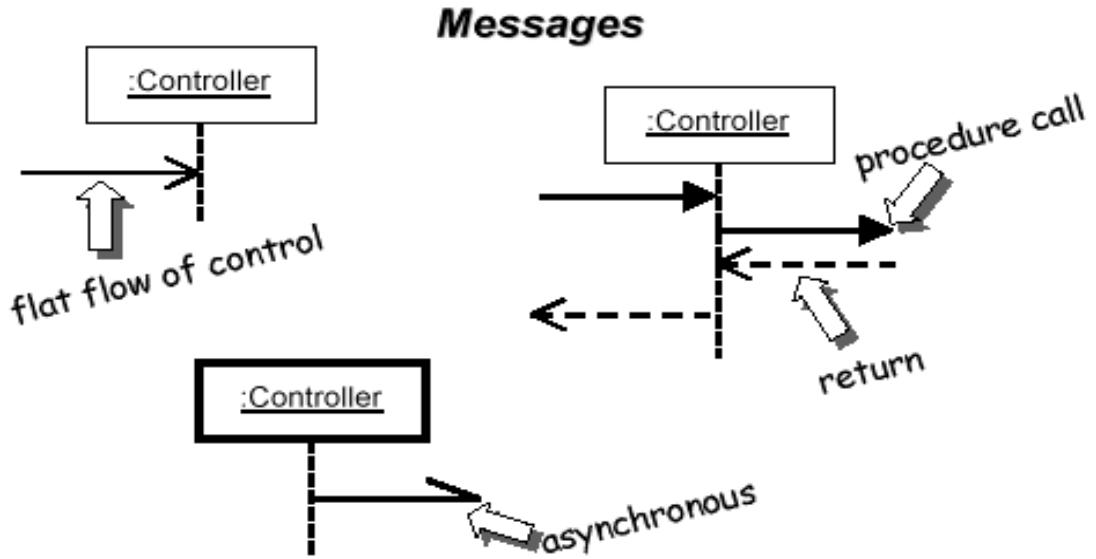
Messages between objects

- message (method call) indicated by horizontal arrow to other object
  - write message name and arguments above arrow

