

# 소프트웨어 공학 개론

## 강의 7: 시퀀스 다이어그램

# 최은만

## 동국대학교 컴퓨터공학과



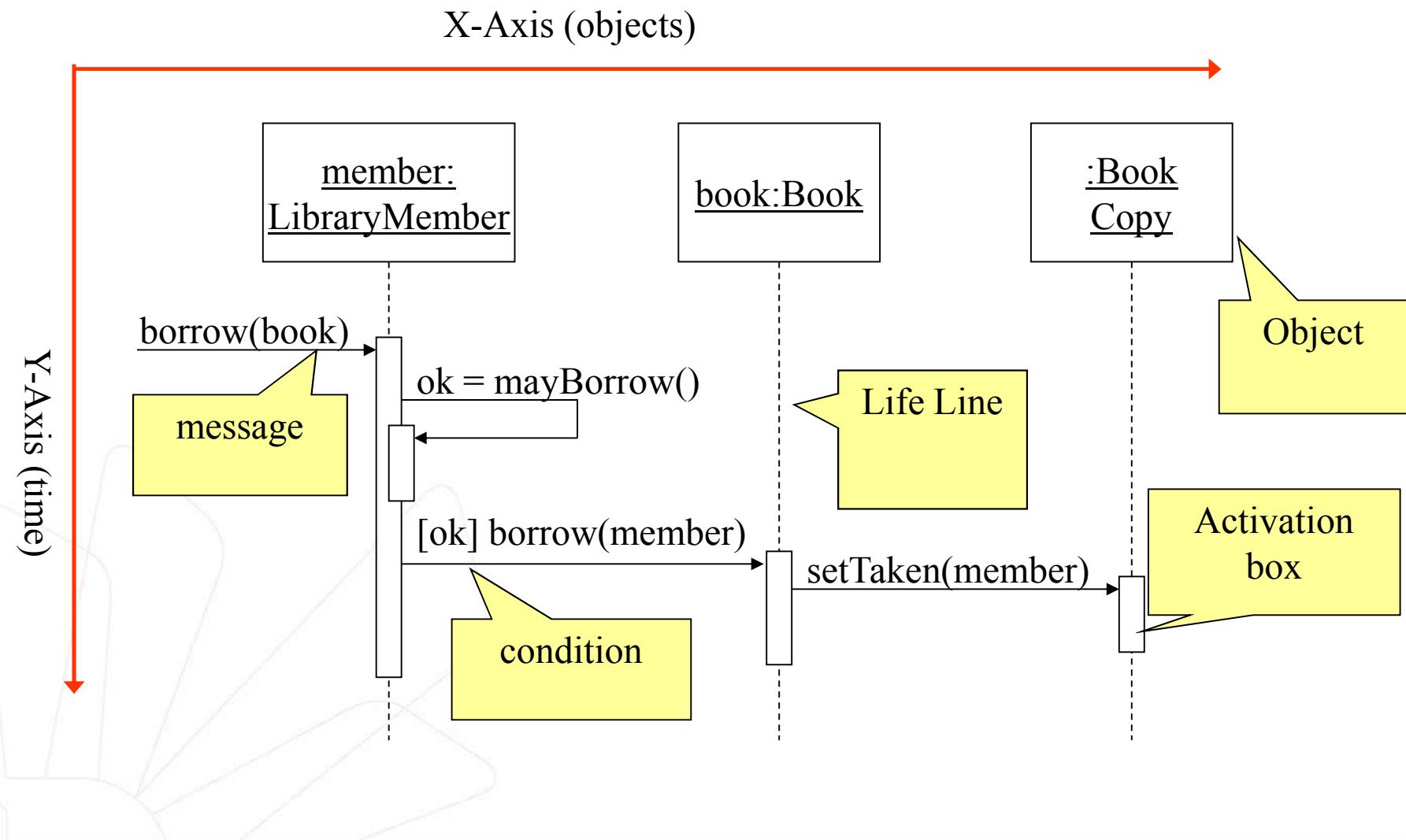
새로 쓴 소프트웨어 공학

*New Software Engineering*

# UML 시퀀스 다이어그램

- 시퀀스 다이어그램(Sequence Diagram)
  - 사용 사례가 어떻게 수행되는지 어떤 **메시지**가 언제 보내지는지 나타낸 그림
- 시스템의 동적인 측면을 캡처한 것
  - **동적 뷰(dynamic view)**
- **시간의 흐름**에 따라 정리해 놓은 것
  - 페이지 내려갈 수록 시간이 흐름
- 객체는 왼쪽에서 오른쪽으로 나열
  - 메시지 호출의 흐름에 언제 참여하였느냐에 따라

# 시퀀스 다이어그램의 요소



# 시퀀스 다이어그램 작성

- Step 1. 참여하는 객체를 파악
- Step 2. 파악한 객체를 X축에 나열하고 라이프라인을 그음
- Step 3. 사용사례에 기술된 이벤트 순서에 따라 객체의 메시지 호출

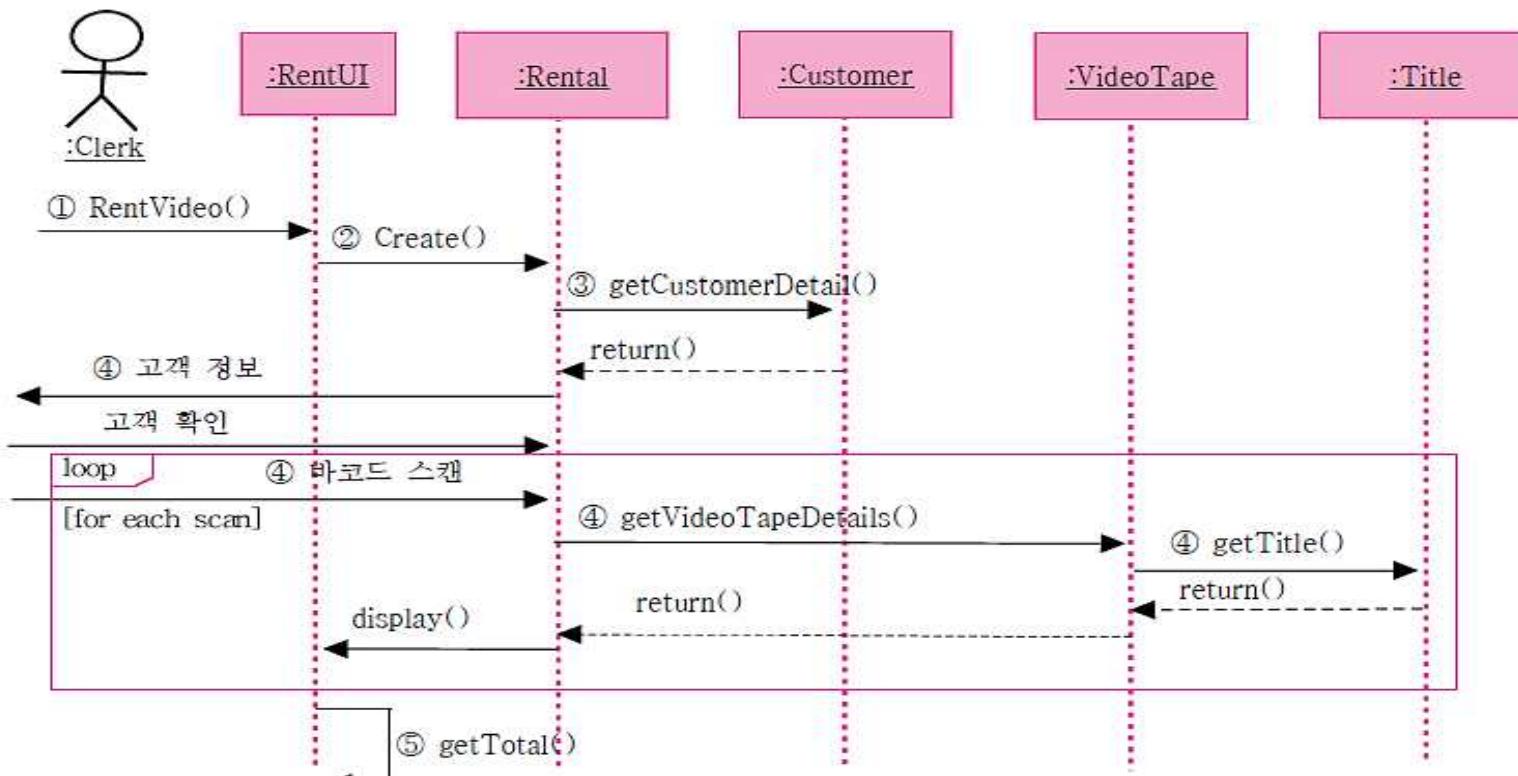
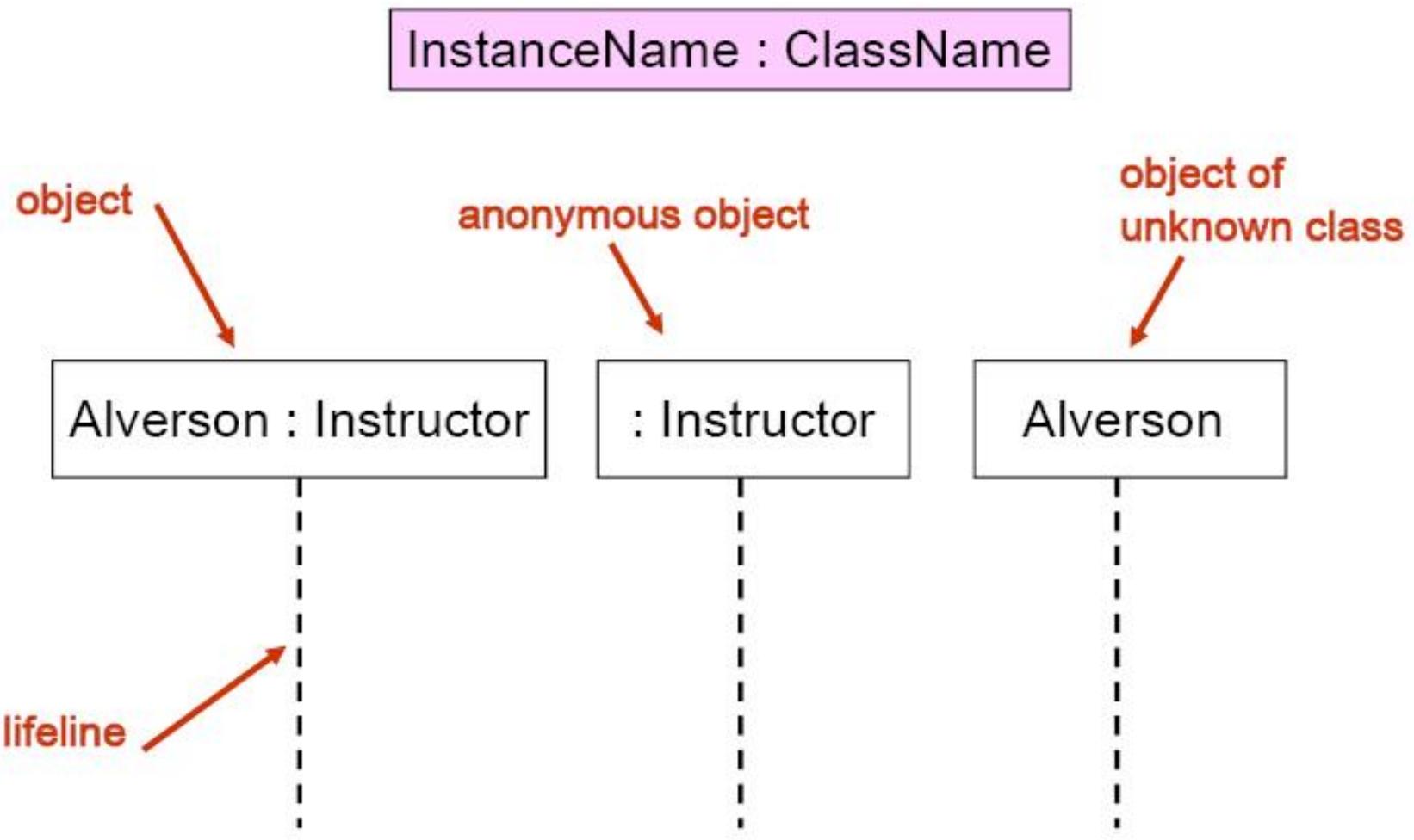


