LECTURE 7

UML 순서 다이어그램

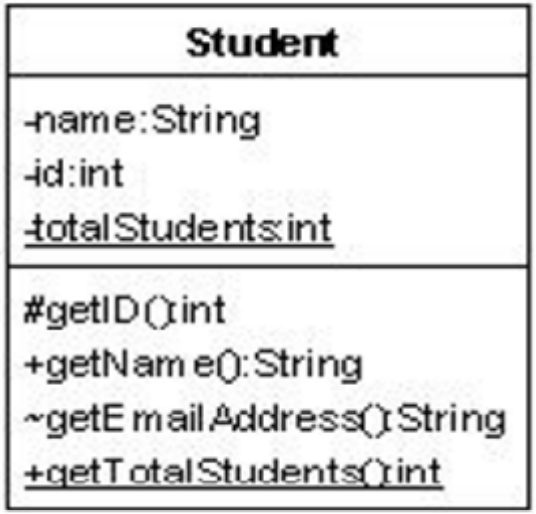
–순서 다이어그램은 시스템의 동적 스냅샷

목 차

- UML 클래스 다이어그램 복습
- UML 순서 다이어그램
- UML 정리

UML 클래스 다이어그램

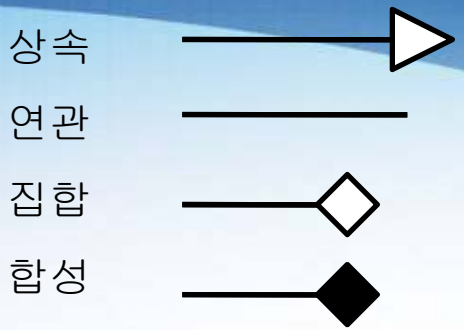
- 클래스 이름
- 속성 – 객체가 갖는 데이터 필드
  - *visibility name: type*
- 오퍼레이션 – 간단한 메소드, 상속 메소드 생략
  - *visibility name(prameters) : return_type*
  - visibility: + public
    # protected
    - private
  - Underline <u>static variable</u>



클래스의 관계

- 일반화 – 클래스 사이의 상속
- 연관 – 클래스 사이의 연결
  - 의존(dependency)
    - 실선, 임시적인 의존이면 점선
  - 집합(aggregation) 
    - 다른 클래스를 포함(contains)
  - 합성(composition) 
    - 포함된 클래스가 컨테이너 클래스 없이는 존재하지 않음
- 다중도, 방향성

관계 파악 연습



Leap Year Table

Calendar

Calendar Entry

Executive Calendar

Conference Room

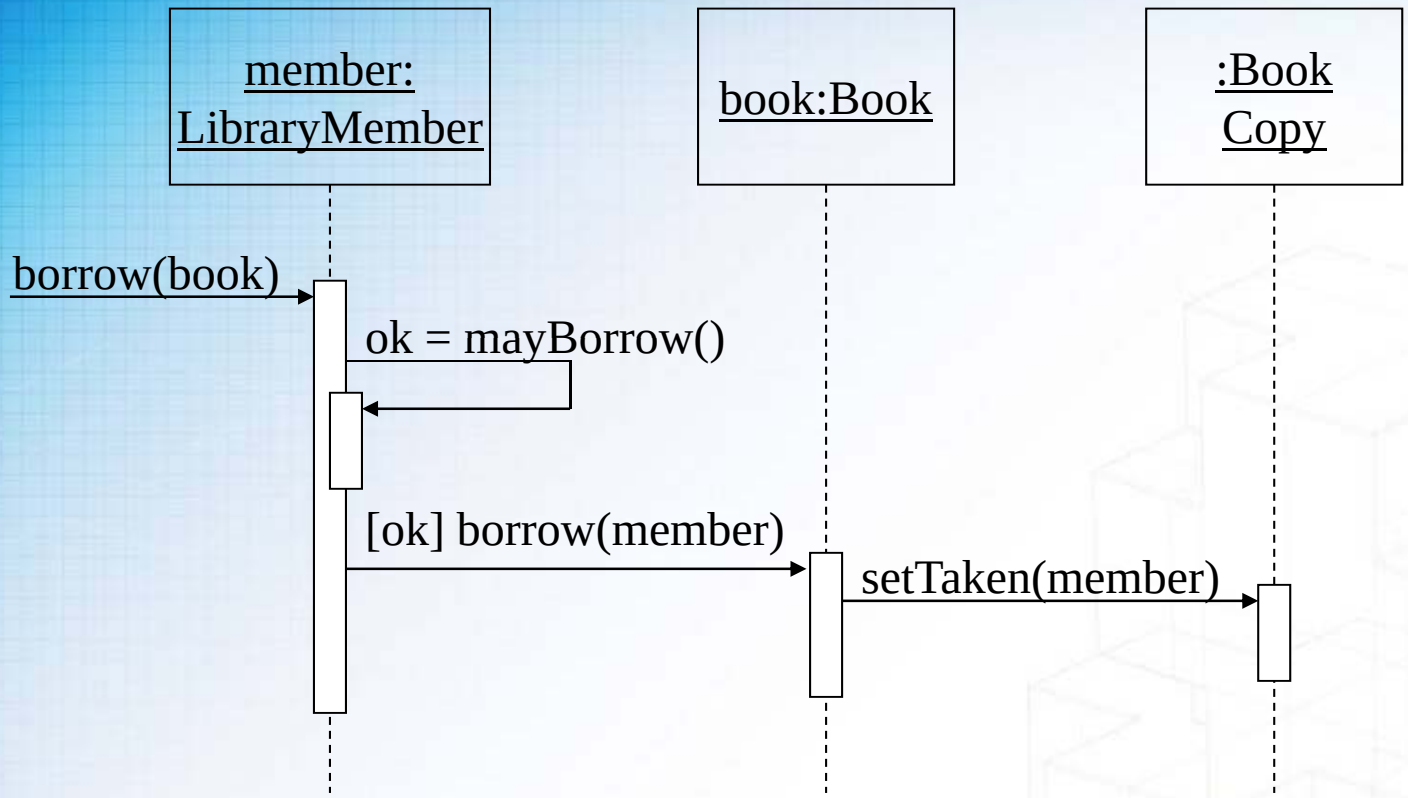
UML 순서 다이어그램

❑ 순서 다이어그램(Sequence Diagram)

