객체지향소프트웨어공학

04주차-3 : 객체지향 개발방법론(Object-Oriented Development Methods)(2)

10장 객체지향 분석기법(Object-Oriented Analysis)

10.1 객체지향 분석 기법 개요
10.2 객체지향 분석 프로세스
10.3 정보 모델링
10.4 동적 모델링
10.5 기능 모델링

kit 국립금오공과대학교 Kumoh National Institute of Technology

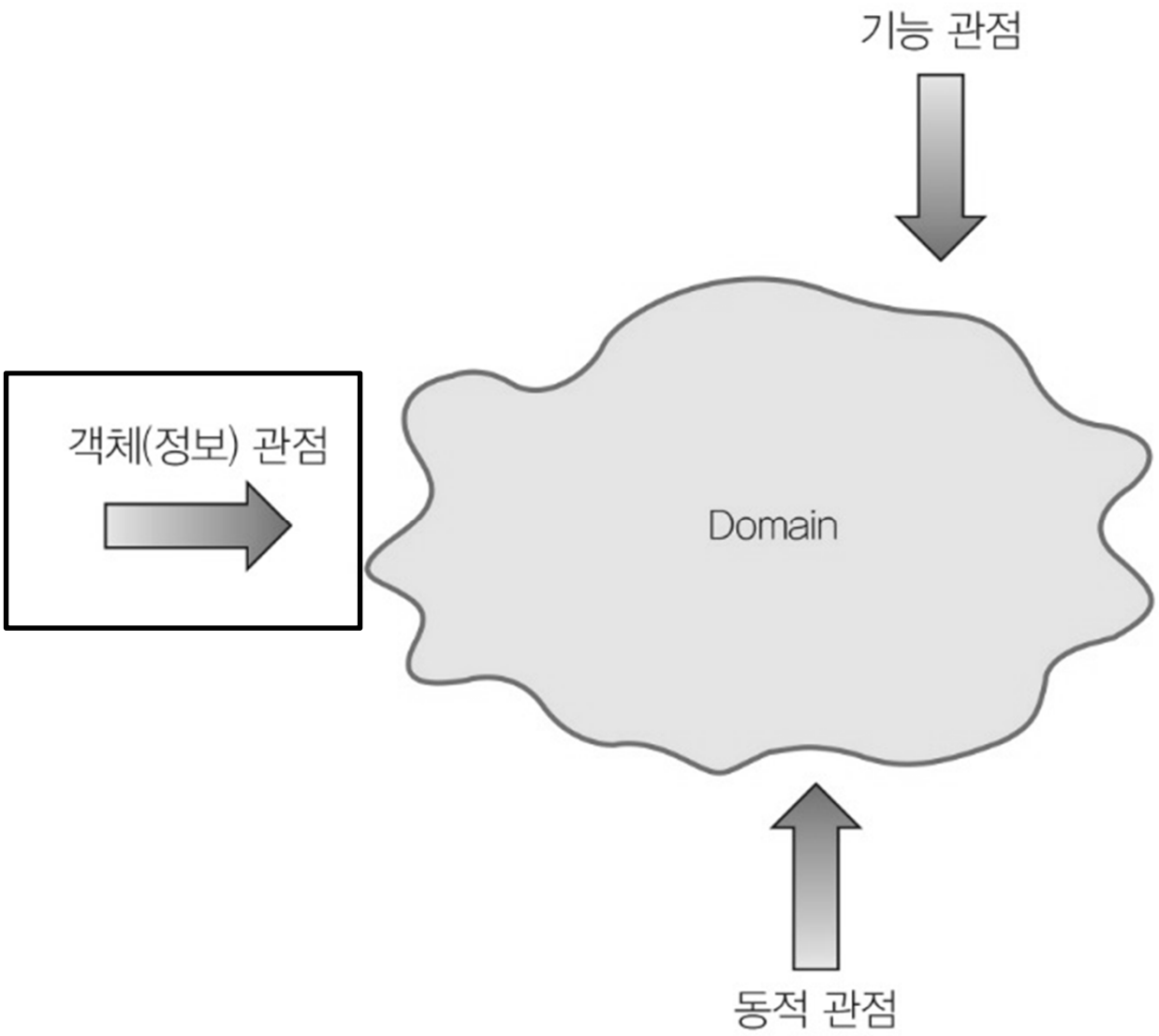
10장 학습 목표

- 객체지향 개발방법론의 근본 개념
- 객체지향 용어(객체, 클래스, 일반화, 집단화 등)
- 객체지향 분석기법의 3가지 모델(객체, 동적, 기능 모델)
- 3가지 모델의 적용 및 통합을 통한 객체 기술
- 객체지향 분석과 설계 과정
- 객체지향 개발방법론의 장점 및 문제점
- 객체지향 개발방법론과 기존 개발방법론과의 차이점
- 객체지향 개발방법론의 개발 초점
- 객체지향 개발방법론에 관심을 두는 이유
- 객체지향 개발방법론과 소프트웨어 품질과의 관계

10.0 객체지향개발방법론 : 배경

- 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 하나 기존의 개발방법은 시스템 확장 및 변경이 용이하지 못해 많은 애로를 겪고 있는데 이를 극복 가능한 유용한 방법론
- **데이터와 행위를 하나로 묶어 객체를 정의 및 추상화 하는 작업**
- 기존의 데이터와 행위가 분리되었던 개발방법의 복잡성과 통합의 어려움 극복 노력 중

