Sequence Diagrams and introduction to UML

COMP 3831

Larman: Chapter 10

The UML Diagrams

1. Class
2. Sequence
3. Activity
4. State
5. Object
6. Use Case
7. Collaboration
8. Component
9. Deployment

Sequence Diagrams in the Elaboration Phase

- Sequence diagrams are used in elaboration phase after decision is made to continue with serious investigation.

During Elaboration - Iteration 1

- Tackle a simple success scenario.
- Target design and implementation that touches on many major architectural elements of the new system.
- Skip tasks related to environment (e.g. tools, people, processes, and setting).
- Clarify input and output system events related to system with UML sequence diagrams.

System Sequence Diagram (SSD)

- A fast and easily created artifact that illustrates input and output events related to the system under discussion.
- A picture that shows, for a particular scenario of a use case, the following:
  - Events that external actors generate
  - Their order
  - Inter-system events

System Sequence Diagram (SSD)

- An SSD should be done for:
  - Main success scenarios
  - Frequent or complex alternative scenarios
- An SSD should take few minutes to ½ hour

From use cases to sequence diagrams

- Use cases describe how external actors interact with the software system
  - Actor generates events to a system
  - Event initiates an operation upon the system

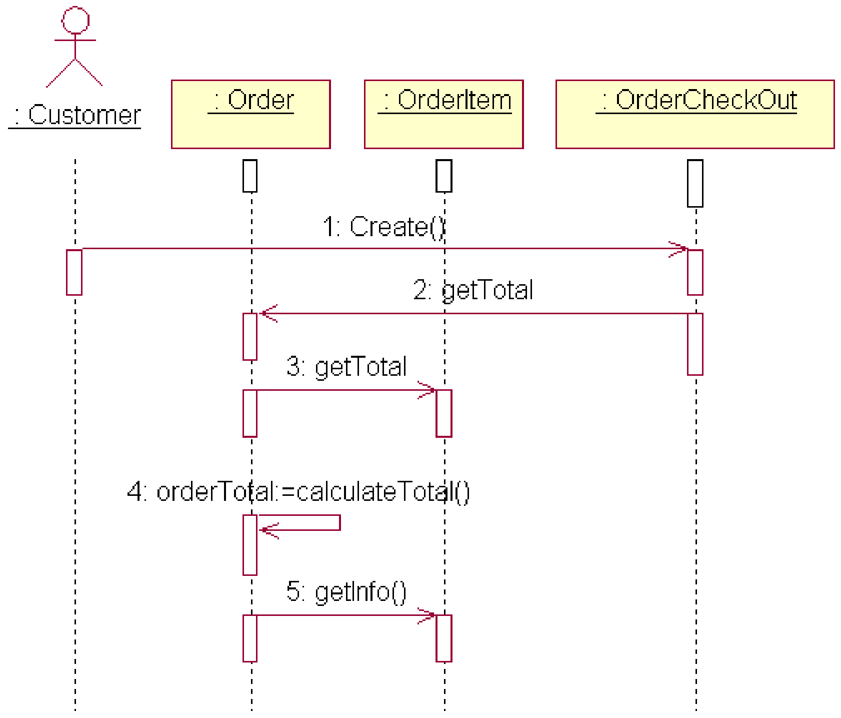
UML includes sequence diagrams as a notation that illustrates actor interactions and operations initiated by them

Example:

The Cashier

- Generates *Event* ➔ *request POS system to record item sale*
- Initiates *Operation* ➔ *makeNewSale()*

