5.2 UML(Unified Modeling Language)

- 객체지향 소프트웨어를 모델링 하는 표준 그래픽 언어
  - 시스템의 여러 측면을 그림으로 모델링
  - 하드웨어의 회로도 같은 의미

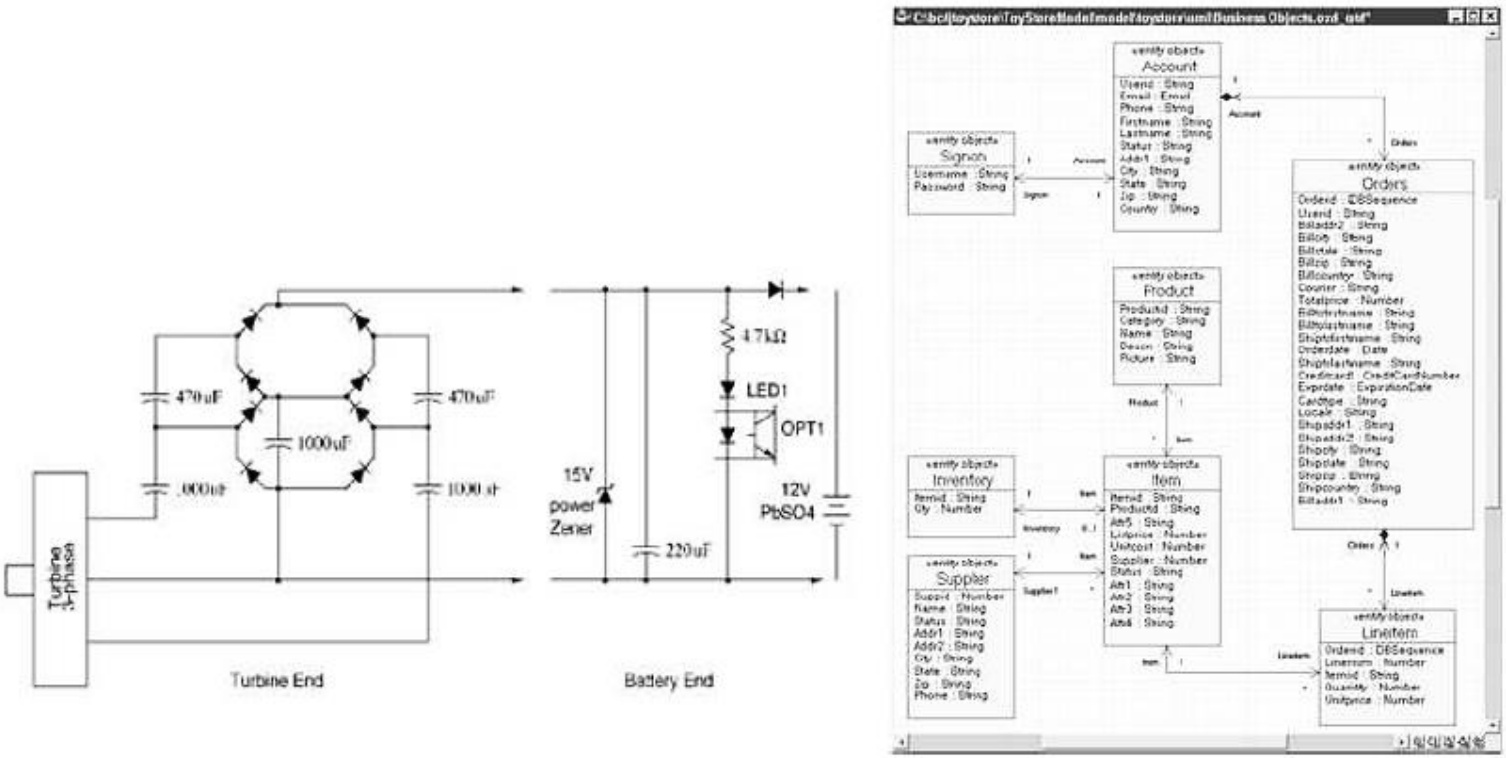


그림 5.8 ▶ UML과 회로도

UML의 배경과 역사

- **1988년부터 1992년까지 제안된 객체지향 분석 및 설계 방법**
  - Sally Shlaer와 Steve Mellor가 저술한 두 책에서 제안된 분석 및 설계한 이 방법은 재귀적 접근 방법(recursive design)으로 발전 [Shlaer, 1997]
  - Peter Coad와 Ed Yourdon이 제안한 프로토타입 중심의 가벼운 방법 [Coad,1991a][Coad, 1991b][Coad, 1995]
  - Smalltalk 그룹에서 제안한 의무 중심 (Responsibility-Driven) 설계 및 클래스-의무-협동 (Class-Responsibility-Collaboration) 방법[Beck, 1989]
  - Rational Software에서 일한 Grady Booch의 방법[Booch, 1994, 1995]
  - General Electric 연구소의 Jim Rumbaugh의 팀에서 제안한 객체 모델링 테크닉(Object Modeling Technique) 방법[Rumbaugh 1991, 1996]
  - Jim Odell이 James Martin과 함께 정보 공학 방법론을 기초로 제안한 방법[Martin, 1994, 1996].
  - Ivar Jacobson이 UseCase 개념을 소개한 방법[Jacobson, 1994, 1995].