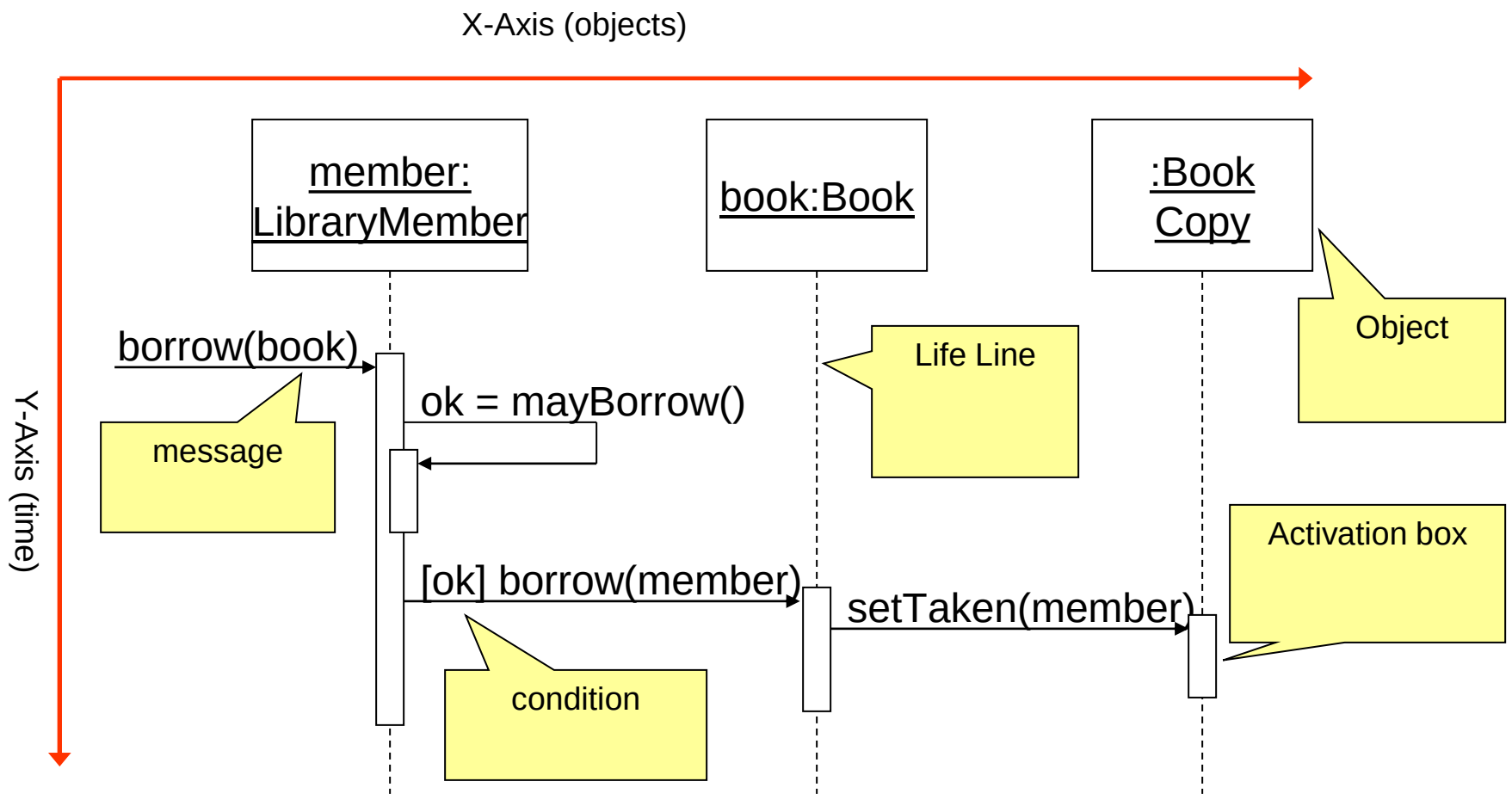


Messages (continued)

- message (method call) indicated by horizontal arrow to other object
  - dashed arrow back indicates return
  - different arrowheads for normal / concurrent (asynchronous) methods



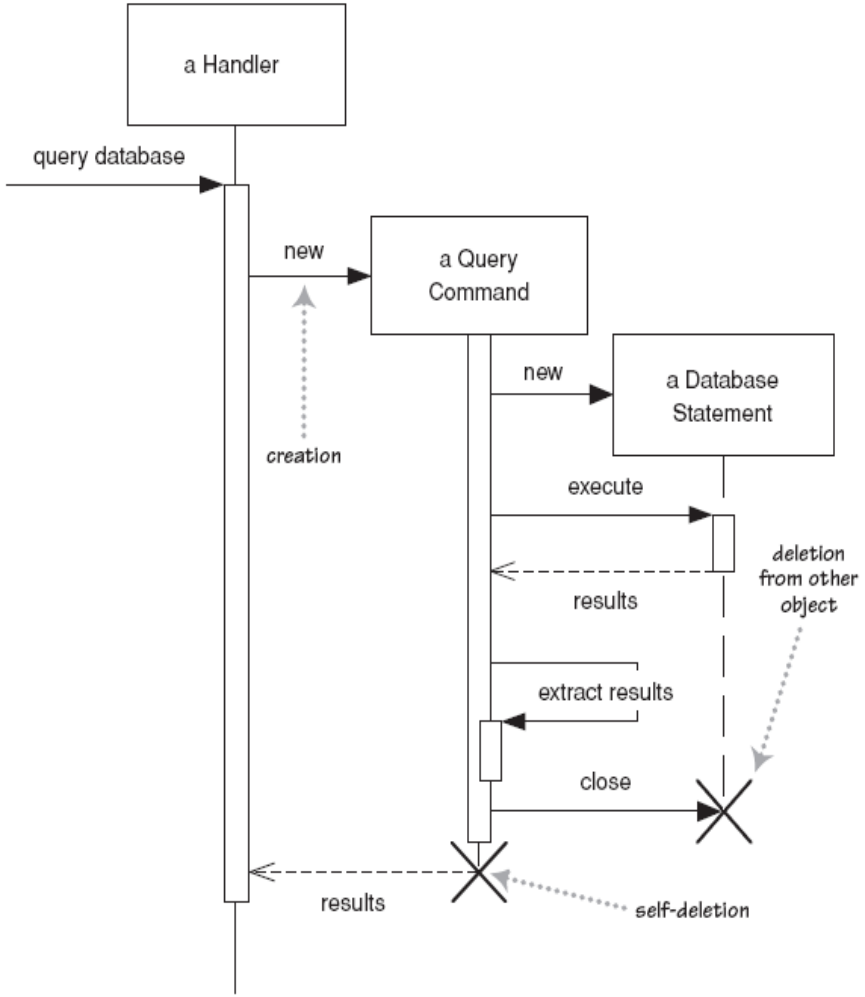
Sequence diagram from use case



Lifetime of objects

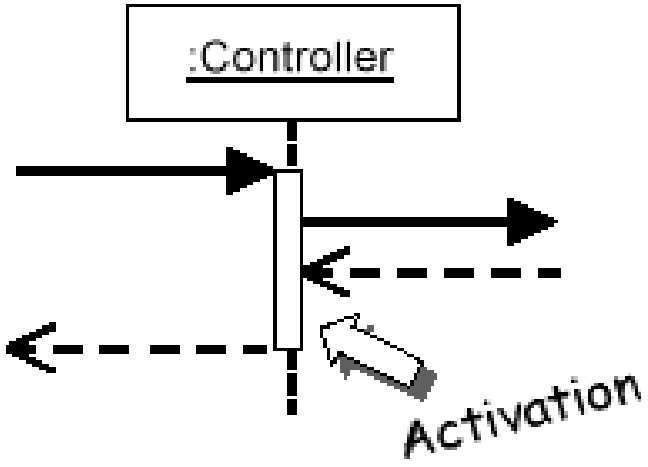
- *creation*: arrow with 'new' written above it
  - notice that an object created after the start of the scenario appears lower than the others