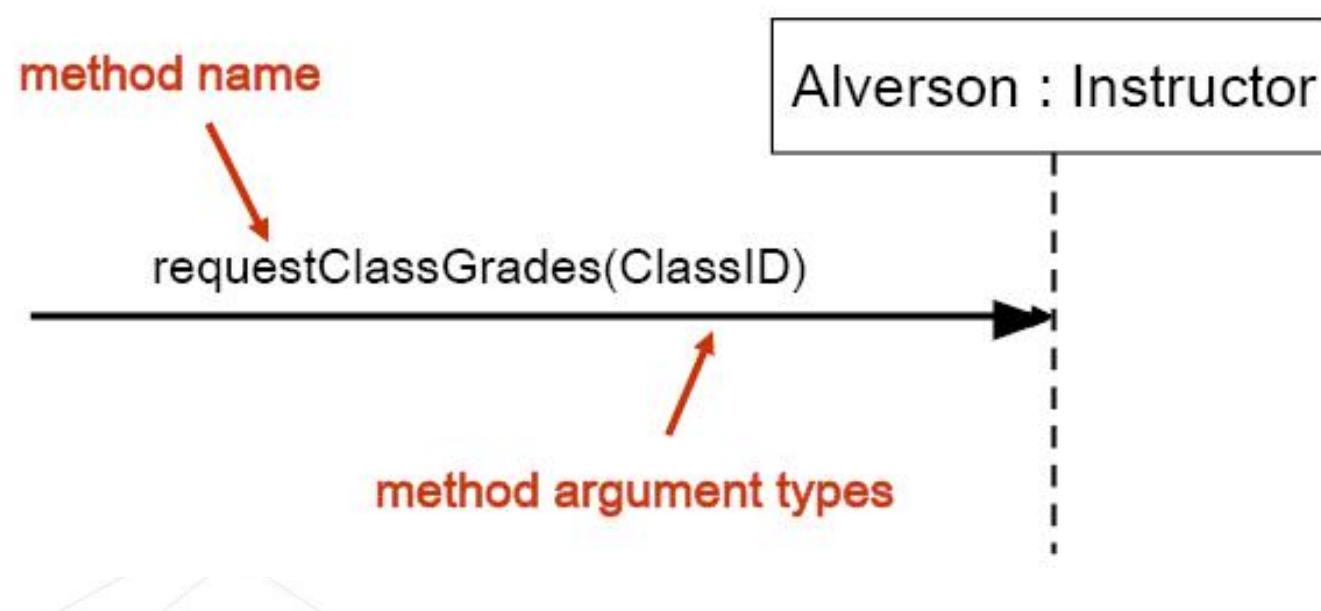
그림 5.32 ▶ 시퀀스 다이어그램(비디오 대여 사용 사례)

# 객체 나타내기



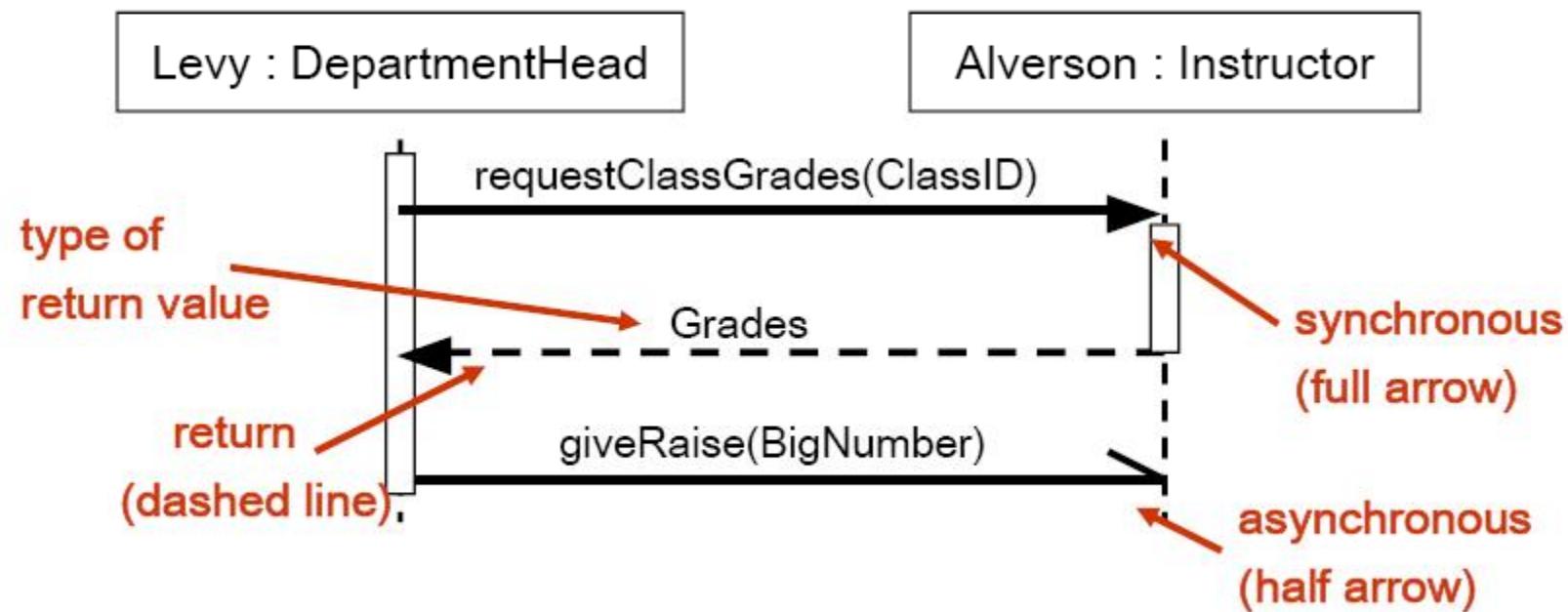
# 객체 사이의 메시지

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



# 메시지

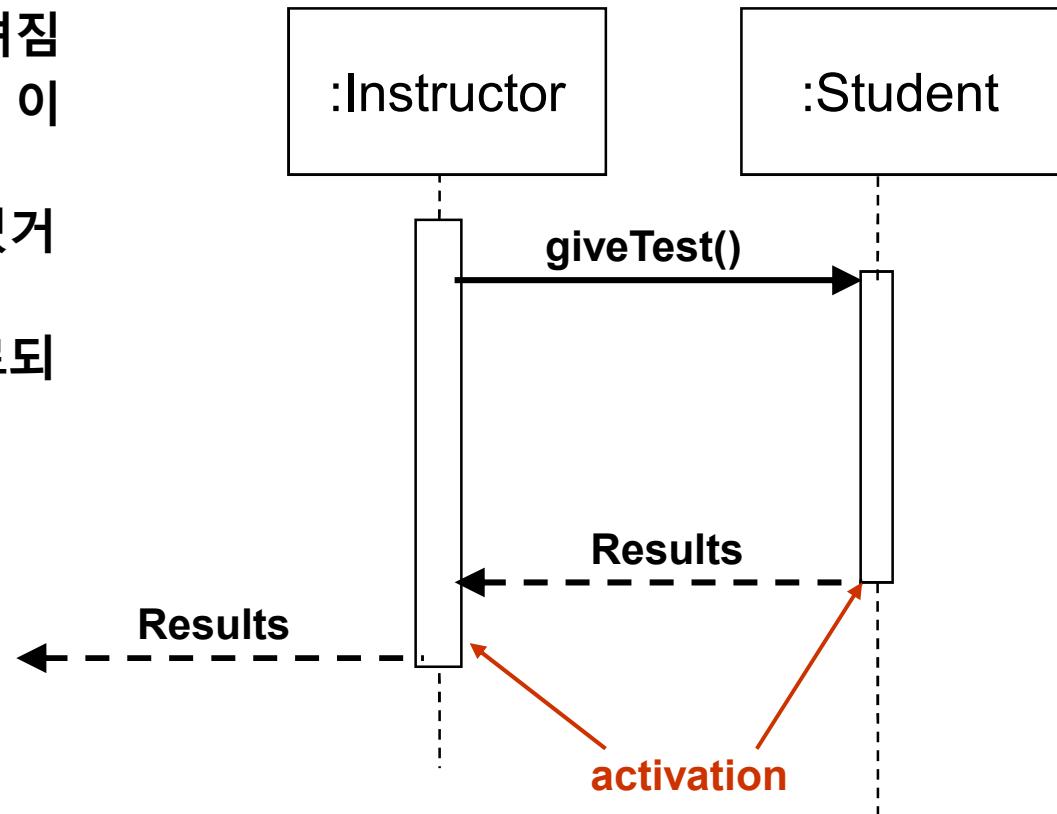
- 메시지는 수평 화살표로 표시됨
  - 점선 화살표는 리턴을 표시
  - 화살표 헤드의 모양으로 정상/ 비동기(asynchronous) 표시