UML의 배경과 역사

- UML은 OMT(Object Modeling Technique)[Rumbaugh, 1991]와 Booch[Booch,1994], OOSE(Object-Oriented Software Engineering)[Jackson, 1992] 방법의 통합으로 만들어진 표현

OMG: Object Management Group

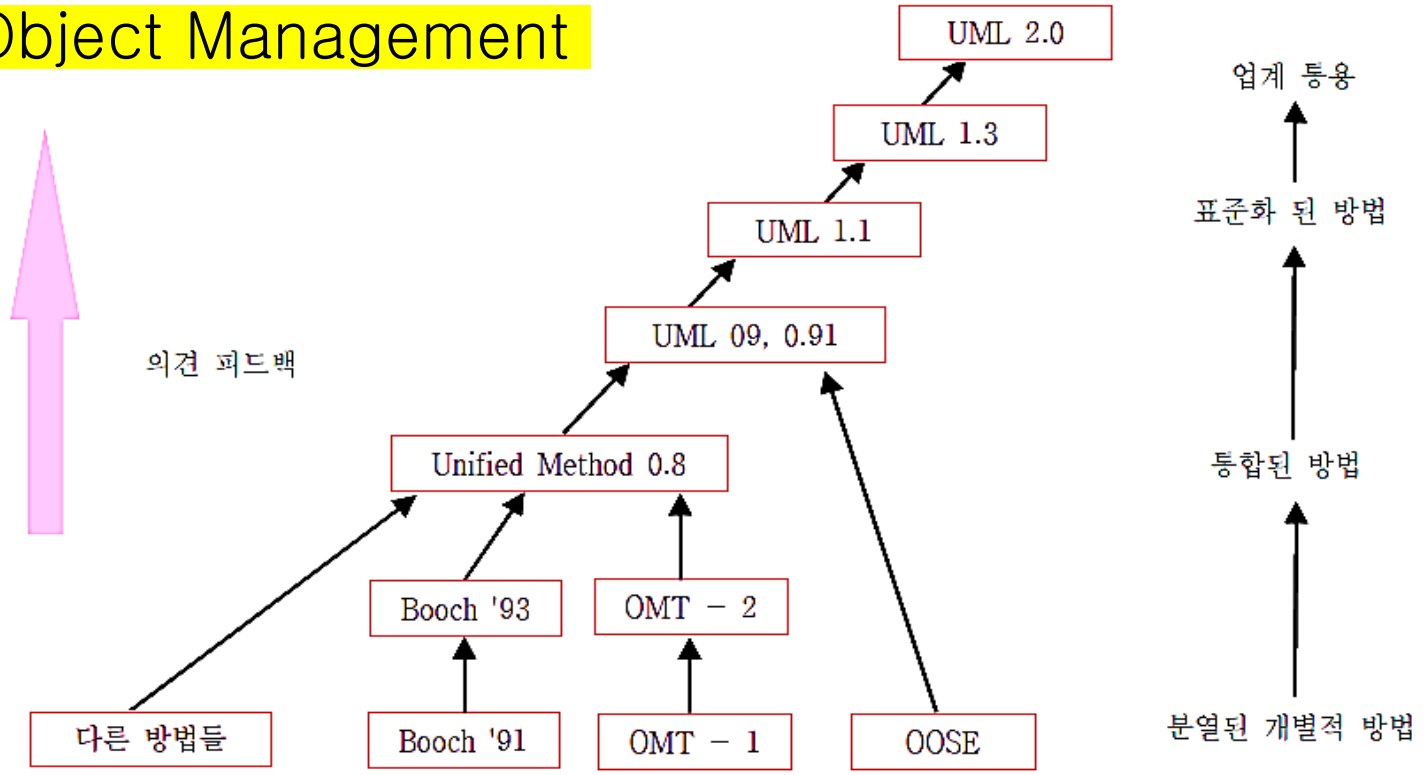


그림 5.9 ▶ UML의 진화

UML 다이어그램

- 시스템의 모델링은 기능적 관점, 구조적 관점, 동적 관점으로 구성

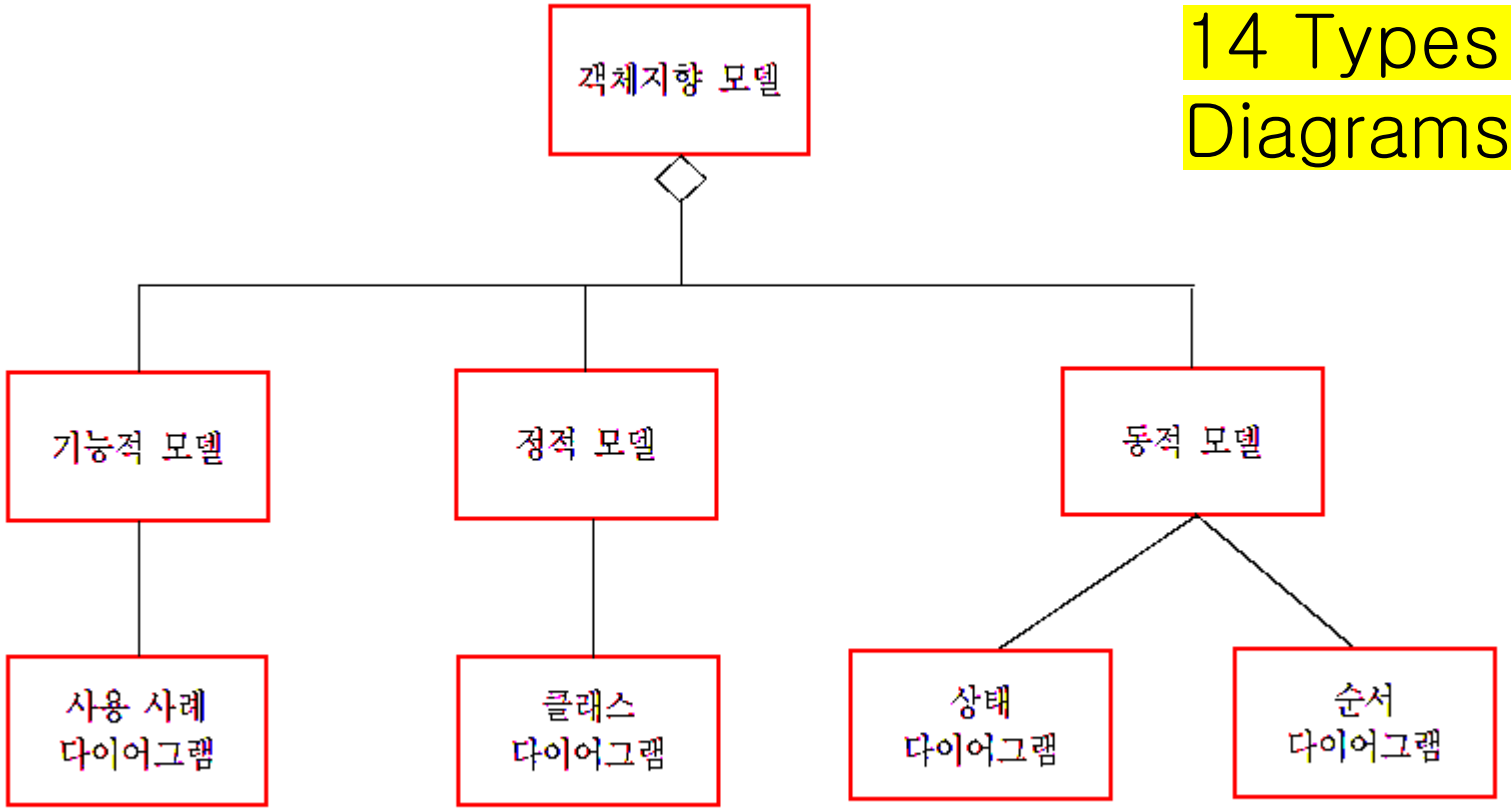


그림 5.10 ▶ 세 가지 측면의 모델

14 Types of Diagrams

UML 다이어그램

- **UML 다이어그램과 모델링**

다이어그램 이름	설명	모델링 적용
사용 사례 (use case) 다이어그램	업무 프로세스를 나타내는 사용 사례와 액터가 정점에 표시된 그래프 간선은 어떤 액터가 업무 프로세스와 상호작용하는지 나타냄	현재 존재하는 어플리케이션이나 사용자가 개발 요구한 시스템의 **업무 프로세스의 개관을** 나타내는 데 사용됨 시스템 또는 서브시스템의 범위를 나타냄
클래스 다이어그램	정점은 클래스. 방향이 있는 간선에는 클래스의 관계를 나타내는 방향성 그래프. 정점에는 클래스가 가지고 있는 속성과 오퍼레이션의 정보가 표시되어 있음	도메인 모델을 나타내는데 사용. 개발자가 **도메인 개념과 이들 사이의 관계**를 이해하고 전달하는 데 도움이 됨
시퀀스 다이어그램	정점은 객체를 나타냄. 방향성 있는 간선은 객체 사이에 오가는 메시지를 시간 순으로 나타냄	개발팀이 현재의 **업무프로세스를 이해하고 분석하는** 데 도움이 됨