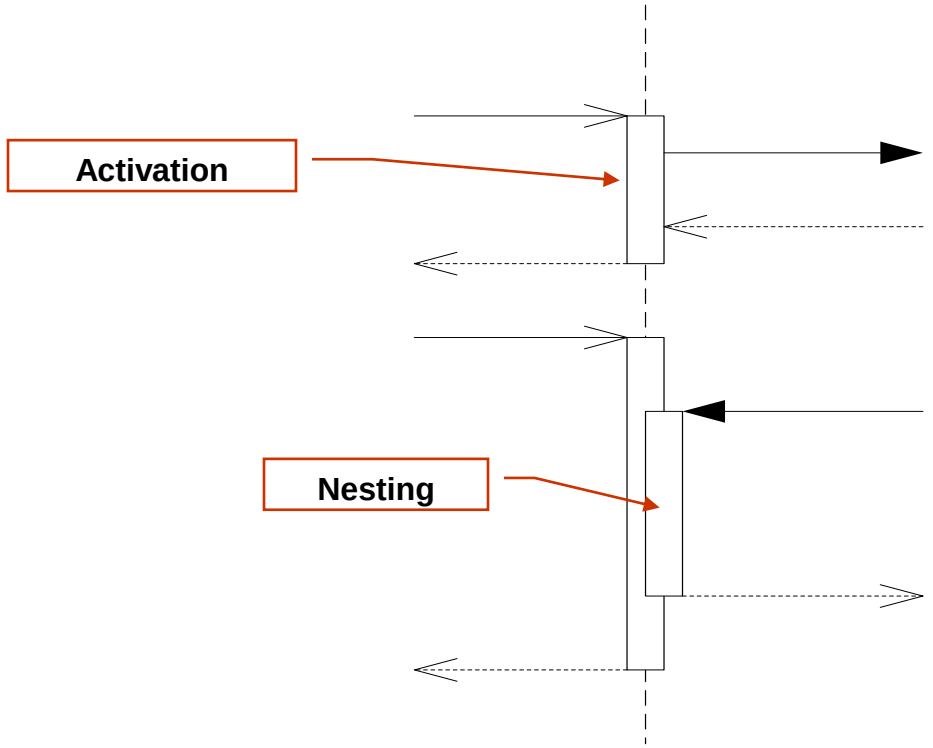
- *deletion*: an X at bottom of object's lifeline
  - Java doesn't explicitly delete objects; they fall out of scope and are garbage-collected



Indicating method calls

- **activation**: thick box over object's life line; drawn when object's method is on the stack
  - either that object is running its code, or it is on the stack waiting for another object's method to finish
  - nest to indicate recursion

