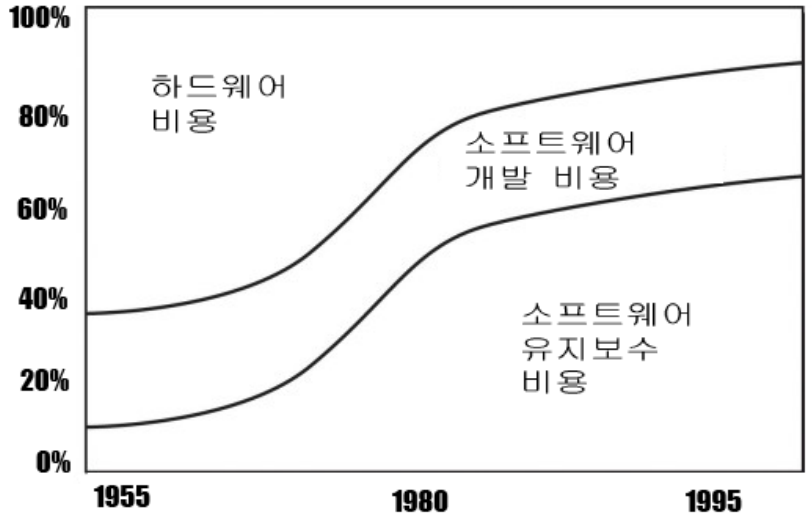
Object Oriented Software Engineering OOSE

객체지향 분석 기법

- 객체지향 분석 기법 개요
- 객체지향 분석 프로세스
- 정보(객체) 모델링
- 동적 모델링
- 기능 모델링
- 객체지향 설계 : 모델 통합

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

객체지향 분석 기법 개요



- 객체지향 개발 방법론
  - 소프트웨어 위기 극복을 위한 가장 최근의 개발방법론
  - 소프트웨어 개발 문제점을 해결해 줄 많은 장점 보유
  - 시스템은 요구사항 변경을 수용할 수 있어야 하고,
    이를 위해 유연성과 적응력을 갖도록 설계해야 하지만,
    기존 개발 방법의 시스템 확장 및 변경이 쉽지 않은 문제를 극복 불가
    => 새로운 방법 필요
  - 데이터와 행위를 하나로 묶어 객체를 정의 및 추상화 하는 작업
  - 기존의 데이터와 행위가 분리되었던 개발 방법의 복잡성과
    통합의 어려움 극복 노력 중

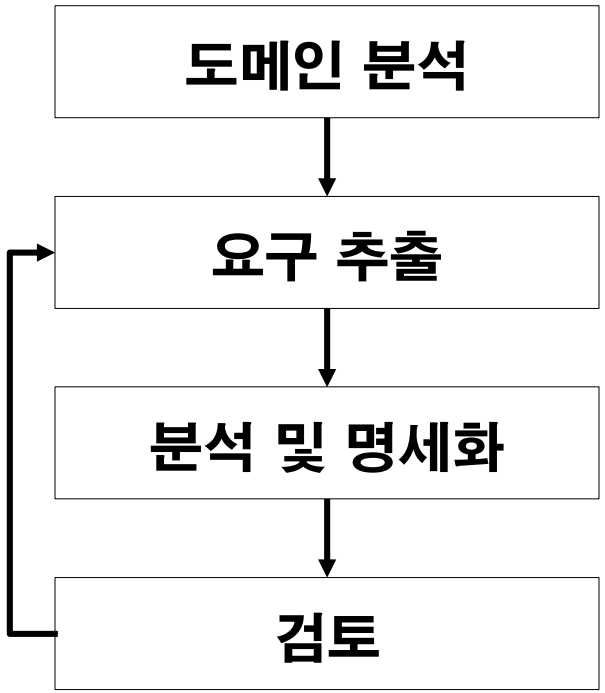
객체지향 분석 기법 개요

❑ 객체지향 개발 방법론의 절차와 단계별 작업 항목

개발 단계	작업 항목	결과물	설명
객체지향 분석	기능 모델링	Data Flow Diagram Activity Diagram	• 입력에 대한 처리 결과를 확인
	동적 모델링	Sequence Diagram State Diagram	• 객체 사이의 변화를 시간에 따라 조사 • 상태, 사건, 동작
	객체 모델링	Object Diagram Class Diagram	• 시스템의 정적구조 포착 • 추상화, 분류화, 일반화, 집단화
객체지향 설계	시스템 설계	Package Diagram Component Diagram Deployment Diagram	시스템의 구조를 설계 성능 최적화 및 자원 분배 방안 마련
	객체 설계	Object Diagram Class Diagram Sequence Diagram State Diagram	세부 자료구조와 알고리즘 구현
객체지향 구현	객체지향 언어	Source Code	캡슐화, 상속성, 다형성을 제공하는 프로그래밍 언어

객체지향 분석 기법 개요

- 일반적인 요구 분석 과정



문제의 배경, 성격, 범위 식별

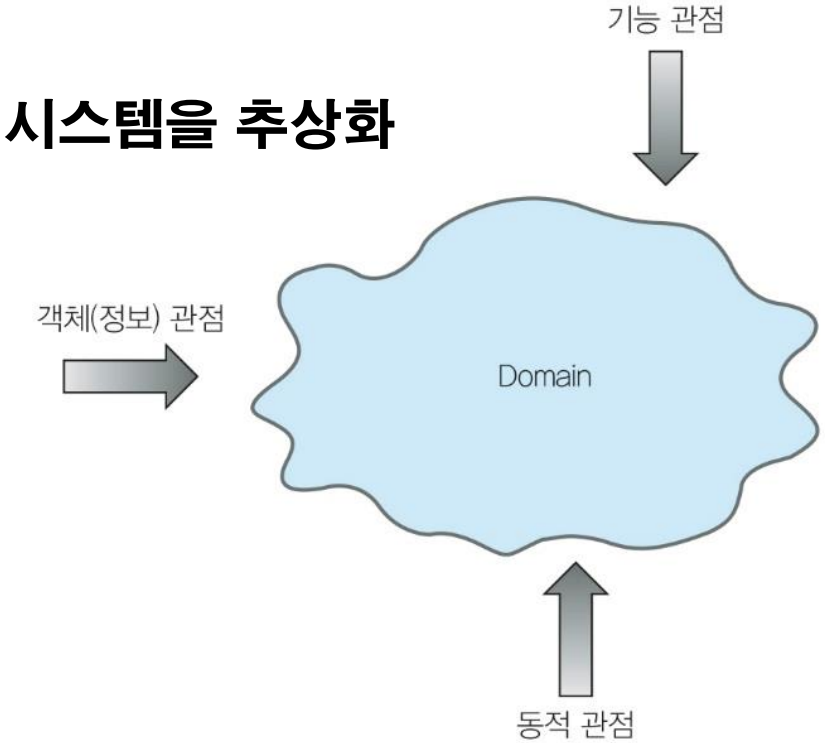
고객에게 무엇이 필요한지 식별
(기능요구, 비기능요구)

식별된 요구사항이 실현 가능한지 확인하고,
결정된 요구사항에 대해 문서로 정리

고객이 필요한 것이 맞는지 확인

객체지향 분석 기법 개요

- 객체지향 분석의 3가지 관점
  - 시스템을 모델링하기 위해 여러 관점에서 시스템을 추상화
  - 정보(객체) 관점 => 정보 모델
  - 동적 관점 => 동적 모델
  - 기능 관점 => 기능 모델



  - 객체지향 분석 기법
    = 기능 모델 + 동적 모델 + 정보 모델
    - 객체 중심의 상향식 접근방법 도입
    - 기능 중심이 아닌 정보 중심의 시스템 개발 방법
    - 분석과 설계의 표현에 큰 차이점이 없어 시스템 개발 용이

