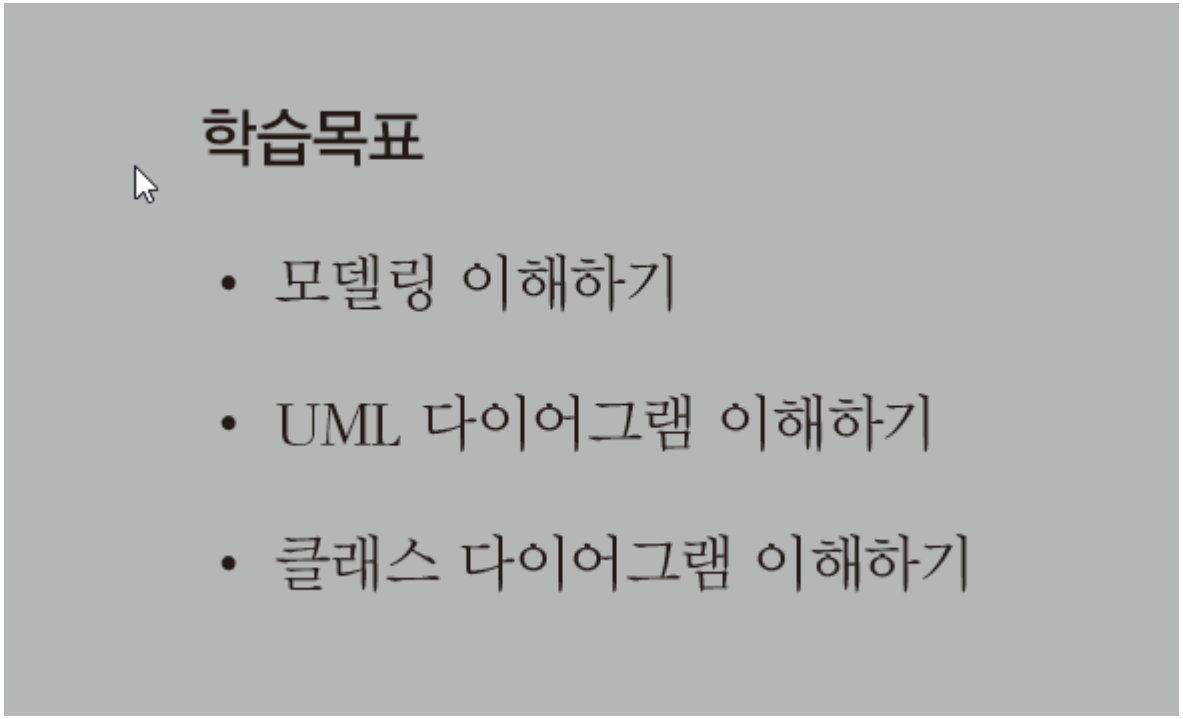
1. 객체지향 모델링

학습목표



1.1 모델링

- 모델의 역할
  - 서로의 해석을 공유해 합의를 이루거나 해석의 타당성을 검토
  - 현재 시스템 또는 앞으로 개발할 시스템의 원하는 모습을 가시화
  - 시스템의 구조와 행위를 명세
  - 시스템을 구축하는 틀 제공

