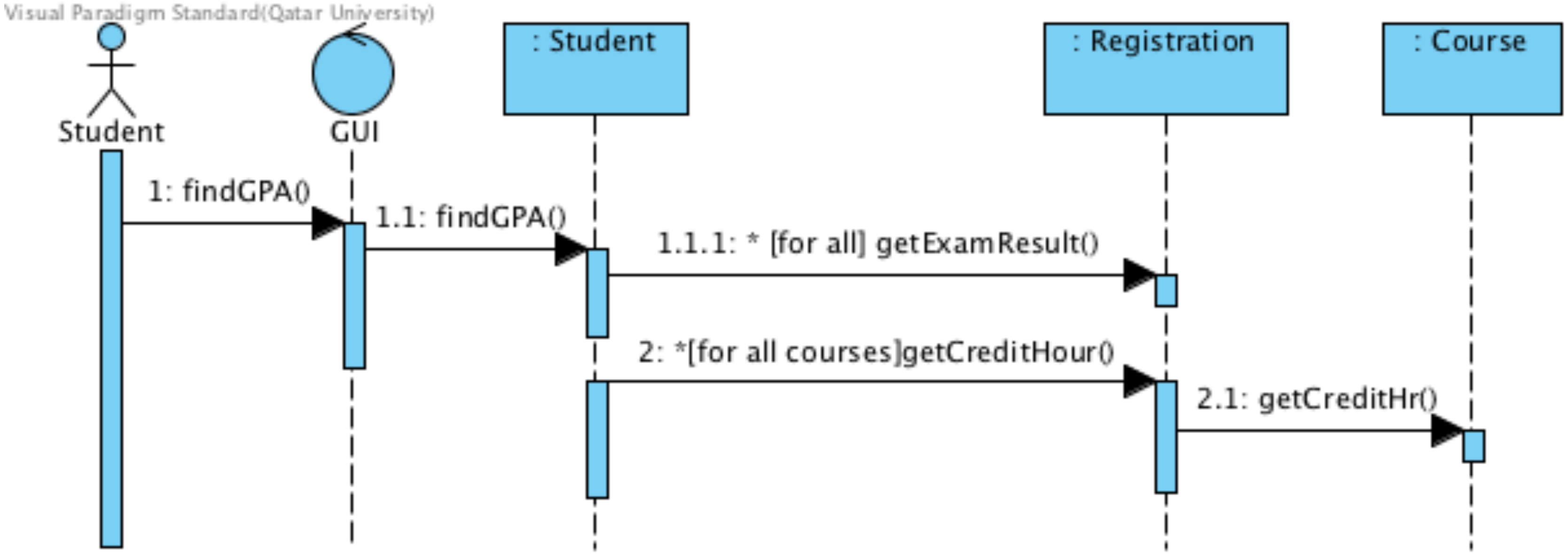
CMPS411
Fall 2018

Lecture 8

Design Sequence Diagram

Example : Design Sequence Diagram


Visual Paradigm Standard(Qatar University)
Student
GUI
: Student
: Registration
: Course
1: findGPA()
1.1: findGPA()
1.1.1: * [for all] getExamResult()
2: *[for all courses]getCreditHour()
2.1: getCreditHr()

- Use case “Process GPA Calculation”
- Remember this Use case?

Objects Need to Collaborate

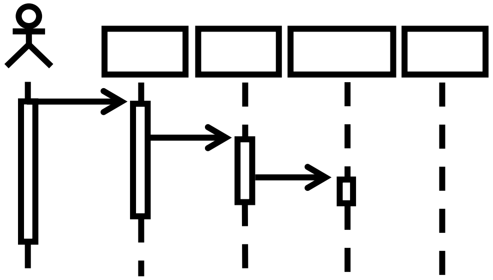
- Objects are useless unless they can collaborate to solve a problem.
  - **No one object can carry out every responsibility on its own**.
- How do objects interact with each other?
  - They interact through messages.
    - A method call is the most common type of message.
  - An ***interaction* is a set of messages exchanged** among a set of objects in order to accomplish a specific goal.

Elements of Design Sequence Diagrams

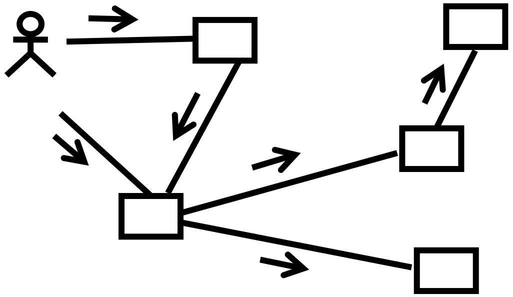
- **Instances of classes**
  - Shown as boxes with the class and object identifier underlined
- **Actors**
  - Use the stick-person symbol as in use case diagrams
- **Messages**
  - Shown as arrows from actor to object, or from object to object
  - A message is the vehicle by which communication between objects is achieved.
    - A *call* is the most common type of message.
  - The *return* of data as a result of a function call is also considered a message.
  - A message may result in a **change of state for the receiver of the message**.
  - The receipt of a message is considered an **instance of an event**.

Three Interaction Diagrams

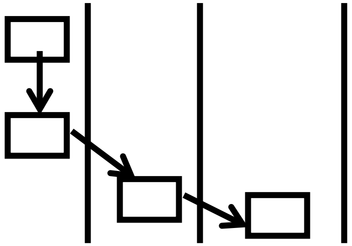
- **Design Sequence Diagram**
  - **Time oriented view** emphasize the time ordering of the interactions. The diagram shows:
    - The **objects participating** in the interaction.
    - The **sequence of messages exchanged**.
- **Communication Diagram**
  - **Shows how the objects related to each other**
  - Emphasize the **structural organization** of the objects participating in interactions:
    - The objects participating in the interaction.
    - **Links between the objects**.
    - Messages passed between the objects.
- **Activity Diagram**
  - Show the flow of activity of an use case
  - Which actor does which activity and how the system responses in terms of a sequence of activity



Sequence Diagram



Communication Diagram



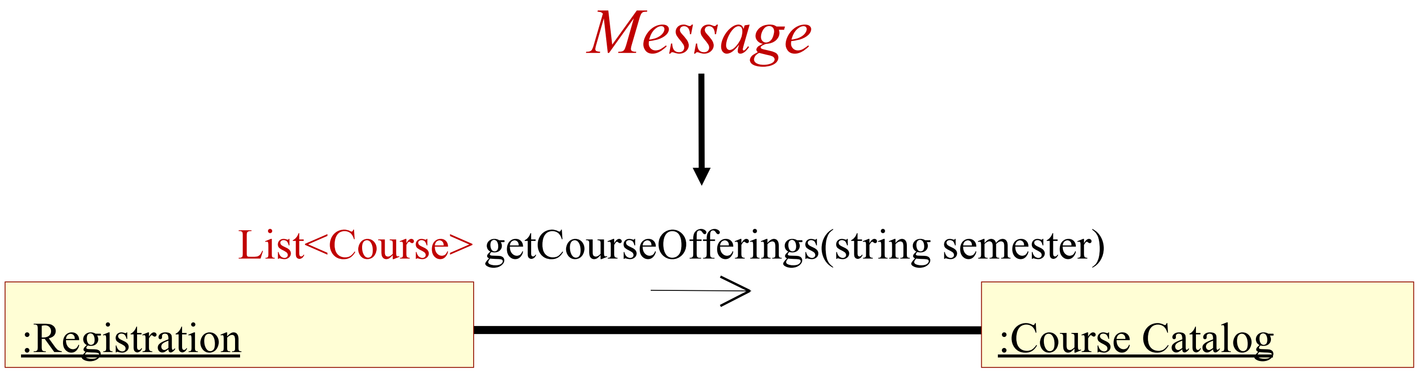
Activity Diagram

Design Sequence Diagrams

- A design sequence diagram shows the sequence of messages exchanged by the set of objects performing a certain task
  - Represents the white box representation of the system
  - Shows the internal objects of the system
  - Model which object handles which message
  - Shows how messages propagate inside the system boundary
  - The objects are arranged horizontally across the diagram.
  - An actor that initiates the interaction is often shown on the left.
  - The **vertical dimension represents time**.
  - A vertical line, called a ***lifeline***, is attached to each object or actor.
  - The lifeline becomes a broad box, called an *activation box* during the ***live activation*** period.
  - A **message** is represented as an arrow between activation boxes of the sender and receiver.
    - A message is numbered and labelled and can have an argument list and a return value.

