2023학년도 2학기

소프트웨어 공학

6. Unified Modeling Language (UML)

SW융합대학 컴퓨터공학과
남 재 현 (jaehyun.nam@dankook.ac.kr)

목 차

UML의 이해

- Unified Modeling Language (UML)
  - 시스템 개발을 위한 시각적인 설계 표기 제공
  - 객체 지향 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화하는 데 사용
  - 개발하는 시스템을 이해하기 쉬운 형태로 표현
  - 분석가, 설계자, 의뢰인 등이 효율적으로 의사소통 할 수 있게 해 줌

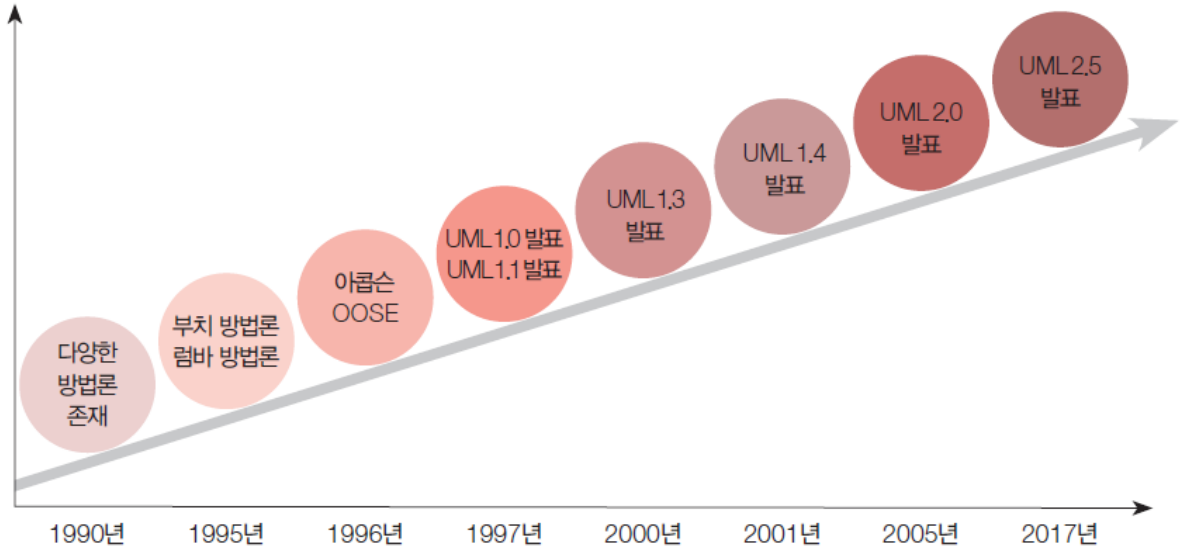


그림 1-2 UML의 발전 과정

UML의 이해

- UML이 제공하는 표준화된 다이어그램
  - 유스케이스 다이어그램 (Use-case Diagram)
  - 클래스 다이어그램 (Class Diagram)
  - 순차 다이어그램 (Sequence Diagram)
  - 통신 다이어그램 (Communication Diagram)
  - 활동 다이어그램 (Activity Diagram)
  - 상태 다이어그램 (State Diagram)
  - 컴포넌트 다이어그램 (Component Diagram)
  - 배치 다이어그램 (Deployment Diagram)
  - 패키지 다이어그램 (Package Diagram)

UML의 이해

- UML의 특징
  - 시각화Visualization 언어
    - 소프트웨어의 개념 모델을 시각적인 형태로 표현하며 명확히 정의된 표준화된 다이어그램을 제공
    - 이를 이용해 오류 없는 원활한 의사소통 가능
  - 명세화Specification 언어
    - 소프트웨어 개발 과정인 분석, 설계 단계의 각 과정에서 필요한 모델을 정확하고 완전하게 명세화
    - 명세화에서 각 다이어그램의 기호는 의미를 담고 있으며 추상적이지만 고유의 특성을 갖고 있음
  - 구축Construction 언어
    - 자바Java, C++, 비주얼 베이직Visual Basic, C# 같은 다양한 프로그래밍 언어로 표현가능
    - UML로 설계된 모델을 프로그램 코드로 자동 변환할 수 있으며,
      이미 구축된 소스 코드를 UML로 역변환하여 분석하는 역공학Reverse Engineering도 가능
  - 문서화Documentation 언어
    - StarUML, 투게더Together 등 케이스 툴CASE Tool을 이용하여 설계한 내용을 자동으로 문서화