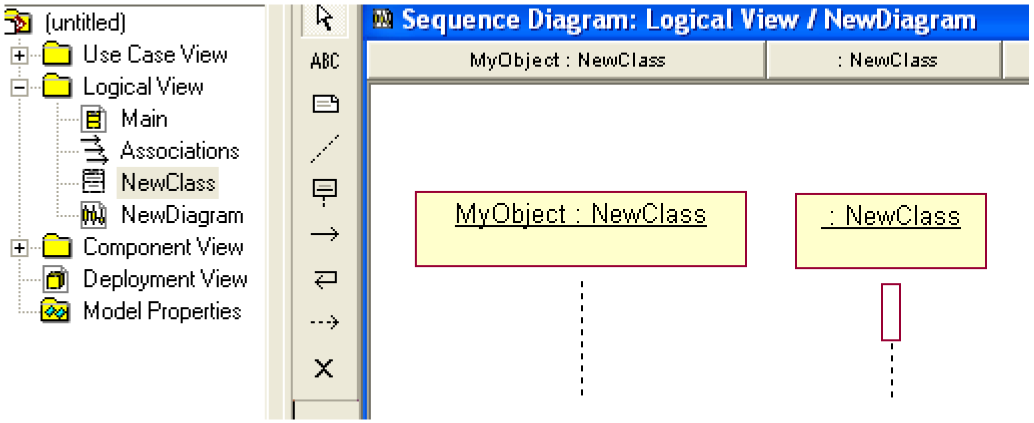
Sequence Diagram Symbols & Notations



What is a UML Sequence Diagram?

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

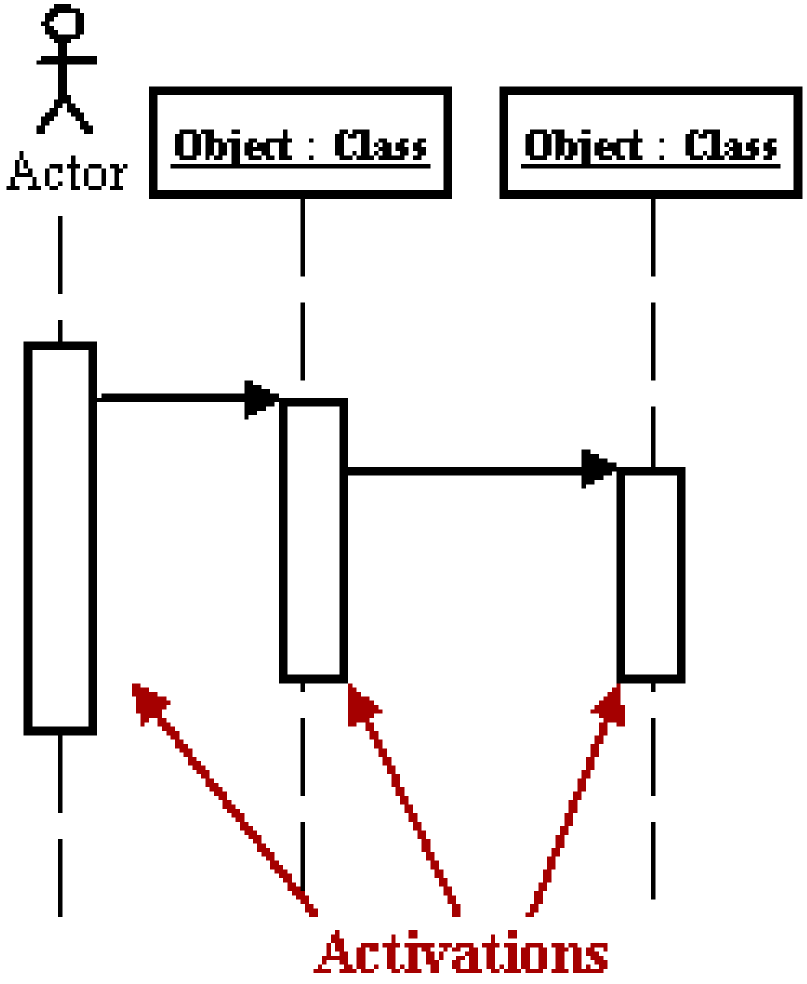
Class Roles



Class roles – “MyObject” - describe the way an object will behave in a context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