Objects Interact with Messages

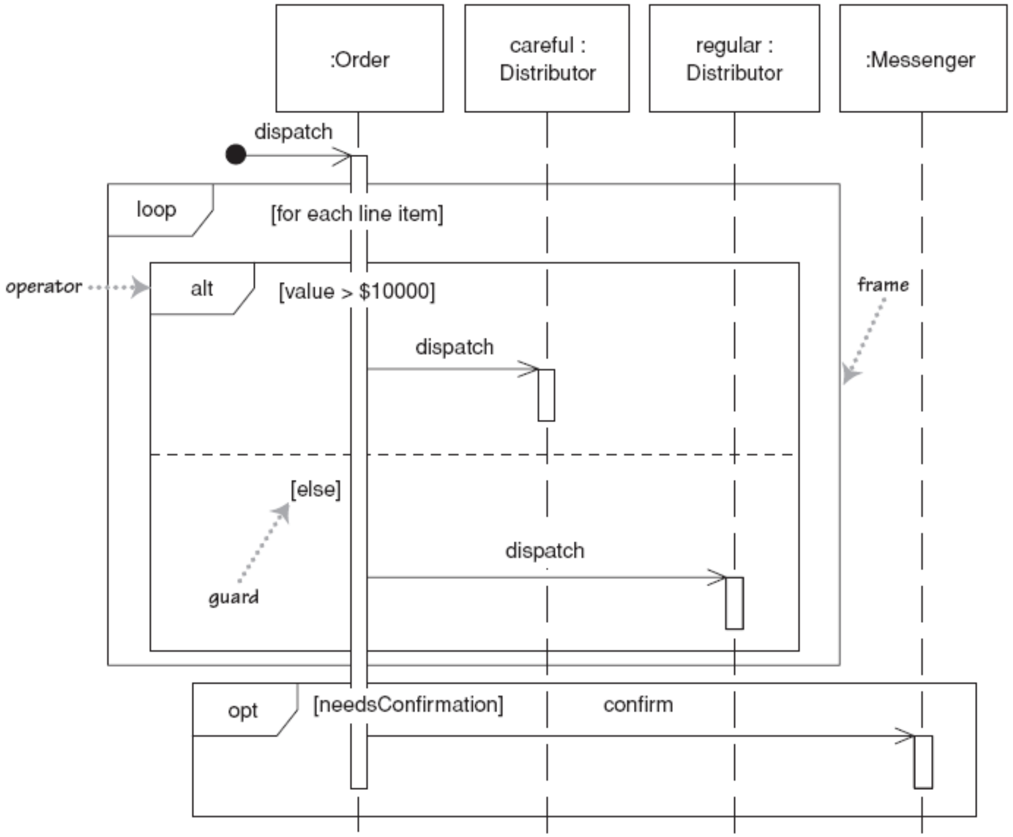
- **Prticipant**: an object or entity that acts in the sequence diagram
  - sequence diagram starts with an unattached "found message" arrow
- **Message**: communication between participant objects
- the axes in a sequence diagram:
  - horizontal: which object/participant is acting
  - vertical: time (down -> forward in time)
- A message (i.e., method call) shows how one object asks another object to perform some activity.



- When the activity has been executed, the control is returned to the caller along with a return value.

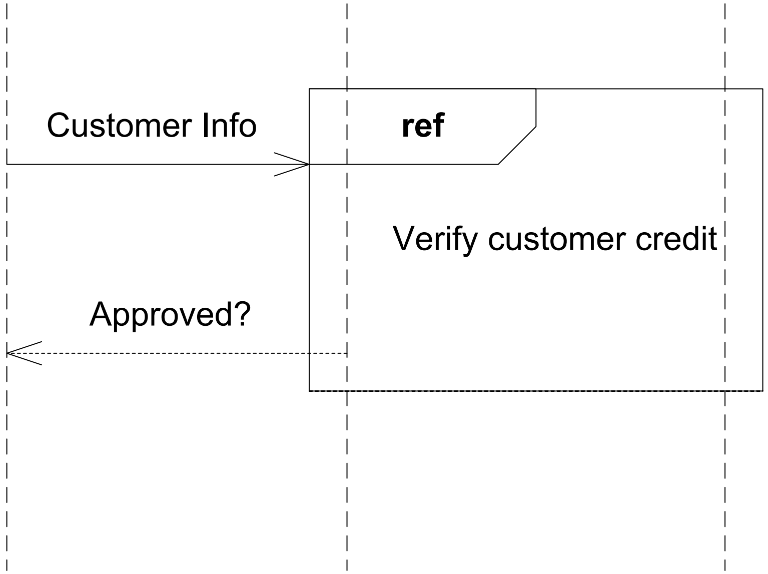
Indicating selection and loops

- 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?