# 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

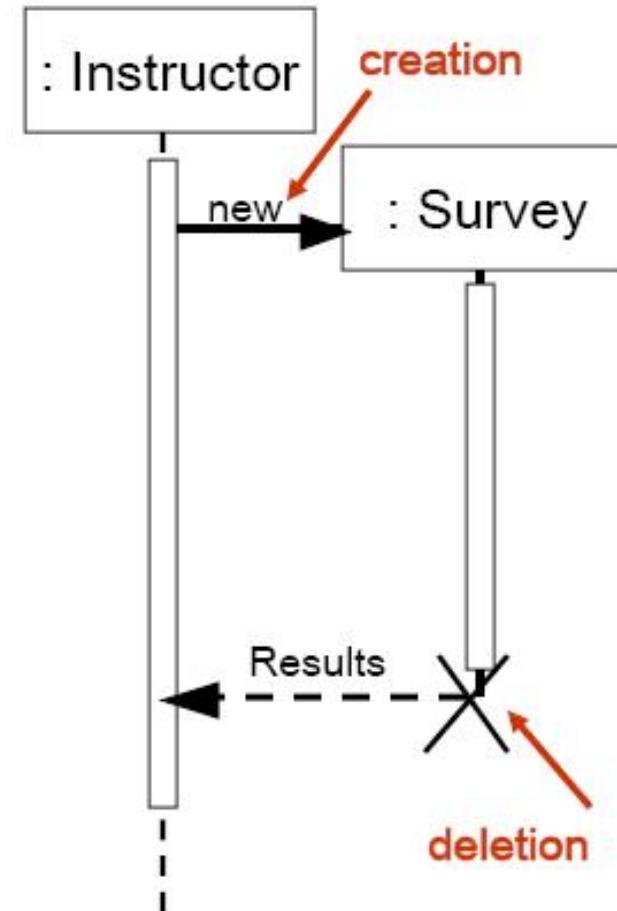
## • 생성

- 'new'라고 위에 쓴 화살표로 표시
- 생성된 객체는 다른 객체보다 조금 아래 위치

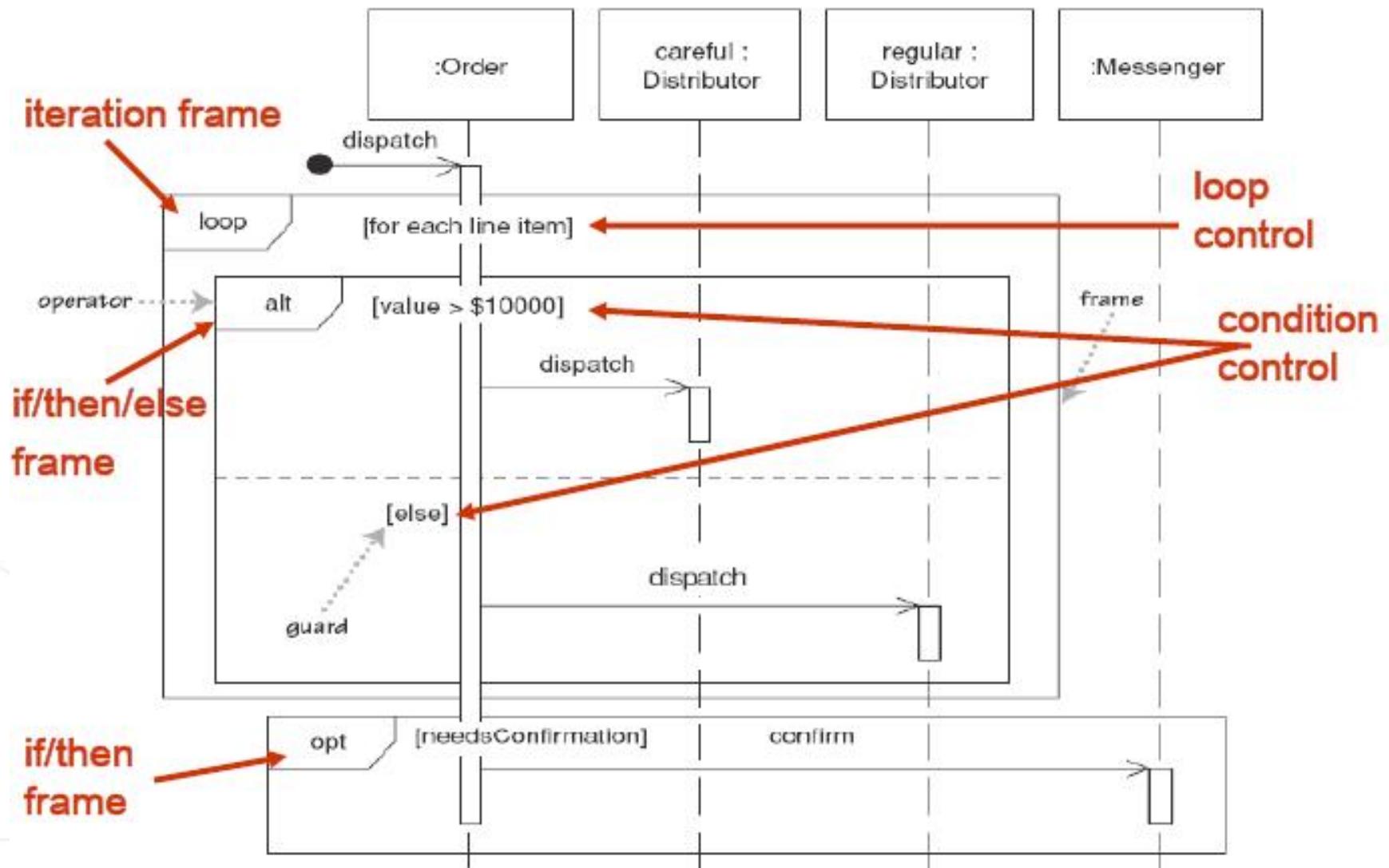
## • 삭제

- 객체 라이프 라인의 끝에 X 표시

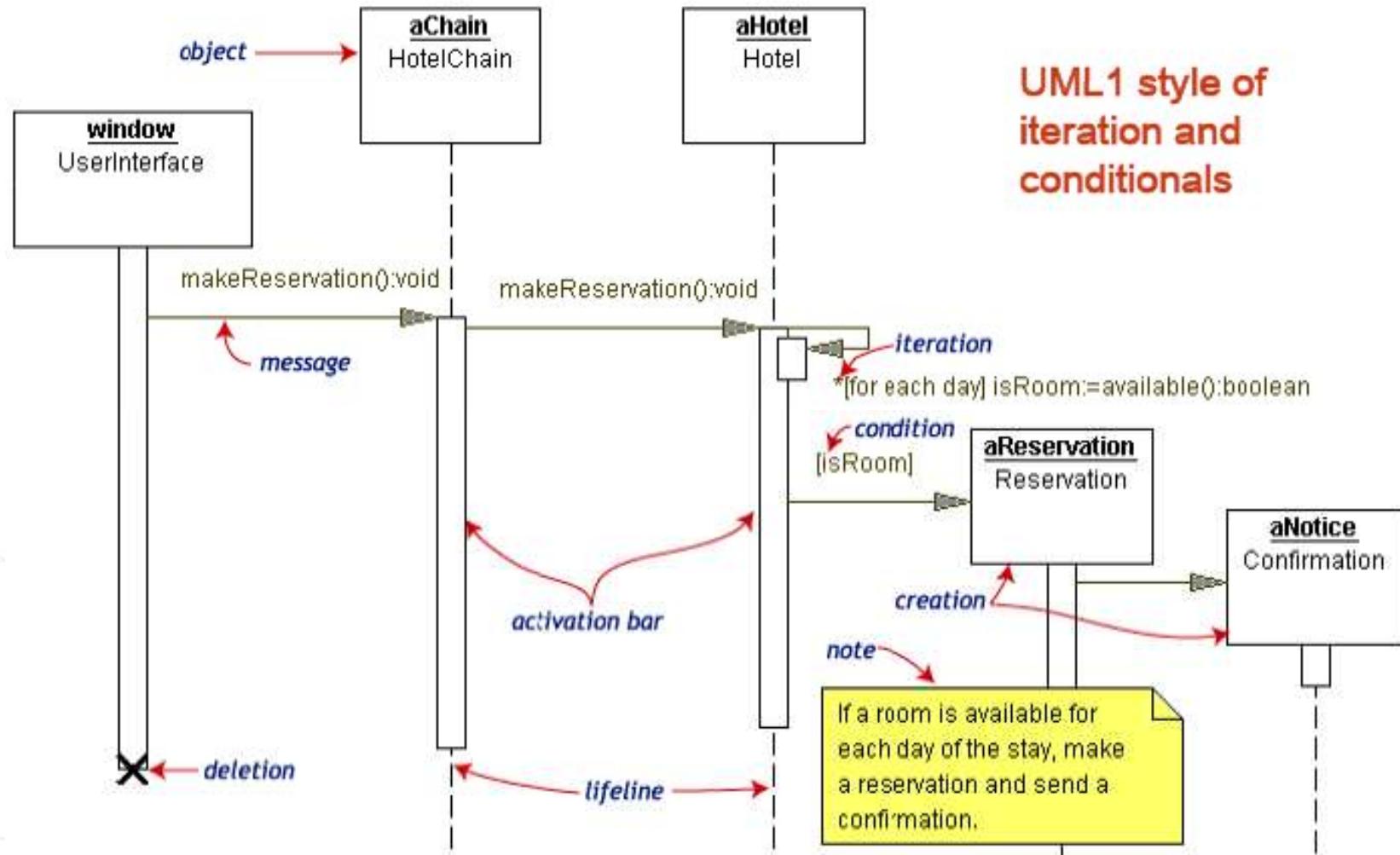
## • Java 언어/ C++ 언어에서 객체의 소멸은?



