2. 클래스 다이어그램

먼저, 클래스 **계좌**에 대한 **클래스 다이어그램**을 만들어 설명한다.

〈클래스 계좌의 클래스 다이어그램〉

계좌	→ 클래스명
- 계좌번호 : String - 비밀번호 : int # 잔고 : int	→ 속성
+ 입금하다(금액 : int) : void + 출금하다(금액 : int) : void + 조회하다() : int	→ 메서드(연산)

- **클래스 다이어그램**은 3부분으로 구성된다.
- **3부분** : 클래스명, 속성, 메서드
- 속성과 메서드는 생략될 수 있다.

〈star_UML 사용〉

↓

Java 원시코드 생성 (UML에서 설계한 것을 Java 원시코드로 변환할 수 있다)

↓

〈Java 원시코드〉

```
public class 계좌                                    //클래스 계좌
{
  private String 계좌번호;                           //클래스 계좌의 속성
  private int 비밀번호;
  protected int 잔고;
  public void 입금하다(int 금액) { 잔고 = 잔고 + 금액; }  //클래스 계좌의 메서드
  public void 출금하다(int 금액) { 잔고 = 잔고 - 금액; }
  public int 조회하다() { return 잔고; }
}
```

◈ UML의 클래스 다이어그램에서 접근지정자 기호

접근지정자	기호	설 명
public	+	모든 클래스의 객체들이 접근 가능(**공용**)
protected	#	모든 자식클래스에서 접근 가능(**보호**)
private	-	선언된 클래스 내에서만 접근 가능(**전용, 외부 접근 불가**)
package	~	같은 **패키지**에 있는 클래스에서 접근 가능(자바의 default)

2 http://cafe.daum.net/pass365(홍재연)

◈ **클래스 사이의 관계 및 그리는 방법**

클래스 다이어그램은 시스템을 구성하는 **클래스**를 **정의**한다.
클래스 다이어그램의 주된 목적은 클래스 사이의 **정적인 관계**를 표현하는 것이다.
즉, 클래스 다이어그램은 시스템의 정적구조를 모델링 한다.

// 클래스 다이어그램에서 관계

관계	설 명
일반화 generalization	• 상속관계 (IS-A 관계) • 부모클래스와 자식클래스 사이의 관계
실체화 realization	• 인터페이스와 자식클래스 사이의 관계 • 인터페이스에서 메서드를 **선언**하고 • 자식클래스에서 메서드를 **구현**한다.
연관 association	• 연관은 **오랜 시간 동안** 서로 함께하는 클래스들의 관계이다. • **소유** 관계 (포함 관계)
의존 dependency	• 의존은 **짧은 시간 동안** 사용하는 관계이다. • **사용** 관계 (using)
집합 aggregation	• 집합은 **생명주기가 일치하지 않는** 특별한 연관 관계이다. • 예 : 회사와 부서의 관계 (회사에는 여러 부서가 존재한다)
합성 composition	• 합성은 **생명주기가 일치하는** 특별한 연관 관계이다. • 예 : 손(hand)은 손가락을 포함하고 있다.

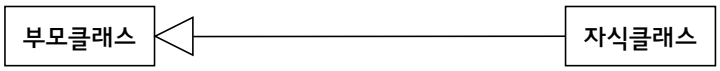
• 클래스 다이어그램은 클래스 사이의 정적인 관계를 한 눈에 쉽게 파악하는 것에 있다.

• 클래스 다이어그램은 시스템 분석 단계와 설계 단계에 걸쳐 여러 번 작성된다.

• 집합과 합성은 특수한 연관 관계이다.

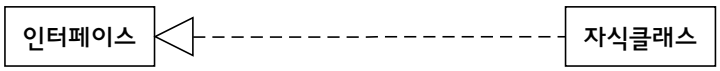
클래스 다이어그램에서 클래스 사이의 **관계**를 **그리는 방법**을 살펴본다.

