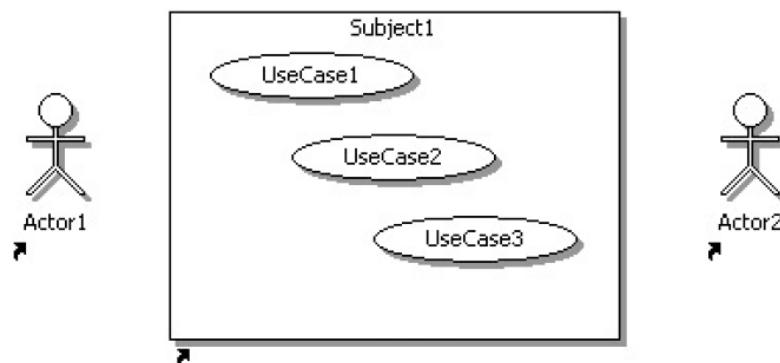


# UseCase Diagram

# 유스케이스와 UML 표기 기법

- ❖ 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 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 1] 시스템의 경계

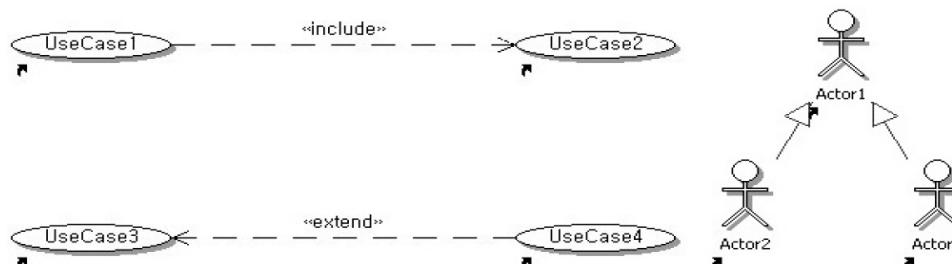
# 유스케이스와 유스케이스 간의 관계

- ❖ 유스케이스의 역할 : 사용자의 시점에서 시스템을 모델링한다는 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 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

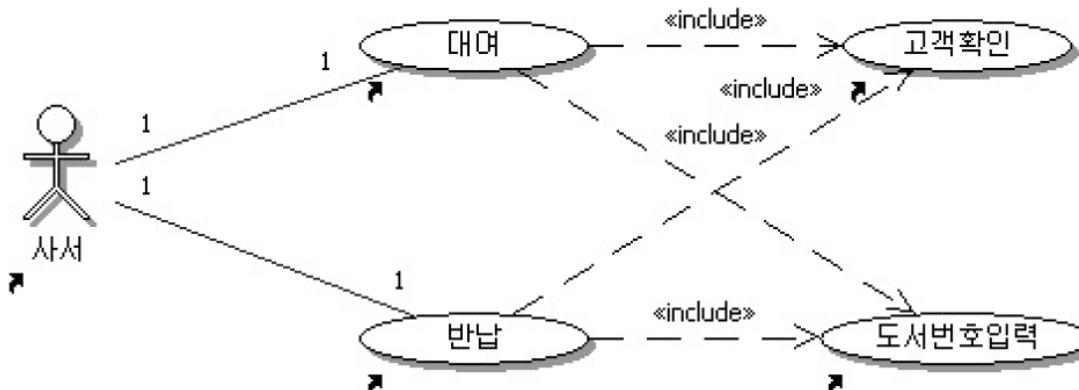


[그림 3] 유스케이스 관계

# 유스케이스와 유스케이스 간의 관계

## ◆ 포함관계

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



[그림 4] 도서관 시스템의 포함관계

[표 1] 포함관계를 이용한 ‘대여’ 와 ‘반납’ 유스케이스의 이벤트 흐름

대여 유스케이스 이벤트 흐름	반납 유스케이스 이벤트 흐름
<ol style="list-style-type: none"><li>1. ‘고객확인’ 유스케이스를 포함한다.</li><li>2. 사서는 ‘대여’를 선택한다.</li><li>3. ‘도서번호입력’ 유스케이스를 포함한다.</li><li>4. 도서관 시스템은 고객이 대여가 가능한지 확인한다.</li><li>5. 고객에게 대여를 가능 여부를 표시한다.</li><li>6. 도서관 시스템은 도서를 대여 처리한다.</li></ol>	<ol style="list-style-type: none"><li>1. ‘고객확인’ 유스케이스를 포함한다.</li><li>2. 사서는 ‘반납’을 선택한다.</li><li>3. ‘도서번호입력’ 유스케이스를 포함한다.</li><li>4. 도서관 시스템은 도서를 반납 처리한다.</li></ol>

# 유스케이스와 유스케이스 간의 관계

## ◆ 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] 금액계산 유스케이스의 확장

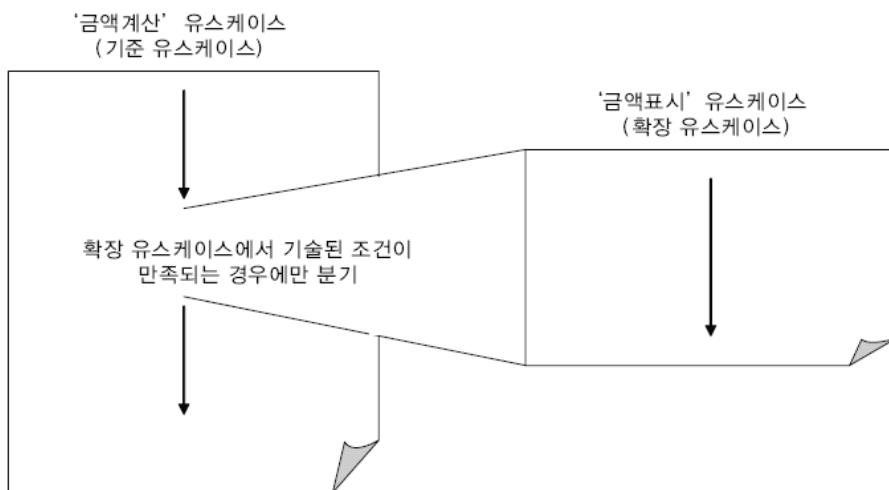
[표 2] 금액계산 유스케이스와 금액표시 유스케이스의 이벤트 흐름

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>

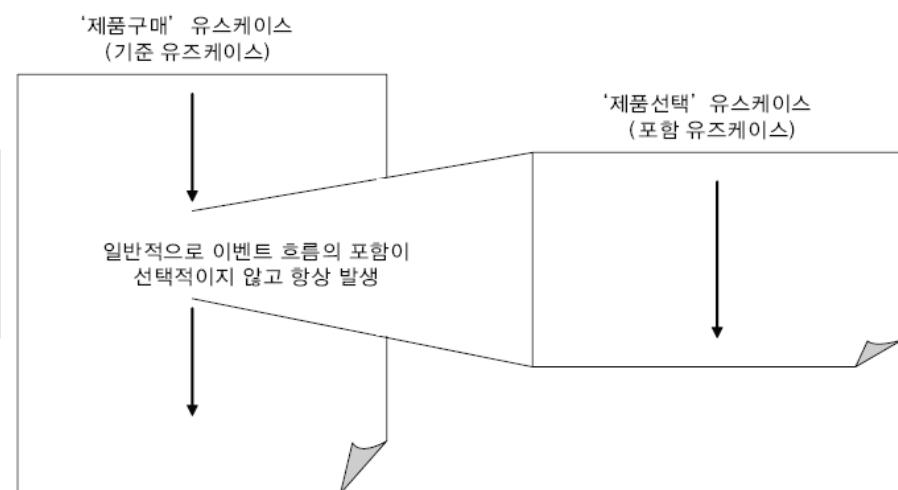
# 유스케이스와 유스케이스 간의 관계

## ◆ 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 7】: 기준 유스케이스와 확장 유스케이스 사이의 이벤트 흐름 수행관계
- 【그림 8】: 포함관계가 있는 두 유스케이스 사이의 이벤트 흐름



[그림 7] 확장관계의 이벤트 흐름



[그림 8] 포함관계의 이벤트 흐름

# 유스케이스와 유스케이스 간의 관계

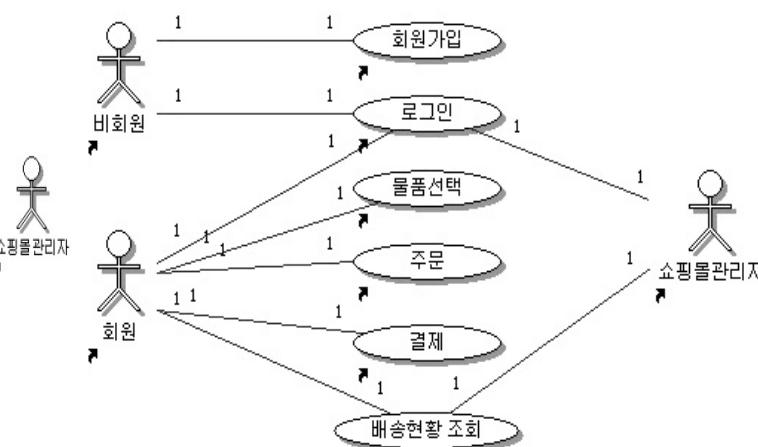
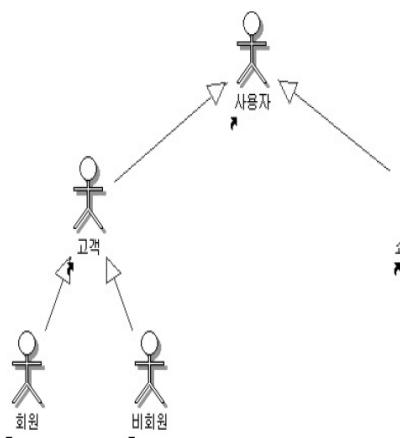
- ❖ 포함 유스케이스로의 분기는 필수적, 반면에 확장 유스케이스로 분기는 선택적
  - 포함된 유스케이스는 대부분의 경우에 포함된 유스케이스의 수행 결과에 따라서 기준 유스케이스의 이벤트 흐름이 영향을 받는다
  - 확장 유스케이스는 선택적으로 이벤트 흐름이 수행되고, 기준 유스케이스 이 후의 이벤트 흐름은 확장 유스케이스의 수행 결과에 의존하지 않는다.

[표 3] 포함관계와 확장관계의 비교

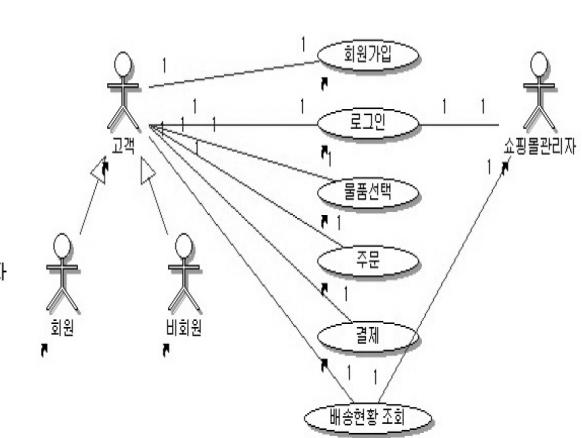
	포함관계	확장관계
목적	<ul style="list-style-type: none"><li>• 여러 유스케이스에 공통적인 기능을 표현하기 위해 사용된다.</li></ul>	<ul style="list-style-type: none"><li>• 기준 유스케이스에 부가적으로 추가된 기능을 표현하기 위해 사용된다.</li></ul>
이벤트 흐름	<ul style="list-style-type: none"><li>• 포함된 유스케이스로의 이벤트 흐름 분기가 필수적이다.</li><li>• 기준 유스케이스 이후의 이벤트 흐름이 포함된 유스케이스의 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>

# 액터의 일반화

- ❖ 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 9] 액터의 일반화관계



[그림 10] 일반화를 사용하지 않은 경우

[그림 11] 일반화관계를 적용한 경우

# 유스케이스 모델링 단계

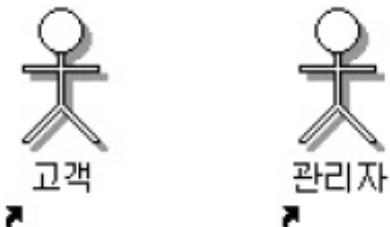
## ◆ 유스케이스 다이어그램을 만드는 단계

- **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

# 유스케이스 모델링 예제

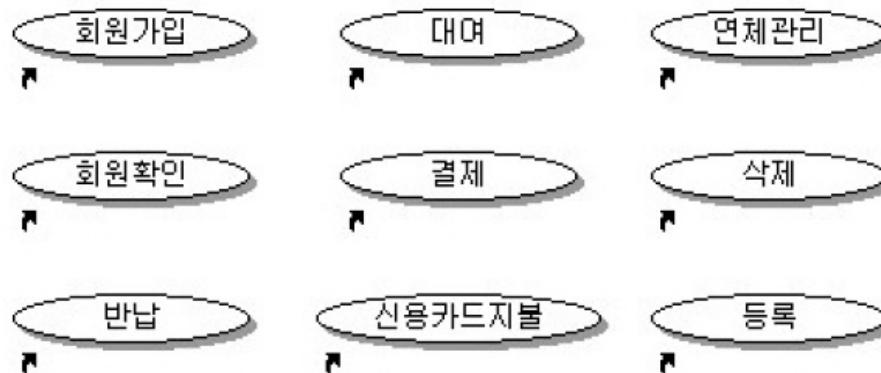
## ▶ 비디오숍 관리 시스템 문제

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



[그림 14] 액터 식별 예

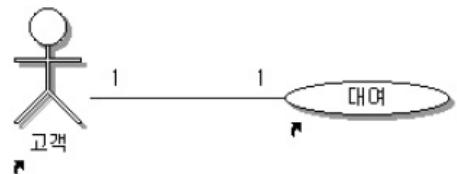
- 3단계 : 유스케이스 식별



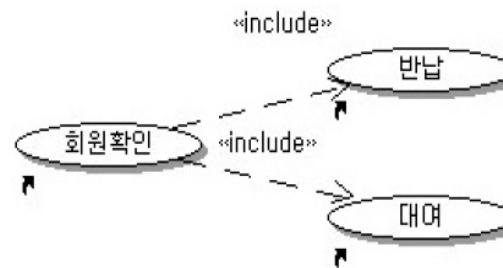
[그림 15] 유스케이스 식별 예

# 유스케이스 모델링 예제

## ◆ 4단계 : 유스케이스 다이어그램 작성



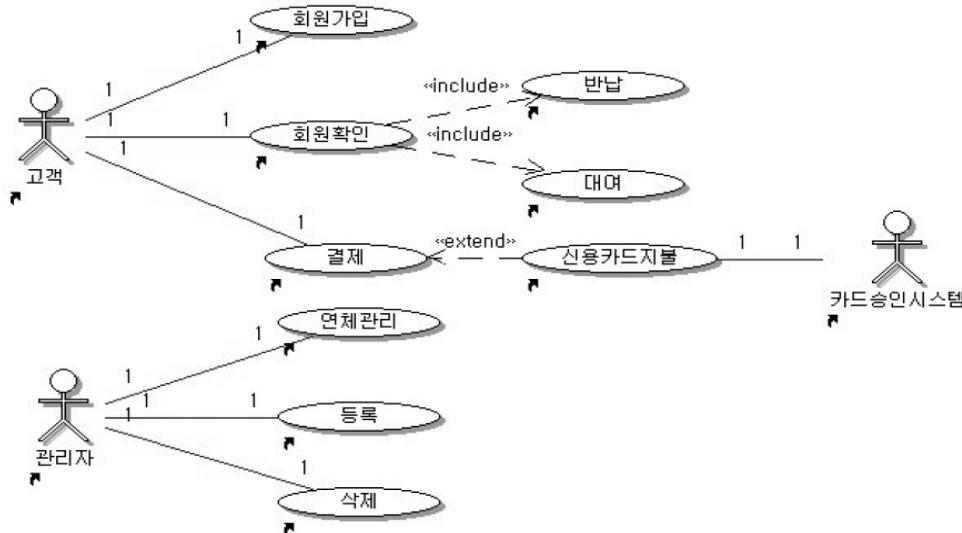
[그림 16] 액터와 유스케이스 사이의 관계 설정



[그림 17] <<include>> 관계



카드승인시스템 [그림 18] <<extend>> 관계



[그림 19] 비디오숍 관리 시스템의  
유스케이스 다이어그램

# 유스케이스 모델링 예제

## ◆ 5단계 : 유스케이스 명세서 작성

- 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(액티).
    - ③ 회원약관을 보여준다(시스템).
    - ④ 회원약관에 동의한다(액티).
    - ⑤ 회원정보 입력항목을 보여준다(시스템).
    - ⑥ 회원정보, 항목(이름, 주민등록번호, 전화번호, 핸드폰번호, 이메일 등)을 입력하고 등록 요청을 한다(액티).
    - 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.

# 수강신청하기 - 유스케이스 시나리오

## ❖ Brief Description

- 학생이 금번학기에 개설된 강좌 중 수강하고자 하는 과목을 신청한다.