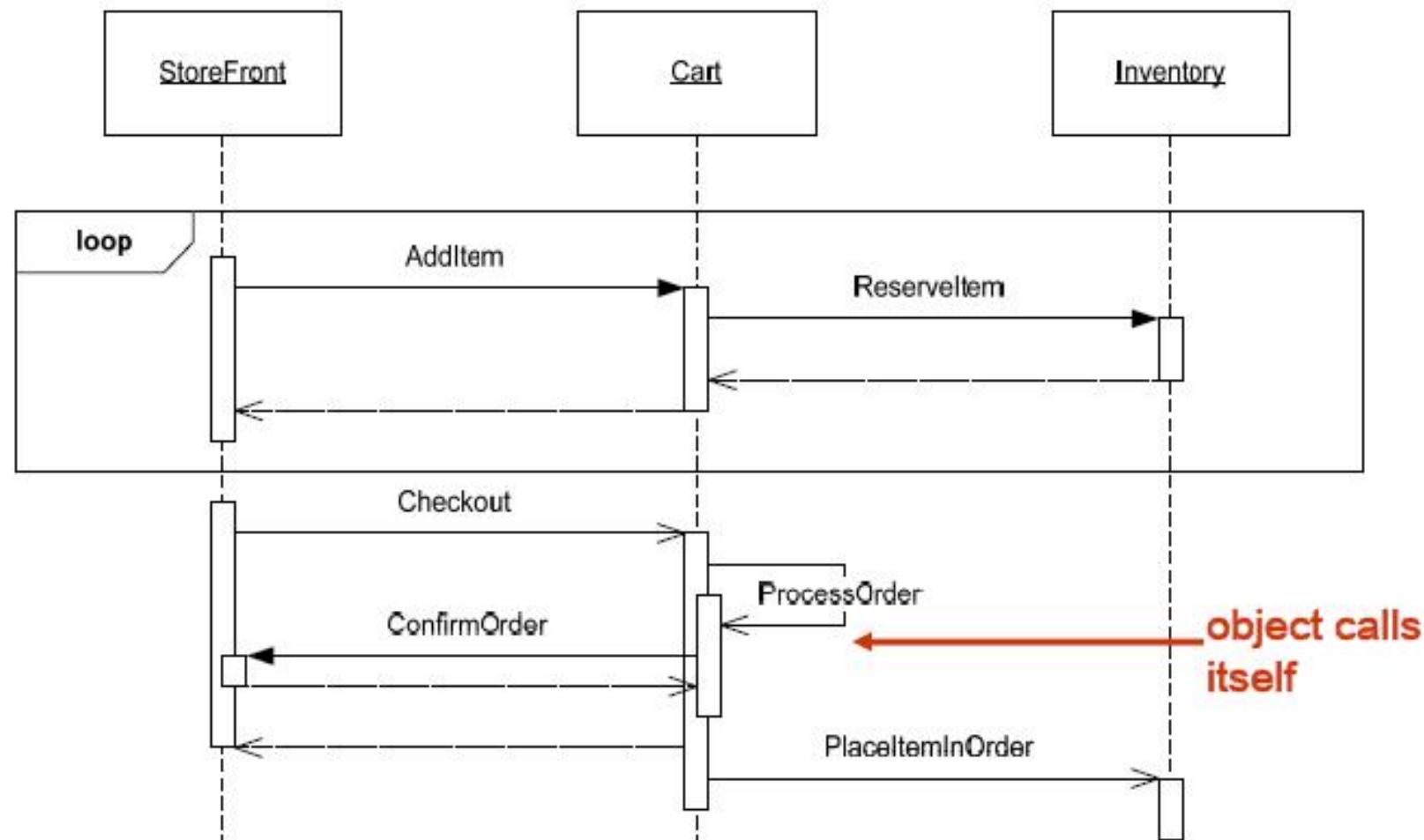
# 조건과 반복(UML 2.0)



# 시퀀스 다이어그램 사례 #1

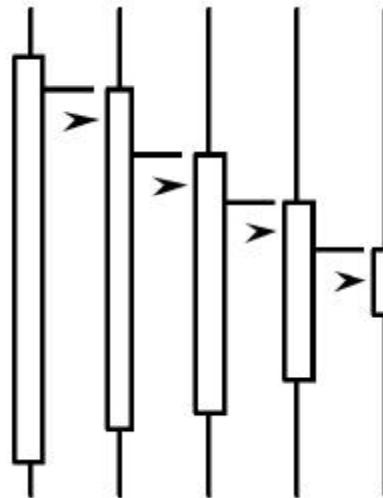
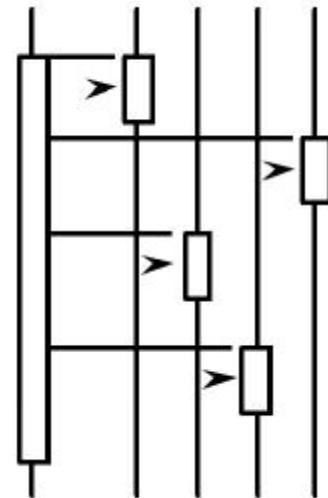


