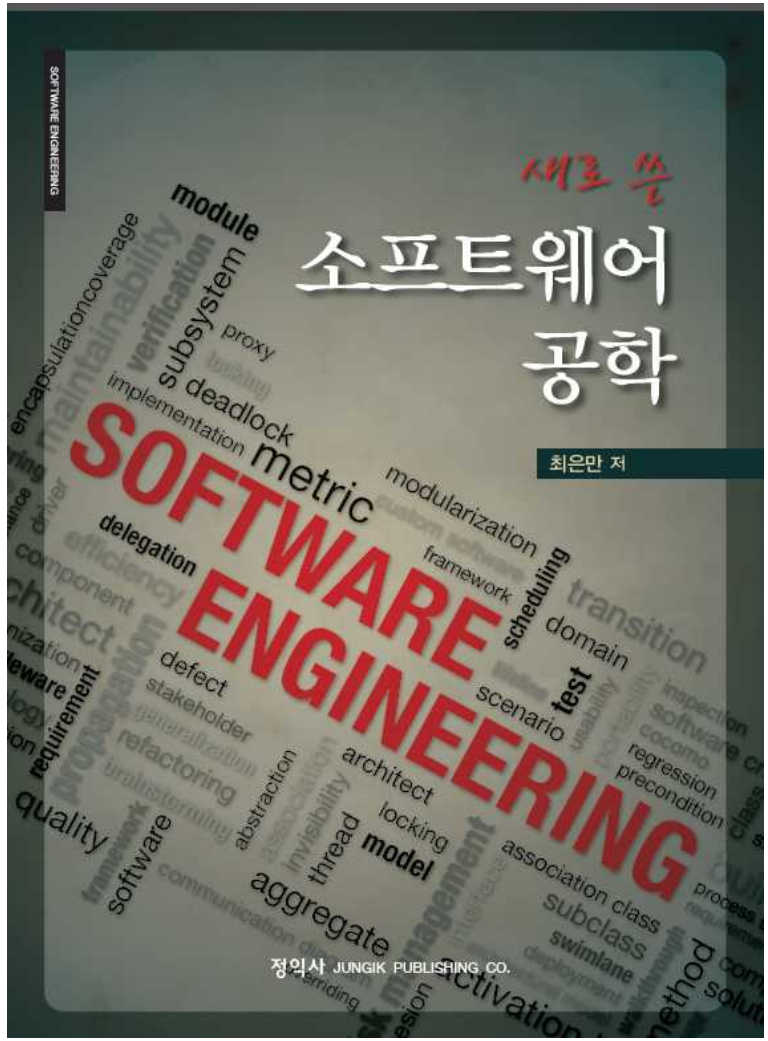
소프트웨어 공학 개론

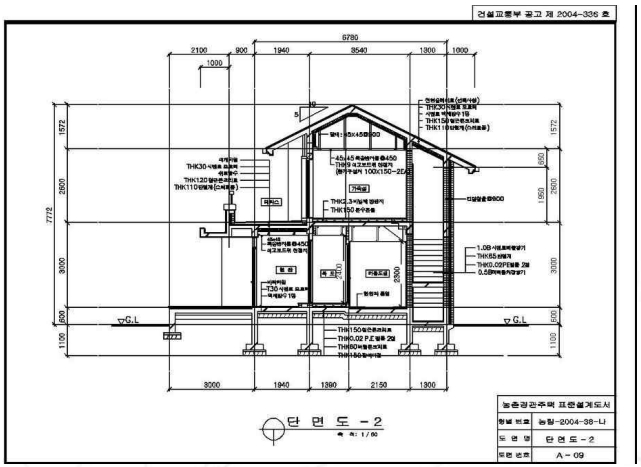
강의 6: 클래스 다이어그램

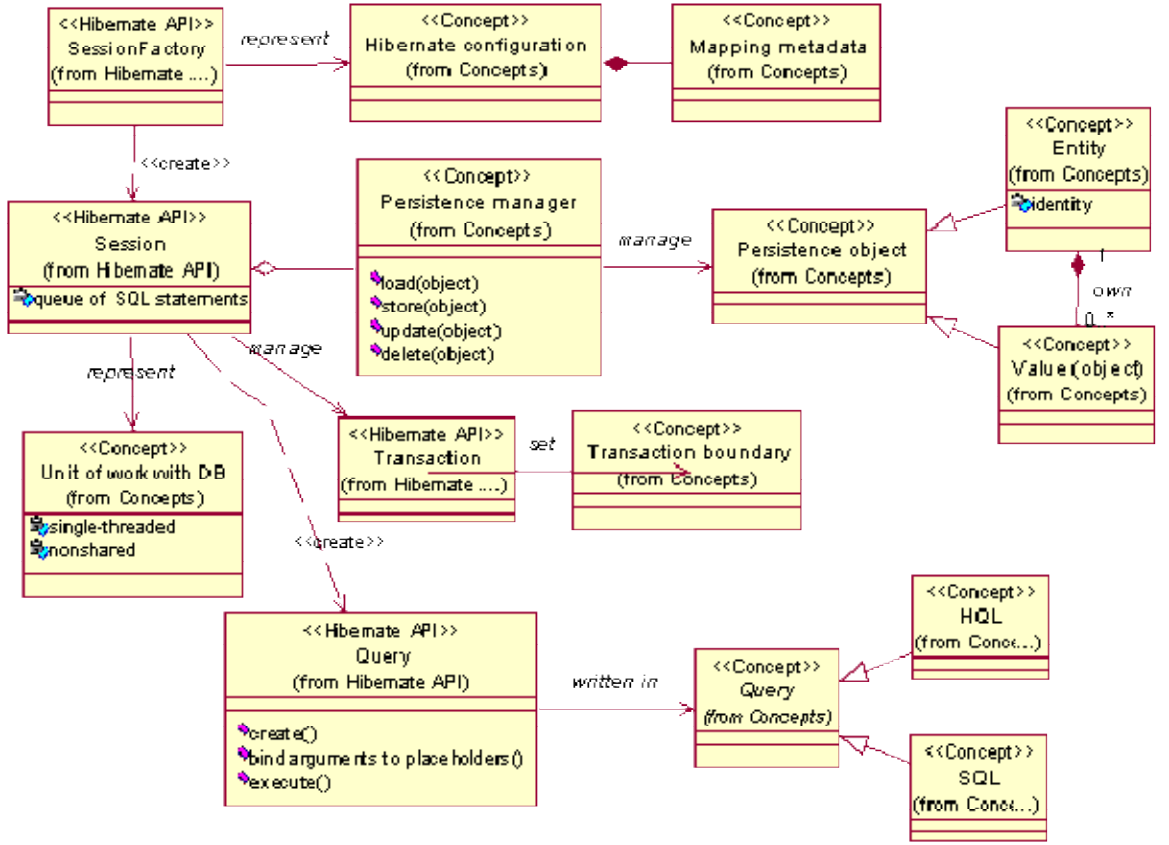
최은만
동국대학교 컴퓨터공학과



모델링

- 설계와 모델링
- UML
- 클래스 다이어그램





UML

- 분석, 설계를 비주얼 화, 문서화 하기 위한 그래픽 언어

- Unified
  - 이전의 OO 방법들의 통합
- Modeling
  - 객체지향 분석 설계를 위한 비주얼 모델링
- Language
  - 모형화된 지식(의미)을 표현

UML은 [ ] 이다.

- 시스템에 대한 지식을 찾고 표현하기 위한 언어
- 시스템을 개발하기 위한 탐구 도구
- 비주얼 모델링 도구
- 근거가 잘 정리된 가이드라인
- 분석, 설계 작업의 마일스톤
- 실용적 표준

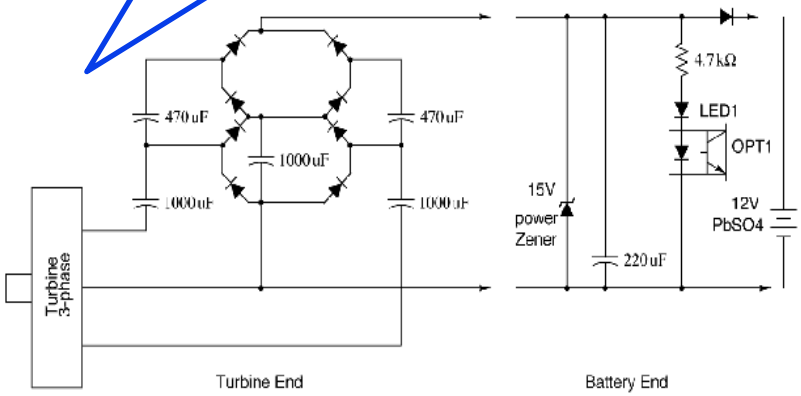
UML은 [ ] 이 아니다.