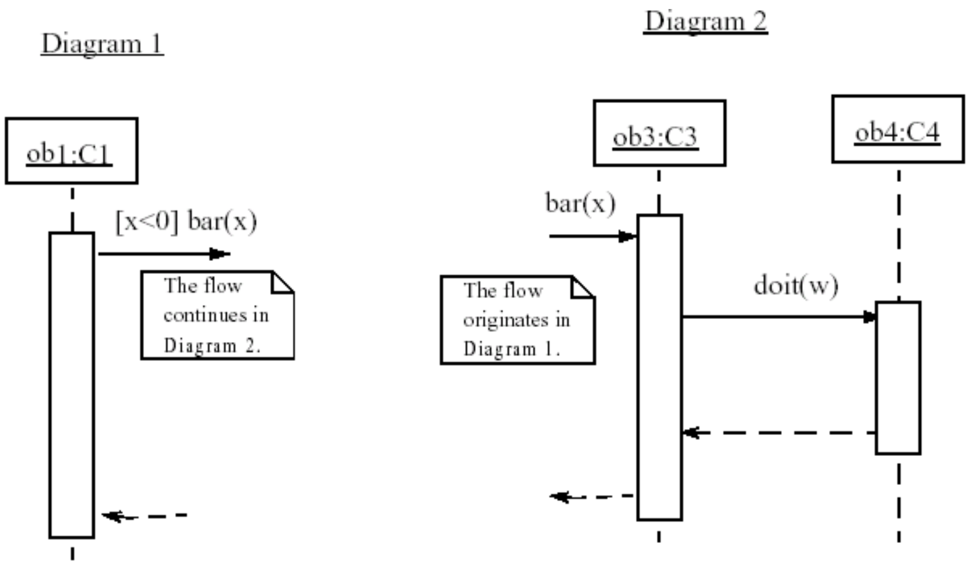
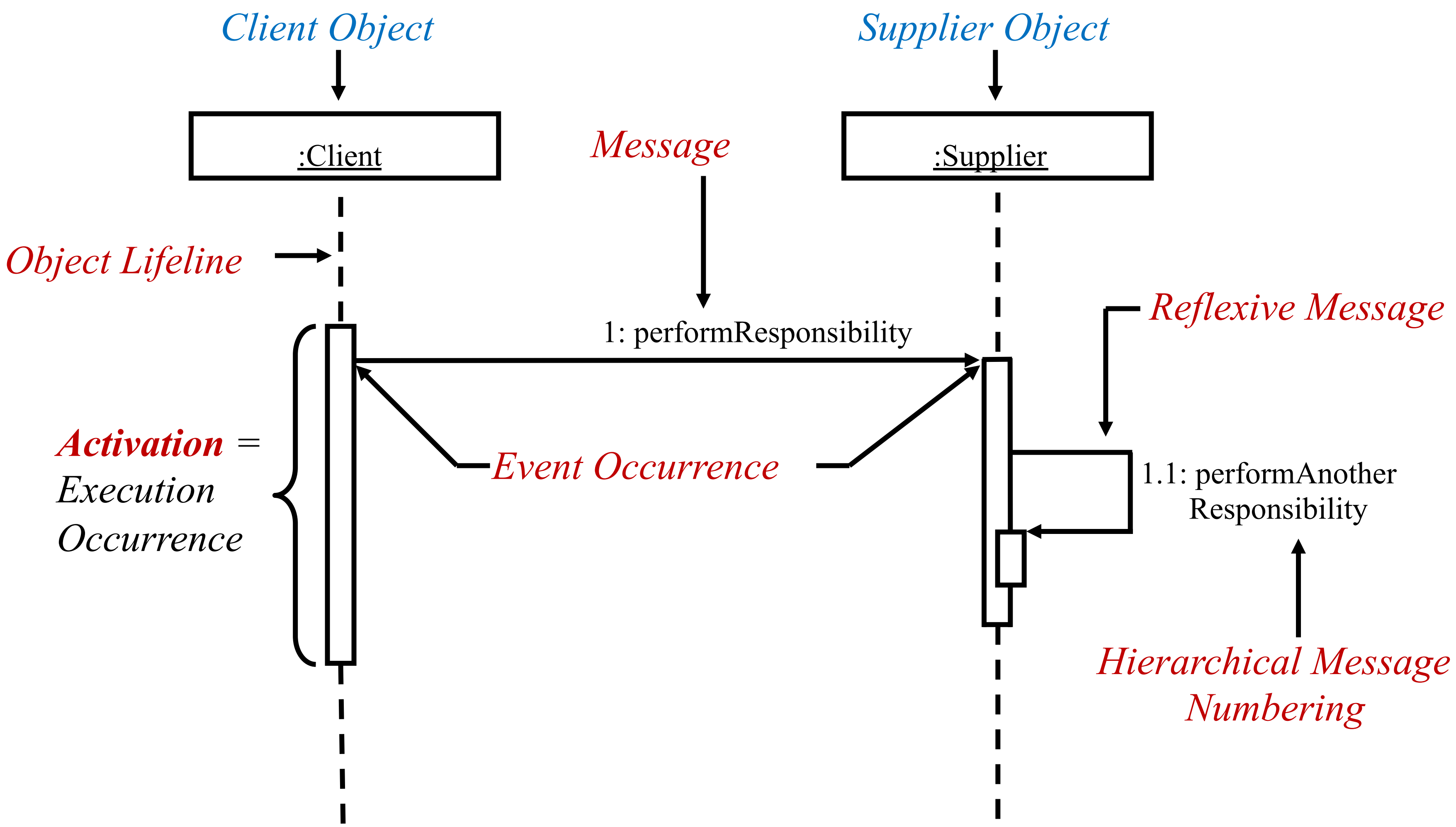


Diagram 1
ob1:C1
[x<0] bar(x)
The flow continues in Diagram 2.
Diagram 2
ob3:C3
ob4:C4
bar(x)
doit(w)
The flow originates in Diagram 1.

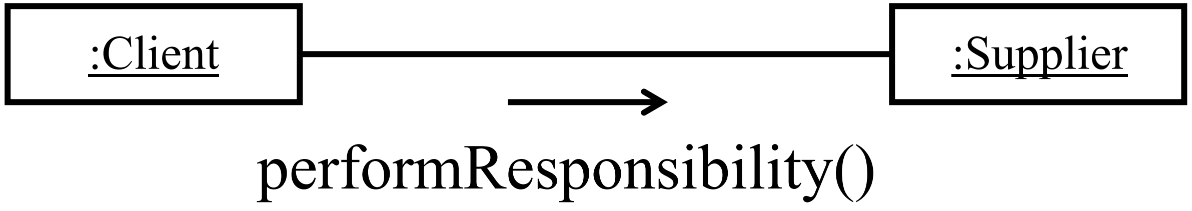
The Anatomy of Design Sequence Diagrams



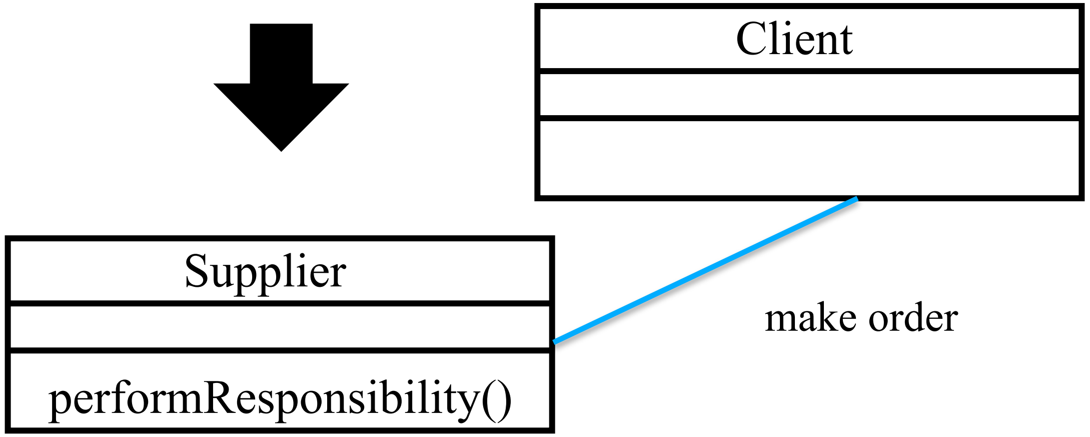
Relationship between classes and Design Sequence Diagram

- Sequence diagram messages are methods of in the supplier class!