10.0 객체지향개발방법론 : 객체지향 분석의 3관점



10.1 객체지향 분석기법 개요

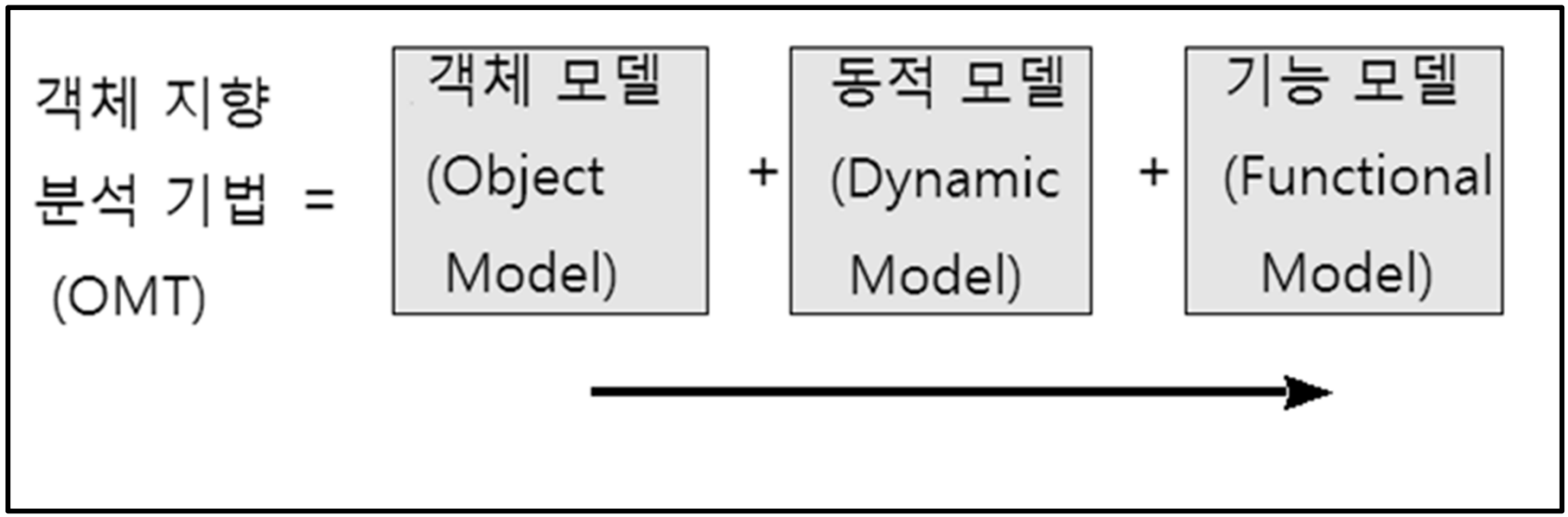
- 태어나면서부터 인식한 객체 개념에 기초한 방법
  - 인간(객체)은 자라면서 상대방(객체)을 인식하고 속성을 찾아내고 관계 파악
  - 인간이 살아가는 과정은 자신(객체)과 다른 사람(객체) 과의 관계를 넓혀 가는 객체지향 과정
- **객체**와 객체의 **속성**과 **동작**, 유사한 객체의 집합으로 나누어진 **클래스**, 객체 사이의 **관계** 등 기본 개념 제공
- 소프트웨어 개발 전과정에 걸쳐 동일한 방법론과 표현 기법의 적용
- 기존 분석기법에 비해 실세계의 현상을 보다 정확히 모델링 할 수 있어 어려운 응용분야에 적용 가능

10.1 객체지향 분석기법 개요

- 분석과 설계의 표현에 큰 차이점이 없어 시스템 개발 용이
- 분석, 설계, 프로그래밍 결과가 큰 변화 없이 재사용이 가능해서 확장성이 좋고 시스템 개발 시 **시제품**이나 **나선형 패러다임**에 적용 가능
- 프로그래밍 기법만이 아니며 시스템 개발에 있어 근본적인 사고의 변환을 요구하는 새로운 기법
- 만들려는 시스템을 객체 중심으로 기술하며, 우리가 주로 해왔던 하향식 방식과는 근본적으로 다름
- **객체 중심의 상향식 접근방법이 도입되고, 기능 중심이 아니라 정보 중심, 데이터 중심으로 시스템 개발**
- 기존의 정보 모델링에 기초하고 있으며, 도출된 객체의 **정적인 정보**에 **객체의 동작을 추가**시켜 객체를 완벽하게 기술하고 구현하는 방법론

10.1 객체지향 분석기법 개요 : 3가지 관점 통합

- 시스템의 각기 서로 다른 면을 보여주며, 이를 통합하는 소프트웨어 설계는 쉬운 일이 아님
- 3가지 관점을 체계적으로 통합해 유연성과 적응력을 가진 우수한 품질의 소프트웨어를 개발할 수 있는 최적의 방법
- 시스템의 요구사항을 분석하기 위해 앞에서 다룬 3가지 모델링 기법을 단계별로 적용하여 통합



10.1 객체지향 분석기법 개요 : 객체지향 분석의 3가지 관점

- 객체지향 분석의 3가지 모델링 관점
  - 객체(정보) 모델링 : 시스템에서 요구되는 객체를 찾아내어 객체들의 특성과 객체들 사이의 관계 규명
  - 동적 모델링 : 객체 모델링에서 규명된 객체들의 행위와 객체들의 상태를 포함하는 생명주기를 보여줌
  - 기능 모델링 : 각 객체의 형태 변화에서 새로운 상태로 들어갔을 때 수행되는 동작을 기술