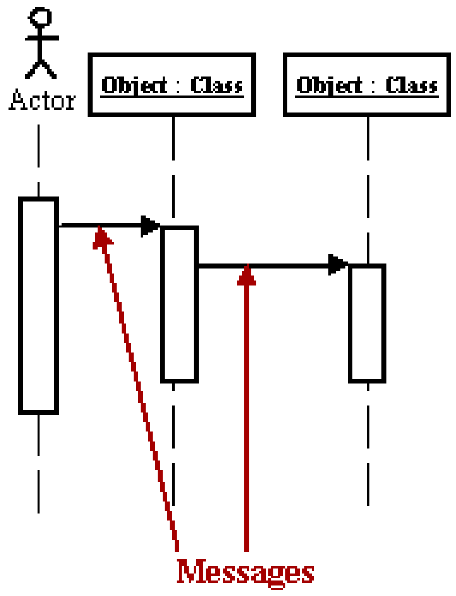
Activation

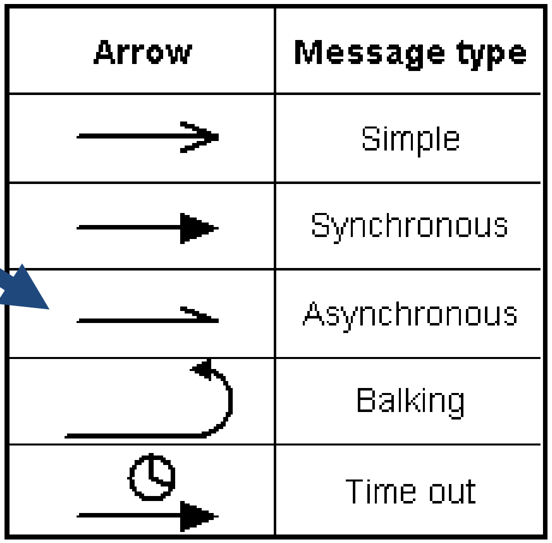
Activation boxes represent the time an object needs to complete a task.



Messages

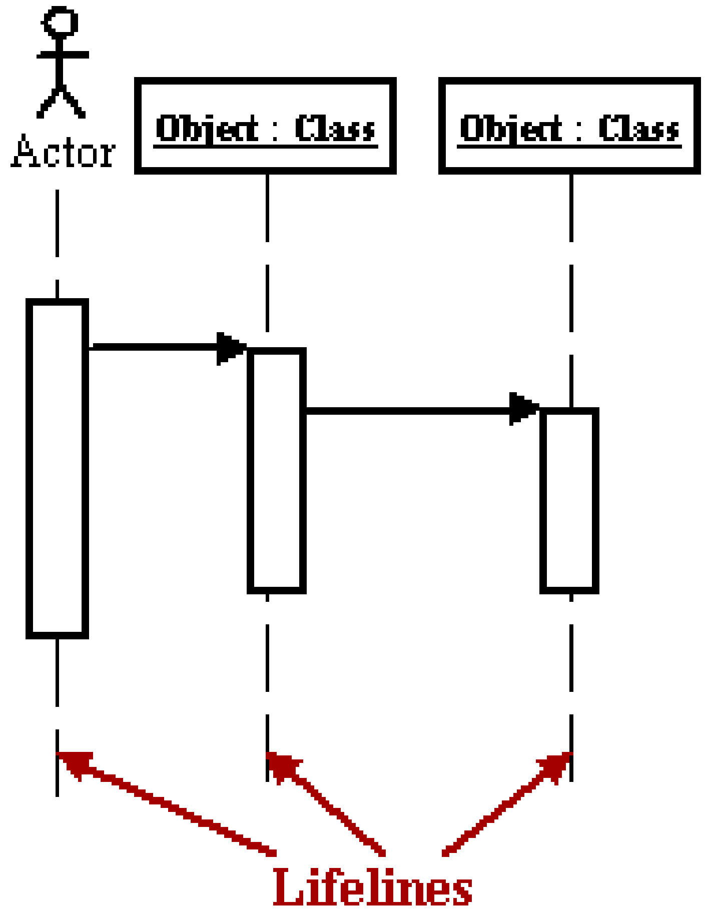
- Messages are arrows that represent communication between objects.
- Order in which messages occur are shown top to bottom on the page
- Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.