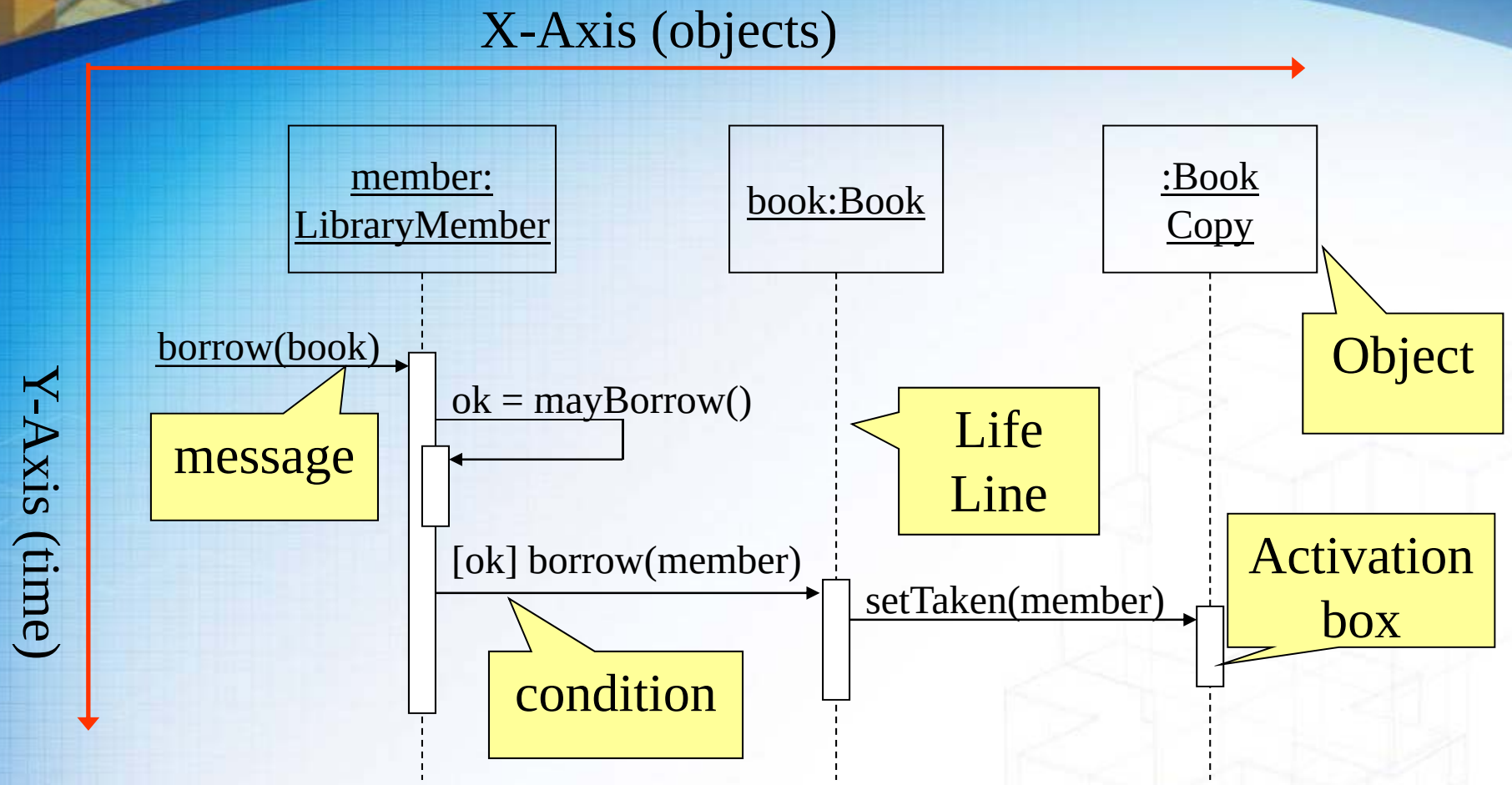
사용 사례가 어떻게 수행되는지 – 어떤 메시지가 언제 보내지는지 나타낸 그림

- 시스템의 동적인 측면을 캡처한 것 – dynamic view
- 시간의 흐름에 따라 정리해 놓은 것 – 페이지 내려갈 수록 시간이 흐름
- 객체는 왼쪽에서 오른쪽으로 나열 메시지 호출의 흐름에 언제 참여하였느냐에 따라

순서 다이어그램


member:
LibraryMember
book:Book
:Book
Copy
borrow(book)
ok = mayBorrow()
[ok] borrow(member)
setTaken(member)

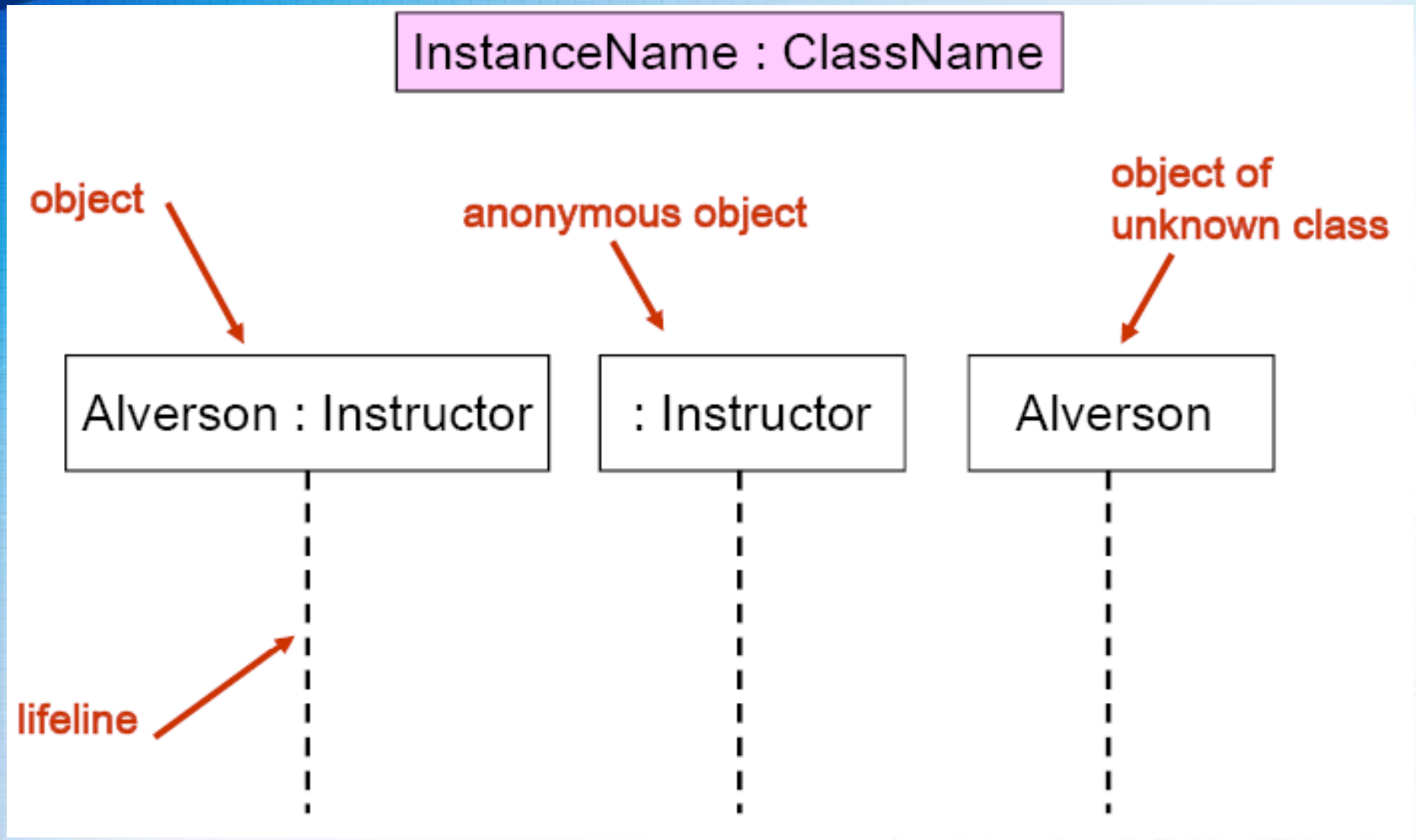
순서 다이어그램의 요소


X-Axis (objects)
Y-Axis (time)
member:
LibraryMember
book:Book
:Book
Copy
borrow(book)
ok = mayBorrow()
[ok] borrow(member)
setTaken(member)
message
Life
Line
Object
Activation
box
condition