- 객체지향 분석의 특징
  - 소프트웨어 공학에서 추구하는 많은 장점 제공
  - 객체지향 개발방법의 활용을 위해 기존의 기술과는 다른 높은 차원의 기술력을 가진 분석가나 디자이너 필요
  - 소프트웨어 개발과정에 대한 이해와 정보 모델, 동적 모델, 기능 모델에 대한 지식이 필요하며, 모델 간의 연관성을 바탕으로 모델링 결과의 통합 요구

10.2 객체(정보) 모델링(Object Modeling)

- 시스템에 요구되는 객체들을 보여줌으로써 주로 시스템의 정적인 구조의 포착에 사용
- 객체지향 분석에서 가장 중요하며 선행 되어야 할 모델링
- 객체 모델링 복잡도를 관리하기 위하여 정보 모델링에서 언급된 **추상화, 분류화, 일반화**의 개념 사용
- **집단화** 개념이 추가되어 다양한 객체들을 모아 더 높은 단계의 객체 추출 작업이 이루어짐
- 시스템 요구사항을 기술한 문제기술로부터 시스템에 요구되는 객체를 추출
- 객체들이 발견된 다음에 객체들 간의 연관성을 찾아내고 객체들을 정의하기 위한 속성 추가

10.2 객체 모델링(Object Modeling) : 객체지향 용어

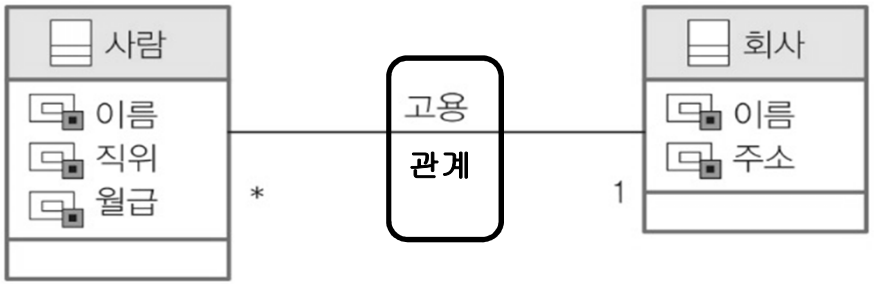
- 객체(Object) vs 엔티티(Entity, 개체)
- 클래스(Class)
- 속성(Attribute)
- 관계(Relationship)
- 동작(Operation)
- 일반화(Generalization)
- 특수화(Specialization)
- 집단화(Aggregation)
- 상위클래스(Superclass)
- 하위클래스(Subclass)
- 상속(Inheritance)
- 재사용(Reuse)

10.2.1 객체와 클래스

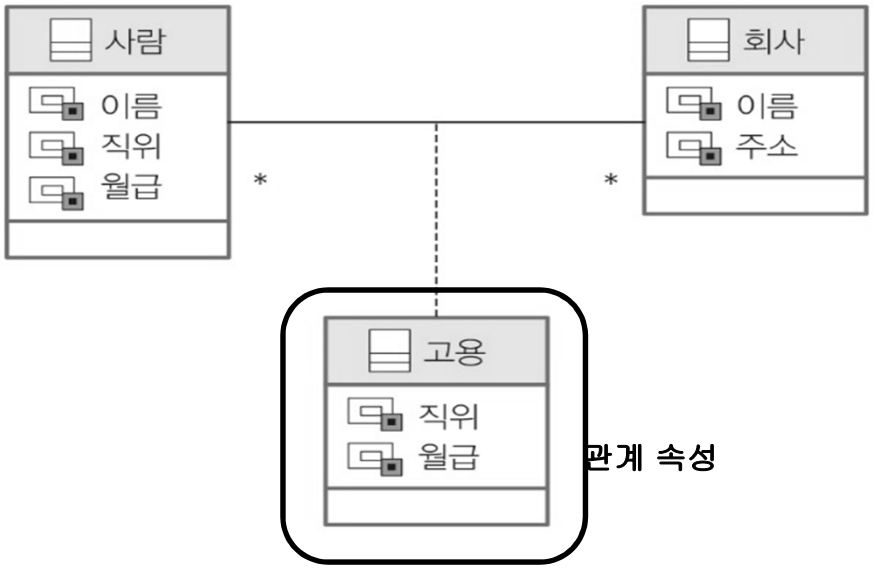
- 객　체 : 객체 모델링의 기본 단위
- 클래스 : 유사한 객체들의 모임
- 정보 모델링의 엔티티와 엔티티 타입에 해당
- 표기법
  - 객　체 : 구석이 다듬어진 사각형
  - 클래스 : 사각형으로　표시