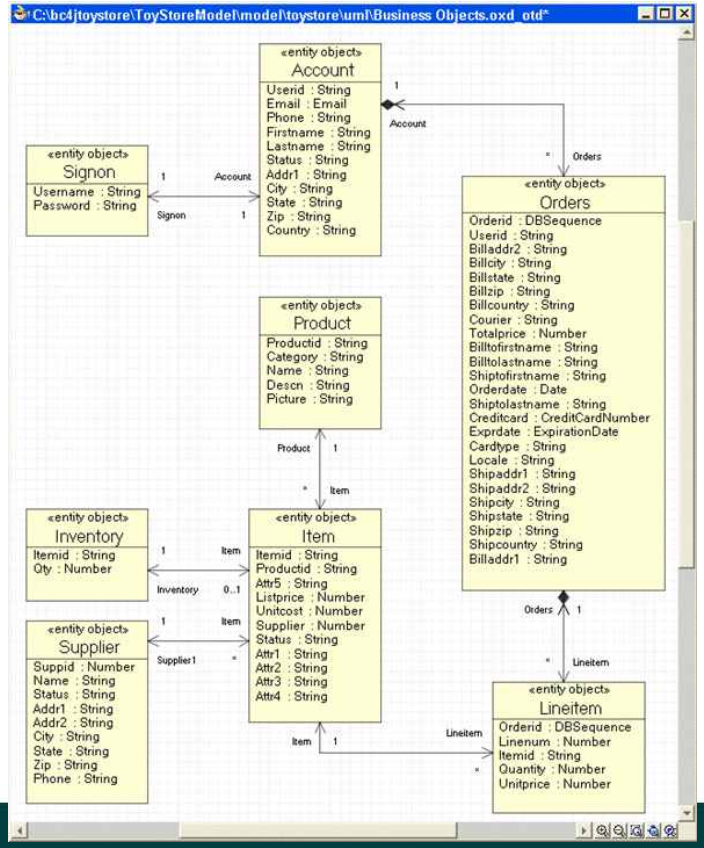
- 비주얼 프로그래밍 언어 환경
- 데이터베이스 표현 도구
- 개발 프로세스(SDLC)
- 모든 문제의 해결책
- 품질 보증 방안

UML

Every s/w engineer WILL understand UML diagrams.

Every h/w engineer understands curcuit diagram.





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

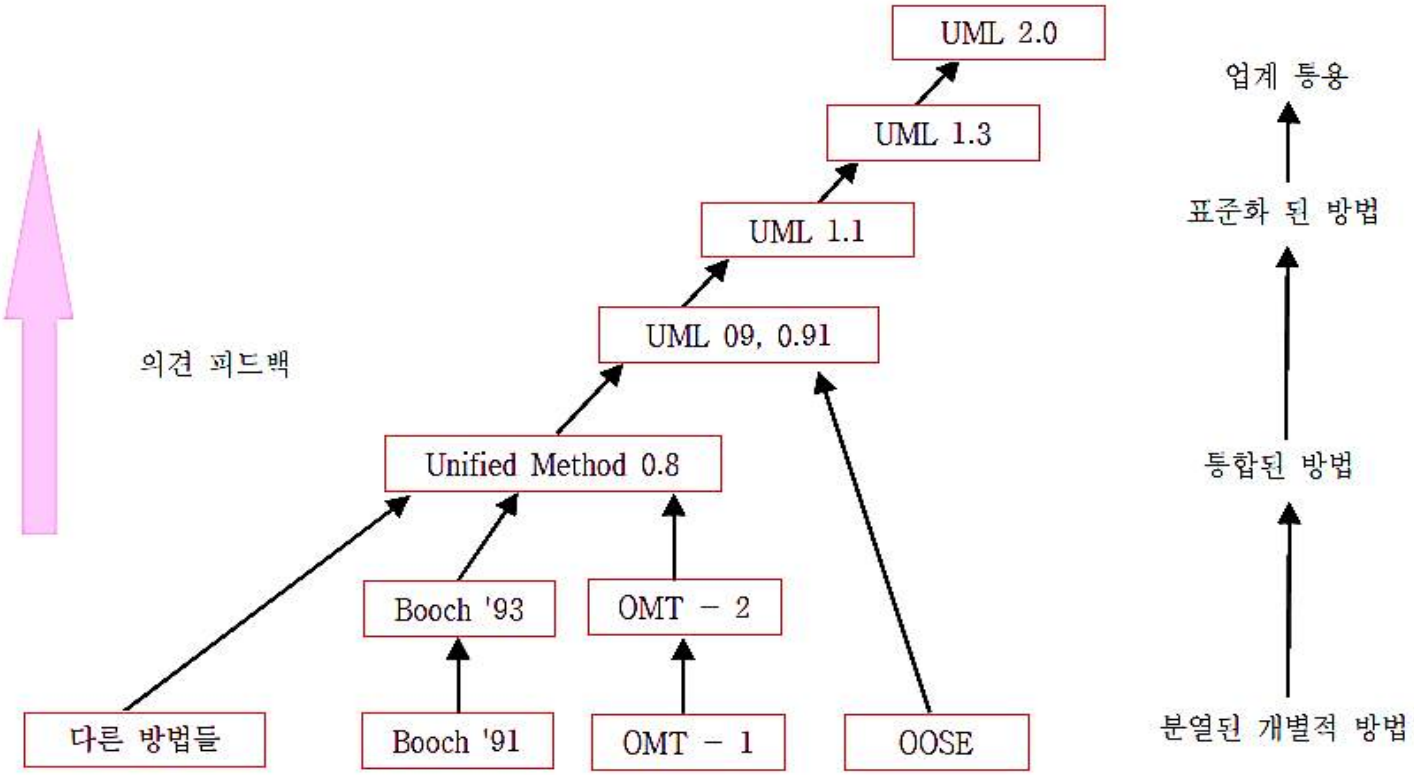
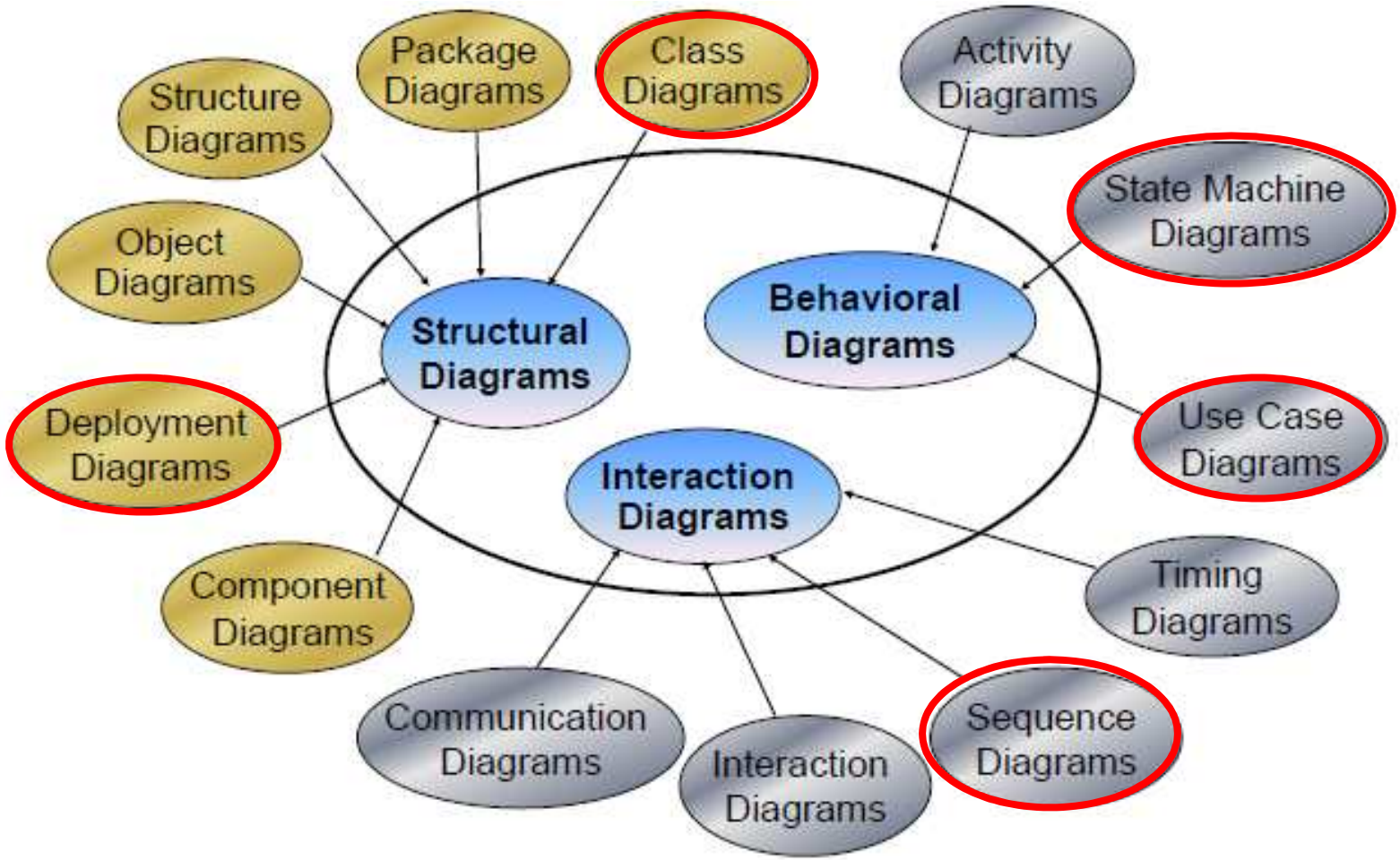


