모델링

- **UML 클래스 다이어그램과 순차 다이어그램을 이용하여 설계 패턴을 기술**
  - **클래스 다이어그램: 설계 패턴의 구조(클래스와 그들 간의 관계) 표현**
  - **순차 다이어그램: 설계 패턴의 행위를 표현**

자주 발생하는 문제

↳ 재사용 가능한 해결책

표현수단 → 대표적 UML

UML

- 대표적인 시스템 모델링 언어
  - 제임스 럼버 OMT + 야콥슨 OOSE + 부치 OOAD
  - 2015년 UML 2.5

구조

행위

분류	다이어그램 유형		목적
구조 다이어그램 (structure diagram)	클래스 다이어그램 (class diagram)		시스템을 구성하는 클래스들 사이의 관계를 표현한다.
	객체 다이어그램 (object diagram)		객체 정보를 보여준다.
	복합체 구조 다이어그램 (composite structure diagram)		복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	배치 다이어그램 (deployment diagram)		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	컴포넌트 다이어그램 (component diagram)		컴포넌트 구조 사이의 관계를 표현한다.
	패키지 다이어그램 (package diagram)		클래스나 유즈 케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램 (behavior diagram)	활동 다이어그램 (activity diagram)		업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	상태 머신 다이어그램 (state machine diagram)		객체의 생명주기를 표현한다.
	유즈 케이스 다이어그램 (use case diagram)		사용자 관점에서 시스템 행위를 표현한다.
	상호작용 다이어그램 (interaction diagram)	순차 다이어그램 (sequence diagram)	시간 흐름에 따른 객체 사이의 상호작용을 표현한다.
		상호작용 개요 다이어그램 (interaction overview diagram)	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
		통신 다이어그램 (communication diagram)	객체 사이의 관계를 중심으로 상호작용을 표현한다.
		타이밍 다이어그램 (timing diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

클래스 다이어그램

- 클래스 다이어그램
  - 문제나 해결책의 정적인 구조 표현
  - 클래스와 그들간의 관계 표현

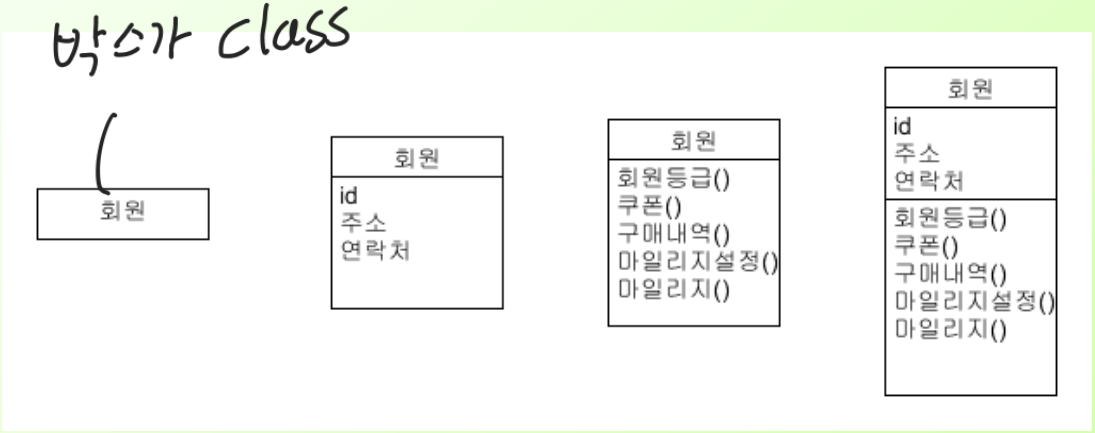
표 1-1 UML 다이어그램의 종류

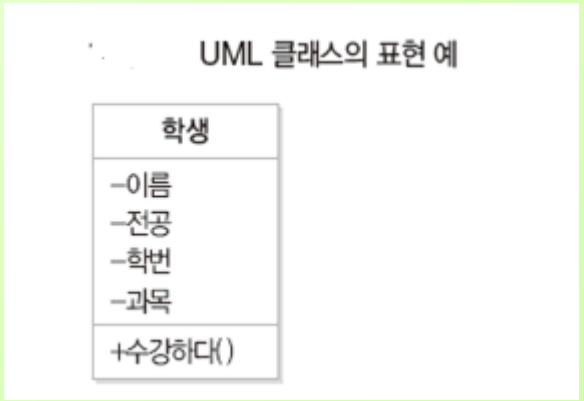
분류	다이어그램 유형		목적
구조 다이어그램 (structure diagram)	클래스 다이어그램 (class diagram)		시스템을 구성하는 클래스들 사이의 관계를 표현한다.
	객체 다이어그램 (object diagram)		객체 정보를 보여준다.
	복합체 구조 다이어그램 (composite structure diagram)		복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	배치 다이어그램 (deployment diagram)		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	컴포넌트 다이어그램 (component diagram)		컴포넌트 구조 사이의 관계를 표현한다.
	패키지 다이어그램 (package diagram)		클래스나 유즈 케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램 (behavior diagram)	활동 다이어그램 (activity diagram)		업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	상태 머신 다이어그램 (state machine diagram)		객체의 생명주기를 표현한다.
	유즈 케이스 다이어그램 (use case diagram)		사용자 관점에서 시스템 행위를 표현한다.
	상호작용 다이어그램 (interaction diagram)	순차 다이어그램 (sequence diagram)	시간 흐름에 따른 객체 사이의 상호작용을 표현한다.
		상호작용 개요 다이어그램 (interaction overview diagram)	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
		통신 다이어그램 (communication diagram)	객체 사이의 관계를 중심으로 상호작용을 표현한다.
		타이밍 다이어그램 (timing diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