UML의 이해

- UML과 모델링

```c
#include <stdio.h>
int main() {
      int a, b;
      scanf("%d %d", &a, &b);
      printf("a + b = %d", a + b);
 }
```

(a) C로 구현한 덧셈 프로그램

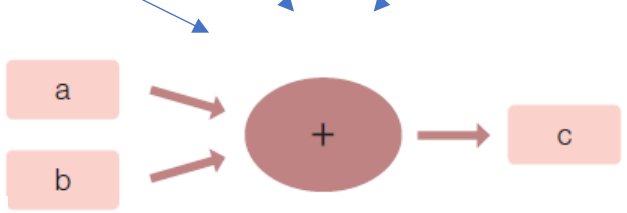
```java
import java.util.Scanner;
public class AddDemo {
      public static void main(String[] args) {
            Scanner in = new Scanner(System.in );
            int a = in.nextInt();
            int b = in.nextInt();
            System.out .printf("%d + %d = %d", a, b, a+b);
   }
}
```

(b) 자바로 구현한 덧셈 프로그램

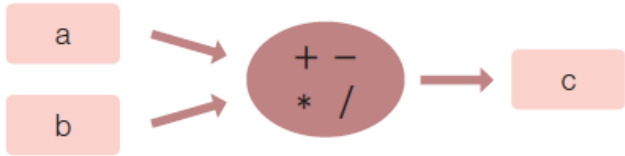
```python
def add():
      num = int(input("num1:"))
      num2 = int(input("num2:"))
      result = num + num2
      print("두수의 합은 " result)
```

(c) 파이썬으로 구현한 덧셈 프로그램

그림 1-3 다양한 언어로 구현한 덧셈 프로그램



(a) 덧셈 연산 프로그램의 시각적 표현



(b) 사칙연산 프로그램의 시각적 표현

그림 1-4 프로그램의 시각적 표현

개발하고자 하는 프로그램을 시각적으로 표현하는 것이며,
이때 의뢰자의 요구에 맞게 쉽게 수정해서
결과적으로 유지보수 기간을 줄여 생산성을 높일 수 있음