## ❖ Flow of Events

### ■ **Basic Flow**

1. 학생이 수강신청하기 메뉴를 요청함으로써 이 **Use Case**는 시작한다.
2. 학생은 “조회하기” 버튼을 클릭하여 수강정보를 요청한다.(E5)
3. 금번학기에 개설된 강의정보(기신청학점수, 학기, 강좌번호, 강좌이름, 담당학과, 학점, 강좌설명, 신청가능여부, 수강신청여부)를 출력한다.(E1)
4. 학생이 수강신청 하고자 하는 수강과목을 선택한다(E2,E3,E4)
5. 선택된 강의정보를 “수강신청하기” 버튼을 클릭하여 신청한다.(E3,E5)
6. “\*건【성공】 \*건【실패】로 수강과목 신청이 처리되었습니다.”라는 메시지를 출력한다.
7. 신청된 수강과목 목록을 출력한다.

### ■ **Alternative Flow**

## ■ *Exception Flow*

- **E1 : 검색 실패**
  - 금번학기 수강신청 가능한 개설된 강의목록이 비어있는 경우 등록된 에러 메시지를 출력하고 **ERR\_FFD\_NO\_CLASS**에러를 반환한다.
- **E2 : 최대수강신청 학점 초과 실패**
  - 학생이 최대신청 수강학점을 초과하였을 경우 등록된 에러 메시지를 출력하고 **ERR\_FFD\_OVERCLASS**에러를 반환한다.
- **E3 : 최대수강신청인원 초과 실패**
  - 최대수강신청인원 초과하였을 경우 등록된 에러 메시지를 출력하고 **ERR\_FFD\_MAXCLASS**에러를 반환한다.
- **E4 : 수강신청 중복 오류 실패**
  - 중복 수강신청이 되었을 경우 등록된 에러 메시지를 출력하고 **ERR\_FFD\_UPCLASS**에러를 반환한다
- **E5 : 강의정보신청 오류 실패**
  - 강의정보 신청 시 시스템 오류가 발생되었을 경우 등록된 에러 메시지를 출력하고 **ERR\_FFD\_SYSTEM**에러를 반환한다.

❖ Conditions

■ ***Pre-Conditions***

- 학생은 금번학기 재학생으로 등록되어 있어야 한다.
- 학생은 이미 로그인 과정을 통해서 본인 인증이 되어있는 상태이다.

■ ***Post-Conditions***

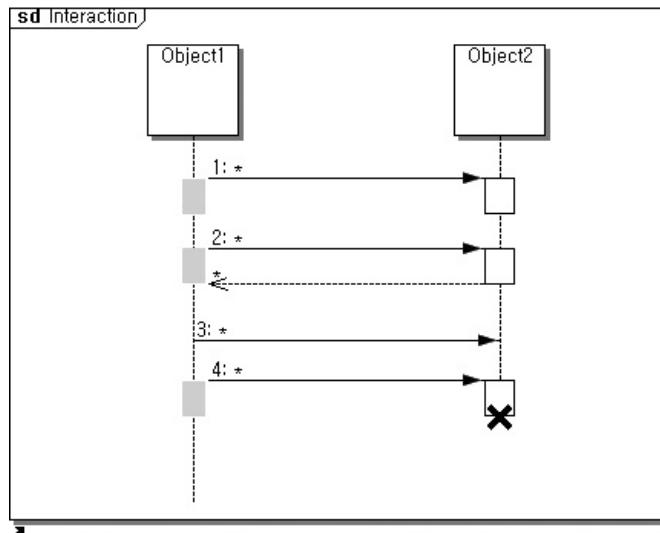
- 학생이 수강 신청을 완료한다.

# Sequence Diagram

# Sequence Diagram 개요

## ❖ Sequence Diagram

- 객체간의 동적 상호작용을 시간적 개념을 중심으로 모델링하는 과정
- 다이어그램의 수직방향이 시간의 흐름을 나타낸다.
- 객체는 다른 객체와 메시지를 주고받는다.
- 각 메시지는 시간의 흐름에 따라 순서를 정하게 된다.



[그림 1] Sequence Diagram의 예

# Sequence Diagram 개요

## ❖ 요구사항 정의 단계에서의 순차 다이어그램

- 대여' 유스케이스 이벤트 흐름을 **Sequence** 다이어그램으로 표현
- **Sequence** 다이어그램은 텍스트로 기술된 이벤트 흐름에 추가적인 정보를 기술하는 것은 아니며, 시각적으로 보다 쉽게 이벤트 흐름을 이해하도록 돋기 위한 것
- 유스케이스별로 이벤트 흐름을 순차 다이어그램으로 모두 표현할 수 있음

□ 유스케이스명 : 대여
□ 액터명 : 사용자(고객, 관리자)
□ 유스케이스 개요 및 설명 : 고객이 비디오를 선택하면 선택된 비디오 코드를 등록하고 대여한다.
□ 사전 조건 : 고객은 회원가입이 되어 있어야 한다.
□ 이벤트 흐름
- 요구사항(기능적) ① 사용자는 시스템에 접속한다. ② 비디오 관리 시스템 화면을 보여준다. ③ 시스템 화면에서 비디오 대여 버튼을 누른다. ④ 시스템은 비디오 코드 입력 화면을 보여준다. ⑤ 사용자는 비디오 레이블에 붙어 있는 코드를 입력한다. ⑥ 시스템은 회원정보가 유효한지의 유효성 메시지를 사용자에게 보여준다. 유효성이 확인되면 대여목록에 고객정보와 비디오 코드를 등록한다.
- 요구사항 (비기능적) • 유효하지 않은 비디오 코드인 경우 a. 시스템은 입력된 비디오 코드가 잘못된 값임을 알려준다. b. 사용자는 다시 비디오 코드를 입력한다. • 고객이 연체인 상태인 경우 a. 시스템은 고객의 연체된 날짜와 비디오 코드를 알려준다. b. 사용자는 연체금과 비디오를 반납 등록하고 유효성 확인을 다시 시도한다.
- 예외 흐름 : 해당사항 없음

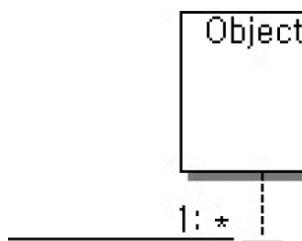


[그림 2] 대여 유스케이스의 이벤트 흐름 모델

# Sequence Diagram 설명

## ◆ 객체

- 객체는 순차 다이어그램의 가장 윗부분에 위치하며, 왼쪽에서 오른쪽으로 배열
- 생명선 : 객체로부터 아래로 뻗어나가는 점선
- 활성화(**Activation**)
  - 생명선을 따라 작은 사각형이 드문드문 나타나는 부분
  - 객체가 수행하는 오퍼레이션이 실행되고 있음을 나타낸다.
  - 활성화 사각형의 길이는 오퍼레이션의 실행 소요 시간을 나타낸다.
  - 활성화 사각형의 길이가 특정 시간 단위를 의미하지 않음



[그림 3] 객체, 생명선, 활성화

# Sequence Diagram 설명

## ◆ 메세지

- 한 객체에서 다른 객체로 전송되는 메시지는 한 객체의 생명선에서 다른 객체의 생명선으로 이동하는 것을 의미
- 화살표로 메시지를 표현하는데, 화살표의 머리 모양이 메시지의 형태를 좌우
- 호출(**Call**) 메시지
  - 실선의 끝에 속을 칠한 화살표([그림 4] (a))
- 답신 메시지
  - 점선 끝에 화살표([그림 4] (b))를 붙인 모양
  - 수신 객체로부터 답신 메시지를 요청하는 경우, 답신 메시지의 표시를 생략



(a) 호출(동기) 메시지



(b) 답신 메시지

[그림 4] 동기 메시지

# Sequence Diagram 설명

## ◆ 메세지

### ■ 동기(**Synchronous**) 메시지/

- 송신 객체가 수신 객체를 기다려주는 메시지

### ■ 비동기(**Asynchronous**) 메시지/

- 송신 객체가 보내는 메시지로서 수신 객체의 오퍼레이션을 실행하게 하는 것은 동일하지만, 오퍼레이션이 완료될 때까지 송신 객체가 기다리지 않는다.
- [그림 5]과 같이 실선에 화살표모양을 붙여 나타낸다

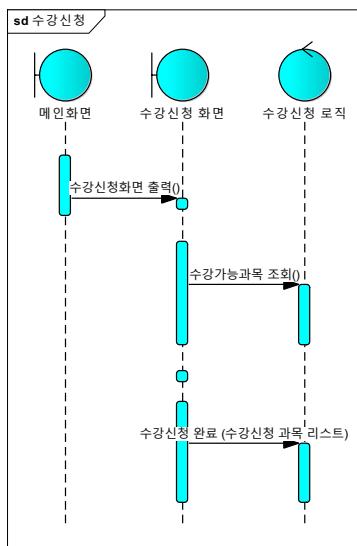


[그림 5] 비동기 메시지

# Sequence Diagram 설명

## ◆ 시간

- 순차 다이어그램은 시간을 수직 방향으로 나타낸다.
- 시간은 가장 윗부분에서 아래를 향해 흐르기 시작
- 왼쪽에서 오른쪽 방향은 객체의 배열을, 위에서 아래는 시간의 흐름을 나타낸다
- 
- 화살표 머리가 달린 실선은 객체의 생명선 사이를 연결하여 한 객체에서 다른 객체로 메시지가 전송됨을 나타낸다.
- 객체는 자기 자신에게도 메시지를 보낼 수 있다.

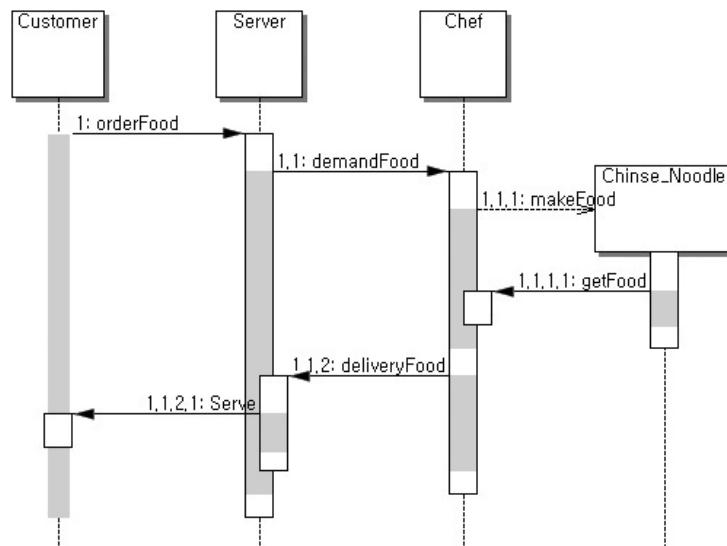


[그림 6] Sequence 다이어그램의 기호

# Sequence Diagram 예제

## ◆ 식당 음식 주문

- 고객이 식당에서 음식을 주문할 때에 음식이 나오기까지의 흐름을 간단하게 나타낸 것
- 첫 번째 메시지는 액터인 고객(**Customer**)이 종업원(**Server**)에게 자장면을 주문(**order**)한다고 가정하였을 때, 종업원은 요리사(**Chef**)에게 음식을 요청(**demandFood**)
- 요리사는 자장면을 생성(**makeFood**)하게 되고, 이를 종업원에게 전달
- 전달받은 종업원은 고객에게 완성된 자장면을 전달

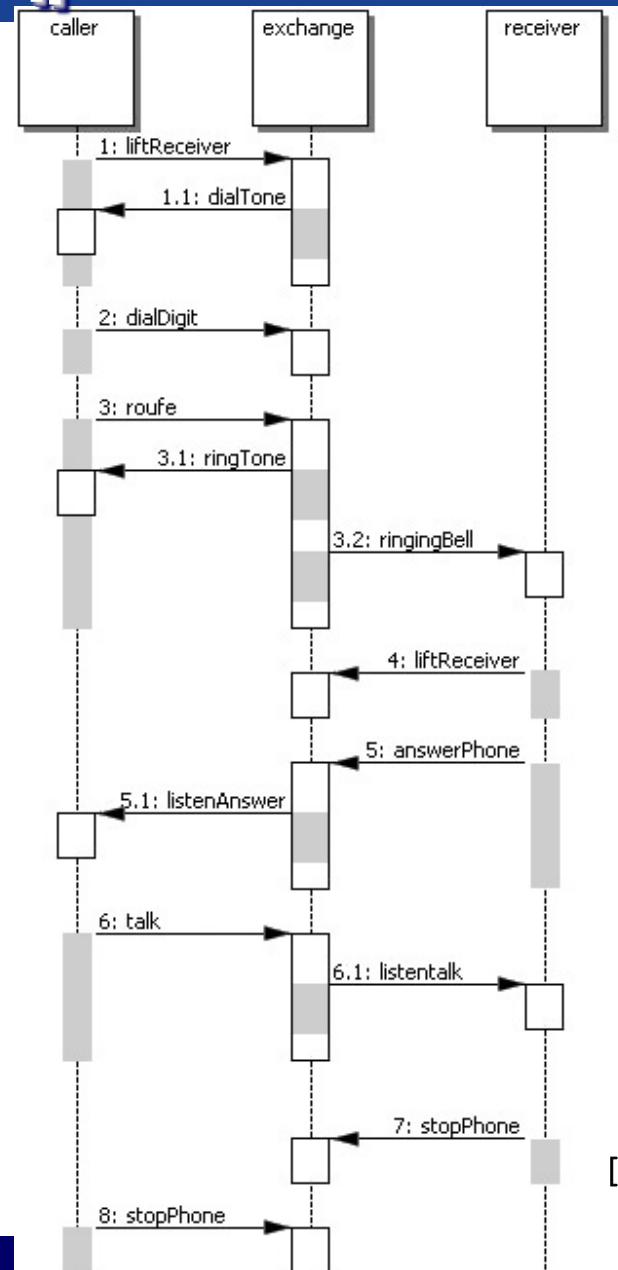


[그림 7] 식당 음식 주문

# Sequence Diagram 예제

## ◆ 전화 통화

1. 전화 통화에서 발신자(caller)와 수신자(receiver)가 상호 신호를 교환하며 연락하는 흐름을 나타내는 것
2. 발신자가 수신자에게 전화하기 위해 수화기를 들면(liftReceiver) 전화기의 신호음(dialTone)이 전달되고, 이에 디아일을 둘리고(dialDigit) 신호를(roufe) 모두 교환기(exchange)에 전달
3. 교환기에서는 발신자의 신호를 받게 되어 발신자에게는 신호음이, 수신자에게는 전화벨이 각각 전달
4. 수신자는 전화를 받게(liftReceiver) 되며, 수신자는 응답(answerPhone)
5. 발신자는 수신자의 응답을 듣게(listenAnswer) 되며, 이후 발신자와 수신자는 말하고(talk), 듣게(listenTalk) 된다.
6. 이야기가 끝나면 수신자와 발신자는 수화기를 내려 놓는다(stopPhone)



[그림 8] 전화 통화

# Activity Diagram

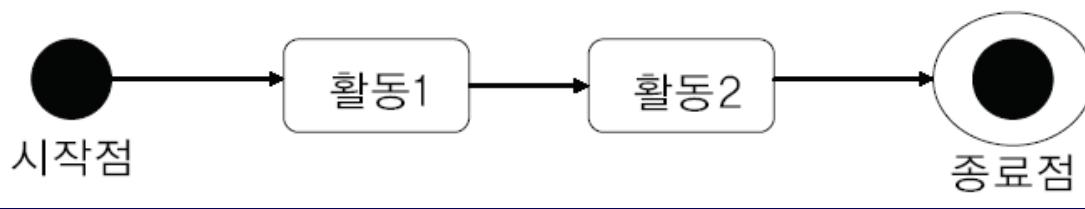
# Activity Diagram 개요

## ◆ 활동 다이어그램

- 오퍼레이션이나 처리 과정이 수행되는 동안 일어나는 일들을 단계적으로 표현
- 활동 상태 및 전이, 분기, 동기화 막대, 신호, 구획면 등으로 표현

## ◆ 활동 및 전이

- [그림 1]
  - 시작점, 활동 1, 활동2, 종료점 그리고 전이로 구성
  - 전이 : 화살표를 의미
- 시작점
  - 활동의 시작을 의미 ( 검은색 동그라미로 표현 )
- 활동 상태
  - 어떠한 일들의 처리와 실행을 의미 ( 모서리가 둥근 사각형으로 표현 )
- 종료점
  - 처리의 종료를 의미 ( 이중 동그라미로 표현 )
- 활동 다이어그램에서 하나의 활동이 처리되면 그 다음 활동으로 자동적으로 옮겨지며, 활동 상태의 시작과 종료는 항상 존재