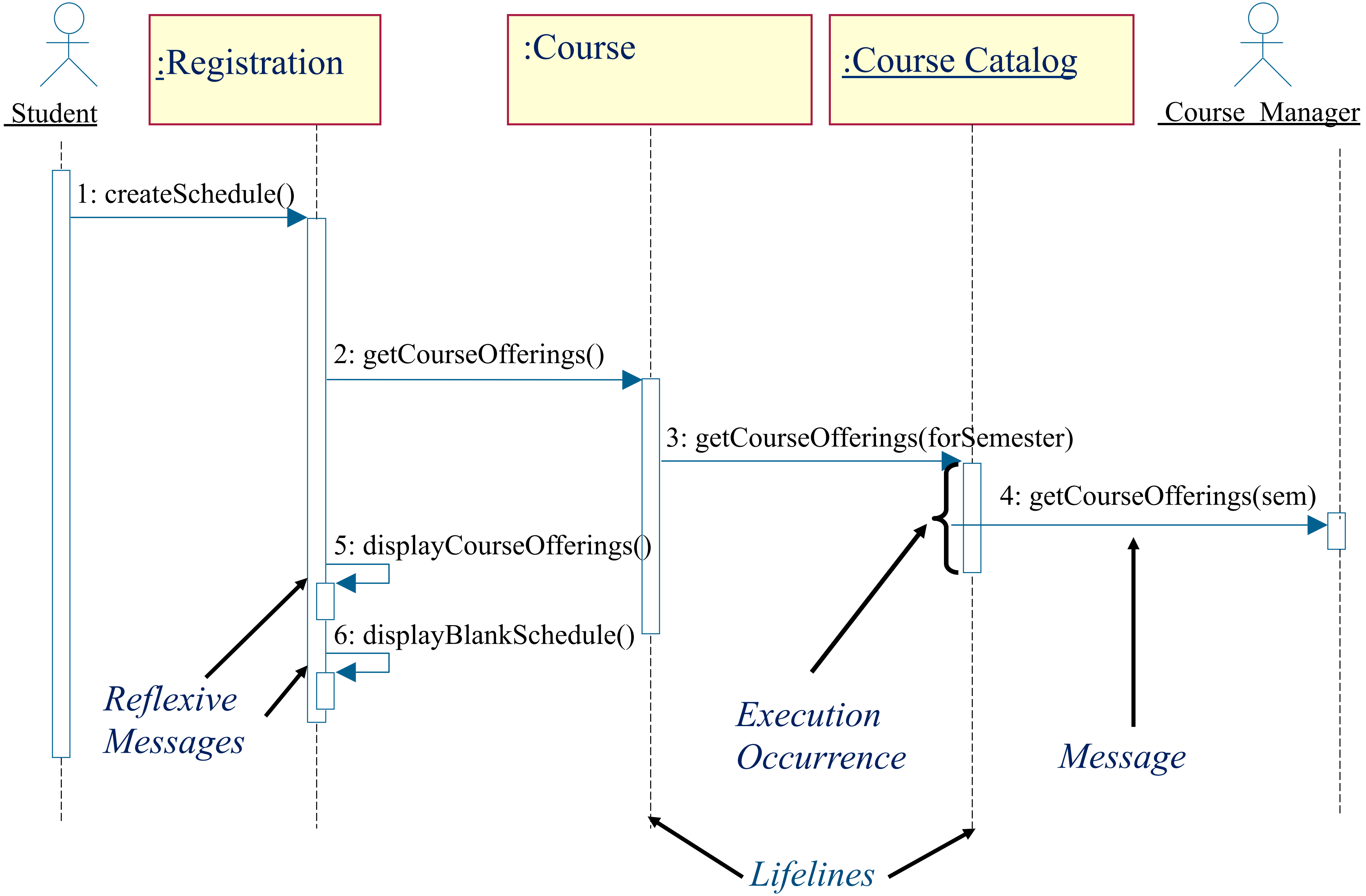
Sequence Diagram



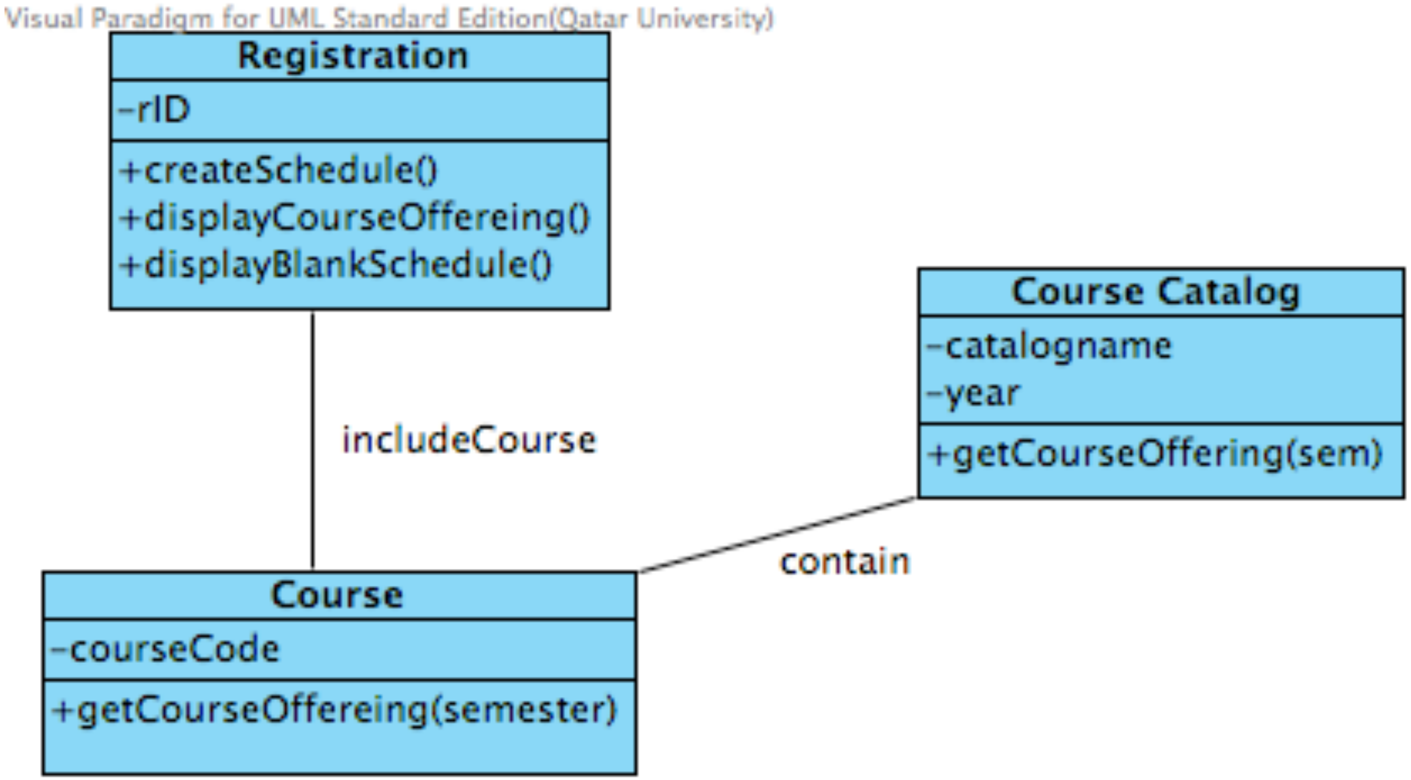
Class Diagram



Sequence Diagram for Register for Courses Use Case



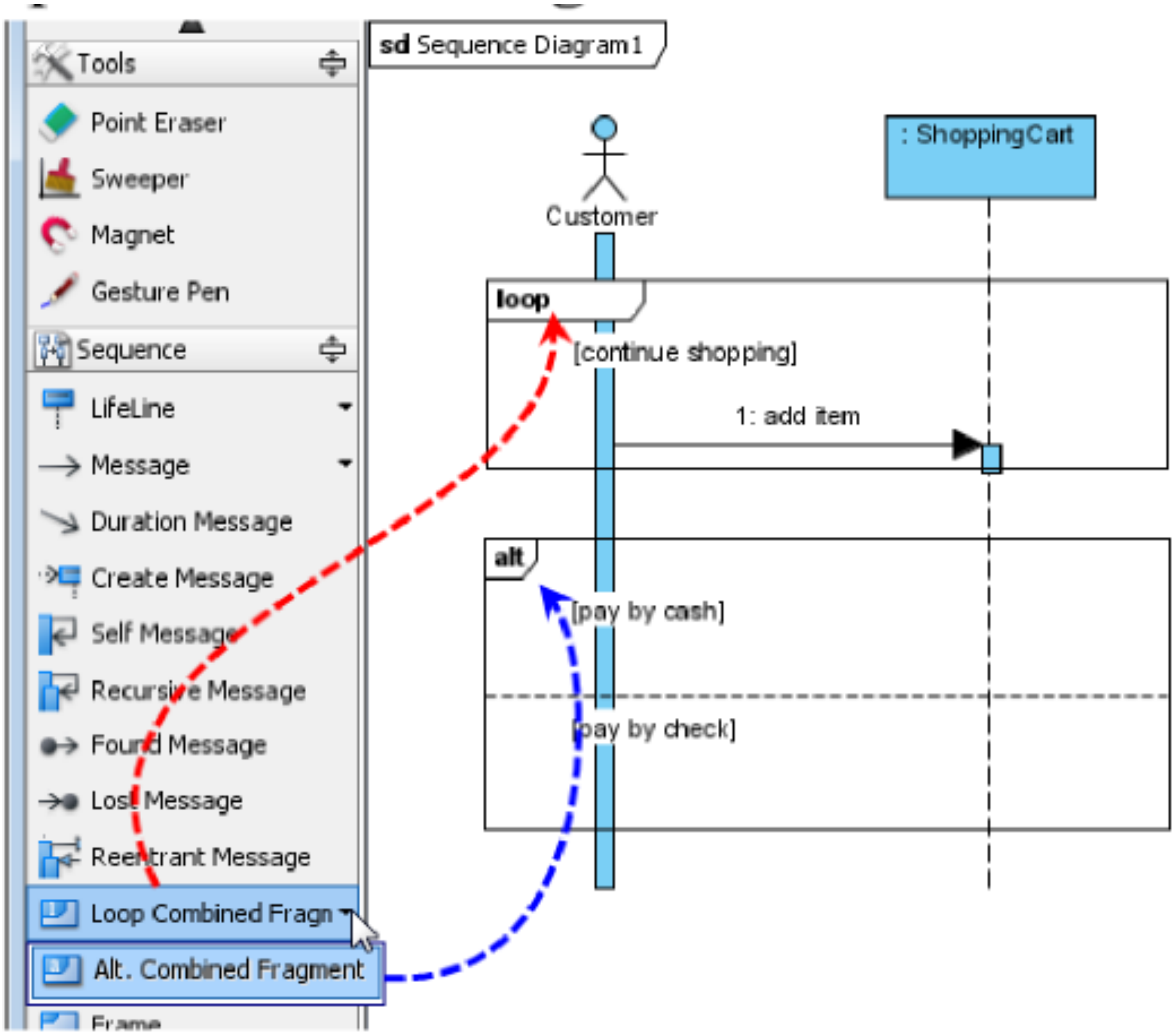
Class Diagram of The Previous Diagram



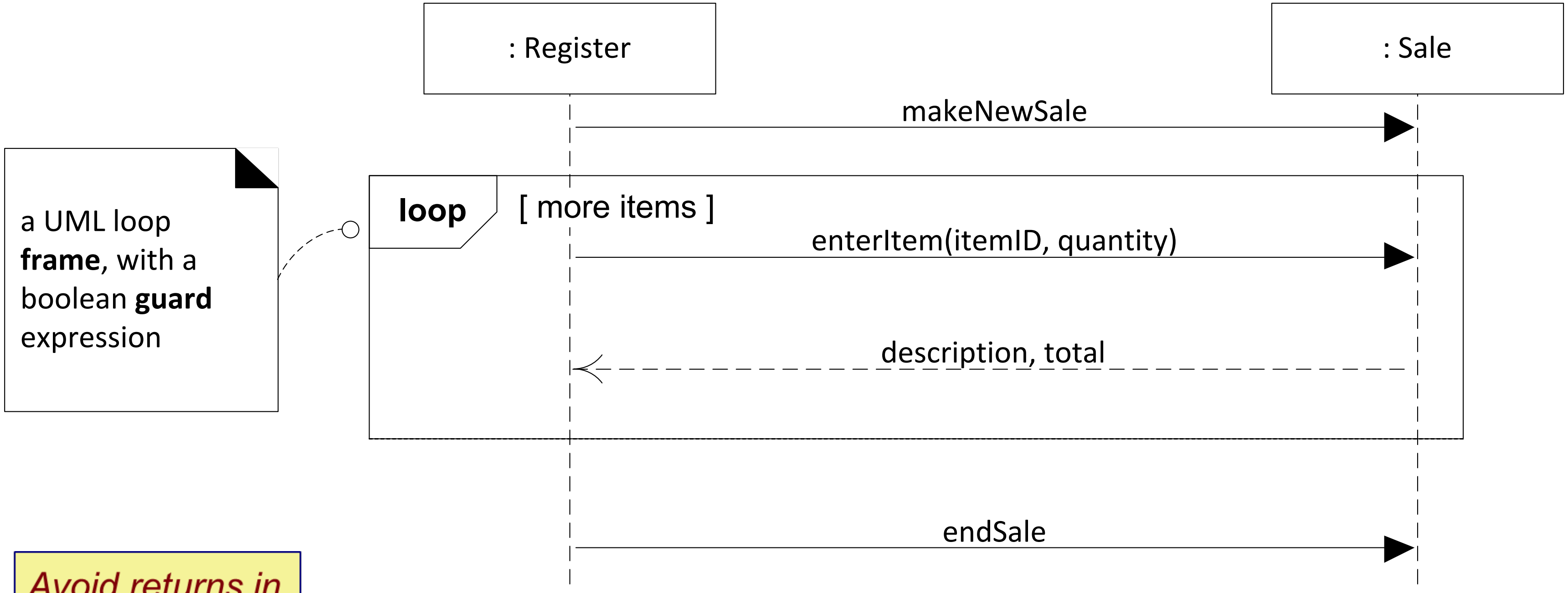
Design Sequence Diagram Common Operators

- Alternative fragment (denoted “**alt**”) models **if…then…else constructs.**
  - Guard condition specify the true case for the execution of the interaction
- Option fragment (denoted “**opt**”) models **switch constructs**.
  - Guard condition specified for each case.
- **Loop** fragment encloses a series of messages which are repeated.
  - Guard condition specify the lower and upper limit of the loop.
- **“ref”** refers to an interaction defined on another diagram.
- Parallel fragment (denoted “**par**”) models concurrent processing.