UML의 이해

- UML과 모델링

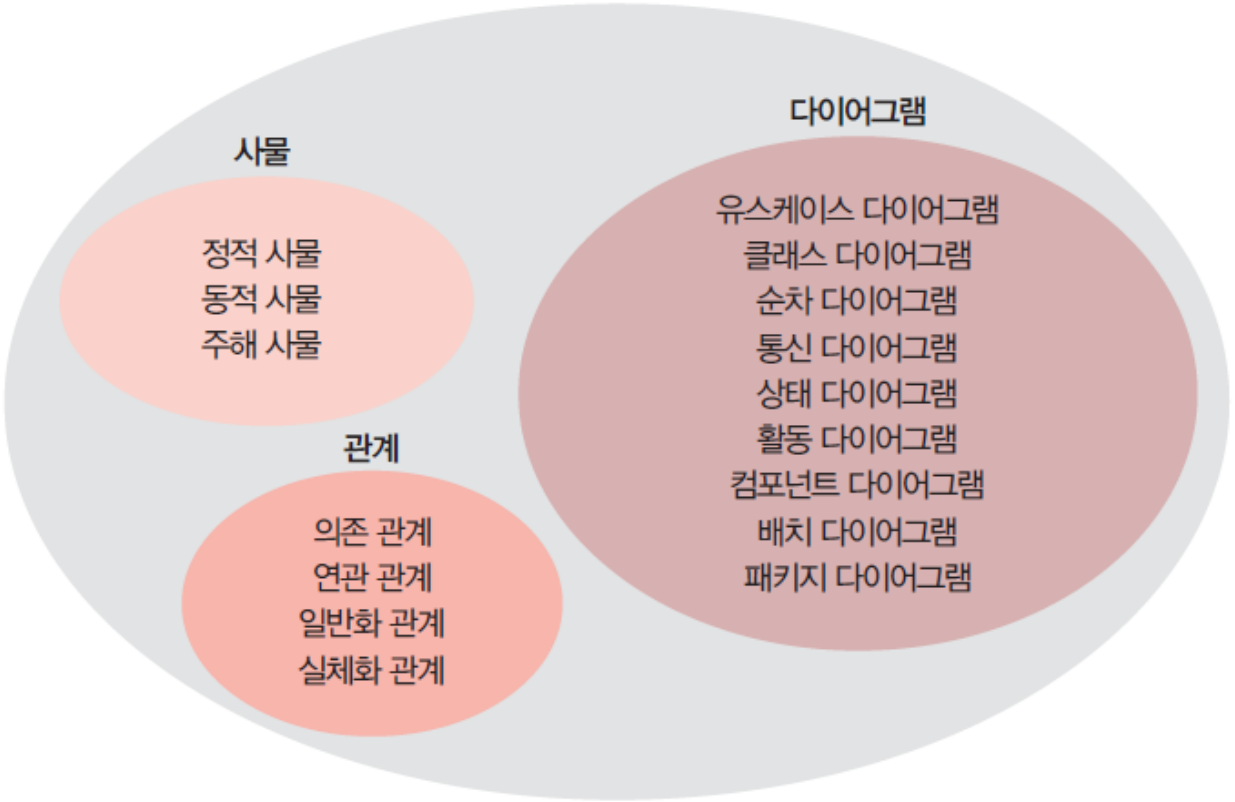
(a) 개인이 제어할 수 있는 작업 : 개집 짓기

(b) 개인이 제어할 수 없는 작업 : 큰 건축물 짓기

그림 1-5 모델링이 필요한 이유

UML의 구성 요소

- UML 구성요소
  - 사물Things, 관계Relationship, 다이어그램Diagram의 세 가지 구성 요소로 이루어짐



UML의 구성 요소

- 사물
  - 정적사물Structural Things
    - 모델의 구조, 즉 개념적 · 물리적 요소를 표현하는 명사
    - 클래스, 인터페이스, 통신, 컴포넌트, 패키지, 노드 등
  - 클래스Class
    - 동일한 속성, 오퍼레이션, 관계, 의미를 공유하는 객체를 기술한 것 (직사각형 표시)
  - 인터페이스Interface
    - 클래스 또는 컴포넌트의 서비스를 명세화 하는 오퍼레이션을 모아 놓은 것 (원 표시)
    - 특정 클래스나 컴포넌트의 전체 또는 일부 동작을 나타낼 수 있음

그림 2-2 클래스의 예

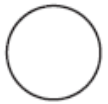


그림 2-3 인터페이스의 예

UML의 구성 요소

- 사물
  - 정적사물Structural Things
    - 통신Communication
      - 서로 다른 요소와 역할이 모여 교류Interaction를 정의 (실선 사각형 표시)
      - 동작과 구조에서 중요하며 클래스 하나가 다수의 통신에 참여할 수 있음
    - 컴포넌트Component
      - 전체 시스템을 구성하는 단위 (탭이 달린 직사각형 표시)
      - 독립적으로 개발되고 배포되며 조립, 교환이 가능한 응집도가 높은 소프트웨어 산출물

그림 2-4 통신의 예

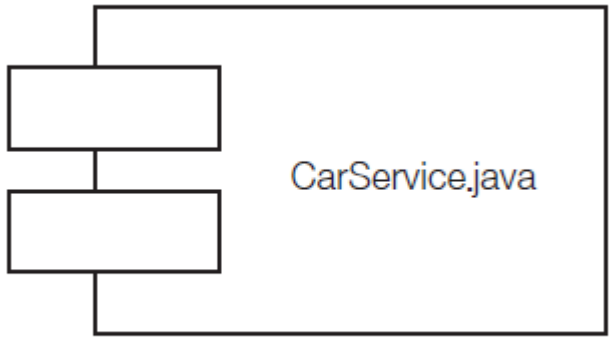
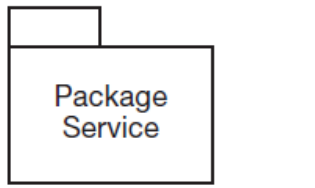


그림 2-5 컴포넌트의 예

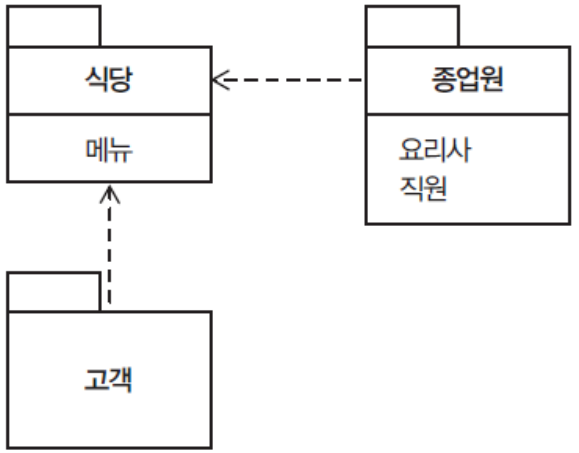
UML의 구성 요소

- 사물
  - 정적사물Structural Things
    - 패키지Package
      - 요소를 그룹으로 묶음, 정적 사물이나 동적 사물도 하나에 들어갈 수 있음 (탭이 달린 폴더 형태 표시)
      - 컴포넌트가 물리적인 데 반해 패키지는 개념적
    - 노드Node
      - 실행할 때 존재하는 물리적 요소 (육면체 표시)
      - 컴포넌트가 노드에 존재하며, 노드에서 노드로 이동할 수 있음