Arrow	Message type
→	Simple
➞	Synchronous
⇀	Asynchronous
↩	Balking
◷ ➞	Time out

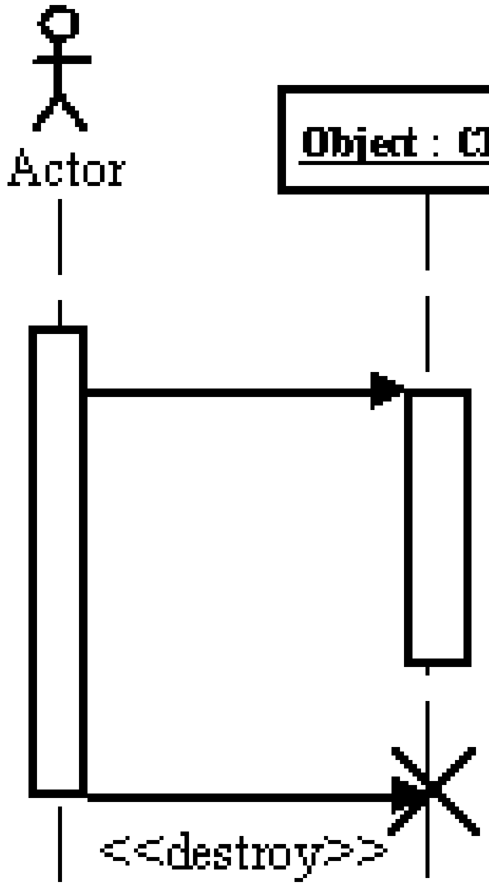
Lifelines



Lifelines are vertical dashed lines that indicate the object's presence over time.

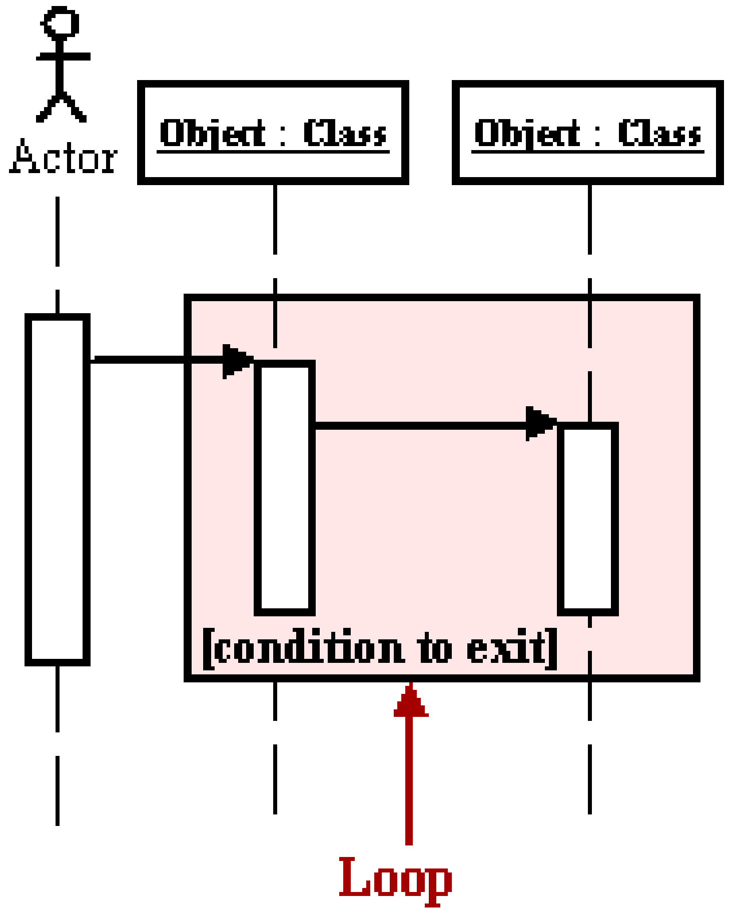
Destroying Objects

Objects can be terminated early using an arrow labelled "< < destroy > >" that points to an X.