UML 다이어그램

- **UML 다이어그램과 모델링**

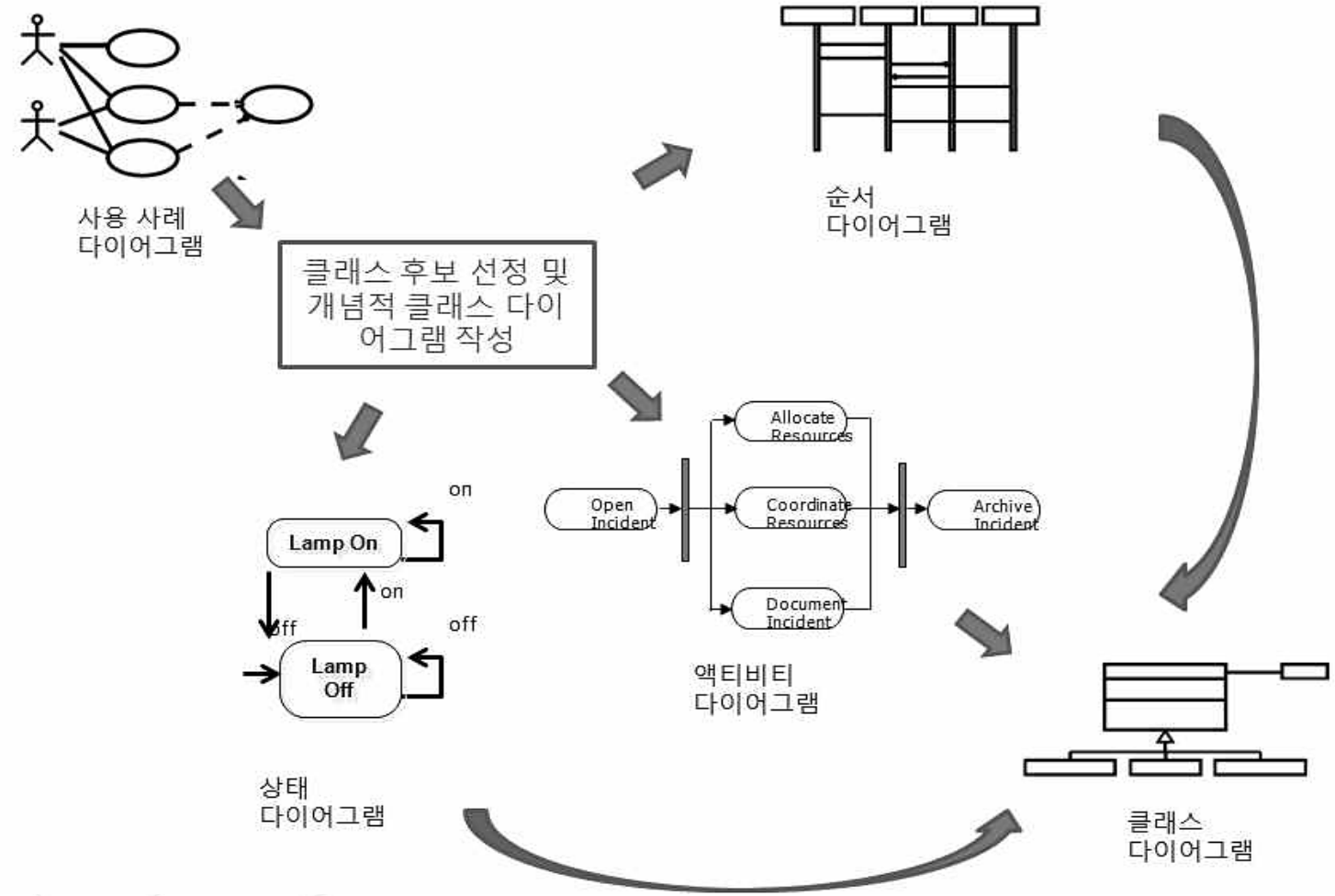
다이어그램 이름	설명	모델링 적용
상태 다이어그램	정점에는 시스템의 상태. 방향이 있는 간선에는 상태의 변환을 나타낸 방향성 있는 그래프	상태 의존적이며 반응적인 시스템 또는 서브시스템의 동작을 나타내는 데 사용
액티비티 다이어그램	각 정점은 정보를 처리하는 작업을 나타내며, 방향이 있는 간선은 자료 및 제어 흐름을 나타내는 방향성 있는 그래프. 제어흐름은 순차, 병렬, 동기화를 나타냄	시스템 또는 서브시스템의 복잡한 작업 흐름을 나타내는 데 사용됨
패키지 다이어그램	정점은 클래스의 묶음인 패키지, 방향이 있는 간선은 패키지의 의존관계를 나타냄	복잡한 클래스를 묶어서 서브시스템으로 조직화하는 데 사용
배치 다이어그램	정점은 분산 시스템의 물리적인 컴퓨팅 파워와 그 위에 실행되는 컴포넌트를 나타내며, 간선은 네트워크 연결을 나타냄	배피 다이어그램은 분산 시스템의 각 컴퓨팅 노드, 컴포넌트, 커넥터 등 시스템의 물리적 자원 배치를 나타내는 데 사용됨