클래스

- **UML의 클래스 표현**
  - 세 부분으로 나누어진 박스로 표현
  - 가장 윗부분: 클래스 이름 중간 부분: 속성 마지막 부분: 연산

- **여러 가지 클래스 표현 예**





클래스

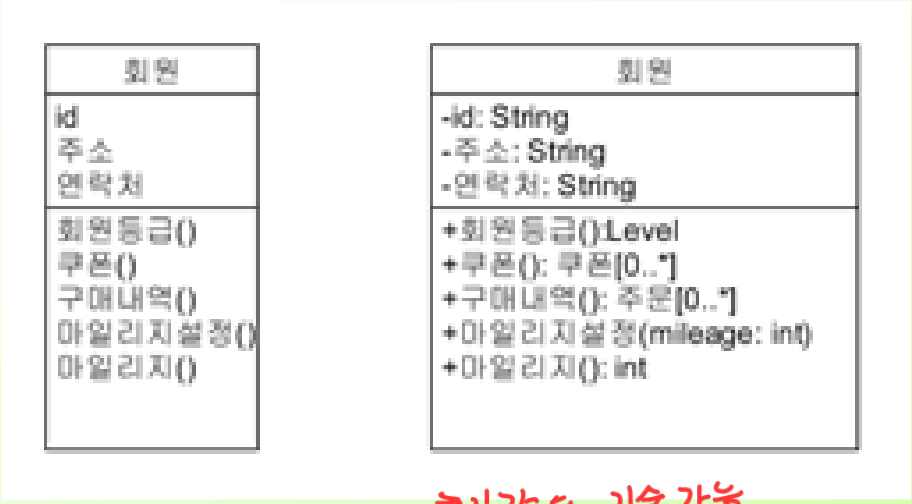
- **접근제어자**

접근제어자	표시	설명
public	+	어떤 클래스의 객체에서도 접근 가능
protected	#	이 클래스와 상속 관계에 있는 하위클래스의 객체들만 접근 가능
package	~	동일 패키지에 있는 클래스의 객체들만 접근 가능
private	-	이 클래스로부터 생성되어진 객체들만 접근 가능

속성과 오퍼레이션 표기

	표현법
속성	[+|-|#|~]이름:타입[다중성 정보] [=초기값]
오퍼레이션	[+|-|#|~]이름(인자1: 타입1, ..., 인자n:타입n):반환 타입

분석 단계의 클래스와 설계 단계의 클래스



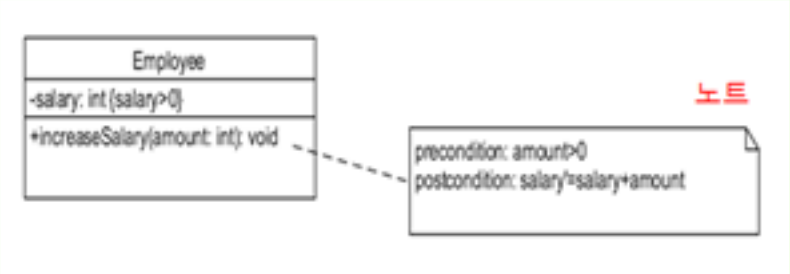
정적 연산과 속성 표기


회원
-id: String
-주소: String
-연락처: String
+회원등급():Level
+쿠폰(): 쿠폰[0..*]
+구매내역(): 주문[0..*]
+마일리지설정(mileage: int)
+마일리지(): int
+평균회원구매액():Money

UMLet

- UML 모델링 도구
  - 오픈 소스
  - GPL 라이선스
- www.umlet.com

제약


Employee
-salary: int {salary>0}
+increaseSalary(amount: int): void
노트
precondition: amount>0
postcondition: salary'=salary+amount

또는 {} 이용

관계

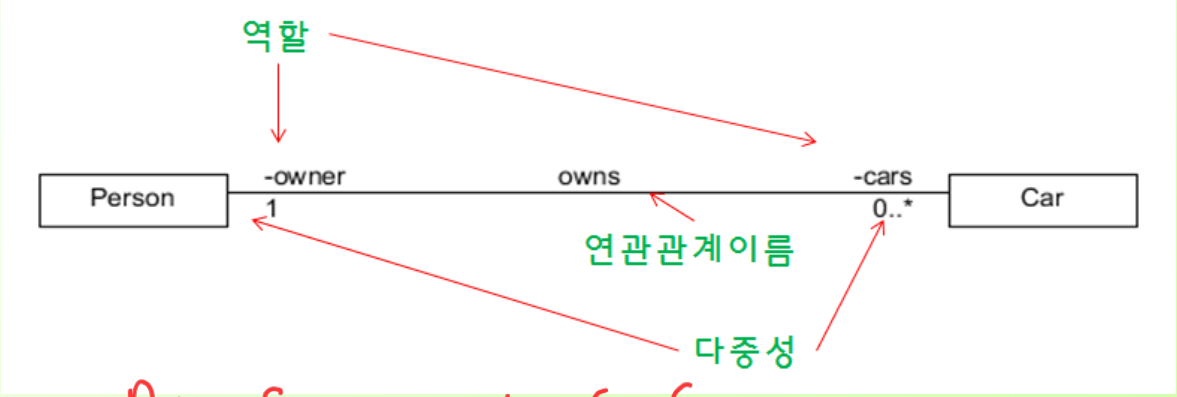
- **객체지향 시스템은 상호관계를 맺는 여러 클래스에서 생성된 객체들이 기능을 수행한다.**

관계

관계	설명
연관 관계 (association)	클래스들이 개념상 서로 연결되었음을 나타낸다. 실선이나 화살표로 표시하며 보통은 한 클래스가 다른 클래스에서 제공하는 기능을 사용하는 상황일 때 표시한다.
일반화 관계 (generalization)	객체지향 개념에서는 상속 관계라고 한다. 한 클래스가 다른 클래스를 포함하는 상위 개념일 때 이를 IS-A 관계라고 하며 UML에서는 일반화 관계로 모델링한다. 속이 빈 화살표를 사용해 표시한다.
집합 관계 (composition, aggregation)	클래스들 사이의 전체 또는 부분 같은 관계를 나타낸다. 집약aggregation 관계와 합성composition 관계가 존재한다.
의존 관계 (dependency)	연관 관계와 같이 한 클래스가 다른 클래스에서 제공하는 기능을 사용할 때를 나타낸다. 차이점은 두 클래스의 관계가 한 메서드를 실행하는 동안과 같은, 매우 짧은 시간만 유지된다는 점이다. 점선 화살표를 사용해 표시한다.
실체화 관계 (realization)	책임들의 집합인 인터페이스와 이 책임들을 실제로 실현한 클래스들 사이의 관계를 나타낸다. 상속과 유사하게 빈 삼각형을 사용하며 머리에 있는 실선 대신 점선을 사용해 표시한다.

