

# 소프트웨어공학개론

오유수, Ph.D.

[Ref] “소프트웨어 공학의 소개”, 한혁수, 홍릉과학출판사

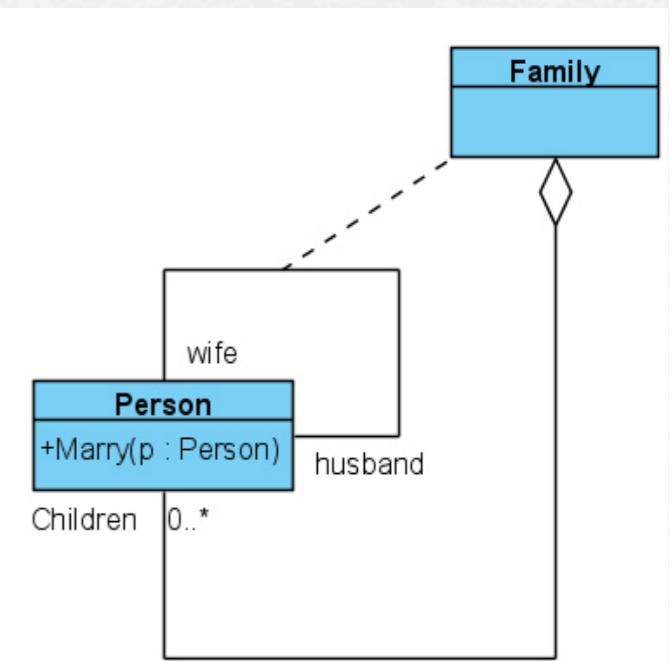
“성공적인 소프트웨어 개발 방법론”, 윤청, 생능출판사

# 오늘의 강의 목차

- 오늘의 강의 내용
- UML과 객체지향 모델링
- UML의 구성
- 유스케이스 다이어그램
- 오늘의 강의 요약
- 요약, Homework, 다음 강의 소개

# UML의 개념

- 통합모델링 언어
- UML은 1994년 소프트웨어 방법론의 선구자인 그래디 부치(Grady Booch), 제임스 럼바(James Rumbaugh), 이바 야콥슨(Ivar Jacobson)에 의해 연구
- 1997년 객체관리그룹(OMG, Object Management Group)에서 여러 표기법을 통합하여 UML 발표
- UML은 객체지향 시스템 개발 분야에서 가장 우수한 모델링 언어로 인식되고 있다.



# UML의 특징

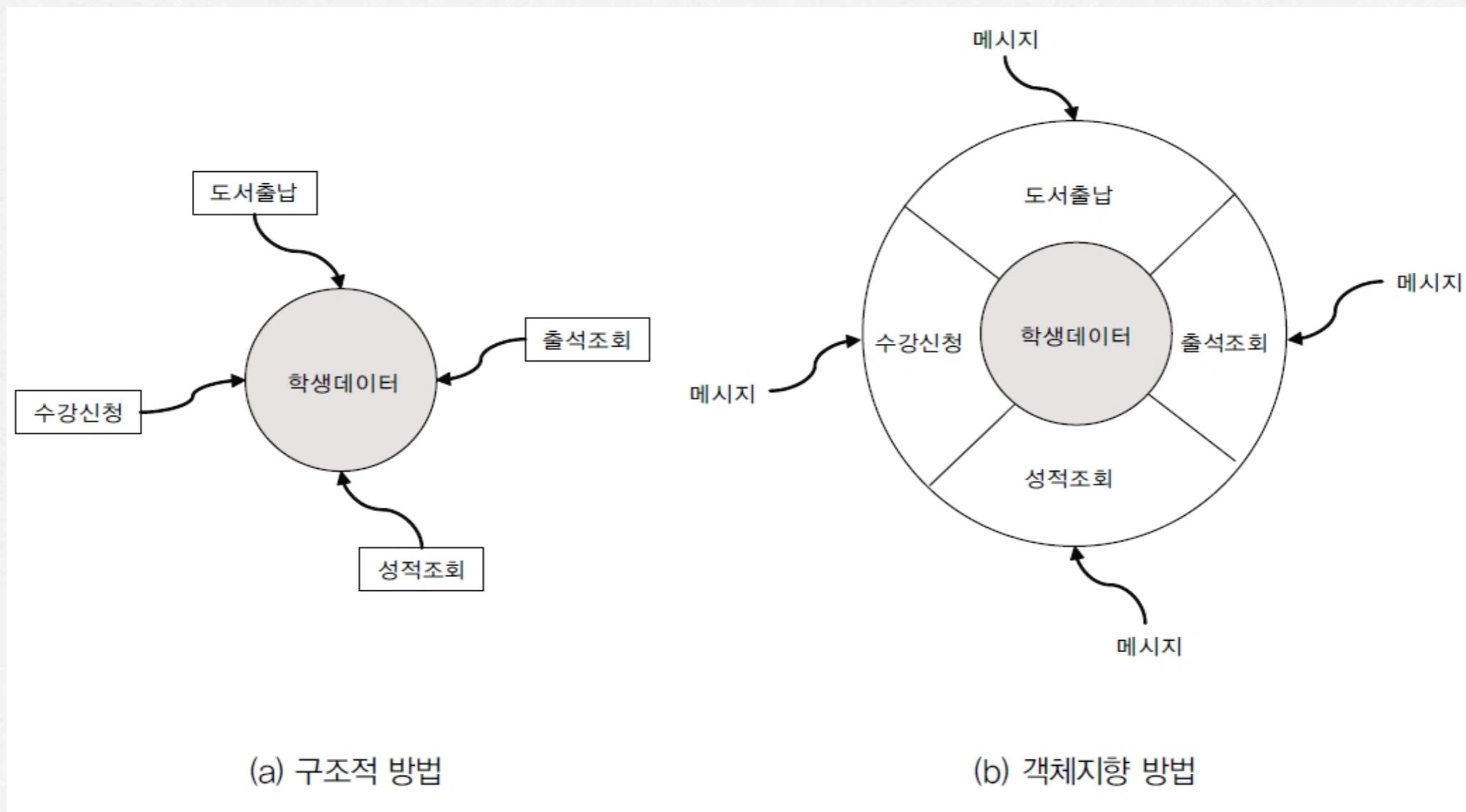
- UML은 가시화 언어이다
  - UML은 소프트웨어의 개념 모델을 시각적인 그래픽 형태로 작성
  - 표기법에 있어서는 각 심벌에 명확한 정의가 존재
  - 개발자들 사이에 오류 없는 원활한 의사소통이 가능
- UML은 명세화 언어이다
  - 명세화란 정확하고, 명백하며, 완전한 모델을 만드는 것을 의미
  - UML은 소프트웨어 개발 과정인 분석, 설계, 구현 단계의 각 과정에서 필요한 모델을 정확하고 완전하게 명세화할 수 있는 언어
- UML은 구축 언어이다
  - UML은 Java, C++, Visual Basic과 같은 다양한 프로그래밍 언어로 표현
  - UML로 명세화된 설계 모델은 프로그램 소스 코드하여 구축 가능
  - 구축되어 있는 소스를 UML로 역변환하여 분석하는 역공학이 가능
- UML은 문서화 언어이다
  - UML은 시스템 아키텍처와 이에 대한 모든 상세 내역에 대한 문서화를 다루며, 요구사항을 표현하고 시스템을 테스트하는 언어 제공

# UML의 용도

- 시스템을 만들기 전에 모델을 만드는 것은 건물을 짓기 위한 설계도처럼 아주 중요한 역할
- 시스템을 만드는데도 어휘와 규칙을 마련하여 시스템을 개념적, 물리적으로 표현하는 모델이 필요
- 성공적으로 시스템을 만들기 위해서는 객체지향적인 분석과 설계를 위한 표준으로 인정받는 모델링 언어인 UML.이 필요

시스템 종류	설명	시스템 예
정보 시스템	사용자로부터 정보를 입수, 가공, 저장하고 적절한 양식으로 사용자에게 출력하는 작업을 한다.	각종 공공기관, 금융기관, 회사에서 해당 기관 고유의 업무 지원
기술적 시스템	기계 등을 제어하는 시스템이다.	통신 장비, 군 장비, 공장의 기계
내장 시스템	하드웨어에 내장되어 수행되는 시스템이다.	휴대전화기, 세탁기, TV, 전기밥솥

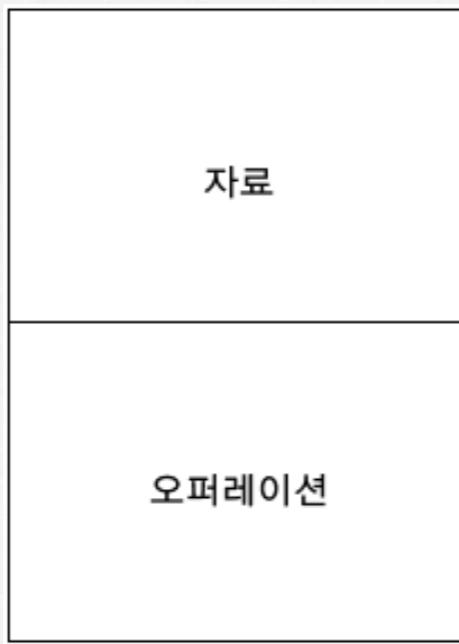
# 객체지향 개념



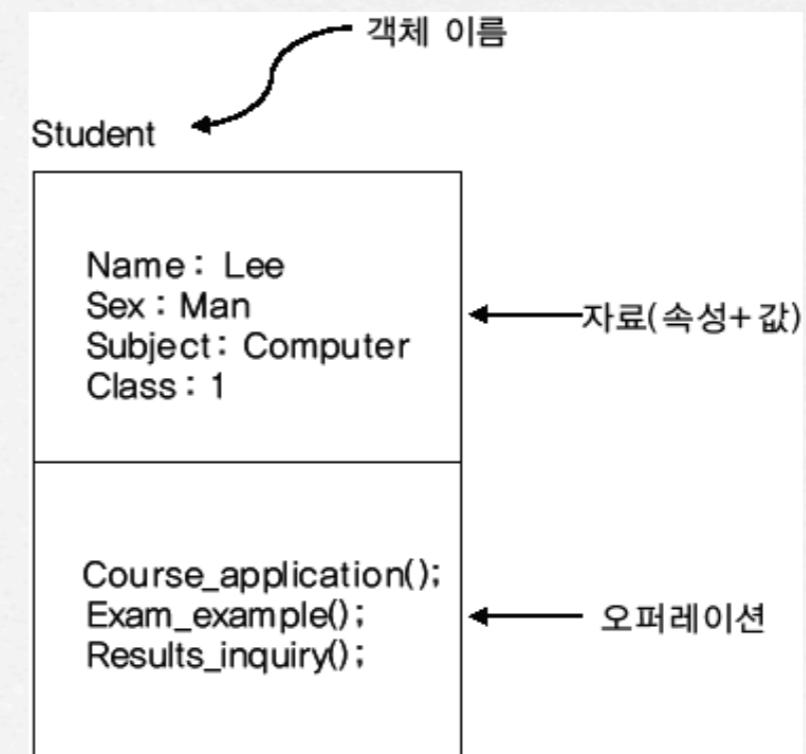
[그림 1-1] 구조적 방법과 객체지향 방법

# 객체지향 기본 개념

- 객체.
- 객체란 우리가 살아가는 세계에 존재하거나 생각할 수 있는 것을 말함
- 현실세계에 존재하는 개념들 중 소프트웨어 개발 대상이 되는 것들은 모두 객체라고 할 수 있다.