그리는 방법

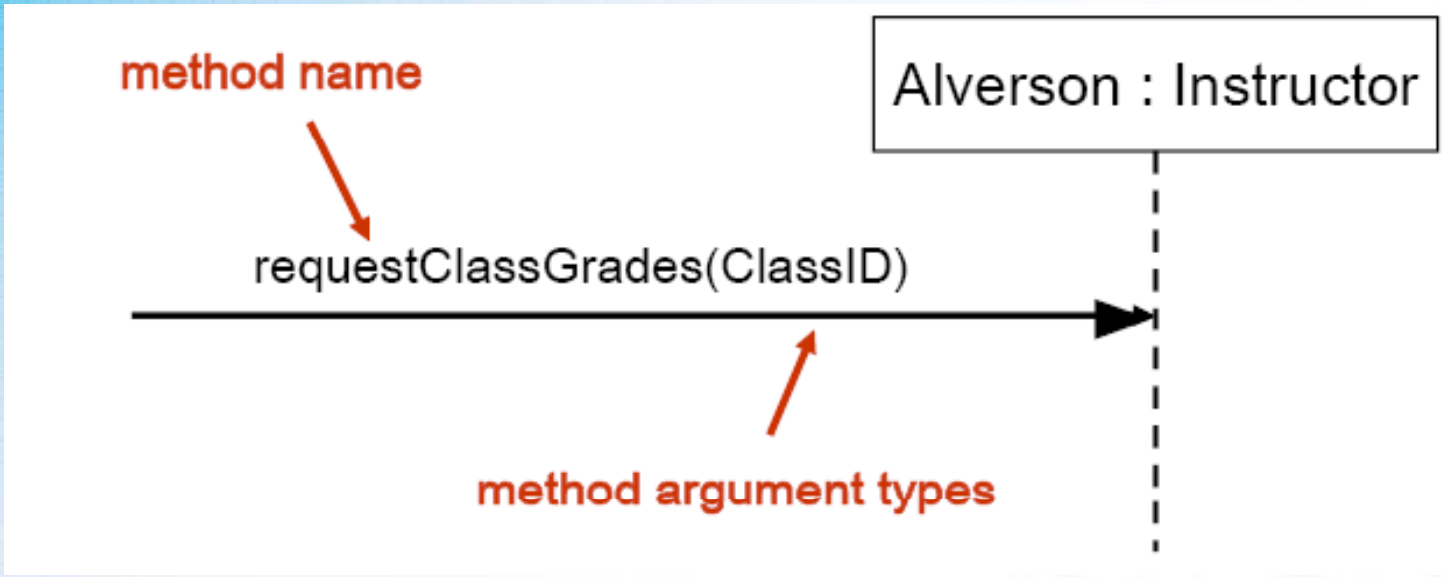
- 그리기 원하는 사용 사례의 프로세스/알고리즘 등을 잘 익힌다.
- 주요 객체를 찾아낸다.
- 제어 흐름과 메시지를 찾아낸다.

객체 나타내기


InstanceName : ClassName
object
anonymous object
object of
unknown class
Alverson : Instructor
: Instructor
Alverson
lifeline