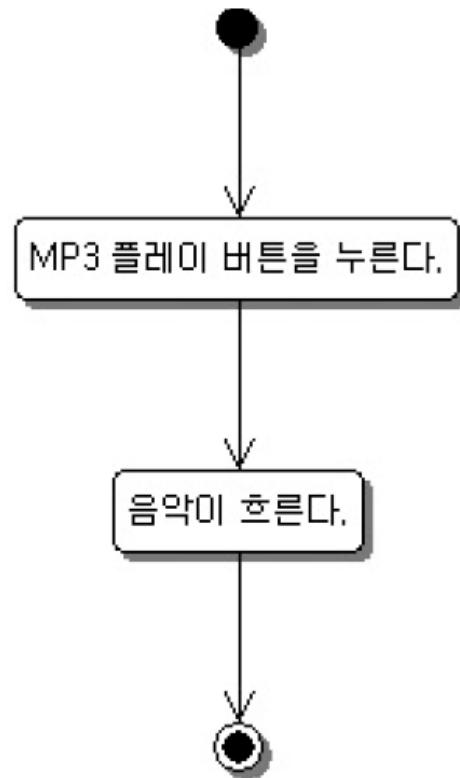


[그림 1] 활동 다이어그램 구조

# Activity Diagram 개요

## ◆ [그림 2]

- 시작점, 종료점, 2개의 활동 상태, 그리고 전이에 대한 예
- 시작 상태에서 조건 없이 “MP3 플레이 버튼을 누른다”는 활동으로 전이가 발생
- 이 활동의 행동이 완료되면 다음으로 “음악이 흐른다”의 활동으로 전이가 발생
- 그 후 활동 종료

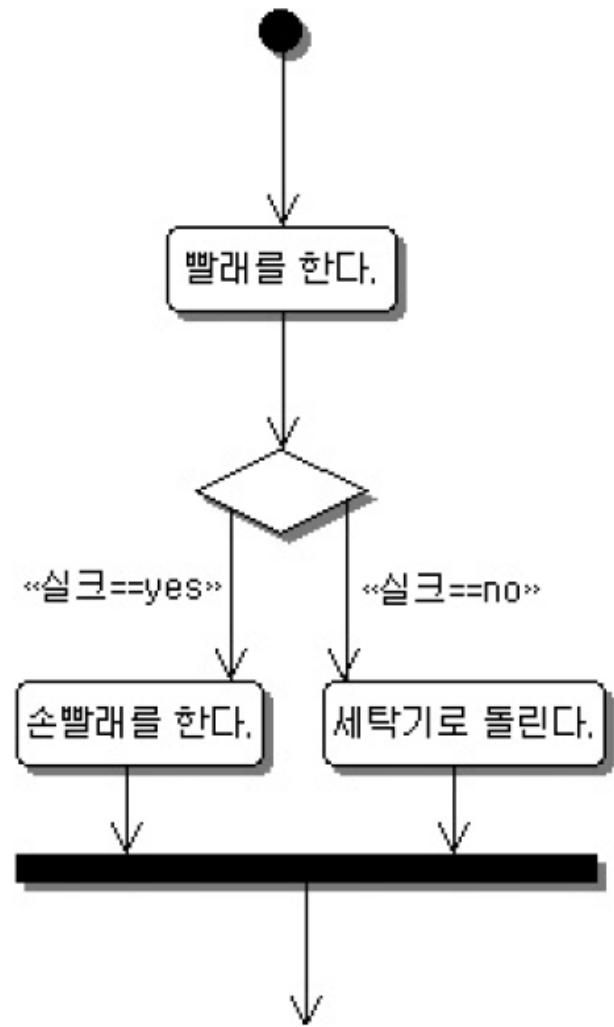


[그림 2] 시작점과 종료점을 가진 활동 상태와 전이

# Activity Diagram 설명

## ◆ 분기

- 활동 흐름이 2가지로 나뉘며,  
1개의 활동상태에서 전이할 때  
여러 가지의 활동 상태로 분기
- 어떤 조건이냐에 따라 처리 경  
로가 결정
- [그림 3]
  - 분기는 마름모(판단)를 사용  
하여 활동 전이를 나누는 방  
법을 보여주고 있다.
  - 마름모 표기시 해당 처리 경  
로 옆에는 대괄호를 이용한  
조건문을 써주면 된다.

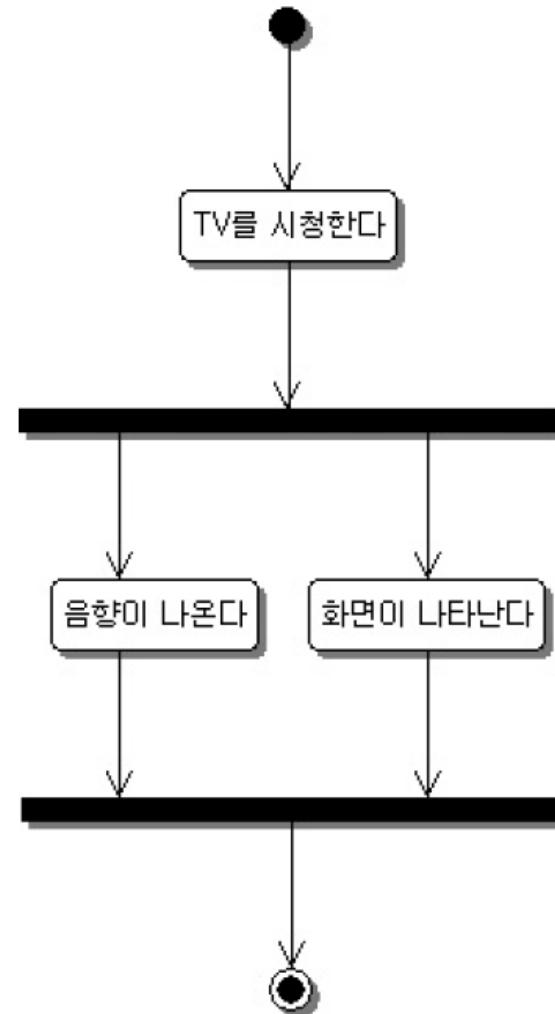


[그림 3] 분기 표현 방법

# Activity Diagram 설명

## ◆ 동기화 막대

- 활동 다이어그램에서는 한 가지 활동만 수행하지 않고 병행해서 수행하는 경우에 사용
- 동시 처리의 시작과 끝을 보여준다.
- [그림 4]
  - 가로 방향의 동기화 막대 사용 예
  - TV는 음향과 화면이 동시에 나온다
  - 2가지가 동시에 처리되기 때문에 동기화 막대를 사용

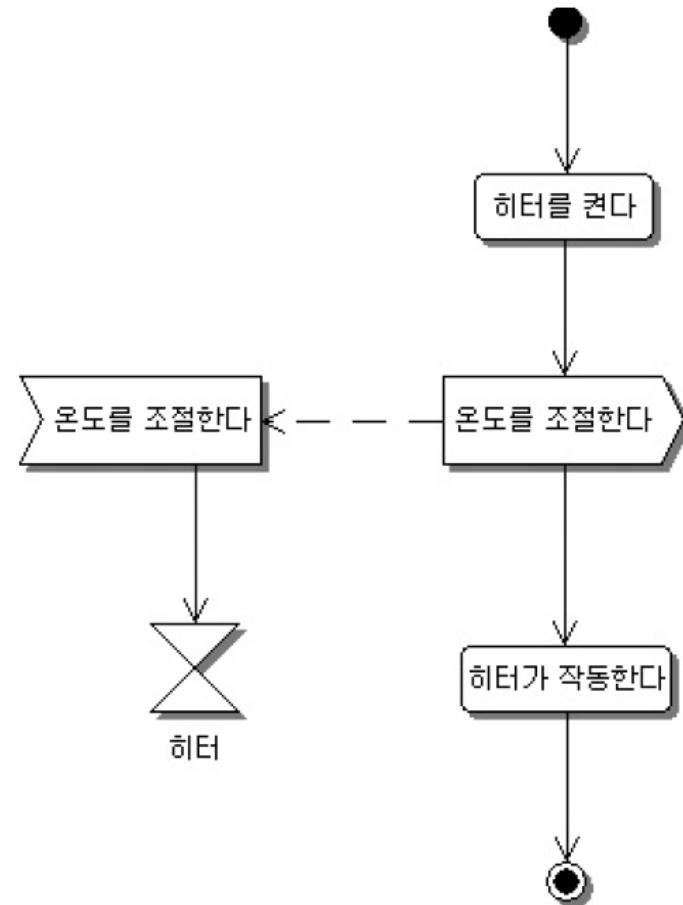


[그림 4] 동기화 막대 사용 예

# Activity Diagram 설명

## ◆ 신호

- 활동이 진행되는 도중에 보내는 방식으로, 활동과 객체 사이에 이루어지는 거래
- 뾰족한 오각형 모양의 송신 시그널과 쐐기 모양의 파인 다각의 수신 시그널로 표현

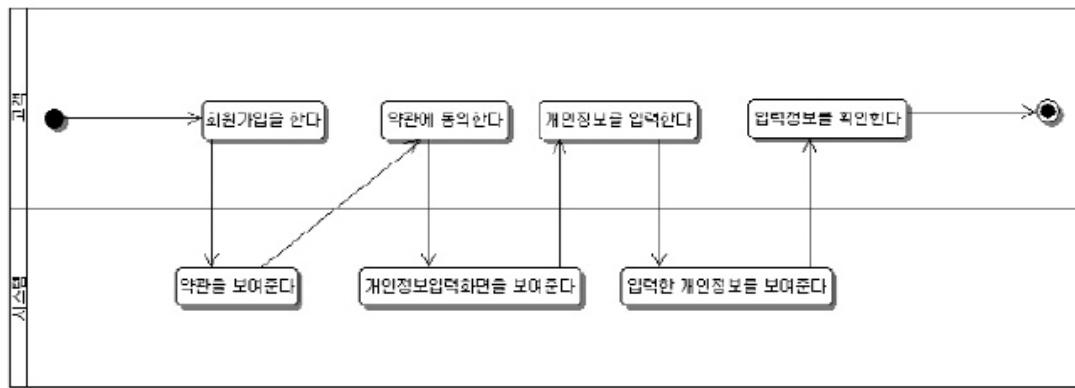


[그림 5] 신호의 전송과 수신 예

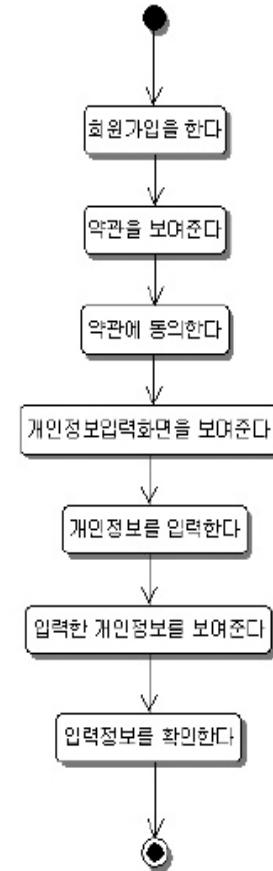
# Activity Diagram 설명

## ◆ 구획면

- 활동 다이어그램에서 그려지는 세로 방향의 영역
- 각 활동 상태를 담당하는 역할을 나타낸다.
- 2개 이상의 사각형으로 영역을 표시하고, 구획면의 이름이 기술
- [그림 6]
  - 회원가입에 대한 처리 과정
  - 총 7개의 활동이 시간적인 전이로 수행
  - (a) : 해석이 불명확하여 그 주체가 명확치 않다
  - (b) : 고객과 시스템의 활동영역을 구별하여 볼



(b) 구획면으로 표현한 회원가입 과정

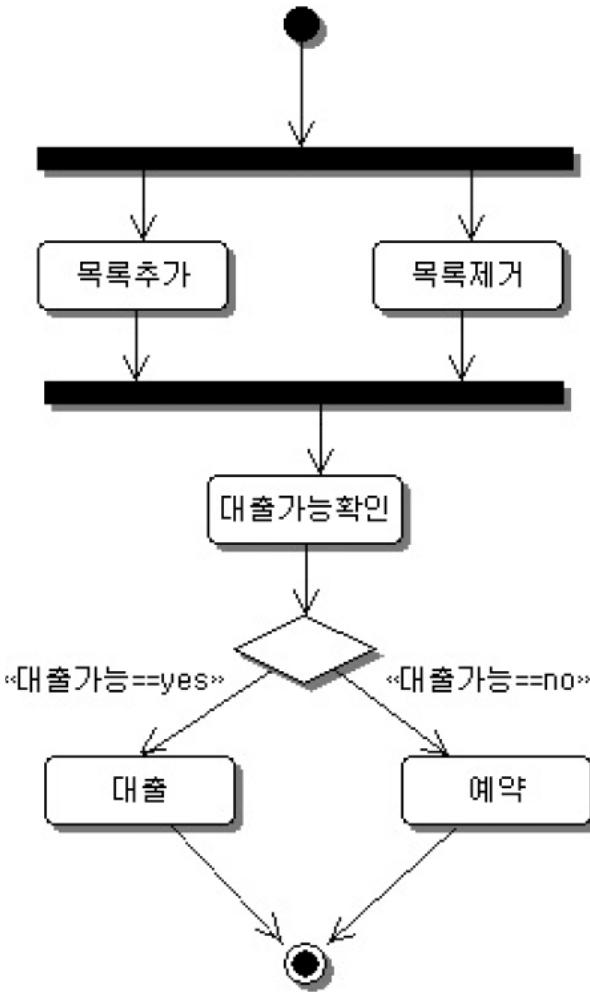


[그림 6] 회원가입에 대한 처리 과정

# Activity Diagram 예제

## ◆ 도서관리

- ▶ 도서관리에 대한 활동 디아그램의 진행 단계
  - 1단계 : 사서는 구입한 도서를 도서 목록에 추가
  - 2단계 : 사서는 제거한 도서를 도서 목록에서 삭제
  - 3단계 : 고객은 도서목록에서 대출이 가능한지 확인
  - 4단계 : 대출이 가능한 도서는 대출
  - 5단계 : 대출이 완료
  - 6단계 : 대출이 불가능한 도서는 예약
  - 7단계 : 예약이 완료



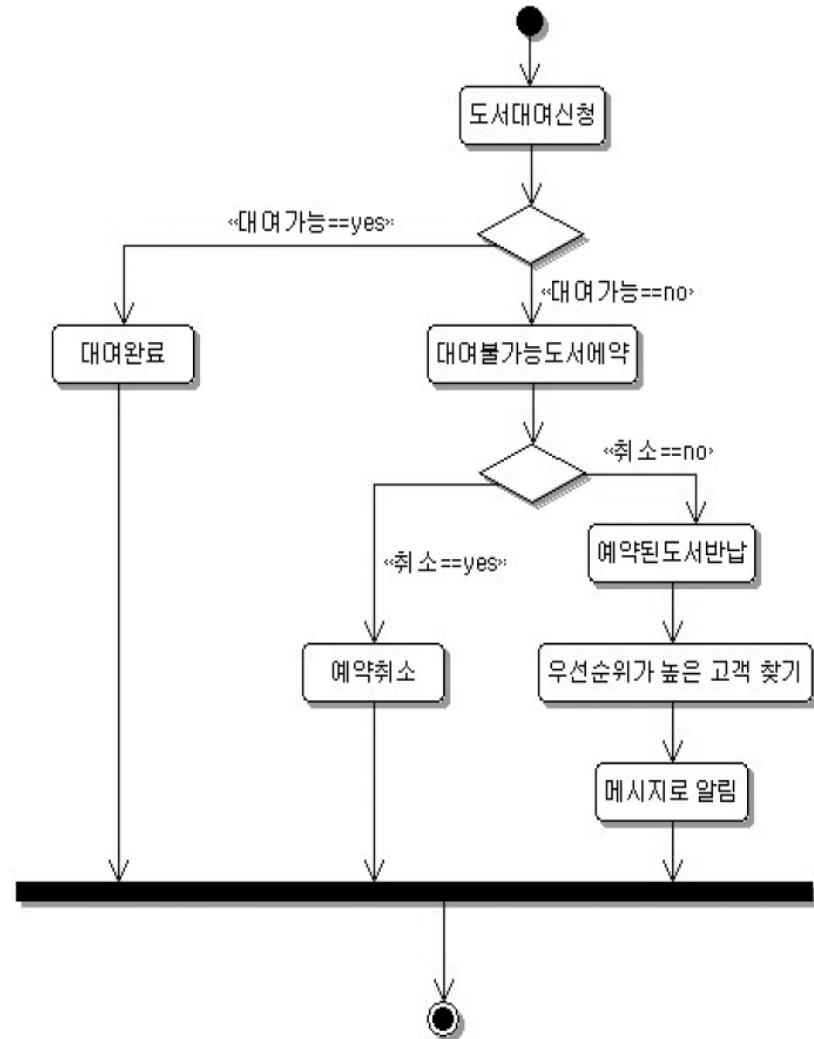
[그림 7] 도서관리

# Activity Diagram 예제

## ◆ 도서예약

### ▶ 도서예약 과정

- 1단계 : 고객이 도서 대출을 신청
- 2단계 : 사서는 대출이 가능한지 확인
- 3단계 : 대출이 불가능한 도서를 예약
- 4단계 : 고객이 예약 취소
- 5단계 : 취소가 완료
- 6단계 : 대출 중인 도서가 반납
- 7단계 : 예약 고객 중 우선순위 높은 고객을 찾는다.
- 8단계 : 대출 가능 메시지를 보낸다.
- 9단계 : 예약 시스템이 완료