UML 모델링 과정

1. 요구를 사용 사례로 정리하고 사용 사례 다이어그램을 작성
2. 클래스 후보를 찾아내고 개념적인 객체 모형을 작성
3. 사용 사례를 기초하여 순서 다이어그램을 작성
4. 클래스의 속성, 오퍼레이션 및 클래스 사이의 관계를 찾아 객체 모형을 완성
5. 상태 다이어그램이나 액티비티 다이어그램 등 다른 다이어그램을 추가하여 UML 모델을 완성
6. 서브시스템을 파악하고 전체 시스템 구조를 설계
7. 적당한 객체를 찾아내거나 커스텀화 또는 객체를 새로 설계

UML 모델링 과정

- **UML 모델링 과정**

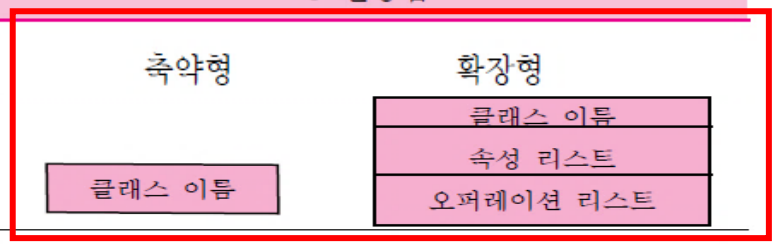
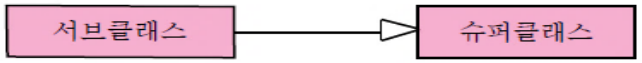
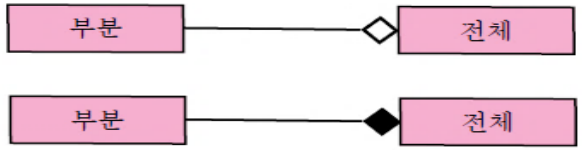


UML 모델링 과정

- 성공적인 모델링을 위한 기타 조건
  - 복잡한 문제라면 도메인을 잘 아는 전문가와 같이 모델링 함
  - 각 모델의 목적을 잘 이해하고 모델링을 위하여
    어떤 정보가 필요한지 잘 알아둠
  - 한번 그린 모델로 만족하지 않고 계속 논의하고 향상시켜 나감
    (이를 리팩토링이라 함)
  - 소그룹 회의를 열어 모델을 칠판에 그리고 토의
  - 디자인 패턴을 잘 숙지하고 필요하면 이를 이용

5.3 정적 모델링

- 정적 모델 : 시간의 개념이 개입되지 않은 모델
  - 클래스 다이어그램이 대표적
    - 클래스 다이어그램의 표현 요소

요소	의미	표현방법
클래스 속성 오퍼레이션	클래스는 타입이며 속성과 오퍼레이션을 클래스의 객체의 특징을 정의한다.	
상속	두 클래스의 일반화 및 상세화 관계	
집합	두 클래스 사이의 전체-부분 관계	
연관 방향 다중도 역할 연관 클래스	두 클래스 사이의 관계 연관을 나타내는 클래스	