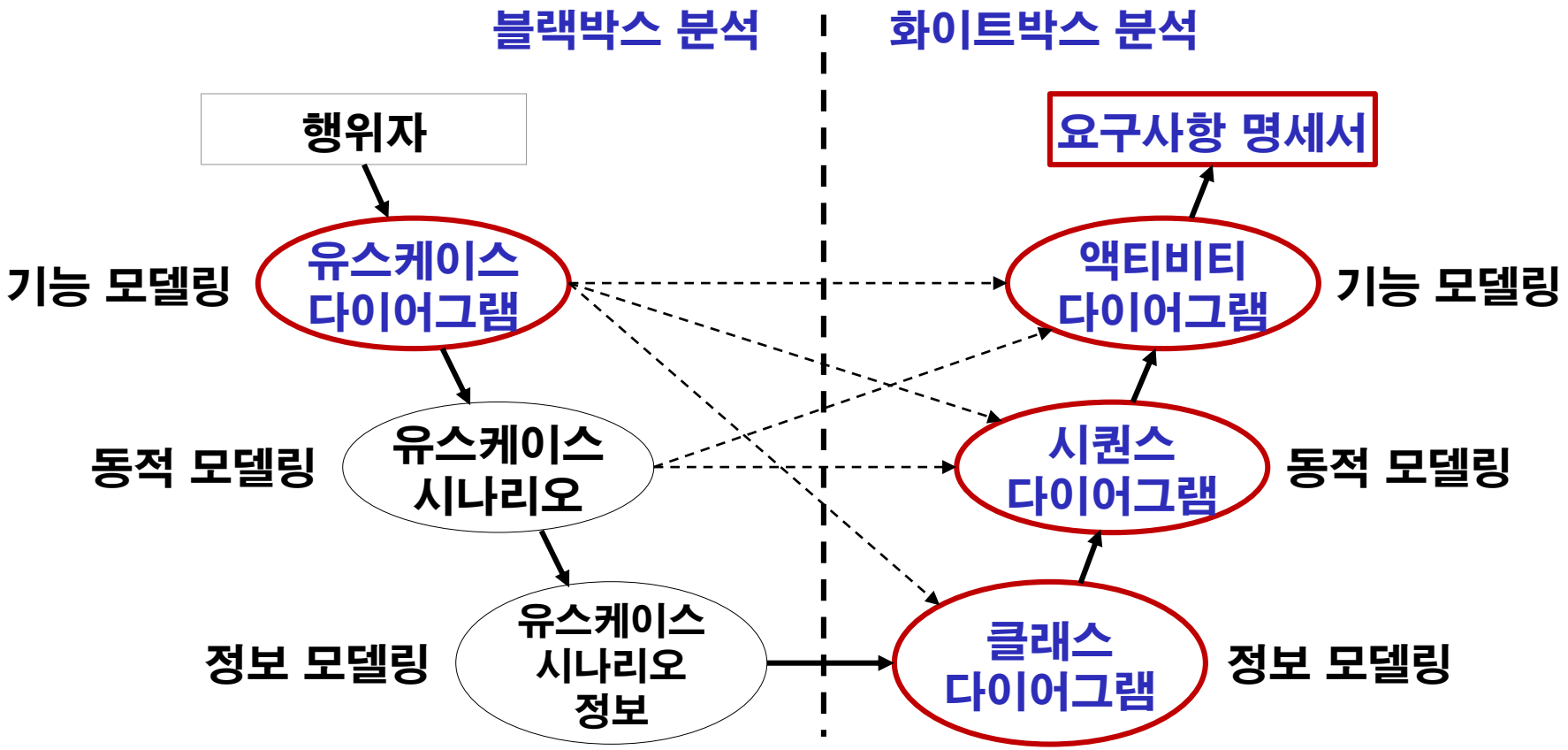
객체지향 분석 기법 개요

- 객체지향 분석 기법의 3가지 모델링
  - 정보(객체) 모델링 : 시스템에서 요구되는 객체(정보)를 먼저 찾아내어 객체들의 특성과 객체들 사이의 관계 규명
  - 동적 모델링 : 객체 모델링에서 규명된 객체들의 행위와 객체들의 상태를 포함하는 생명주기 보여줌
  - 기능 모델링 : 각 객체의 형태 변화에서 새로운 상태로 들어갔을 때 수행되는 동작 기술

  - 객체지향 개발 방법의 활용을 위해 기존 기술과는 다른 높은 차원의 기술력을 가진 분석 전문가 및 설계 전문가 필요
  - 소프트웨어 개발과정에 대한 이해와 정보 모델, 동적 모델, 기능 모델에 대한 지식이 필요하며, 모델 간 연관성을 바탕으로 모델링 결과의 통합 요구

객체지향 분석 프로세스

- UML V 프로세스 (1)
  - 행위자(Actor) : 사용자를 동질성 있는 집단으로 분류한 것



객체지향 분석 프로세스

❑ UML V 프로세스 (2)

[블랙박스 분석]

① 각 액터가 시스템을 사용하는 용도와 목적에 따라 유스케이스를 정의
*유스케이스(Use case) : 시스템이 가지는 최상위 수준의 기능
=> 유스케이스 다이어그램 작성

② 각 유스케이스에 대해 액터와 시스템 사이의 사건 흐름과 과정을 정의
=> 유스케이스 시나리오 작성

③ 각 유스케이스 시나리오에서 액터와 시스템 사이에 주고 받는 정보를 식별
=> 식별된 정보들은 시스템 내부에 저장되고 관리되어야 할 데이터들
=> 정보 모델링

객체지향 분석 프로세스

❑ UML V 프로세스 (3)

[화이트박스 분석]

④ 각 유스케이스 시나리오에서 찾은 정보를 활용하여 클래스 식별
클래스들 사이의 관계, 클래스의 속성 식별
=> 클래스 다이어그램 작성

⑤ 밝혀진 클래스를 바탕으로 시스템 내부의 객체들의 상호작용 과정 식별
유스케이스 시나리오를 내부로 확장하여 시스템을 화이트박스로 보고 수행
=> 시퀀스 다이어그램 작성

⑥ 시퀀스 다이어그램을 통해 밝혀진 객체들 사이의 이벤트에 대한
논리적 과정이 존재하거나, 업무 프로세스 내의 상호작용에 대한
추가적인 이해가 필요할 경우 => 액티비티 다이어그램 작성

정보(객체) 모델링

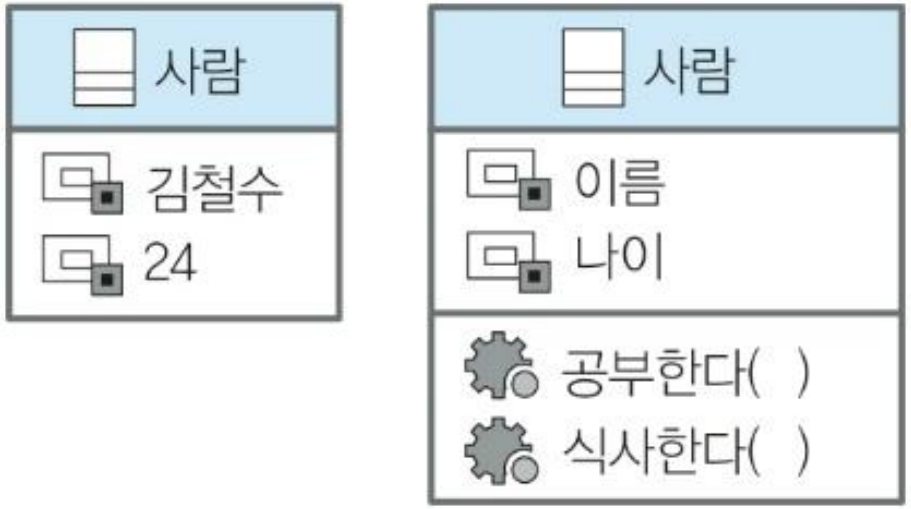
❑ 정보(객체) 모델링

- 객체지향 분석에서 가장 중요하며 선행되어야 할 모델링
- 시스템의 정적인 구조를 포착하는데 사용
- 문제 기술서, 유스케이스 시나리오로 부터 객체 추출
- 객체들 사이의 연관성 식별
- 객체들을 정의하기 위한 속성들 추가
- UML의 객체 모델링에서 사용되는 용어와 기호 숙지