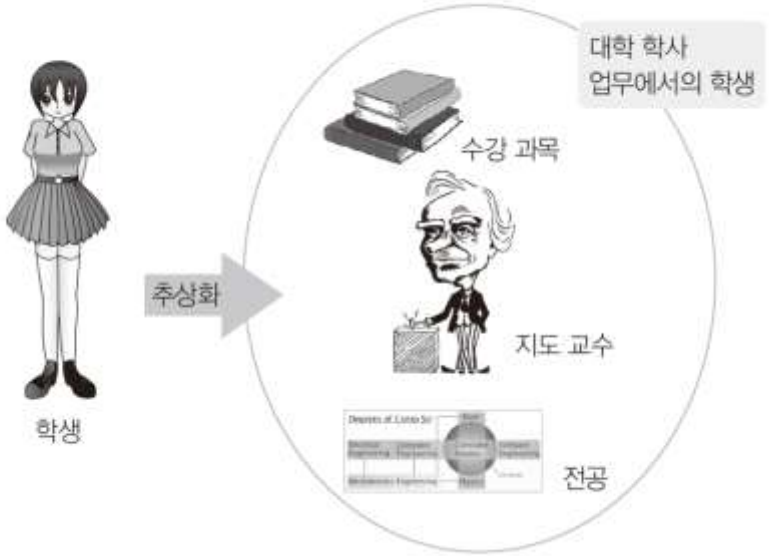
그림 1-1 자동차의 전체적 구조를 파악할 수 있는 자동차 모델

모델링

❖ 모델은 추상화에 바탕을 두고 있다

- 특정 관점에서 관련이 있는 점은 부각 관련이 없는 것은 무시

그림 1-2 대학 학사 업무의 추상화



1.2 UML

❖ 대표적인 시스템 모델링 언어

- 제임스 럼버 OMT(Object Modeling Technique) + 이바 야콥슨 OOSE(Object-Oriented Software Engineering) + 그래디 부치 OOAD(Object Oriented Analysis and Design)
- 2012년 UML 2.5(가장 최신 버전)

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.

1.3 클래스 다이어그램

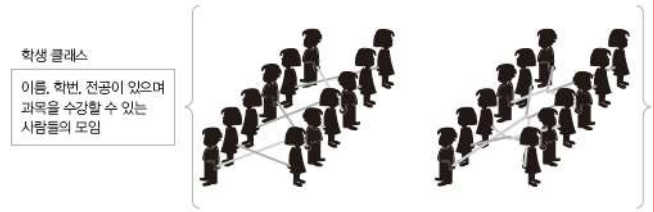
- 클래스 다이어그램
  - 시스템의 정적인 구조 표현
  - 시간에 따라 변하지 않는다
  - 클래스와 그들간의 관계 표현

1.3.1 클래스

❖ 클래스

- 동일한 속성을 가지고 있고 동일한 행위를 수행하는 객체의 집합
- 객체를 생성하는 설계도

그림 1-3 클래스와 객체



설계도

코드 1-1

```
public class Cat {
  private String name;

  public void meow() {
    System.out.println(name + "~~~~~~" + "웁니다");
  }

  public Cat(String name) {
    this.name = name;
  }
}
```

그림 1-4 Cat 클래스에서 생성된 고양이 객체

Cat cat2 = new Cat("냥냥이");

Cat cat1 = new Cat("야옹이");

클래스

❖ UML의 클래스 표현

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

그림 1-5 UML 클래스의 표현 예

❖ 여러 가지 클래스 표현 예

- 경우에 따라 속성이나 연산 부분 생략 가능

그림 1-6 여러 가지 클래스의 표현 예

클래스

❖ 접근제어자

표 1-2 접근 제어자

접근 제어자	표시	설명
public	+	어떤 클래스의 객체에서든 접근 가능
private	–	이 클래스에서 생성된 객체들만 접근 가능
protected	#	이 클래스와 동일 패키지에 있거나 상속 관계에 있는 하위 클래스의 객체들만 접근 가능
package	~	동일 패키지에 있는 클래스의 객체들만 접근 가능

1.3.2 관계

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

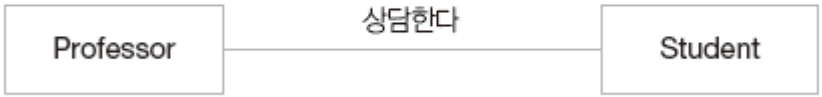
표 1-4 관계

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

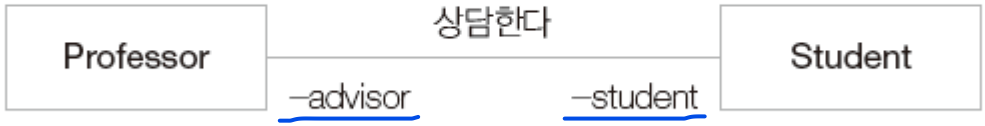
그림 1-8 Professor 클래스와 Student 클래스의 연관 관계



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

그림 1-9 연관 관계에서의 역할

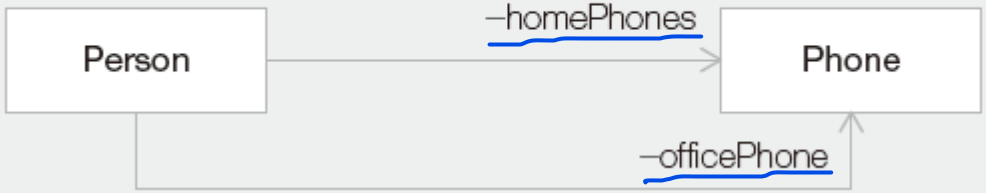


체크포인트 ← = 퀴즈

체크포인트_ 다음 클래스 다이어그램을 코드로 작성하라.