[그림 1-2] 객체의 구조

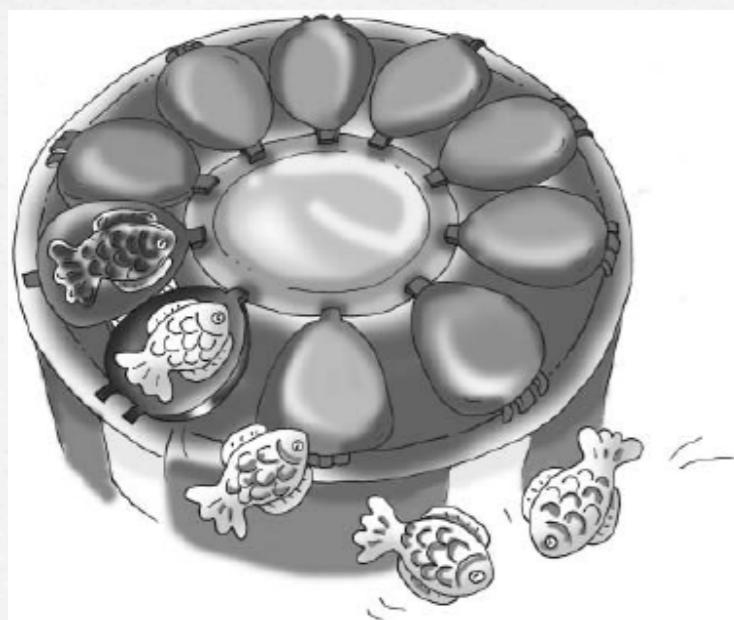


[그림 1-3] 학생에 대한 객체의 예

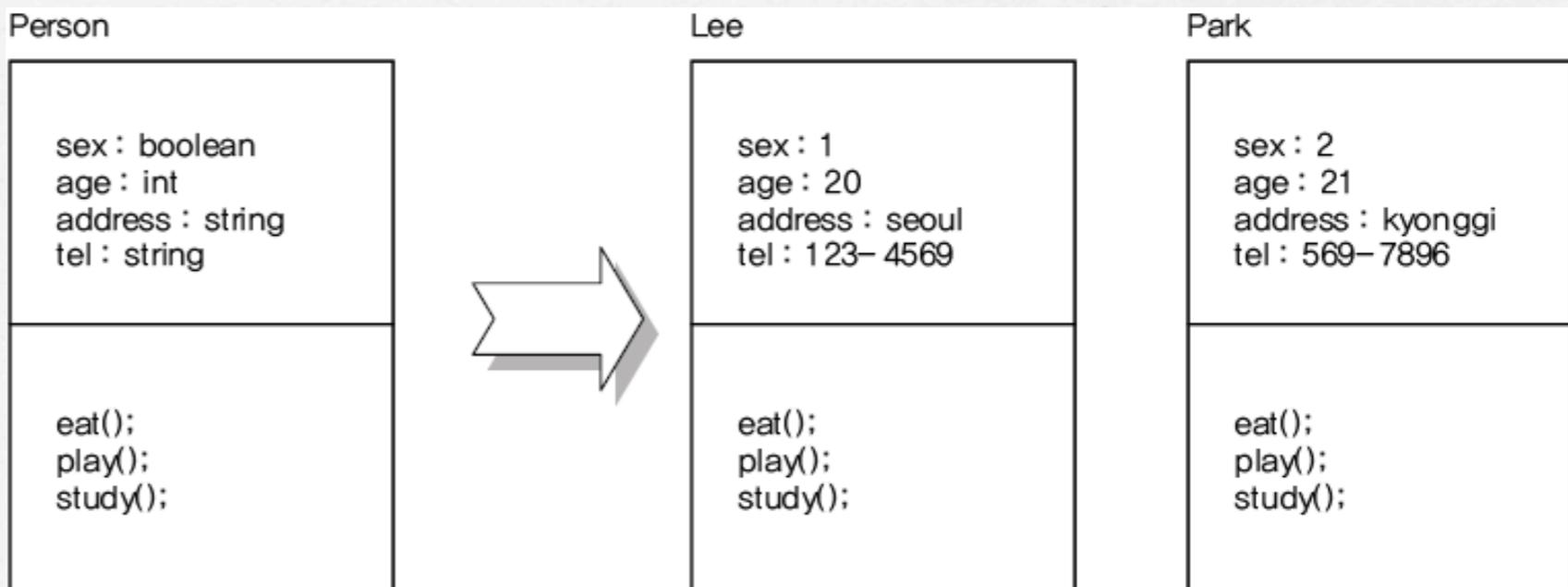
# 객체지향 기본 개념

## □ 클래스

- 클래스는 객체를 생성할 수 있는 구조와 정보를 가지는 틀이라고 정의할 수 있다.
- 붕어빵 기계를 클래스라고 하면 이 기계에서 나오는 붕어빵은 객체
- 클래스는 개념적인 의미이며, 객체는 구체적인 의미
- 클래스에서 생성된 객체들은 같은 속성과 같은 오퍼레이션에 대한 정의를 갖음



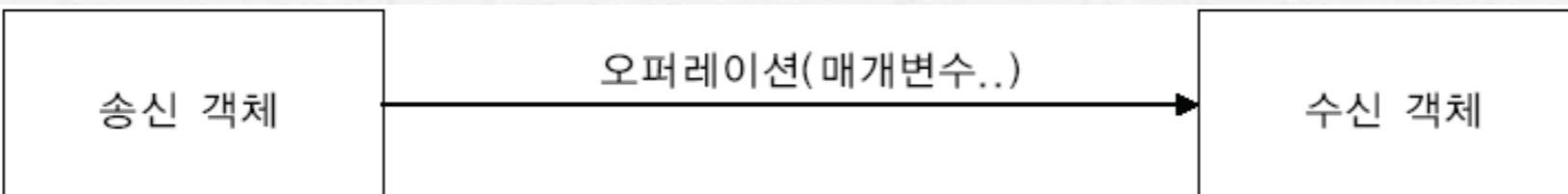
[그림 1-4] 붕어빵 기계



[그림 1-5] 클래스와 객체

# 메시지

- 객체들 사이의 상호작용 수단이 바로 메시지
- 메시지는 어떤 한 객체가 다른 객체에게 특정 작업을 요청하는 신호
- 메시지를 보내는 객체를 송신 객체(Sender)라 하고, 메시지를 받아서 동작을 수행하는 객체를 수신 객체(Receiver)라 한다.



- 메시지 전달을 프로그래밍 방법으로 표현하면,
  - .수신 객체 이름 . . . 수행할 오퍼레이션 이름 . . . 수행할 때 전달되는 매개변수로 표현

메시지=수신객체 이름+수행할 오퍼레이션 이름+전달되는 매개변수

예 myStudent.get(컴퓨터공학과, 2008);

Student.setCourse("UML");

myPoint.move(10,2)

(객체이름) (오퍼레이션) (매개변수)

# 모델링 방법

## □ 부처 방법론

- 설계 중심의 방법론으로서 시스템을 몇 개의 뷰(View)로 분석할 수 있다고 보았다. (뷰를 모델 다이어그램으로 나타냄)
- 거시적 개발 프로세스와 미시적 개발 프로세스를 모두 포함하고 단계적 접근과 자동화 도구를 지원

## □ 야콥슨의 OOSE

- 유스케이스를 강조한 방법론이다. 유스케이스는 액터와 상호작용하는 시스템의 요구사항을 정의하고, 정의된 유스케이스는 개발, 테스트, 검증 단계에서 사용 함
- 방법론이 복잡하여 초보자에게는 어렵지만, 큰 규모의 시스템 개발에 효율적

## □ 럼바의 OMT

- 하나의 시스템을 기술하기 위하여 3가지 모델을 사용
- 객체 모델(Object Model), 동적 모델(Dynamic Model), 기능 모델(Functional Model)
- 시스템 분석에서 3가지 모델을 이용하여 시스템이 요구하는 객체를 완벽히 기술

# UML

- 부치 방법론과 OOSE, 그리고 OMT를 하나로 합한 방법론
- 분산객체 (Distributed Objects)의 표준화와 OMG의 CORBA (Common Object Request Broker Architecture)의 표준 분석설계 방법론으로 채용