(a) 패키지의 예

그림 2-6 패키지 형태와 예



(b) 패키지 다이어그램의 예

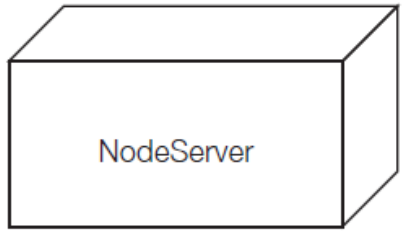


그림 2-7 노드의 예

UML의 구성 요소

- 사물
  - 동적사물Behavioral Thing
    - 주로 모델의 동적인 부분을 동사로 표시
    - 교류, 유스케이스, 상태 머신 등
  - 교류Interaction
    - 목적을 달성하기 위해 특정 문맥에 속한 객체들 간에 주고받는 메시지로 구성된 동작 (직선 표시)
  - 유스케이스
    - 시스템이 수행하는 활동들을 순차적으로 기술 (실선 타원 표시)
    - 액터Actor에게 의미 있는 결과 값을 제공

그림 2-8 메시지

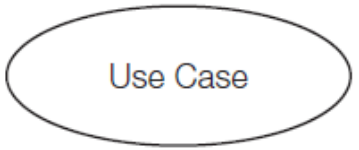


그림 2-9 유스케이스

UML의 구성 요소

- 사물
  - 동적사물Behavioral Thing
    - 상태 머신state machine
      - 외부 이벤트에 대한 객체의 상태와 상태의 변화 순서를 기술 (모서리가 둥근 직사각형 표현)
      - 서로 다른 요소들이 들어있음
        - 상태전이 : 상태에서 다른 상태로의 흐름
        - 사건 : 전이를 유발하는 것
        - 활동 : 전이에 따른 응답

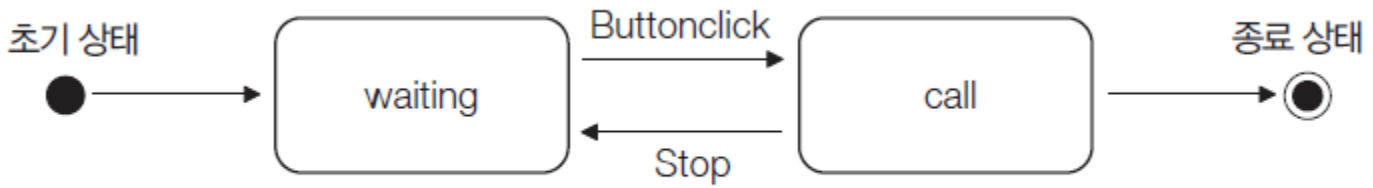


그림 2-10 상태 머신

UML의 구성 요소

- 사물
  - 주해 사물Annotation Things
    - 모델링에 참여하지는 않음
    - 모델링에 필요한 모든 정보를 표시하기 위해 사용
    - 노트가 있음

    - 노트Note
      - 첨부되는 주석 또는 제약을 기술
      - 모서리가 접힌 직사각형 표현

그림 2-11 노트

UML의 구성 요소

- 관계
  - 의존Dependency 관계
    - 두 사물 간의 의미적 관계
    - 한 사물의 명세가 바뀌면 다른 사물에 영향을 끼침
    - 반드시 반대가 성립하진 않음
    - 의존하는 사물을 향하는 점선 표현

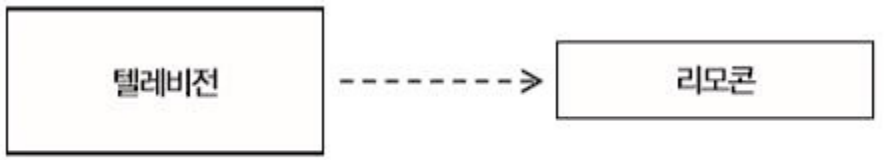


그림 2-12 의존 관계

UML의 구성 요소

- 관계
  - 연관Association 관계
    - 객체 사이의 연결 관계 (실선으로 표기)
    - 지속적으로 유지되는 관계
    - 한쪽 객체에서 다른 객체로 옮겨갈 수 있음

