학습 목표

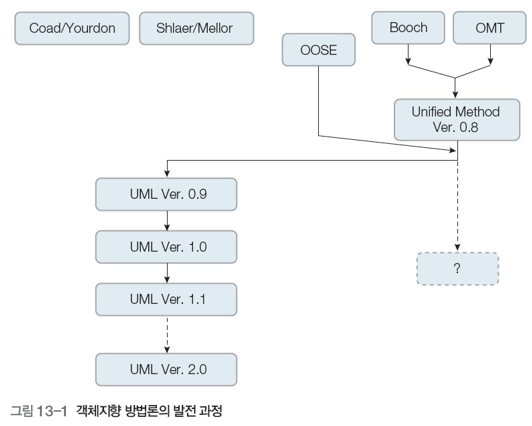
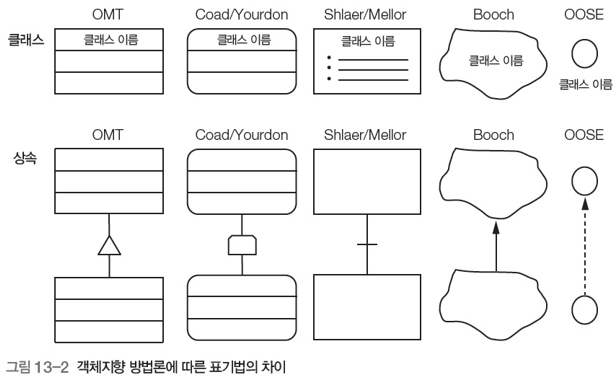
객체지향 방법론의 탄생과 발전에 관한 개략적인 역사를 이해한다.

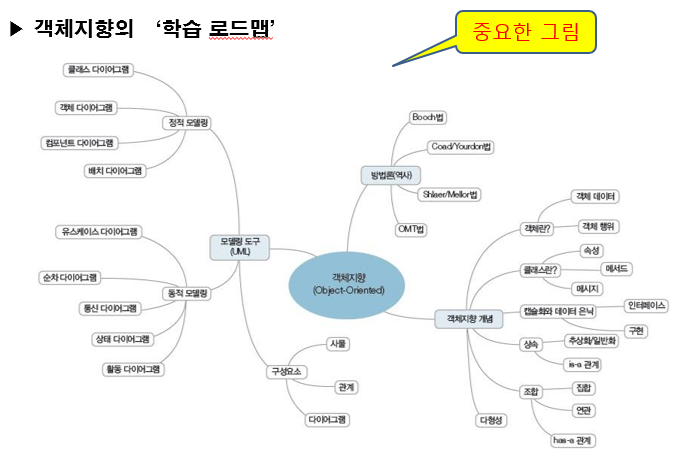
객체지향 방법론이 전통적인 방법론과 어떤 차이점을 갖고 있는지 알아본다.

객체지향 방법론의 핵심적인 개념을 이해한다.