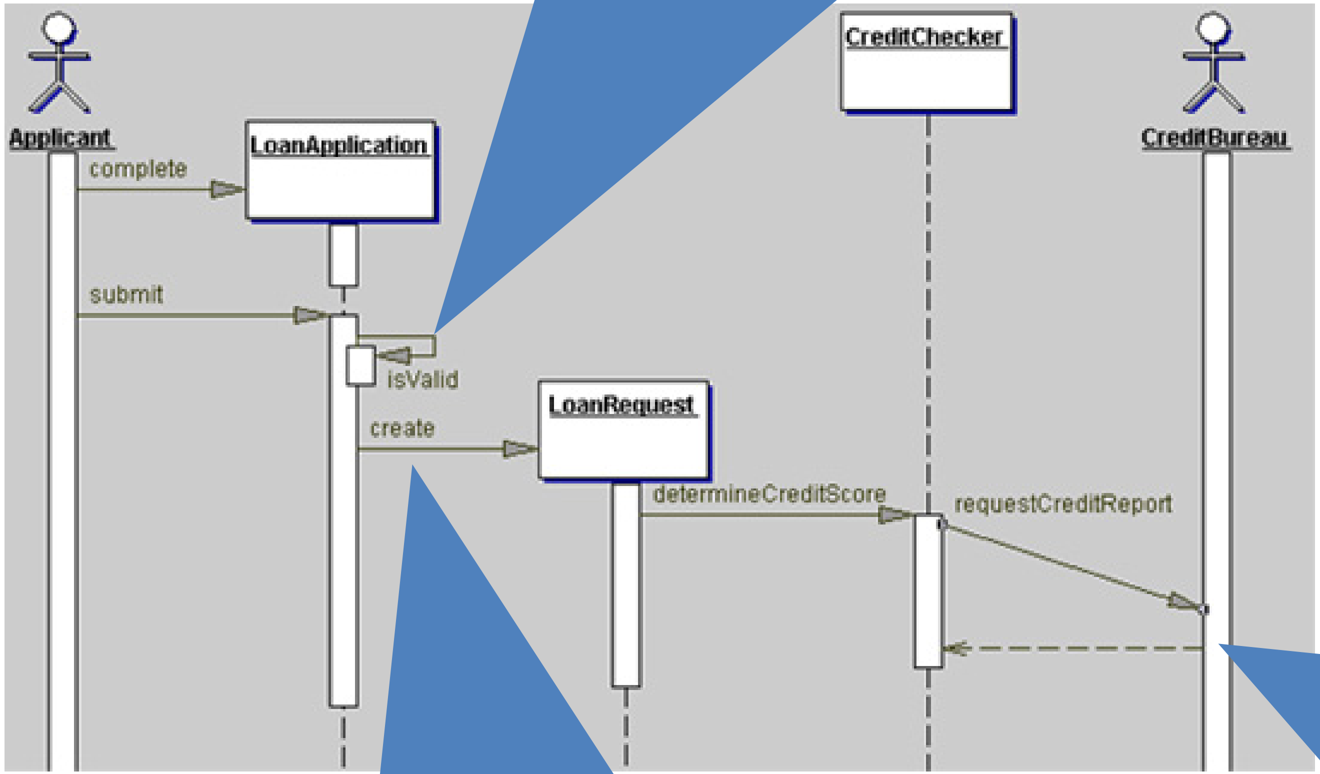
Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