10.2.2 클래스의 관계

- 관계(Relationship) : 클래스 간의 연관성
- 표기법
  - 정보 모델링 : 마름모로 표시
  - UML : 관계 이름만 표시

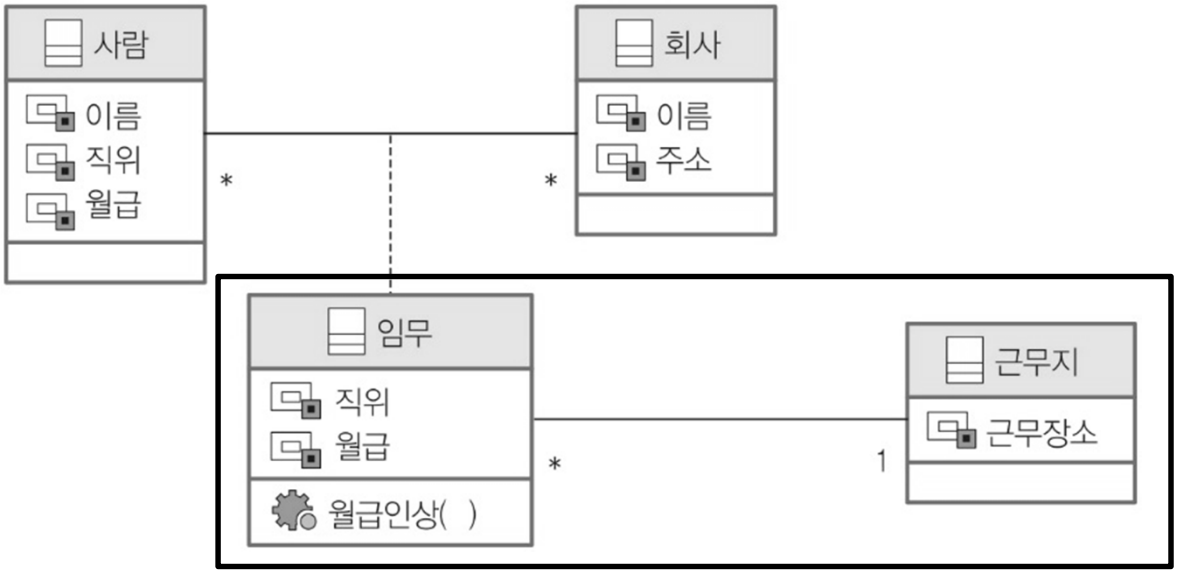


  - 만약 관계가 새로운 속성이나 동작을 가질 경우
    - 클래스 사이의 관계를 새로 도출시키고 관계 속성 표시



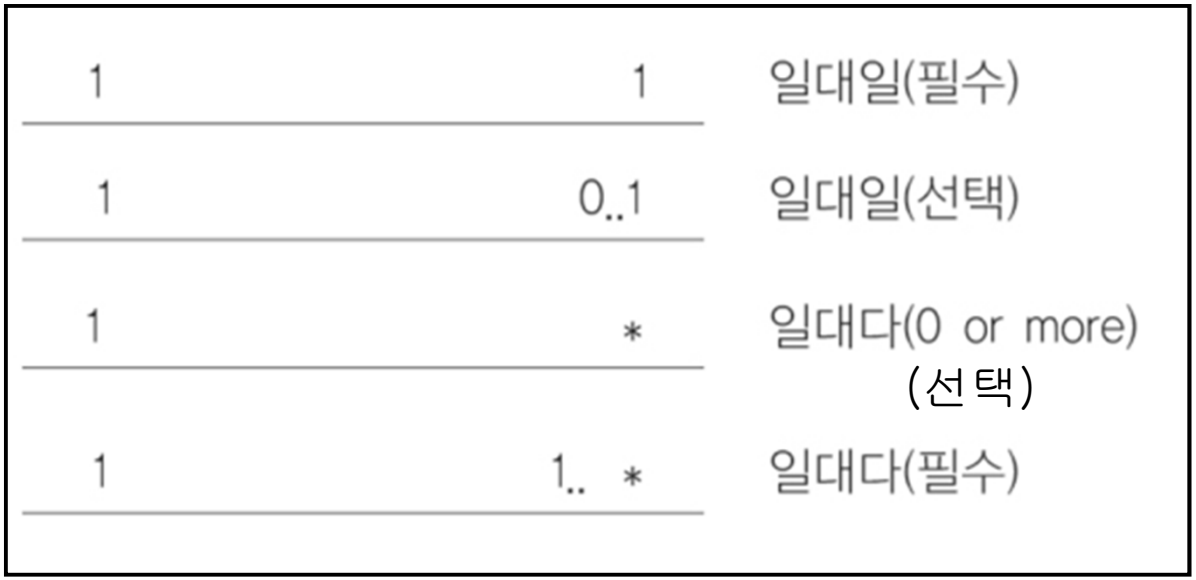
10.2.2 클래스의 관계

- 경우에 따라서 관계를 한 클래스로 모델링 하는 것이 효과적임
- 관계가 하나의 클래스로 정의되어 속성과 동작을 가지게 되며 다른 새로운 클래스와 연관 가능
- 클래스로 정의된 관계 예



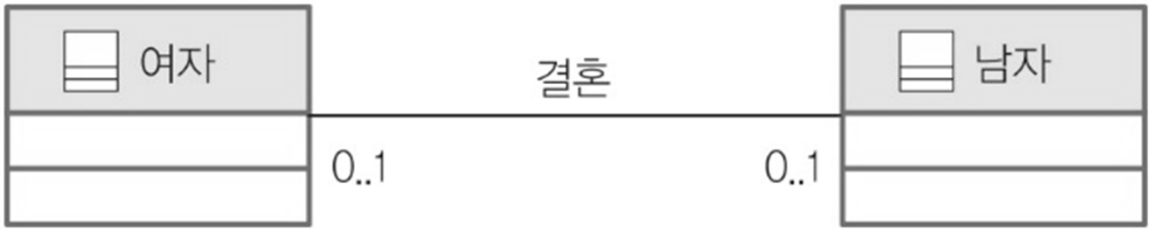
10.2.2 클래스의 관계 : 매핑 제약조건

- 클래스 간에 대응 관계를 통해 객체들이 지켜야 하는 제약조건을 지정하는 것
- 각 클래스의 객체들 사이에 맺어질 수 있는 매핑 제약조건과 참여 제약조건은 정보 모델링에서 다루어진 개념과 동일(what?)
- UML에서 매핑 수와 참여 제약조건 표시