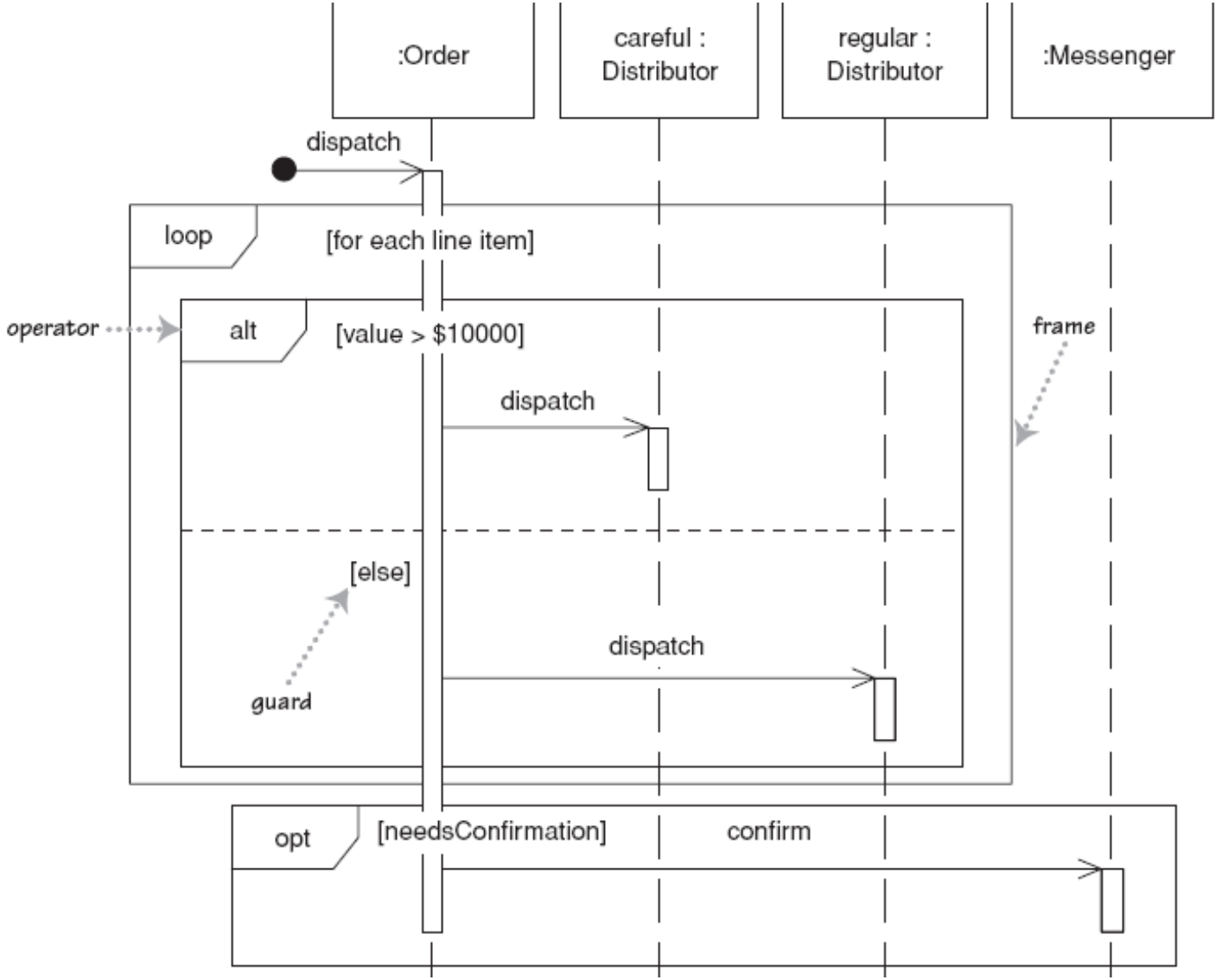




Indicating selection and loops

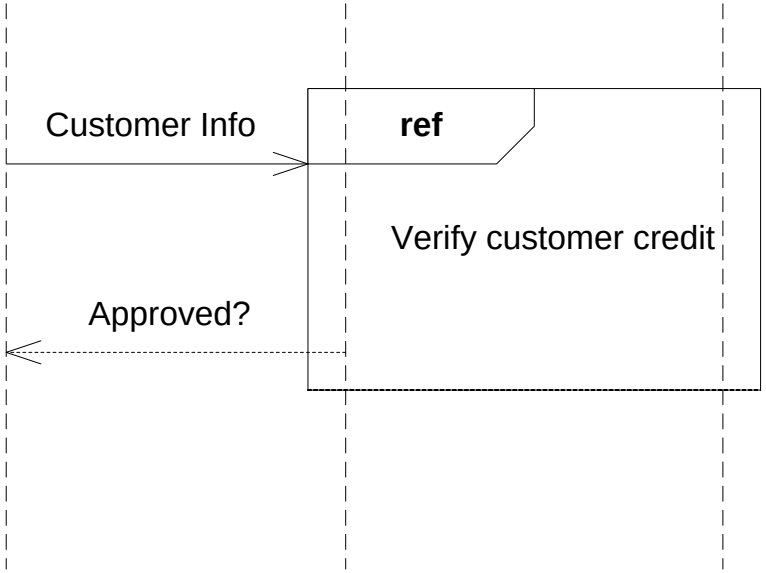
frame: box around part of a sequence diagram to indicate selection or loop

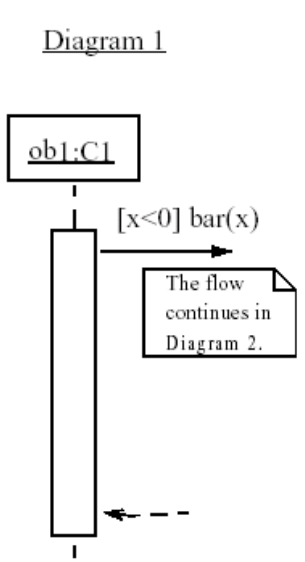
- `if` -> (opt) [condition]
- `if/else` -> (alt) [condition], separated by horizontal dashed line
- loop -> (loop) [condition or items to loop over]