그림 5.9 ▶ UML의 진화

UML 2.0 다이어그램 체계



UML 다이어그램

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

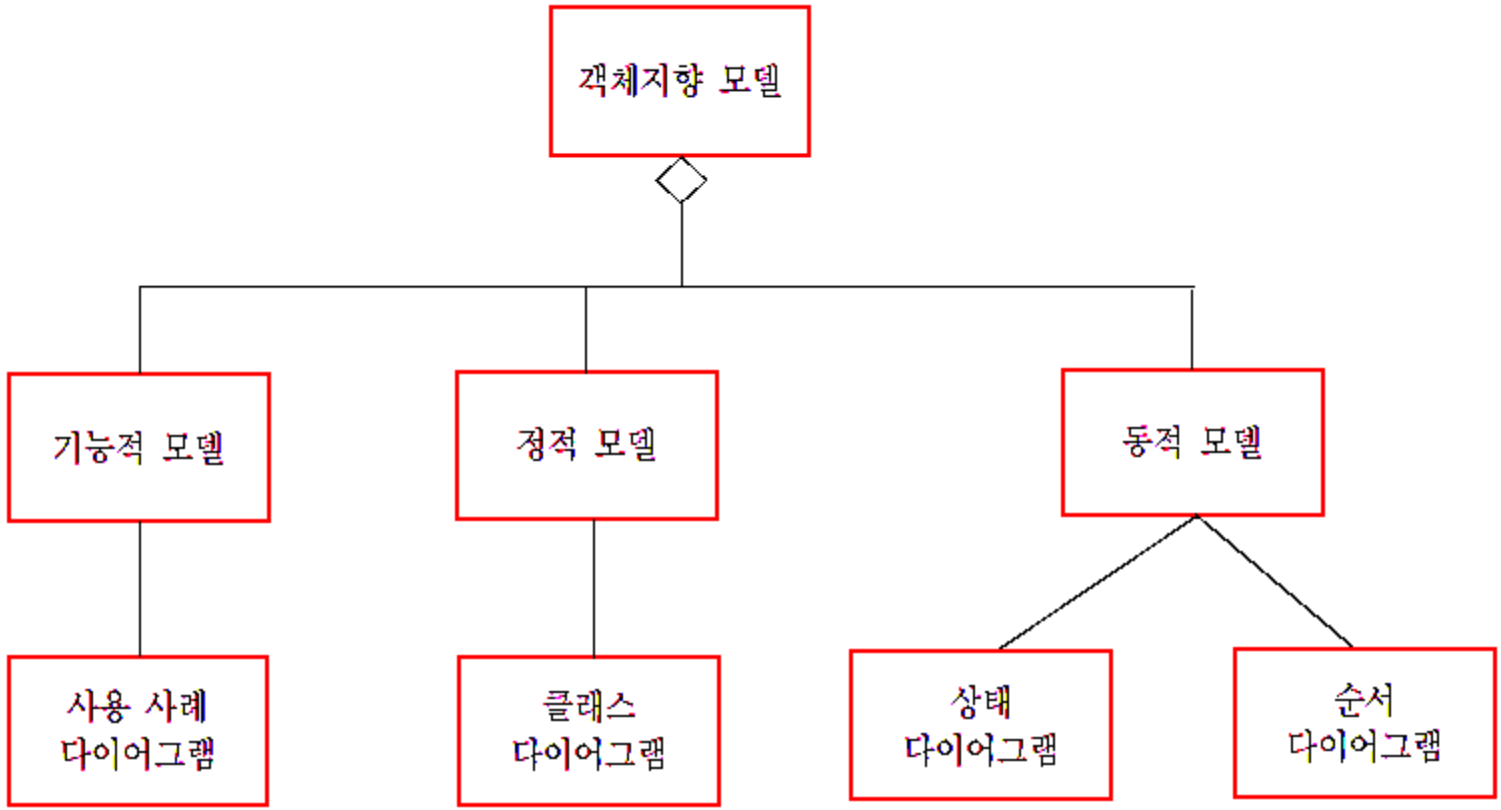


그림 5.10 ▶ 세 가지 측면의 모델

UML 모델링 과정

❶ 요구를 사용 사례로 정리하고 사용 사례 다이어그램을 작성

❷ 클래스 후보를 찾아내고 개념적인 객체 모형을 작성

❸ 사용 사례를 기초하여 순서 다이어그램을 작성

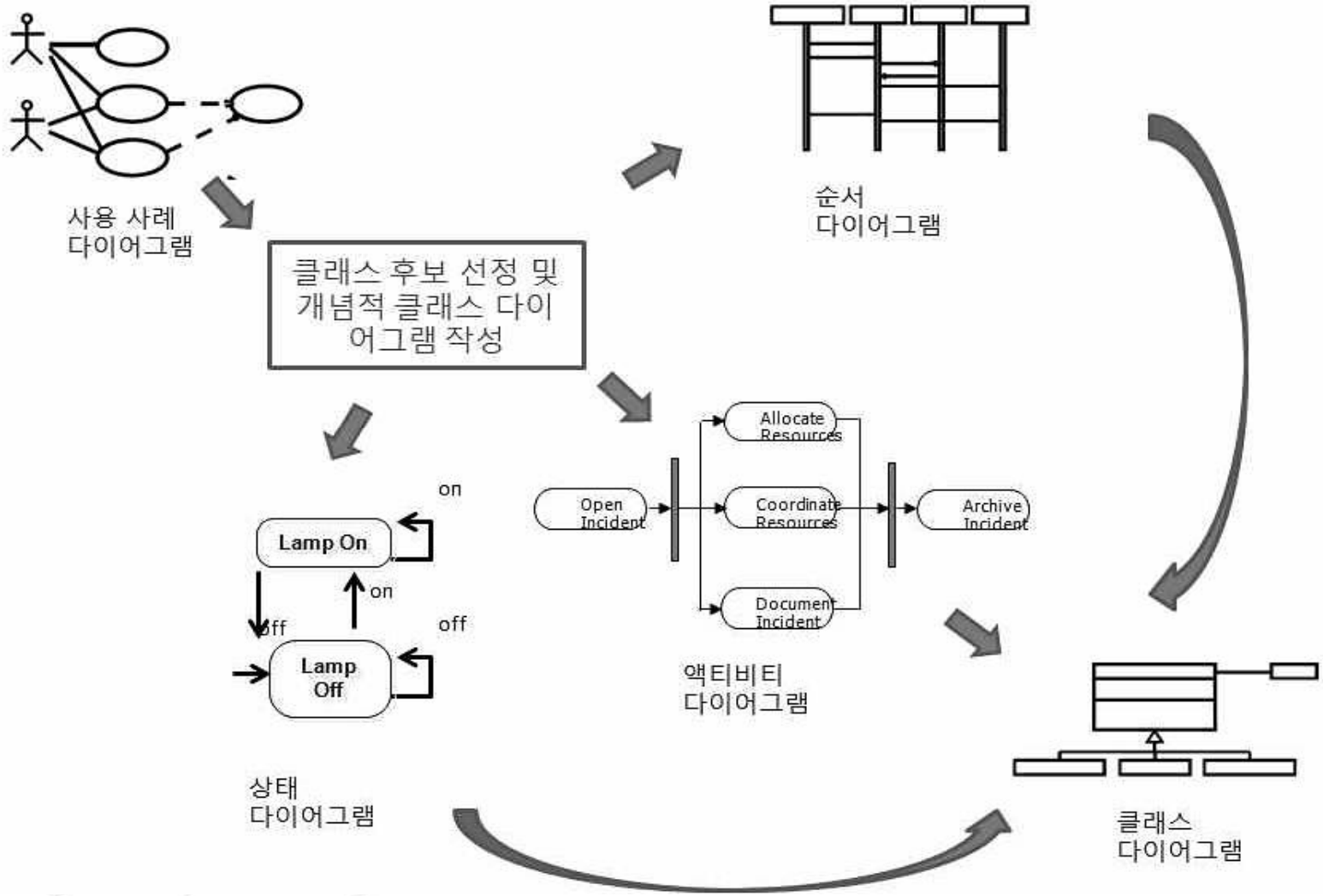
❹ 클래스의 속성, 오퍼레이션 및 클래스 사이의 관계를 찾아
객체 모형을 완성