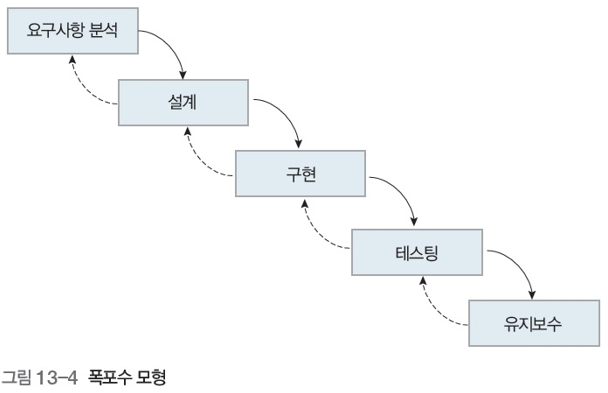
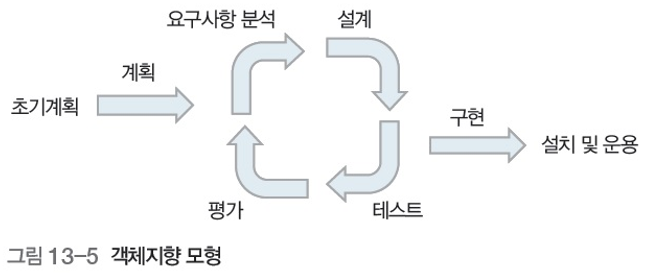
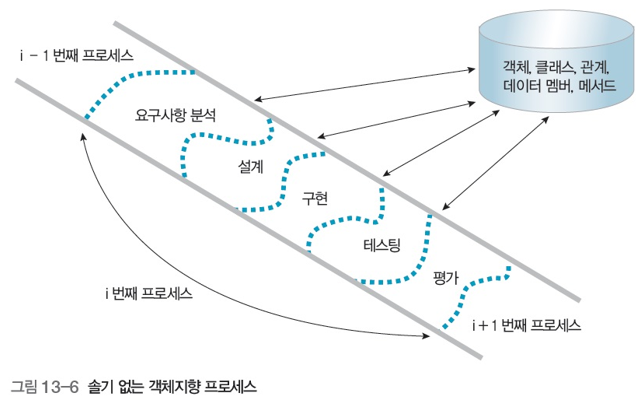
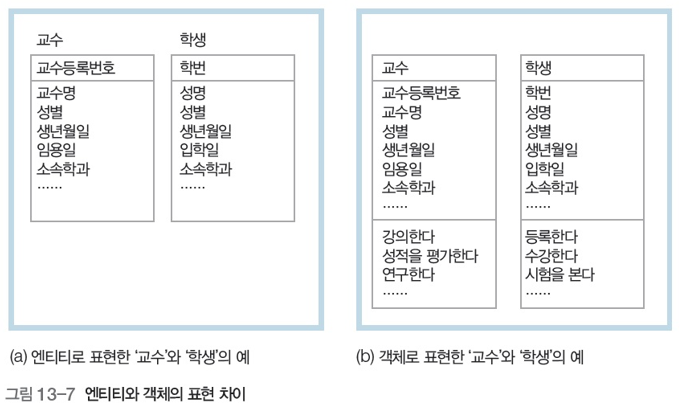
객체지향 방법론의 광범위한 활용 현황과 미래에 대해 생각해 본다.

**13.1 객체지향 방법론의 이해**

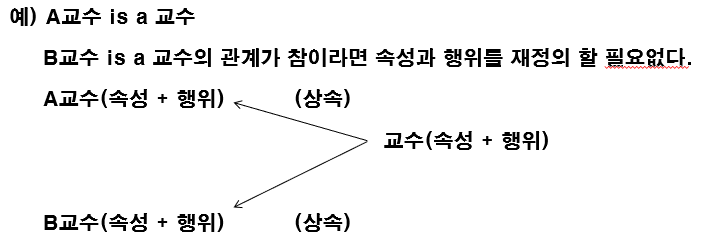
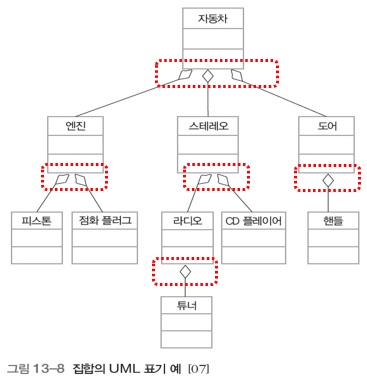
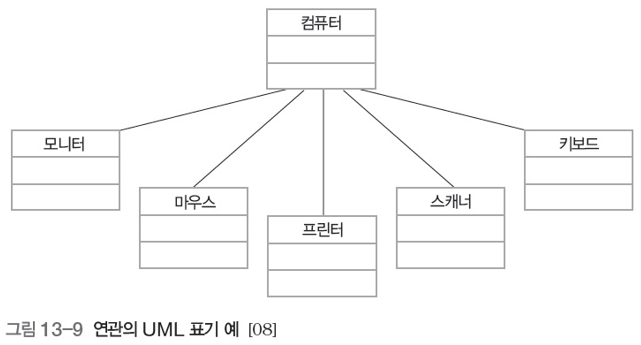
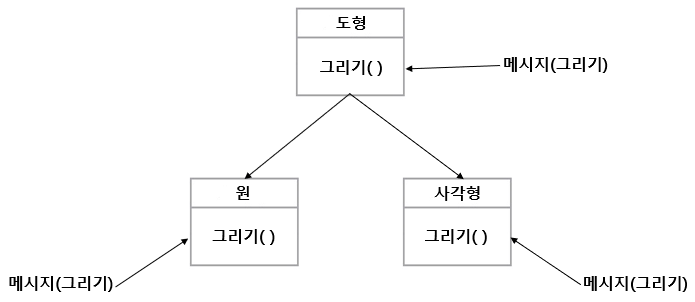
1. 객체지향 방법론의 역사
   * 객체지향 입문 중
     1. 객체지향이라는 개념은 객체지향 언어의 탄생에서부터 시작
     2. 최초의 객체지향 언어는 1967년 노르웨이에서 개발된 시뮬라(Simula)
   * 4대 방법론
     1. 1980년대 후반부터 1990년대 초반까지 객체지향 설계 방법론
     2. 부치(G.Booch>가 제안한 Booch법
     3. 코드(P.Coad)와 요돈(E.Yourdon)이 제안한 Coad.Yourdon법
     4. 슐레이어(S.Shlaer)와 멜러(S.Mellor)가 제안한 Shlaer/Mellor법
     5. 럼보(J.Rumbaugh)가 제안한 OMT(Object-Modeling-Technique)법
   * 객체지향 방법론의 발전 과정  
     