# 다이어그램(Diagram)

- . 다이어그램 은 요소들을 그림으로 표현한 것
- 서로 다른 관점에서 시스템을 가시화하기 위해 다이어그램으로 표현하기 때문에 각 다이어그램은 시스템을 여러 방면으로 투영하는 것
- 시스템의 정적인 부분(구조 모델링)을 가시화하기 위해 4가지 다이어그램 사용
  - . 클래스 다이어그램 (Class Diagram)
  - 객체 다이어그램 (Object Diagram)
  - 컴포넌트 다이어그램 (Component Diagram)
  - 배치 다이어그램 (Deployment Diagram)
- 시스템의 동적인 부분(행위 모델링)을 가시화하기 위해 5가지 다이어그램 사용
  - 유스케이스 다이어그램 (usecase Diagram)
  - . 순차 다이어그램 (Sequence Diagram)
  - 통신 다이어그램 (Communication Diagram)
  - . 상태 다이어그램 (State Diagram)
  - 활동 다이어그램 (Activity Diagram)

# 다이어그램(Diagram)

- . 클래스 다이어그램 .
  - 클래스, 인터페이스, 통신, 그리고 이들의 관계들을 나타내며, 객체지향 시스템 모델링에서 가장 공통적으로 쓰이는 다이어그램
  - 시스템의 정적 설계 뷰를 다루며, 활성 클래스를 갖는 클래스 다이어그램은 시스템의 정적 프로세스 뷰를 다룬다.
- 객체 다이어그램
  - 객체와 객체들 사이의 관계를 나타낸다.
  - 특정 시점의 객체들의 구조적 상태를 표현
  - 클래스 다이어그램처럼 시스템의 정적 설계 뷰와 정적 프로세스 뷰를 다루지만 실제 사례나 프로토타입 사례의 시각에서 표현
- 컴포넌트 다이어그램
  - 컴포넌트 사이의 구성과 의존을 나타내며, 시스템의 정적 구현 뷰를 다룬다.
  - 클래스 다이어그램과 관련이 있는데, 일반적으로 하나의 컴포넌트는 클래스 다이어그램에 있는 하나 또는 그 이상의 클래스, 인터페이스, 통신들과 대응

# 다이어그램(Diagram)

- 배치 다이어그램
  - 실행 시 처리하는 노드와 그 노드에 있는 컴포넌트들의 구성을 나타내며, 시스템의 정적 배치 뷰를 나타낸다.
  - 컴포넌트 다이어그램과 관련이 있는데, 일반적으로 하나의 노드는 컴포넌트 다이어그램 안에 있는 하나 또는 그 이상의 컴포넌트를 수용하기 때문
- 유스케이스 다이어그램
  - 유스케이스(시스템을 사용하는 다양한 경우)와 행위자(외부 행위자)의 관계를 구조적으로 나타낸다.
  - 시스템의 정적 유스케이스 뷰를 다룬다.
  - 시스템 행동을 조직화하고 모델링하는데 특히 중요
- 상태 다이어그램.
  - 상태 머신을 나타내며, 시스템의 내부 전이를 표현
  - 상태 머신은 상태.(State), 전이.(Transition), 이벤트.(Event), 활동.(Activity)으로 구성
  - 시스템의 동적 뷰를 다룬다.
  - 인터페이스, 클래스, 또는 통신의 행동을 모델링하는데 중요
  - 이벤트에 따라 순차적으로 일어나는 객체 행동에 중점을 두기 때문에 빠른 반응이 필요한 시스템을 모델링할 때 유용

# 다이어그램(Diagram)

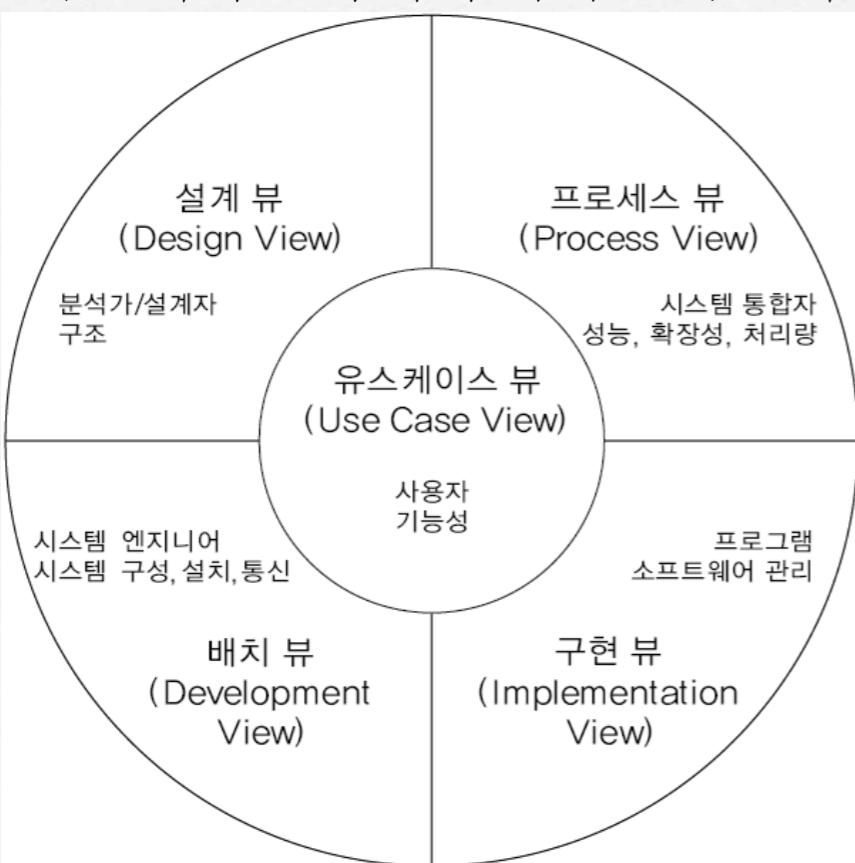
- 순차 다이어그램과 통신 다이어그램
  - 순차 다이어그램과 통신 다이어그램은 교류도(Interaction Diagram)의 한 종류
  - 교류도는
    - 객체와 관계, 그리고 이들 사이에 보낼 수 있는 메시지로 구성되는 교류
    - 교류도는 시스템의 동적 뷰를 다룬다.
  - 순차 다이어그램은
    - 메시지의 시간적 순서를 강조하는 교류도이며, 통신 다이어그램은 메시지를 주고받는 객체의 구조적 구성을 강조하는 교류도
    - 시스템 외부 이벤트를 처리하기 위하여 시스템 내부 객체간에 주고받는 동적 메시지를 시간의 흐름에 따라 표현한 것
  - 통신 다이어그램은
    - 순차 다이어그램과 동일한 내용을 객체 상호관계의 관점에서 표현한 것
    - 순차와 통신 다이어그램은 동일한 구조로 되어 있어서 서로 변환이 가능
- 활동 다이어그램
  - 시스템 내부에 있는 활동의 흐름
  - 시스템의 동적 뷰를 다루며, 시스템 기능을 모델링 할 때 중요하고, 객체간의 제어흐름에 중점을 둔다.

# UML 뷰

- 큰 규모의 건축물을 만들 때는 설계라는 과정이 반드시 필요
- 소프트웨어도 매우 복잡하기 때문에 몇 가지 관점에서 기술할 필요가 있다.
- 객체지향 전문가들은 5가지 중요한 관점 (view)을 정의
- 소프트웨어의 5가지 관점을 표현하기 위하여 UML 다이어그램을 이용
  - . 유스케이스 뷰 . (use case view)
  - . 설계 뷰 . (Design view)
  - . 프로세스 뷰 . (Process view)
  - . 구현 뷰 . (Implementation view)
  - . 배치 뷰 . (Development view)

# 유스케이스 뷰

- 외부 행위자에 의해 인식되는 시스템의 기능 요구사항을 보여주는 관점
- 다른 4개의 관점은 어떻게 하면 원하는 기능을 제공할 수 있는가에 관한 것인 반면에,
  - 유스케이스 뷰는 사용자가 시스템으로부터 원하는 기능이 무엇인지를 정의하는 것
- 유스케이스 뷰에서 파악된 유스케이스에 따라 4가지 관점의 내용이 달라짐



[그림 2-22] 소프트웨어 아키텍처의 5개 뷰

구분	이용되는 UML 다이어그램
정적인 측면	유스케이스 다이어그램
동적인 측면	상태 다이어그램, 순차 다이어그램 통신 다이어그램, 활동 다이어그램

[표 2-1] 유스케이스 뷰

# 설계 뷰

- . 유스케이스 뷰.는
  - 유스케이스 뷰에서 정의된 기능을 시스템이 제공하기 위해서는 어떤 클래스·컴포넌트가 필요하고,
  - 클래스·컴포넌트들이 서로를 어떻게 이용하고 호출하는지에 초점을 맞춘다.
- 시스템 내부의 클래스와 컴포넌트를 파악하고 기술하는 것
- 시스템 내부는 정적인 측면과 동적인 측면으로 나눌 수 있다.