연관 관계

- 연관된 클래스 상에 실선을 그어 표시
- 두 클래스 상이의 관계가 명확한 경우에 이름을 사용하지 않아도 됨



Java

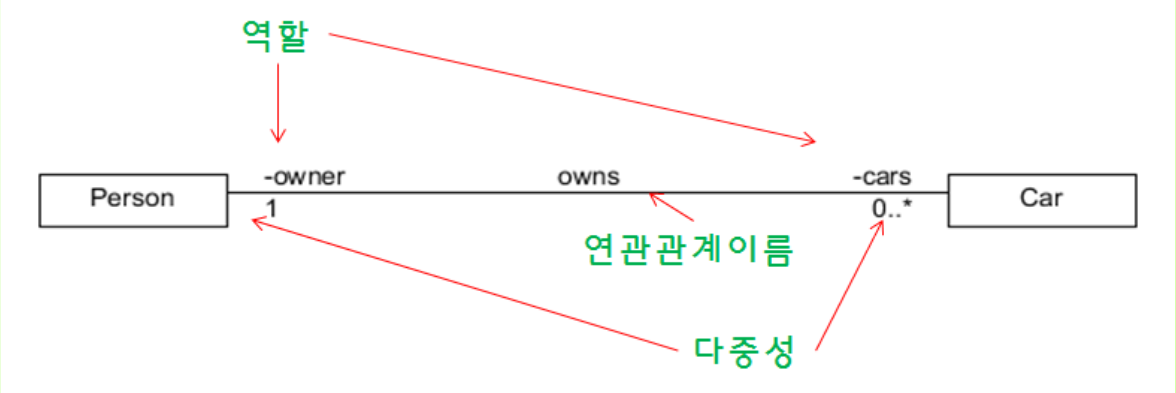
```
class Person {
    Car[] cars;
}
```

```
class Car {
    Person owner;
}
```

양방향 관계
→ 서로가 서로를 안다.

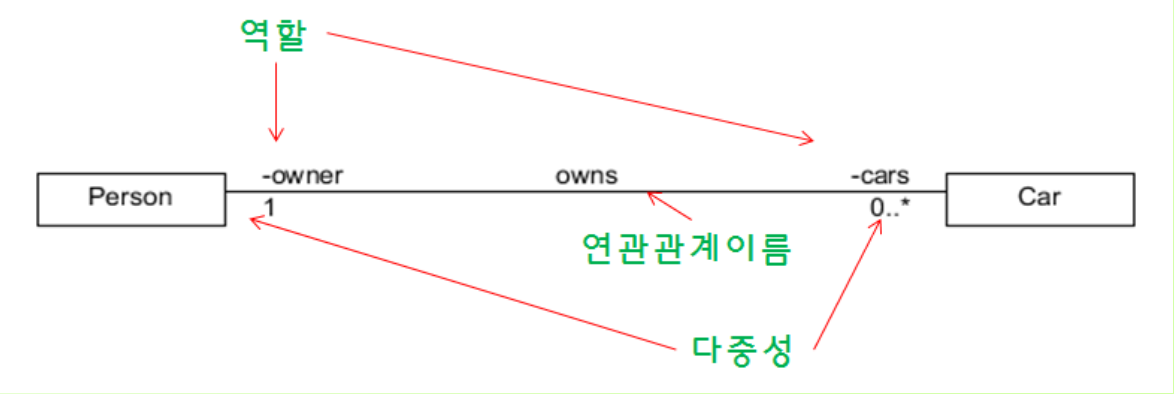
연관 관계에서의 역할

- 연관 관계에서 각 클래스 객체의 역할은 클래스 바로 옆 연관 관계를 나타내는 선 가까이 기술
- 역할 이름은 연관된 클래스의 객체들이 서로를 참조할 수 있는 속성의 이름으로 활용

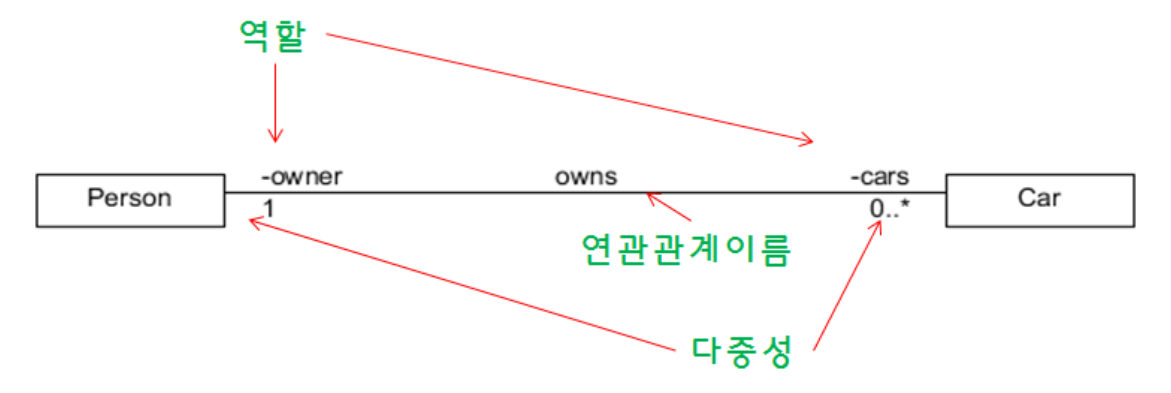


다중성

다중성 표기	의미
1	엄밀하게 1
*	0 또는 그 이상
0..*	0 또는 그 이상
1..*	1 이상
0..1	0 또는 1
2..5	2 또는 3 또는 4 또는 5
1,2,6	1 또는 2 또는 6
1, 3..5	1 또는 3 또는 4 또는 5