❺ 상태 다이어그램이나 액티비티 다이어그램 등 다른 다이어그램을
추가하여 UML 모델을 완성

❻ 서브시스템을 파악하고 전체 시스템 구조를 설계

❼ 적당한 객체를 찾아내거나 커스텀화 또는 객체를 새로 설계

UML 모델링 과정

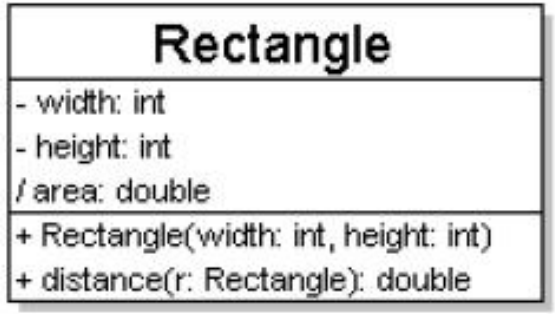

사용 사례
다이어그램
순서
다이어그램
클래스 후보 선정 및
개념적 클래스 다이
어그램 작성
Allocate
Resources
Open
Incident
Coordinate
Resources
Archive
Incident
Document
Incident
액티비티
다이어그램
on
Lamp On
on
off
off
Lamp
Off
상태
다이어그램
클래스
다이어그램

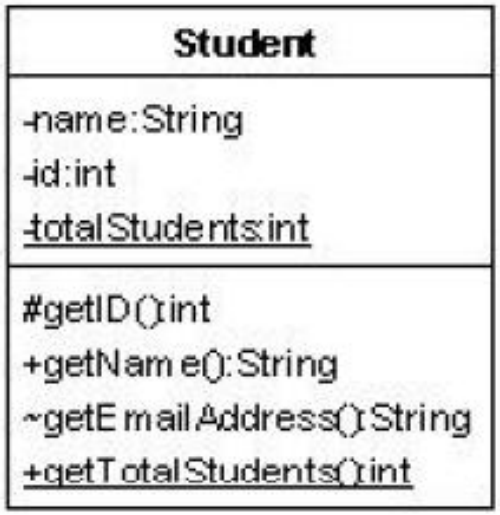
UML 클래스 다이어그램

- UML 클래스 다이어그램
  - 객체지향 시스템에 존재하는 클래스, 클래스 안의 필드, 메소드, 서로 협력하거나 상속하는 클래스 사이의 연결 관계를 나타내는 그림
- 나타내지 않는 것
  - 클래스가 서로 어떻게 상호작용 하는지
  - 자세한 알고리즘
  - 특정한 동작이 어떻게 구현되는지

클래스 나타내기

- 박스 위에 클래스 이름
  - 추상 클래스는 이탤릭체
  - 인터페이스 클래스는 <<interface>> 추가
- 속성
  - 객체가 가지는 모든 필드를 포함
- 오퍼레이션/메소드
  - 아주 흔한 메소드(get/set)는 생략
  - 상속된 메소드도 포함할 필요 없음





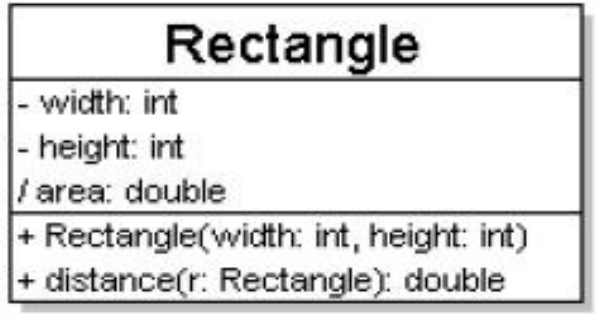
클래스 속성

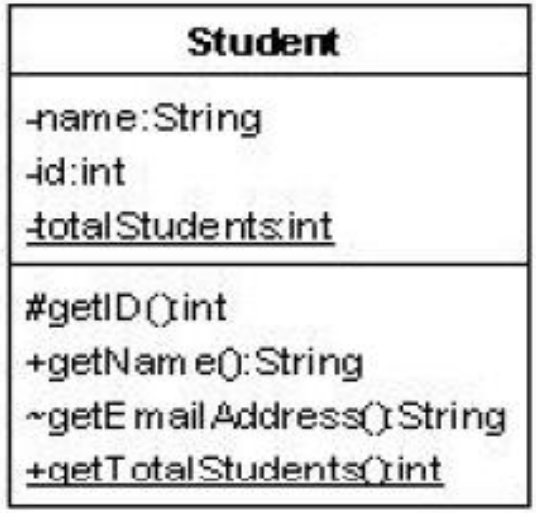
- 속성(필드, 인스턴스 변수)

visibility name: type[count] = default value

visibility:

+	public
#	protected
-	private
~	package(디폴트)
/	derived
_	static variable





- 파생된 속성: 저장되지 않고 다른 속성값으로부터 계산됨

클래스 오퍼레이션/메소드

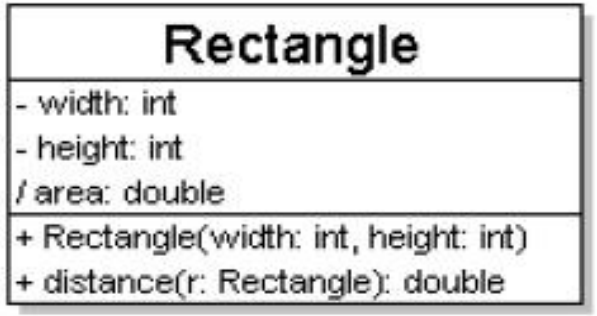
- 오퍼레이션/메소드

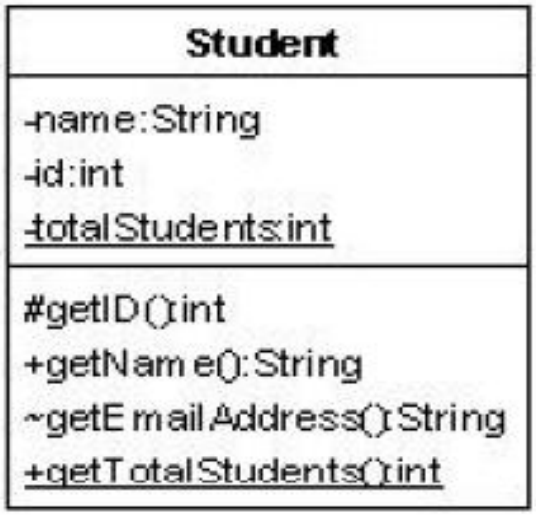
  visibility name(parameters) : return_type

  visibility: + public
  # protected
  - private
  ~ package(디폴트)
  _ static method

- 파리메타 타입 (name: type)

- 생성자나 리턴 타입이 void인 경우는 return_type 생략





클래스 사이의 관계

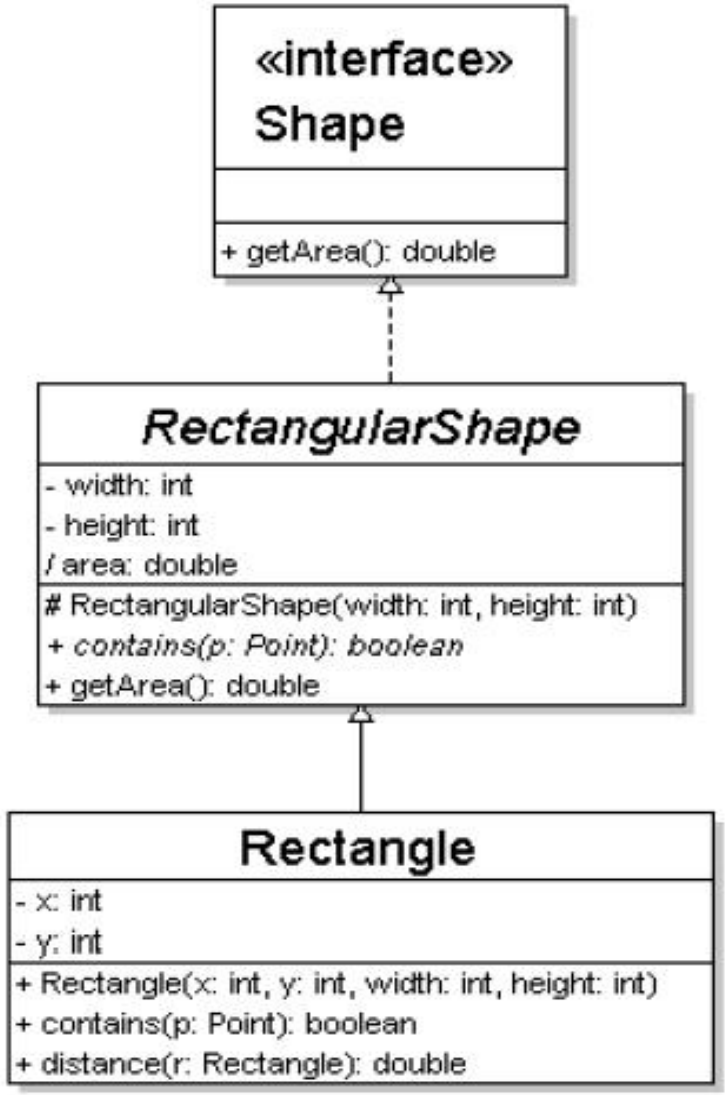
- 일반화(generalization): 상속(is_a) 관계
  - 클래스 사이의 상속
  - 인터페이스 구현