# 시퀀스 다이어그램 사례 #2



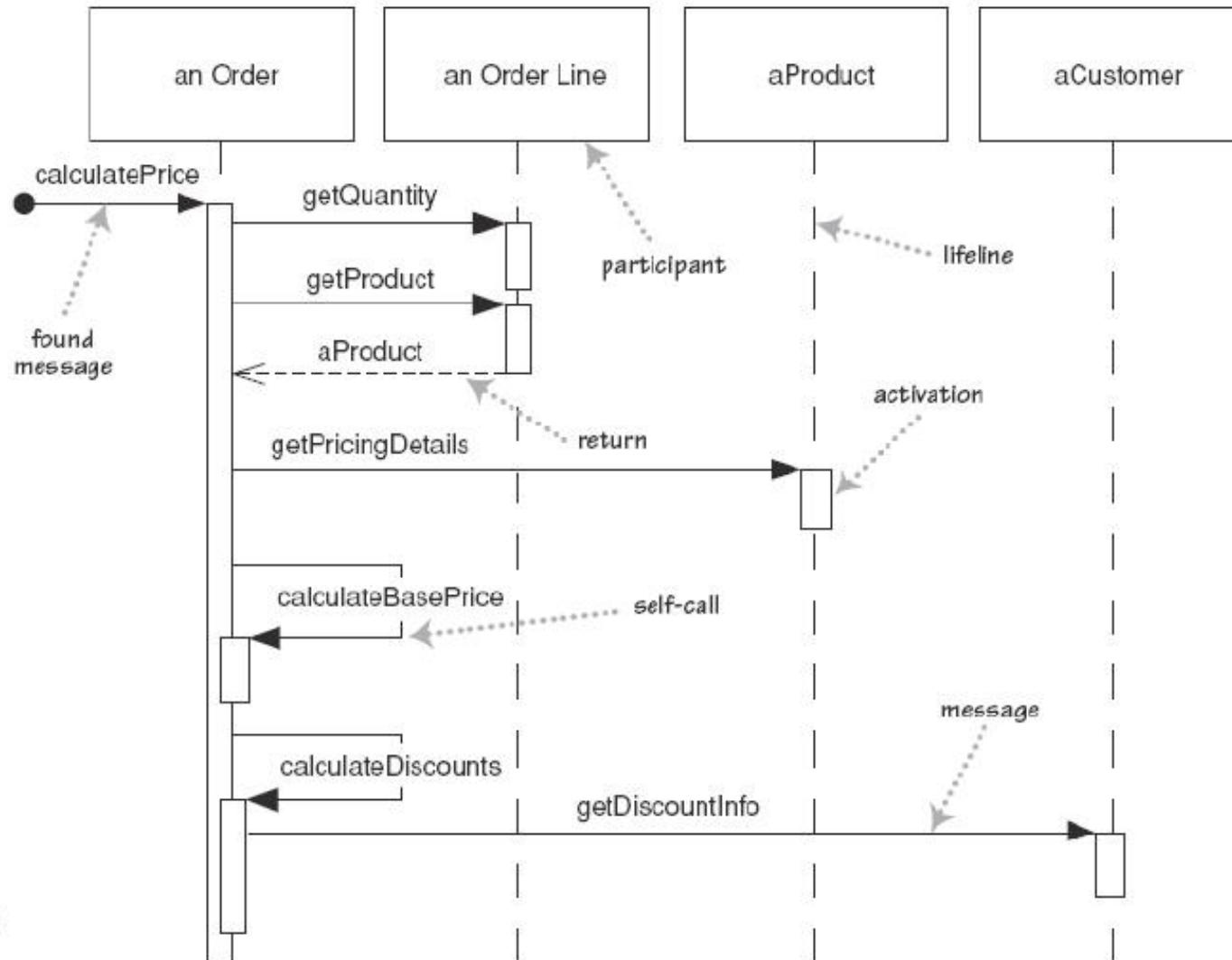
# 시스템의 콘트롤 형태

- 다음 시스템의 제어 흐름은 어떤 형태인가?
  - 중앙 집중형
  - 분산형

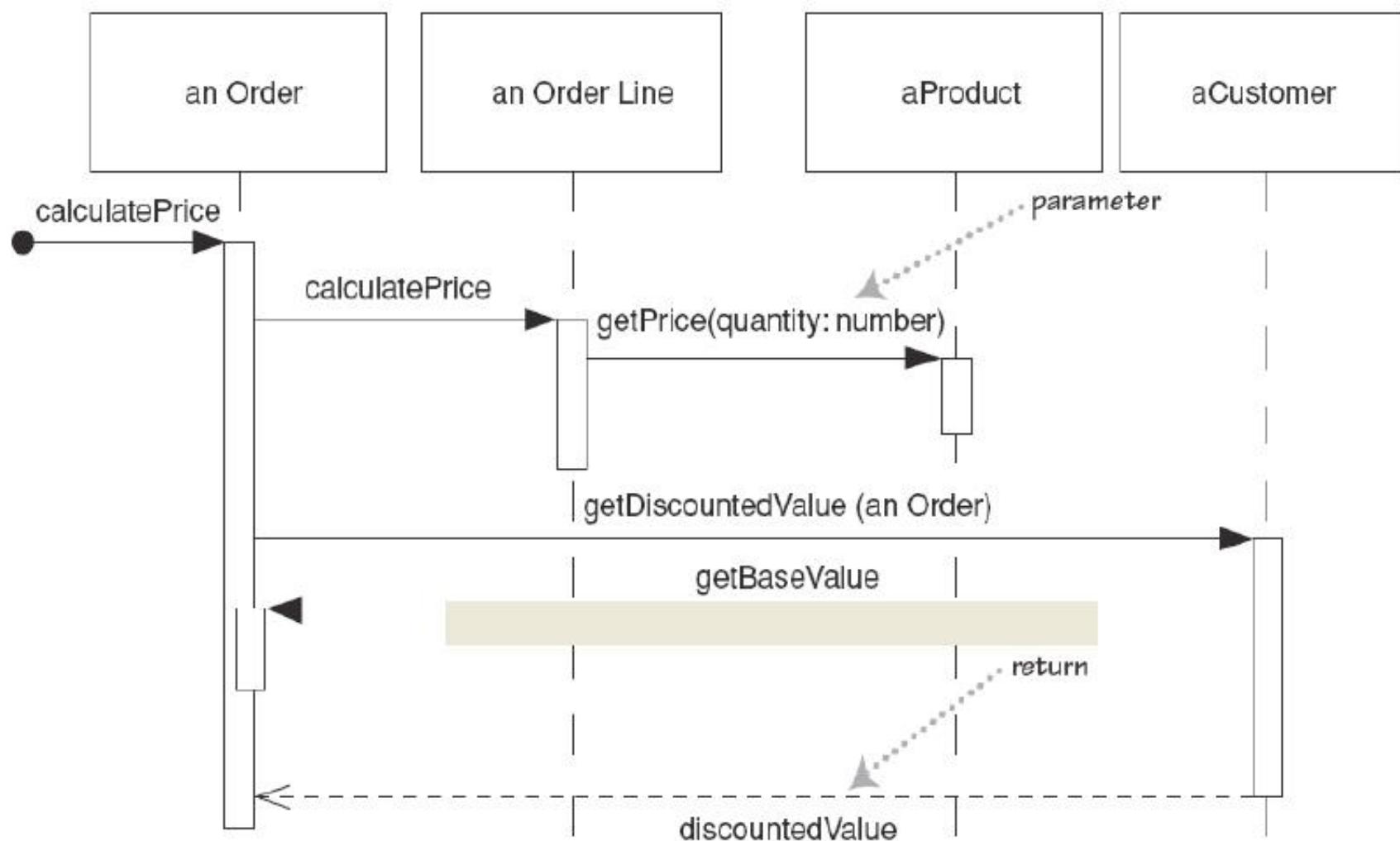


- 순서 다이어그램은 이런 것을 보여주는데 도움이 되는가?

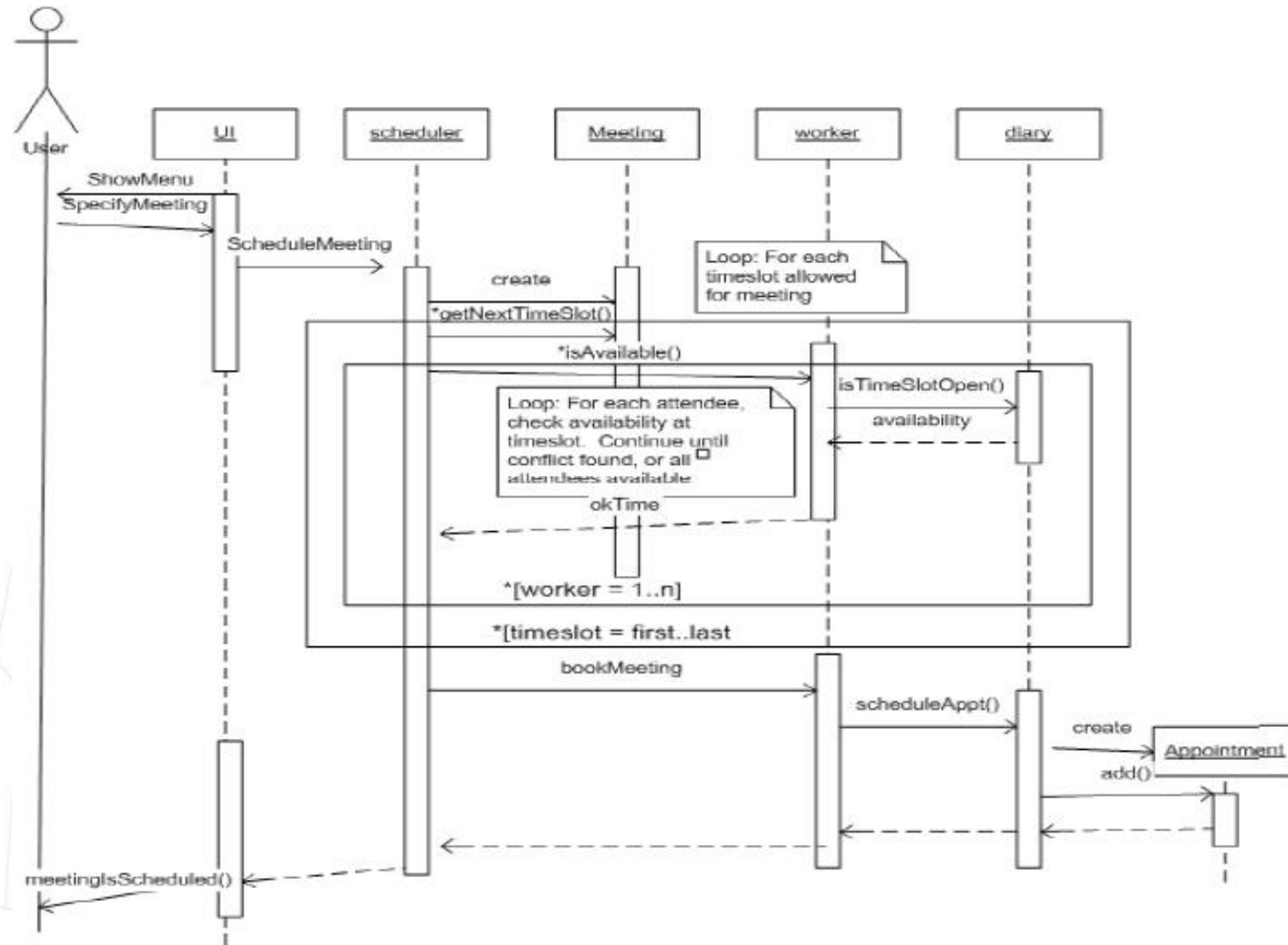
# 다음 다이어그램의 제어 패턴은?