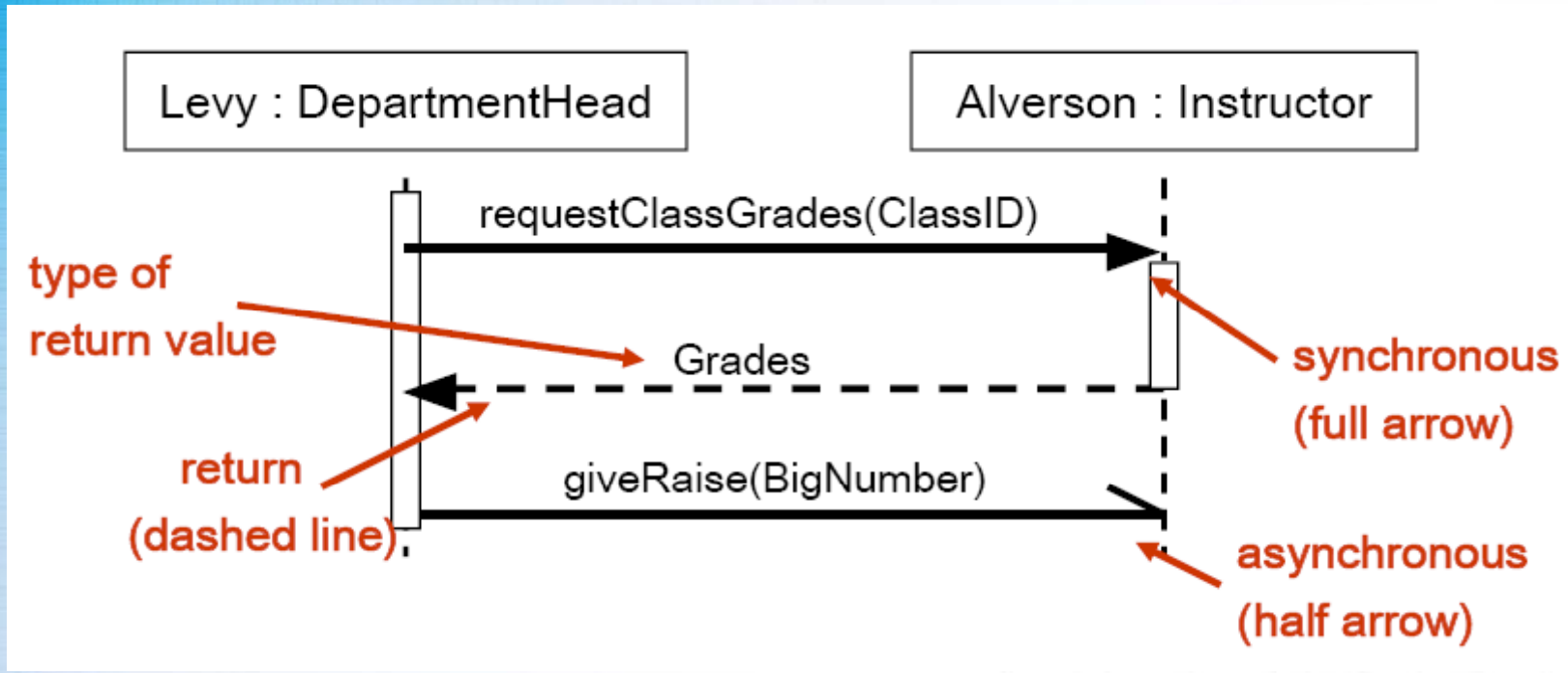
객체 사이의 메시지

- 호출한 메시지를 가진 객체에 수평화살표로 표시
  - 메시지 이름과 매개 변수를 화살표 위에 표시



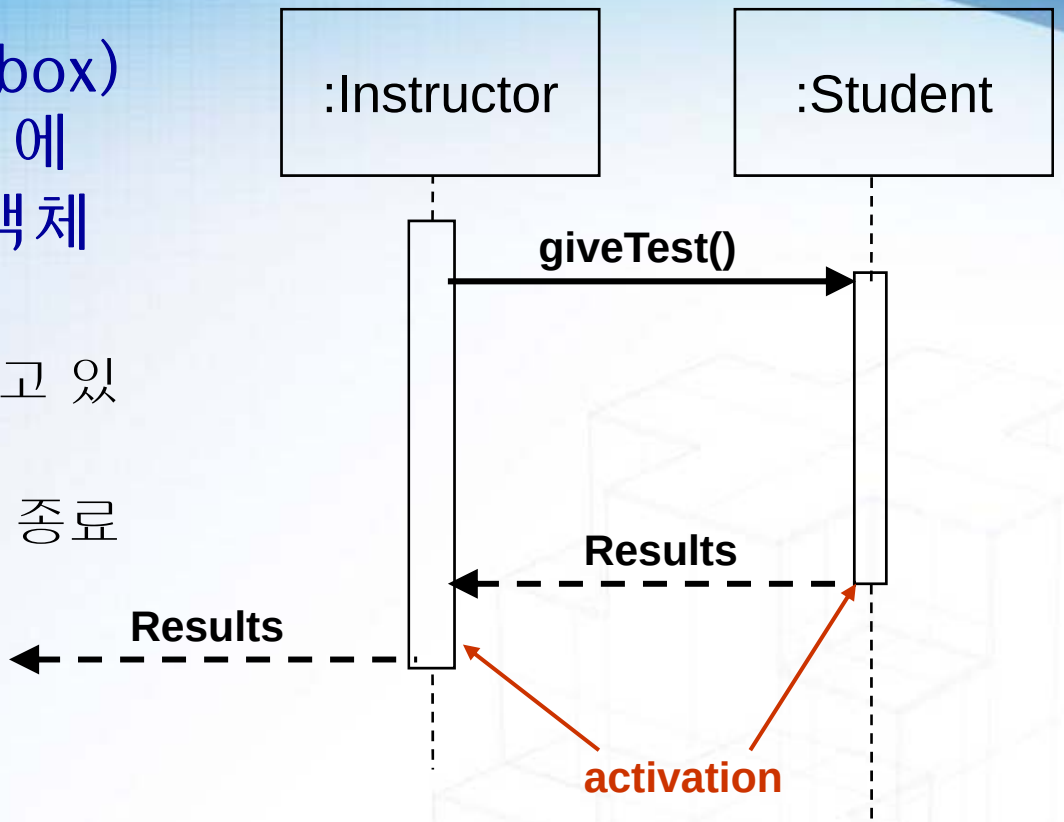
메시지

- 메시지는 수평 화살표로 표시됨
  - 점선 화살표는 리턴을 표시
  - 화살표 헤드의 모양으로 정상/ 비동기(asynchoronous) 표시



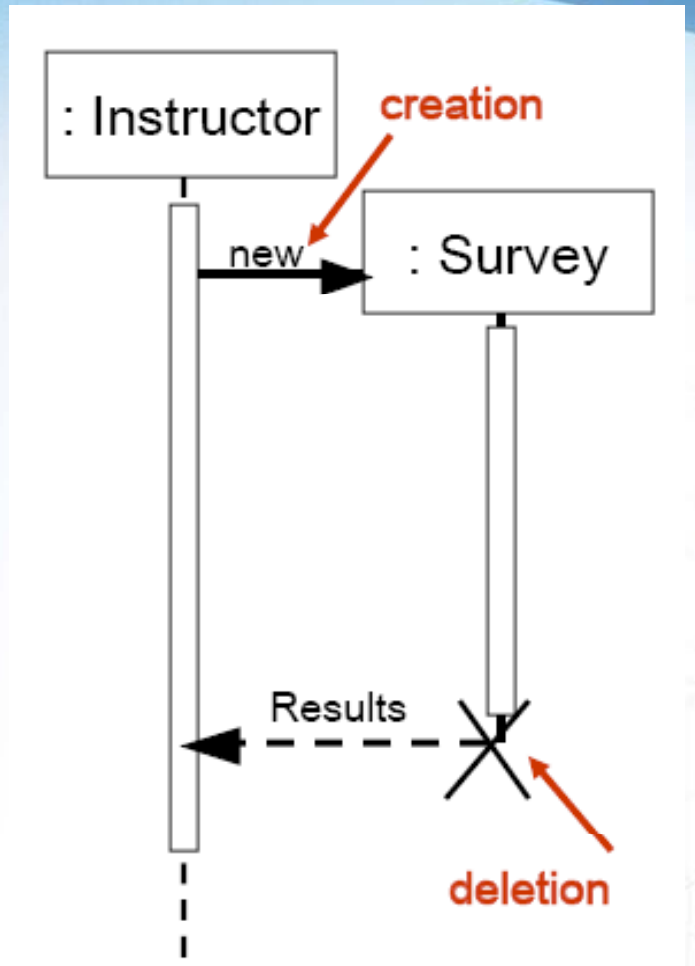
메시지 호출의 표시

- 활성 박스(activation box)
  - 객체 라이프 라인 위에 그려짐. 박스 위에서 객체의 호출이 이루어짐
    - 객체의 코드가 실행되고 있거나
    - 다른 객체의 메소드가 종료되기를 기다림.