   * 객체지향 방법론 표기법의 차이가 문제로 대두
     1. 레셔널사는 표기법만이라도 통일하자는 취지로 UML을 제안  
        
2. 객체지향의 숲에서 길 찾기
   * 객체지향의 ‘학습 로드맵’
     1. 방법론(역사): 통일된 표기법이 UML(래셔널사)
     2. 객체지향 개념: 객체, 클래스, 캡슐화, 데이터 은닉, 상속, 조합, 다형성 등
     3. 모델링 도구 (14장에서 학습 예정)
        1. 정적 모델링: 클래스 다이어그램(Class Diagram)
        2. 동적 모델링: 유스케이스 다이어그램(Usecase Diagram), 시퀀스 다이어그램(Sequence Diagram) 등



**13.2 객체지향 방법론의 핵심 개념**

1. 객체지향 프로세스의 특성
   * 소프트웨어 공학이 추구하는 목적
     1. 성공적인 소프트웨어 제품을 만들자. (고객의 요구사항 충족)
     2. 성공적인 프로세스를 달성하자. (프로젝트의 수행 성공)
   * 객체지향 방법론의 주요 특성
     1. 반복적인(Iterative) 프로세스
     2. 솔기 없는(Seamless) 프로세스
     3. 상향식(Bottom Up) 접근 방식
     4. 재사용(Reuse)을 고려
   * 반복적인 프로세스
     1. 구조적 방법론에 따른 소프트웨어 개발 프로세스를 폭포수 모형이라 부름  
        
     2. 객체지향 방법론은 소프트웨어 생명주기를 반복하여 적용하도록 제안  
        
   * 솔기 없는 프로세스
     1. 프로세스를 구성하는 각 단계 간의 경계선이 불분명하다는 것을 뜻함  
        
   * 상향식 프로세스
     1. 구조적 방법론: 하향식(Top Down) 프로세스를 사용
     2. 객체지향 방법론: 상향식 프로세스를 사용 (작은 덩어리의 문제를 해결한 후 큰 덩어리로 해결)  
        최소단위 부속객체 구현 -> 조합하여 큰 객체 구현 방식
   * 재사용에 대한 고려
     1. 구조적 방법론: 성공적인 소프트웨어 산물을 만들고자 하는 개발 공정에 치중
     2. 객체지향 방법론: 재사용이 고려되어 프로세스가 진행  
        부 시스템 구현을 위해 기존 클래스를 재사용  
        고려사항은 구현단계 이후 Refactoring이 중요.
2. 객체지향의 핵심 개념
   * 객체: 엔티티의 속성에 해당하는 객체 데이터 뿐만 아니라 객체 행위를 포함  
     

객체 = 객체 데이터(속성) + 객체 행위  
Object = Object Data + Object Behavior

* + 클래스: 공통 속성과 행위를 가진 기존 객체를 묶어 추상화한 개념  
    즉, 클래스 = 객체를 추상화  
    객체 = 클래스를 추상화
  + 상속: 객체를 생성할 때 ‘상속하여’ 사용하면 시스템의 신뢰성과 재사용성을 높일 수 있어 효율적임.  
    