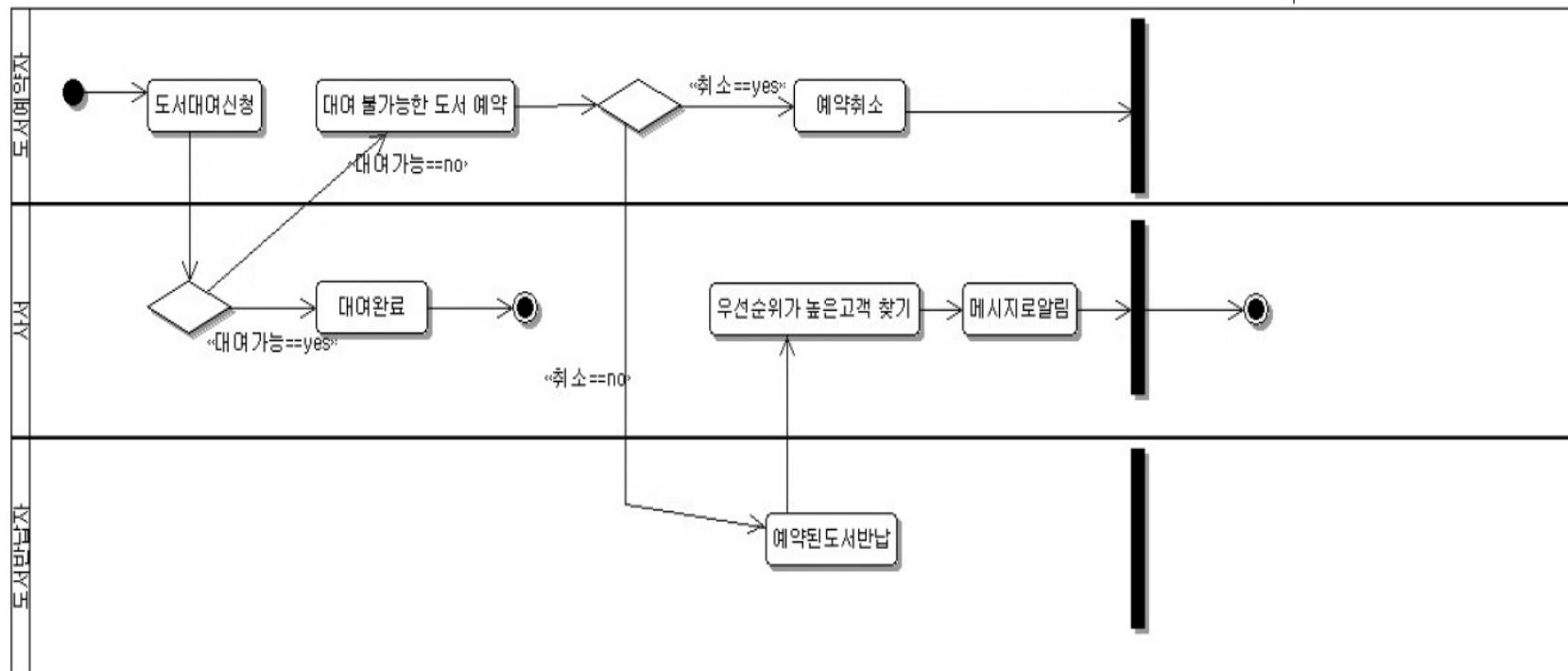


[그림 8] 도서예약

# Activity Diagram 예제

## ◆ [그림 9]

- 활동 다이어그램에 각 활동의 주체까지 표현하기 위해 구획면을 이용한 그림



[그림 9] 구획면을 이용한 도서예약

# State Diagram

# 상태

## 표기법

§ 반응적인 성격을 가지는 세탁기 객체의 행동을 상태 다이어그램으로 표현한 것

세탁중

건조중

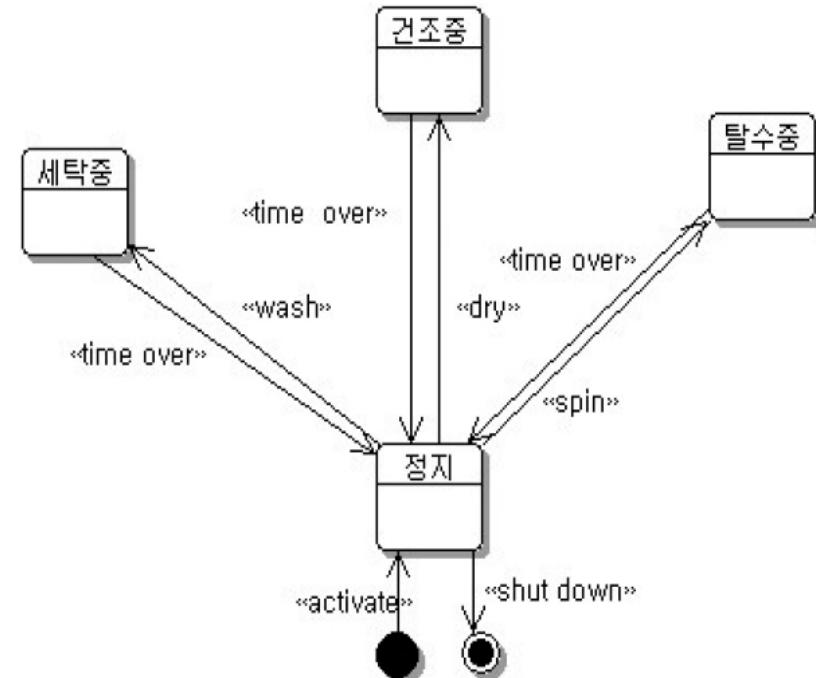
탈수중

정지

세탁기 객체가 가질 수 있는 상태들

### ▶ 상태

- § 객체가 존재할 수 있는 조건 중의 하나
- § 둥근 모서리를 가진 사각형으로 표시
- § 둥근 모서리를 가진 사각형의 안쪽 상단에 상태의 이름을 기술



세탁기 객체의 행동을 표현한 상태 다이어그램

# 이벤트

## ▶ 이벤트 전이

§ 객체에 외부로부터 자극이 전달된 경우

ü 이 자극에 의해서 객체가 다른 상태로 변경

§ 전이(Transition)

ü 객체의 상태가 다른 상태로 변경되는 것

§ 이벤트

ü 객체의 전이를 유발하는 자극

§ 상태 사이의 전이 : 실선으로 표시 (UML 상태 다이어그램)

§ 상태의 전이를 유발하는 이벤트는 전이 위에 이벤트 이름이 표시



time over 이벤트에 의해 '선택 중' 상태에서 '정지' 상태로의 전이

# 표기법

## 표기법

§ 상태와 이벤트를 상세하게 표현할 수 있음

§ 상태 아이콘은 클래스 아이콘처럼 두 영역으로 나누어 정보를 넣을 수 있다.

- ü 가장 위에는 상태 이름 (필수),
- ü 가장 아래 부분에는 활동 (선택적)

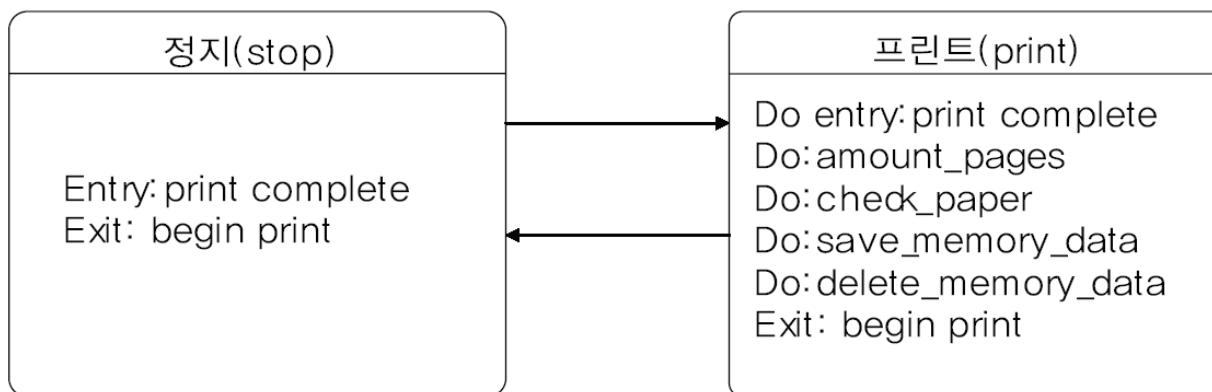


상태 아이콘의 확장된 표현

# 표기법

## § 예제 : 프린터

- ü 프린터는 컴퓨터로부터 프린트할 데이터를 메모리에 저장
- ü 프린트가 끝나면 메모리에 저장된 데이터를 삭제
- ü 상태 변수
  - » 메모리에 저장·삭제하는 save\_memory\_data, delete\_memory\_data
  - » 프린트할 종이가 있는지를 체크하는 check\_paper
  - » 프린트 할 페이지를 알아내는 amount\_pages



상태변수와 활동으로 표현된 프린터의 상태 다이어그램

# 예제

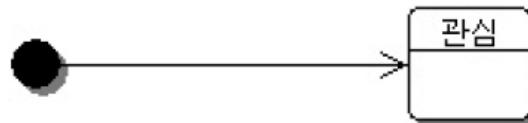
## 온라인 진료예약 시스템 상태를 기술하기 위한 상태 다이어그램

온라인 진료예약 시스템의 상태를 나타내기 위한 문제 기술서

A 병원은 환자들의 향상된 진료 서비스와 효율적인 환자관리를 위해 진료예약 시스템을 개발한다. 이 시스템은 현재 환자의 상태를 파악하고 어떠한 상태에 있느냐에 따라 효율적으로 환자를 관리하는 시스템이다. 우선 병원을 방문하여 한 번이라도 진료를 받은 환자이면 ‘관심’ 상태가 된다. 지속적인 통근 진료를 요하는 환자의 경우 ‘진료중’ 상태로 한다. ‘관심’이나 ‘진료중’ 상태에서 의사의 결정에 따라 진료가 종료되면 ‘진료종료’로 한다. ‘진료중’에서 의사의 결정에 따라 입원이 결정되면 ‘입원상태’가 된다. 입원상태에서 퇴원이 결정되면 ‘진료중’이나 ‘진료종료’ 상태가 된다.

### § 첫번째 단계 : 환자의 초기 상태 확인

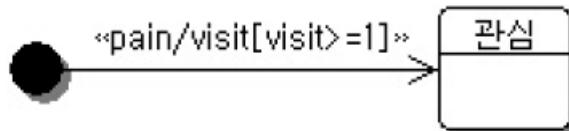
- ü 기술에서 “우선 병원을 방문하여 한 번이라도 진료를 받은 환자”이면 ‘관심’ 상태가 된다.
- ü 초기 상태 표현 : 시작 상태, 전이 화살표, 첫 번째 상태 요소 필요



환자의 초기 상태: 관심 상태

### § 초기 상태에서 사건

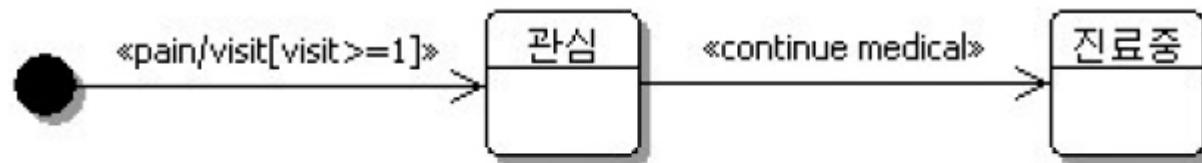
- ü 환자가 한 번이라도 방문해야 한다.



사건과 조건의 표현

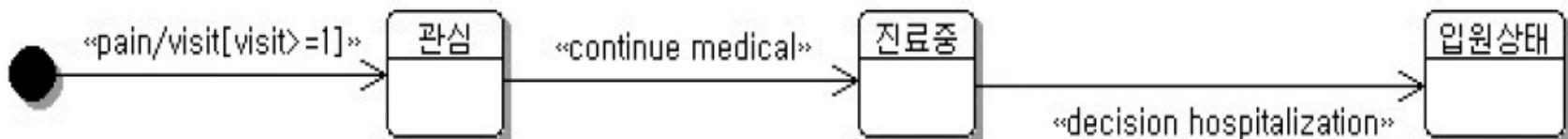
# 예제

- § ‘관심’ 상태를 다른 상태로 바꾸는 이벤트 확인
- ü 기술서 : “지속적인 통근 진료를 요하는 환자의 경우 ‘진료중’상태로 한다”
  - ü 상태 변화 모델링 : 전이 화살표, 이벤트, 객체가 전이할 새로운 상태 필요



‘관심’ 상태에서 ‘진료중’ 상태로의 전이

- § ‘진료중’ 상태를 다른 상태로 바꾸는 이벤트 확인
- ü 기술서 : “‘진료중’에서 의사의 결정에 따라 입원이 결정되면 ‘입원상태’가 된다”
  - ü ‘진료중’에서 ‘입원상태’로 전이
  - ü 이벤트 : “입원결정”

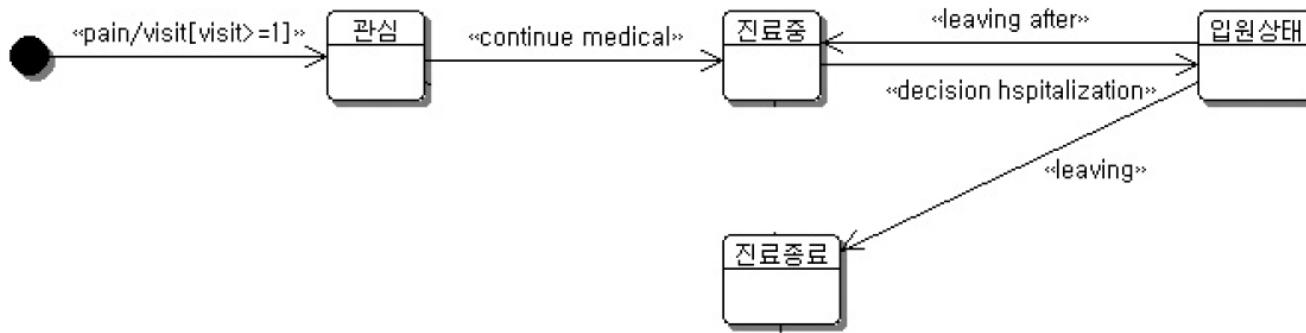


‘진료중’에서 ‘입원상태’로의 전이

# 예제

## § 입원상태를 다른 상태로 바꾸는 이벤트 확인

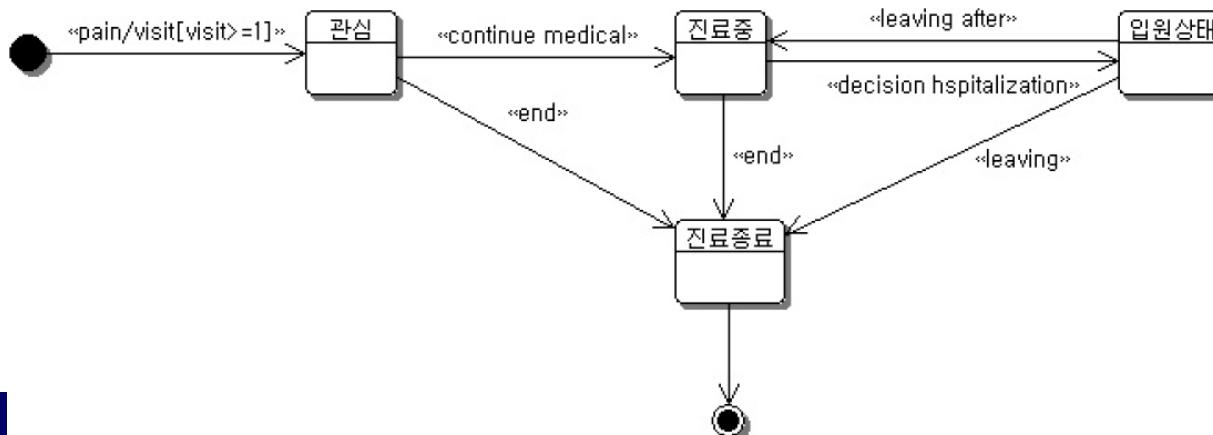
- ü 기술서 : “입원상태에서 퇴원이 결정되면 ‘진료중’이나 ‘진료종료’상태 가 된다”
- ü ‘입원상태’에서 ‘진료중’상태나 ‘진료종료’로 전이



‘입원상태’에서  
‘진료종료’로의 전이

## § 관심 상태에서나 진료중 상태에서 진료종료 상태로 전이

- ü 기술서 : “‘관심’이나 ‘진료중’상태에서 의사의 결정에 따라 진료가 종료되면 ‘진료종료’로 한다”