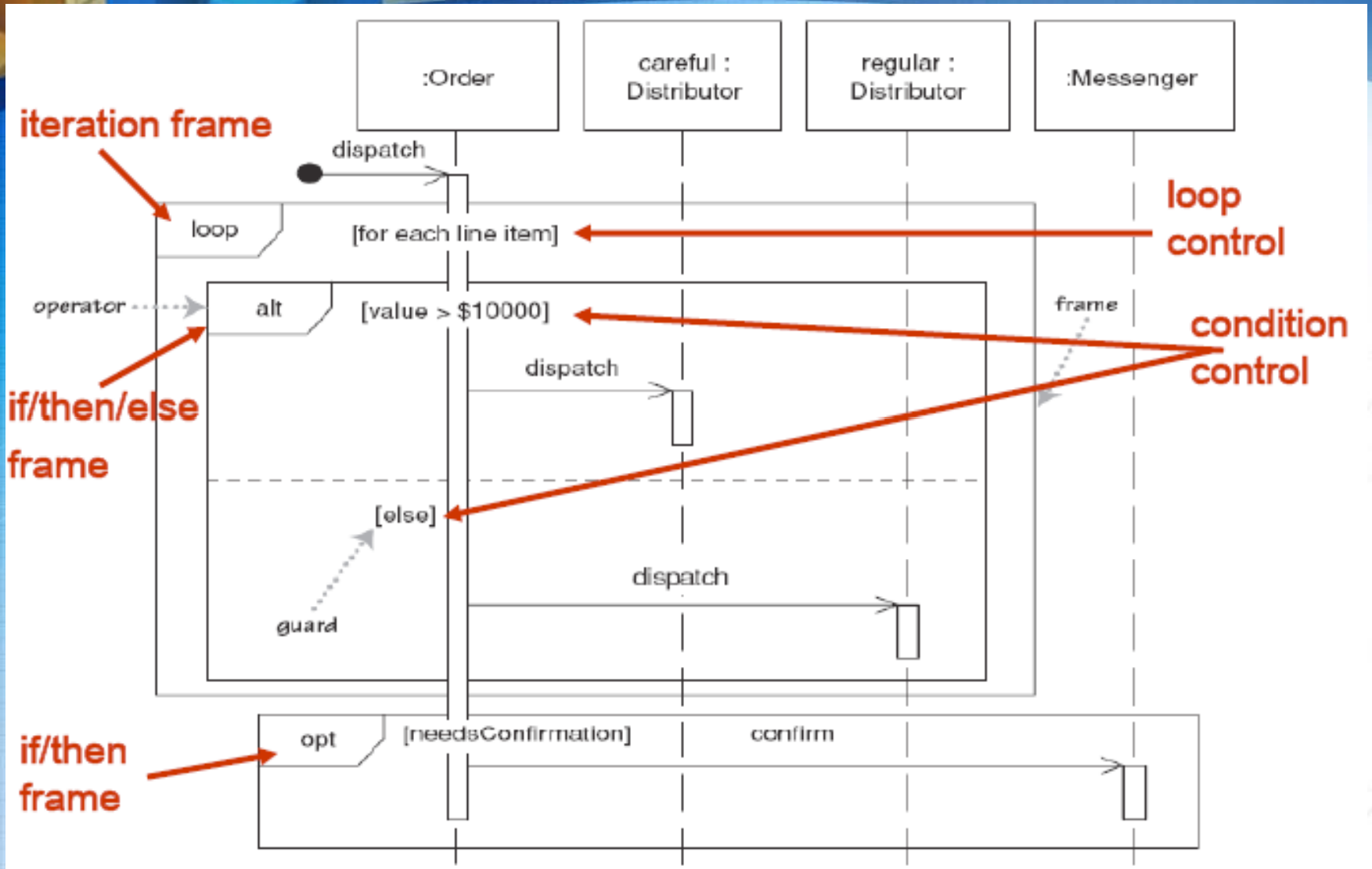


객체의 라이프 타임

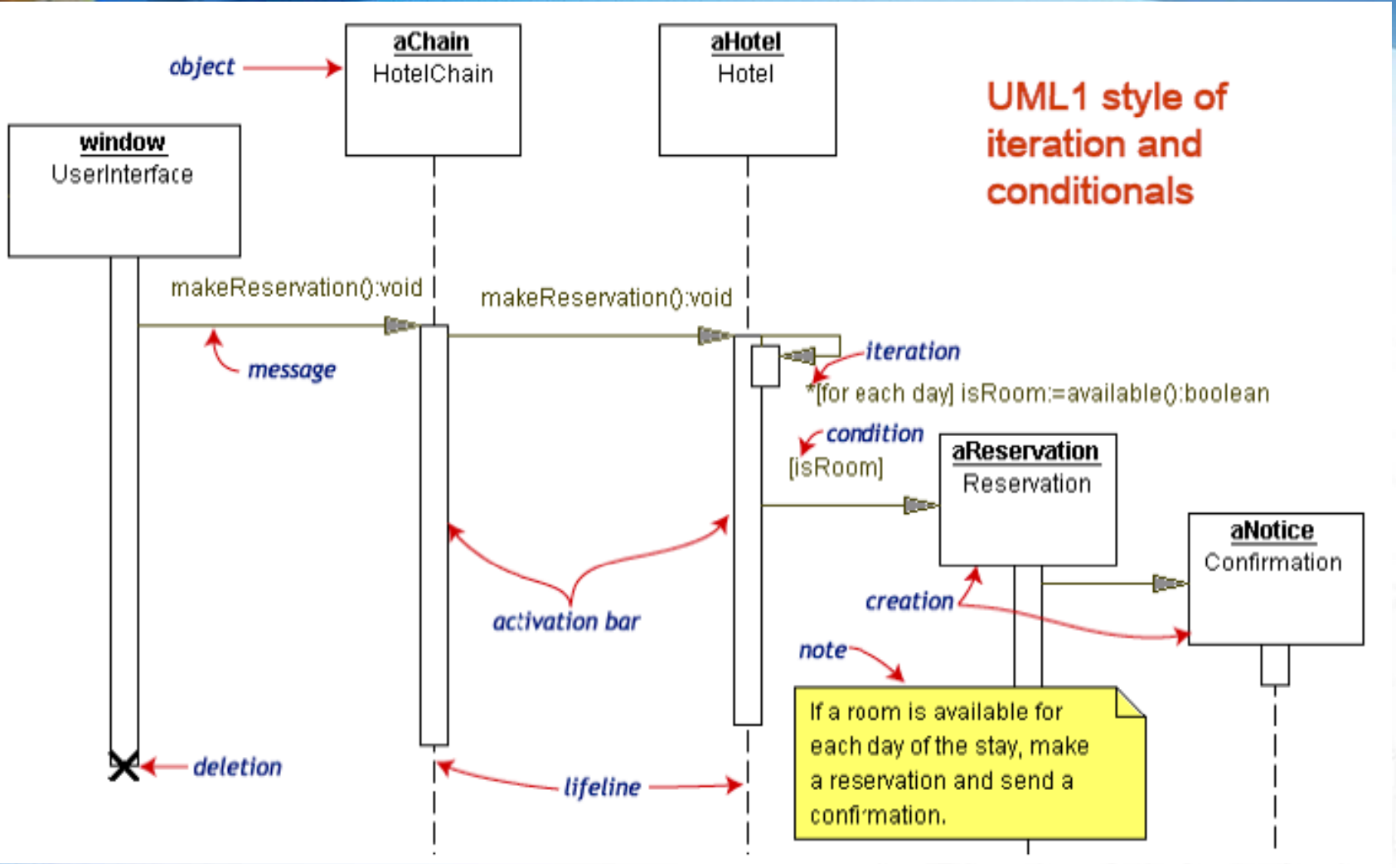
- 생성: 'new'라고 위에 쓴 화살표로 표시
  - 생성된 객체는 다른 객체보다 조금 아래 위치
- 삭제: 객체 라이프 라인의 끝에 X 표시
  - Java 언어/ C++ 언어에서 객체의 소멸은?



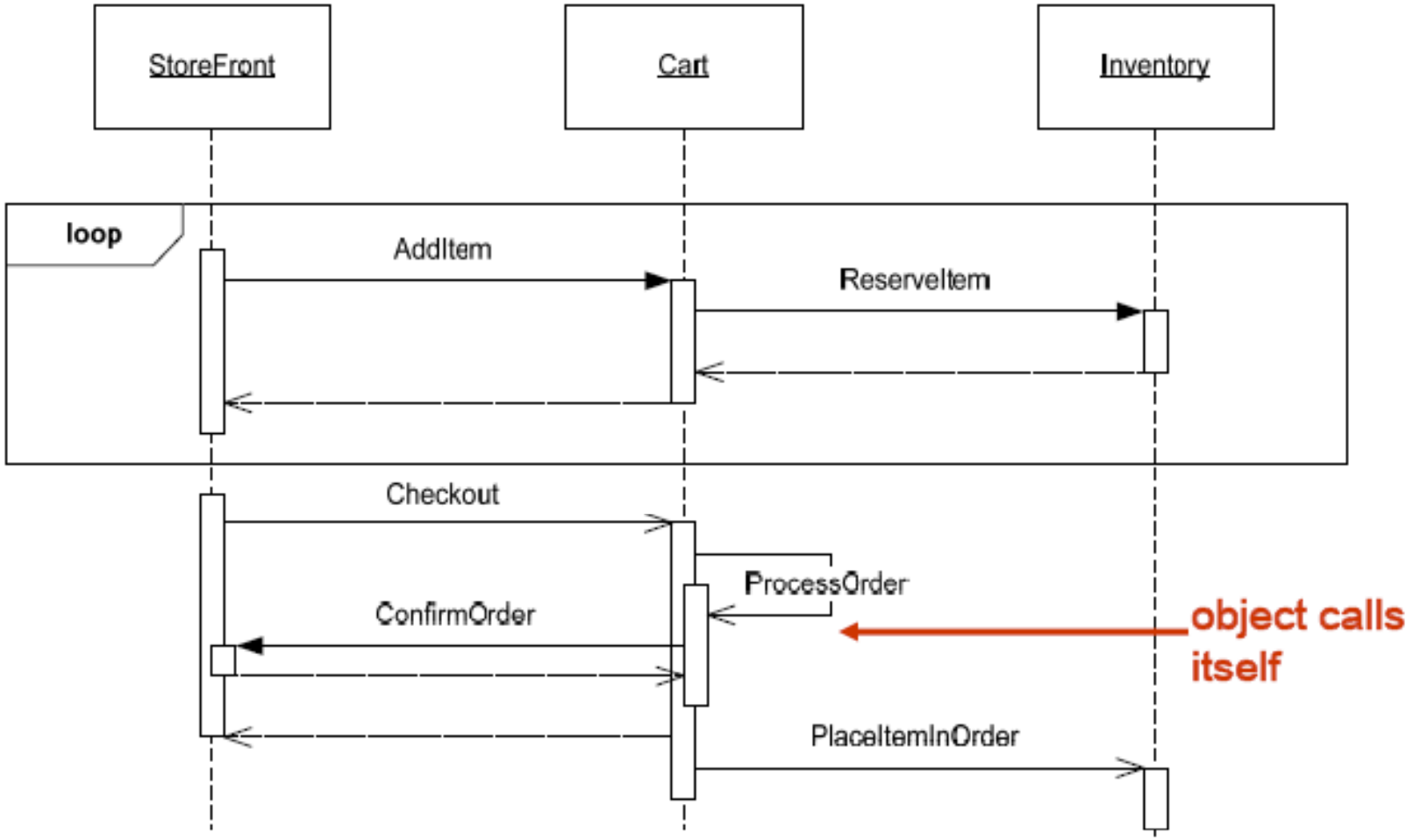
조건과 반복(UML 2.0)