10.2.2 클래스의 관계 : 매핑 관계

- 일대일(1:1) 관계



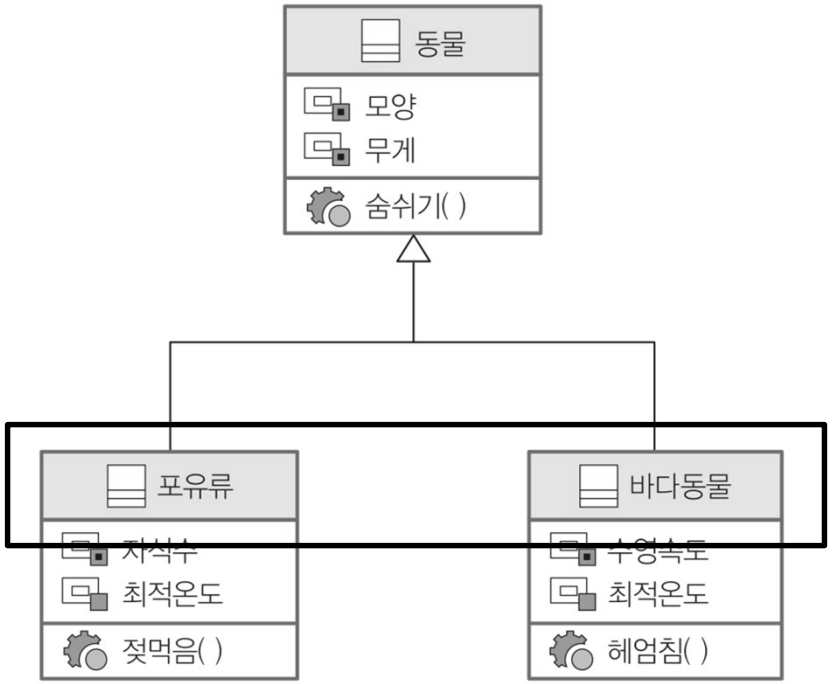
- 일대다(1:n) 관계

10.2.3 일반화(Generalization)

- 일반화
  - 유사한 클래스 간의 공유 속성과 동작을 묶어 주고, 그들 간의 다른 점을 보존할 수 있게 해주는 효과적인 추상화 기법
  - 클래스 간의 특수한 관계
  - 클래스 간의 계층 구조가 만들어지며 각 하위 클래스는 상위 클래스의 속성, 동작, 그리고 다른 클래스와의 연관성 상속
  - 공통 정보는 한번만 정의될 수 있어 분석 결과를 재사용할 수 있게 해주며 데이터 무결성 향상
- 특수화
  - 일반화의 역작용
  - 하위 클래스로 정의되는 경우는 하위 클래스가 고유의 속성이나 동작을 가지고 있거나 하위 클래스가 다른 클래스들과 고유의 관계를 가지고 있을 때를 특수화라 함

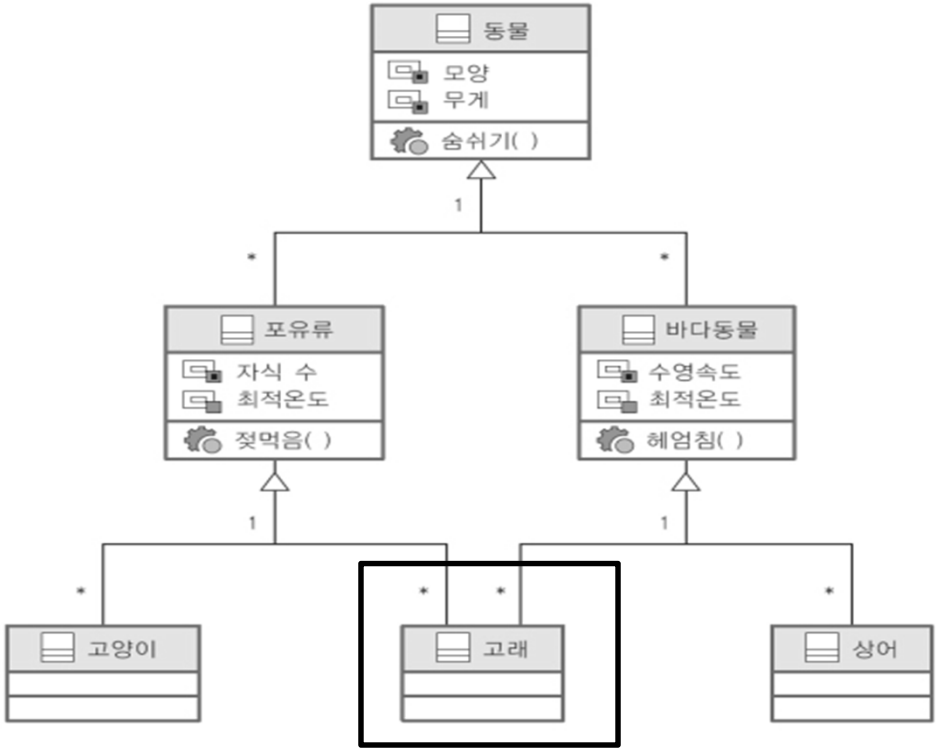
10.2.3 일반화(Generalization) : 클래스의 일반화

- 'is_a' 또는 'kind_of' 관계
- 각 하위 클래스의 인스턴스는 상위 클래스의 인스턴스
- 다계층 구조를 만들어 나갈 수 있으며 이 경우 하나의 하위 클래스는 이 구조 상에 있는 모든 상위 클래스의 속성과 동작 상속