Example of using Alt and Loop

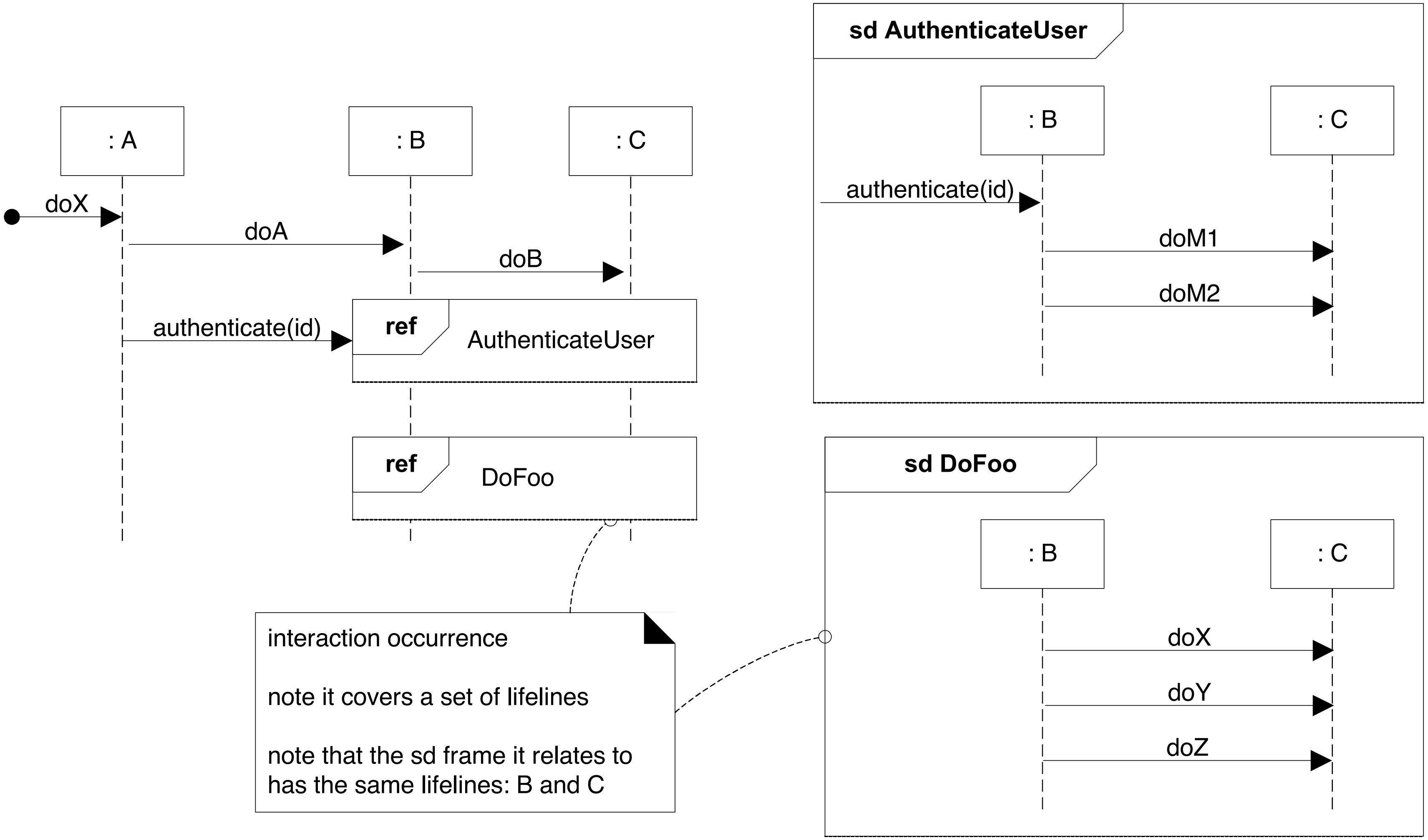


Design Sequence Diagram with loop


: Register
: Sale
makeNewSale
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale
a UML loop **frame**, with a boolean **guard** expression

Avoid returns in sequence diagrams unless they add clarity.

Referencing Another Design Sequence Diagram (DSD)



interaction occurrence

note it covers a set of lifelines

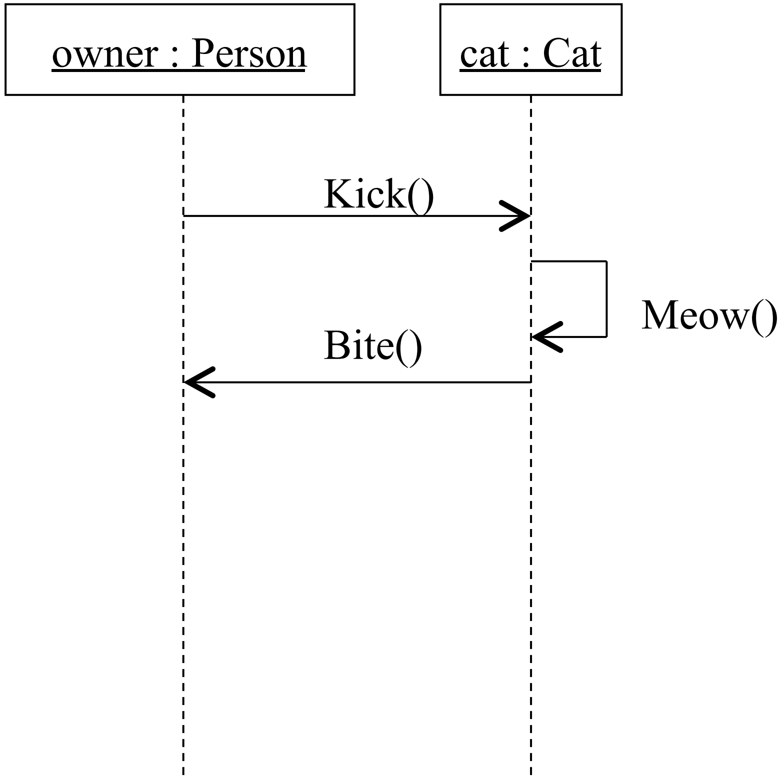
note that the sd frame it relates to has the same lifelines: B and C

Modeling Interaction – How-To

- Determine what scenario need to be modeled
  - Start drawing one interaction diagram per scenario. If, at some point, the diagram becomes complex, split your interaction diagram and draw one diagram per system function.
- Identify the objects that play a role in the scenario.
- Lay the objects out in a sequence diagram left to right, with the *most important* objects to the left.
  - *Most important* in this context means objects that are the principle initiators of events.
- Draw in the message arrows, top to bottom.
  - Each message is assigned a *sequence number* to show the time order of the message.
  - Method calls should show the parameters and the return type.

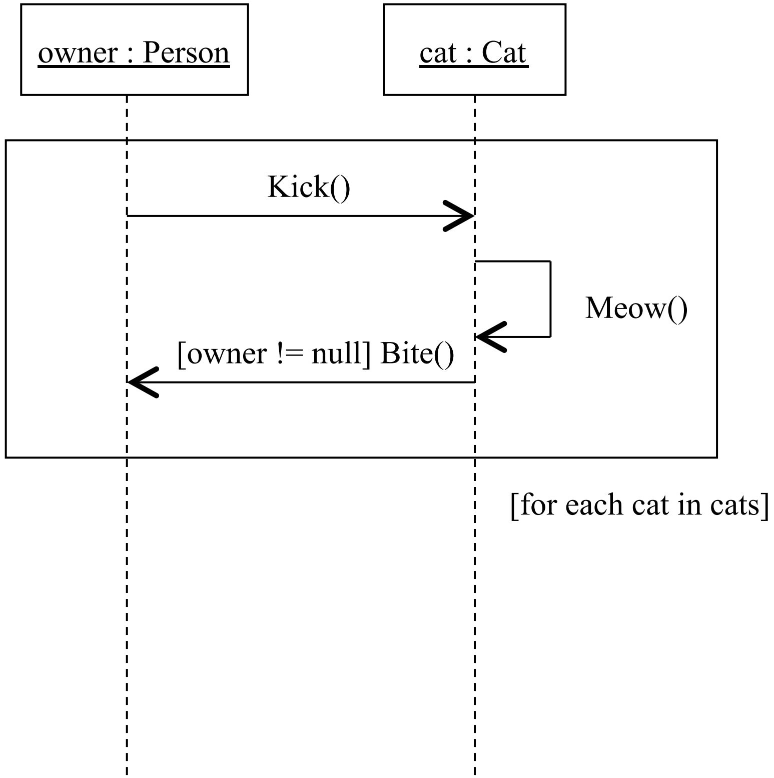
Relationship between Design Sequence Diagram and Code (1)

```
public class Cat
{
        private Person owner;

        public void Kick()
        {
                Meow();
                owner.Bite();
        }

        public void Meow()
        {
        }
}

public class Person
{
        private Cat cat;

        public void KickCat()
        {
           // start here
          cat.Kick();
        }

        public void Bite()
        {
        }
}
```



Relationship between Design Sequence Diagram and Code (2)

```
public class Cat
{
            private Person owner;

            public void Kick()
            {
                        Meow();
                        if(owner != null)
                        {
                          owner.Bite();
                        }
            }

            public void Meow()
            {
            }
}

public class Person
{
            private IList cats;

            public void KickCat()
            {
                  // start here
                  foreach(Cat cat in cats)
                  {
                    cat.Kick();
                  }
            }

            public void Bite()
            {
            }
}
```