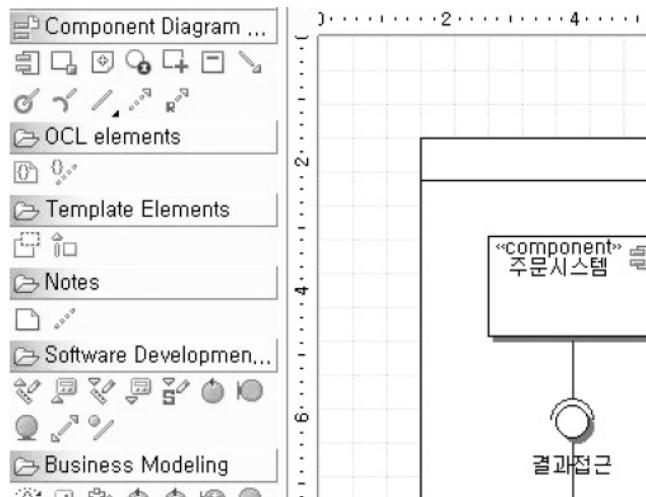
‘관심’과 ‘진료중’상태에서  
‘진료종료’상태로 전이

# Component Diagram

# 컴포넌트 정의

## ◆ 컴포넌트

- ❖ 시스템을 구성하는 임의의 물리적인 요소를 의미
- ❖ 물리적인 요소란 가상의 모델을 실제로 구현하여 나타내는 것을 의미
- ❖ 객체지향의 원리에 따라 업무 기능과 관련 데이터를 하나의 단위로 처리
- ❖ 인터페이스에 의해서 기능이 정의된, 독립적으로 개발·배포·조립이 가능한 시스템의 구성 단위로 정의



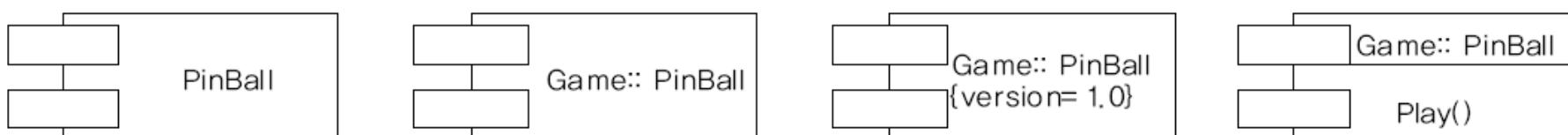
[표 1] 컴포넌트 다이어그램의 구성 요소

컴포넌트	논리적 요소들이 물리적으로 패키지화
인터페이스	컴포넌트가 실현하고자 하는 여러 오퍼레이션의 모임
의존관계	컴포넌트와 컴포넌트 간의 관계
지원관계	컴포넌트와 인터페이스 간의 관계

[그림 1] 컴포넌트 다이어그램에 사용하는 심벌

# 컴포넌트 다이어그램

- ◆ 컴포넌트
  - ❖ 시스템을 구성하는 물리적인 컴포넌트와 그들 사이의 의존관계를 나타내는 다이어그램
  - ❖ 컴포넌트, 인터페이스, 의존관계로 표현
- ◆ 컴포넌트
  - ❖ 컴포넌트는 탭이 달린 직사각형으로 표현
  - ❖ 모든 컴포넌트는 반드시 이름을 가지고 있어야 한다.
  - ❖ 컴포넌트가 패키지에 포함되어 있다면 컴포넌트의 이름 앞에 패키지 이름을 붙일 수 있으며,
    - 클래스처럼 컴포넌트에 꼬리표 값을 달아주거나 컴포넌트 내부의 오퍼레이션

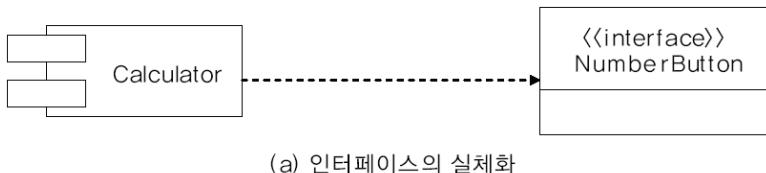


[그림 3] 컴포넌트의 표현 방법

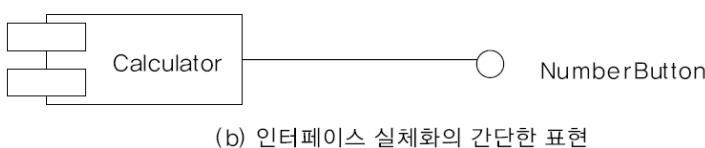
# 컴포넌트 다이어그램

## ◆ 인터페이스

- ❖ 컴포넌트 인터페이스는 2가지 방식으로 표현
  - 컴포넌트와 인터페이스, 그리고 이를 연결하는 화살표 모양의 점선(의존관계)으로 나타낼 수 있다.
- ❖ 인터페이스를 실체화한다는 의미
  - 실제로 동작하는 컴포넌트에 인터페이스를 적용한다는 것
- ❖ 컴포넌트 다이어그램은 실체화 관계뿐만 아니라 의존관계도 표현
  - 의존관계는 컴포넌트와 필수의 인터페이스 사이에 설정

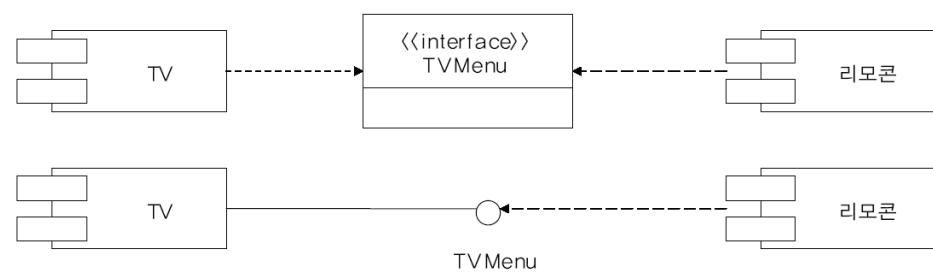


(a) 인터페이스의 실체화



(b) 인터페이스 실체화의 간단한 표현

[그림 4] 인터페이스의 표현 방법



[그림 5] 컴포넌트와 인터페이스의 실체화와  
의존관계 표현 방법

# 컴포넌트 다이어그램

## ◆ 호출관계

- 컴포넌트 사이의 호출관계는
  - ▶ 한 컴포넌트에 다른 컴포넌트를 호출하여 원하는 값을 요청하거나 데이터를 주는 경우에 사용한다.

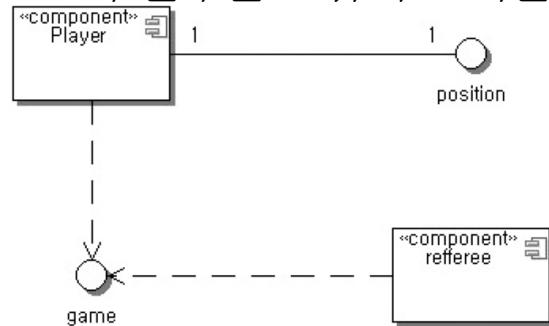


[그림 6] 컴포넌트간의 호출관계

# 컴포넌트 다이어그램 예제

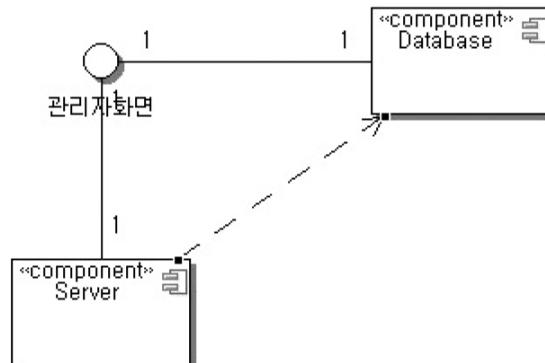
## ◆ 축구 경기

- 축구 경기는 선수(*Player*)와 심판(*Referee*)으로 구성
- 각 선수는 고유의 포지션(*position*)을 가져 경기에 임한다.



## ◆ 서버와 데이터베이스

- 서버는 데이터베이스를 바탕으로 되어 있으며,
- 관리자를 통해 서버와 데이터베이스를 관리

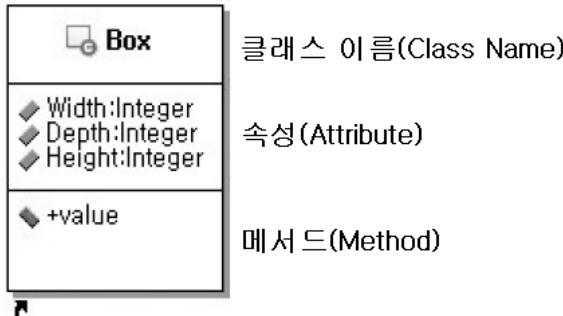


# Class Diagram

# 클래스 구성 요소

## ◆ 구성 요소

### ■ 클래스 이름, 속성, 메서드



[표 1] 메서드의 종류와 기호

메서드의 종류	부호	내용
public	+	자신의 속성이나 동작을 외부에 공개하는 접근 제어
private	-	상속된 파생 클래스만이 액세스할 수 있는 접근 제어
protected	#	구조체의 멤버 함수만 접근할 수 있으며 외부에서 액세스할 수 없는 접근 제어

[그림 1] 클래스 구성 요소

[코드 4-1] [그림 4-1]에 대한 자바 코드

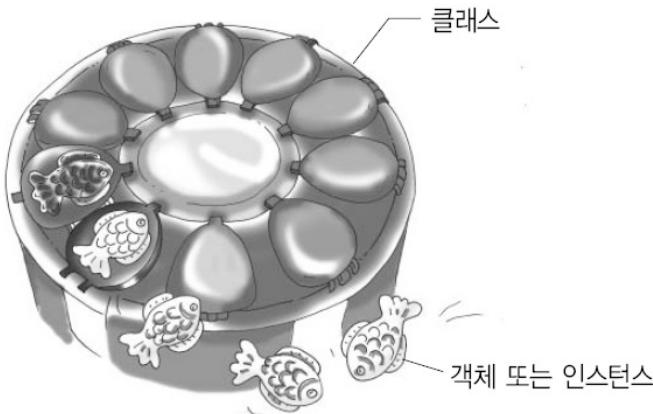
```
01 // class name
02 Box{
03     // attribute
04     private int width;
05     private int depth;
06     private int hight;
07     // method
08     void value(){
09     }
10 }
```

# 클래스 구성 요소

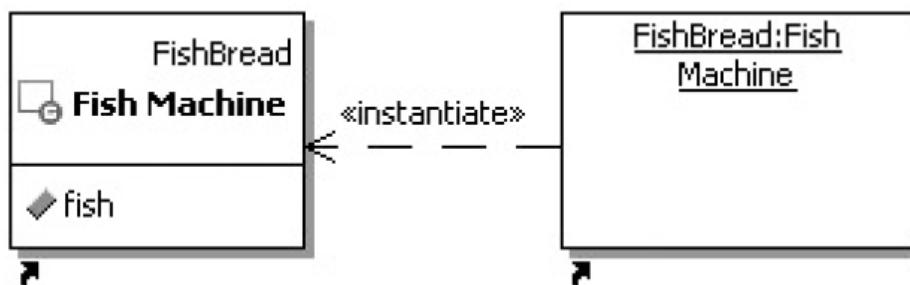
- ❖ 클래스(class)
  - 공통의 속성, 메서드(오퍼레이션), 관계, 의미를 공유하는 객체들의 집합
- ❖ 속성(attribute)
  - 클래스의 구조적 특성에 이름을 붙인 것으로 특성에 해당하는 인스턴스가 보유할 수 있는 값의 범위를 기술
- ❖ 메서드(method)
  - 오퍼레이션이라고도 하며,
  - 이름, 타입, 매개변수들과 연관된 행위를 호출하는데 요구되는 제약사항들을 명세하는 클래스의 행위적 특징

# 객체와 클래스

- ◆ 객체 : 실제 현실에서 존재하는 사물을 의미
- ◆ 클래스 : 객체들을 추상화한 개념



[그림 2] 봉어빵 기계와 봉어빵(클래스와 객체의 관계)



[그림 3] 클래스와 객체와의 관계

# 객체와 클래스

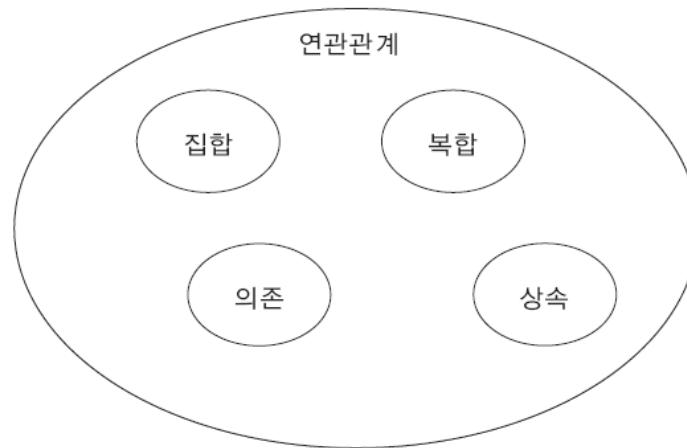
- ◆ 예제 : 다음의 다이어그램은 하나의 계좌에 입금되는 클래스(**Account**)와 객체를 생성하여 실행하는 메인 메서드를 포함하는 클래스(**Application**)로 구성

[코드 4-2] 객체 생성의 자바 코드

```
01 class Account
02 {
03     int balance;
04     int deposit(int amount){
05         balance = balance + amount;
06         return (balance);
07     }
08 }
09
10 class Application{
11
12     public static void main(String args[])
13     {
14         Account account1;
15         // 객체 생성
16         account1 = new Account();
17         account1.balance = 5000;
18         account1.deposit(5000);
19         System.out.println("Balance = " +account1.balance);
20     }
21 }
```

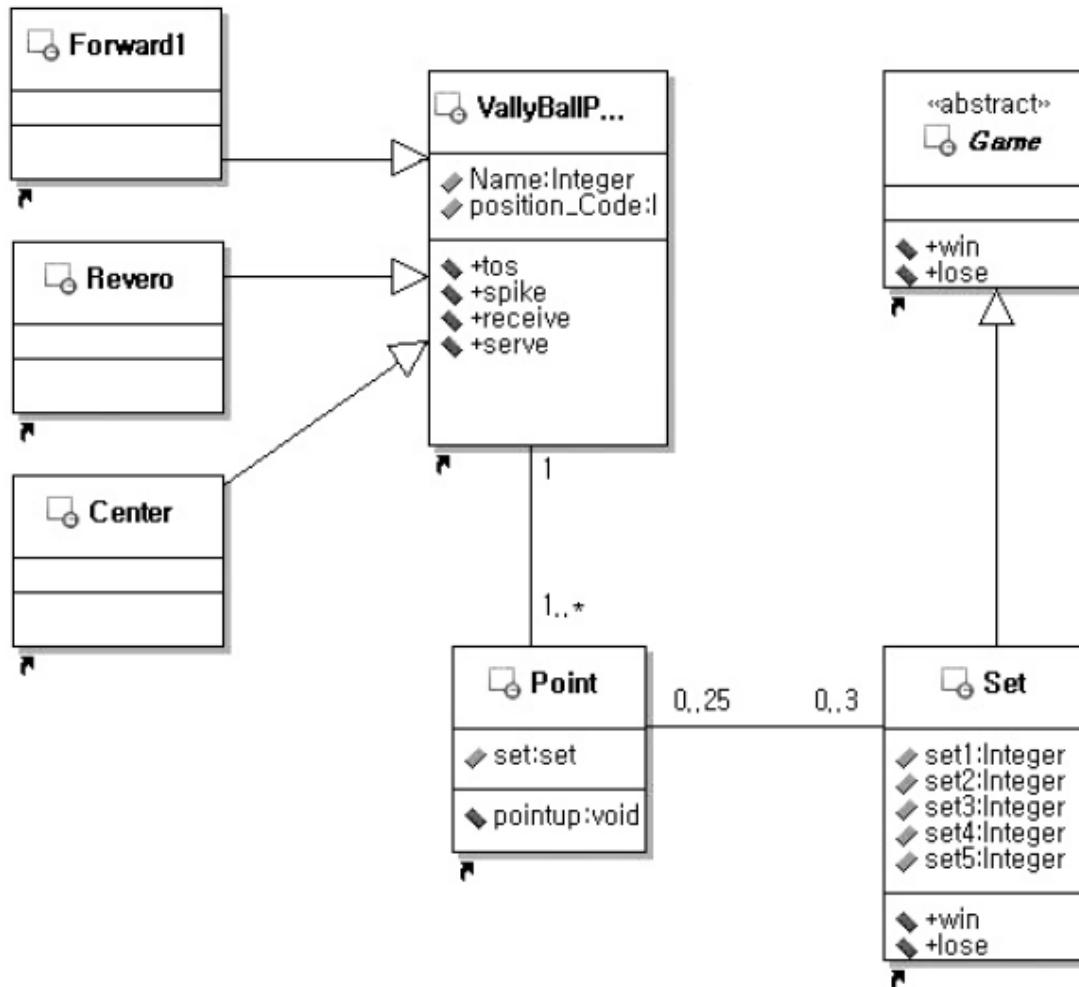


[그림 4] 연관관계의 종류



[그림 5] 연관관계의 종류

# 클래스의 관계 예시



[그림 6] 배구에 관한 클래스 다이어그램

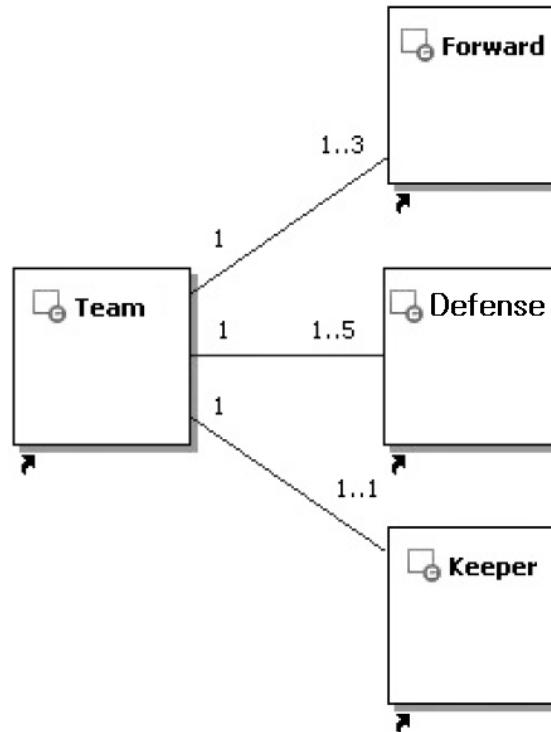
# 클래스의 관계 - association

## ◆ 연관관계(Association)

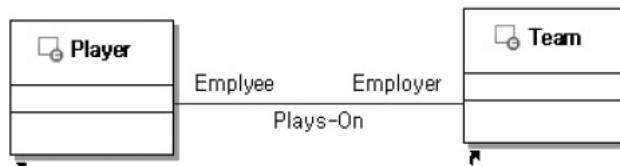
- 연관관계는 클래스가 서로 개념적으로 연결된 선을 의미
- 클래스 다이어그램에서 연관관계는 서로 개념적으로 연관없는 클래스는 없으므로 의존, 상속, 집합, 복합 등의 관계와 같이 표시하는 건 별로 중요하지 않다.
- 예제 : 축구팀과 선수와의 관계