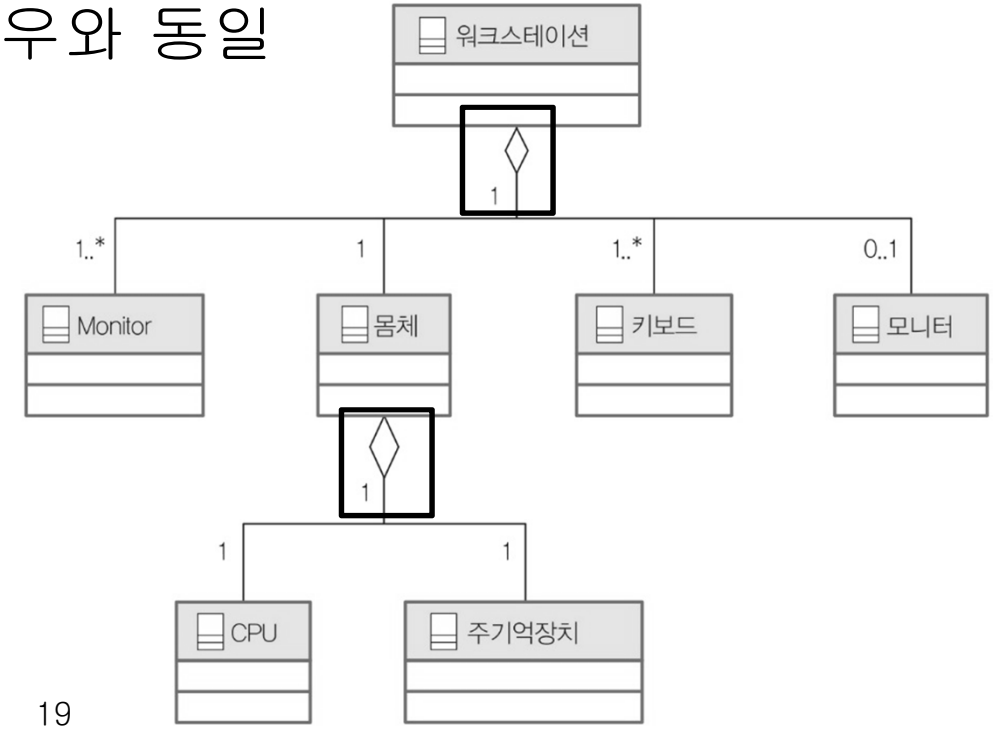
10.2.3 일반화(Generalization) : 다중 상속

- 어떤 클래스의 상위 클래스가 2개 있을 땐 문제 발생!
- 하위 클래스는 2개 이상의 상위 클래스에서 동시에 속성과 동작 상속
- 다중 상속으로 분석과정에서 속성 및 동작에서 모순 발견 시 모호성 제거 위해 요구사항 명세서나 자료사전에 기록



10.2.4 집단화(Aggregation)

- 클래스 간에 "부분-전체(is a whole)"나 "부분(is a part of)" 관계로 설명되는 연관성
- 여러 부속 객체가 모여 하나의 객체가 구성되는 것
- EER 모델에서는 구체적으로 나타내지 않던 개념이며 객체지향 방법론에서 많이 사용
- 작은 마름모 기호로 표시, 매핑 제약조건과 참여 제약조건의 기호는 일반화 경우와 동일
- 다계층 집단화의 예