〈일반화(generalization)〉



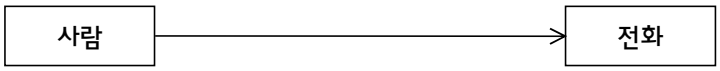
- UML에서 상속관계를 일반화 관계라고 한다.(is-a)
- 빈 삼각형을 부모클래스(일반 분류자) 쪽으로 연결한다.
- 자식클래스(특정 분류자)는 부모클래스(일반 분류자)에서 정의된 일부를 상속받는다.

〈실체화(realization)〉



- 실체화는 인터페이스와 실제로 구현하는 클래스들 사이의 관계이다.
- 실체화는 **인터페이스**에서 메서드를 **선언**하고
- 자식클래스에서 상속받아서 실제 기능을 **구현**한다.
- 인터페이스와 클래스 사이의 관계는 점선과 인터페이스 쪽의 빈 삼각형으로 연결한다.

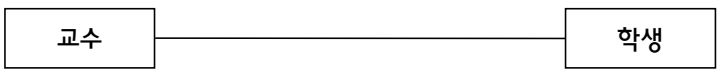
〈단방향 연관〉



〈직접연관(directed association)〉

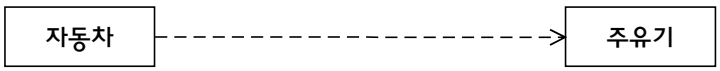
- 연관 관계는 방향성을 가질 수 있다. 단방향 연관 관계는 화살표로 표시한다.
- 단방향 연관은 **방향성이 있는 연관** 관계이다. 직접연관(directed association)이라 한다.
- 단방향 연관 : 상대방의 존재를 한 쪽은 알지만 다른 쪽은 모른다.**(사람이 전화를 참조)**

〈양방향 연관〉



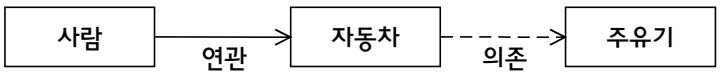
- 양방향 연관은 실선으로 표시한다.**(화살표를 표시하지 않는다)**
- 양방향 연관은 화살표를 표시하지 않으므로 **방향성이 없는 연관**이라고도 한다.
- 양방향 연관 : 두 클래스가 서로의 존재를 인식한다.**(교수와 학생이 서로 참조 가능)**
- 일반적으로, 다대다 연관 관계는 양방향 연관 관계로 표현하는 것이 적절하다.
- 하지만, 양방향 연관은 복잡하여 다대다 연관을 일대다 단방향 연관으로 변환하여 구현한다.

〈방향성 있는 의존〉



- 의존(dependency)은 어떤 클래스가 다른 클래스를 참조하는 것을 말한다.
- 의존은 짧은 시간 동안 **사용(use)**하는 관계이다.
- 의존은 UML에서 점선으로 나타낸다.
- 일반적으로, 의존은 클래스 다이어그램에서 가장 많이 사용되는 관계이다.

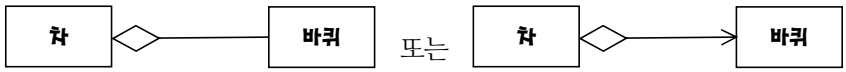
〈연관과 의존의 차이〉



연관	• 연관은 오랜 시간 동안 서로 함께하는 클래스들의 관계이다. • **연관**은 한 클래스가 다른 클래스를 **속성(멤버변수)으로 가지는 경우**이다.**(클래스를 포함)** • 연관은 **소유** 관계라고 한다.
의존	• 의존은 짧은 시간 동안 다른 클래스 내용을 사용하는 관계이다. • 클래스가 다른 클래스의 **지역변수, 매개변수 등을 일시적으로 사용**하는 관계이다. • 의존은 **사용** 관계라고 한다.(using)

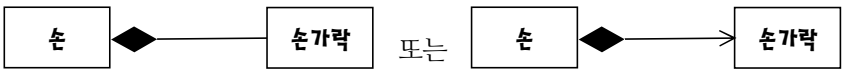
- 일반적으로, 연관과 의존 관계는 어떤 클래스가 다른 클래스를 **이용**하는 것이다.