•객체(Object) •클래스(Class) •속성(Attribute) •관계(Relationship) •동작(Operation)	•일반화(Generalization) •특수화(Specialization) •집단화(Aggregation) •상위클래스(Superclass) •하위클래스(Subclass)	•상속(Inheritance) •재사용(Reuse)

정보(객체) 모델링

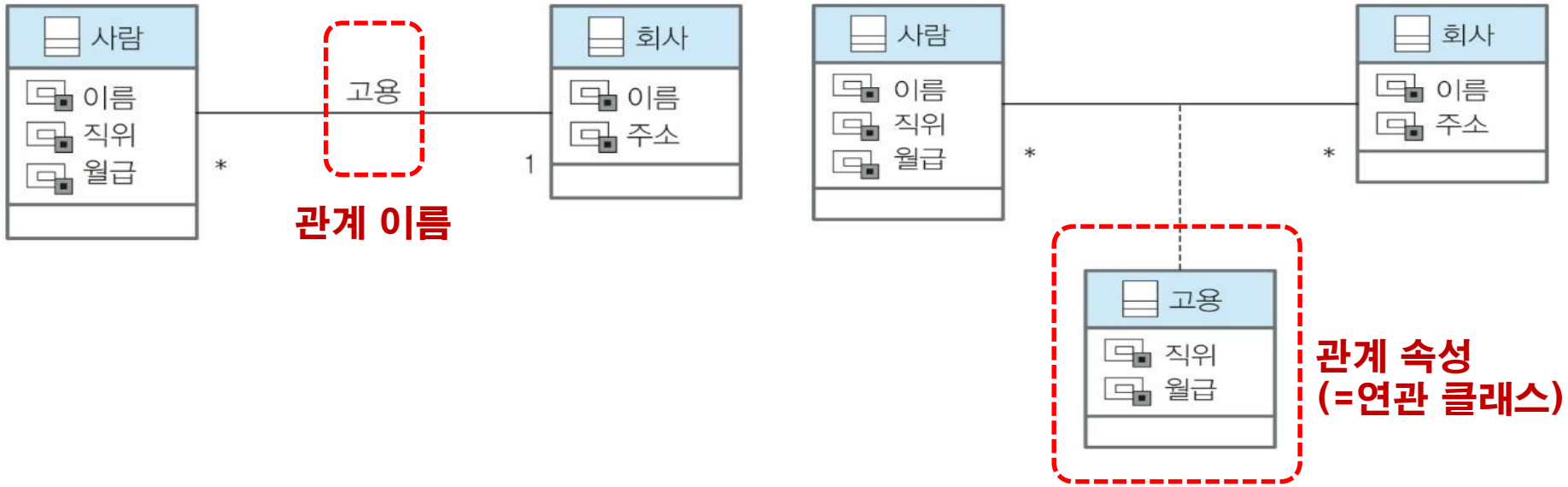
- 객체와 클래스
  - 객체(Object) : 객체 모델링의 기본 단위 (=엔티티)
  - 클래스(Class) : 유사한 객체들의 모임 (=엔티티 타입)
  - 표기법
    - 객체 : 사각형, 2칸, 객체의 이름과 속성값 표시
    - 클래스 : 사각형, 3칸, 클래스 이름, 객체(정보) 속성, 객체 동작 표시



정보(객체) 모델링

❑ 클래스의 관계 (1)

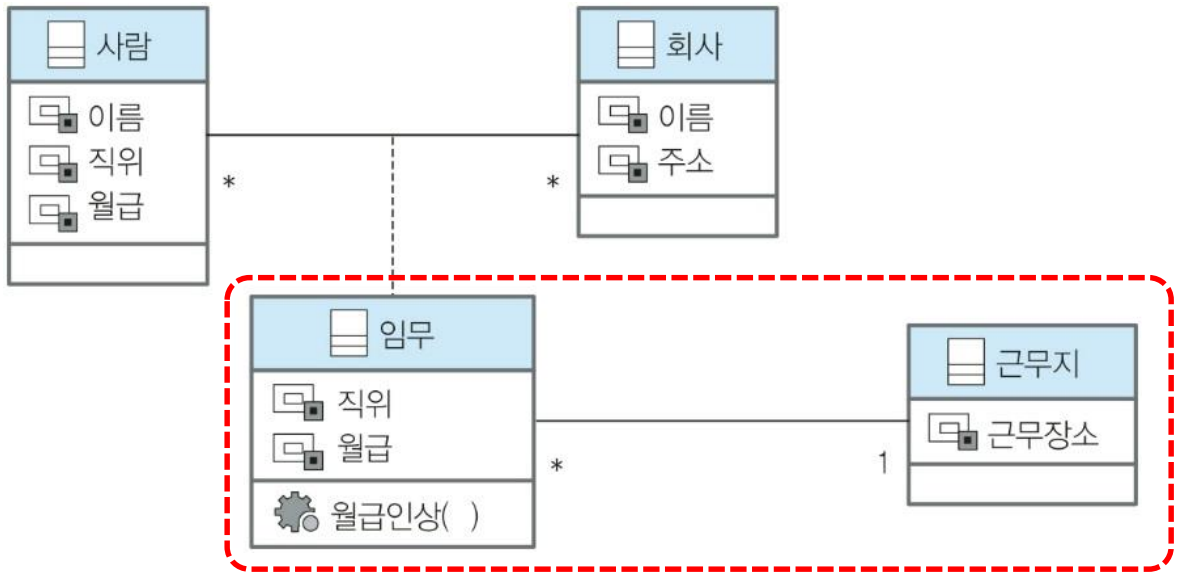
- 관계(Relationship) : 클래스들 사이의 연관성 정의
- 표기법
  - 관계 이름만 연결선 위에 표시
  - 관계가 새로운 속성이나 동작을 가질 경우, 관계 속성 표시



정보(객체) 모델링

❑ 클래스의 관계 (2)

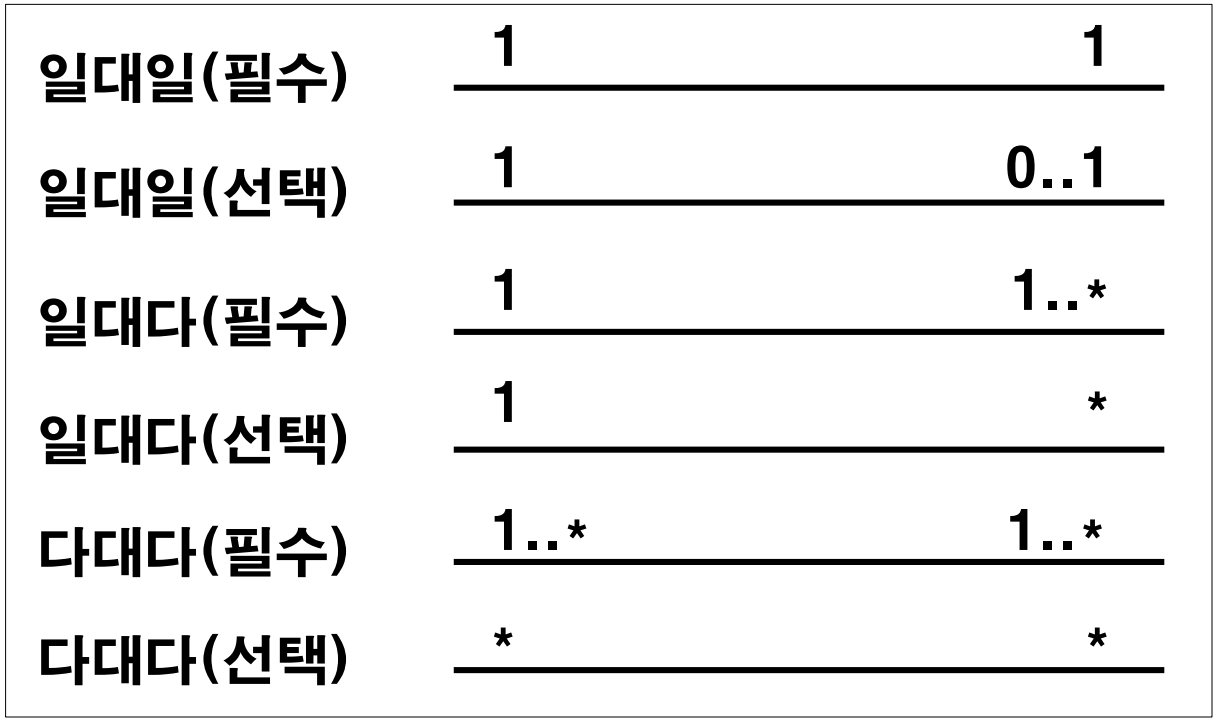
- 경우에 따라서 관계를 1개의 클래스로 모델링 하는 것이 효과적임
- 관계가 1개의 클래스로 정의되어 속성과 동작을 가지게 되면 다른 새로운 클래스와 연관 가능