More on Seq. Diagram UML notations

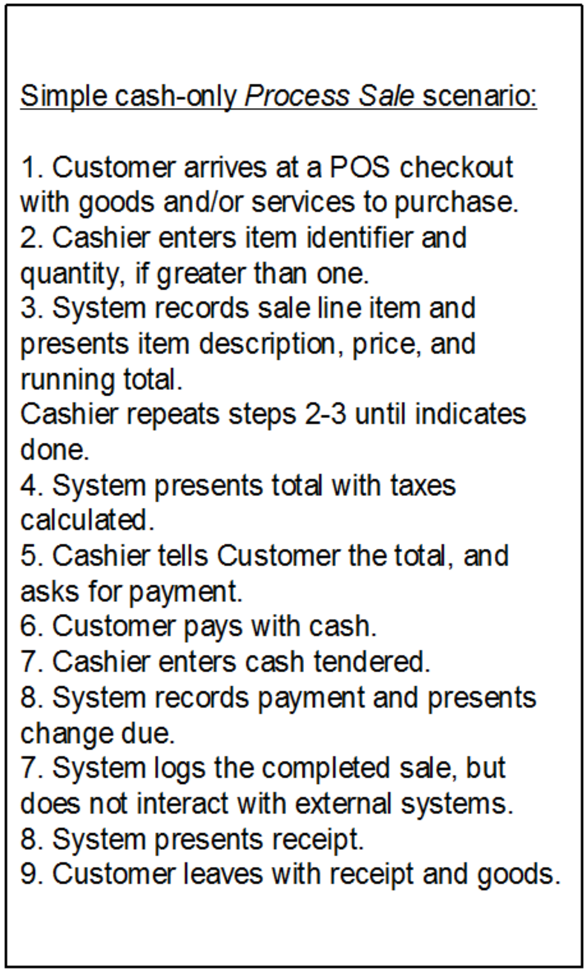
You can show a "self-call', which is a message that an object sends to itself



Return messages differ from regular messages in that the line is dashed. Return messages are not normally drawn as they clutter diagrams.

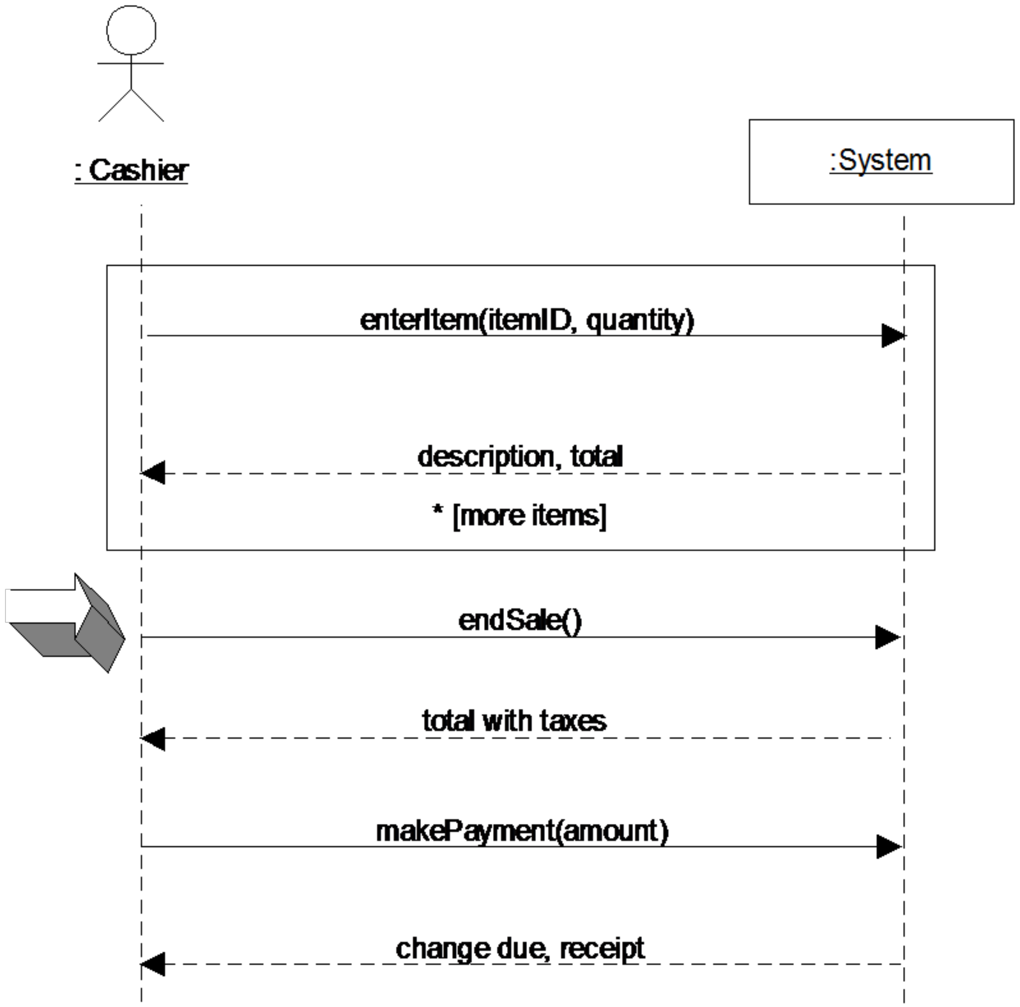
Each message is labeled at a minimum with the message name. You can also include arguments.