- 연관(association): 사용(usage) 관계(3 종류)
  - 의존(dependence): 상대의 정보가 필요
  - 집합(aggregation): 어떤 클래스가 다른 클래스의 모임으로 구성
  - 합성(composition): 포함된 클래스가 컨테이너 클래스가 없이는 존재할 수 없는 집합관계의 변형

일반화 관계

- 일반화(상속)
  - 부모를 향한 화살표로 표시되는 하향 계층 관계
  - 선/화살표는 부모 클래스의 종류에 따라 다름
    - ❖클래스:
      실선/검은 헤드 화살표
    - ❖추상 클래스:
      실선/흰 헤드 화살표
    - ❖인터페이스:
      점선/흰 헤드 화살표

r



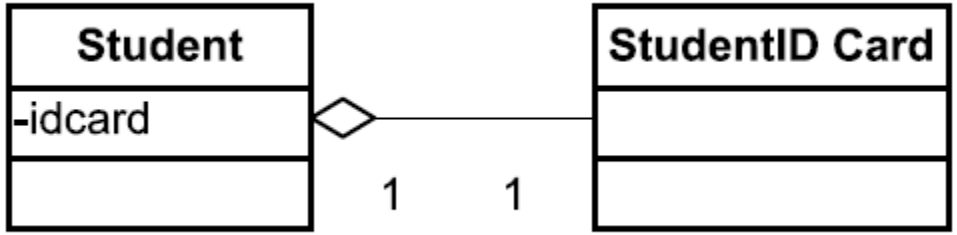
연관 관계

- 연관(association)
  - 어떤 클래스의 인스턴스가 작업을 수행하기 위하여 다른 클래스를 알아야 함

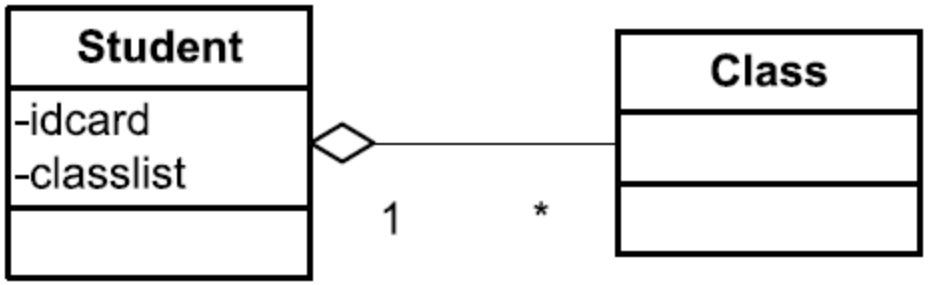
1. 다중도(multiplicity)
   - * ⇨ 0, 1, or more
   - 1 ⇨ 정확히 1개
   - 2..4 ⇨ 2개 내지 4개
   - 3.. * ⇨ 3개 이상
2. 이름 – 객체들의 관계 이름
3. 방향성(navigability) – 질의의 방향, 객체 사이의 선으로 표시하며 양쪽 방향인 경우는 화살표시 없음

연관 관계의 다중도

- 1 대 1
  - 학생 1명이 학생증(id card) 한 개만을 가진다.

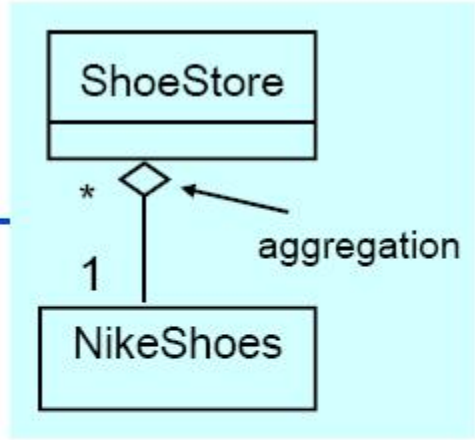


- 1 대 다
  - 학생 1명이 여러 클래스를 수강할 수 있다.

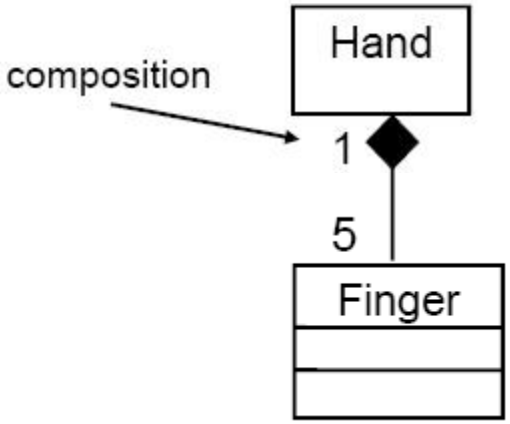


연관의 타입

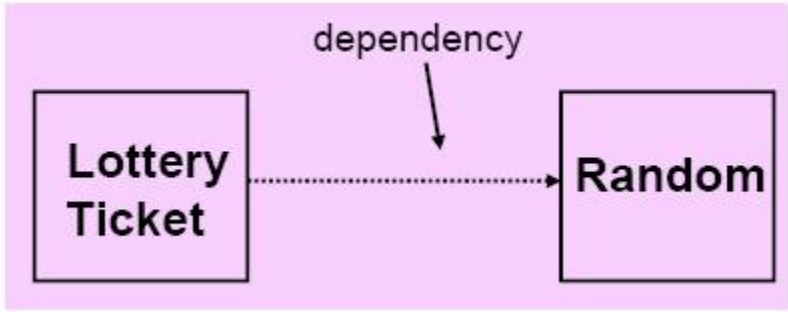
- 집합(aggregation): “contains”
  - 포함하고 있는 클래스 쪽에 하얀 다이아몬드 표시



- 합성(composition): “이 목적을 위하여만 포함됨”
  - 집합보다 더 끈끈한 관계
  - 부분은 전체가 살고 죽느냐에 좌우 됨
  - 포함하고 있는 클래스 쪽에 검은 다이아몬드로 표시