    - 이름
      - 연관 관계의 의미설명
      - 원하는 방향으로 방향 삼각형을 표기
    - 역할
      - 클래스 옆에 원하는 역할을 써서 연관 관계에서의 역할 표시

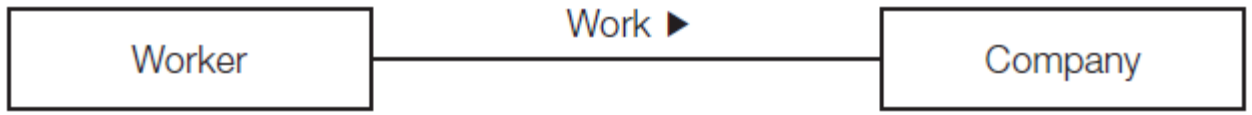


그림 2-13 이름

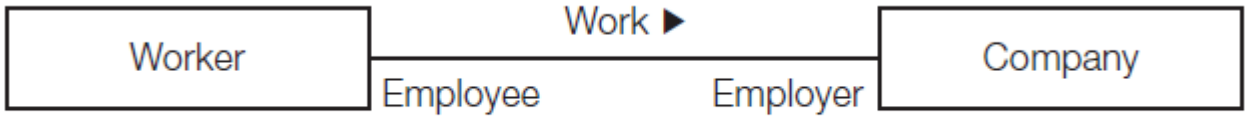


그림 2-14 역할

UML의 구성 요소

- 관계
  - 연관Association 관계
    - 다중성
      - 객체 하나에 몇 개의 객체가 연결되어 있는지를 밝히는 것
      - 범위 값으로 나타내는 표현식 (1..*)이나 명시적인 값으로 표현
        - 하나(1), 제로(0) 혹은 하나(0..1), 다수(0..*), 하나 이상 (1..*) 등으로 표현

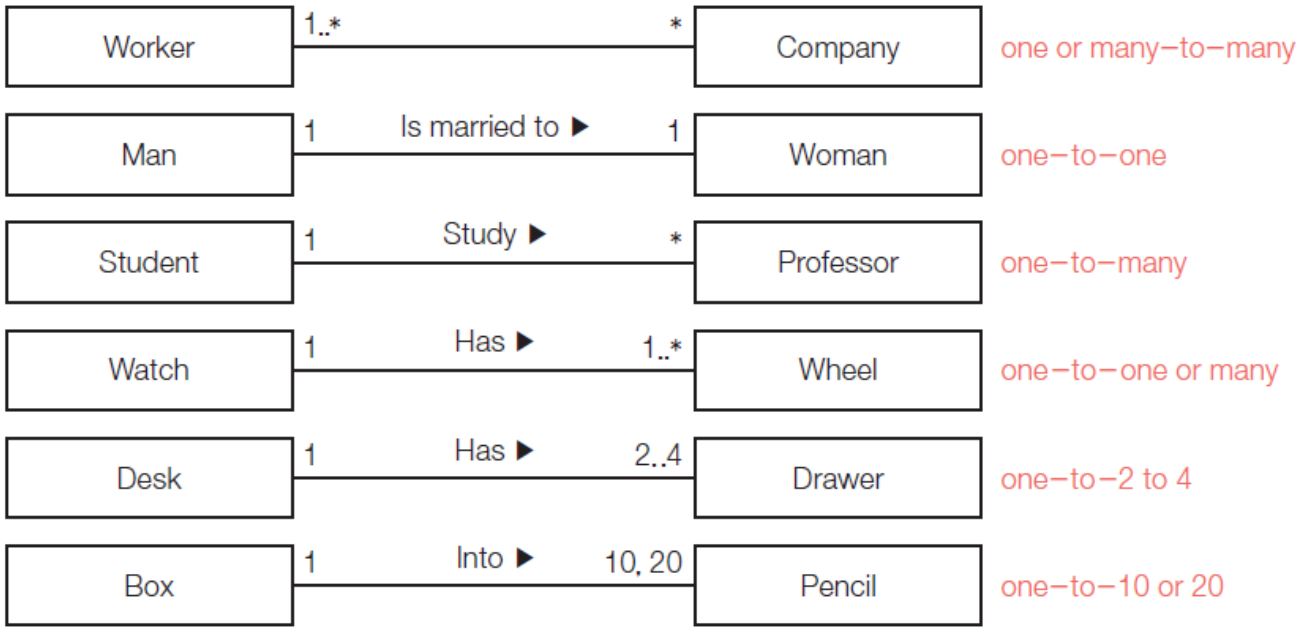


그림 2-15 다중성 관계

UML의 구성 요소

- 관계
  - 연관Association 관계
    - 집합 연관
      - 전체 쪽 객체 하나가 부분 쪽 객체들을 소유
      - has-a 관계라고도 함