다중성과 코드

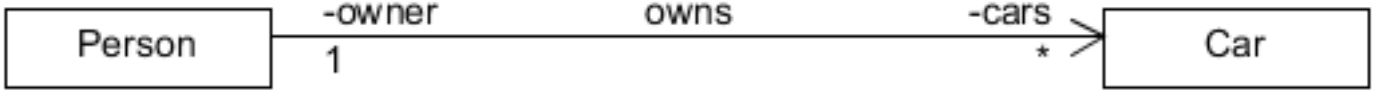



```
class Person {
      private Car[] cars;
}
```

양방향, 단방향 연관 관계

- **양방향 연관 관계**
  - 두 클래스를 연결한 선에 화살표를 사용하지 않음
  - 서로의 존재를 인지
- **단방향 연관 관계**
  - 한쪽으로만 방향성이 있는 관계
  - 한 쪽은 알지만 다른 쪽은 상대방의 존재를 모른다는 의미

사람은 자동차를 안다 But 자동차는 사람을 모른다.




```
class Person {
  private Car[] cars
}
```

```
class Car {
}
```

Car 클래스에는 Person 클래스를 참조하는 속성이 없다.

체크포인트

체크포인트_ 다음 설명에 맞는 클래스 다이어그램을 작성하라.

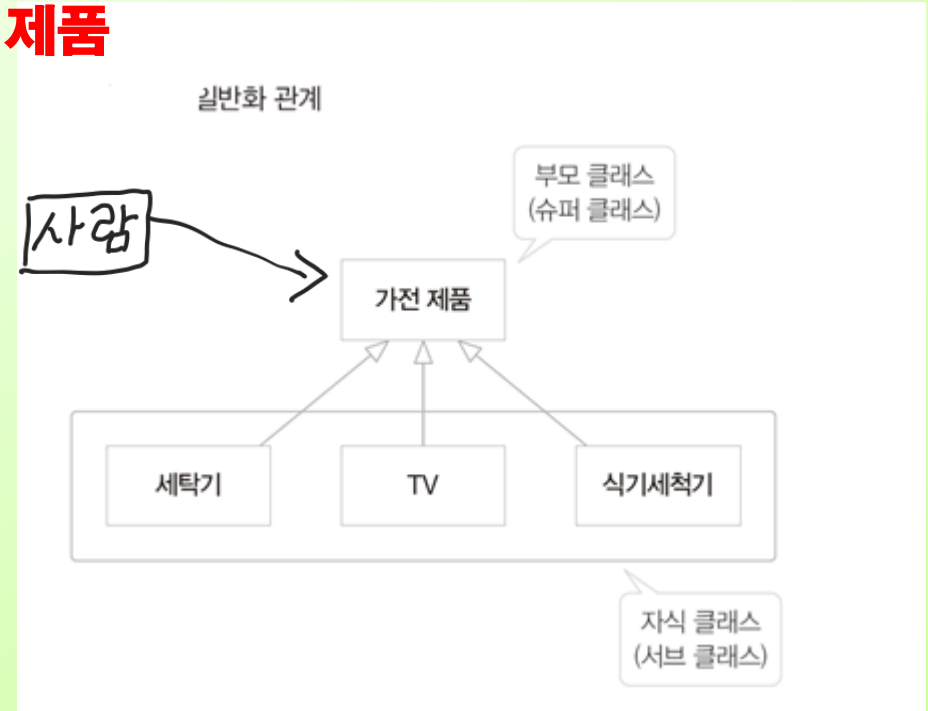
- 학생은 반드시 한 학교에 소속되어야 한다.
- 학교는 학생이 반드시 100명 이상 있어야 한다.

일반화

- 일반화는 "is a kind of" 관계
  - 세탁기 is a kind of 가전 제품
  - TV is a kind of 가전 제품
  - 식기세척기 is a kind of 가전 제품

abstract class 가전제품{
}
class 식기세척기 extends 가전제품{...}
" 세탁기 extends 가전제품{...}
⋮

```
class 사람 {
  가전제품 m;
  public void setM(가전제품 m){
    this.m = m;
  }
  public 가전제품 getM(){
    return this.m;
  }
}
```