10.2.5 현금 자동 지급기의 예

- ATM 시스템 문제기술의 일부

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

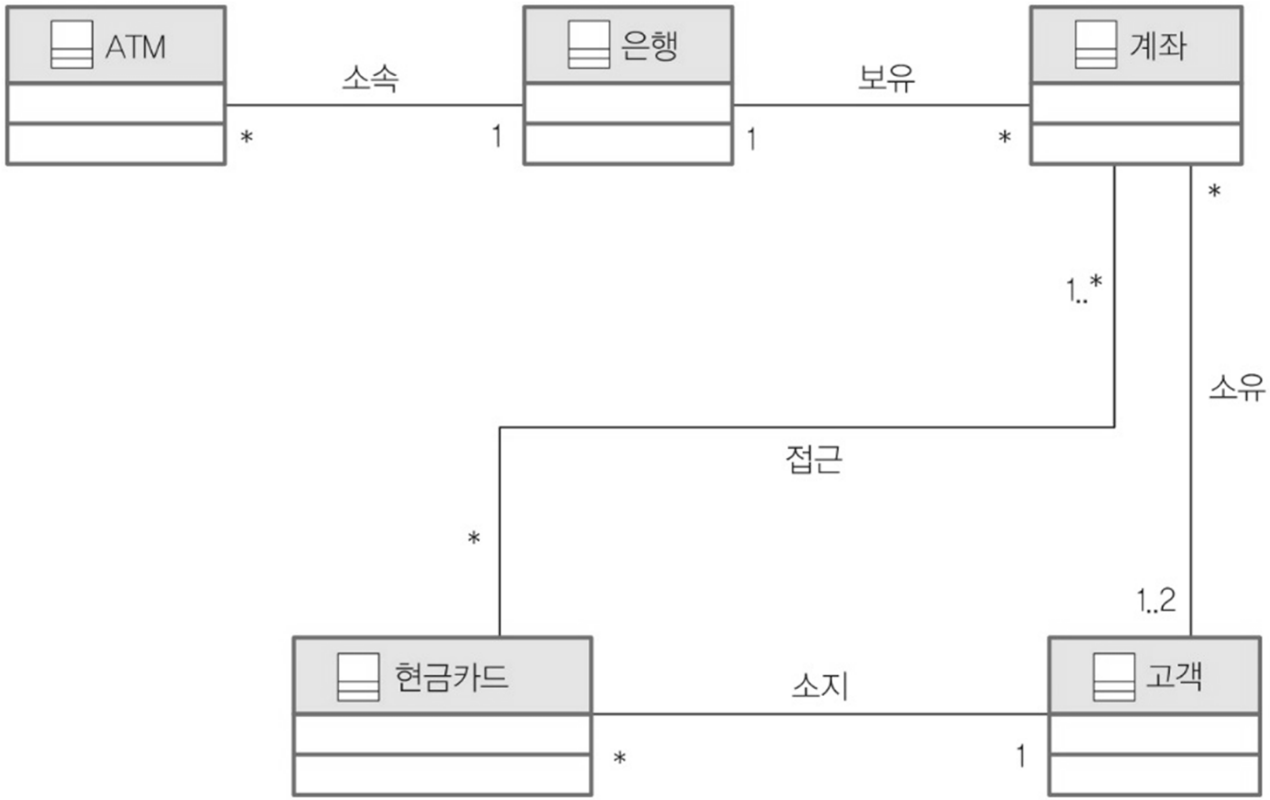
① 문제기술에서 요구되는 객체들을 추출

- ATM 시스템 클래스 추출 결과

10.2.5 현금 자동 지급기의 예 : 객체들 사이의 연관성

② 문제기술서에 상세한 사항이 기술되지 않아서 사용자(은행 담당자)와 상의하여 연관성과 구체적인 요구사항 규명

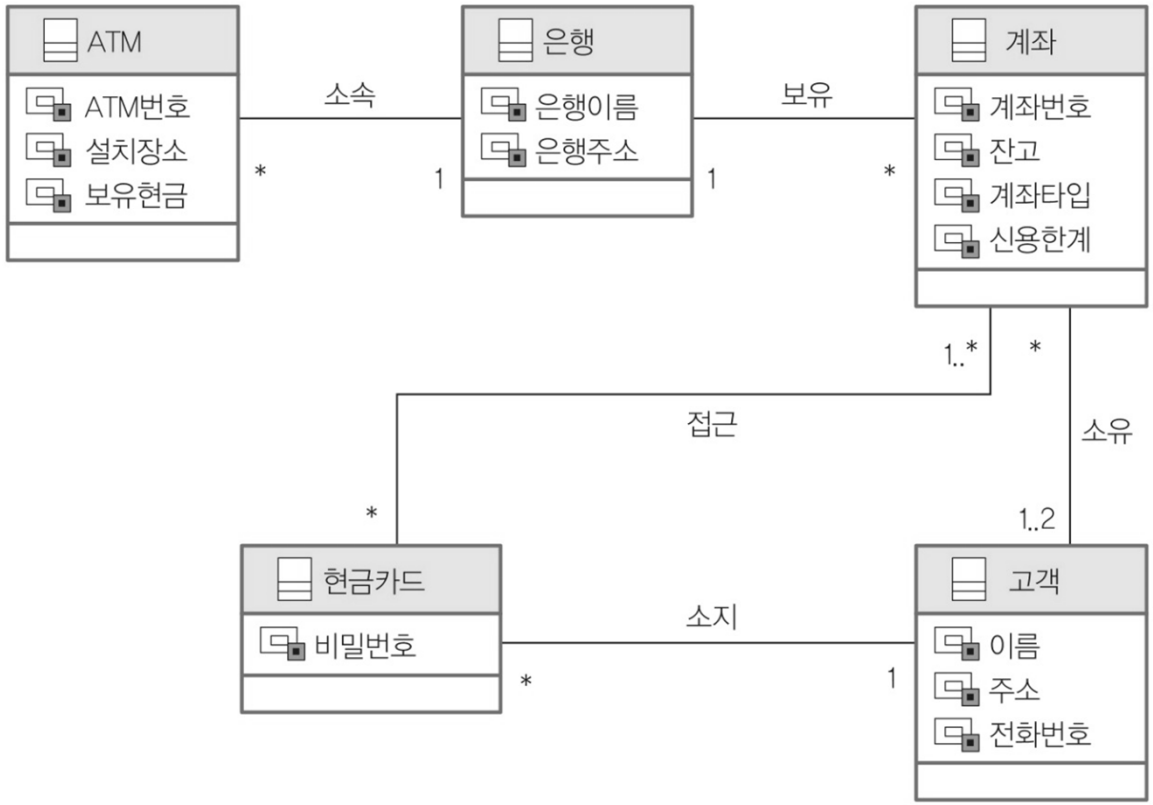
- ATM 시스템 클래스들 사이의 관계 규명 결과



10.2.5 현금 자동 지급기의 예 : 클래스의 속성

③ 클래스와 클래스들 사이의 연관성이 밝혀진 다음 클래스의 특징을 밝혀주는 속성과 동작 규명

- 객체 모델링에서는 속성에 우선 초점 맞춤
- ATM 시스템 클래스의 속성 규명 결과



10.3 동적 모델링(Dynamic Modeling)