정보(객체) 모델링

❑ 클래스의 관계 (3)

- 클래스 간 대응관계를 통해 객체들이 지켜야 할 제약조건 지정 가능
- 매핑 제약조건과 참여 제약조건은 정보 모델링과 같은 개념
- UML에서 매핑 수와 참여 제약조건 표시



정보(객체) 모델링

❑ 클래스의 관계 (4)

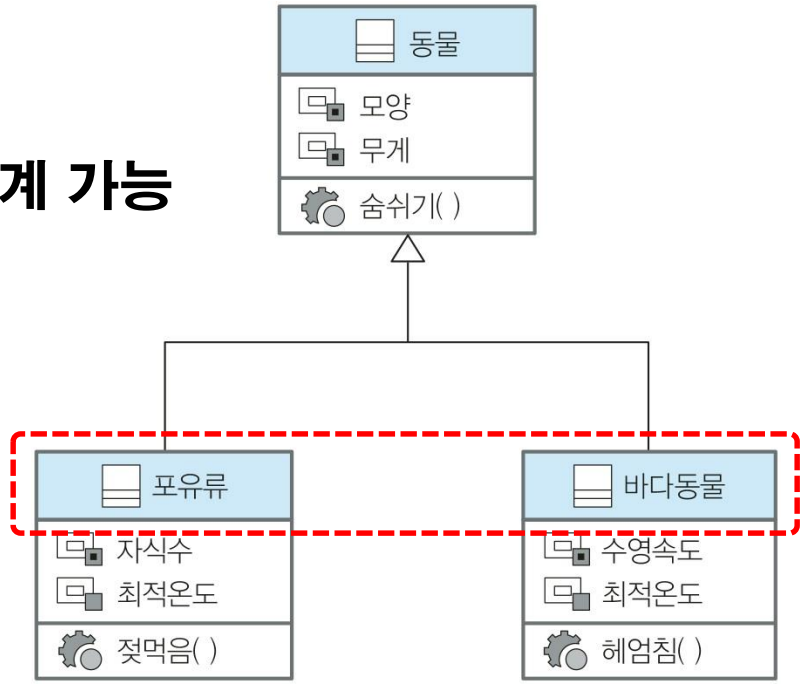
- 1 : 1 관계

- 1 : n 관계

정보(객체) 모델링

❑ 일반화(Generalization)

- 유사한 클래스 간 공유 속성과 동작을 묶어 주고,
  그들 사이의 다른 점을 보존할 수 있게 해 주는 효과적인 추상화 기법
- 공통의 정보는 오직 한 번 정의될 수 있어 분석 결과를 재사용 할 수 있어
  데이터의 무결성(integrity) 향상 가능
- 'is_a' 또는 'kind_of' 관계
- 클래스 간 계층구조가 형성되어 상속 관계 가능
  - 하위 클래스는 모든 상위 클래스의
    속성과 동작을 상속
  - 다른 클래스와의 연관성도 상속

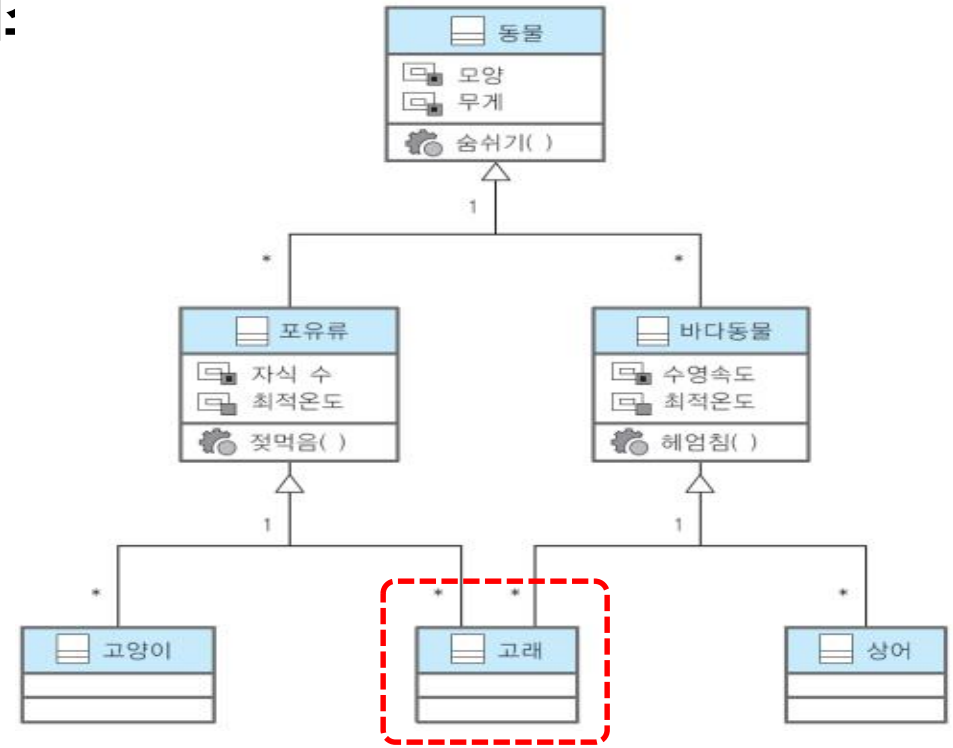


정보(객체) 모델링

- 특수화(Specialization)
  - 일반화의 역작용
  - 하위 클래스로 정의되는 경우에 정의
    - 하위 클래스가 고유 속성이나 동작을 가지고 있거나,
    - 하위 클래스가 다른 클래스들과 고유 관계를 가지고 있을 때