구분	관심 사항	이용되는 UML 다이어그램
정적인 측면	클래스 및 클래스 사이의 관계	클래스 다이어그램, 객체 다이어그램
동적인 측면	클래스 내의 동작	상태 다이어그램
	클래스간의 상호작용	순차 다이어그램, 통신 다이어그램
	클래스의 연산 동작	활동 다이어그램

[표 2-2] 설계 뷰

# 프로세스 뷰

- . 프로세스 뷰 .는
  - 설계 뷰와 마찬가지로 시스템 내부의 구조 즉, 클래스와 클래스 사이의 관계 그리고 클래스의 행동 및 클래스 사이의 상호작용에 초점을 맞춘다.
  - 설계 뷰를 기술할 때 사용되는 다이어그램이 프로세스 뷰에서도 동일하게 사용
  - 활성 클래스 : UML에서 이와 같이 자신의 독자적인 제어 스레드를 가진 클래스
    - 실제 시스템에서 프로세스와 스레드로 구현된다.
    - 프로세스와 스레드는 <<process>>와 <<thread>>의 스테레오 타입을 이용하여 표현
- 구현 뷰
  - 구현 뷰는 시스템 구현 형태를 나타내기 위하여 컴포넌트와 같은 구현 모듈과 그들 사이의 관계, 또는 각종 파일과 파일들간의 의존관계 등을 보여주는 뷰
  - 구현 뷰는 컴포넌트 다이어그램으로 표현

구분	이용되는 UML 다이어그램
정적인 측면	컴포넌트 다이어그램
동적인 측면	상태 다이어그램, 순차 다이어그램 통신 다이어그램, 활동 다이어그램

[표 2-3] 구현 뷰

# 배치 뷰

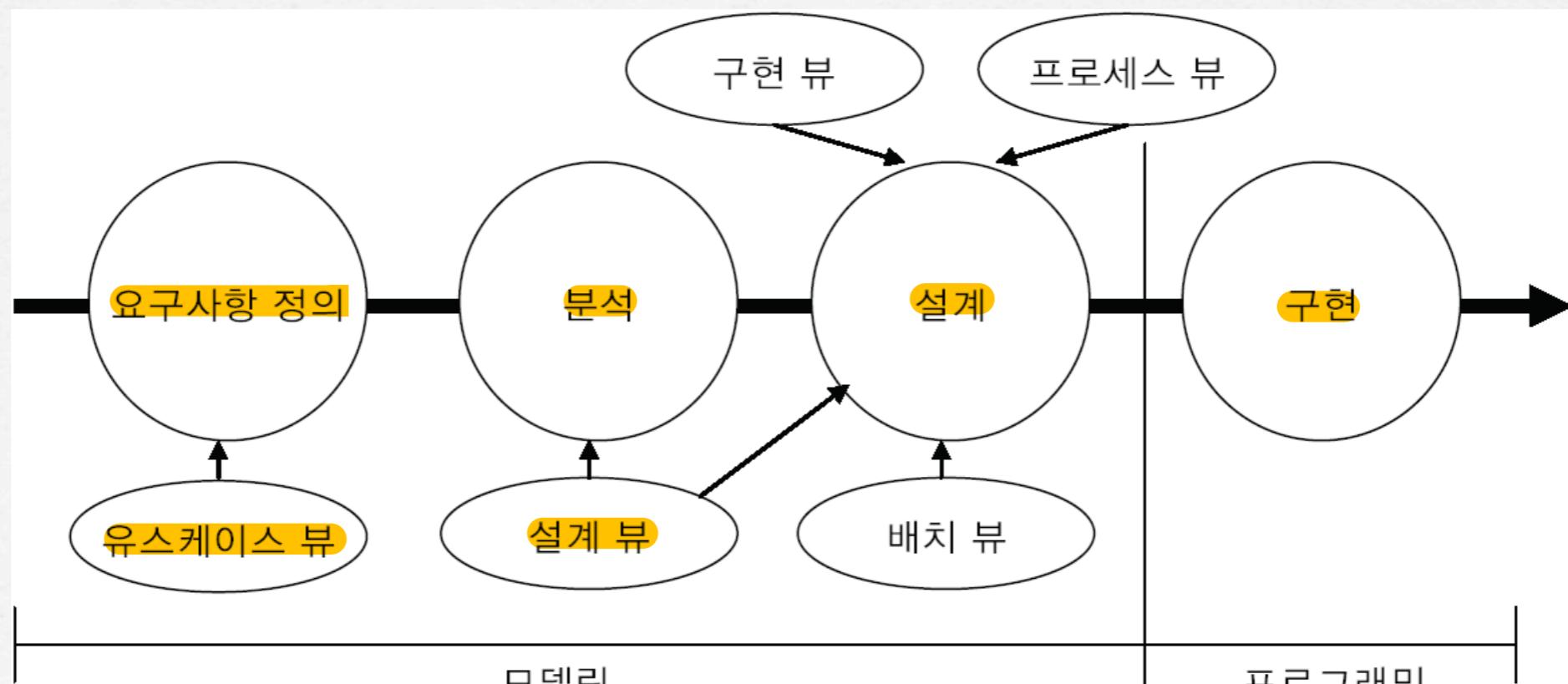
- 시스템을 구성하는 처리장치, 즉 컴퓨터와 컴퓨터 간의 통신방법에 초점을 둔다.
- 배치 다이어그램으로 표현

구분	이용되는 UML 다이어그램
정적인 측면	배치 다이어그램
동적인 측면	상태 다이어그램, 순차 다이어그램 통신 다이어그램, 활동 다이어그램

[표 2-4] 배치 뷰

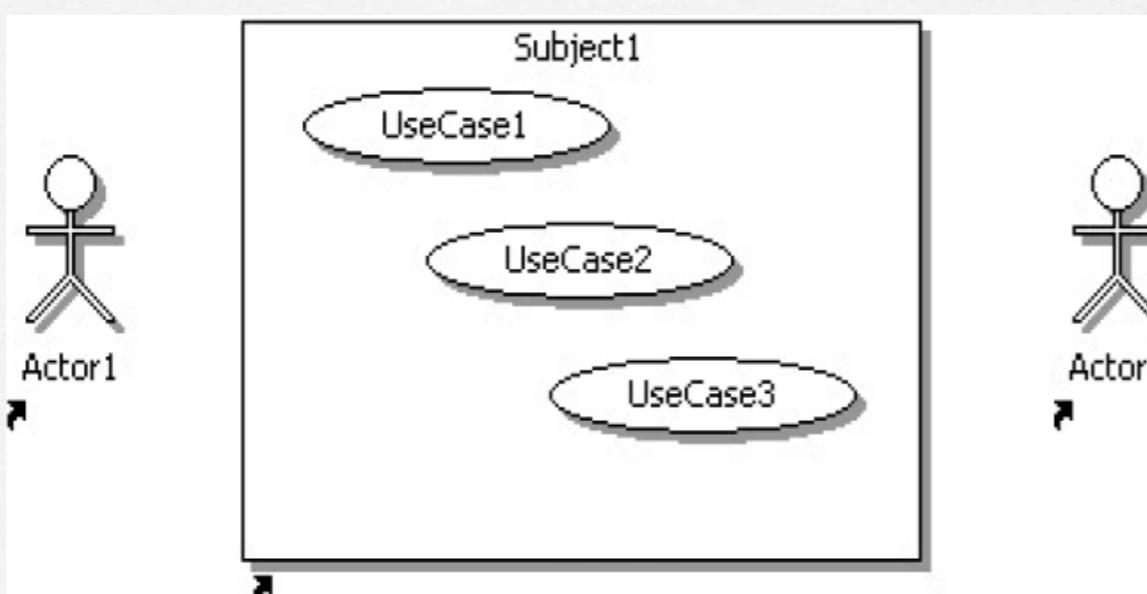
# 개발 활동과 UML 뷰

- 시스템은 요구사항 정의·분석·설계·구현 활동을 순차적으로 수행하면서 구축
- 유스케이스 뷰, 설계 뷰 등 각 관점에서 초점을 두는 부분은 특정 개발 절차와 관련이 있다.
- 시스템을 구축하기 위한 개발 활동이 진행되면서 초점을 두는 관점도 달라진다.



[그림 2-23] 개발 활동과 UML 뷰

- 시스템 개발에 참여하는 사람들을 큰 부류로 나누면,
  - . 의뢰인, 개발팀, 사용자(User) .
- 사용자의 관점을 빨리 이해해야만 쓸모 있고 유용한 시스템을 만들 수 있다.
- 개발자는 가능한 한 모든 요구사항을 파악하여 사용자의 승인을 받아야만 후일 요구사항 변경에 대한 위험부담을 줄일 수 있다.
- 요구사항 정의 활동이 개발과 설계에 커다란 비중을 차지한다.
- [그림 3-1]
  - 네모난 창은 시스템의 경계
  - *usecase1, 2, 3*은 구축할 시스템의 기능을 표현
  - 유스케이스 모델링에서 개발할 시스템 외부 존재를 *Actor*라는 개념으로 정의
  - 시스템의 범위에 해당되어 개발될 시스템의 단위 기능을 *usecase* 개념으로 정의



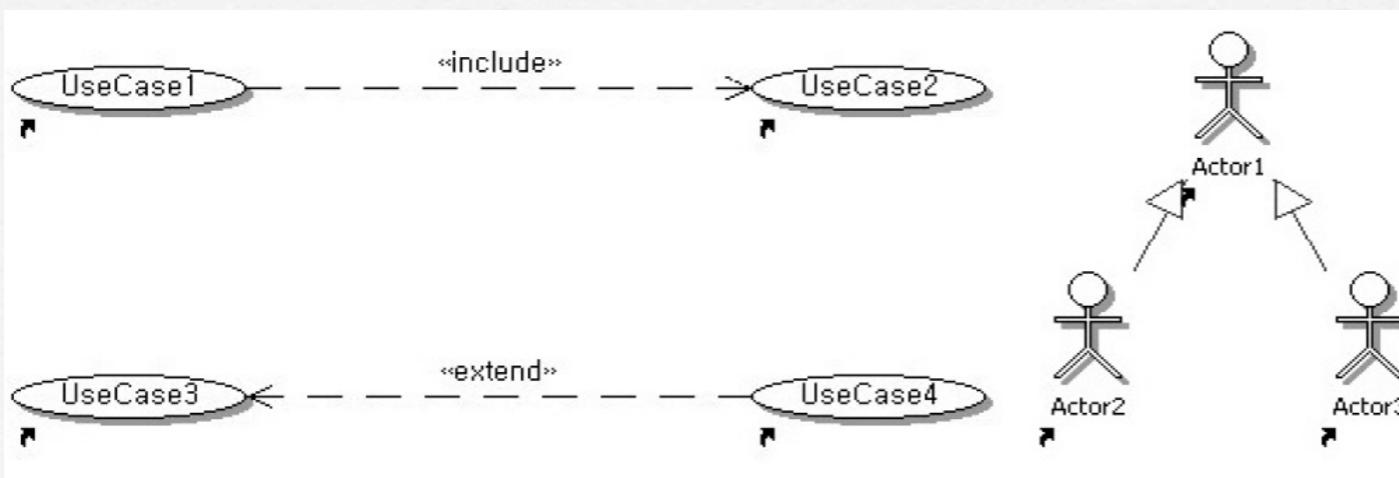
[그림 3-1] 시스템의 경계

- 유스케이스의 역할 : 사용자의 시점에서 시스템을 모델링 한다는 것.
- 유스케이스 : 시스템에 대한 시나리오의 집합
- 시나리오 : 발생되는 이벤트의 흐름이 나타나 있다.
- 이벤트의 흐름 : 사람, 시스템, 하드웨어, 혹은 시간의 흐름에 의해 시작
- 액터 : 이벤트 흐름을 시작하게 하는 객체