# 이 다이어그램의 제어 패턴은?

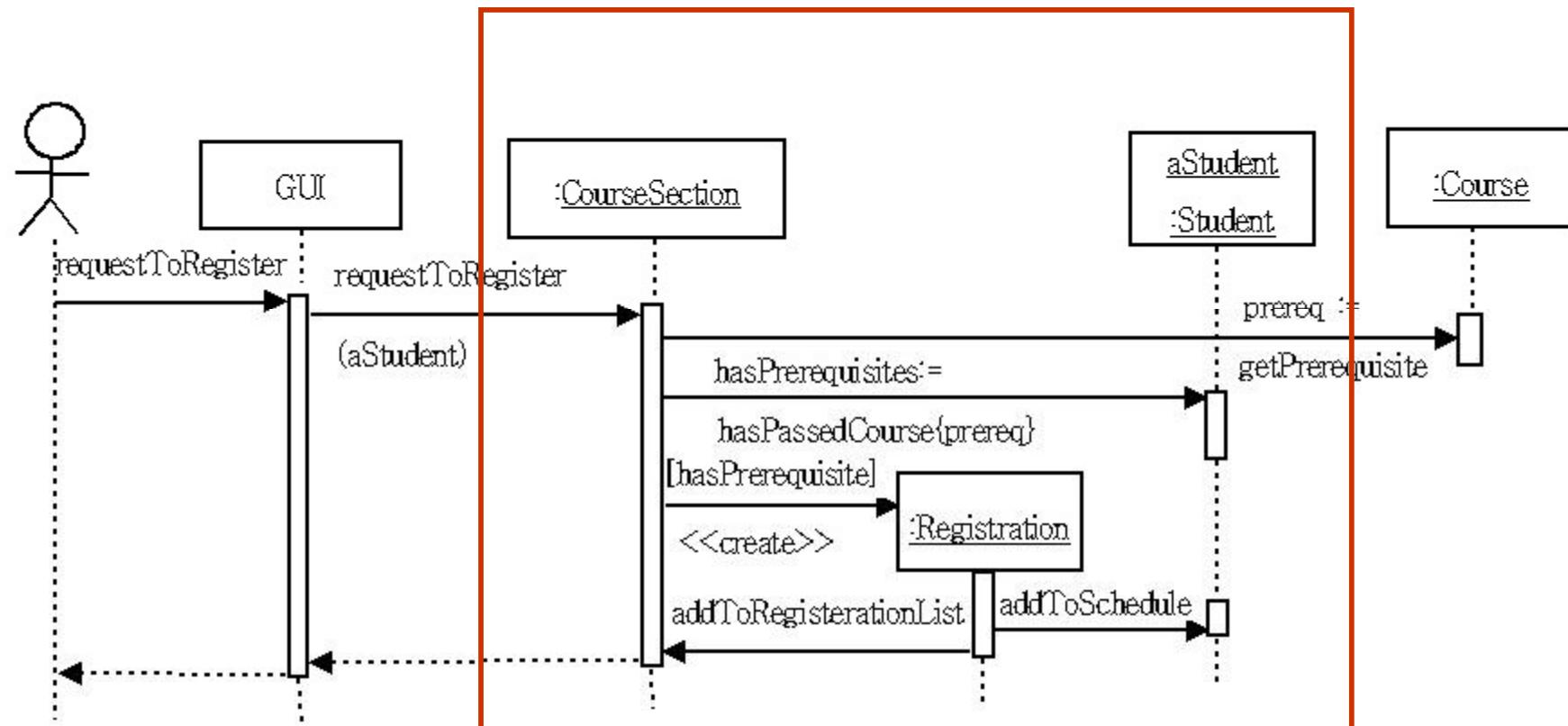


# 다음 시퀀스 다이어그램은?



# 시퀀스 다이어그램과 코딩

- 수강 과목 신청



# 시퀀스 다이어그램과 코딩

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

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

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

# 시퀀스 다이어그램과 코딩

```
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;
        }
    }
}
```

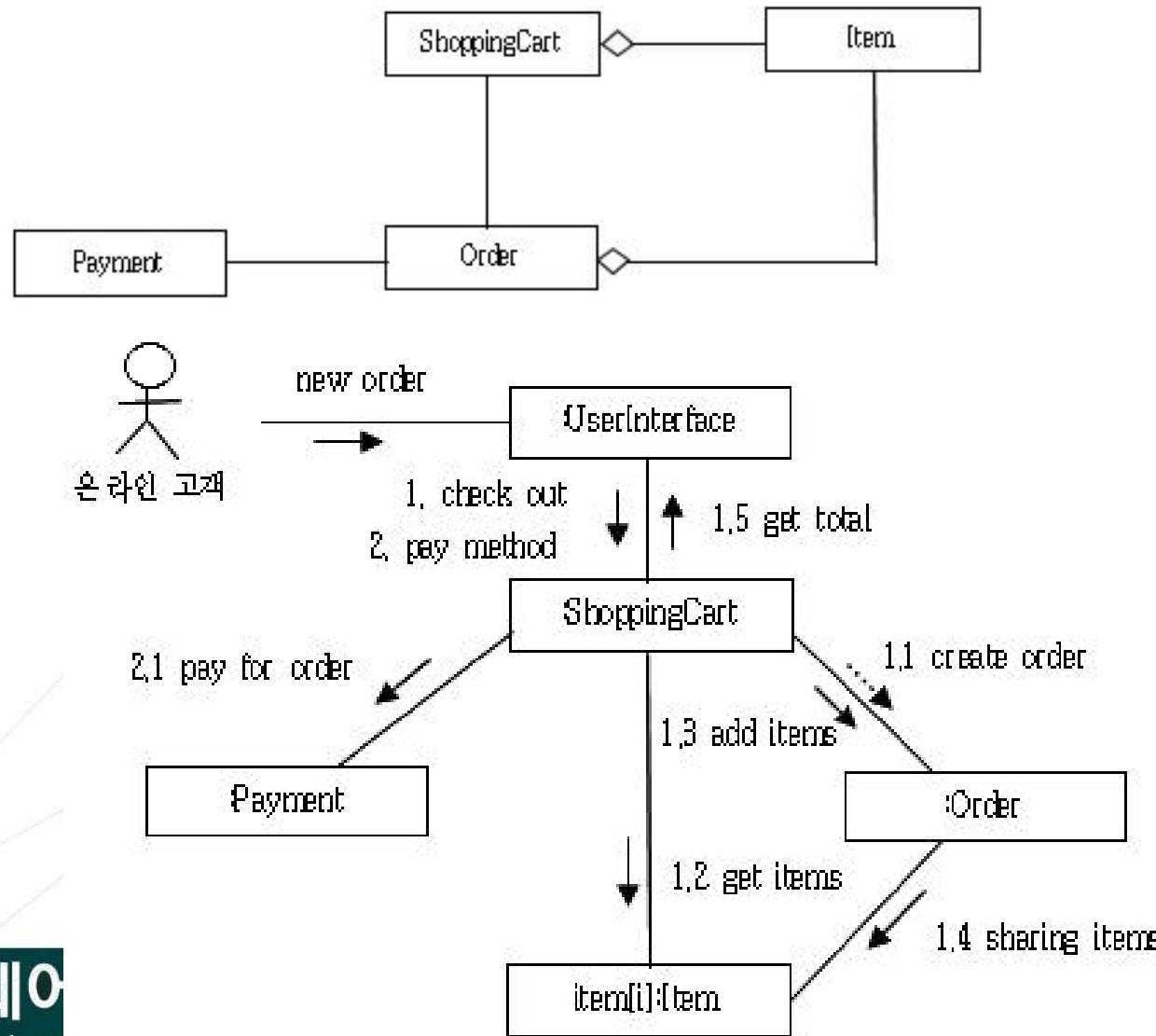
# 왜 바로 코딩하지 않는가?

- 시퀀스 다이어그램은 코드와 매우 밀접하다. 다이어그램을 그리기 전에 왜 바로 코딩하지 않을까?
  - 좋은 시퀀스 다이어그램은 **추상적** 가치를 가짐
  - 시퀀스 다이어그램은 언어 효과를 노린 것
  - 개발자가 아닌 사람도 작성 가능
  - 여러 객체를 한 페이지에서 볼 수 있어
    - 이해가 쉬움
    - 리뷰가 용이
  - 좋은 **커뮤니케이션** 도구