  + 상속 캡슐화와 데이터 은닉의 개념
    1. 하드웨어 제품이 장애가 났을 때 필요한 부품만 교체하여 복구하는 것처럼, 소프트웨어도 장애가 발생한 모듈만 교체하자는 아이디어에서 출발
  + 데이터 은닉
    1. 각각의 객체는 자신의 속성(데이터)과 메서드(행위)를 다른 객체에게 숨김
    2. 인터페이스: 플러그 외부에 노출된 부분
    3. 구현: 내부에 숨겨진 부분
  + 캡슐화
    1. 구현부가 외부에 노출되지 않도록 싸여진 상태
  + 상속
    1. 다른 클래스로부터 속성과 행위를 상속받는 것
    2. 부모/자식 관계가 성립
    3. 상속은 ‘is-a 관계’ 예) 개는 포유류이다.
  + 조합
    1. 다른 객체를 사용하여 객체를 구성
    2. 다른 클래스를 사용하여 보다 복잡한 클래스를 만드는 일종의 조립
    3. 조합은 ‘has-a 관계’ 예) 자동차의 엔진은 실린더 블록을 갖는다.
    4. 조합은 집합과 연관의 두 가지 유형이 있음
  + 집합: 복잡한 객체가 다른 객체들로 구성됨.
    1. 사례) 자동차  
       
  + 연관: 하나의 객체가 서비스를 위해 다른 객체를 원한다
    1. 사례) 컴퓨터 시스템처럼 클래스 간에 제공하는 서비스일 뿐이다.  
       
  + 다형성(여러 형태를 의미)
    1. 동일한 메서드(그림에서는 그리기)에 각 객체별로 별도의 행위를 할 수 있음  
       

**13.3 객체지향 방법론의 과거, 현재 및 미래**

1. 객체지향 언어의 출현과 발전
   * 객체지향 언어의 시초
     1. 1960년 노위지안 컴퓨터 센터의 조한 달과 크리스틴이 발표한 시뮬라 67
   * 스몰토크
     1. 객체지향 언어로서 실질적 원조
     2. Xerox의 팰러앨토 연구소에서 앨런 케이의 책임 하에 만들어진 스몰 토크
   * 에이다(Ada)
     1. 1980년대 초 객체지향 언어로 미 국방성에서 개발
   * C++
     1. AT&T의 벨 연구소에서 비야네 스트로스트툽 등에 의해 개발
     2. 가장 많은 사용자를 확보하고 있는 객체지향 언어
   * 자바
     1. 1990년대 중반 이후로 각광받고 있는 객체지향 언어
     2. 썬 마이크로시스템즈의 제임스 고슬링에 의하여 고안된 언어
   * 오브젝티브-C
     1. 브래드 콕스가 개발
     2. C언어와 객체지향 개념을 혼합한 언어
   * DCOM, CORBA
     1. 객체지향 기술을 인터넷에서 활용하기 위해 발전한 분야가 ‘분산객체 기술’
     2. 마이크로소프트의 윈도우 플랫폼을 기반으로 한 DCOM
     3. 800여 업체가 컨소시엄을 결성해 사양을 확정한 CORBA
2. 객체지향의 미래
   * 객체지향 입문의 저자 터커:  
     만약 가까운 미래에 어떤 것이 객체지향을 대신하더라도 지금까지 그랬던 것처럼 그것은 반드시 객체지향을 기반으로 할 것이다.
   * 현재 IT 환경은 인터넷과 모바일 네트워크 기반으로 형성
   * 사물 인터넷(IoT)이 더욱 발전할 전망