정보(객체) 모델링

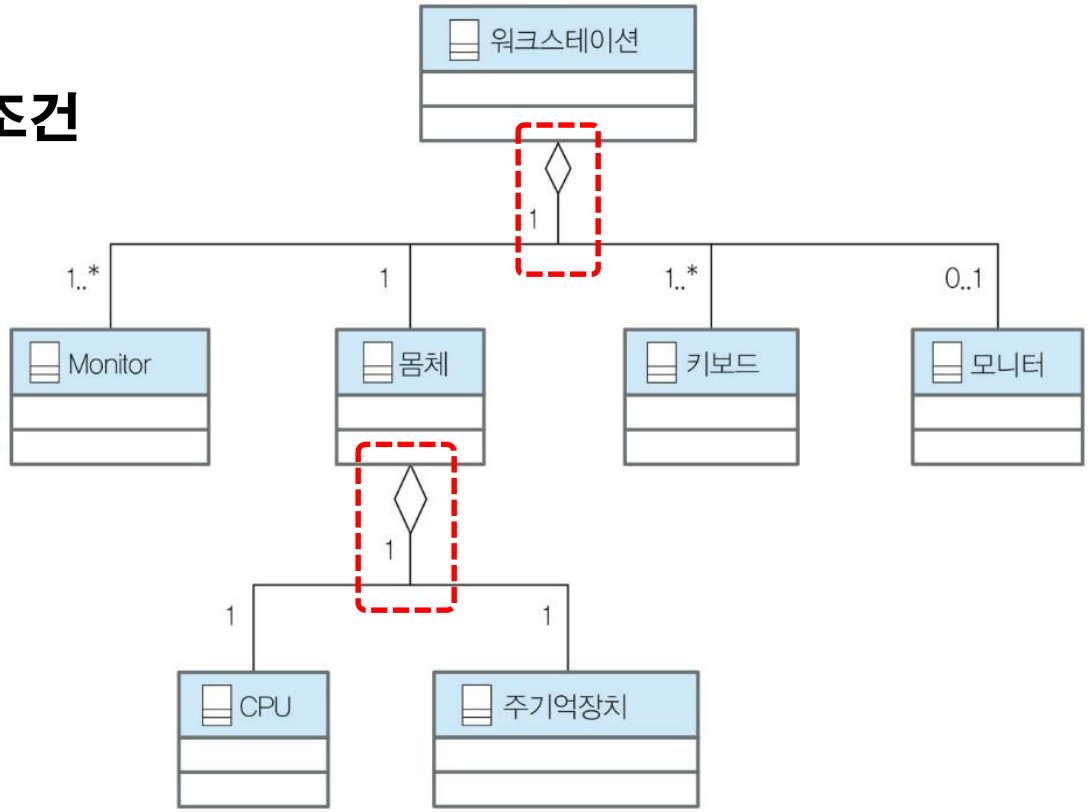
- 다중 상속
  - 어떤 클래스의 상위 클래스가 2개 있을 때 문제 발생 가능
  - 하위 클래스는 2개 이상의 상위 클래:
    동시에 속성과 동작을 상속
  - 분석 과정에서
    속성과 동작에서 다중상속으로 인한
    모순이 발견되면,
    모호성 제거를 위해
    요구사항 명세서나 자료사전에 기록



정보(객체) 모델링

❑ 집단화(Aggregation)

- 클래스 간에 "전체(is a whole)" 혹은 "부분(is a part of)" 관계로 설명
- 여러 부속 객체가 모여 하나의 객체가 구성되는 것
- 기호 : 작은 마름모로 표시
- 매핑 제약조건과 참여 제약조건 표시는 일반화와 동일



정보(객체) 모델링

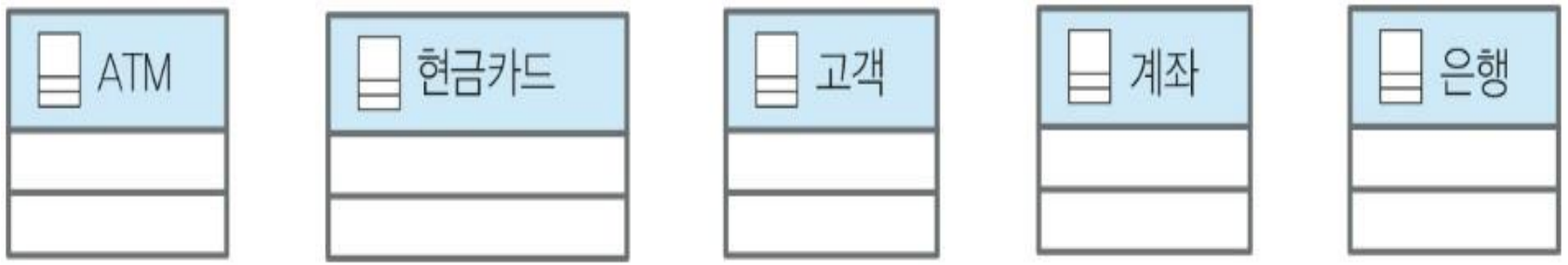
❑ 현금 자동 지급기(ATM)의 예 (1)

▪ 문제기술서의 일부 내용

> ATM은 은행 직원의 도움 없이 현금을 찾을 수 있게 해주는 장치이다. ATM은 현금카드를 받아들여 고객이 가지고 있는 계좌에서 현금을 지급하고 영수증을 출력한다. 은행은 고객의 계좌를 관리하며 ATM은 은행에 소속되어 있다.

① 문제기술에서 요구되는 객체 추출

- ATM 시스템 클래스 추출 결과

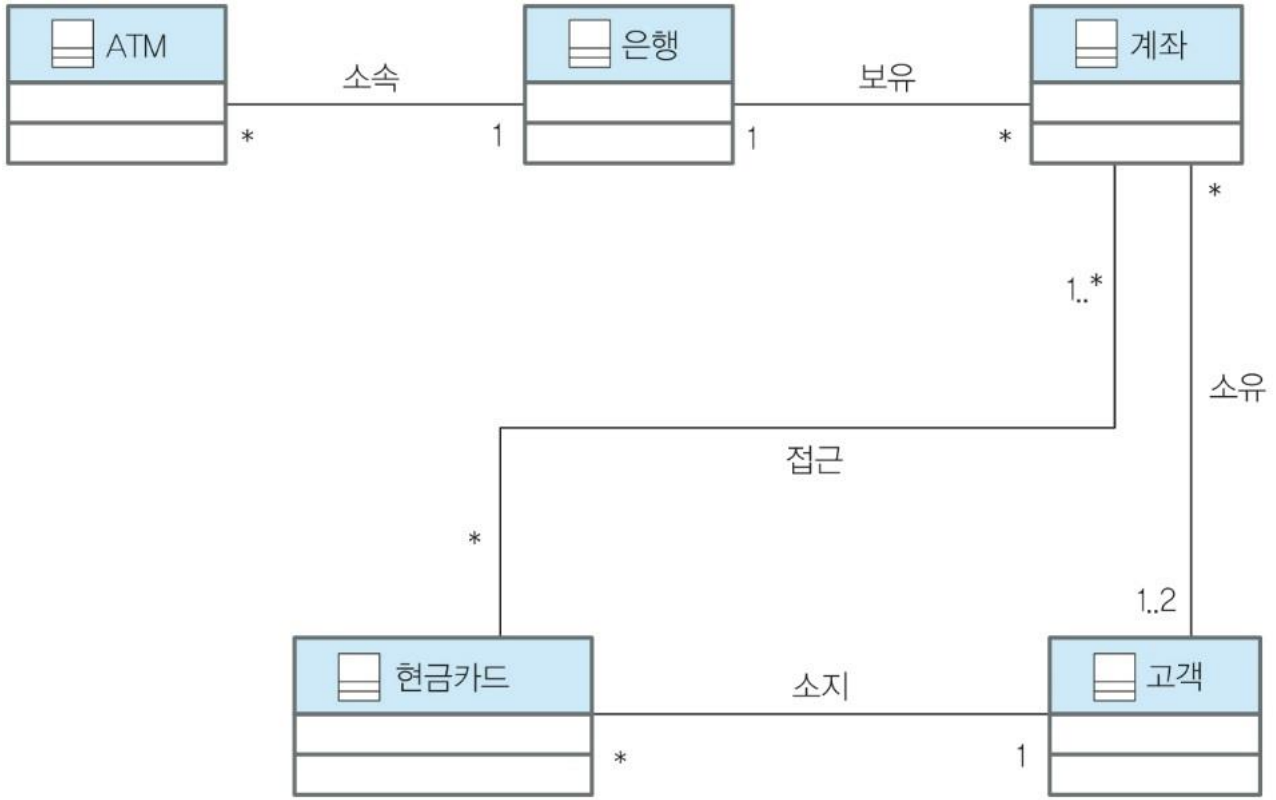


정보(객체) 모델링

❑ 현금 자동 지급기(ATM)의 예 (2)