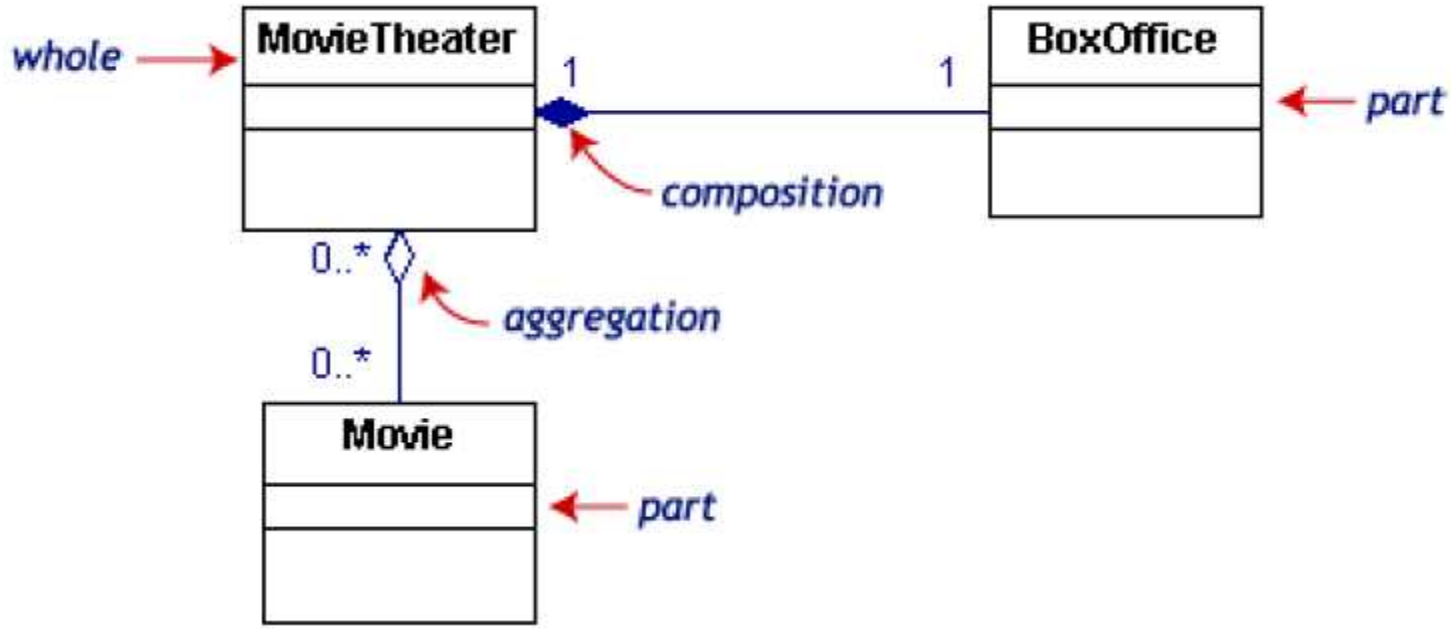


- 의존(dependency): “일시적 사용”
  - 점선으로 표시

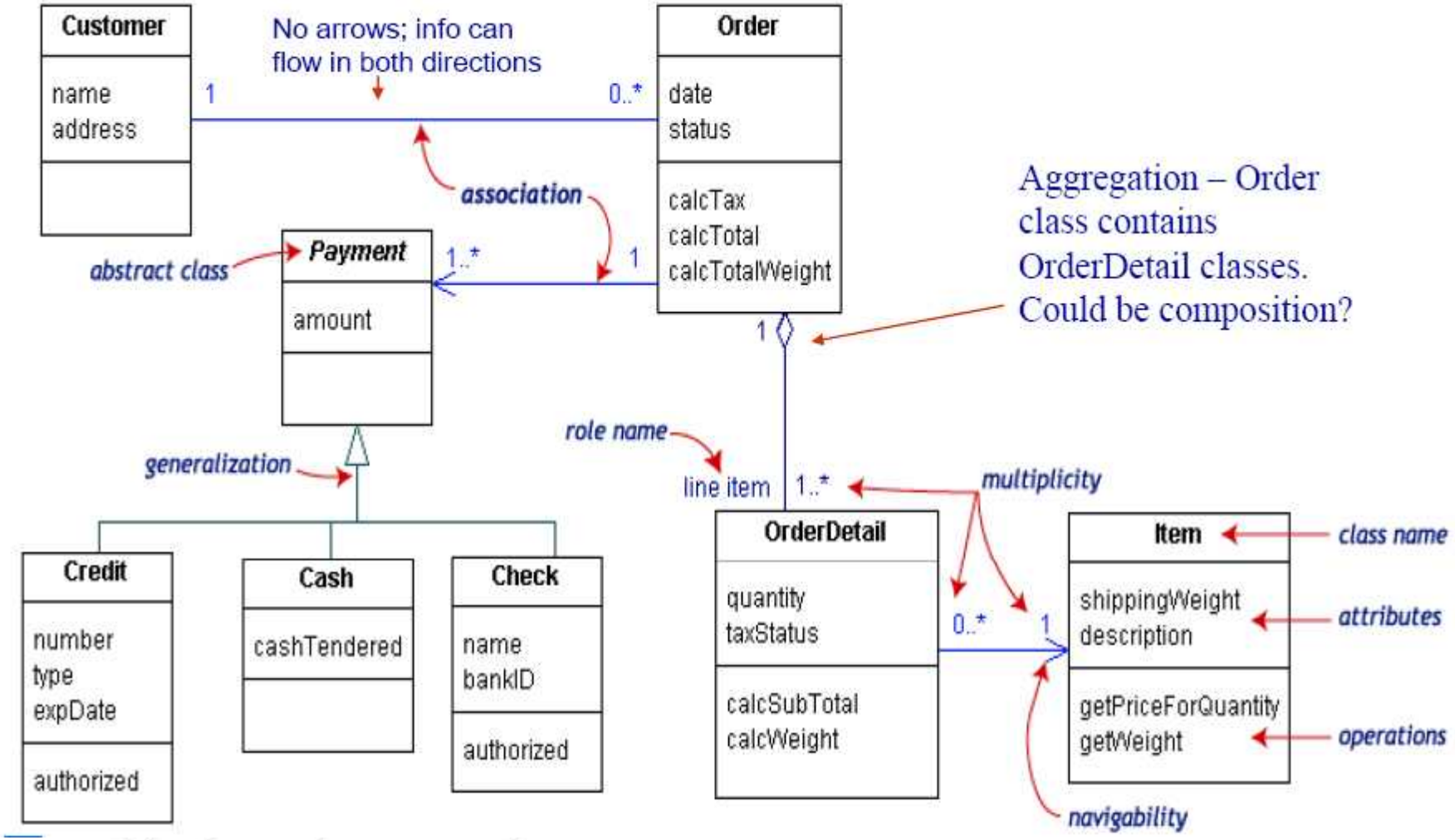


합성/집합 관계의 예

- 영화관이 없어지면
  - 매표소도 없어짐 ⇒ 합성
  - 그러나 영화는 아직 존재 ⇒ 집합

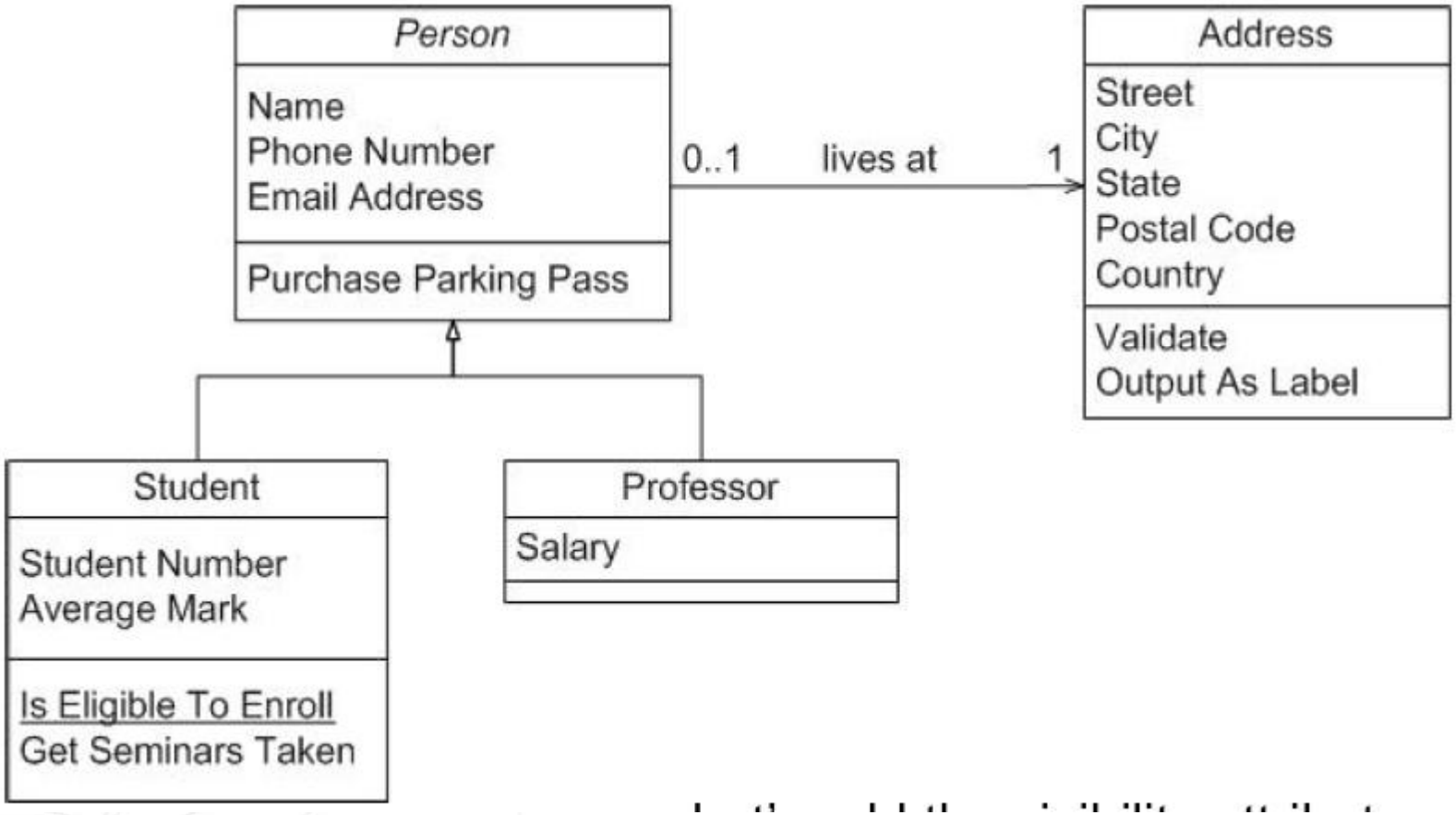


사례: 클래스 다이어그램


Customer
name
address
No arrows; info can flow in both directions
1
0..*
Order
date
status
association
calcTax
calcTotal
calcTotalWeight
abstract class
Payment
amount
1..*
1
Aggregation – Order class contains OrderDetail classes. Could be composition?
1
role name
line item
1..*
multiplicity
generalization
Credit
number
type
expDate
authorized
Cash
cashTendered
Check
name
bankID
authorized
OrderDetail
quantity
taxStatus
calcSubTotal
calcWeight
0..*
1
Item
shippingWeight
description
getPriceForQuantity
getWeight
class name
attributes
operations
navigability