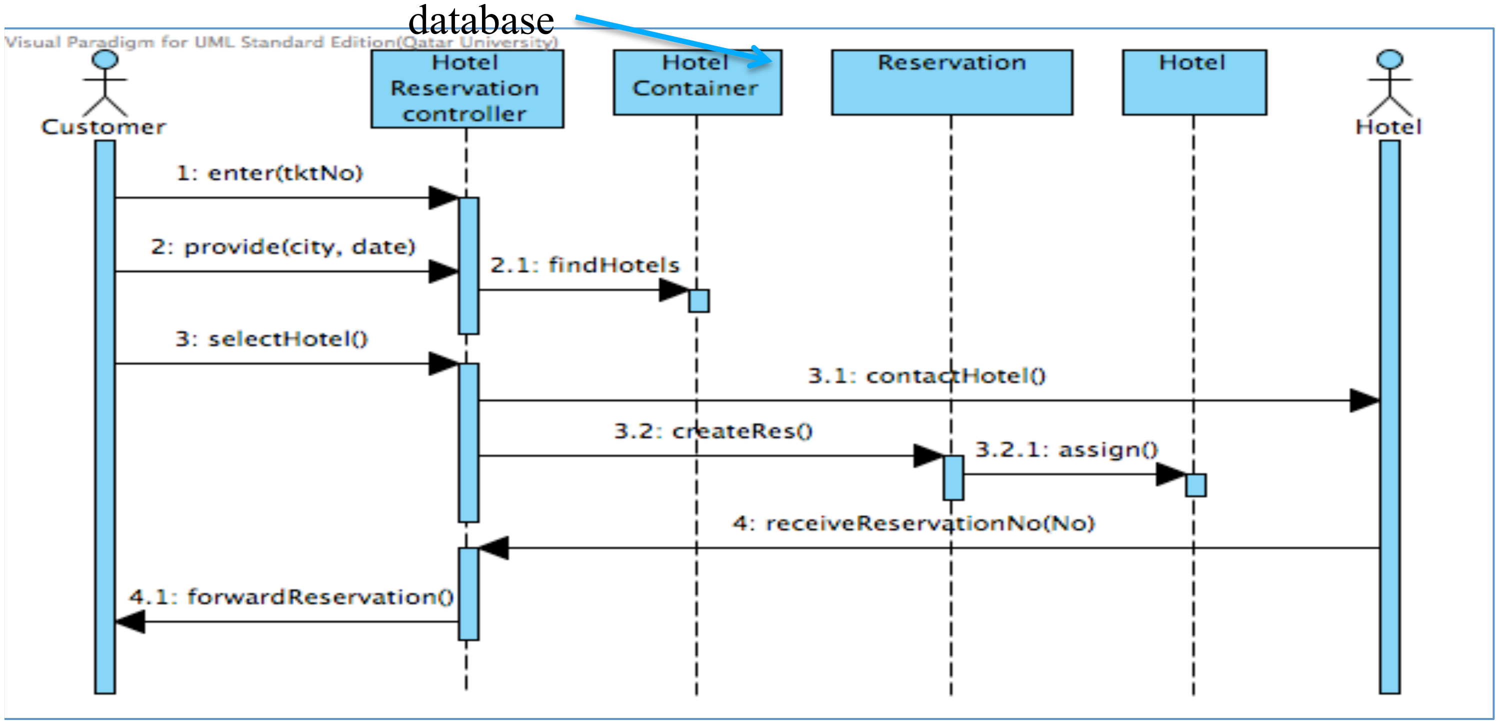
Example: Use case “Book Hotel”

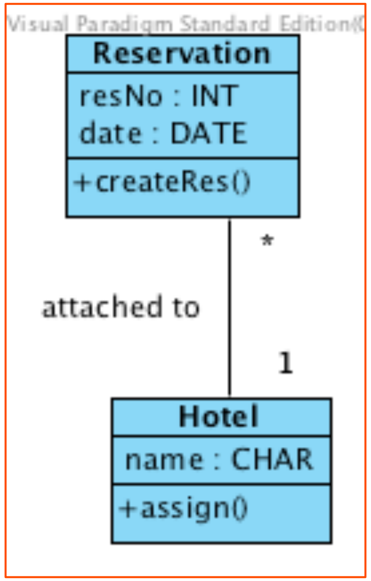
Use case: Book hotel

Actor Action	System Response
1. The customer provides ticket number or booking number	2. Asks for the city, and dates
3. The customer provides the city and dates	4. Find a list of hotel
5. The customer selects the hotel	6. Contacts the hotel
	7. Makes reservation
8. The hotel sends reservation number	9. Forward the reservation information to the customer
10. The customer receives reservation information.	

Use case is transformed to design sequence diagram as shown in the next slide

Sequence Diagram: Book Hotel





- Notice the use of Controller class (Boundary class can also be used here)
- Controller class does not have any method to execute, it simply acts as an interface between the use case and the outside of the system
- Controller class does not have any data or methods.
- “Hotel Container” is used here as a collection of objects (database)
- Container/collection class usually executes some pre-defined operations such as find, search, add, delete, etc.
- Container class in not shown in the class diagram.