② 문제기술서에 상세한 사항이 없지만 사용자(은행 담당자)와 상의하여 연관성과 구체적인 요구사항 식별

- ATM 시스템 클래스 간의 관계 식별 결과

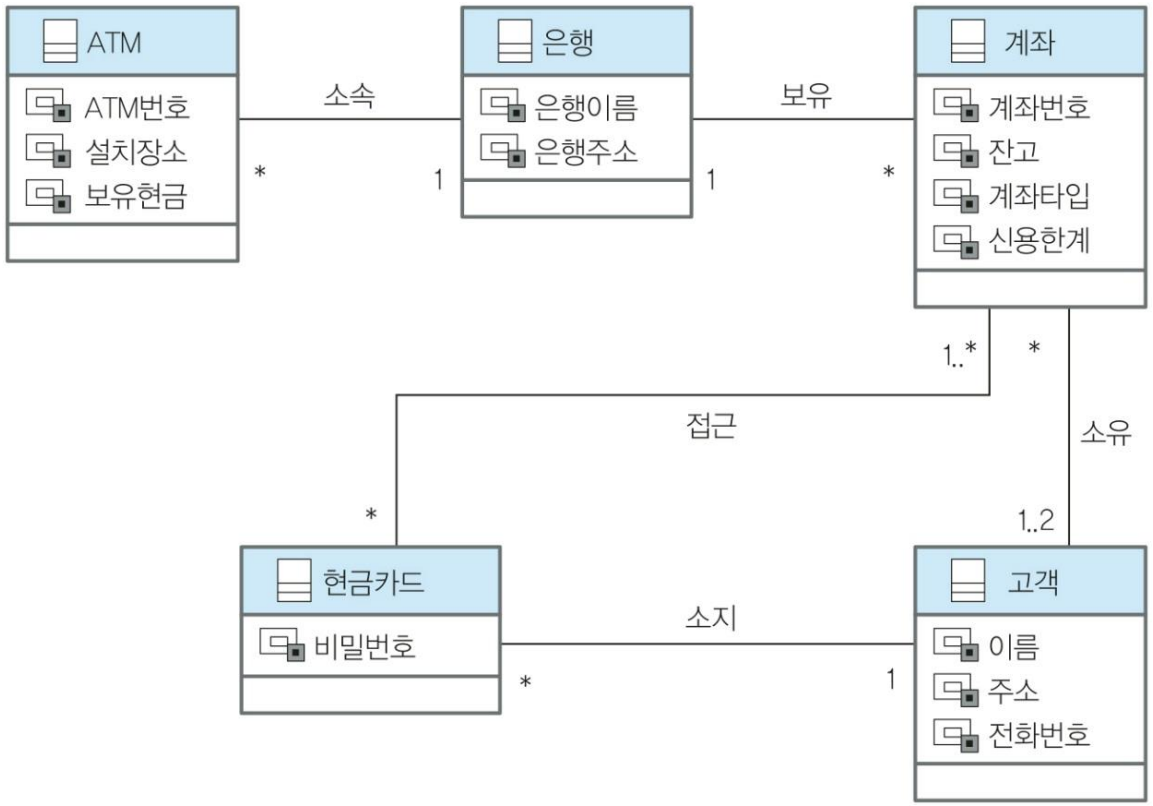


정보(객체) 모델링

❑ 현금 자동 지급기(ATM)의 예 (3)

③ 클래스 간의 연관성을 밝혀낸 다음 속성과 동작 식별

- 객체 모델링에서는 속성에 우선 초점을 맞추어 식별
- ATM 시스템 클래스의 속성 식별 결과



동적 모델링

- 동적 모델링(Dynamic Modeling)
  - 시간 흐름에 따른 객체들과 객체들 사이의 변화 조사
  - 객체들의 동적인 면을 식별하기 위함
  - 객체 간 제어 흐름, 상호작용, 동작 순서 등 식별
  - 중요한 개념 : 상태(state), 사건(event), 동작(operation)
  - 유스케이스 시나리오는 객체 또는 시스템의 실행 과정을 사건 흐름으로 표시
  - 각 사건은 객체 간의 정보 흐름을 나타내게 되고,
    각 사건의 정보를 보내는 객체와 받는 객체가 밝혀짐
  - 사건의 흐름은 시퀀스 다이어그램(Sequence Diagram)으로 구체화 됨
  - 상태의 변화는 상태변화도(State Transition Diagram)으로 구체화 됨
  - 동작은 클래스 내부의 메서드(Method)로 구체화 됨