[그림 7] 팀과 선수의 연관관계

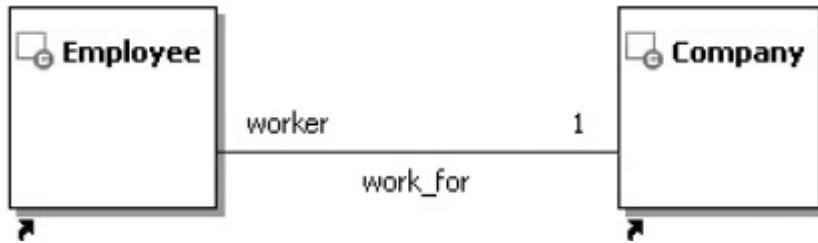


[그림 9] 하나의 클래스와 여러 클래스와의 연관관계

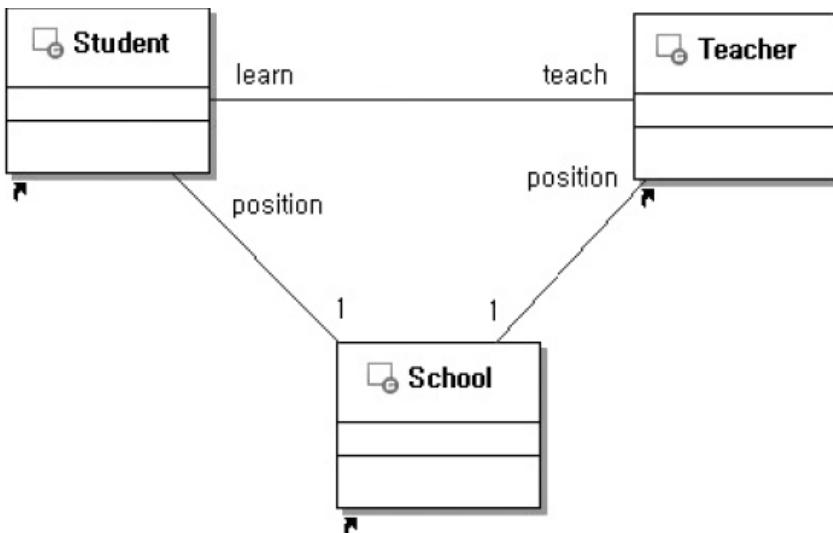


[그림 8] 선수와 구단의 연관관계

# 클래스의 관계 - association



[그림 10] 사원과 회사 사이의 연관관계

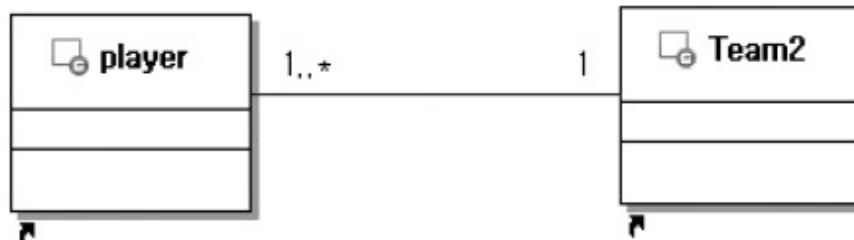


[그림 11] 학생, 교사, 학교 사이의 연간관계

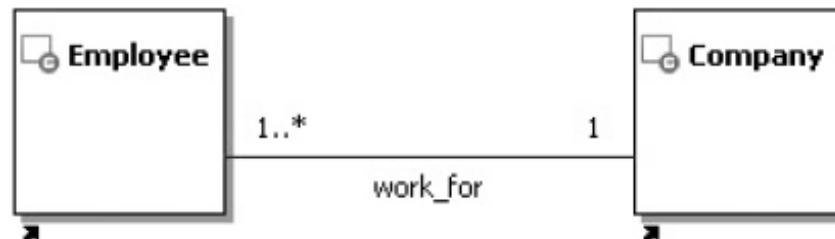
# 클래스의 관계 - association

## ◆ 연관관계의 다중성

- **다중성** : 두 클래스의 연관관계에서 실제로 연관을 가지는 객체의 수를 나타낸다.



[그림 12] 선수와 팀의 다중성



[그림 13] 회사와 사원의 다중성 관계

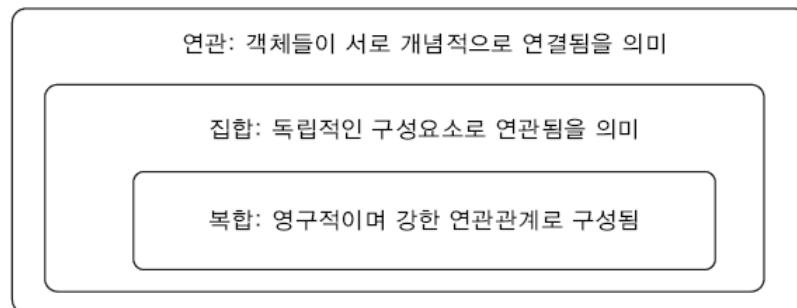
다중성 표현	의미
1	한 객체와 연관된다. 표시하지 않아도 되는 기본값이다.
0..1	0개 또는 1개의 객체와 연관된다.
0..*	0개 또는 많은 수의 객체가 연관됨을 나타낸다.
*	0..*와 동일하다
1..*	1개 이상의 객체와 연관된다.
1..12	1개에서 12개까지의 객체가 연관됨을 나타낸다.
1..2, 4, 11	1개에서 2개까지 또는 4개 또는 11개의 객체가 연관됨을 나타낸다.

[표 2] 다중성의 표현

# 클래스의 관계 - aggregation

## ◆ 집합과 복합관계

- 집합(**Aggregation**) 관계와 복합(**Composition**) 관계 모두 연관관계에 포함되는 개념



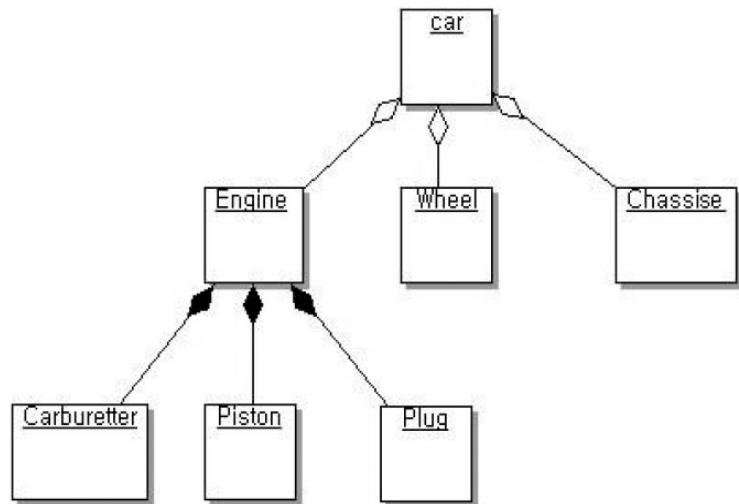
[그림 14] 연관관계와 그 중 집합과 복합의 개념

### ■ 집합관계

- 하나의 객체에 여러 개의 독립적인

### ■ 복합관계

- 더 강한 관계로 구성
- 엔진은 카뷰레터, 피스톤, 플러그로
- 엔진의 구성 요소는 더 강한 관계

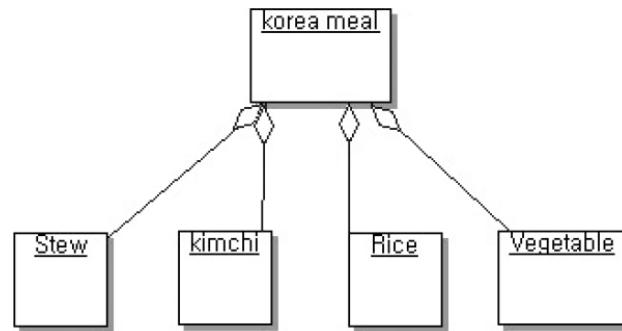


[그림 15] 차와 엔진, 바퀴, 차체(집합관계)  
엔진과 카뷰레터, 피스톤, 플러그(복합관계)

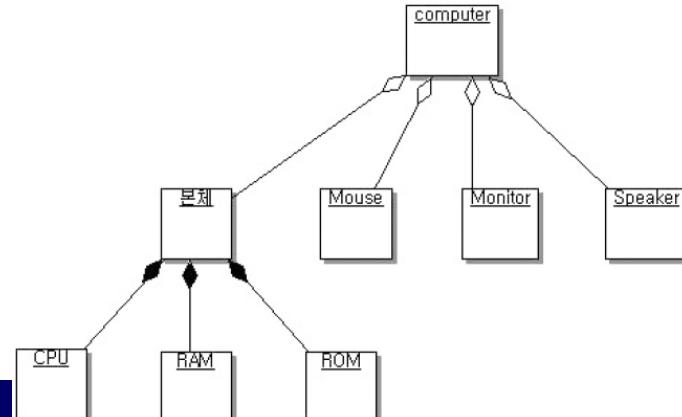
# 클래스의 관계 - aggregation

## ◆ 집합과 복합 예제

- ❖ 식사가 밥,찌개,김치,나물 등으로 구성되어 있을 경우, 이들은 식사에 대한 구성 요소이므로 집합관계
- ❖ 컴퓨터도 마찬가지로 그 구성 요소로 이루어지면 집합관계
- ❖ 컴퓨터 본체는 여러 가지의 구성 요소가 존재하는데 이는 영구적인 요소



[그림 16] 식사 : 밥,찌개,김치,나물

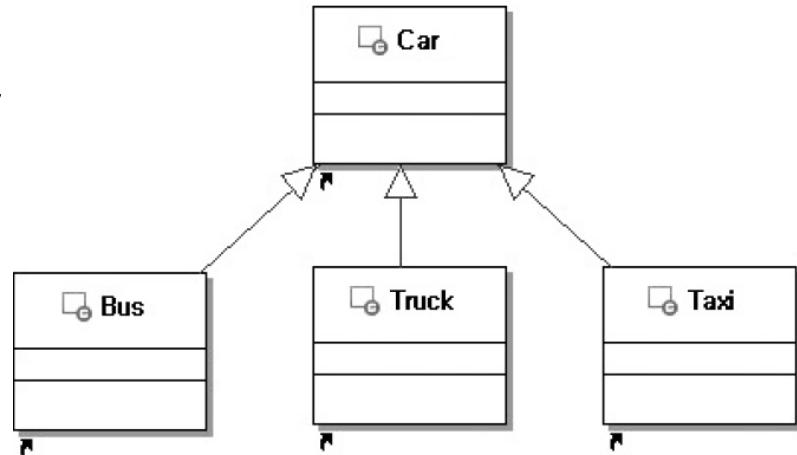


[그림 17] 컴퓨터와 모니터, 마우스, 키보드, 스피커(집합관계).  
본체와 CPU, ROM, RAM(복합관계)

# 클래스의 관계 - generalization

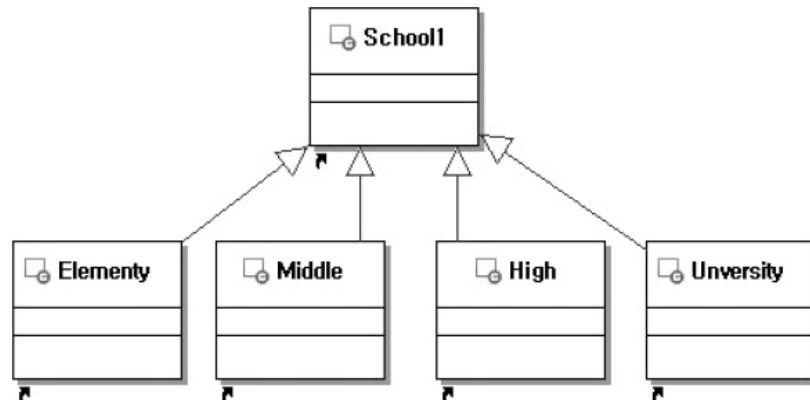
## ◆ 일반화관계

- 일반화관계는 하나의 종류를 의미
- **a kind of**의 관계

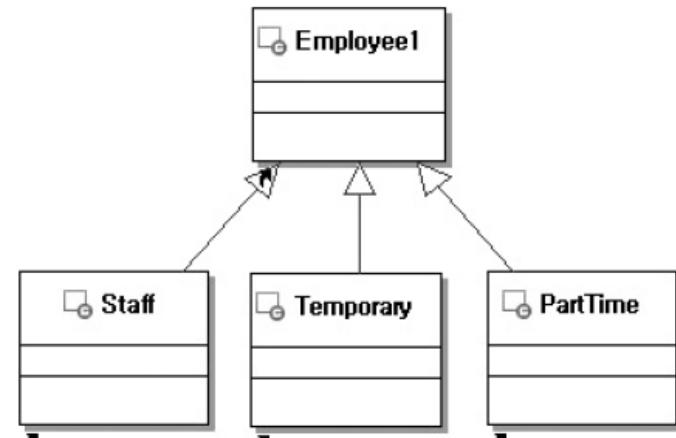


[그림 18] 차와 버스, 트럭, 택시(일반화관계)

- 일반화관계는 다른 의미로 상속관계라고도 한다.