[그림 3-2] 유스케이스와 액터

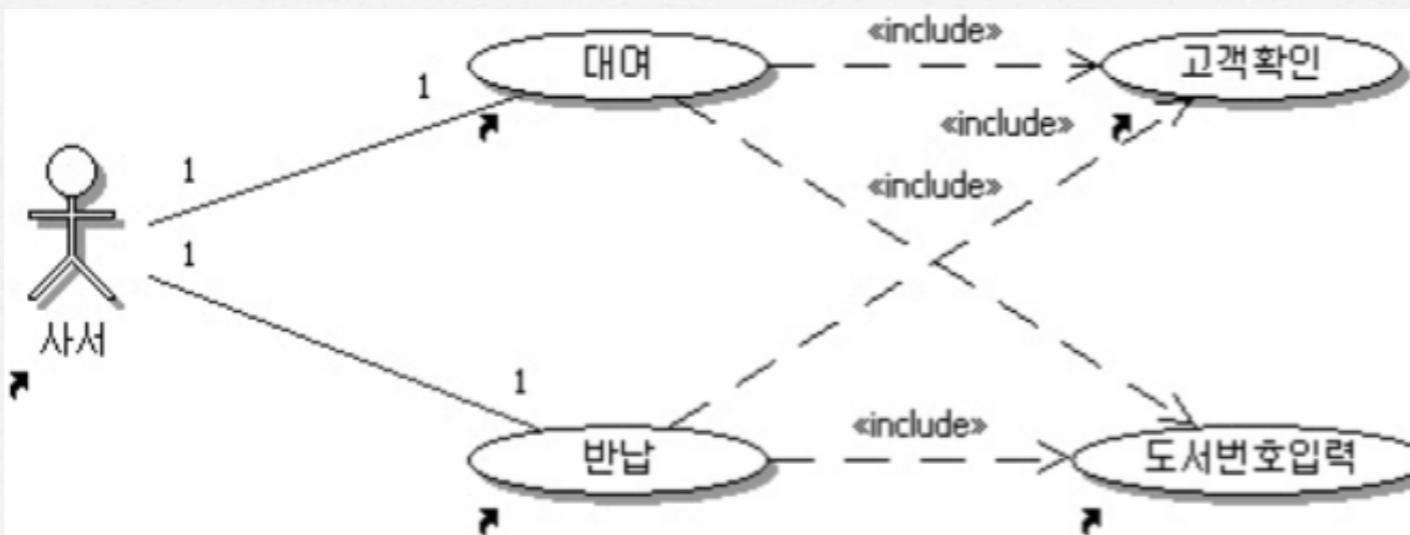
- 유스케이스 사이에는 일반적인 연관관계 이외에 또 다른 관계
  - 포함(include) 관계 : 다른 유스케이스에서 기존의 유스케이스를 재사용하는 관계
  - 확장(Extend) 관계 : 기존의 유스케이스에 진행단계를 추가하여 새로운 유스케이스를 만들어내는 관계
- 액터와 유스케이스에 대한 일반화관계



[그림 3-3] 유스케이스 관계

# 유스케이스간 포함관계

- 포함관계 : 유스케이스를 수행할 때 다른 유스케이스가 반드시 수행되는 것
- 유스케이스 다이어그램에서는 다른 유스케이스가 나타내는 이벤트 흐름을 포함(*include*)하는 관계를 유스케이스간에 표현



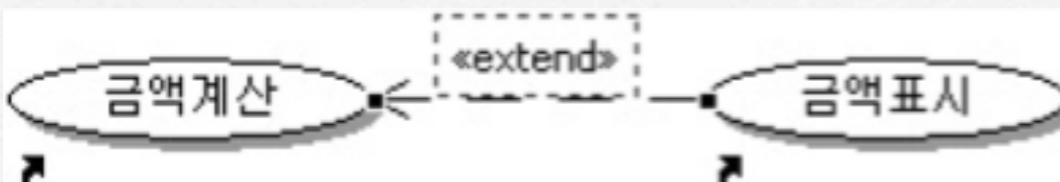
[그림 3-4] 도서관 시스템의 포함관계

[표 3-1] 포함관계를 이용한 '대여' 와 '반납' 유스케이스의 이벤트 흐름

대여 유스케이스 이벤트 흐름	반납 유스케이스 이벤트 흐름
<ol style="list-style-type: none"><li>'고객확인' 유스케이스를 포함한다.</li><li>사서는 '대여'를 선택한다.</li><li>'도서번호입력' 유스케이스를 포함한다.</li><li>도서관 시스템은 고객이 대여가 가능한지 확인한다.</li><li>고객에게 대여를 가능 여부를 표시한다.</li><li>도서관 시스템은 도서를 대여 처리한다.</li></ol>	<ol style="list-style-type: none"><li>'고객확인' 유스케이스를 포함한다.</li><li>사서는 '반납'을 선택한다.</li><li>'도서번호입력' 유스케이스를 포함한다.</li><li>도서관 시스템은 도서를 반납 처리한다.</li></ol>

# 유스케이스간 확장관계

- 유스케이스간 확장관계.
  - 확장관계의 유스케이스는 포함관계처럼 여러 유스케이스에 걸쳐 중복적으로 사용되지 않고, 특정 조건에서 한 유스케이스로만 확장되는 것을 의미
  - 확장(*extend*)하는 유스케이스는 상위 유스케이스로부터 어떠한 특정 조건에 의해 수행됨을 의미
- 포함관계와 확장관계의 차이점
  - 포함관계 : 여러 유스케이스에서 공통적으로 발견되는 기능을 표현
  - 확장관계 : 한 유스케이스에서 추가되거나 확장된 기능을 표현

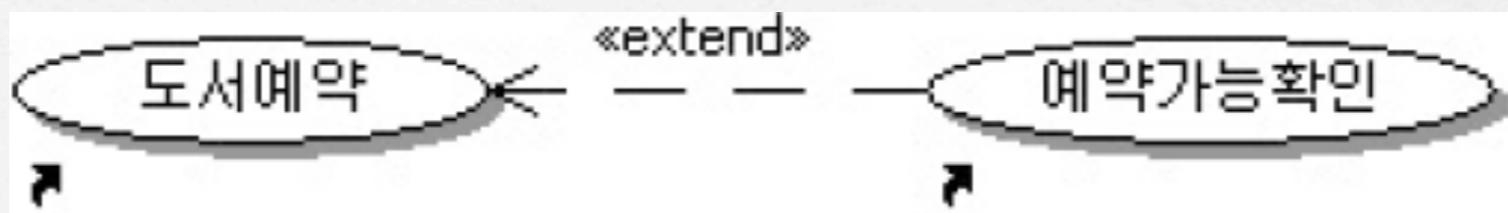


[그림 3-5] 금액계산 유스케이스의 확장

[표 3-2] 금액계산 유스케이스와 금액표시 유스케이스의 이벤트 흐름

금액계산 유스케이스 이벤트 흐름	금액표시 유스케이스 이벤트 흐름
<ol style="list-style-type: none"><li>1. 사용자가 자판기에 동전을 투입하거나 음료수를 선택한다.</li><li>2. 현재 금액에 투입된 동전만큼 액수를 추가한다.</li><li>3. 금액표시 유스케이스 확장</li></ol>	<ol style="list-style-type: none"><li>1. 금액계산 유스케이스로부터 금액을 받는다.</li><li>2. 받은 금액을 표시한다.</li></ol>

- 확장 유스케이스는 특정 조건이 만족되는 상황에서만 확장 유스케이스의 이벤트 흐름이 수행
- 확장 유스케이스의 이벤트 흐름의 수행 여부를 결정짓는 조건은 확장 유스케이스를 포함하는 기준 유스케이스가 아니라 확장 유스케이스에 표현되는 점을 주목



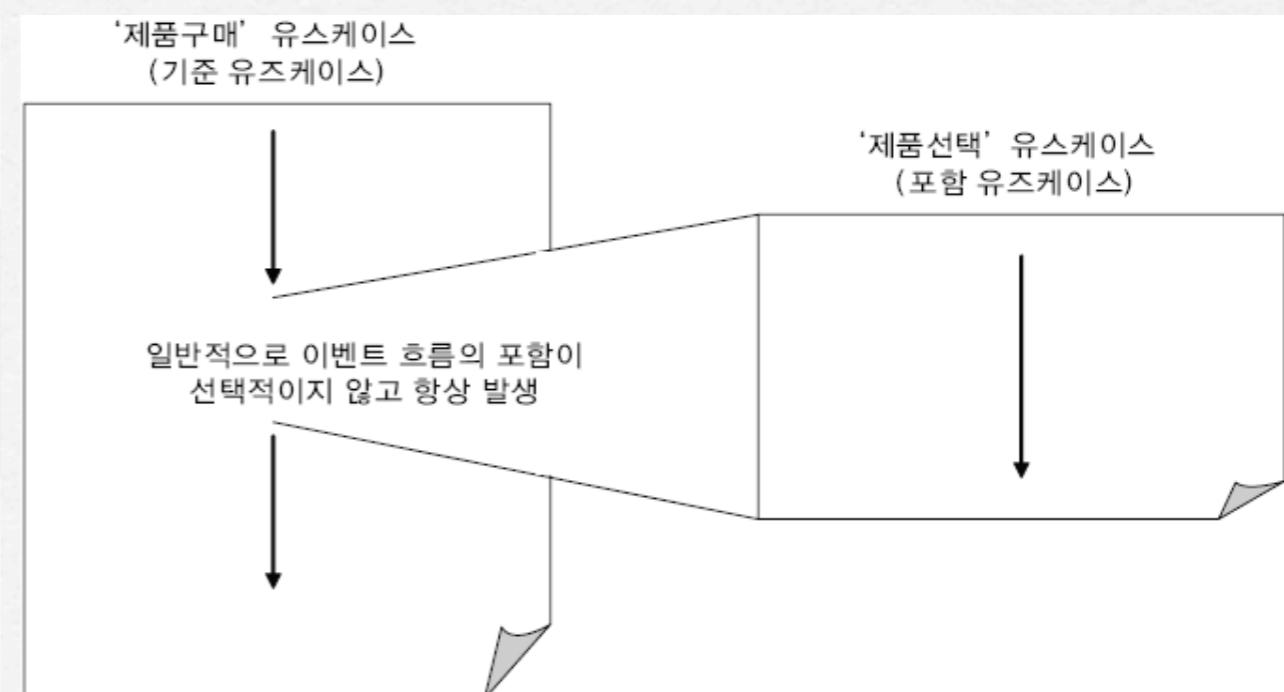
[그림 3-6] 도서예약 유스케이스의 확장

# 확장 유스케이스와 포함 유스케이스

- 확장관계에 있는 유스케이스 사이의 이벤트 흐름은 포함관계에 있는 유스케이스 사이의 이벤트 흐름과 유사
- 기준 유스케이스의 이벤트 흐름이 수행되었다가 확장점을 만나면 지정된 유스케이스의 이벤트 흐름으로 분기