:Order
careful :
Distributor
regular :
Distributor
:Messenger
iteration frame
dispatch
loop
[for each line item]
loop
control
operator
alt
[value > $10000]
frame
condition
control
dispatch
if/then/else
frame
[else]
dispatch
guard
if/then
frame
opt
[needsConfirmation]
confirm

순서 다이어그램 예


object
aChain
HotelChain
aHotel
Hotel
window
UserInterface
UML1 style of iteration and conditionals
makeReservation():void
makeReservation():void
message
iteration
*[for each day] isRoom:=available():boolean
condition
[isRoom]
aReservation
Reservation
aNotice
Confirmation
creation
activation bar
note
If a room is available for each day of the stay, make a reservation and send a confirmation.
deletion
lifeline

순서 다이어그램 예 #2


StoreFront
Cart
Inventory
loop
AddItem
ReserveItem
Checkout
ProcessOrder
ConfirmOrder
object calls itself
PlaceItemInOrder

Exercise #1: 순서 다이어그램 그리기

- 항공권을 예약하는 과정을 표현하는 순서 다이어그램을 그리시오.

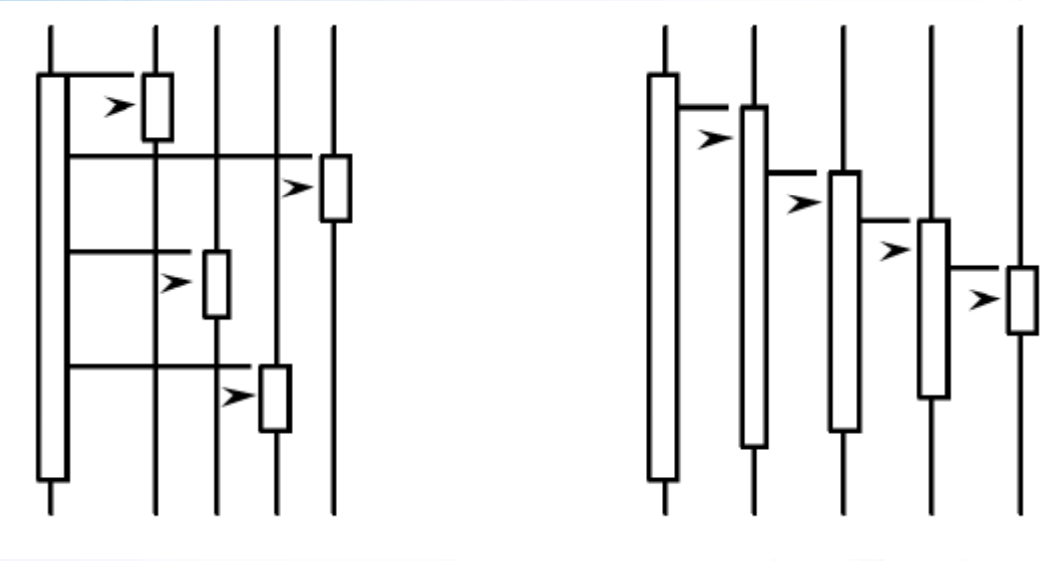
- 객체: Customer, Flight, Reservation

Exercise #2: 순서 다이어그램 그리기

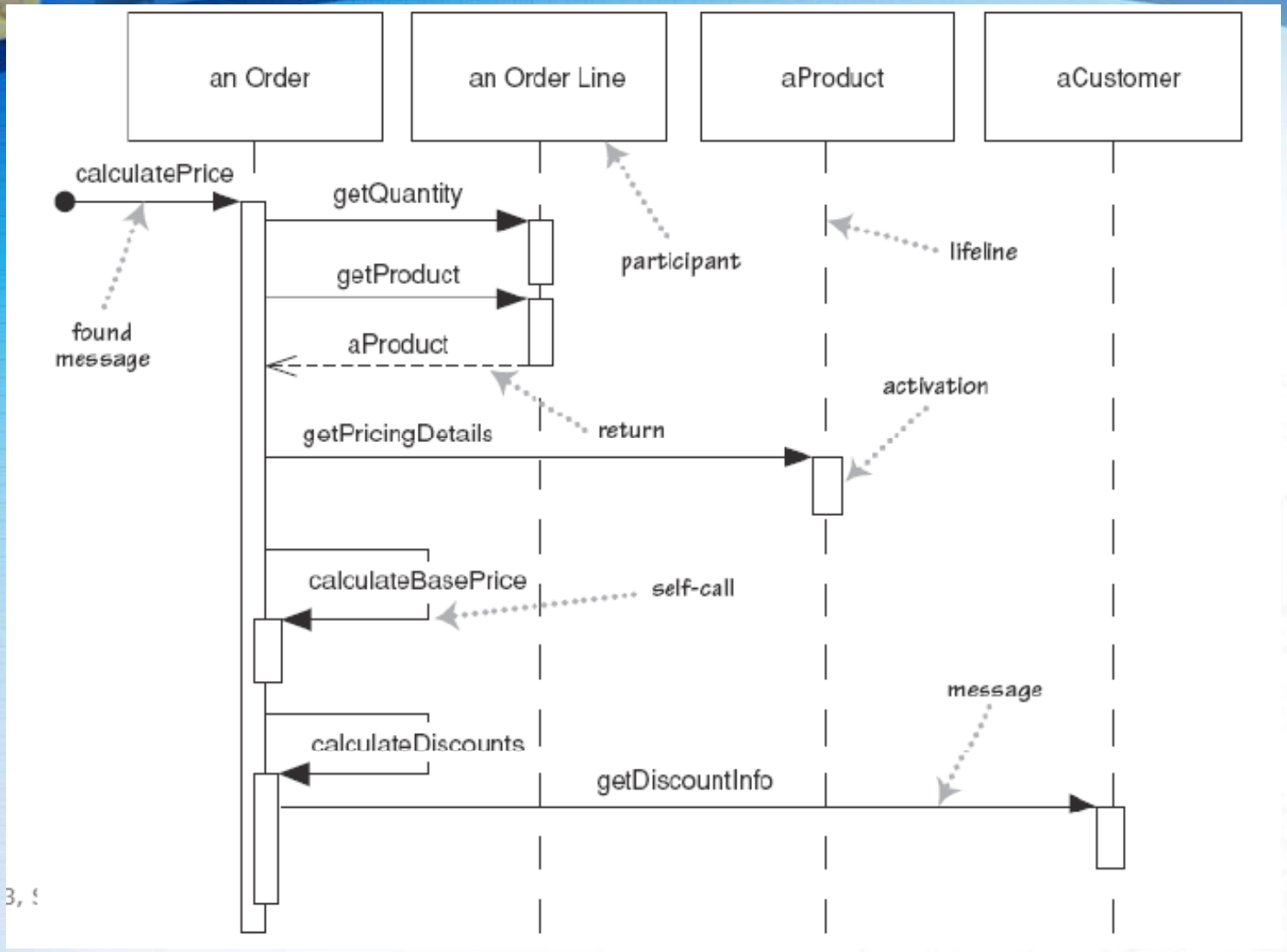
- 자동판매기의 순서 다이어그램을 그려라
- 동전을 넣고 음료수를 구매하는 과정
  - 정상 흐름
  - 확장 흐름
- 객체
  - CoinSlot, Button, Controller, Timer