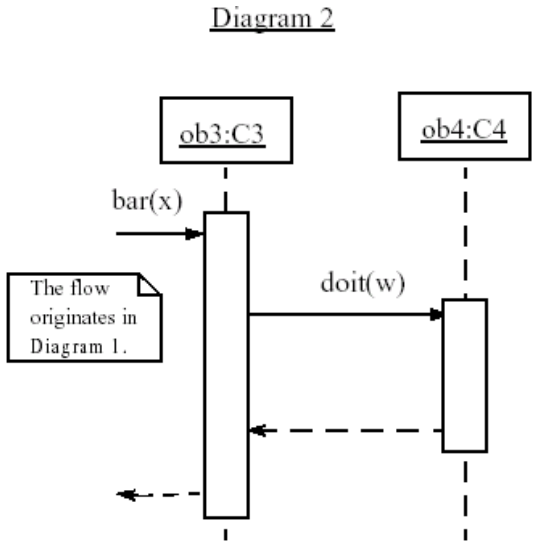


linking sequence diagrams

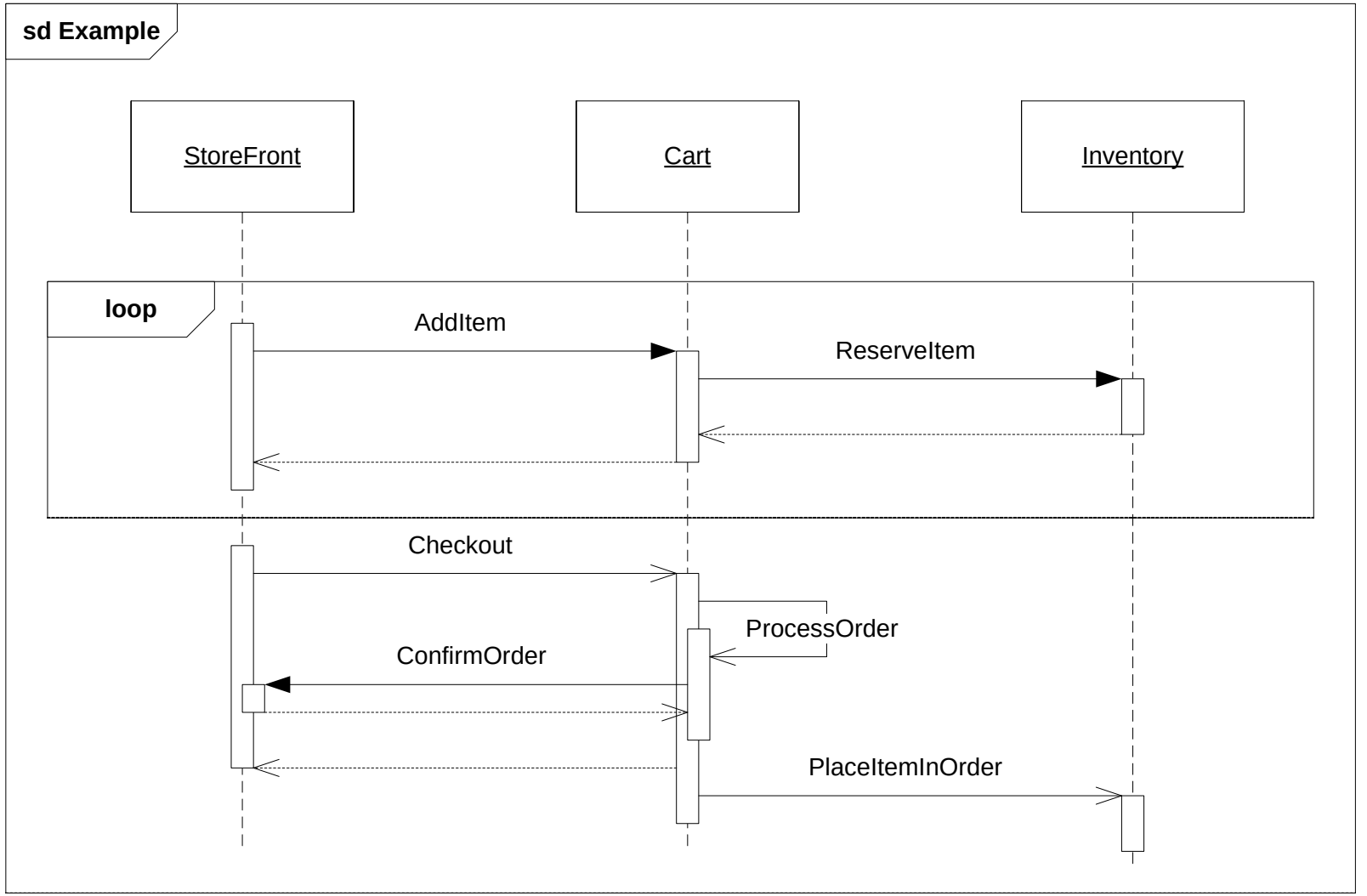
- if one sequence diagram is too large or refers to another diagram, indicate it with either:
  - an unfinished arrow and comment
  - a "ref" frame that names the other diagram
  - when would this occur in our system?