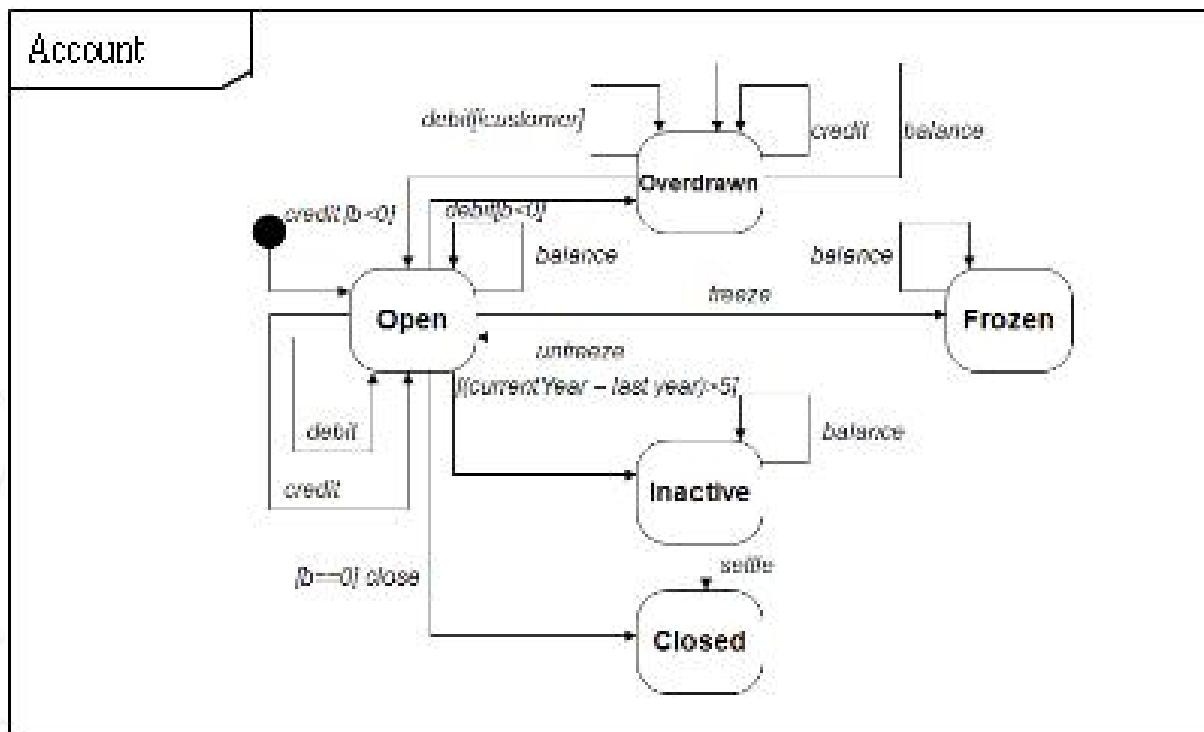
# 커뮤니케이션 다이어그램

- 인터랙션에 참여하는 객체들의 연관을 나타내고 있음



# 상태 다이어그램

- 시스템에서 중요한 역할을 담당하는 클래스의 상태 변화
- 예: ATM – account 클래스



# 상태 다이어그램의 요소

## ● 상태

- 대상이 갖는 생명주기의 한 시점
- 액션이 수행되거나 이벤트를 기다림

aState

## ● 상태 변화

- 상태 사이의 이동
- 이벤트에 대한 반응

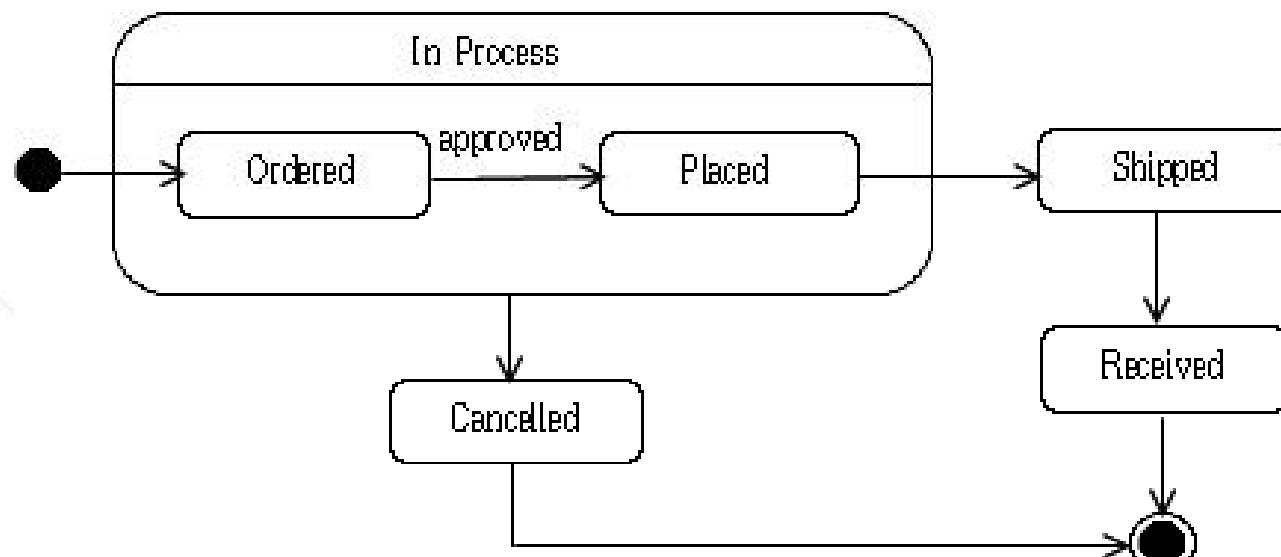


## ● 액션

- On Entry - 상태에 진입할 때 액션이 구동됨
- Do - 상태 안에서 액션이 수행됨
- On Event - 이벤트에 대한 반응으로 액션이 실행됨
- On Exit - 상태에서 빠져나가기 바로 전에 액션이 실행됨
- 형태: action-label / action

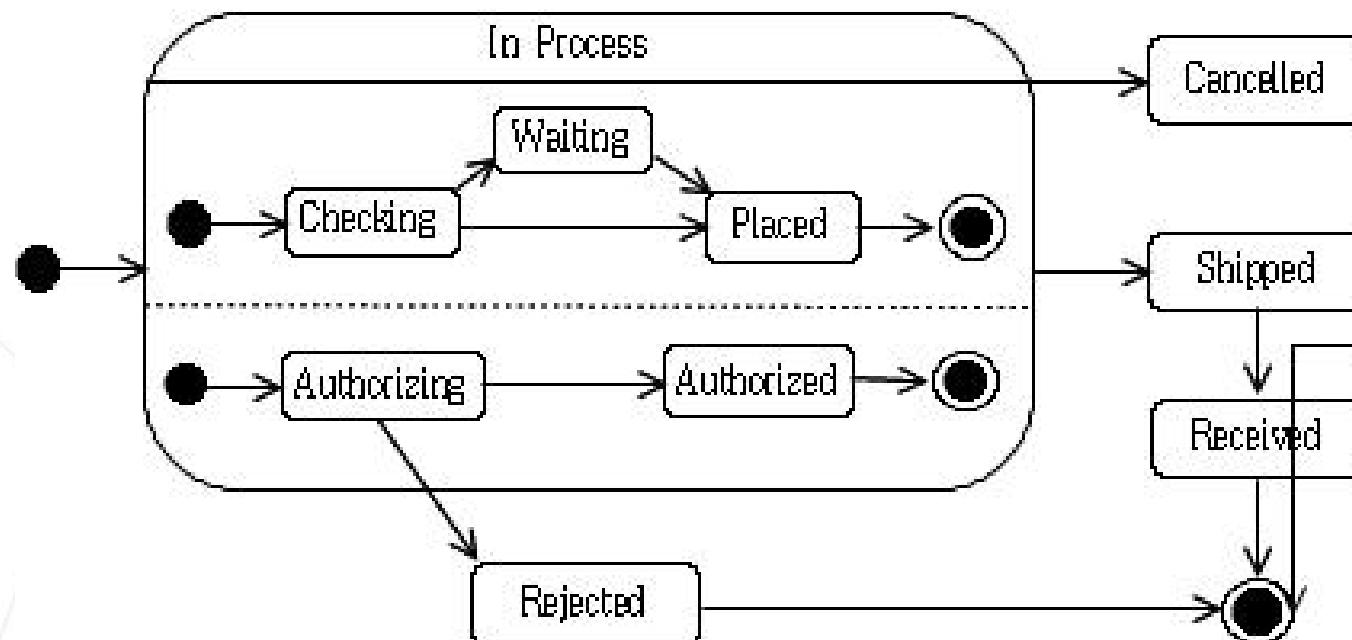
# 복합 상태

- 서브 상태로 분할 할 수 있음
- Order의 상태 변환



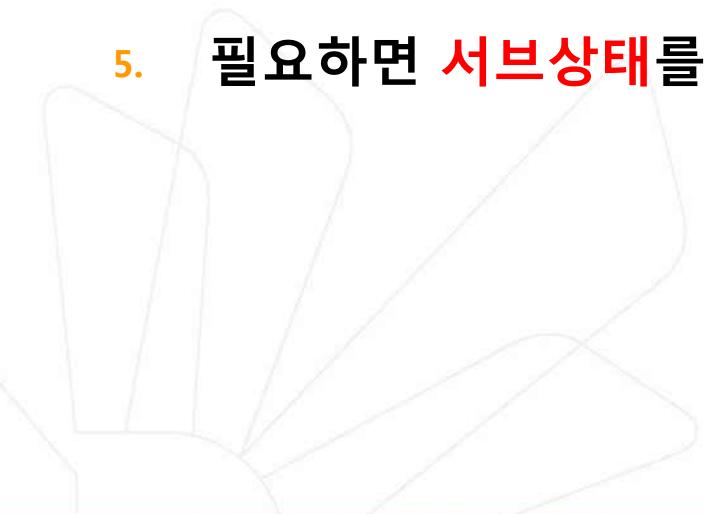
# 병렬 상태

- 복합 상태 안에서 동시에 여러 개의 병렬 서브상태로 구성
- 병행 흐름은 **독립적**

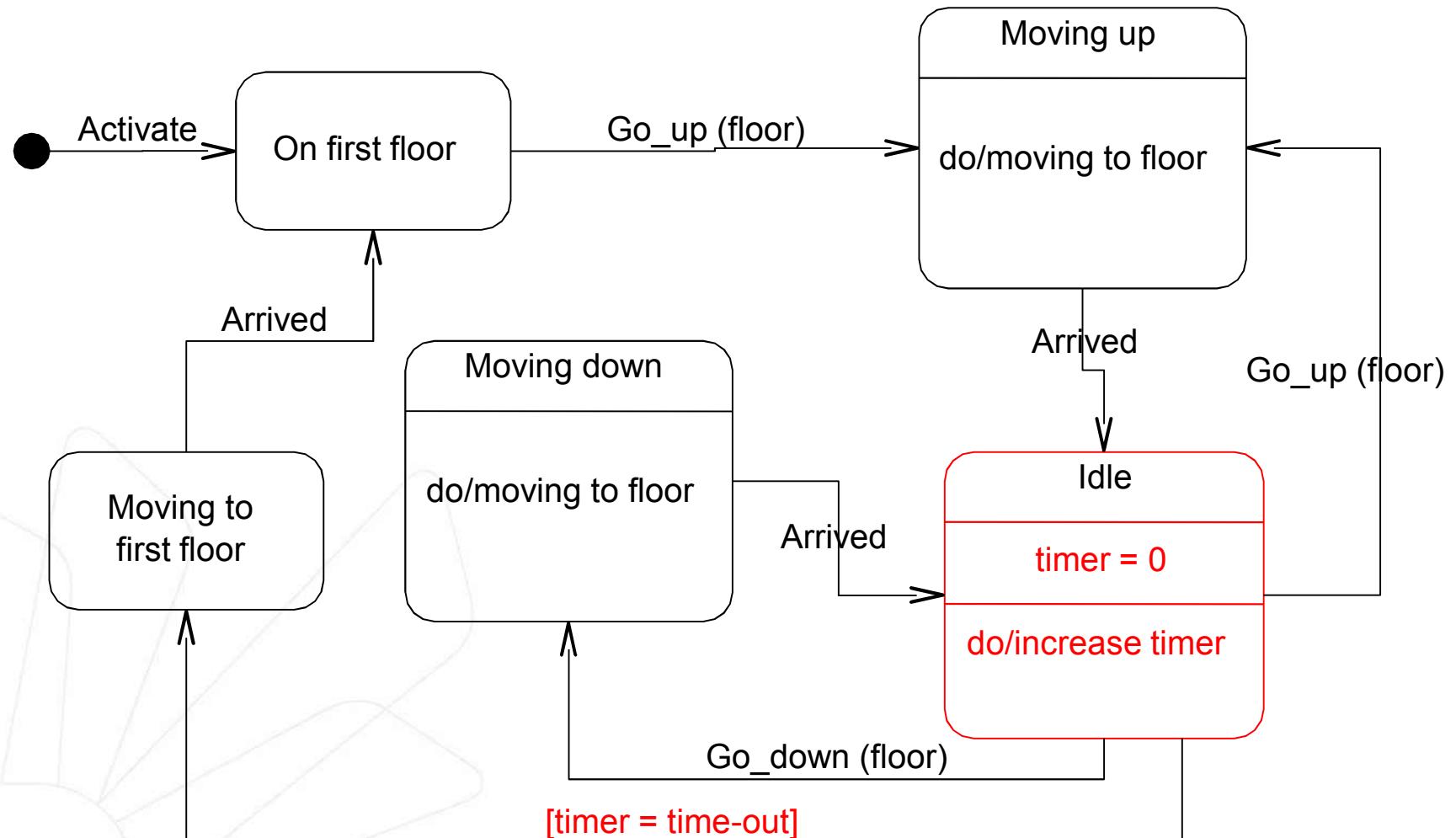


# 상태 다이어그램 그리기

1. 범위를 정한다.
2. 시작, 종료 상태를 파악한다.
3. 객체나 서브시스템이 어떤 상태들을 갖는지 찾아낸다.
4. 상태를 전환시키는 이벤트, 액션, 조건들을 파악한다.
5. 필요하면 서브상태를 이용하여 확장한다.