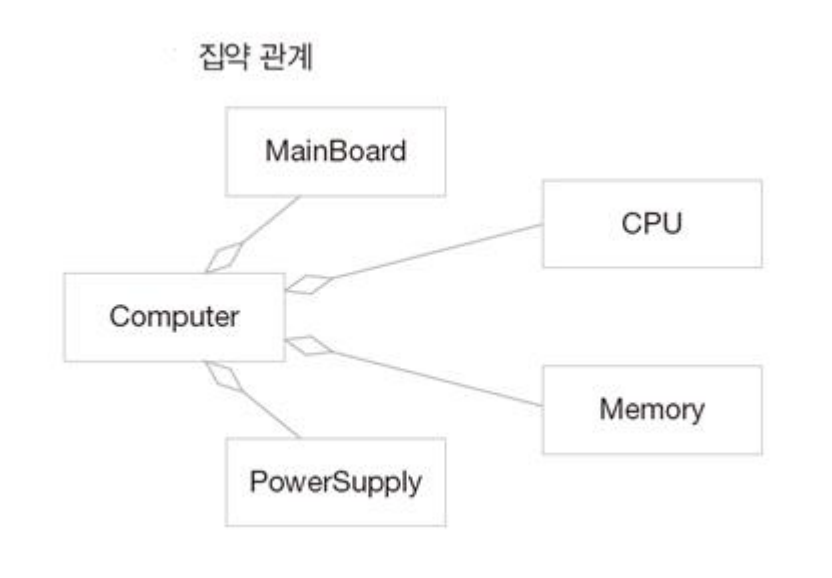


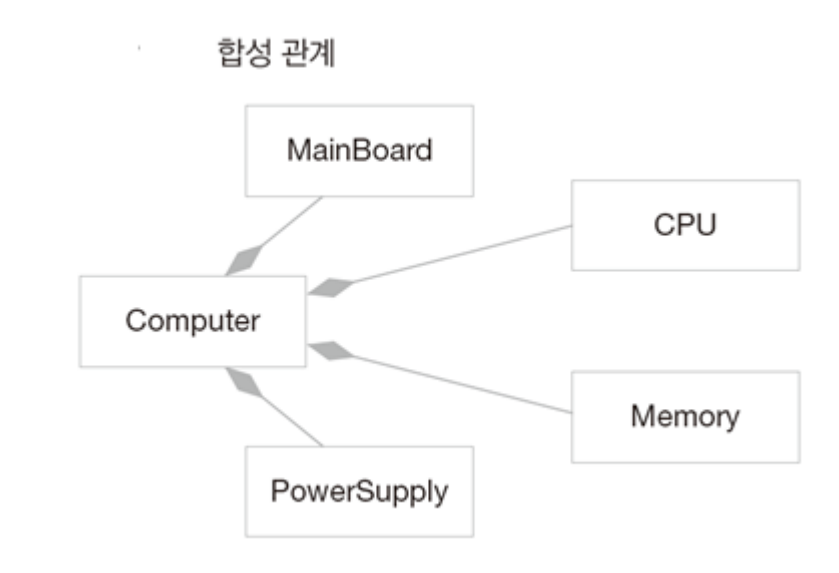
집합 관계

- 전체와 부분간의 관계
- 집약(aggregation):
  - 전체를 나타내는 객체와 부분을 나타내는 개체의 라이프 타임이 독립적
  - 부분을나타내는 객체를 다른 객체와 공유 가능
  - 빈 마름보로 표시
- 합성(composition)
  - 전체를 나타내는 객체에 부분을 나타내는 개체의 라이프 타임이 종속적
  - 전체 객체가 사라지면 부분 객체도 사라짐
  - 채워진 마름보로 표시

합쳐져서 하나를 만듦

집약/합성 관계


집약 관계
MainBoard
CPU
Computer
Memory
PowerSupply

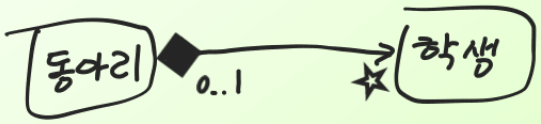

합성 관계
MainBoard
CPU
Computer
Memory
PowerSupply

실습

체크포인트_ 동아리와 학생의 관계에서 다음 사실을 모두 클래스 다이어그램으로 표현하라.

- 학생은 한 동아리에만 가입할 수 있다.
- 한 동아리에는 여러 명의 학생들이 있다.
- 동아리가 없어지면 동아리에서 활동했던 학생들의 정보도 없어진다.

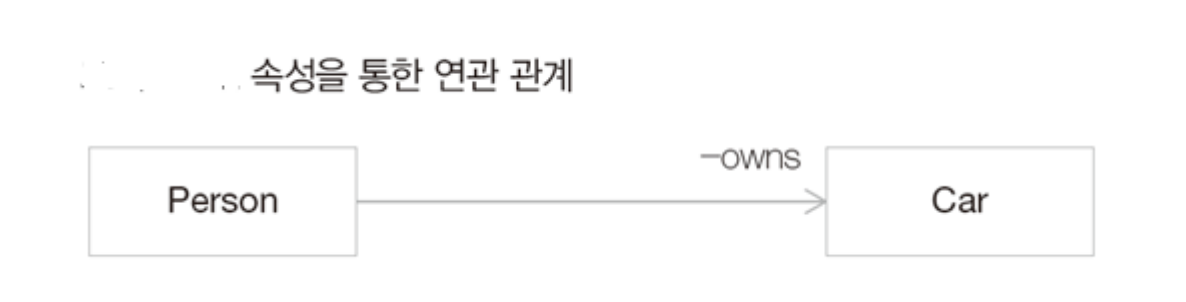
↳ composition (합성)