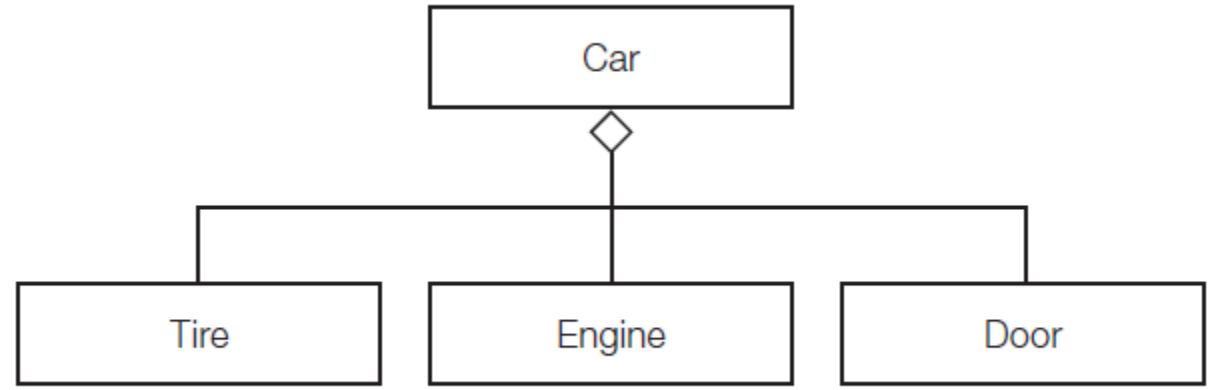
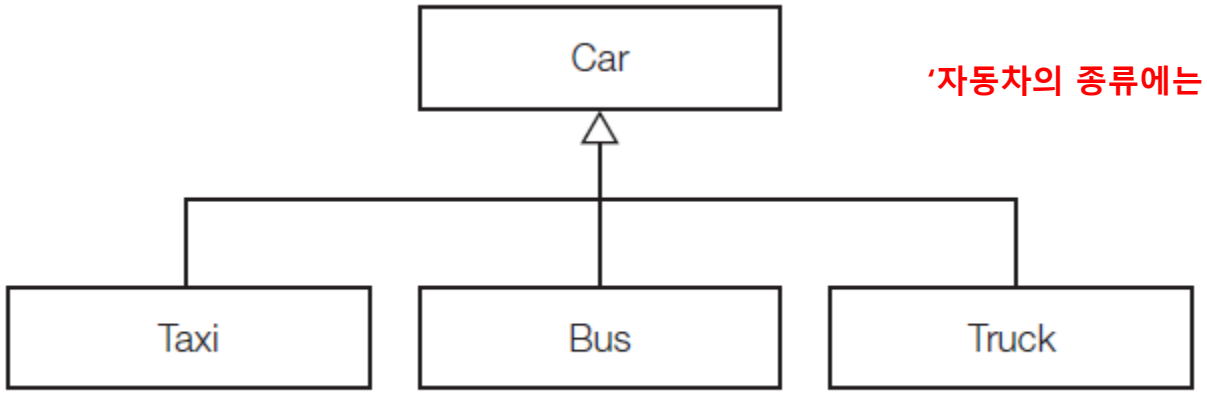


그림 2-16 집합 연관 관계

UML의 구성 요소

- 관계
  - 일반화Generalization 관계
    - 일반화된 사물과 좀 더 특수화된 사물 사이의 관계
    - 자식 객체는 부모 객체의 속성과 오퍼레이션을 상속함
    - 부모에게 없는 속성과 오퍼레이션을 가지기도 함
    - 자식 객체는 부모 객체를 대신 할 수 있으나 그 반대는 불가



'자동차의 종류에는 택시,버스,트럭이 있다.'