코드는 메모장에

체크포인트_ 다음 클래스 다이어그램을 코드로 작성하라.

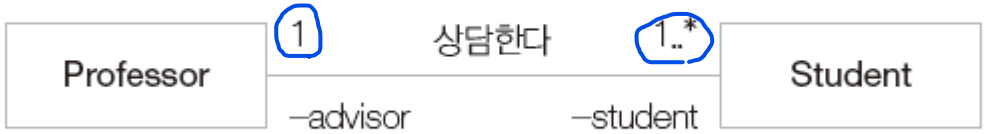


다중성

표 1-5 다중성 표시

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

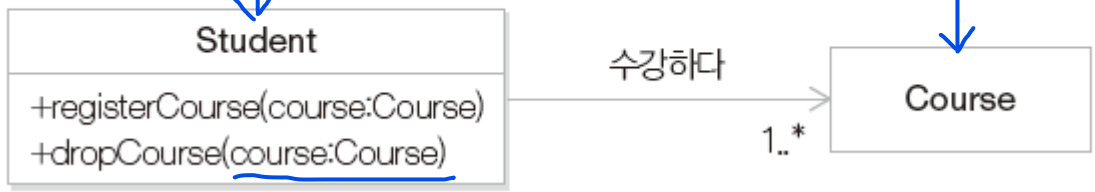
그림 1-10 다중성 예



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

그림 1-11 단방향 연관 관계



체크포인트

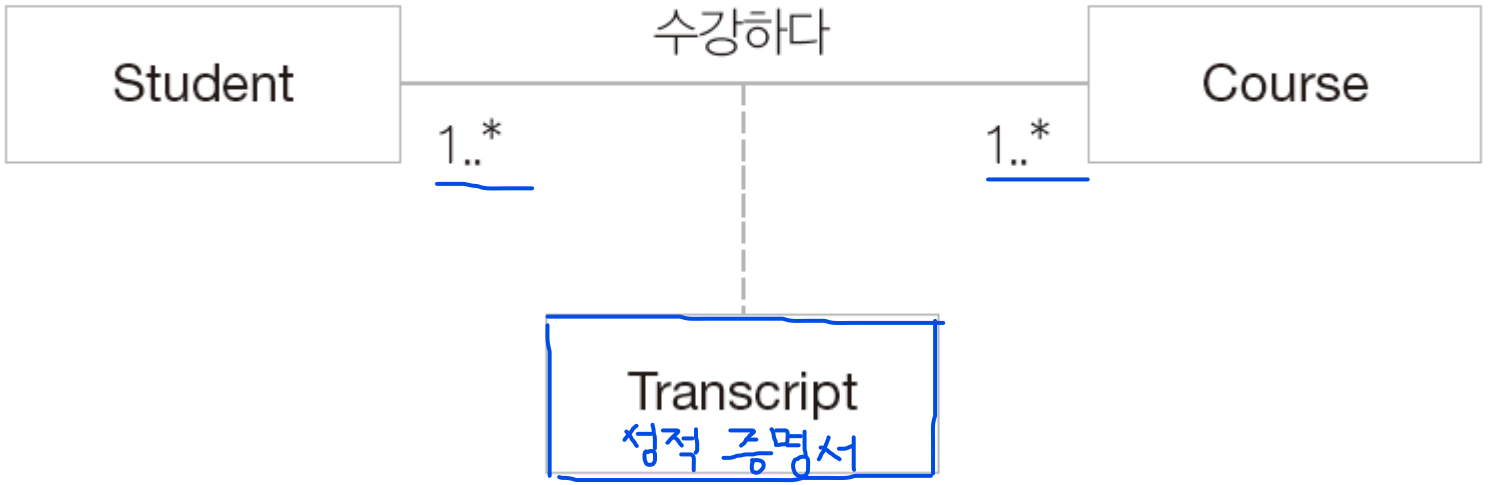
체크포인트_ 다음 설명에 맞는 클래스 다이어그램을 작성하라.