[그림 3-7] 확장관계의 이벤트 흐름



[그림 3-8] 포함관계의 이벤트 흐름

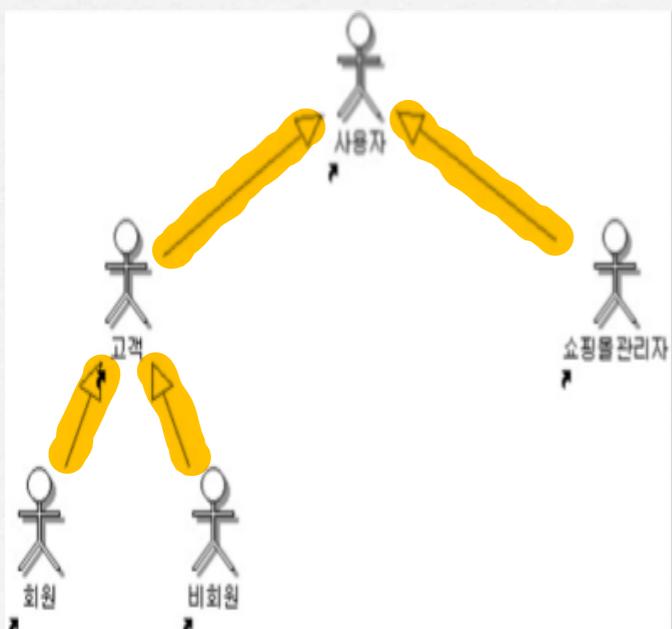
- 포함 유스케이스로의 분기는 필수적, 반면에 확장 유스케이스로 분기는 선택적
- 포함된 유스케이스는 대부분의 경우에 포함된 유스케이스의 수행 결과에 따라서 기준 유스케이스의 이벤트 흐름이 영향을 받는다
- 확장 유스케이스는 선택적으로 이벤트 흐름이 수행되고, 기준 유스케이스 이 후의 이벤트 흐름은 확장 유스케이스의 수행 결과에 의존하지 않는다.

[표 3-3] 포함관계와 확장관계의 비교

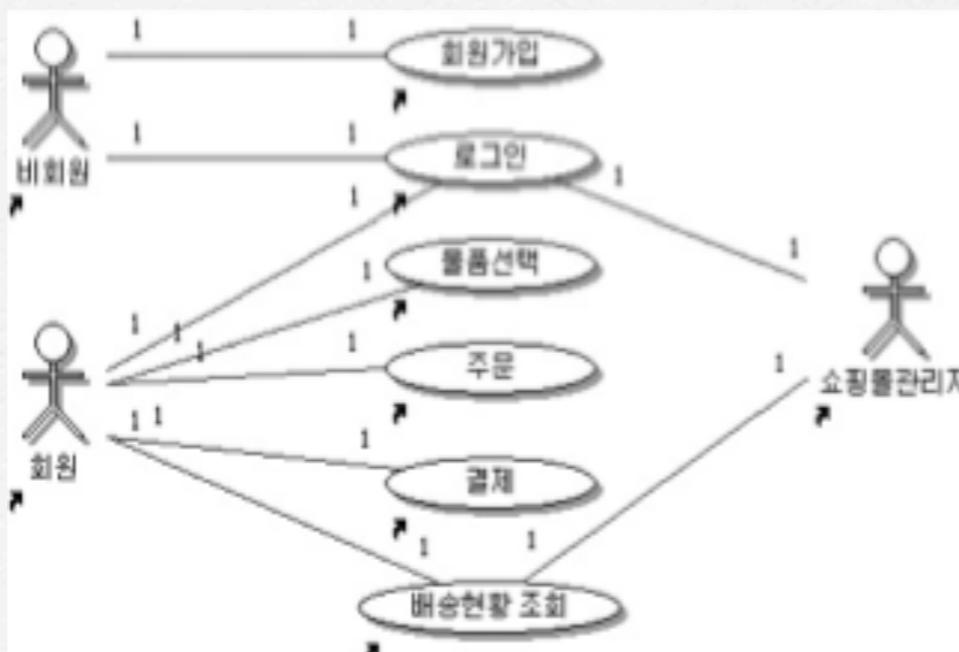
	포함관계	확장관계
목적	<ul style="list-style-type: none"> <li>• 여러 유스케이스에 공통적인 기능을 표현하기 위해 사용된다.</li> </ul>	<ul style="list-style-type: none"> <li>• 기준 유스케이스에 부가적으로 추가된 기능을 표현하기 위해 사용된다.</li> </ul>
이벤트 흐름	<ul style="list-style-type: none"> <li>• 포함된 유스케이스로의 이벤트 흐름 분기가 필수적이다.</li> <li>• 기준 유스케이스 이후의 이벤트 흐름이 포함된 유스케이스의 수행 결과에 의존한다.</li> </ul>	<ul style="list-style-type: none"> <li>• 확장 유스케이스는 확장 유스케이스에 기술된 조건에 따라 선택적으로 수행된다.</li> <li>• 기준 유스케이스 이후의 이벤트 흐름이 확장 유스케이스의 결과에 의존하지 않는다.</li> </ul>

시험 무조건 낸다고 함. (일반화관계 화살표 잘 볼 것)

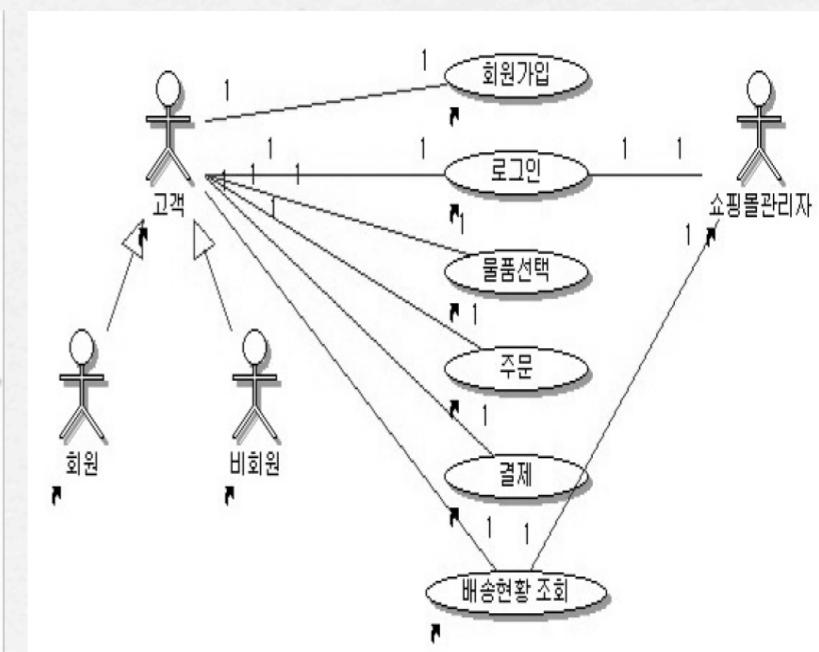
- 액터 사이의 일반화(Generalization)는 클래스 사이의 일반화와 비슷한 개념
- 추상적인 액터와 좀 더 구체적인 액터 사이에 맺어주는 관계로서
  - “한 액터가 다른 액터의 일종이다” 또는
  - “한 액터도 다른 액터에 해당된다”의 의미가 되는 두 액터를 일반화관계로 연결
- 일반화관계를 액터에 적용하면 유스케이스 다이어그램에서 사용되는 여러 액터들의 의미를 좀 더 명확하게 하고 다이어그램도 보다 간결하게 작성



[그림 3-9] 액터의 일반화관계



[그림 3-10] 일반화를 사용하지 않은 경우



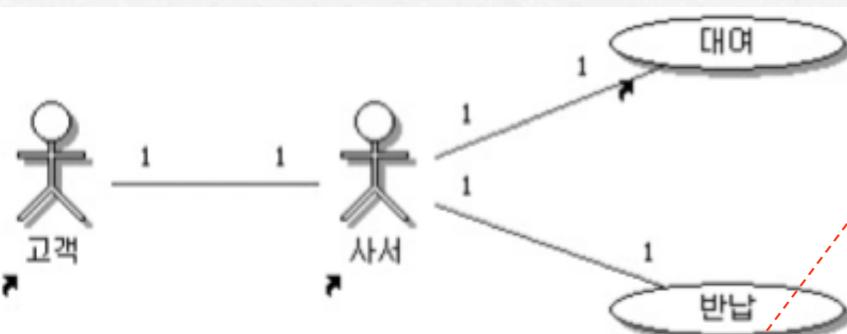
[그림 3-11] 일반화관계를 적용한 경우

- 2개 이상의 유스케이스 이벤트 흐름에서 중복적인 부분이 발생하는 경우
- . 유스케이스간 포함관계를 설정하여 해결

[표 3-4] 대여 및 반납 유스케이스의 이벤트 흐름

대여 유스케이스 이벤트 흐름	반납 유스케이스 이벤트 흐름
1. 사서는 고객의 아이디와 패스워드를 도서관 시스템에 입력한다.	
2. 도서관 시스템은 아이디와 패스워드로 고객의 정보를 확인하고 업무 종류를 선택할 수 있는 화면을 보여준다.	
3. 사서는 '대여'를 선택한다.	3. 사서는 '반납'을 선택한다.
4. 도서관 시스템은 사서에게 도서 번호를 입력하기 위한 화면을 보여준다.	
5. 사서는 도서 번호를 입력한다.	
6. 도서관 시스템은 도서 목록에서 도서 번호가 유효한지 확인한다.	7. 도서관 시스템은 도서를 반납 처리한다.
7. 도서관 시스템은 고객이 대여가 가능한지 확인한다.	
8. 고객에 대한 대여의 가능 여부를 표시한다.	
9. 도서관 시스템은 도서를 대여 처리한다.	