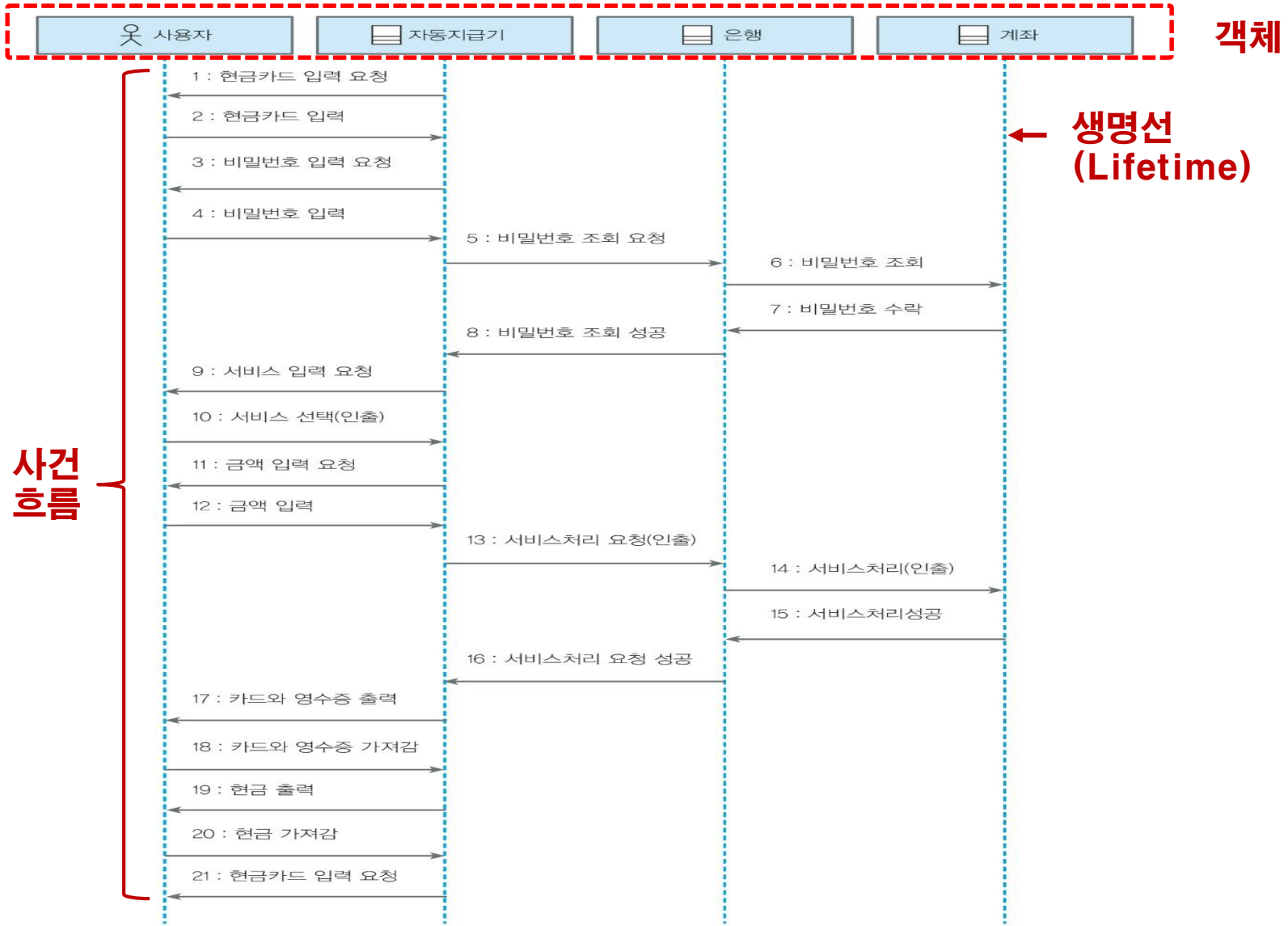
동적 모델링

❑ ATM의 [현금카드로 현금을 인출 한다] 유스케이스 시나리오 예

- 자동 지급기가 현금 카드를 입력할 것을 요구한다.
- 사용자가 현금 카드를 자동 지급기의 카드 입구에 넣는다.
- 자동 지급기는 현금 카드로부터 계좌 번호와 카드 번호를 읽고 사용 자에게 비밀 번호를 요구한다.
- 사용자가 비밀번호를 입력한다.
- 자동 지급기는 현금 카드 소속 은행에게 비밀 번호 대조를 요청한다.
- 은행은 현금 카드에게 비밀 번호 대조를 요청한다.
- 현금 카드는 은행에게 비밀 번호가 일치함을 알린다.
- 은행은 자동 지급기에게 비밀 번호가 일치함을 알린다.
- 자동 지급기는 사용자에게 가능한 서비스를 보여준다.
- 사용자가 현금 인출을 선택한다.
- 자동 지급기는 인출할 금액을 물어본다.
- 사용자가 인출할 금액을 입력한다.
- 자동 지급기는 해당 은행에게 인출할 금액 인출을 요구한다.
- 은행은 해당 계좌에게 인출할 금액 인출을 요구한다.
- 계좌는 잔액에서 인출할 금액을 인출하고 인출이 성공적으로 끝났음 을 은행에 알린다.
- 은행은 자동 지급기에게 현금 인출이 성공적으로 끝났음을 알린다.
- 자동 지급기는 사용자에게 카드와 영수증을 내어준다.
- 사용자가 카드와 영수증을 가져간다.
- 자동 지급기가 인출 금액을 내준다.
- 사용자가 인출 금액을 가져간다.
- 자동 지급기가 현금 카드를 입력할 것을 요구한다.