시스템의 콘트롤 형태

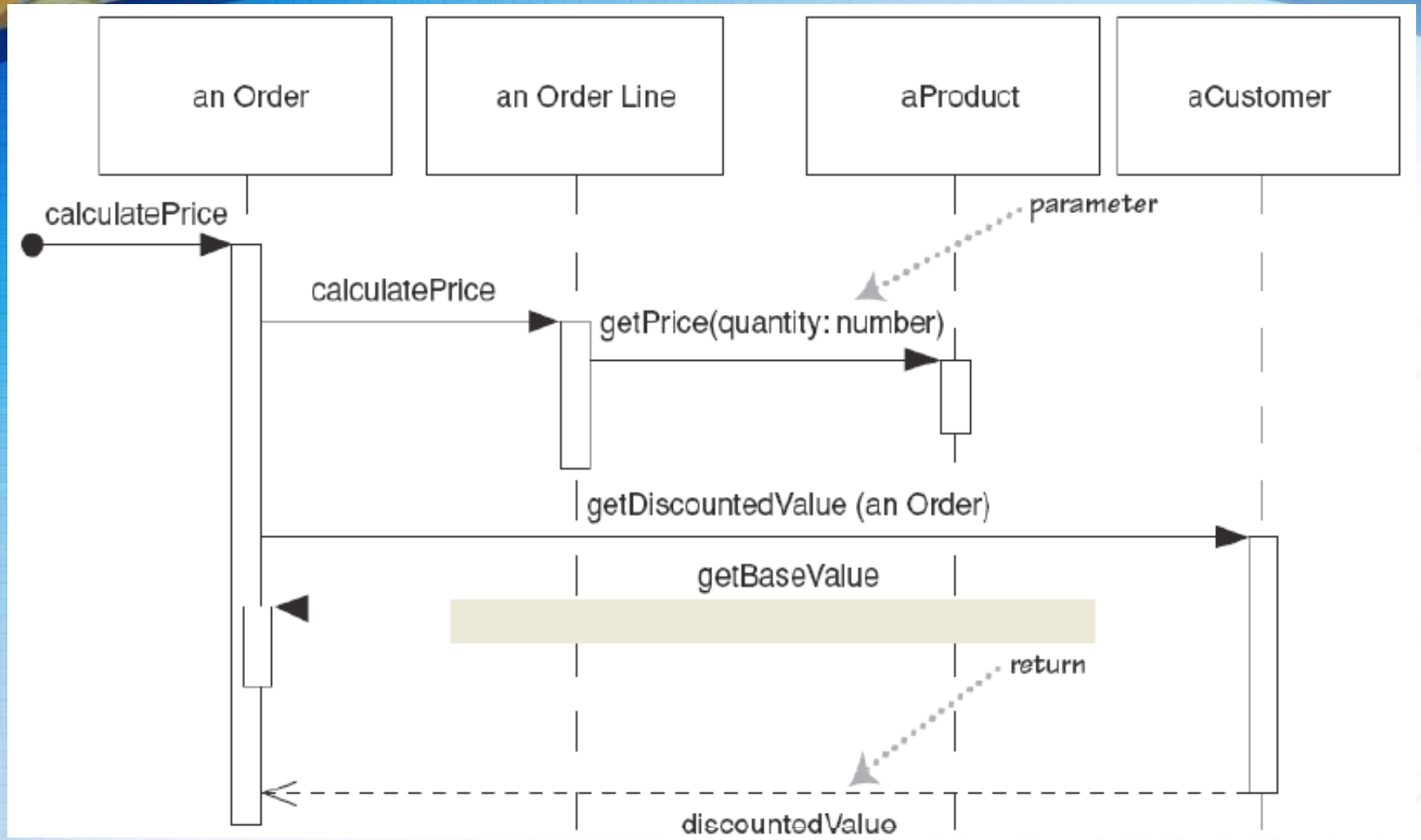
- 다음 시스템의 제어 흐름은 어떤 형태인가?
  - 중앙 집중형
  - 분산형
  - 순서 다이어그램은 이런 것을 보여주는데 도움이 되는가?



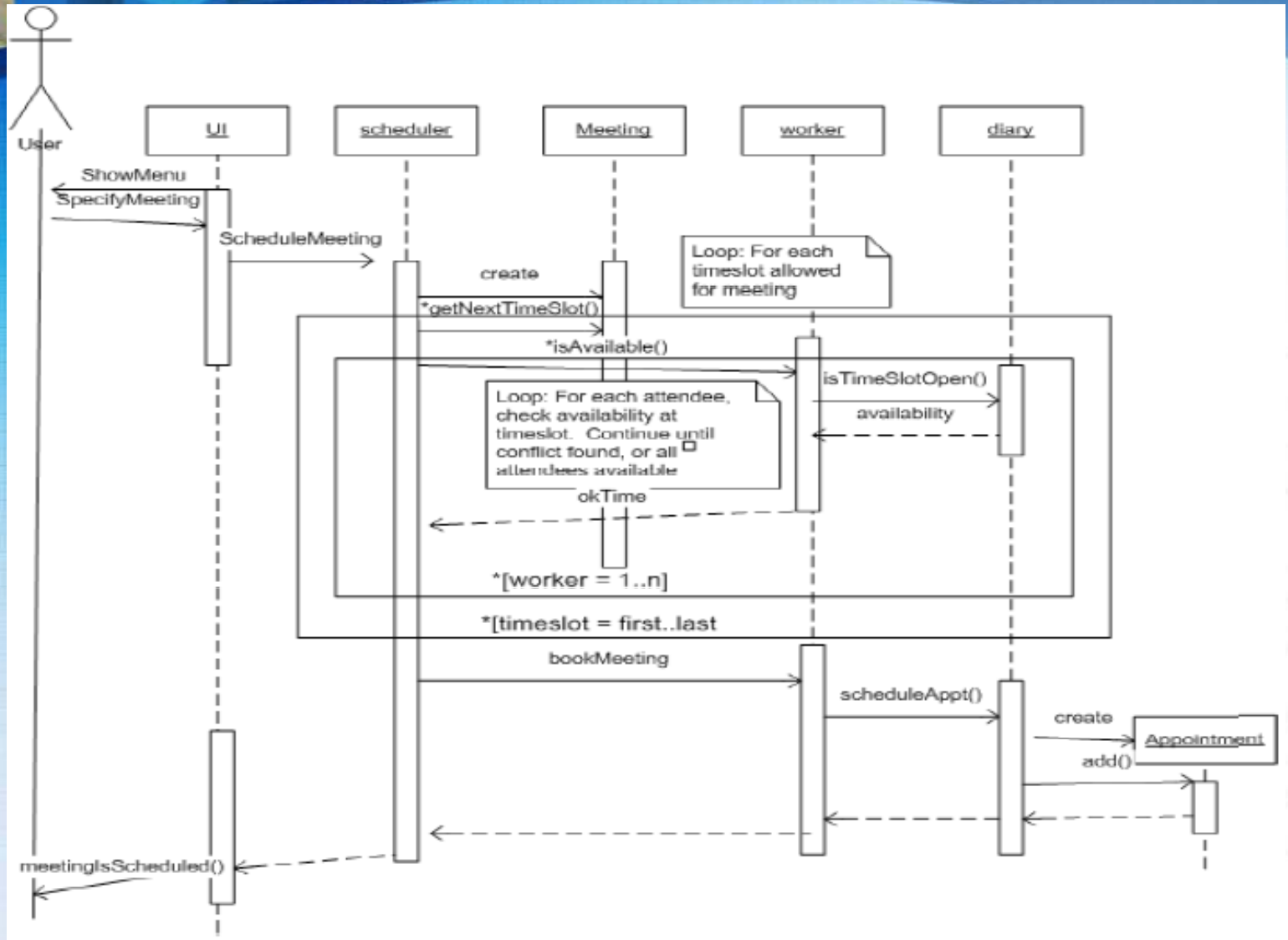
다음 다이어그램의 제어 패턴은?