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

연관 클래스

- 연관 관계에 추가할 속성이나 행위가 있을 때 사용
  다대다

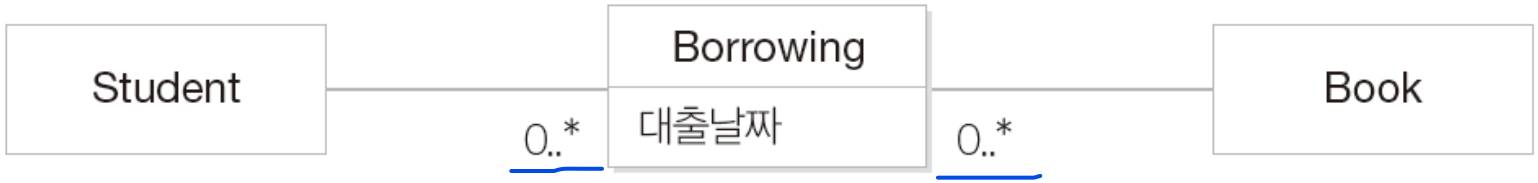
그림 1-13 연관 클래스



연관 클래스

- 사건 이력(event history) 표현
- 대출 이력

그림 1-15 이력의 표현



재귀적 연관 관계

- 동일한 클래스에 속한 객체들 사이의 연관 관계
- 역할을 클래스로 할 때 문제가 발생

그림 1-16 직원 역할을 클래스로 모델링

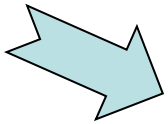
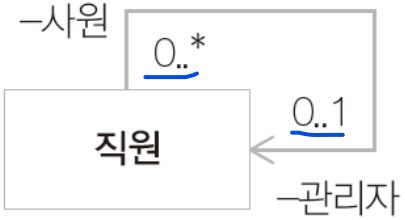
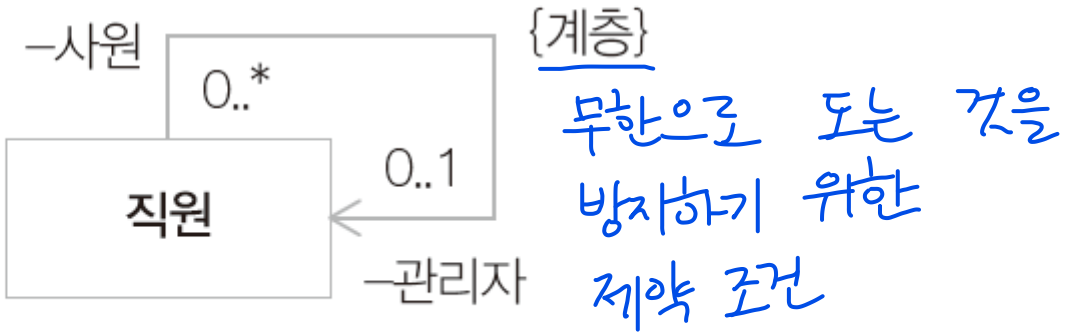