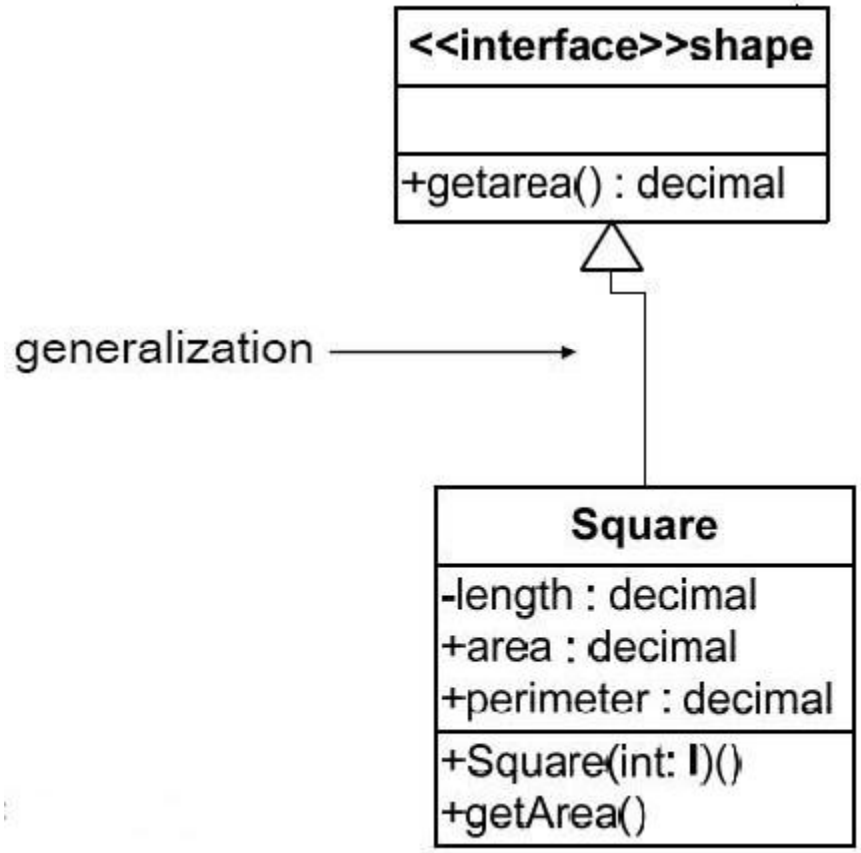
클래스와 속성

- **Address는 Person의 속성?**



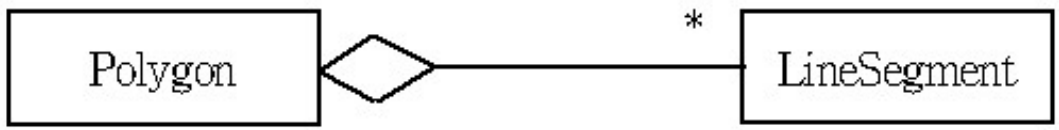
설계 오류

- Visibility에 오류가 있는 것은?
  - area
- 생성자 타입의 오류?

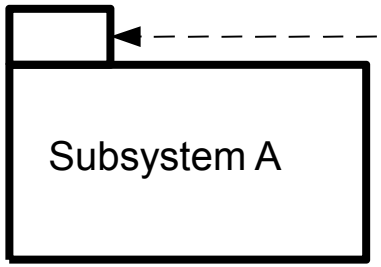


전파(propagation) 현상

- 전체 개념의 오퍼레이션이 부분 개념의 오퍼레이션에 의하여 구현되는 현상
- 동시에 부품의 속성이 전체 개념에 전파되는 현상
- 전파(propagation)와 전체부분 개념의 관계는 상속과 일반화 관계와 유사
  - 중요한 차이는
    - 상속은 묵시적인 메커니즘
    - 전파는 필요할 때 프로그램