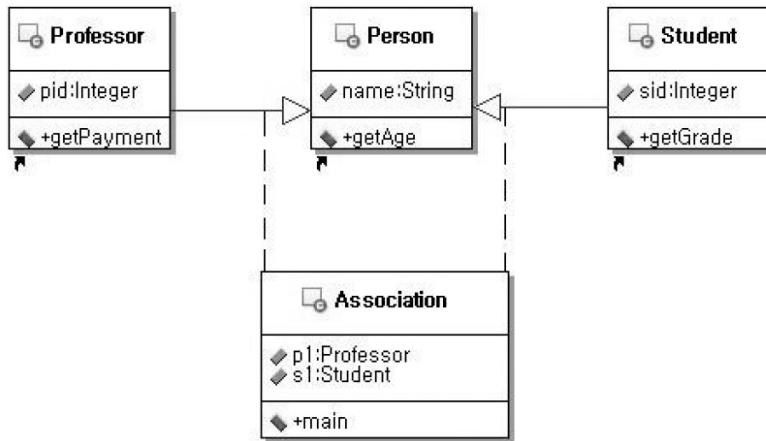


[그림 19] 학생과 초등학생, 중학생, 고등학생, 대학생



[그림 20] 사원과 정사원, 계약사원, 아르바이트생

# 클래스의 관계 - generalization



[그림 21] 사람과 교수, 학생 간의 일반화관계

[코드 4-3] 일반화관계 자바 코드

```
01 class Person
02 {
03     String name;
04     int getAge()
05     {
06         return 49;
07     }
08 }
09
10 class Student extends Person
11 {
12     int sid;
13     int getGrade(){
14         return sid-200400;
15     }

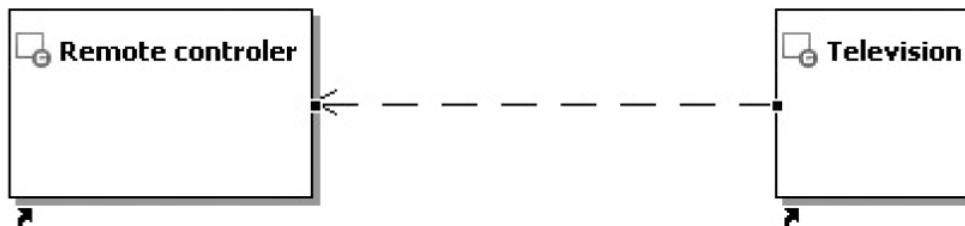
```

```
16 }
17
18 // Person으로 상속받기 때문에 name, getAge(), pid, getPayment()로 구성
19 class Professor extends Person
20 {
21     int pid;
22     int getPayment()
23     {
24         return pid+10000;
25     }
26 }
27
28 class Inheritance
29 {
30     public static void main(String[] args) {
31         Student s1 = new Student();
32         Professor p1 = new Professor();
33         s1.name = "홍길동";
34         s1.sid = 200401;
35         System.out.println("Student name : "+s1.name+ "Student
ID : "+s1.sid);
36         System.out.println("Student Age : "+s1.getAge()+
"Student Grade : "+s1.getGrade());
37         p1.name = "홍교수";
38         p1.pid = 1016;
39         System.out.println("Professor name : "+p1.name+
"Professor ID : "+p1.pid);
40         System.out.println("Professor Age : "+p1.getAge()+
"Professor Payment : "+p1.getPayment());
41     }
42 }
```

# 클래스의 관계 - dependency

## ◆ 의존관계

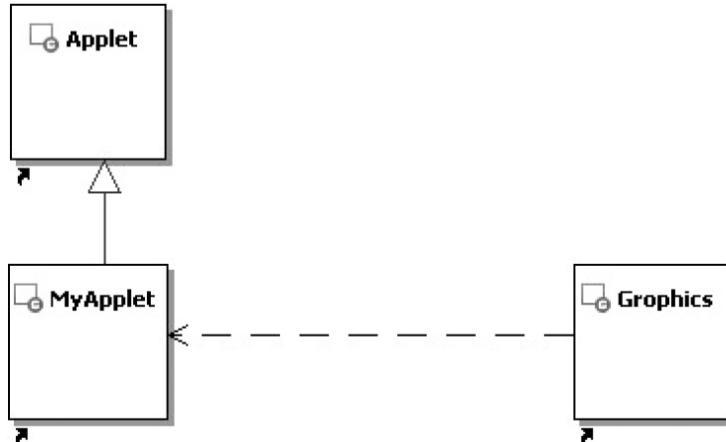
- 의존관계 : 하나의 클래스가 또 다른 클래스를 사용하는 관계
- 다른 클래스의 멤버 함수를 사용하는 경우
- 하나의 클래스에 있는 멤버 함수의 인자가 변함에 따라 다른 클래스에 영향을 미칠 때의 관계를 의미



[그림 22] TV와 리모컨의 의존관계

# 클래스의 관계 - dependency

- ◆ 의존관계에 있어서 클래스 A가 클래스 B의 객체를 생성하는 경우
- ◆ 예 : 애플릿 프로그램인 MyApplet의 point( ) 메서드에서 text를 출력하기 위한 g.drawString( ) 메서드 호출할 경우



[그림 23] 메소드 호출의 경우(의존관계)

[코드 4-5] 애플릿 프로그램

```
01 public class MyApplet extends Applet
02 {
03     init();
04     public void start();
05     public void paint(Graphics G)
06     {
07         g.drawString("Hello Applet World",10,20);
08     }
09 }
```

# 클래스의 관계 - realization

## ◆ 추상클래스와 인터페이스

### ■ 추상클래스

- 이탤릭체로 클래스명을 표시하며 스테레오타입을 이용하여 <<abstract>>로 표기

### ■ 인터페이스

- 스테레오타입을 이용하여 <<interface>>로 표기하고, 이탤릭체로 인터페이스 명을 표시



[그림 24] 추상클래스와 인터페이스

## ◆ 실체화 관계

- 추상클래스나 인터페이스를 상속받아  
자식클래스가 추상메서드를 구현할 때 사용

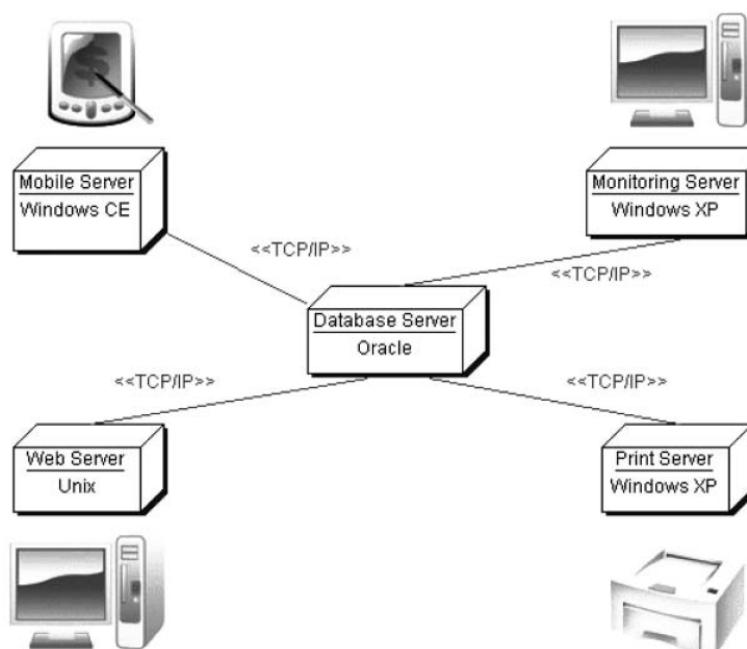
[그림 25] 실체화 관계

# Deployment Diagram

# 개요

## ◆ 배치 다이어그램

- ❖ 네트워크, H/W 또는 S/W 들을 실행파일 수준 컴포넌트들과 함께 표현
- ❖ 컴포넌트 다이어그램과 함께 시스템의 물리적인 요소를 모델링
- ❖ 시스템을 구성하는 처리장치와 그들 사이의 통신 경로를 기술할 때 사용
- ❖ 2가지 요소 : 노드, 커넥션

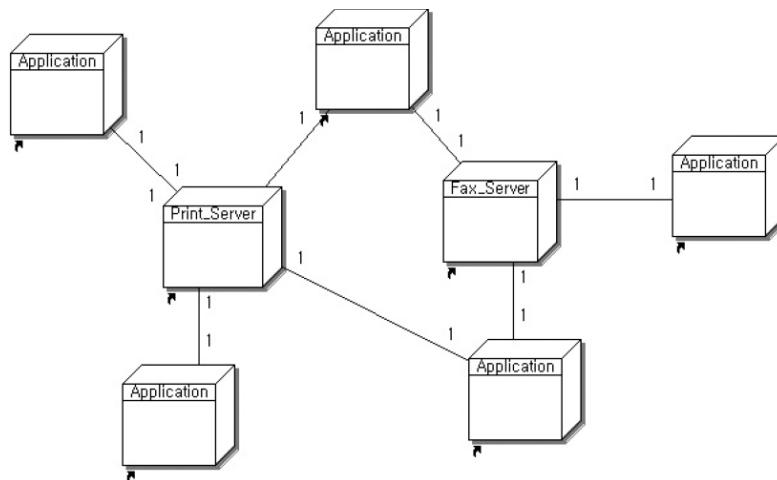


[그림 1] 배치 다이어그램

# 노드

## ◆ 노드

- 노드는 처리 능력을 가진 장치를 의미
- 배치 다이어그램에서 직육면체로 표시
- [그림 2]
  - 여러 대의 컴퓨터가 프린터나 팩스를 공유하는 시스템을 표현

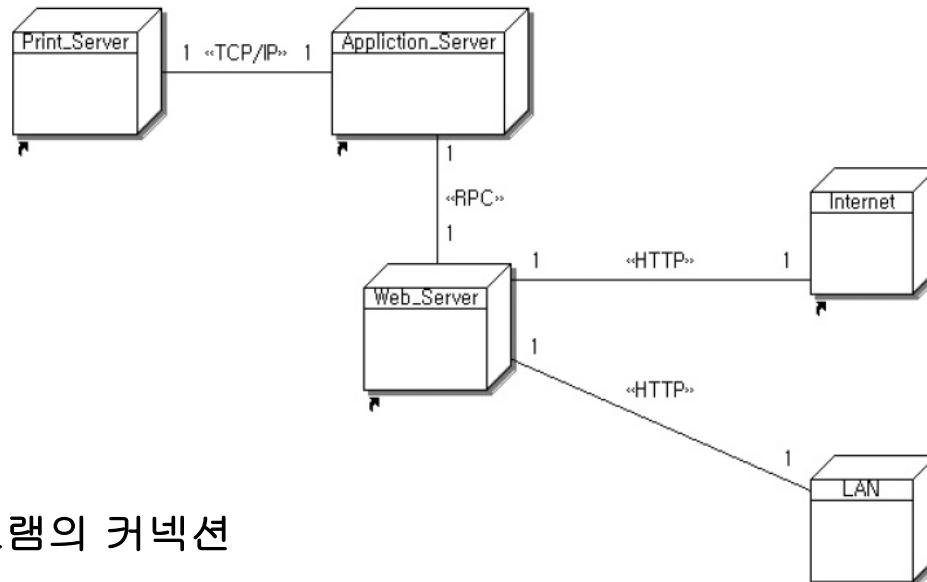


[그림 2] 배치 다이어그램의 노드

# 커넥션

## ◆ 커넥션

- 배치 다이어그램에서 노드들 사이의 연결을 의미
- 해당 노드들의 통신 방식을 표현
- [그림 3]
  - 프린트 서버와 애플리케이션 서버는 TCP/IP 방식으로 통신
  - 애플리케이션 서버와 웹 서버는 RPC 방식을 이용
  - 웹 서버와 인터넷, LAN은 HTTP 방식을 이용



[그림 3] 배치 다이어그램의 커넥션

# 예제

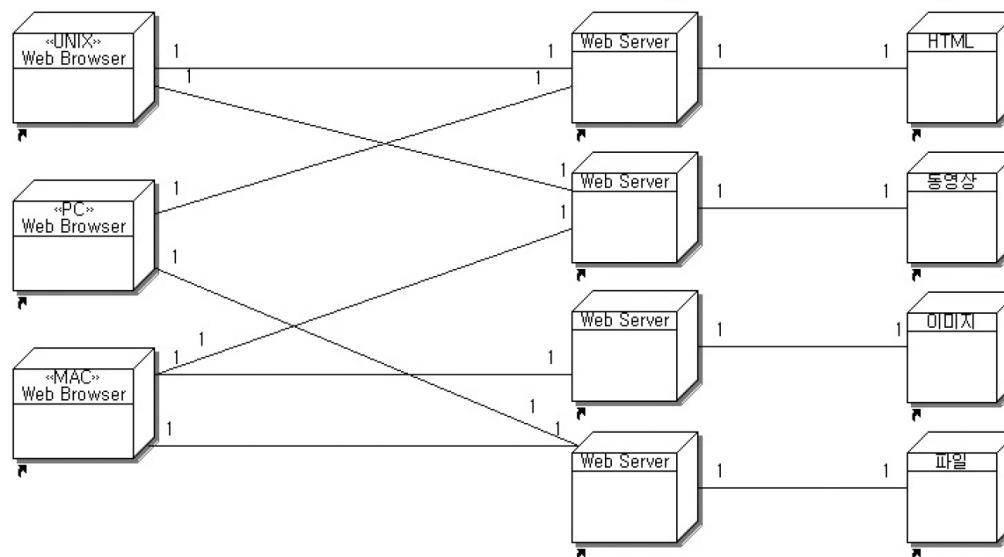
## ◆ 웹(www) 배치 다이어그램

### ■ WWW 서비스

- 전자우편, 네트워크, 뉴스, FTP 등 인터넷에서 제공하는 모든 기본 서비스
- 그림, 동영상, 음성, 문자 등의 멀티미디어 정보 및 하이퍼텍스트 기능 제공

### ■ WWW 구성

- 서버와 클라이언트로 구성
- 서버로 접속하기 위한 클라이언트는 플랫폼을 가리지 않는다.

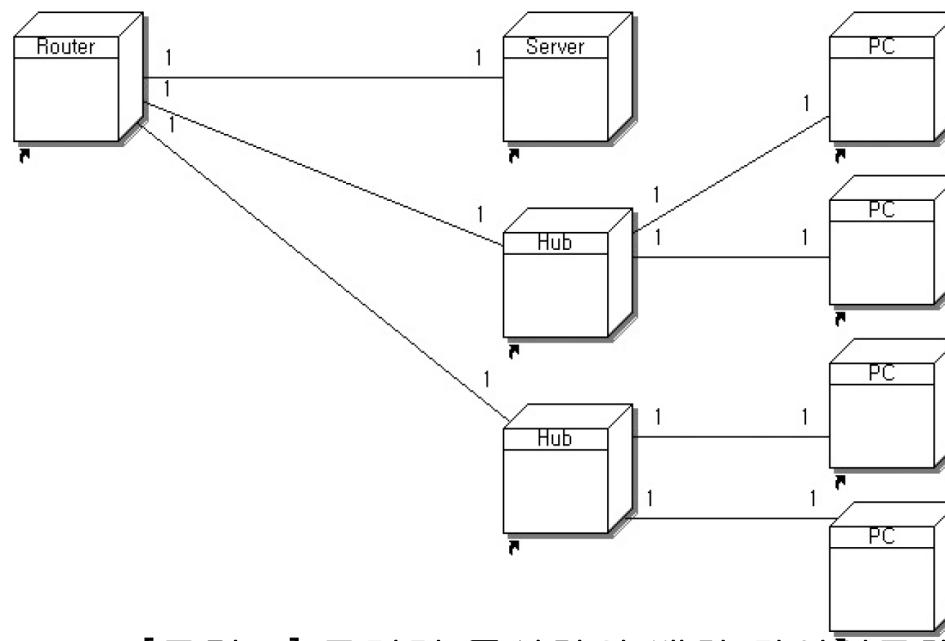


[그림 4] WWW의 배치 다이어그램

# 예제

## ◆ 근거리 통신망 배치 다이어그램

- 근거리 통신망(*LAN, Local Area Network*)은 많이 사용하는 네트워크 형태
- 라우터를 이용하여 건물이나 학교 등의 공간에서 외부 인터넷 망과 연결
- *LAN*
  - 중간 노드의 교환이 필요없이 점 대 점(*Point-To-Point*) 방식의 고유 물리적 매체를 이용하여 통신