[그림 3-13] 도서관 시스템의 유스케이스 다이어그램



[표 3-5] '고객 확인' 유스케이스와 '도서 번호 입력' 유스케이스

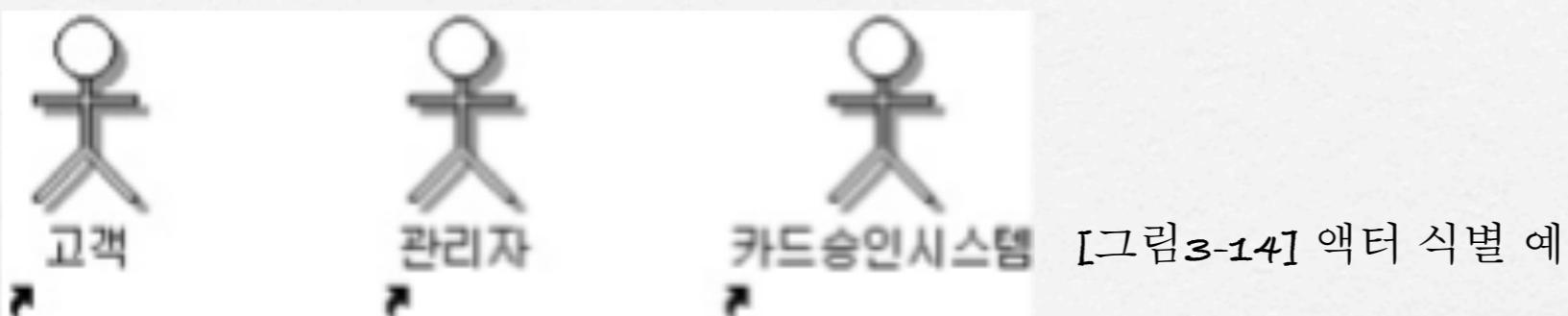
고객 확인	1. 사서는 고객의 아이디와 패스워드를 도서관 시스템에 입력한다. 2. 도서관 시스템은 아이디와 패스워드로 고객의 정보를 확인하고 업무 종류를 선택할 수 있는 화면을 보여준다.
도서 번호 입력	1. 도서관 시스템은 사서에게 도서의 번호를 입력하기 위한 화면을 보여준다. 2. 사서는 도서 번호를 입력한다. 3. 도서관 시스템은 도서 목록에서 도서 번호가 유효한지 확인한다.

# 유스케이스 다이어그램을 만드는 단계

- 1단계 : 시스템 상황을 확인
- 2단계 : 액터 식별.
  - 행위자와 그들의 책임을 확인
- 3단계 : 유스케이스 식별
  - 특정한 목적의 관점에서 볼 때 쓰임새와 시스템의 특성을 확인
- 4단계 : 유스케이스 다이어그램 작성
  - 행위자와 유스케이스에서 정제할 부분이 있는지 평가
  - 유스케이스에서 <<include>> 의존성이 있는지 평가
  - 유스케이스에서 <<extend>> 의존성이 있는지 평가
  - 행위자와 유스케이스를 일반화(또는 공유)할 수 있는지 평가
- 5단계 : 유스케이스 명세서 작성
  - 유스케이스명, 액터명 및 개요를 기술
  - 사전 및 사후 조건과 제약사항들을 식별
  - 작업(정상, 대치, 예외)흐름과 시나리오를 도출
  - 유스케이스 흐름에서 포함이나 확장 유스케이스로 구조화
- 6단계 : 유스케이스 실체화
  - 구현 시스템의 논리적 구성 요소인 클래스를 식별하고 통신관계를 파악하는 데 중점을 두는 과정