의존 관계

- 한 클래스에서 다른 클래스를 사용하는 경우
  - 클래스의 속성에서 참조
  - 연산의 인자로 참조
  - 메소드의 지역 개체로 참조

지속적 관계



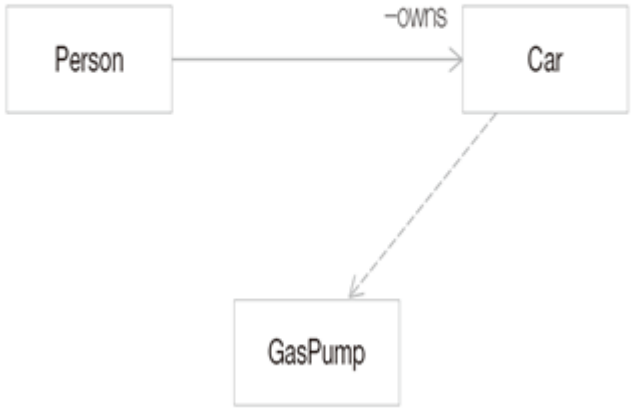
의존 관계

- **연산의 인자나 메소드의 지역 개체로 참조**
  - **찰나적 관계**

의존 관계

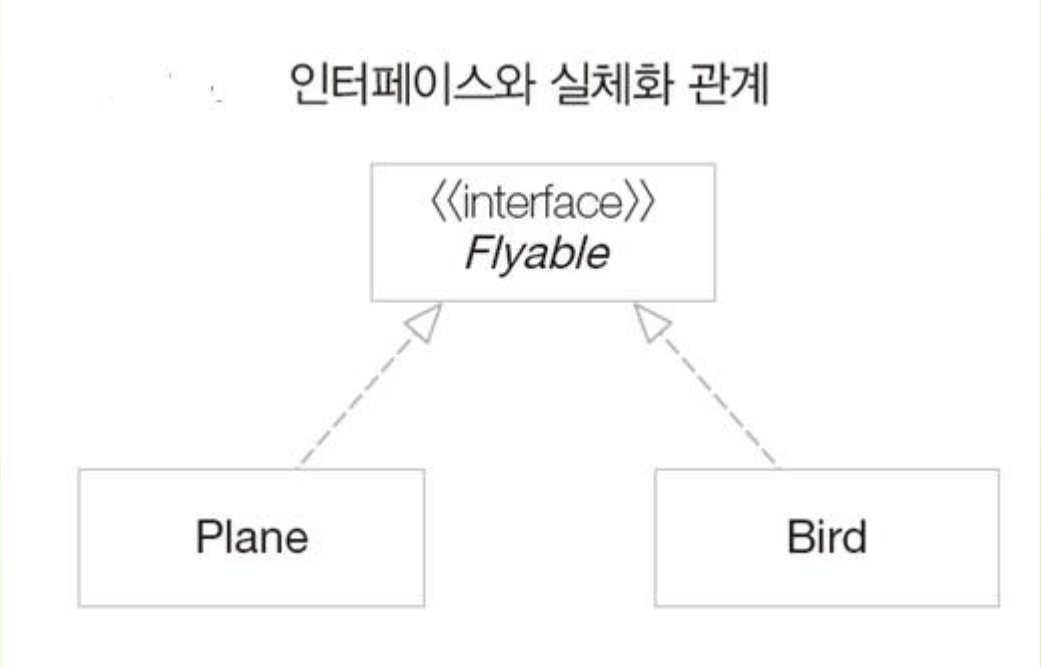
```
public class Car {
    ...

    public void fillGas(GasPump p) {
        p.getGas(amount);
        ...
    }
}
```

의존 관계와 연관 관계



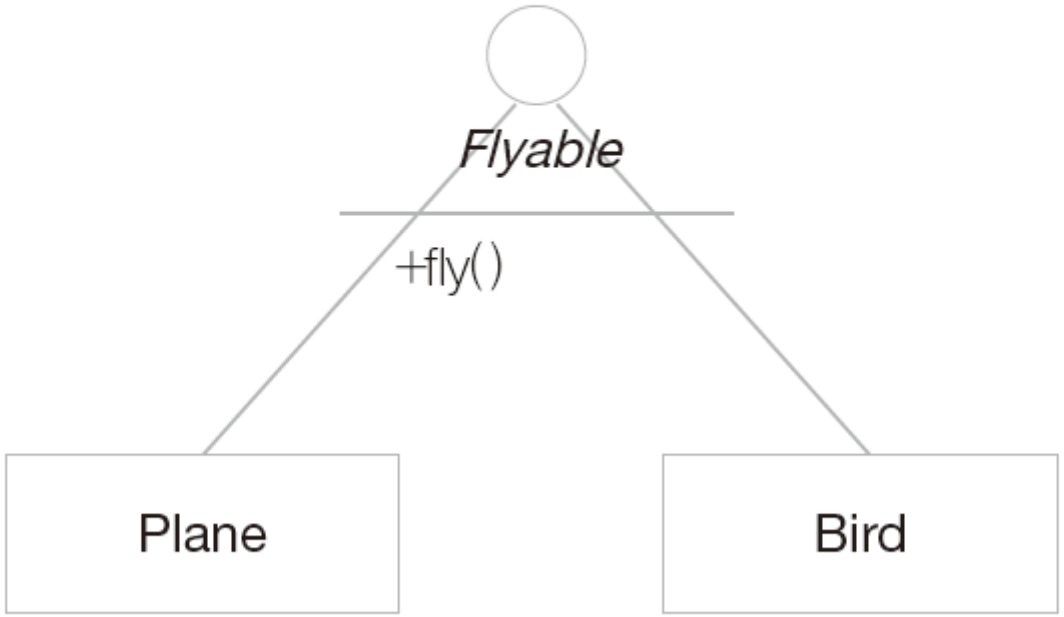
인터페이스와 실체화 관계

- 인터페이스란 책임이다.
  - 리모콘의 책임은 가전 기기를 켜거나 끄거나 볼륨을 높이거나 낮춘다
- 어떤 공통되는 능력이 있는 것들을 대표하는 관점



실체화 관계

그림 1-26 실체화 관계의 또다른 표현



Keypoint

- 일반화 관계는 is a kind of 관계이지만 실체화 관계는 can do this의 관계이다

객체 다이어그램

- **클래스 다이어그램의 인스턴스 즉 사례를 보여준다**
  - 학생은 반드시 한 학교에 소속되어야 한다.
  - 학생은 1개의 학교 밖에 소속할 수 없을 것
  - 학교는 학생이 없거나 여러 명 있을 수 있다.

표 1-1 UML 다이어그램의 종류

분류	다이어그램 유형		목적
구조 다이어그램 (structure diagram)	클래스 다이어그램 (class diagram)		시스템을 구성하는 클래스들 사이의 관계를 표현한다.
	객체 다이어그램 (object diagram)		객체 정보를 보여준다.
	복합체 구조 다이어그램 (composite structure diagram)		복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	배치 다이어그램 (deployment diagram)		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	컴포넌트 다이어그램 (component diagram)		컴포넌트 구조 사이의 관계를 표현한다.
	패키지 다이어그램 (package diagram)		클래스나 유즈 케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램 (behavior diagram)	활동 다이어그램 (activity diagram)		업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	상태 머신 다이어그램 (state machine diagram)		객체의 생명주기를 표현한다.
	유즈 케이스 다이어그램 (use case diagram)		사용자 관점에서 시스템 행위를 표현한다.
	상호작용 다이어그램 (interaction diagram)	순차 다이어그램 (sequence diagram)	시간 흐름에 따른 객체 사이의 상호작용을 표현한다.
		상호작용 개요 다이어그램 (interaction overview diagram)	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
		통신 다이어그램 (communication diagram)	객체 사이의 관계를 중심으로 상호작용을 표현한다.
		타이밍 다이어그램 (timing diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

객체 다이어그램

- **클래스 다이어그램의 인스턴스 즉 사례를 보여준다**
  - 학생은 반드시 한 학교에 소속되어야 한다.
  - 학생은 1개의 학교 밖에 소속할 수 없을 것
  - 학교는 학생이 없거나 여러 명 있을 수 있다.

