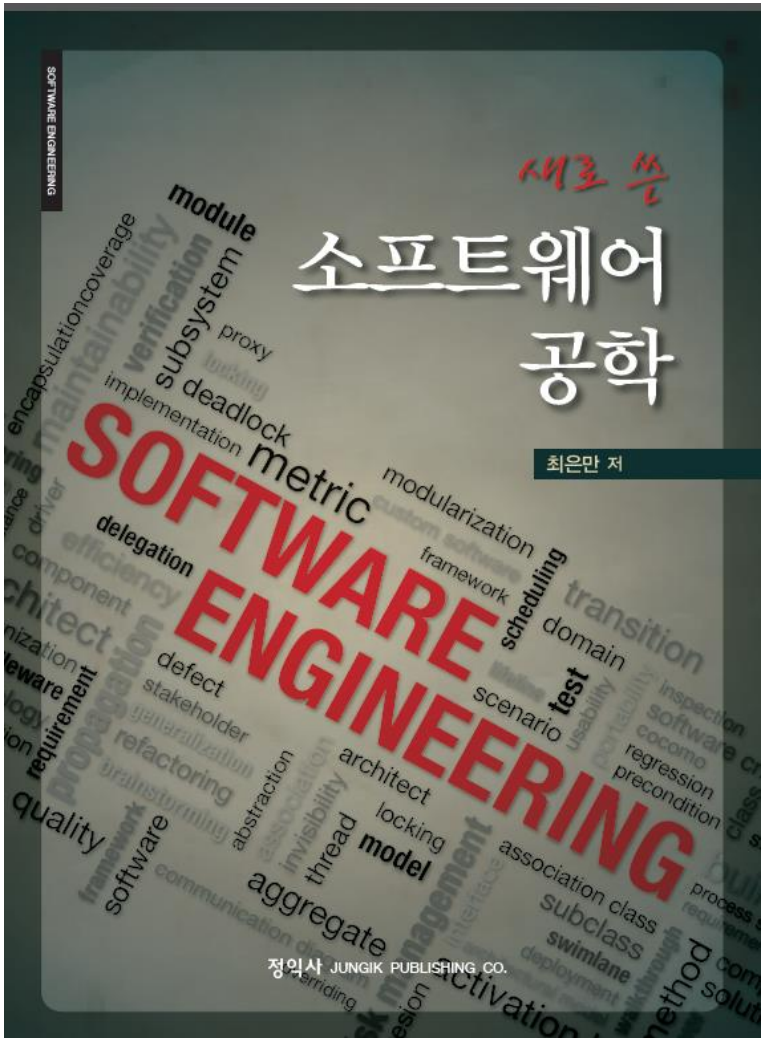
소프트웨어 공학

Lecture #5: 모델링



최은만 저

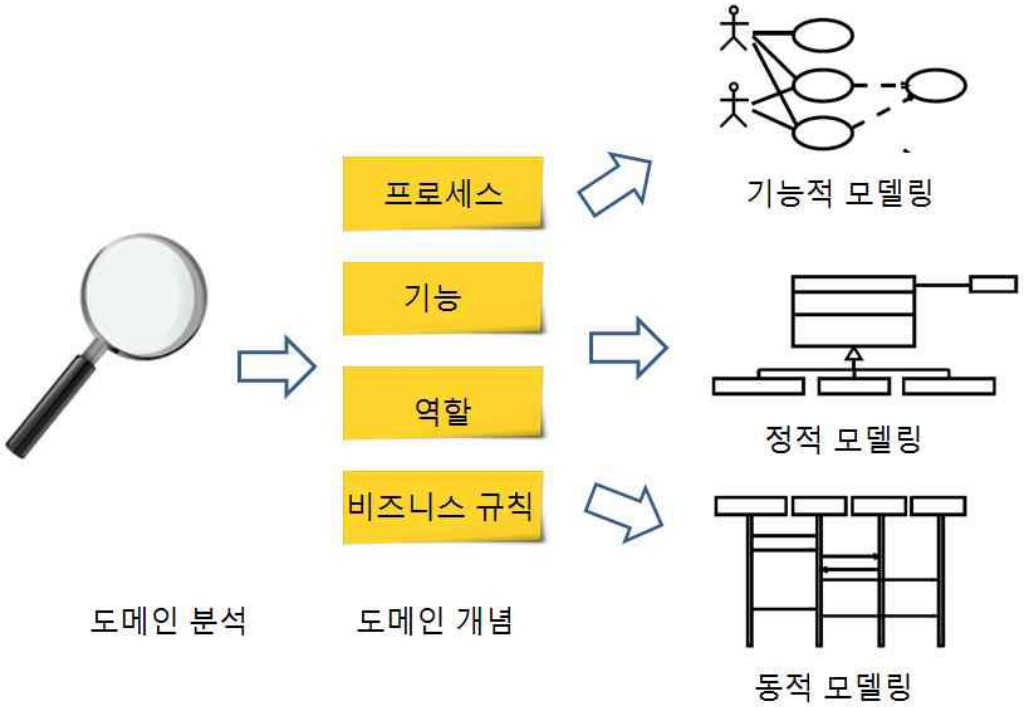
6차 개정판

학습 목표

- 객체지향 개념
- UML
- 정적 모델링
- 동적 모델링
- 모델링 도구

모델링

- 모델링
  - 도메인 지식을 체계화 하는 과정
  - 중요한 도메인 개념과 특성, 관계를 파악하여 다이어그램으로 정형화
  - 모델링 과정



5.1 객체지향 개념

- 모델링이 개발자에게 주는 도움
  - 응용문제를 이해하는 데 도움을 줌
  - 개발팀원들 사이에 응용문제의 공통 개념으로 대화하게 하고 개선시킴
  - 파악한 개념을 사용자와 고객에게 전달 할 때 도움을 줌
  - 후속 작업 즉 설계, 구현, 테스팅, 유지보수에 개념적인 기준을 제공

- 객체지향의 장점
  - 개발자가 설계를 작성하고 이해하기 쉬움
  - 자료와 함수를 함께 추상화 함으로써 변화에 영향을 적게 받음
  - 사용자 중심, 대화식 프로그램의 개발에 적합
  - 프로그램을 뚜렷하게 구별되는 단위(object)로 분할 가능