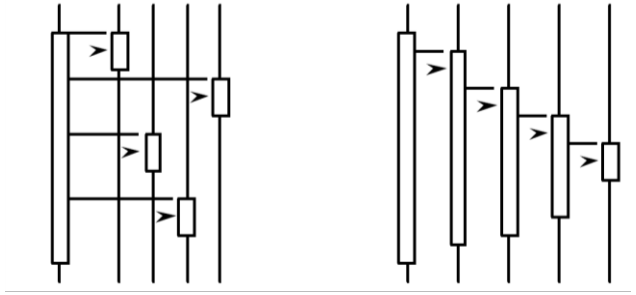


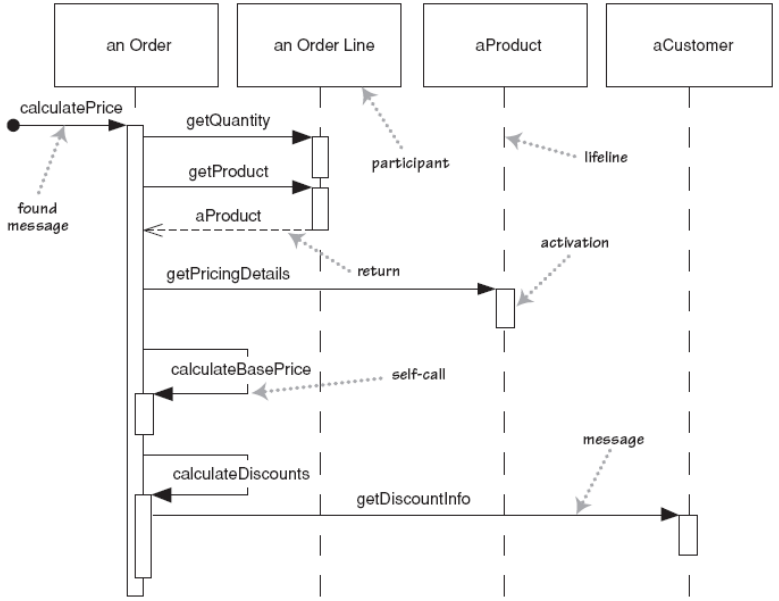
Example sequence diagram

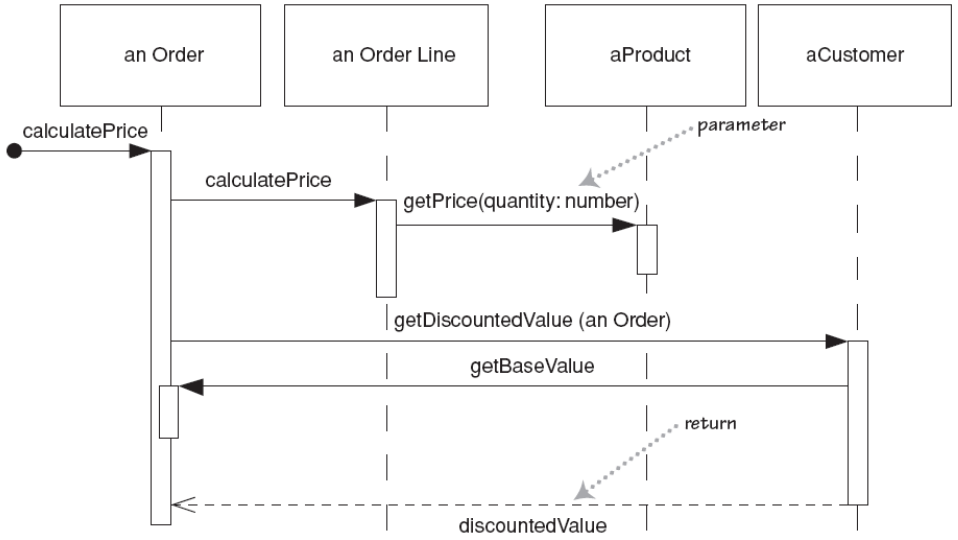


Forms of system control

- What can you say about the control flow of each of the following systems?
  - Is it centralized?
  - Is it distributed?