Use cases to Sequence Diagrams



Simple cash-only *Process Sale* scenario:

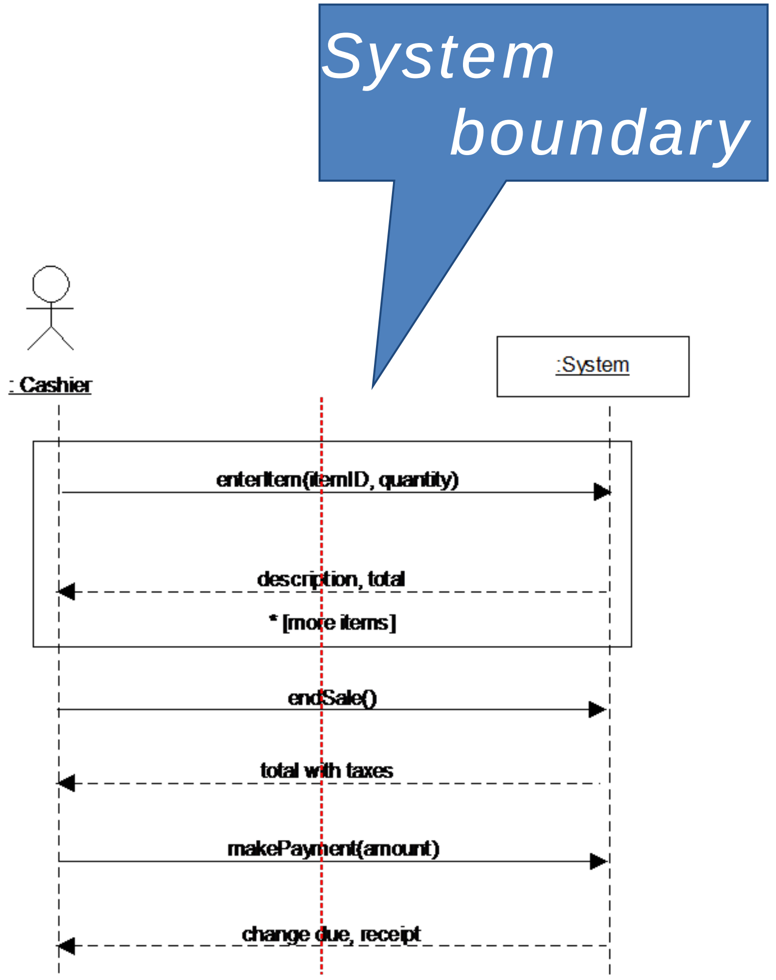
1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier enters item identifier and quantity, if greater than one.
3. System records sale line item and presents item description, price, and running total.
Cashier repeats steps 2-3 until indicates done.
4. System presents total with taxes calculated.
5. Cashier tells Customer the total, and asks for payment.
6. Customer pays with cash.
7. Cashier enters cash tendered.
8. System records payment and presents change due.
7. System logs the completed sale, but does not interact with external systems.
8. System presents receipt.
9. Customer leaves with receipt and goods.



Sequence diagram shows events for a use case scenario

Identify Boundary Classes

- It is necessary to be clear on the choice of system boundary
- The system boundary is usually chosen to be the software system itself.