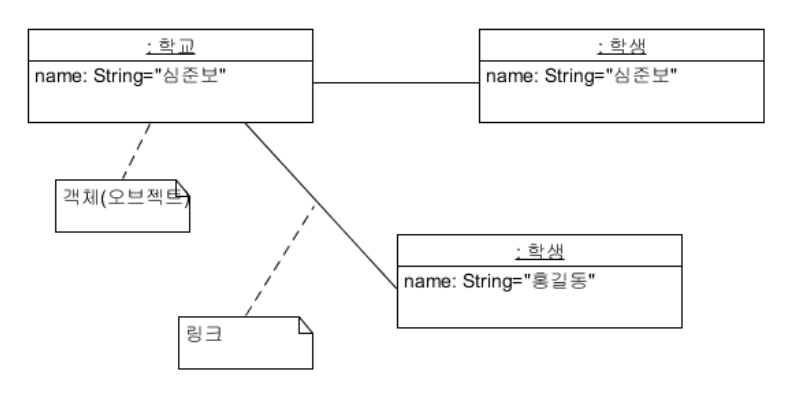
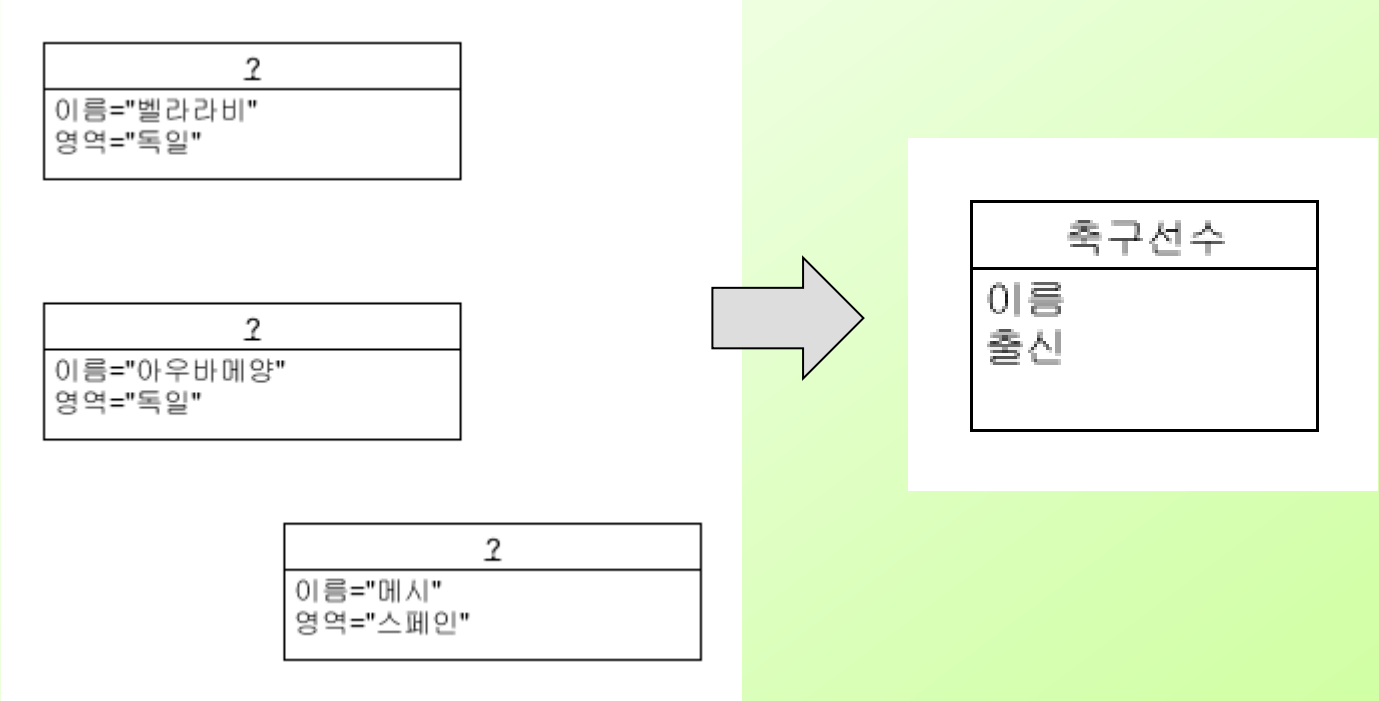


표 1-1 UML 다이어그램의 종류

분류	다이어그램 유형		목적
구조 다이어그램 (structure diagram)	클래스 다이어그램 (class diagram)		시스템을 구성하는 클래스들 사이의 관계를 표현한다.
	객체 다이어그램 (object diagram)		객체 정보를 보여준다.
	복합체 구조 다이어그램 (composite structure diagram)		복합 구조의 클래스와 컴포넌트 내부 구조를 표현한다.
	배치 다이어그램 (deployment diagram)		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조를 표현한다.
	컴포넌트 다이어그램 (component diagram)		컴포넌트 구조 사이의 관계를 표현한다.
	패키지 다이어그램 (package diagram)		클래스나 유즈 케이스 등을 포함한 여러 모델 요소들을 그룹화해 패키지를 구성하고 패키지들 사이의 관계를 표현한다.
행위 다이어그램 (behavior diagram)	활동 다이어그램 (activity diagram)		업무 처리 과정이나 연산이 수행되는 과정을 표현한다.
	상태 머신 다이어그램 (state machine diagram)		객체의 생명주기를 표현한다.
	유즈 케이스 다이어그램 (use case diagram)		사용자 관점에서 시스템 행위를 표현한다.
	상호작용 다이어그램 (interaction diagram)	순차 다이어그램 (sequence diagram)	시간 흐름에 따른 객체 사이의 상호작용을 표현한다.
		상호작용 개요 다이어그램 (interaction overview diagram)	여러 상호작용 다이어그램 사이의 제어 흐름을 표현한다.
		통신 다이어그램 (communication diagram)	객체 사이의 관계를 중심으로 상호작용을 표현한다.
		타이밍 다이어그램 (timing diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