그림 1-17 재귀적 연관 관계



제약 설정

그림 1-18 재귀적 연관 관계에서 제약 설정

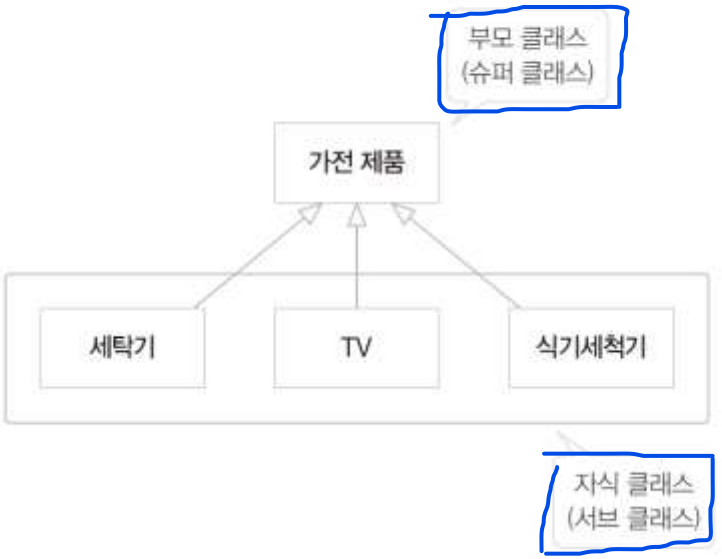


무한으로 도는 것을 방지하기 위한 제약 조건

일반화

- 일반화는 상속화?
- 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

그림 1-19 일반화 관계



집합 관계

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

집약/합성 관계

그림 1-20 합성 관계

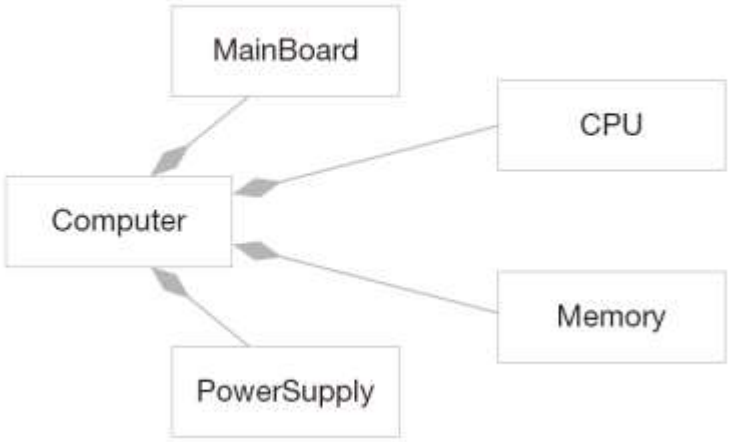
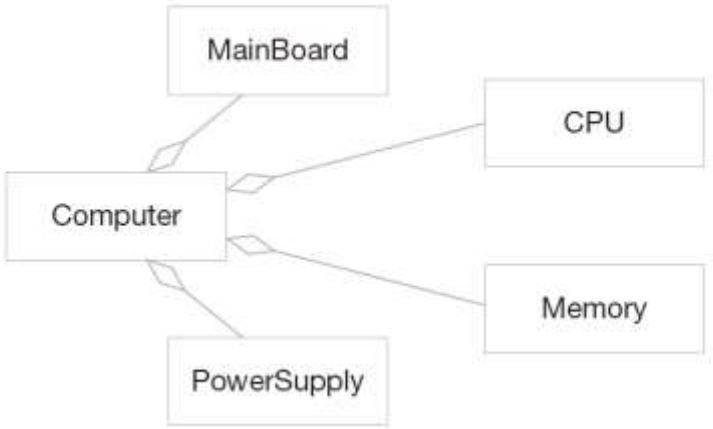


그림 1-21 집약 관계




```
public class Computer {
	private MainBoard mb;
	private CPU c;
	private Memory m;
	private PowerSupply ps;
	public Computer() {
		this.mb=new MainBoard();
		this.c=new CPU();
		this.m=new Memory();
		this.ps=new PowerSupply();
	}
}
```

```
public class Computer {
	private MainBoard mb;
	private CPU c;
	private Memory m;
	private PowerSupply ps;
	public Computer(MainBoard mb, CPU c, Memory m, PowerSupply ps) {
		this.mb=mb;
		this.c=c;
		this.m=m;
		this.ps=ps;
	}
}
```

체크포인트

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.