Naming system events and operators

- Express events and operations at the level of intent rather than in terms of physical input medium

- Start event names with a verb:
  - addXXX
  - enterXXX
  - endXXX
  - makeXXX ….

enterItem

scan

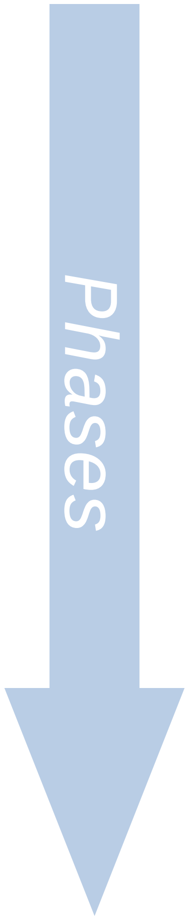
More on sequence diagrams …

- Typically, object that initiates interaction is placed at the left.
- Increasingly, more subordinate objects are placed to the right.

More on sequence diagrams …

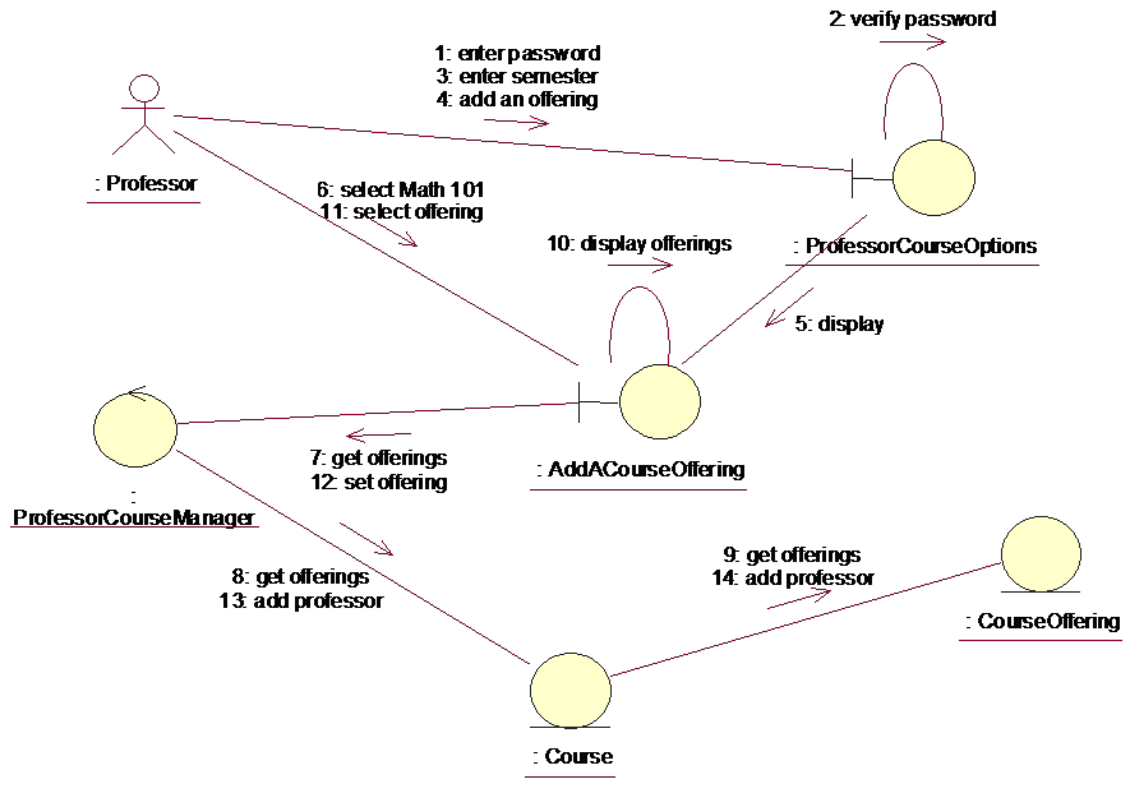
- Objects can be created during an interaction with receipt of stereotype <<create>>.
- Objects can be destroyed during an interaction with receipt of stereotype <<destroy>>.

Phases



- Inception: usually no sequence diagrams
- Elaboration:
  - Most sequence diagrams are created during elaboration
  - Useful for identifying details of the system events
  - Not necessary to create sequence diagrams for all use case scenarios

Collaboration diagrams



- Have a 1:1 correspondence with seq. diagrams
- No lifeline of an object

Questions and Conclusions