5.3 정적 모델링

- 클래스 다이어그램
  - 객체, 클래스, 속성, 오퍼레이션, 연관 등을 표현

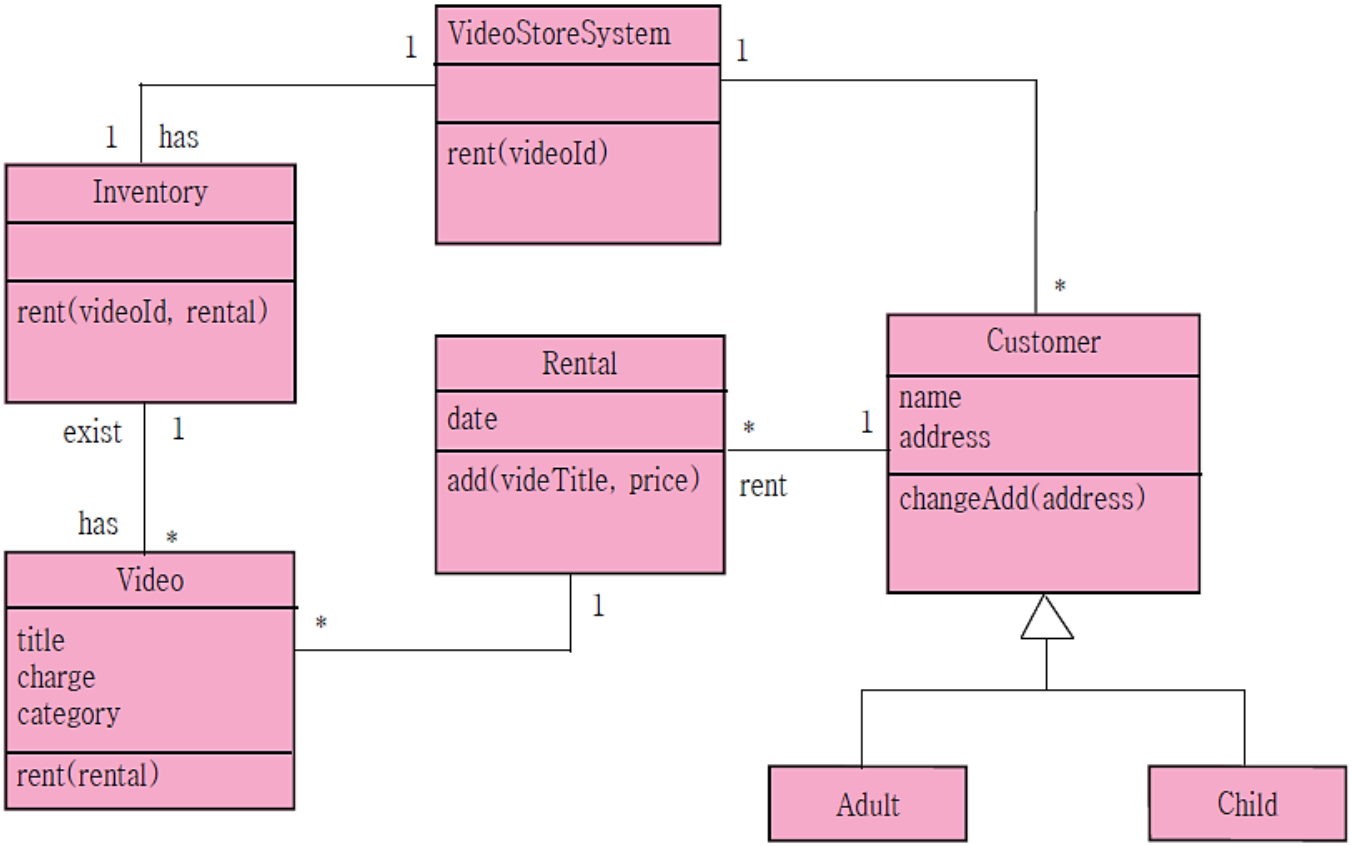


그림 5.12 ▶ 클래스 다이어그램의 예

클래스의 표현

- 클래스 심볼
  - 세 개의 부분으로 나누고 맨 위는 클래스의 이름,
    중간에는 클래스의 속성, 아래 부분은 오퍼레이션을 적음
  - 추상클래스는 이탤릭체, 인터페이스 클래스는 <<interface>>추가
- 속성 : 객체가 가지는 모든 필드를 포함
- 오퍼레이션/메소드
  - 아주 흔한 메소드(get/set)는 생략