객체지향과 절차적 방법의 비교

- 객체지향은 주어진 문제 영역을 그 안에 존재하는 객체의 집합으로 보며, 객체들은 서로 정보를 주고받아 상호 작용한다고 여김

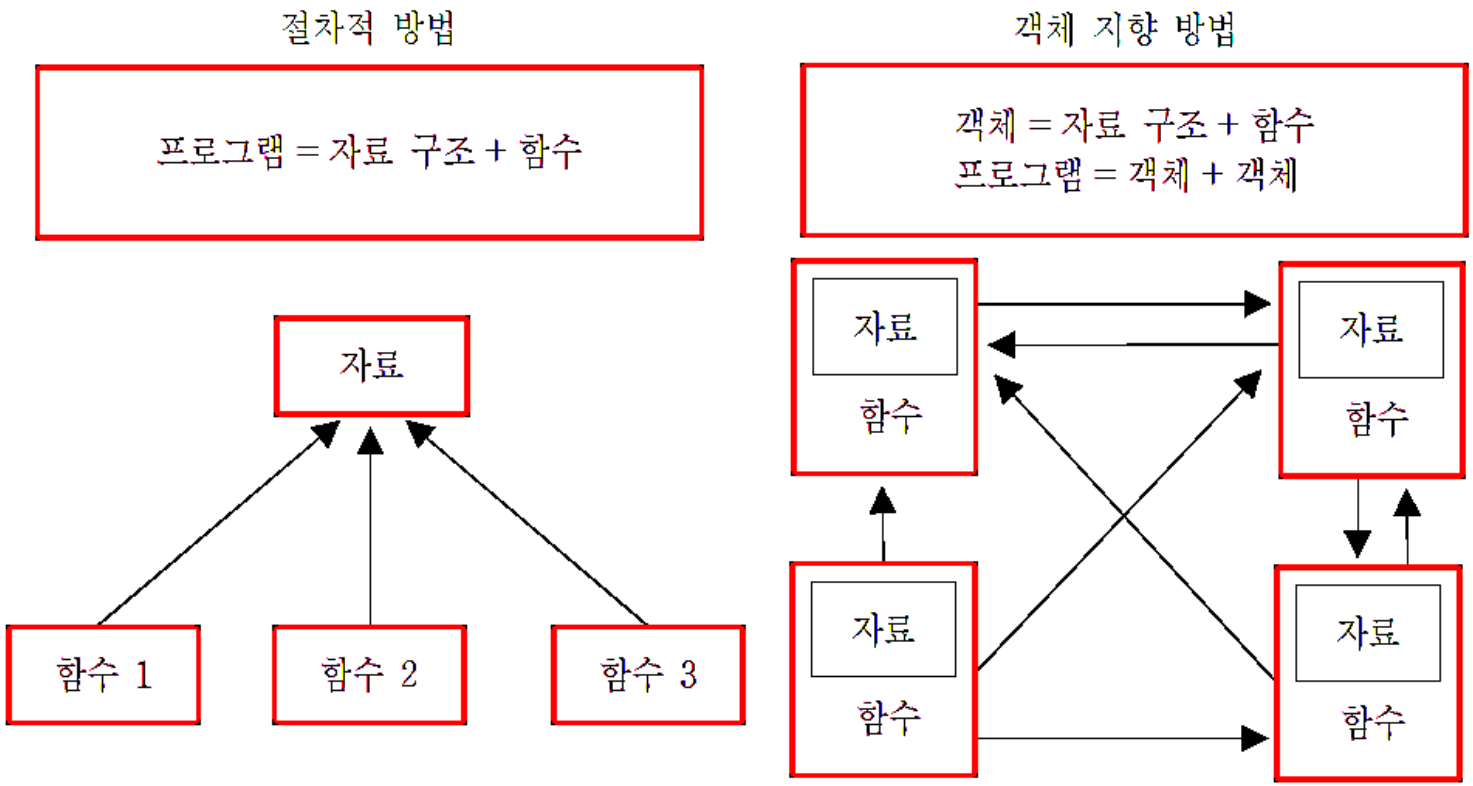


그림 5.2 ▶ 두 가지 방법의 시스템에 관한 시각

클래스와 객체

- 클래스 : 속성과 오퍼레이션을 캡슐화
- 객체 : 클래스의 인스턴스

```
class Employee {
public:
    promote(from, to);
    increase_salary(new_salary);
    change_phone(new_number);
    ...
private:
    char*          name;
    positiontype   position;
    int            salary;
    phonetype      phone;
    ...
}
```

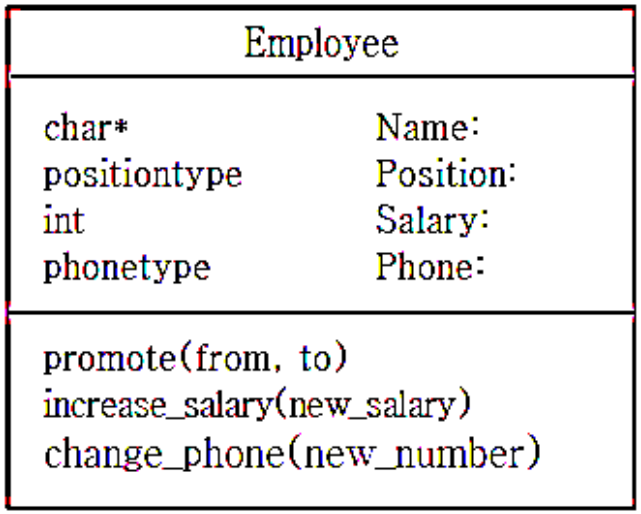


그림 5.3 ▶ 클래스 개념을 사용한 그루핑

객체와 속성

- 객체 : 속성과 오퍼레이션을 가진 애플리케이션의 독립된 존재
- 속성 : 객체의 특징을 결정

- 객체의 구조
  - 소프트웨어 모듈(객체) = 자료구조 + 함수

- 객체는 상태(state), 능력(behavior), 정체성(identity)을 가짐
  - 상태
  - 능력 : 연산(operation)을 수행 할 수 있는 능력
  - 정체성 : 구별 가능성

캡슐화(Encapsulation)

- 캡슐화의 정의
  - 속성과 관련된 오퍼레이션을 클래스 안에 묶어서 하나로 취급하는 것
    - 예> 대학 학사 관리 시스템
    - 데이터 : 학번, 이름, 주소 캡슐화
    - 함수 : 평점 계산, 주소 변경, 수강 신청 캡슐화

```
class Employee {
public:
    promote(from, to);
    increase_salary(new_salary);