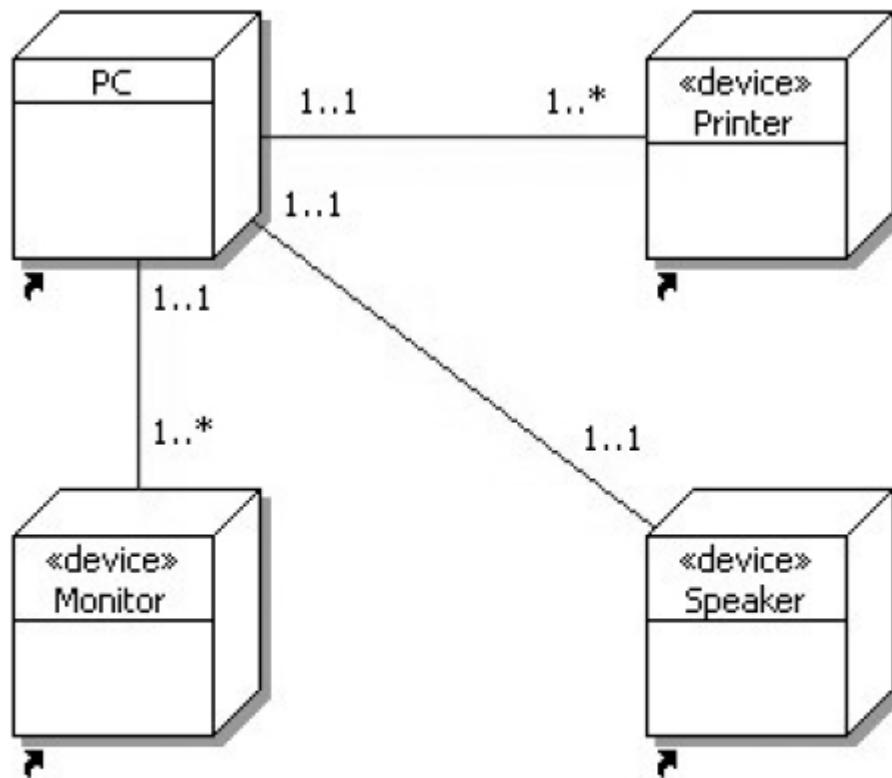


[그림 5] 근거리 통신망의 배치 다이어그램

# 예제

## ◆ 컴퓨터 구성 배치 다이어그램

- 컴퓨터의 구성
  - ▶ PC 프로세스, 모니터, 스피커, 프린트

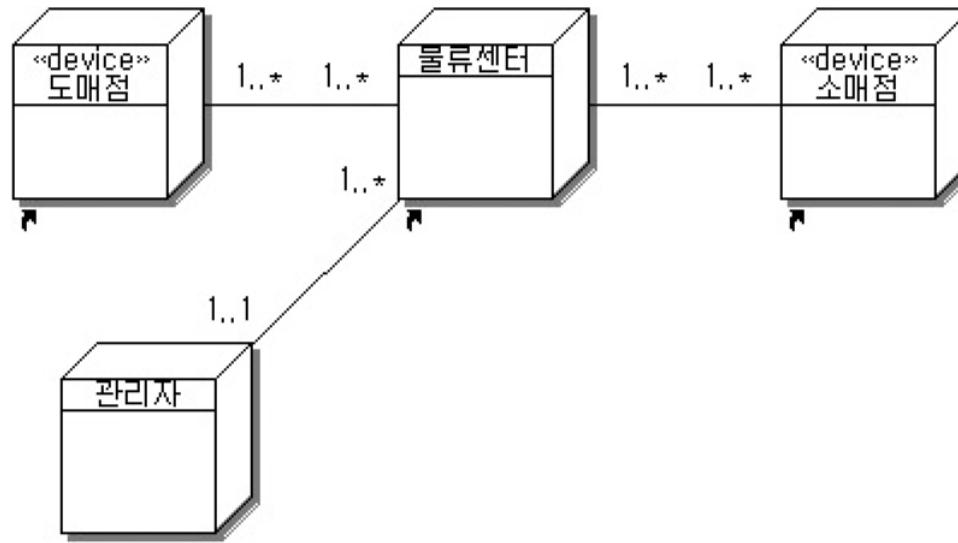


[그림 6] 컴퓨터 구성 배치 다이어그램

# 예제

## ◆ 발주 시스템

- 도매점과 소매점이 있고, 이들을 연결하여 주는 물류센터가 있다.
- 물류센터는 해당 관리자가 관리하여, 도매와 소매의 물류량 조절

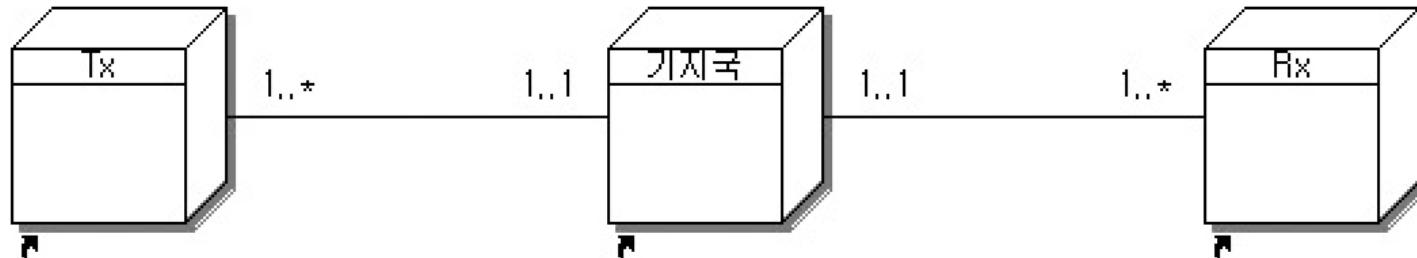


[그림 7] 발주 시스템 배치 다이어그램

# 예제

## ◆ 통신원리

- 무선기를 통하여 데이터를 송신하면 무선 네트워크를 통해 각 파트별 관리자 의 PC에 전송
- 데이터를 전송받은 각 파트의 관리자 PC는 해당 데이터 처리

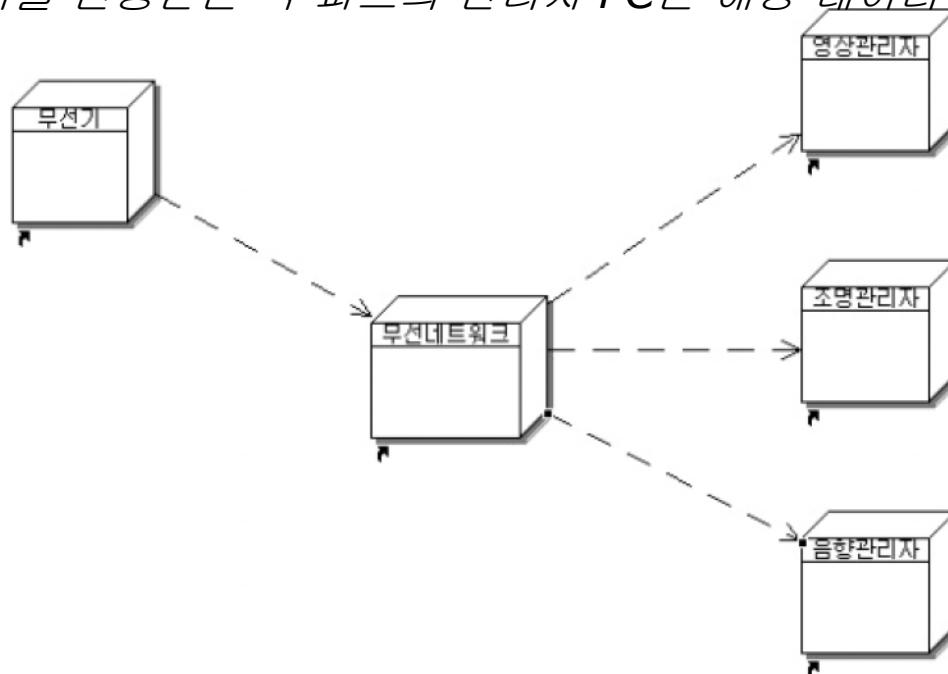


[그림 8] 통신원리 배치 다이어그램

# 예제

## ◆ 방송 무선 네트워크 통신망

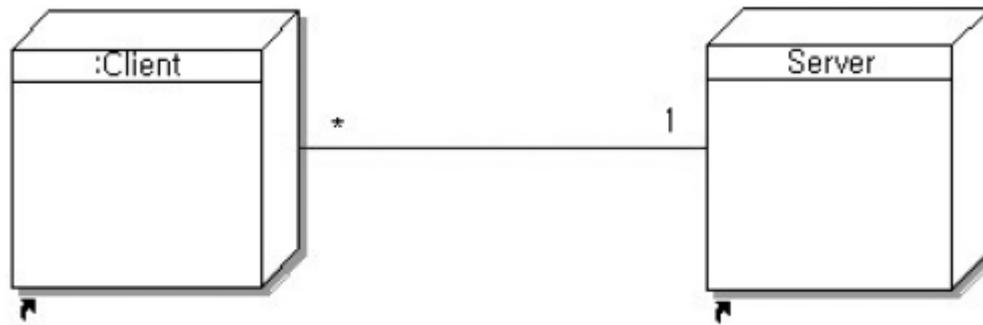
- 무선기를 통하여 데이터를 송신하면 무선 네트워크를 통해 각 파트별 관리자 의 PC에 전송
- 데이터를 전송받은 각 파트의 관리자 PC는 해당 데이터 처리



[그림 9] 방송 무선 네트워크 배치 다이어그램

# 결합 모델

- ◆ 컴포넌트 다이어그램과 배치 다이어그램의 결합된 표현
  - 컴포넌트 다이어그램과 배치 다이어그램의 2가지 물리적인 다이어그램 표기법을 결합하는 것
  - 노드의 내용을 보여주기 위해 확장된 노드 안에 컴포넌트 아이콘을 모델링
  - [그림 4]
    - 메신저의 서버와 클라이언트 간의 접근 방법을 배치 다이어그램으로 표현



[그림 10] 메신저 시스템

# Package Diagram

# 패키지

§ 요소들을 그룹으로 조직하기 위한 범용 메커니즘

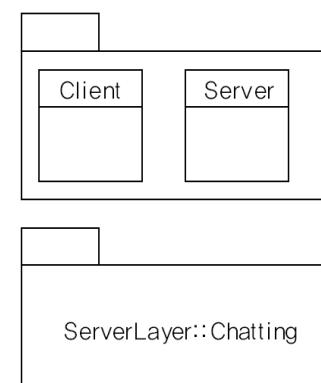
§ 탭이 달린 폴더로 표현

§ 패키지 표기법

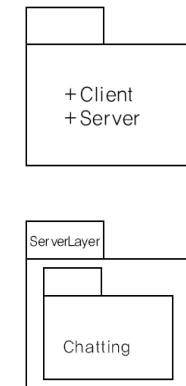
ü 아이콘 안에 패키지 이름만을 표기한 단순표 기법과 패키지에 포함된 내부 패키지나 클래스까지 표현한 확장표기법



패키지의 단순표기법



패키지의 확장표기법



# 관계

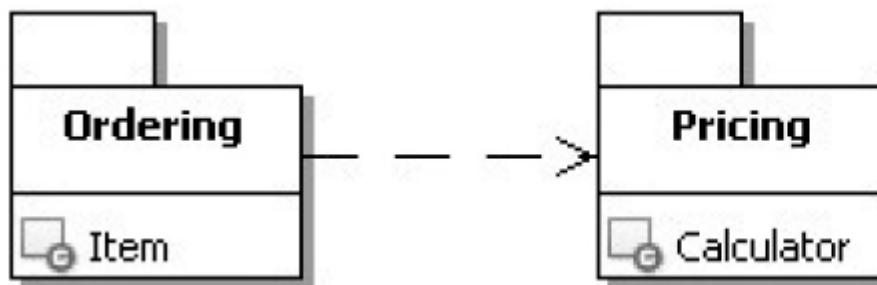
## ● 일반화 관계

§ 패키지들간의 상속

## ● 의존관계

§ 수입(import)과 접근(access) 관계

§ 한쪽이 export 한 것을 다른 한쪽의 패키지 요소가 import 함

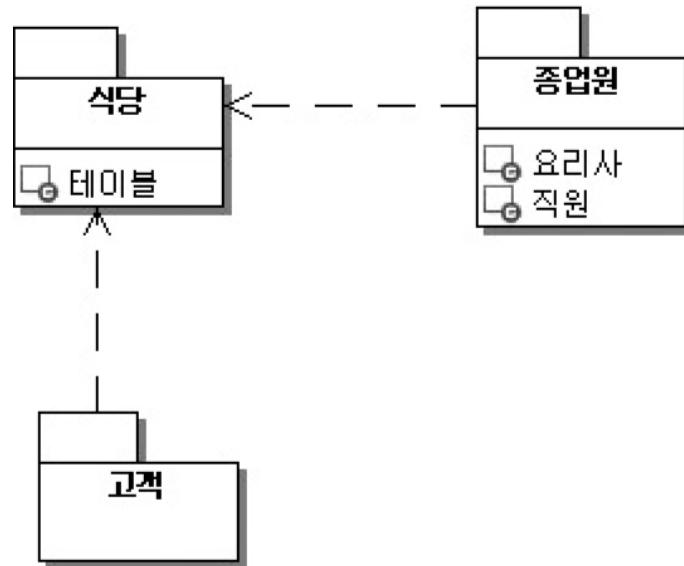


패키지간의 의존관계

# 예제

## 식당관리

- § 식당을 운영하는 종업원이 관리
- § 식당을 이용하는 고객은 종업원을 통해 메뉴를 주문
- § 주문된 메뉴는 요리사를 통해 요리
- § 고객은 식당에 마련된 테이블에서 식사

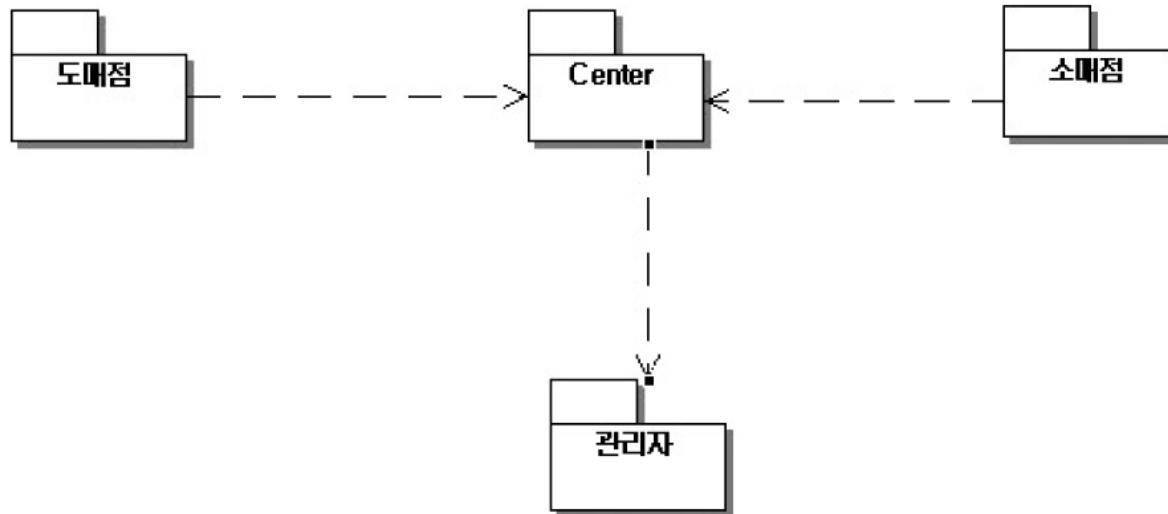


식당관리 패키지 디아이어그램

# 예제

## ● 물류유통과정

- § 도매점에서 출하한 물품을 센터에서 입하
- § 관리자(Admin)를 통해 물품을 관리하고 소매점에 발주
- § 소매점에서는 물품을 입하하고 판매

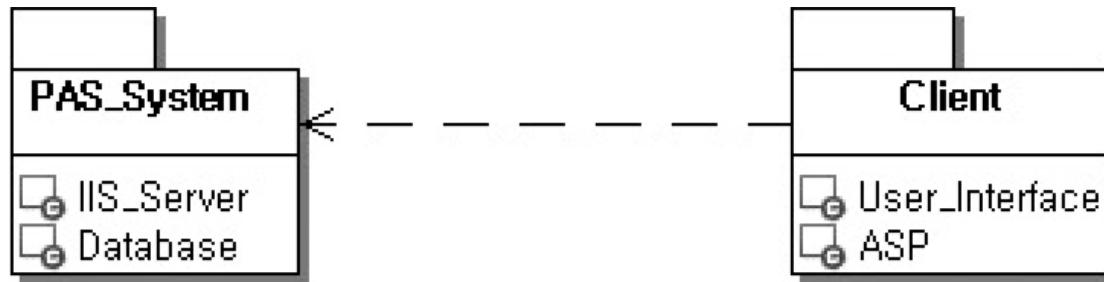


물류유통 과정 패키지 다이어그램

# 예제

## ● 프로젝트 관리 서비스

- § 클라이언트를 통해 작품을 입력·수정
- § PAS\_System 내에 있는 데이터베이스에서 데이터 관리

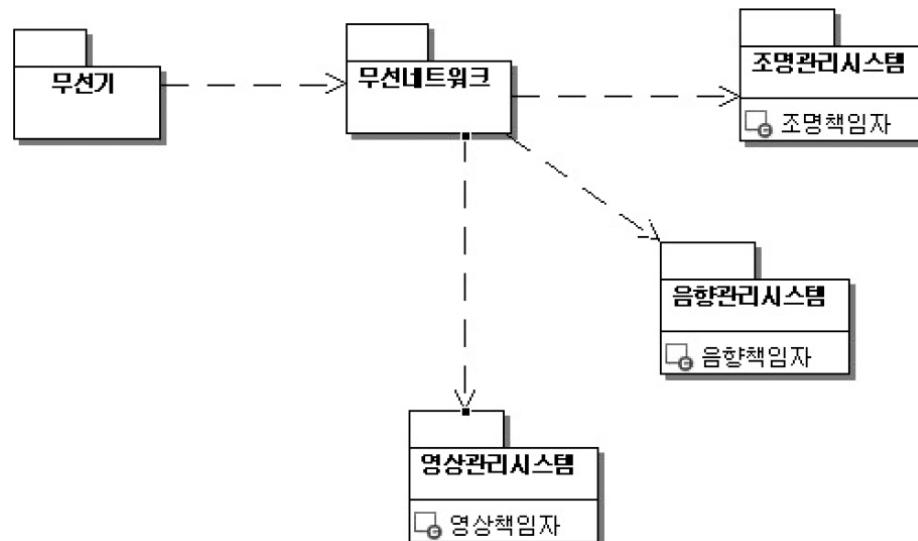


프로젝트 관리 서비스 패키지 다이어그램

# 예제

## ● 방송 무선 네트워크 통신망

- § 무선기를 통해 파트별 책임자에게 무선으로 통신
- § 무선기를 통해 전달되는 데이터는 무선 네트워크를 통해 각 파트별 관리자에게 전달
- § 해당 책임자가 데이터를 바탕으로 일을 진행



방송 무선 네트워크 통신망 패키지 디아어그램