의존 관계

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

지속적 관계

그림 1-22 속성을 통한 연관 관계

의존 관계

- 연산의 인자나 메소드의 지역 개체로 참조
  - 찰나적 관계 ≒ 주황선

→ 지속

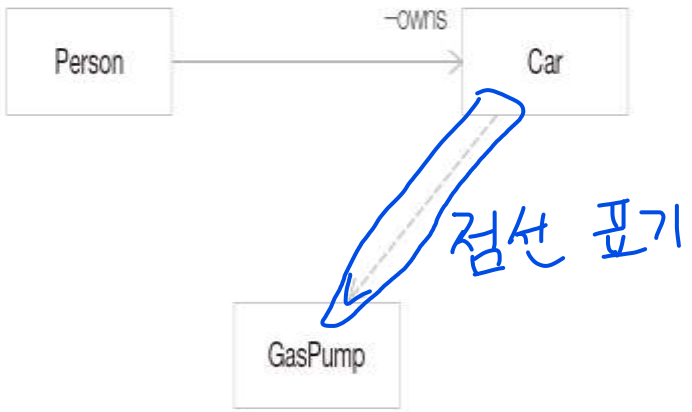
- - - -> 찰나

그림 1-23 의존 관계

```
public class Car {
    ...

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

그림 1-24 의존 관계와 연관 관계

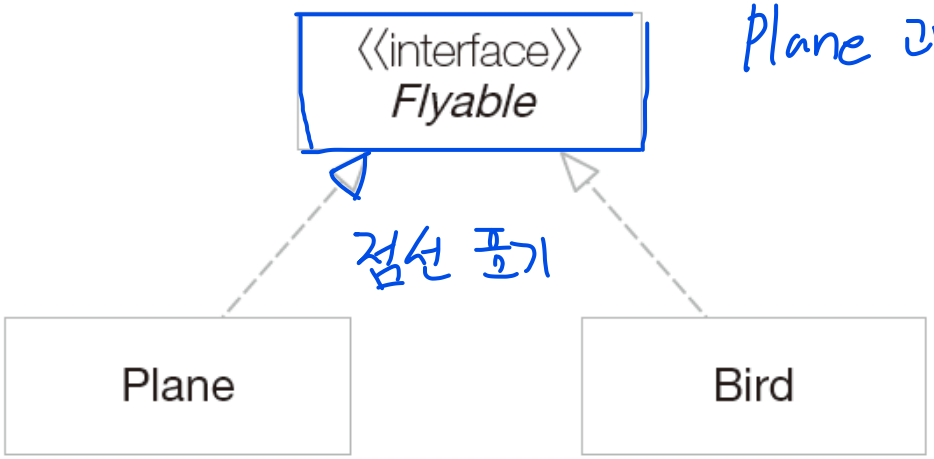


인터페이스와 실체화 관계

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

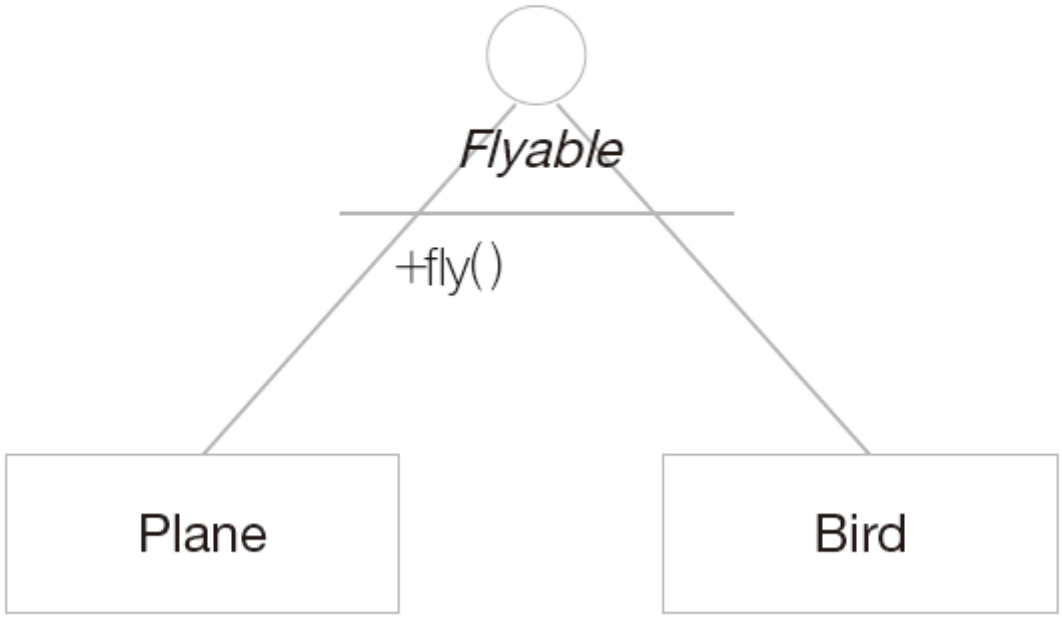
그림 1-25 인터페이스와 실체화 관계

인터페이스의 구현은 Plane 과 Bird가 할 것



실체화 관계

그림 1-26 실체화 관계의 또다른 표현



Keypoint

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