    change_phone(new_number);
    ...
private:
    char*          name;
    positiontype   position;
    int            salary;
    phonetype      phone;
    ...
}
```

- 추상화의 수단
  - 객체의 속성, 오퍼레이션 등의 세부사항은 차후에 생각

➔ 디테일, 불필요한 것은 숨겨서 ➔ 복잡도를 줄임

- 정보은닉(information hiding)
  - 캡슐 속에 있는 항목에 대한 정보를 외부에 감추는 것
  - 외부의 직접적 접근 불가, 일종의 블랙박스

- 구현에 따라 선택 가능
  - 문법 : public, private, protected 참고: default in Java

A class 아닌 클래스로서,
A의 subclass도 아닌 경우

A Class내 멤버
의 접근 한정자

Modifier	A Class	Subclass	World
public	✓	✓	✓
protected	✓	✓	✗
private	✓	✗	✗
default	A와 같은 Package 내 class (in Java)		

연관(association)

- 객체는 일반적으로 상호작용하여 동작
  - 객체에 있는 서비스를 호출하면 두 객체는 관계가 맺어져야 함
  - 상호작용할 필요가 있는지 찾아내는 작업이 필요

- 연관
  - 하나 또는 그 이상의 클래스와의 관계
  - 예> 은행시스템과 학사업무 시스템

Own		Teach-Enrolled-by		
Customer	Account	Student	Course	Professor
홍길동	자유저축1	홍길동	자료구조	이금희
홍길동	정기예금1	홍길동	객체지향 설계	박영희
김동국	자유저축2	김동국	객체지향 설계	박영희
이철수	자유저축3	이철수	소프트웨어공학	최은만
한국남	정기예금2	한국남	소프트웨어공학	최은만

이진관계, 1: N 다중도 그림 5.4 ▶ 연관의 사례 3항 (ternary)관계