Another Example: Modify Seat

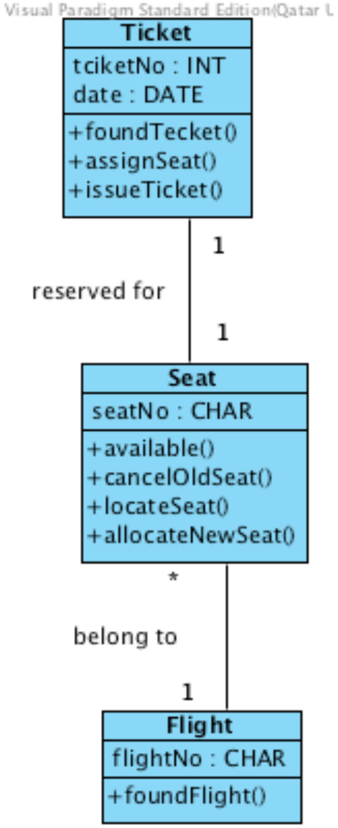
Use Case: Modify seat

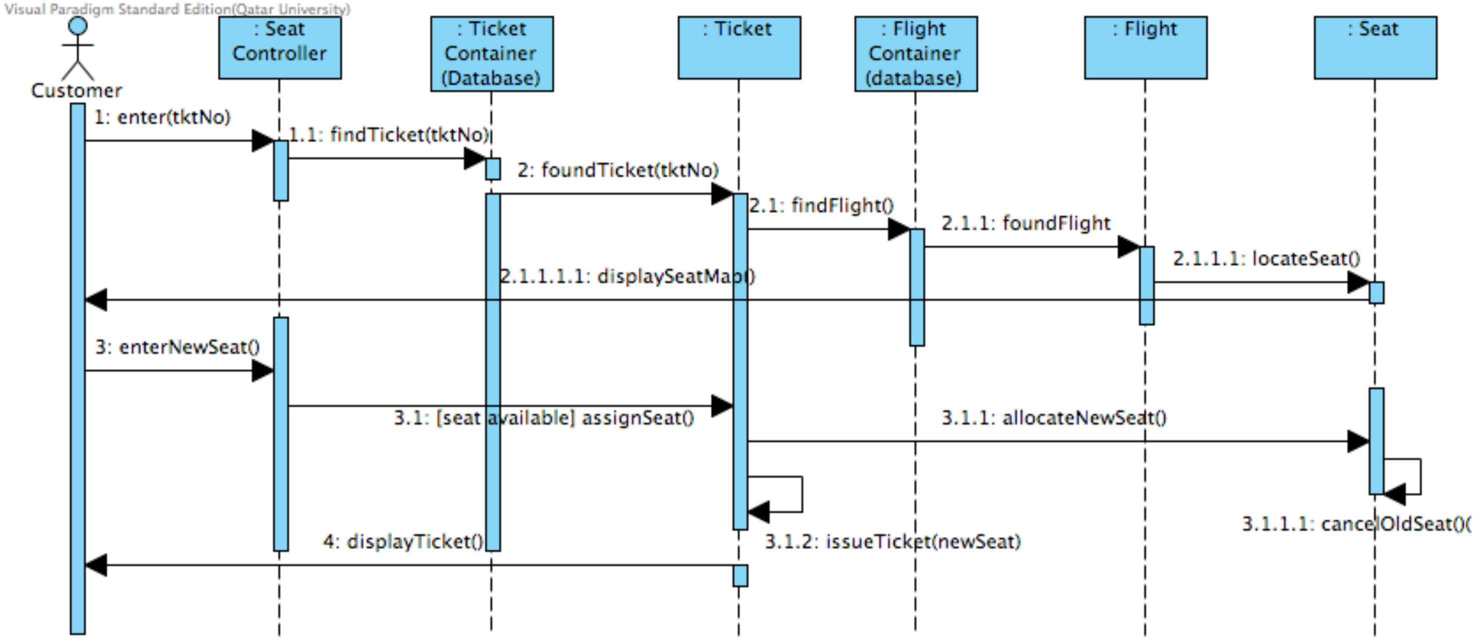
Actor Action	System Response
1. The customer provides the ticket number or booking number	2. Finds the ticket information
	3. Provide the seat map to change the seats if enough seats are available
4. The customer selects new seats	5. Record the new seat information, and release the old seat reservation.
	6. Issue the ticket with new seats

Alternative flows:

3a. If enough seats are not available, produce error message

Transform to




Visual Paradigm Standard Edition(Qatar University)
Customer
: Seat Controller
: Ticket Container (Database)
: Ticket
: Flight Container (database)
: Flight
: Seat
1: enter(tktNo)
1.1: findTicket(tktNo)
2: foundTicket(tktNo)
2.1: findFlight()
2.1.1: foundFlight
2.1.1.1: locateSeat()
2.1.1.1.1: displaySeatMap()
3: enterNewSeat()
3.1: [seat available] assignSeat()
3.1.1: allocateNewSeat()
3.1.1.1: cancelOldSeat()(
3.1.2: issueTicket(newSeat)
4: displayTicket()

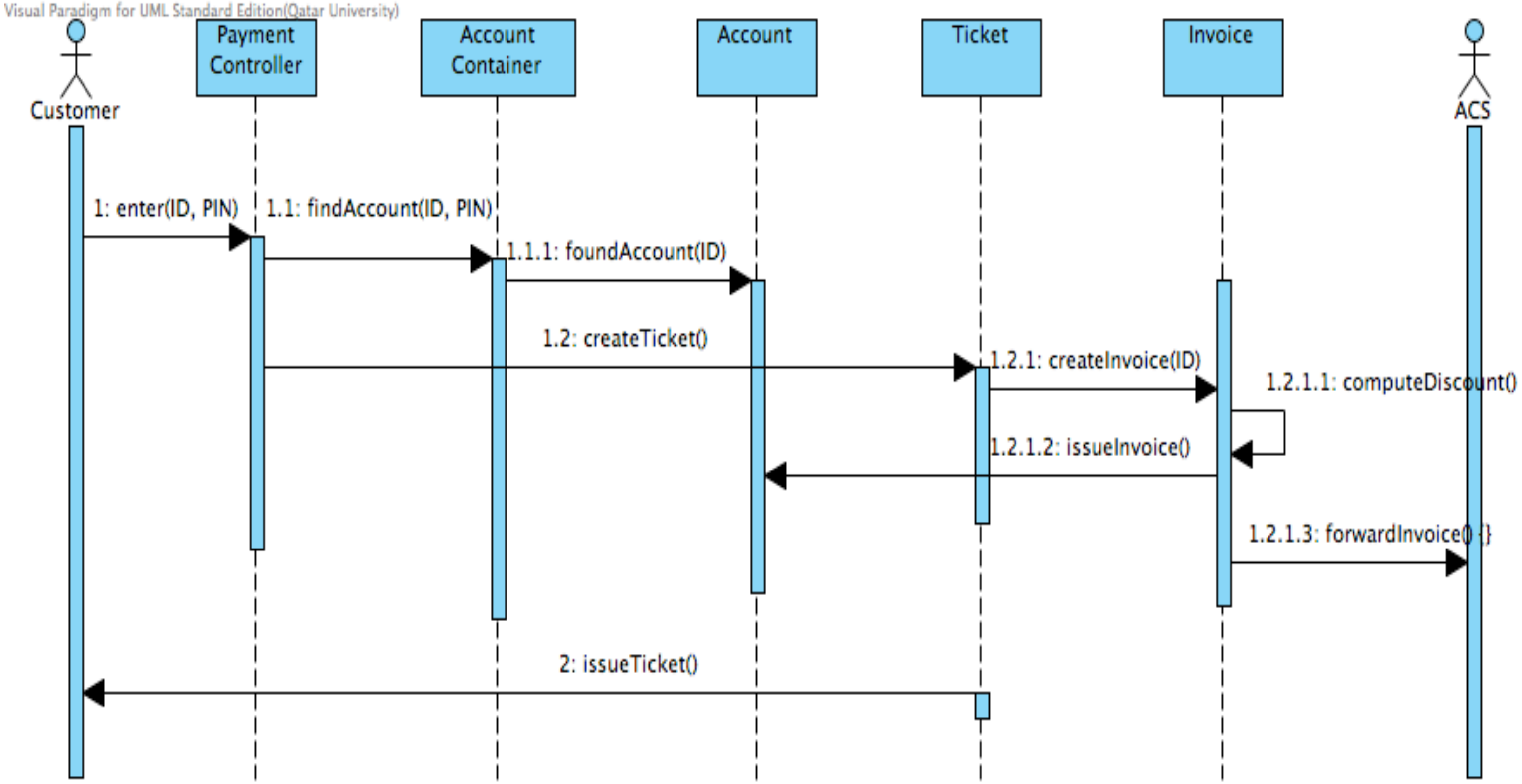
- Notice the use of Controller class (Boundary class can also be used here)
- Two container/collection classes are used here.

Another Example

Use case: Pay for regular customer

Actor action	System response
1. The customer enters ID and PIN	2. Find the account details
	3. Prepare an invoice
	4. Computes 10% discount
	5. Issue the invoice
	6. Forward the invoice to ACS
	7. Attach the invoice with the account
	8. Prepare and issue ticket.

Transform to



Rules for Design Sequence Diagram

① Objects used in a sequence diagram must have corresponding classes in the class diagram

② If an object sends a message to another object in the sequence diagram, there must be an association in the class diagram between the corresponding classes.

③ If an object sends a message to another object in the sequence diagram, there must be a similar methods in the corresponding class of the message receiving object in the class diagram.

④ If an object calls a reflexive message to itself, the corresponding class must have the same method in the class diagram.

⑤ You can use *** [guard condition]** in front of a message to indicate a loop, instead of using loop box.

⑥ You can use **[guard condition]** in front of a message to indicate if..then, without using “alt” box.

⑦ Every sequence diagram must have at least one controller class

⑧ Controller class does not any method to execute, it is just a interface and control messages received from/and to actor(s)

⑨ A container class is a data storage/database representing the objects of a class. In the class diagram, you do not show these container classes because by default, every entity class in the class diagram has a corresponding database.