그림 2-17 일반화 관계

UML의 구성 요소

- 관계
  - 실체화Realization 관계
    - 한 객체가 다른 객체에게 오퍼레이션을 수행하도록 지정하는 의미적 관계
    - 인터페이스와 인터페이스에 오퍼레이션이나 서비스를 제공하는 클래스나 컴포넌트 사이의 관계를 지정
    - 클래스는 2개 이상의 인터페이스를 실체화할 수 있고, 인터페이스는 2개 이상의 클래스로부터 실체화될 수 있음

그림 2-18 실체화 관계 예 : 텔레비전과 리모콘

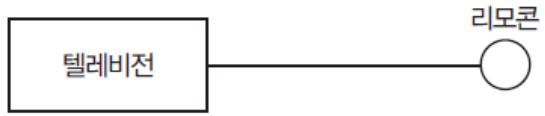
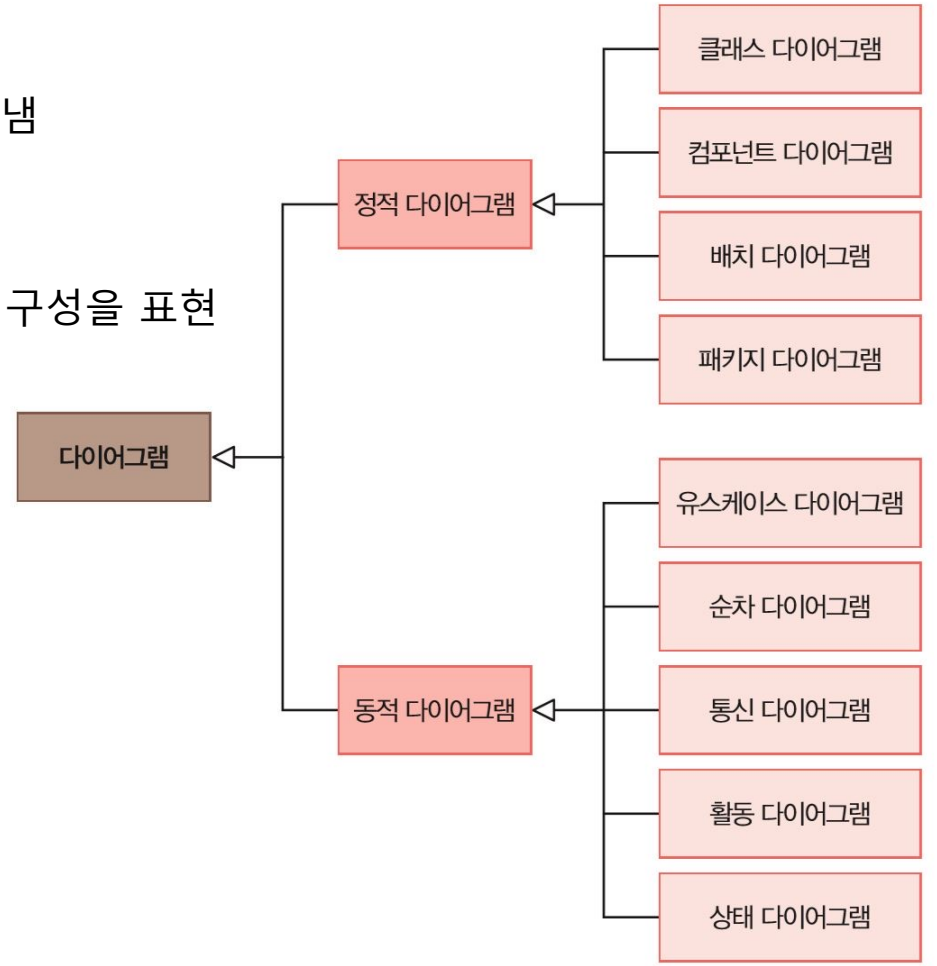


그림 2-19 실체화의 간단한 표현

그림 2-20 실체화와 의존

UML의 구성 요소

- 다이어그램
  - 클래스 다이어그램 - 클래스, 인터페이스, 통신과 함께 이들의 관계를 나타냄
  - 컴포넌트 다이어그램 - 컴포넌트 사이의 구성과 의존을 표현
  - 배치 다이어그램 - 실행 시 처리하는 노드와 그 노드에 있는 컴포넌트들의 구성을 표현
  - 패키지 다이어그램 - 여러 모델 요소를 그룹화하여 패키지를 구성하고, 이들 패키지 사이를 관계로 표현
  - 유스케이스 다이어그램 - 유스케이스와 액터의 관계를 구조적으로 표현
  - 순차 다이어그램과 통신 다이어그램 - 교류 다이어그램의 한 종류
  - 활동 다이어그램 - 시스템 내부에 있는 활동의 흐름을 표현한 것
  - 상태 다이어그램 - 시스템의 동적뷰를 나타냄