# [예제] 비디오숍 관리 시스템 문제

- 1단계 : 시스템 상황 분석(문제 기술서 작성)
- 2단계 : 액터 식별

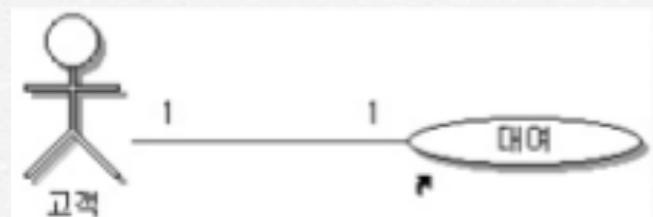


- 3단계 : 유스케이스 식별

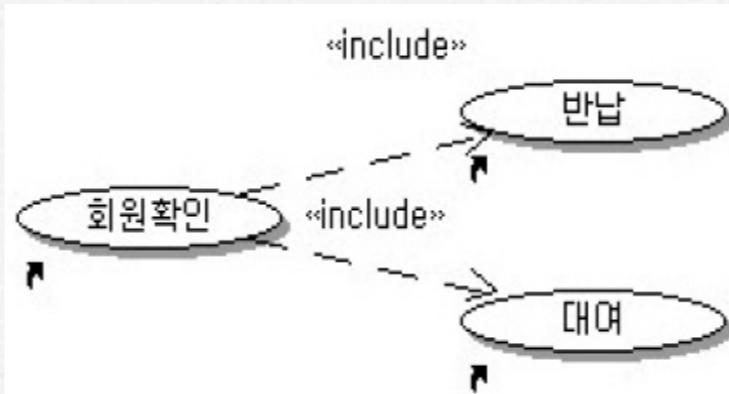


[그림 3-15] 유스케이스 식별 예

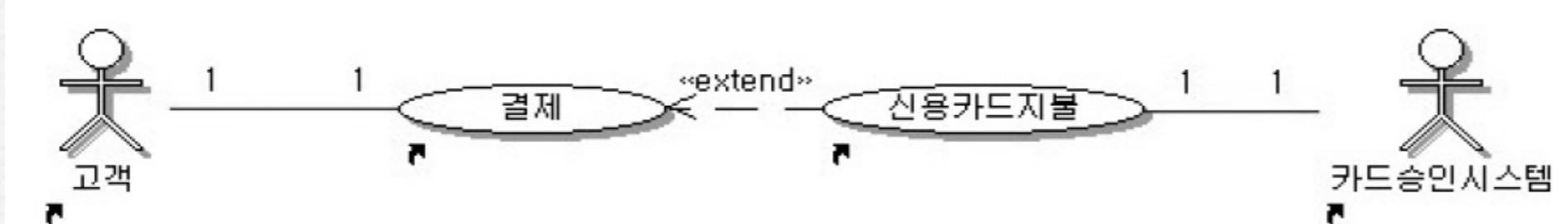
## □ 4단계 : 유스케이스 다이어그램 작성



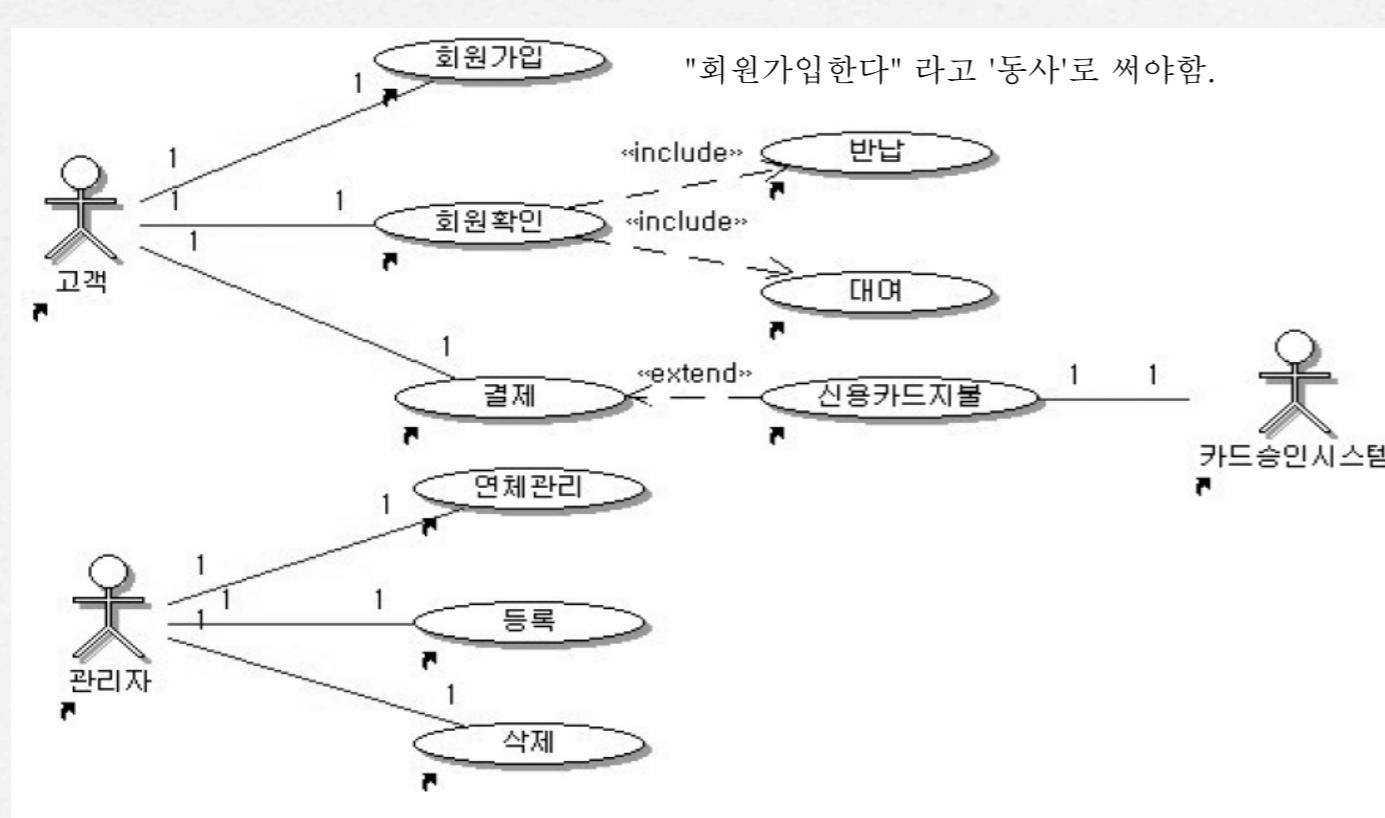
[그림 3-16] 액터와 유스케이스 사이의 관계 설정



[그림 3-17] <<include>> 관계



[그림 3-18] <<extend>> 관계



[그림 3-19] 비디오숍 관리 시스템의 유스케이스 다이어그램

## □ 5단계 : 유스케이스 명세서 작성

- 유스케이스명
- 액터명
- 유스케이스 개요
- 이벤트 흐름

- 정상 흐름(Normal Flow) : 유스케이스가 사건 주 흐름을 표현하는 절차이다.
- 선택 흐름(Alternative Flow) : 유스케이스 내의 사건 주 흐름 외에 수행되는 절차를 표현한다.

- 유스케이스명 : 회원가입
- 액터명 : 고객(비회원)
- 유스케이스 개요 및 설명 : 고객이 비디오숍 관리 시스템을 사용하기 위해 회원가입을 하는 유스케이스이다.
- 사전 조건 : 회원에 가입되어 있지 않은 상태여야 한다.
- 이벤트 흐름

- 정상 흐름

- ① 기존 가입 회원인지 주민등록번호를 검색하여 확인한다(시스템).
- ② 회원가입을 요청한다(액터),
- ③ 회원약관을 보여준다(시스템),
- ④ 회원약관에 동의한다(액터),
- ⑤ 회원정보 입력항목을 보여준다(시스템),
- ⑥ 회원정보, 항목(이름, 주민등록번호, 전화번호, 핸드폰번호, 이메일 등)을 입력하고 등록 요청을 한다(액터),
- ⑦ 입력된 정보를 확인한다(시스템),
- ⑧ 회원정보를 저장, 등록을 완료한다(시스템).

- 선택 흐름

- ▶ 기존에 가입되어 있는 회원인 경우 “이미 가입된 회원입니다”라는 메시지를 보여준다.
- ▶ 회원약관에 동의하지 않을 경우 약관 동의하에 회원가입 가능 오류 메시지를 보여주고 동의를 요청한다.
- ▶ 회원정보 입력항목 중 입력하지 않은 항목이 있을 경우 오류 메시지를 띄우고 재입력을 요청한다.
- ▶ 등록번호의 형식이 틀렸을 경우 메시지를 보여주고 재입력을 요청한다.

## □ 6단계 : 유스케이스 실체화

- 유스케이스 실체화는 도출된 기능 중심의 유스케이스를 구현 시스템의 구성 요소로 구체화시키는 작업
- 실체화 과정을 통해 명세서 중심의 유스케이스를 구현 시스템의 논리적 구성 요소인 클래스로 식별하고 통신관계를 파악
- UML의 순차 다이어그램과 활동 다이어그램이 사용

[표 3-6] 요구사항 정의 활동의 산출물

산출물	UML 다이어그램	필수 여부
요구사항 모델	유스케이스 모델	유스케이스 다이어그램 필수
	유스케이스 명세서	이용 안 함 필수
	이벤트 흐름 모델	순차 다이어그램 선택
	화면 흐름 모델	활동 다이어그램 선택

# [예제] 재고관리 시스템

- 1단계 : 시스템 상황 분석(문제 기술서 작성)
  - 한 인터넷 쇼핑몰에서 원활한 창고의 재고관리를 위해 재고관리 시스템 구축
  - 재고관리 시스템은 크게 입고관리 기능, 출고관리 기능, 현황관리 기능 제공
- 입고관리 기능 :
  - 창고로 입고된 상품을 현황관리에 추가
  - 입고는 업체로부터 새로운 상품을 입고 받거나 고객의 반품에 의한 것
  - 입출고 담당자는 입고된 제품 상태를 파악하고 불량조치하여 업체에 반품
- 출고관리 기능 :
  - 창고로 출고된 상품을 현황관리에서 뺀다.
  - 출고는 고객이 구매한 상품을 발주하는 것과 판매하고 남은 상품을 통신업체로 반품한 것
- 현황관리 기능 :
  - 입고와 출고된 현황을 실시간으로 인터넷 쇼핑몰에 업데이트하는 기능
  - 입출고 담당자는 현황을 조회할 수 있고 현황관리 담당자는 재고현황을 관리하여 협력업체에 주문 혹은 반품을 요청하고 쇼핑몰에 업데이트

## □ 2단계 : 액터 식별



[그림 3-20] 재고관리 시스템의 액터

## □ 3단계 : 유스케이스 식별

기능 범주	사용자	기능(유스케이스)
입고관리	입출고 담당자	통신업체 재품 입고 기능 고객 반품 입고 기능
출고관리	입출고 담당자	통신업체 반품 출고 기능 고객 출고 기능 발주 기능
현황관리	입출고 담당자	현황 조회
	현황관리 담당자	현재 현황 등록 기능(업데이트) 통신업체 주문 기능
	쇼핑몰 시스템	실시간 현황 반영 기능

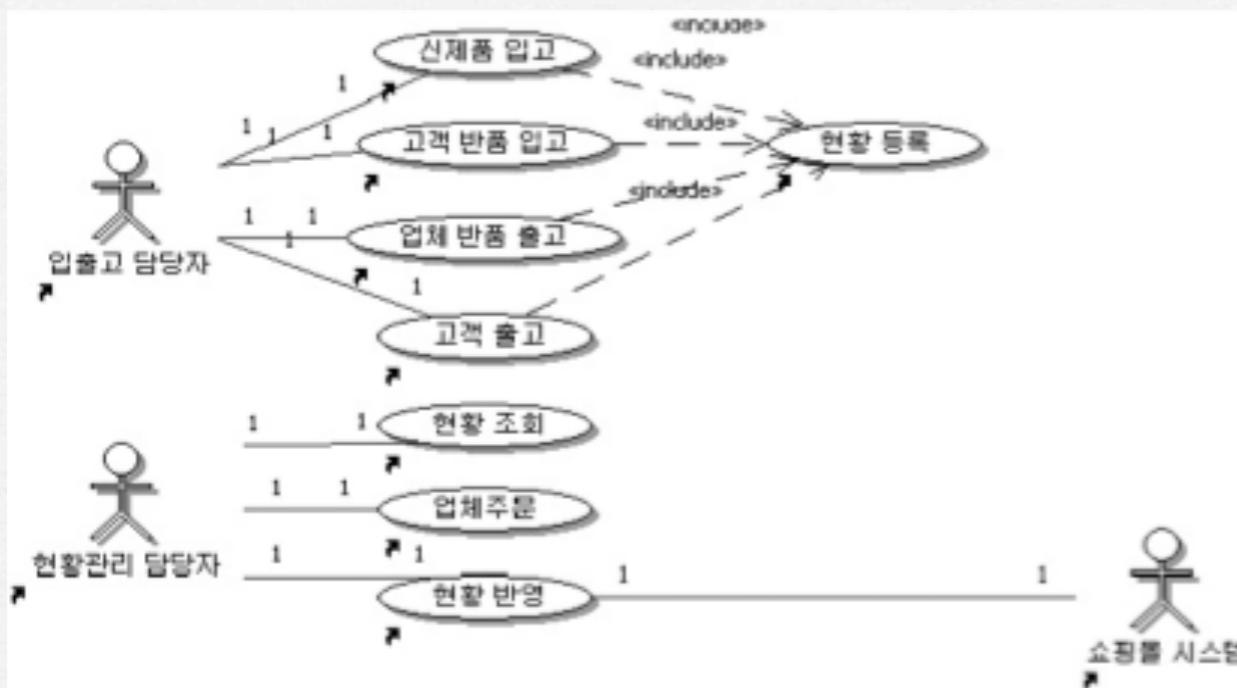
[표 3-7] 재고관리 시스템의 사용자별 기능



[그림 3-21] 재고관리 시스템의 유스케이스

□ 4단계 : 유스케이스 다이어그램 작성

□ 5단계 : 유스케이스 명세서 작성



[그림 3-22] 재고관리 시스템의 유스케이스

- ▶ 유스케이스명 : 신제품 입고
- ▶ 액터명 : 입출고 담당자
- ▶ 유스케이스 개요 및 설명 : 입출고 담당자는 신제품이 입고되면 제품의 상태를 확인하고 입고 또는 반품시킨다.
- ▶ 사전 조건 : 현황관리 담당자가 통신업체에 주문한 제품이다.

□ 이벤트 흐름

- 정상 흐름

- ① 통신업체로부터 제품 입고를 요구한다.
- ② 입출고 담당자는 제품이 주문한 제품인지 확인한다.
- ③ 제품의 상태를 파악한다.
- ④ 제품을 입고하고 제품 목록을 현황관리 담당자에게 알린다.

- 선택 흐름

- ① 제품에 하자가 발생하면 현황관리 담당자에게 하자를 알리고 반품한다.

# 오늘의 강의 요약

- UML과 객체지향 모델링
- UML의 구성
- 유스케이스 다이어그램

# Q & A

More Information?

오유수, Ph.D.

[yoosoo.oh@daegu.ac.kr](mailto:yoosoo.oh@daegu.ac.kr)

공대5호관 5506D호