# 상태 다이어그램의 예



## 상태 다이어그램의 코딩

- 예> 비디오 대여 시스템의 비디오테이프 객체의 상태 변화 모델링

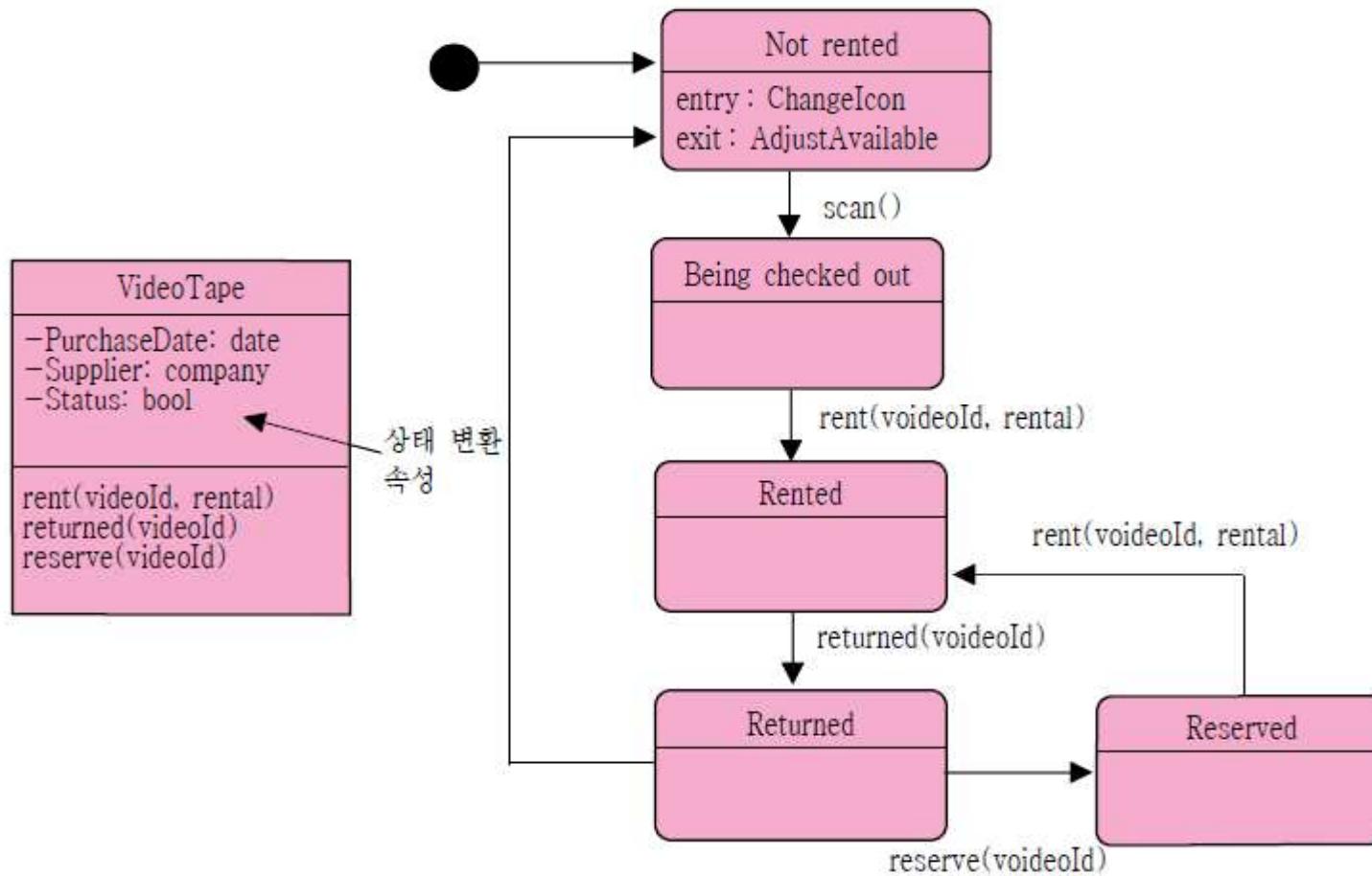


그림 5.37 ▶ Video의 상태 다이어그램

# UML 정리

- UML은 아래서 좋다.

- 공통 언어

- 요구, 명세, 설계를 공유할 수 있게 한다

- 비주얼 구문이 좋다

- 정보를 요약
    - 개발자/기술자가 아닌 사람들에게도 이해 가능

- 도구 지원

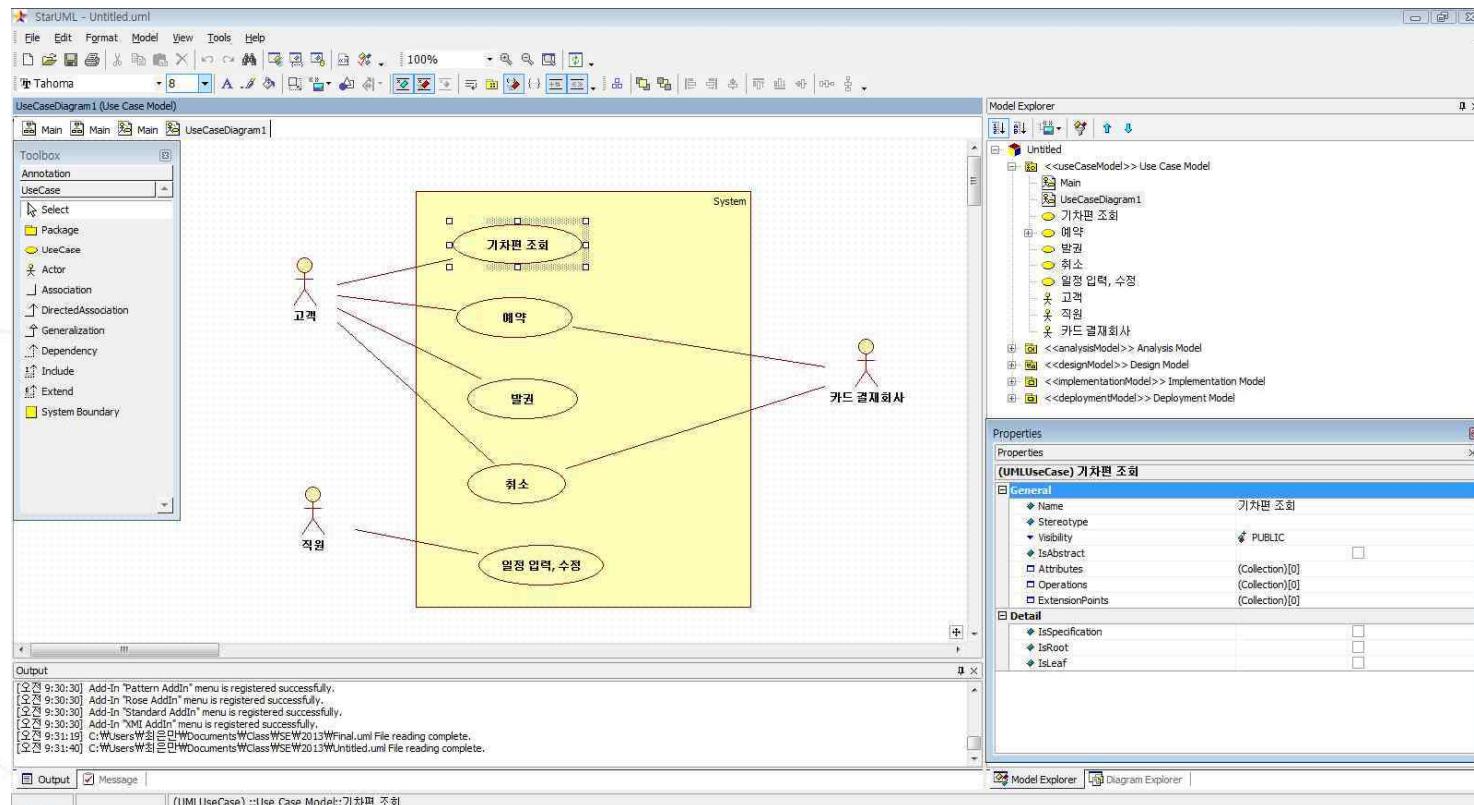
- Visio, Rational, Eclipse, Together
    - 어떤 도구는 UML에서 코드로 자동 변환



# UML 모델링 도구

## ● 많이 알려진 모델링 도구

- IBM의 Rational Modeler, 마이크로소프트의 Visio, ArgoUML, StarUML, NetBeans UML 플러그인
- 예 > StarUML





# Questions?



**dongguk**  
UNIVERSITY