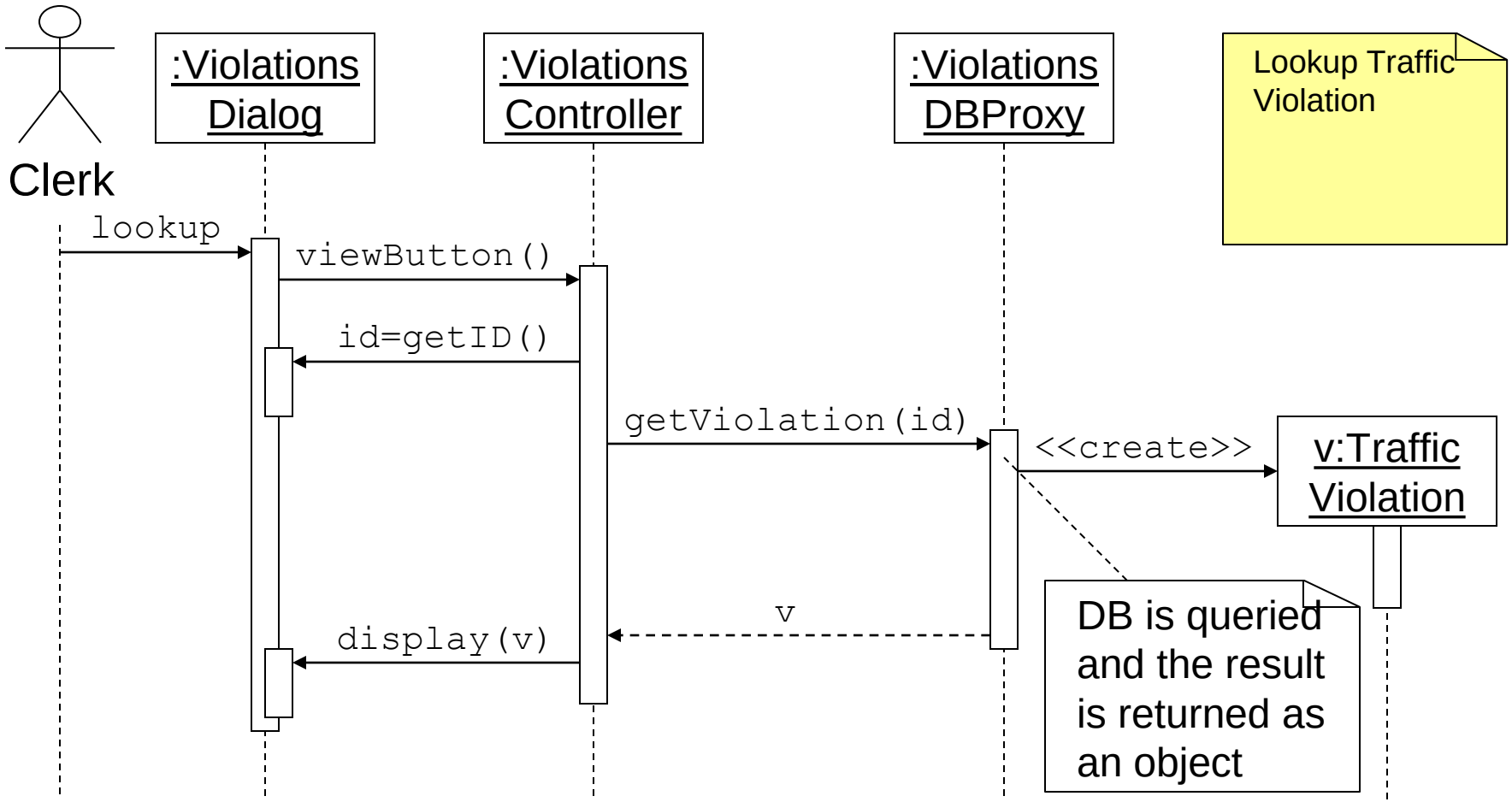




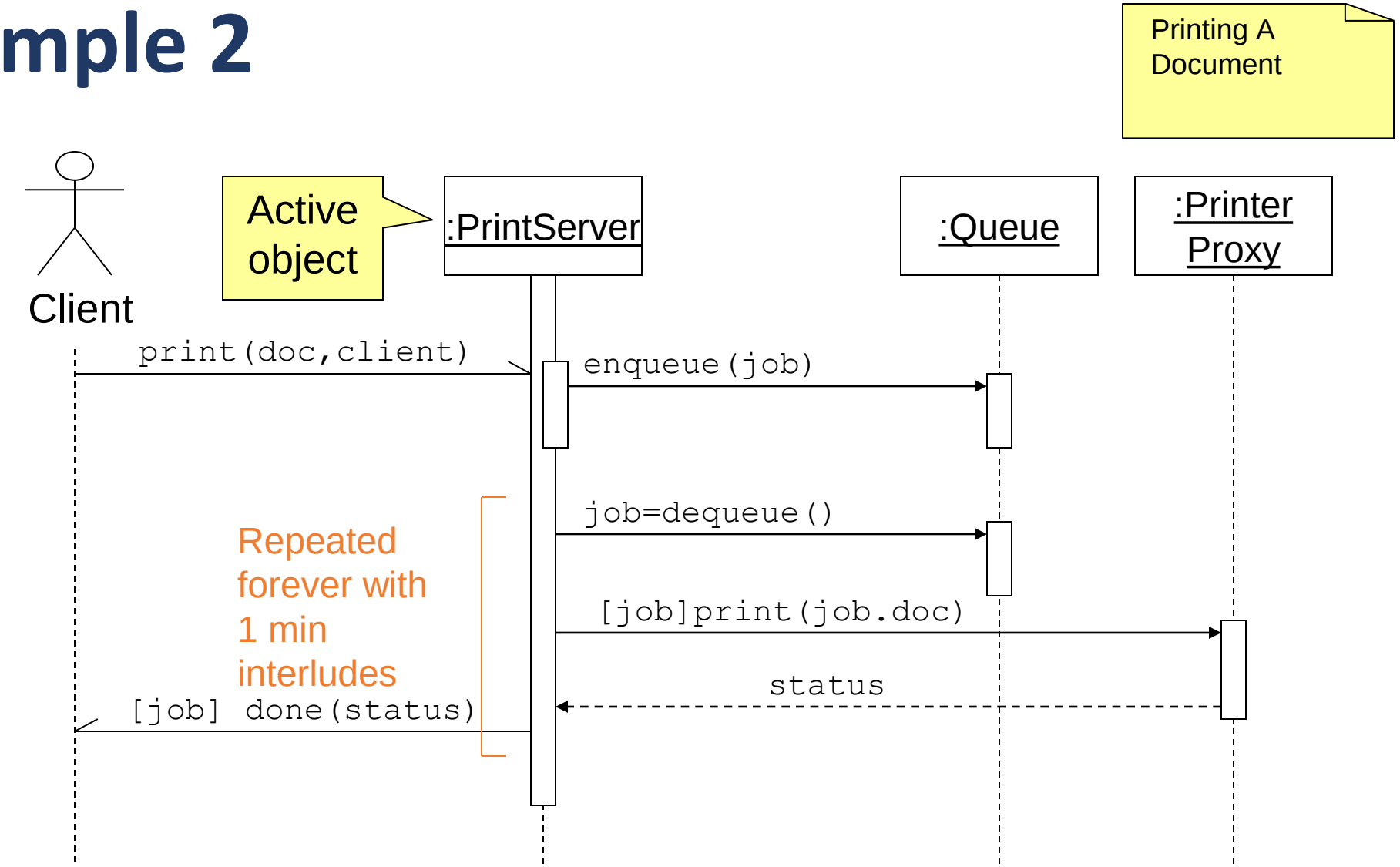
Why not just code it?