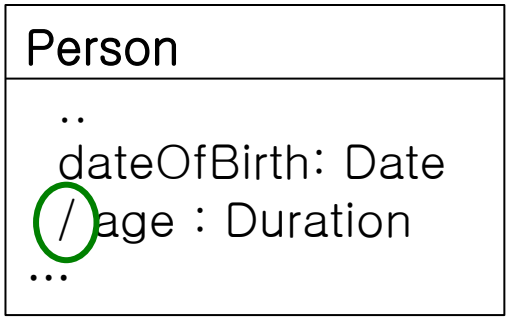


그림 5.15

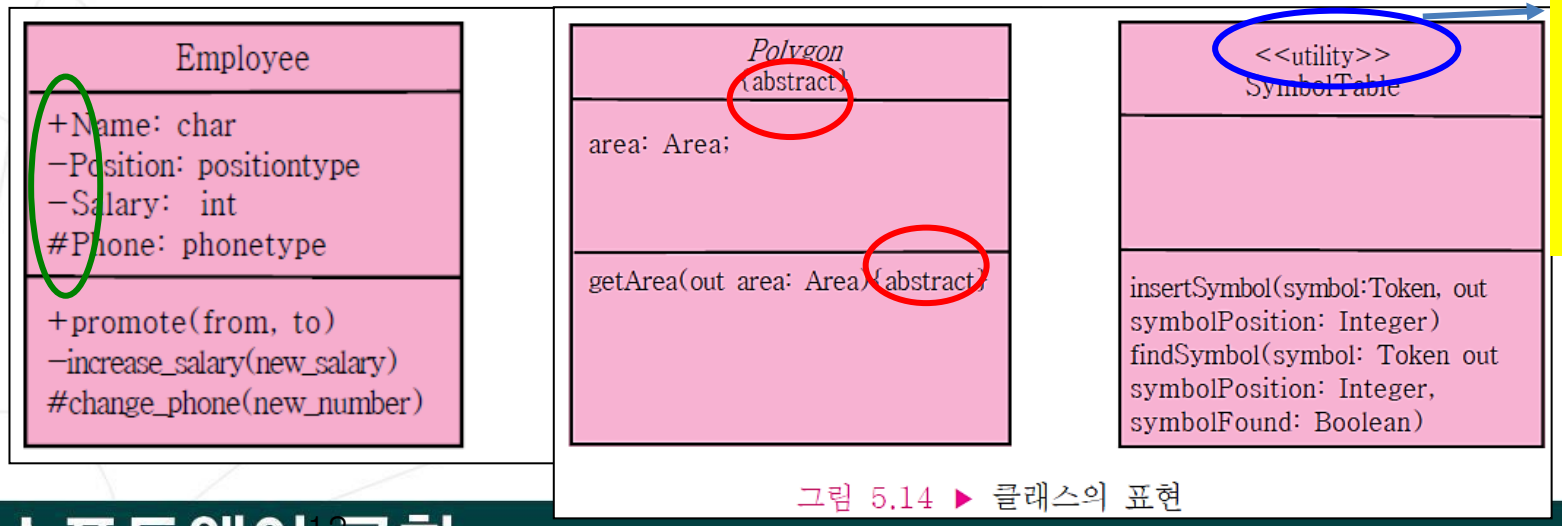


그림 5.14 ▶ 클래스의 표현

관계의 표현

- 연관(association):
  - 객체 사이에 관련
  - 예> Inventory와 Video 클래스 사이의 연관 관계

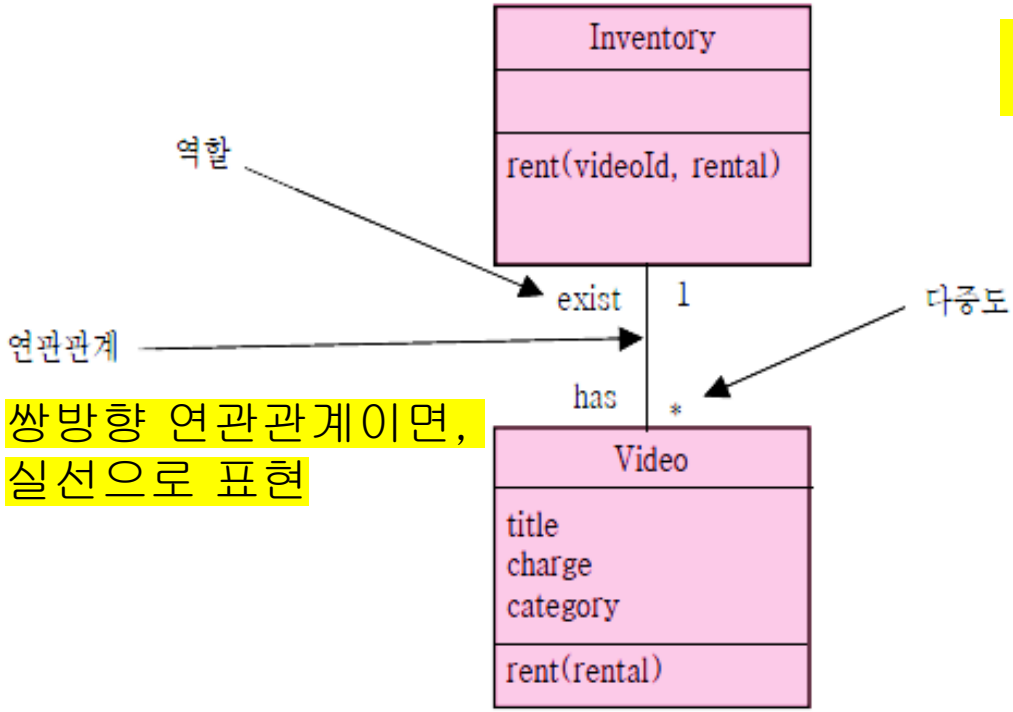
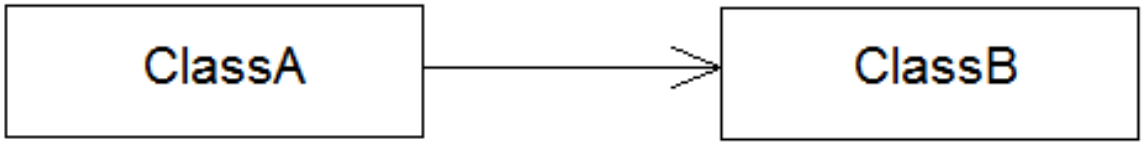