이 다이어그램의 제어 패턴은?

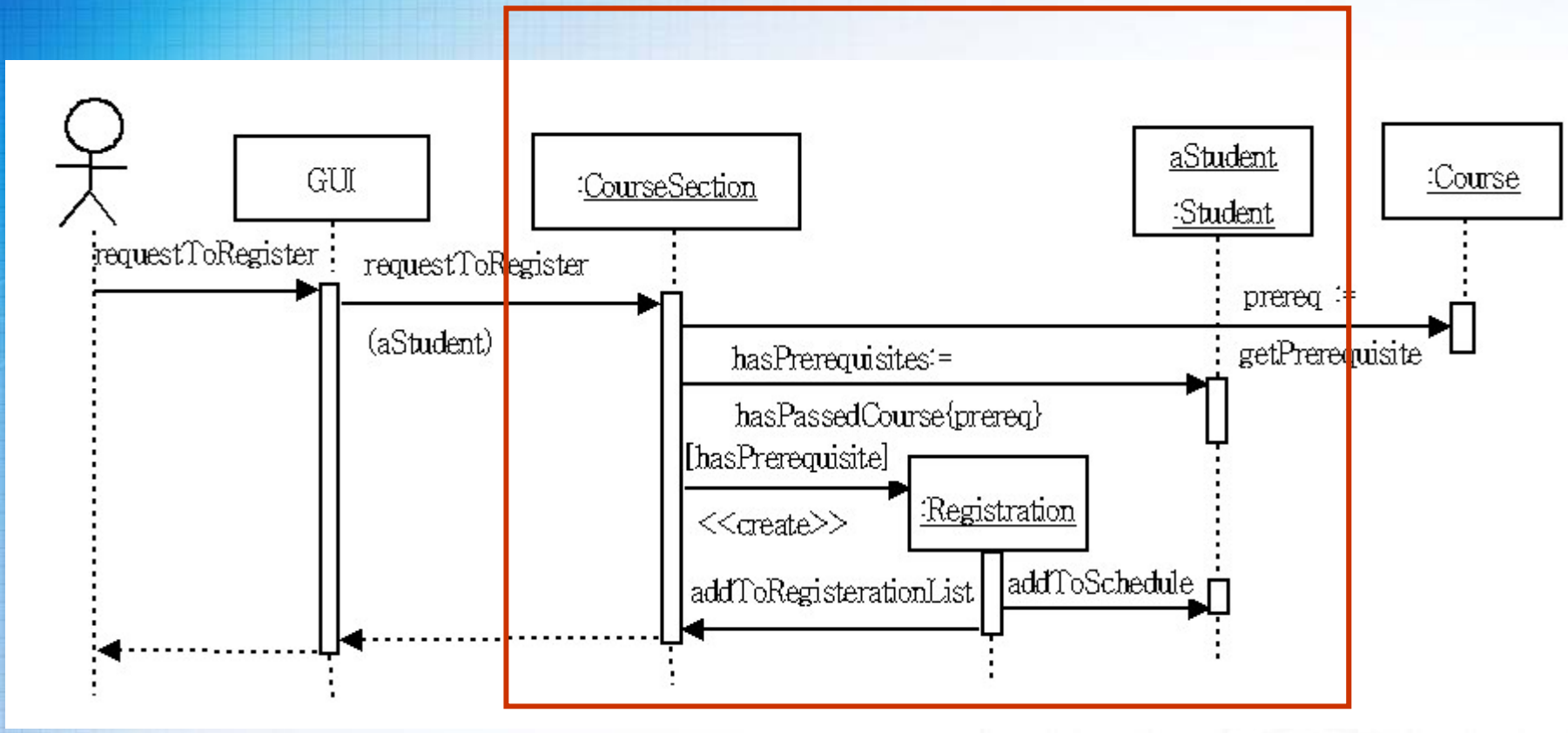


다음 순서 다이어그램은?


User
UI
scheduler
Meeting
worker
diary
ShowMenu
SpecifyMeeting
ScheduleMeeting
create
*getNextTimeSlot()
Loop: For each timeslot allowed for meeting
*isAvailable()
isTimeSlotOpen()
availability
Loop: For each attendee, check availability at timeslot. Continue until conflict found, or all
okTime
*[worker = 1..n]
*[timeslot = first..last
bookMeeting
scheduleAppt()
create
Appointment
add()
meetingIsScheduled()

순서 다이어그램과 코딩

❑ 수강 과목 신청



순서 다이어그램과 코딩

```
public class CourseSection
{
  // The many-1 abstraction-occurence association
  private Course course;

  // The 1-many association to class Registration
  private List registationList;

  // The following are present only to determine the state
  // The initial state is 'Planned'
  private boolean open = false;
  private boolean closedOrCancelled = false;
 ...
```

```
public void requestToRegister(Student student)
{
  if (open)  // must be in one of the two 'Open' states
    {
      // The interaction specified in the sequence diagram
      Course prereq = course.getPrerequisite();
      if (student.hasPassedCourse(prereq))
      {
         // Indirectly calls addToRegistrationList
         new Registration(this, student);
      }

      // Check for automatic transition to 'Closed' state
      if (registrationList.size() >= course.getMaximum())
      {
        // to ‘Closed’ state
        open = false;
        closedOrCancelled = true;
      }
    }
  }
}
```

왜 바로 코딩하지 않는가?

- 순서 다이어그램은 코드와 매우 밀접하다. 다이어그램을 그리기 전에 왜 바로 코딩하지 않을까?

  - 좋은 순서 다이어그램은 추상적 가치를 가짐
  - 순서 다이어그램은 언어 효과를 노린 것
  - 개발자가 아닌 사람도 작성 가능
  - 여러 객체를 한 페이지에서 볼 수 있어
    - 이해가 쉬움
    - 리뷰가 용이
  - 좋은 커뮤니케이션 도구

UML 정리

- UML은 이래서 좋다.
  - 공통 언어
    - 요구, 명세, 설계를 공유할 수 있게 한다
  - 비주얼 구문이 좋다
    - 정보를 요약
    - 개발자/기술자가 아닌 사람들에게도 이해 가능
  - 도구 지원
    - Visio, Rational, Eclipse, Together
    - 어떤 도구는 UML 에서 코드로 자동 변환