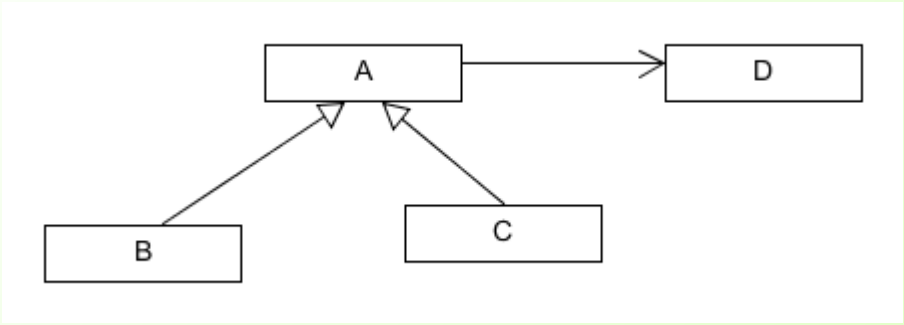
객체다이어그램의 쓰임새

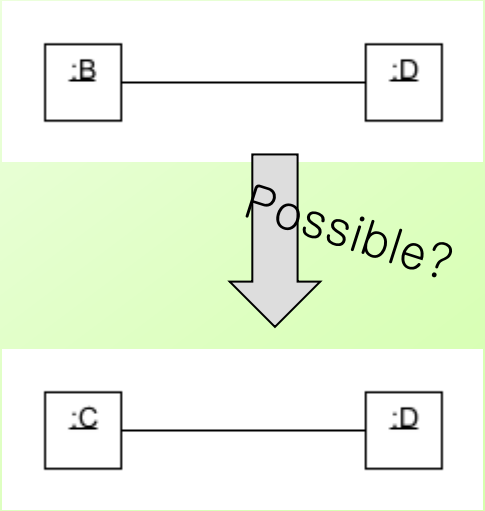
- **클래스 식별**



객체다이어그램의 쓰임새

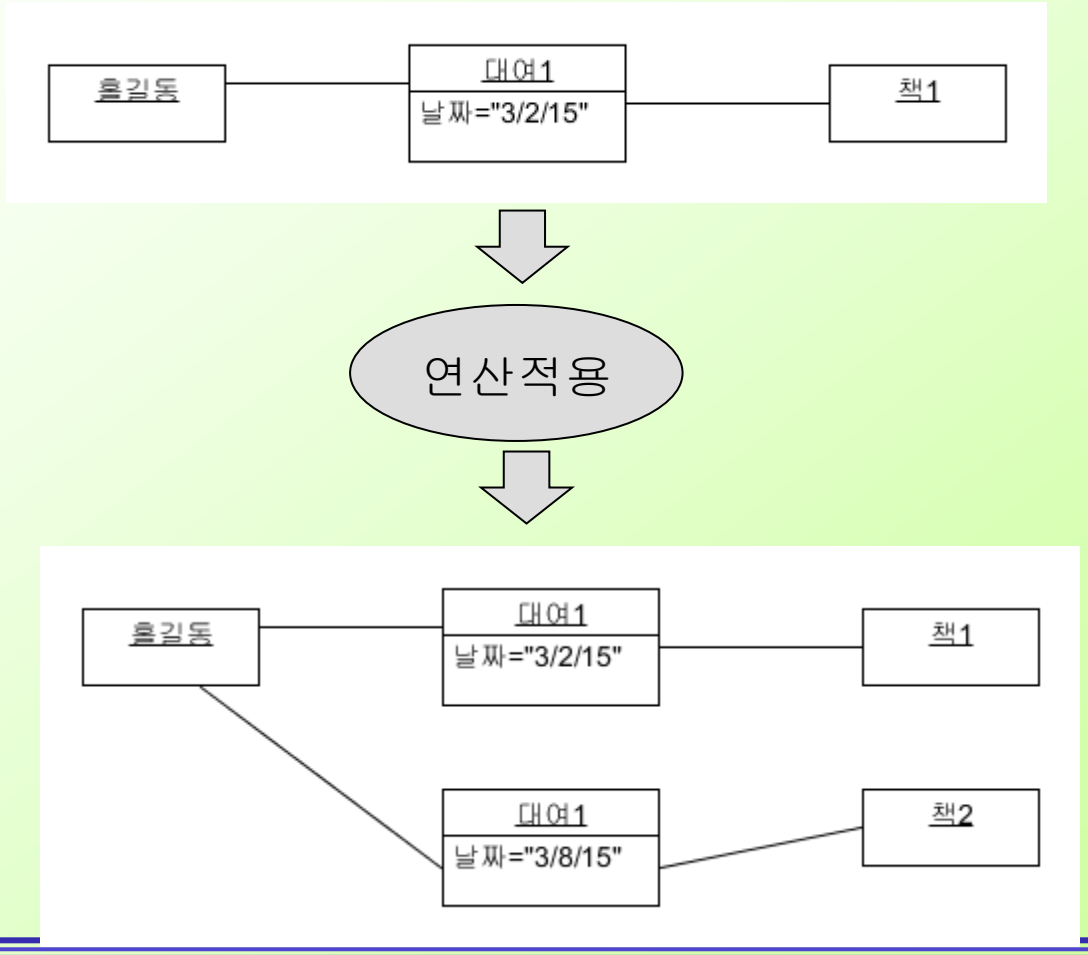
- 클래스 다이어그램의 확인





객체다이어그램의 쓰임새

- 연산의 작용



실습

- **다음 클래스 다이어그램의 객체 다이어그램을 작성하라**