그림 5.16 ▶ 연관의 표현

쌍방향 연관관계이면, 실선으로 표현

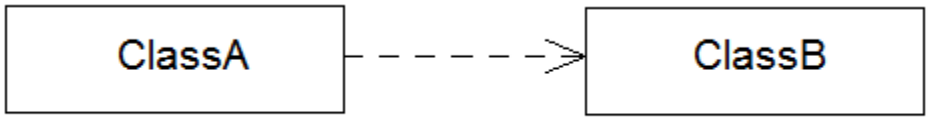
다중도 예)
1
1..5
1..*
0..10
0..*



(일반) 연관 관계

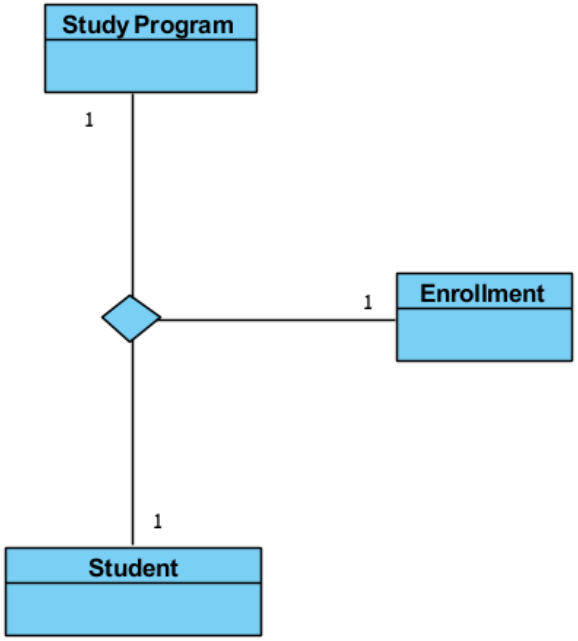
```
public class ClassA
{
    ClassB theClassB
= …
    …
}
```

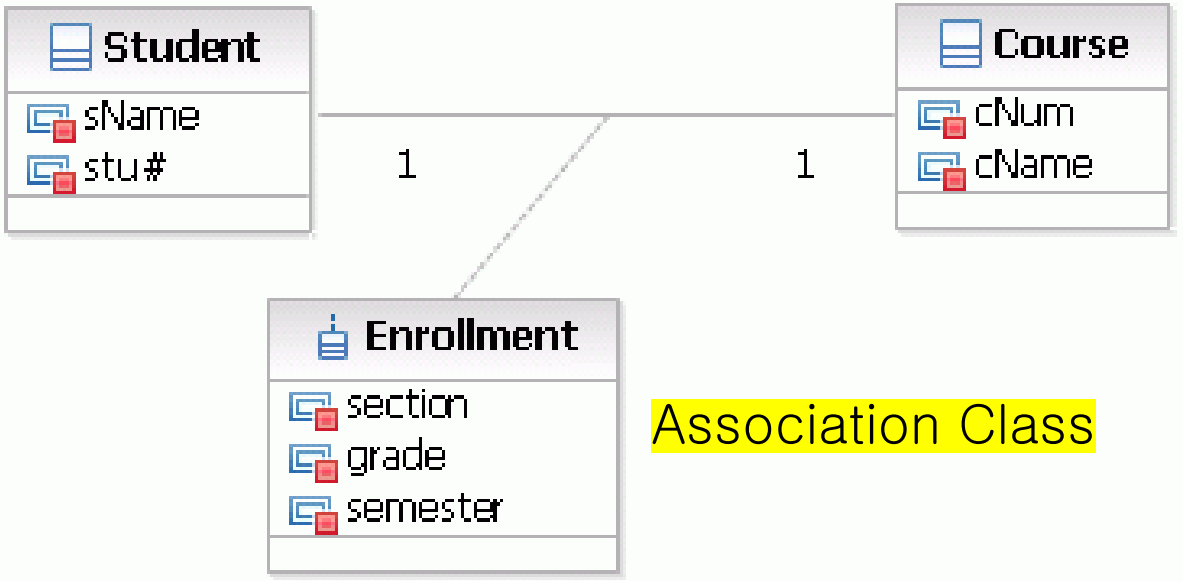
종속 관계




```
public class ClassA {
    …
    public void someMethod(ClassB arg1) {…}
    …
    public void someOtherMethod() {
      ….
    }
    …
}
```

N-ary Association





Association Class

Association Class:

a UML construct that enables an association to have attributes and operations.