- Sequence diagrams can be somewhat close to the code level. So why not just code up that algorithm rather than drawing it as a sequence diagram?
  - a good sequence diagram is still a bit above the level of the real code (not all code is drawn on diagram)
  - sequence diagrams are language-agnostic (can be implemented in many different languages
  - non-coders can do sequence diagrams
  - easier to do sequence diagrams as a team
  - can see many objects/classes at a time on same page (visual bandwidth)

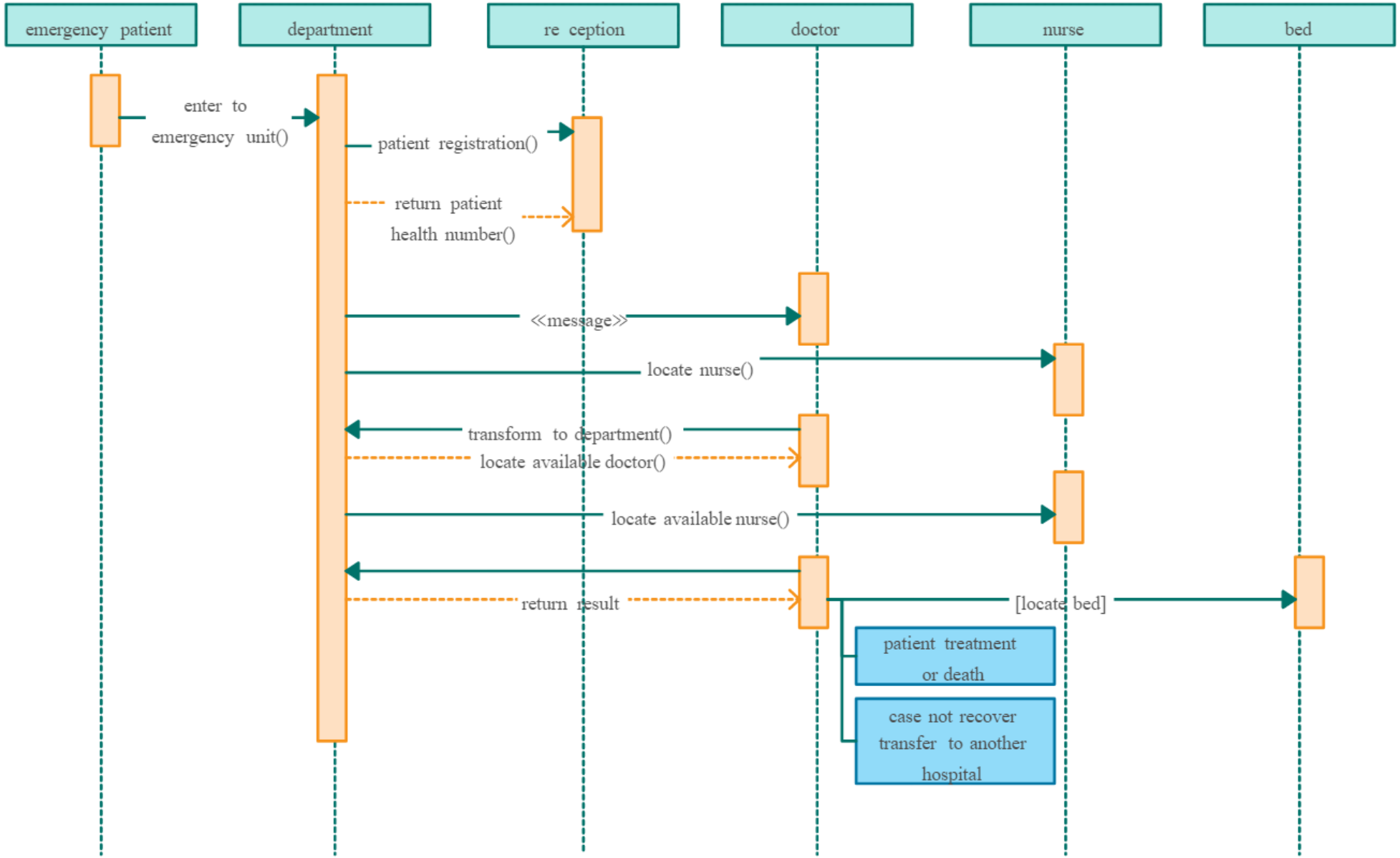
Example 1


Clerk
:Violations Dialog
:Violations Controller
:Violations DBProxy
Lookup Traffic Violation
lookup
viewButton()
id=getID()
getViolation(id)
<<create>>
v:Traffic Violation
v
display(v)
DB is queried and the result is returned as an object

Example 2


Printing A Document
Client
Active object
:PrintServer
:Queue
:Printer Proxy
print(doc,client)
enqueue(job)
job=dequeue()
Repeated forever with 1 min interludes
[job]print(job.doc)
status
[job] done(status)

Clinic Management System: SD



Sequence diagram exercise 2

- Let's do a sequence diagram for the following casual use case, *Add Calendar Appointment* :

  The scenario begins when the user chooses to add a new appointment in the UI. The UI notices which part of the calendar is active and pops up an Add Appointment window for that date and time.

  The user enters the necessary information about the appointment's name, location, start and end times. The UI will prevent the user from entering an appointment that has invalid information, such as an empty name or negative duration. The calendar records the new appointment in the user's list of appointments. Any reminder selected by the user is added to the list of reminders.

  If the user already has an appointment at that time, the user is shown a warning message and asked to choose an available time or replace the previous appointment. If the user enters an appointment with the same name and duration as an existing group meeting, the calendar asks the user whether he/she intended to join that group meeting instead. If so, the user is added to that group meeting's list of participants.