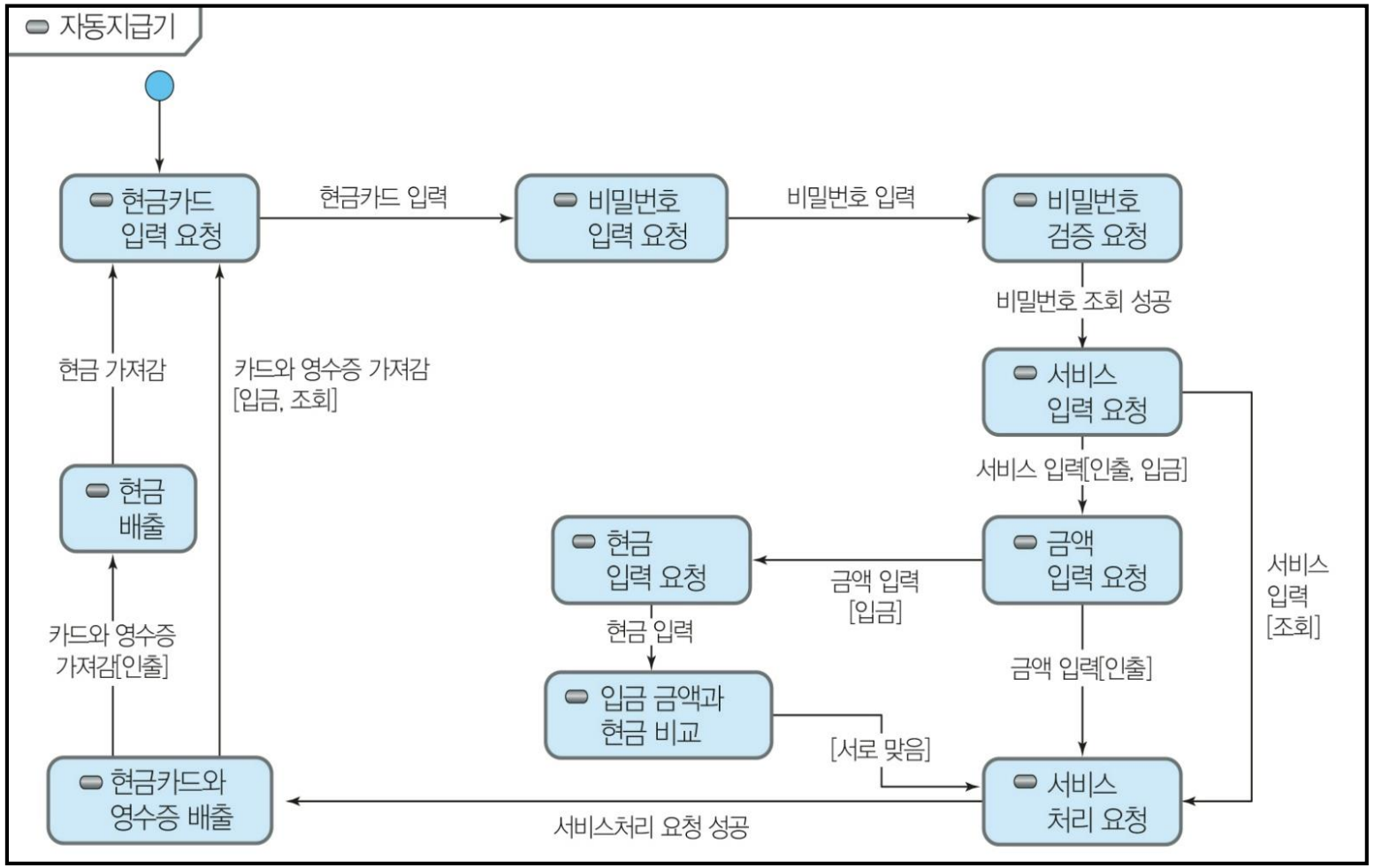
동적 모델링

- ATM의 [현금카드로 현금을 인출 한다]에 대한 시퀀스 다이어그램 예



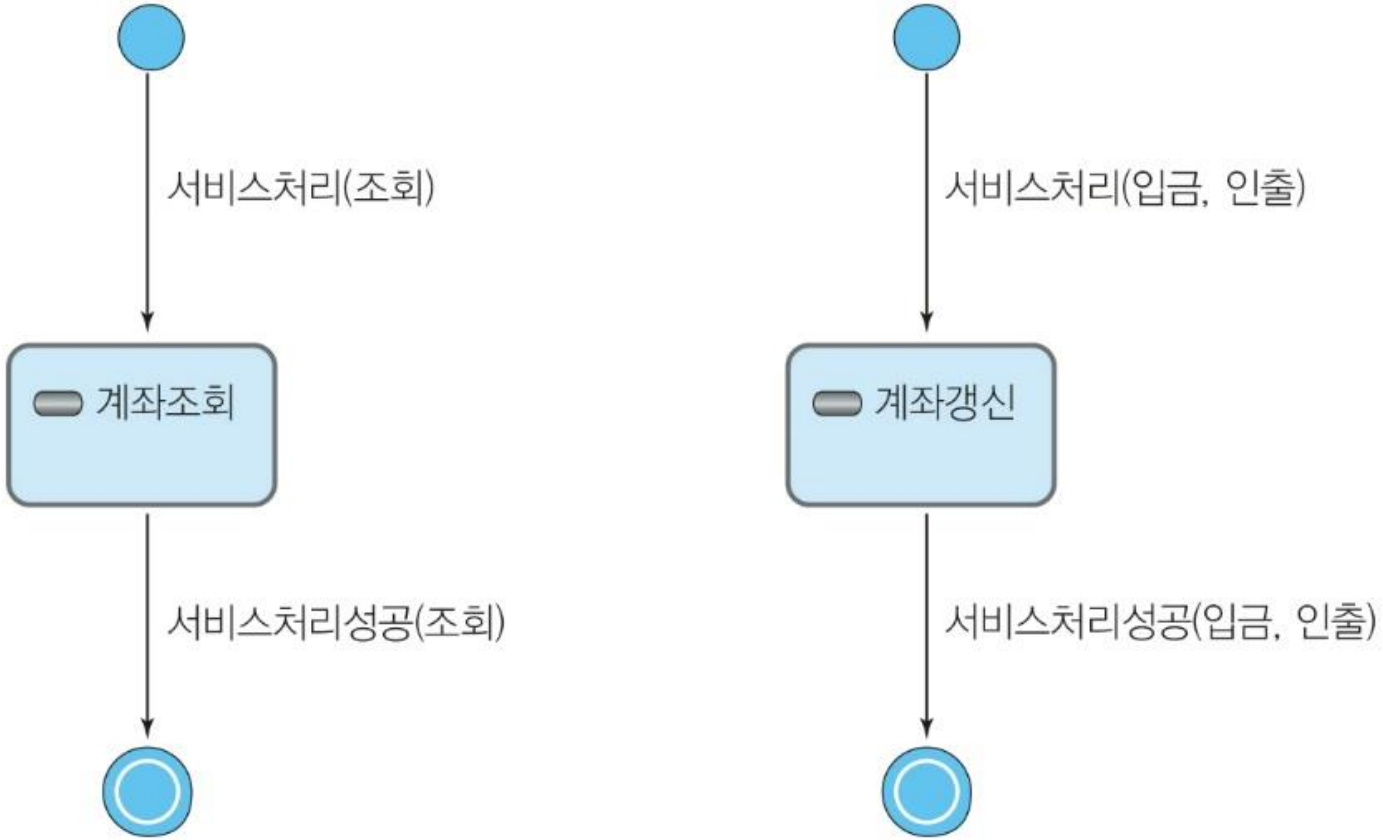
동적 모델링

- ATM의 [현금카드로 현금을 인출 한다]에 대한 상태변화도(STD) 예
  - "ATM"의 상태변화도(STD, State Transition Diagram)



동적 모델링

- ATM의 [현금카드로 현금을 인출 한다]에 대한 상태변화도(STD) 예
  - "계좌"의 상태변화도



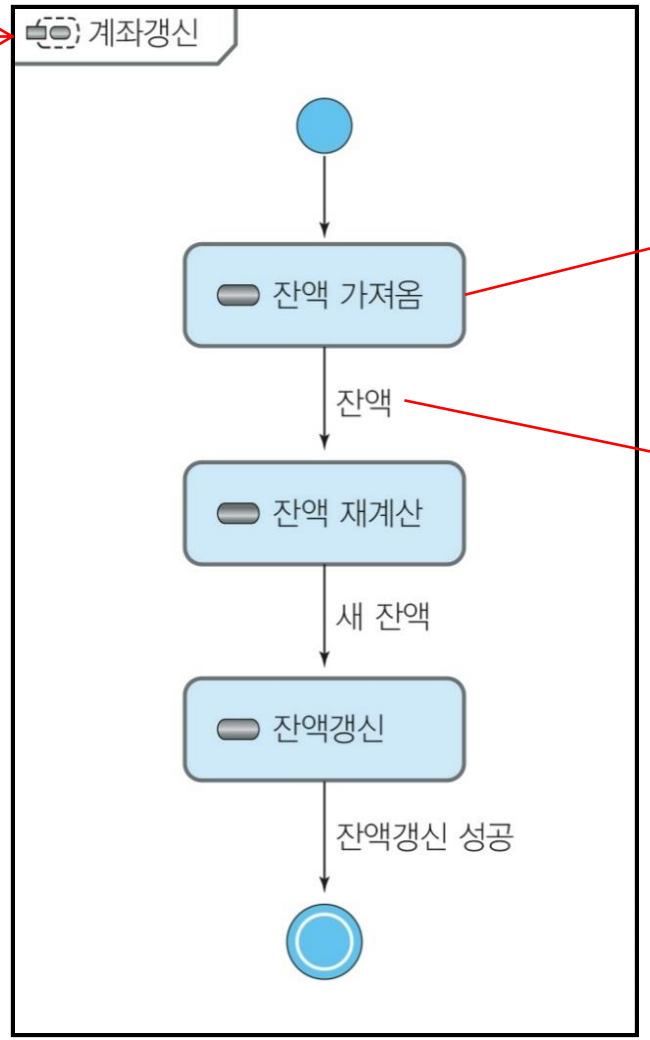
기능 모델링

- 기능 모델링(Functional Modeling)
  - 입력값부터 계산을 거쳐서 어떤 결과가 나타나는지 보여주는 것
  - 구현 방법은 고려하지 않음
  - 구조적 방법론과 같이 자료흐름도(DFD) 사용 가능
    - 입력 흐름은 프로세스를 통해 변환되어 출력 흐름으로 바뀌고, 다른 프로세스의 입력이나 외부로의 출력으로 작용
    - 동적 모델링에서 식별된 동작들이 어떠한 기능을 이루는지 보여주며, 자료흐름도의 정보 흐름들은 객체에 해당

기능 모델링

❑ 예> "계좌 갱신" 동작에 대한 액티비티 다이어그램

여러 개의 기능과
객체들로 이루어진 기능



동적 모델링에서 식별된 동작이며,
여러 개의 동작들이 모여서 하나의
기능을 이룸

정보(객체) 모델링에서
식별된 정보(객체)들

객체지향 설계 : 모델 통합

- 분석 단계에서 개발된 3가지 모델의 통합 작업
  - 객체의 정적 구조와 동작을 함께 포함하여 객체를 정의하는 것
  - 동적 모델에서의 사건, 동작 및 활동을
    ① 객체의 동작에 매핑하고,
    ② 기능 모델의 프로세스를 객체 모델의 동작에 통합시키는 것
  - 객체 수준의 상태변화도는
    - 한 객체가 생명주기 동안 가질 수 있는 상태를 기술한 것이며,
    - 상태의 변환은 객체 동작으로 매핑 되며,
    - 객체에 주어진 사건은 다른 객체의 동작으로 표현
  - 하나의 사건은 이전 사건에 의하여 초기화된 활동의 결과이며, 동작과 이에 반응하는 동작으로 매핑

객체지향 설계 : 모델 통합

- 정보(객체) 모델 + 동적 모델 + 기능 모델 통합 가이드 라인
  - 프로세스가 입력 흐름을 거쳐 새로운 결과를 만들어내는 경우
    => 입출력 흐름은 같은 객체이며 입력 흐름이 대상 객체
  - 프로세스가 여러 입력 자료 흐름으로부터 하나의 출력값을 생성한다면
    => 이 프로세스와 연관된 동작은 출력 클래스에 적용되는 동작으로 해석
  - 프로세스가 자료저장소나 외부 객체의 데이터를 읽거나 결과를 저장하는 경우
    - 자료저장소나 외부 객체가 이 동작의 대상 객체가 되며,
    - 이 결과는 객체에 속한 모든 가능한 정보, 객체들 사이의 관계나 객체의 동작들을 나타낼 수 있게 되어 시스템에 대한 완벽한 기술이 가능
  - 객체 간의 관계
    => 상대 객체를 나타내는 포인터 변수를 객체 속성으로 표현