UML의 특성

- 명세서
  - 예를 들어, 클래스의 명세 표기법은 클래스 이름, 속성, 오퍼레이션 등을 표현하는 방법을 제공
- 장식
  - 중요 특징을 표현하기 위해 고유한 그래픽 표기

그림 2-24 장식

UML의 특성

- 공통 분할
  - 클래스와 객체의 분할
  - 추상개념을 구체적으로 명시

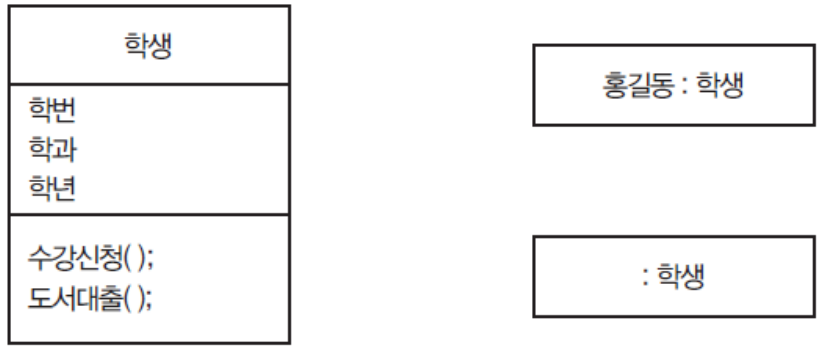


그림 2-25 클래스와 객체의 공통 분할

  - 인터페이스와 구현의 분할
  - 계약과 계약의 구체적인 실현을 명시

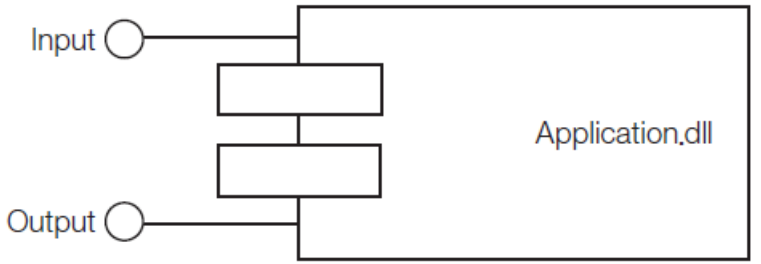


그림 2-26 인터페이스와 구현의 공통 분할

UML의 특성

- 확장
  - 스테레오 타입 << >>
    - 기본 요소 외에 새로운 요소를 만들어 내기 위한 확장
  - 꼬리표 값 {tag=value}
    - UML 구성 요소의 속성을 확장
    - 구성 요소의 명세서에 새로운 정보를 생성을 도움
  - 제약 { }
    - 이전 규칙을 수정하거나 새롭게 생성할 수 있음

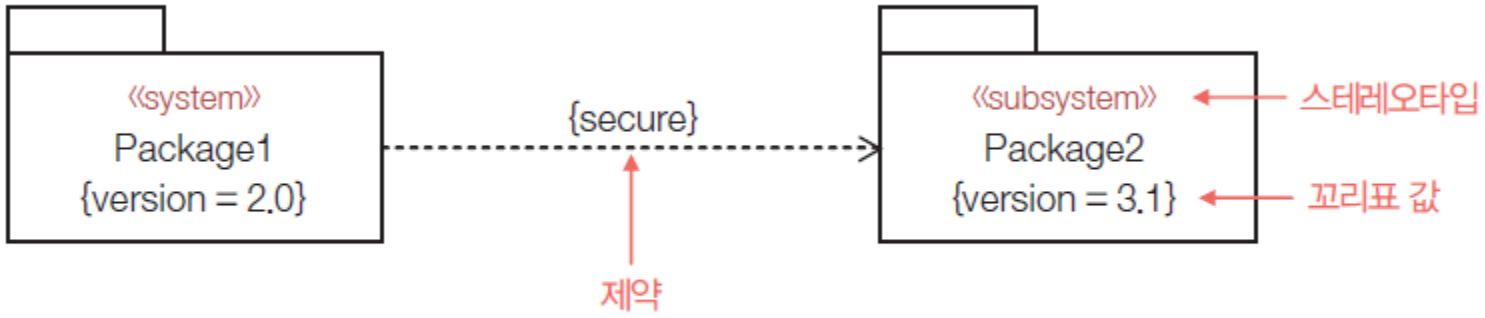


그림 2-27 확장의 예

요약 정리

- UML
  - 시스템 개발을 위한 시각적인 설계 표기 제공