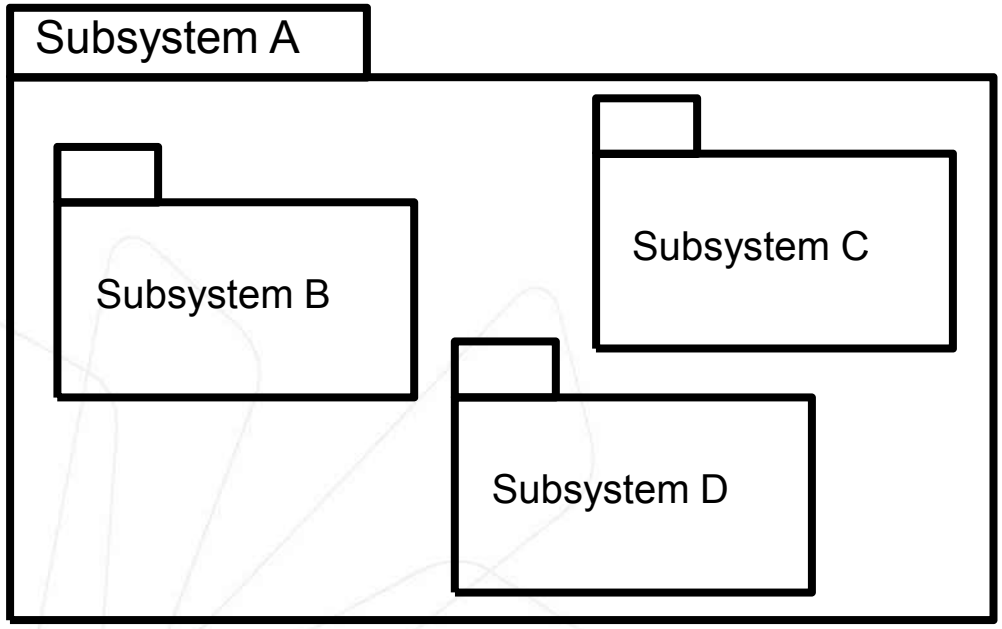


패키지 다이어그램



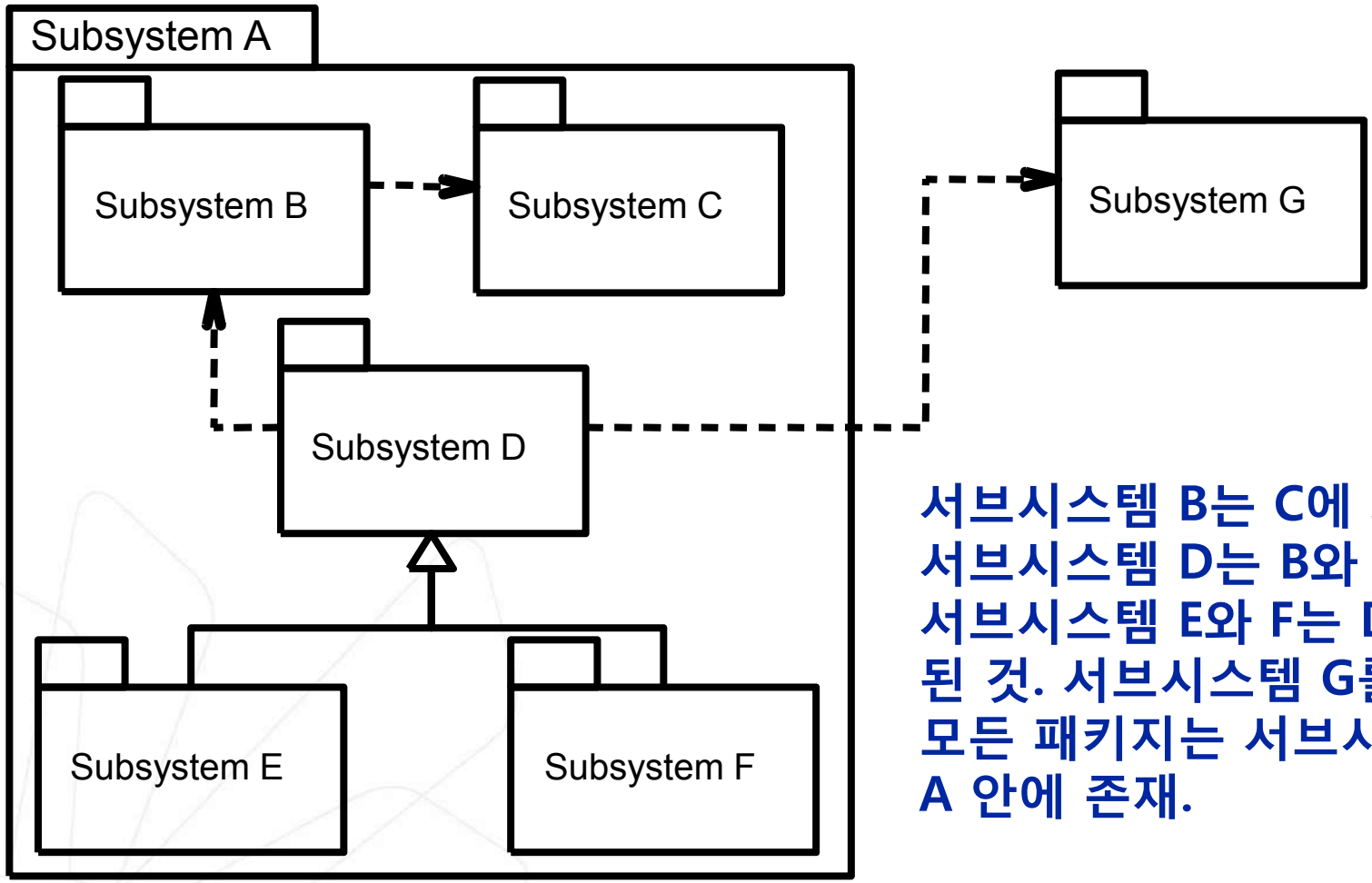
패키지 탭

"Subsystem A"로 부르기로 한 패키지의 외부관점
패키지는 서브시스템으로 부름



"서브 시스템 A는 세 개의 다른 패키지 "서브시스템 B, C, D"를 그루핑, 확장된 패키지의 이름은 패키지 탭에 표기.

패키지 사이의 관계



서브시스템 B는 C에 의존한다.
서브시스템 D는 B와 G에 의존
서브시스템 E와 F는 D의 상세화된 것. 서브시스템 G를 제외한 모든 패키지는 서브시스템 A 안에 존재.

연습 문제 #1

- 항공권 예약 문제
  - 예약의 기록은 항상 탑승객 한 명 단위로 이루어짐. 탑승객이 없는 예약은 없음
  - 예약에 탑승객이 여러 명인 경우는 없음
  - 탑승객이 다수의 예약을 할 수 있음
  - 탑승객이 예약이 하나도 없을 수 있음

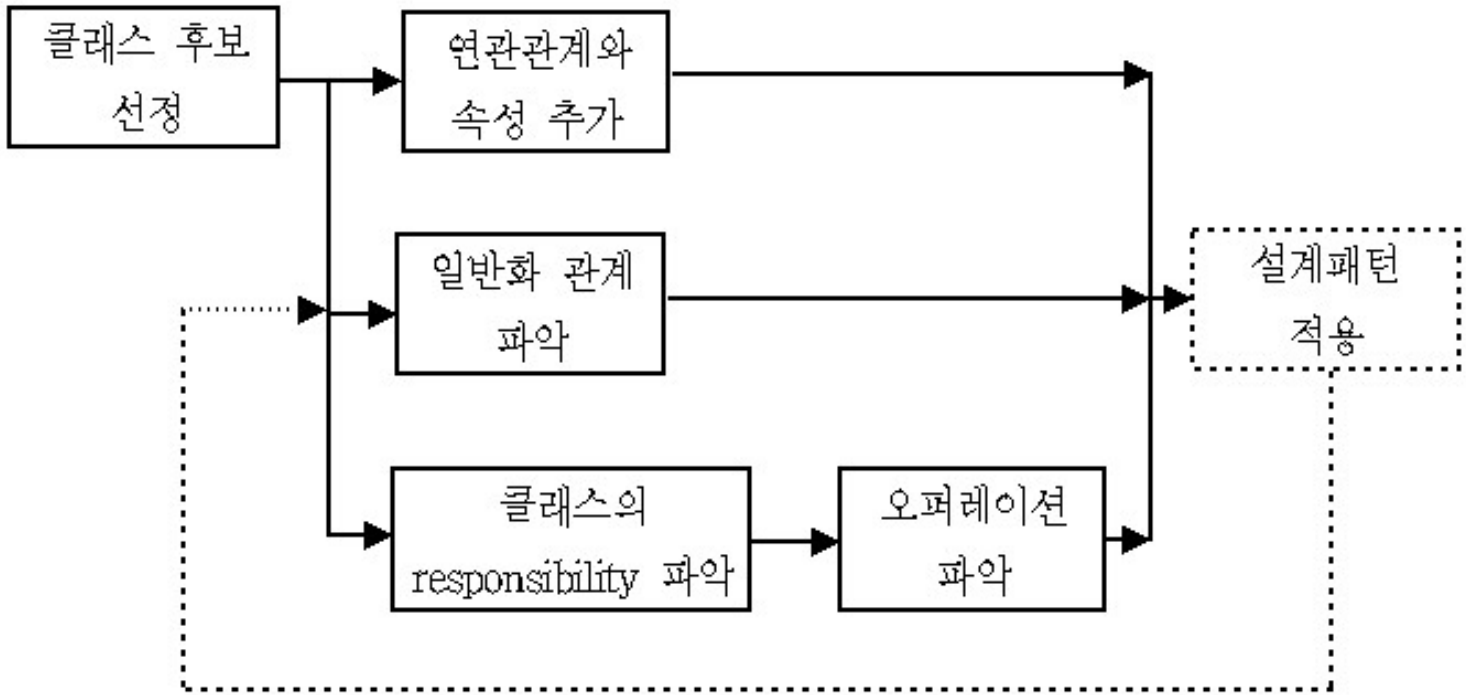
연습 문제 #2

- 자동판매기를 객체지향으로 개발하기로 하였다. 자판기 안에 있는 여러 컴포넌트들의 관계를 나타내는 UML 클래스 다이어그램을 완성하시오.

  - 자판기에는 동전을 일정 시간 넣지 않으면 자동으로 동전을 내뱉기 위하여 클락이 내장되어 있다.
  - 음료수 선택을 위한 버튼
  - 동전 슬롯
  - Shelf 센서와 배출구

클래스 다이어그램 작성 과정

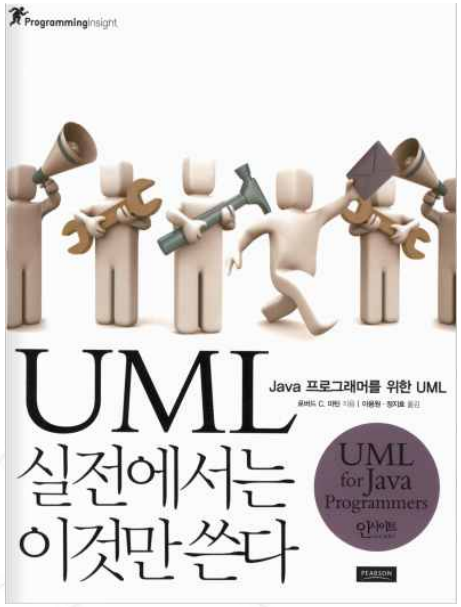
- 반복, 점증적 방법
  - 초벌로 작성 후 계속 추가, 삭제

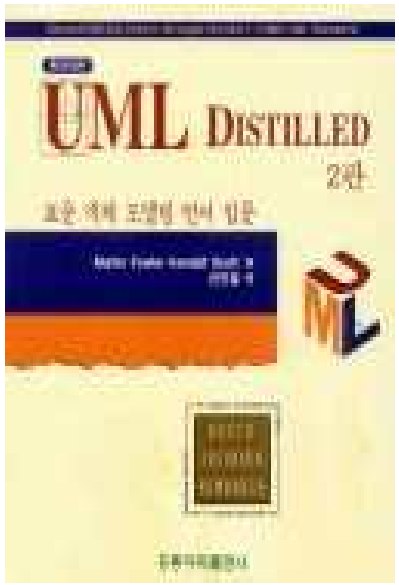


참고 문헌

- UML 실전에서는 이것만 쓴다, 인사이트

- UML에 관련된 많은 서적과 웹 튜토리얼이 있음
  "UML Distilled", by Martin Fowler.





Questions?

dongguk UNIVERSITY