〈집합(aggregation)〉



- 집합(aggregation) : 차와 바퀴의 관계(생명주기 불일치)
- 표기는 전부(whole)와 부분(part)을 실선으로 연결하고, 전부 쪽에 **빈 다이아몬드**를 표기
- 부분 쪽에는 화살표를 표시해도 되고, 표시하지 않아도 된다.

〈합성(composition)〉



- 합성(composition) : 손과 손가락의 관계(생명주기 일치)
- 표기는 전부와 부분을 실선으로 연결하고, 전부 쪽에 속이 찬 **진한 다이아몬드**를 표기
- 역시, 부분 쪽에는 화살표를 표시해도, 되고 표시하지 않아도 된다.

기출문제 분석

1. UML의 클래스 다이어그램에서 클래스 사이의 관계에 대한 설명으로 옳지 않은 것은? [2021년 계리]

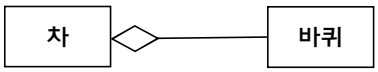
① 일반화(generalization) 관계는 일반화한 부모클래스와 실체화한 자식클래스 간의 상속 관계를 나타낸다.
② 연관(association) 관계에서 다중성(multiplicity)은 관계 사이에 개입하는 클래스의 인스턴스 개수를 의미한다.
③ 의존(dependency) 관계는 한 클래스가 다른 클래스를 참조하는 것으로 지역변수, 매개변수 등을 일시적으로 사용하는 관계이다.
④ 집합(aggregation) 관계는 강한 전체와 부분의 클래스 관계이므로 전체 객체가 소멸되면 부분 객체도 소멸된다.

♤ **UML의 클래스 다이어그램**

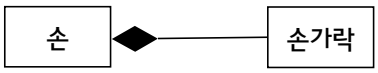
- 집합(aggregation) 관계는 **강한** 전체와 **부분**의 클래스 관계이므로 전체 객체가 소멸되면 부분 객체도 소멸된다.**(×)** → 집합은 **약한 전체 부분관계**이다.

집합 (aggregation)	• **약한 전체 부분관계,** 집합은 생명주기가 일치하지 않는다. • 부분은 자체적으로 존재할 수 있고, 없어질 수도 있다. • 그리는 방법 : **포함**하고 있는 **클래스 쪽**에 끝이 **빈 다이아몬드**로 표시 • 예 : 회사와 부서의 관계 - 회사에는 여러 부서가 존재한다.
합성[복합] (composition)	• **강한 전체 부분관계,** 합성은 생명주기를 공유한다. • 부분만으로는 스스로 존재할 수 없다. • 합성은 집합 관계보다 더 끈끈한 관계이다.(더 강한 집합) • 그리는 방법 : **포함**하고 있는 **클래스 쪽**에 **검은 다이아몬드**로 표시 • 예 : 손(hand)은 손가락을 포함하고 있다.

- **집합**(aggregation) : 차와 바퀴의 관계(생명주기가 불일치하는 경우)



- **합성**(composition) : 손과 손가락의 관계(생명주기가 일치하는 경우)



정답 : ④

2. 클래스 다이어그램의 특징으로 가장 옳지 않은 것은? [2018년 서울 7급]

① 클래스들의 인터페이스를 나타낸다.
② 클래스는 객체의 집합이 가지는 속성과 동작을 추상적으로 나타낸다.
③ 시스템 동적구조를 모델링한다.
④ 속성값의 타입을 정하는 것은 시스템 정의에서 중요한 일이 아니므로 객체 설계 단계까지 미뤄도 된다.

♤ **클래스 다이어그램**

• 시스템 동적구조를 모델링한다.(×) → 클래스 다이어그램은 시스템 정적구조를 모델링한다.

정답 : ③