- 객체 모델링 : 시스템에 요구되는 객체들의 구조와 객체 간의 관계 정립
- 동적 모델링
  - **시간 흐름에 따른 객체들과 객체들 사이의 변화 조사**
  - 객체 간의 제어 흐름, 상호작용, 동작의 순서 등을 다룸
  - 동적 모델의 중요한 개념은 **상태, 사건, 동작** 등
  - 객체들의 동적인 면을 찾아내기 위함
  - 기능 모델과 관계되며, 시스템 동작은 기능 수행을 위해 필요

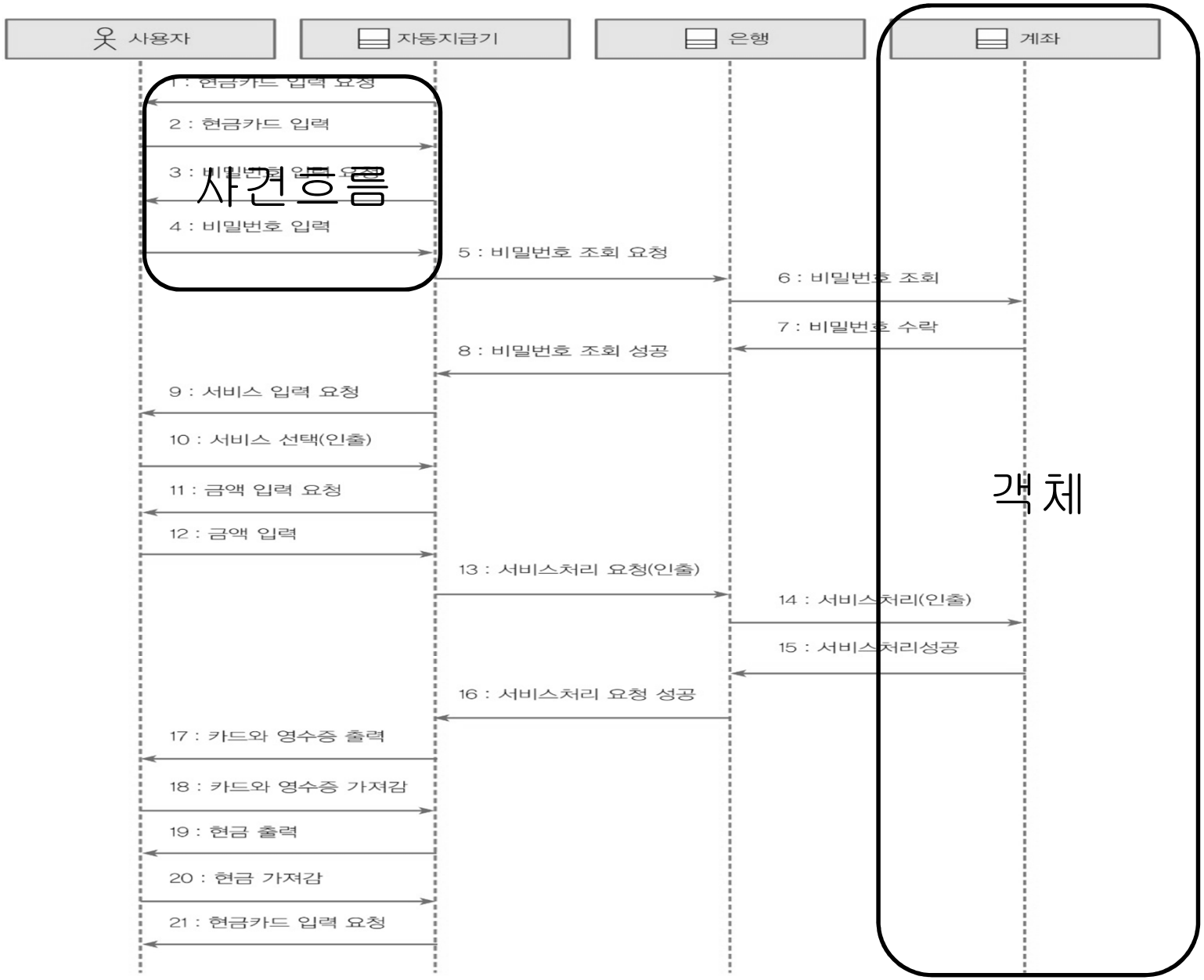
10.3 동적 모델링(Dynamic Modeling)

- 제안서로부터 **시나리오**를 만들 때 시나리오는 객체 또는 시스템의 한 실행과정을 사건의 흐름으로 표시
- 각 사건들은 객체 모델링에서 규명된 객체들 사이의 정보 흐름을 나타내게 되고, 각 사건의 정보를 보내는 객체와 정보를 받는 객체가 밝혀짐
- 사건의 순서와 사건을 주고받는 객체들을 **사건추적도**에 나타냄
  - 표기법 : 수직선 :- 객체, → :- 사건 흐름
  - 사건은 위에서 아래로의 순서에 의해 수행

10.3 동적 모델링(Dynamic Modeling) : 시나리오

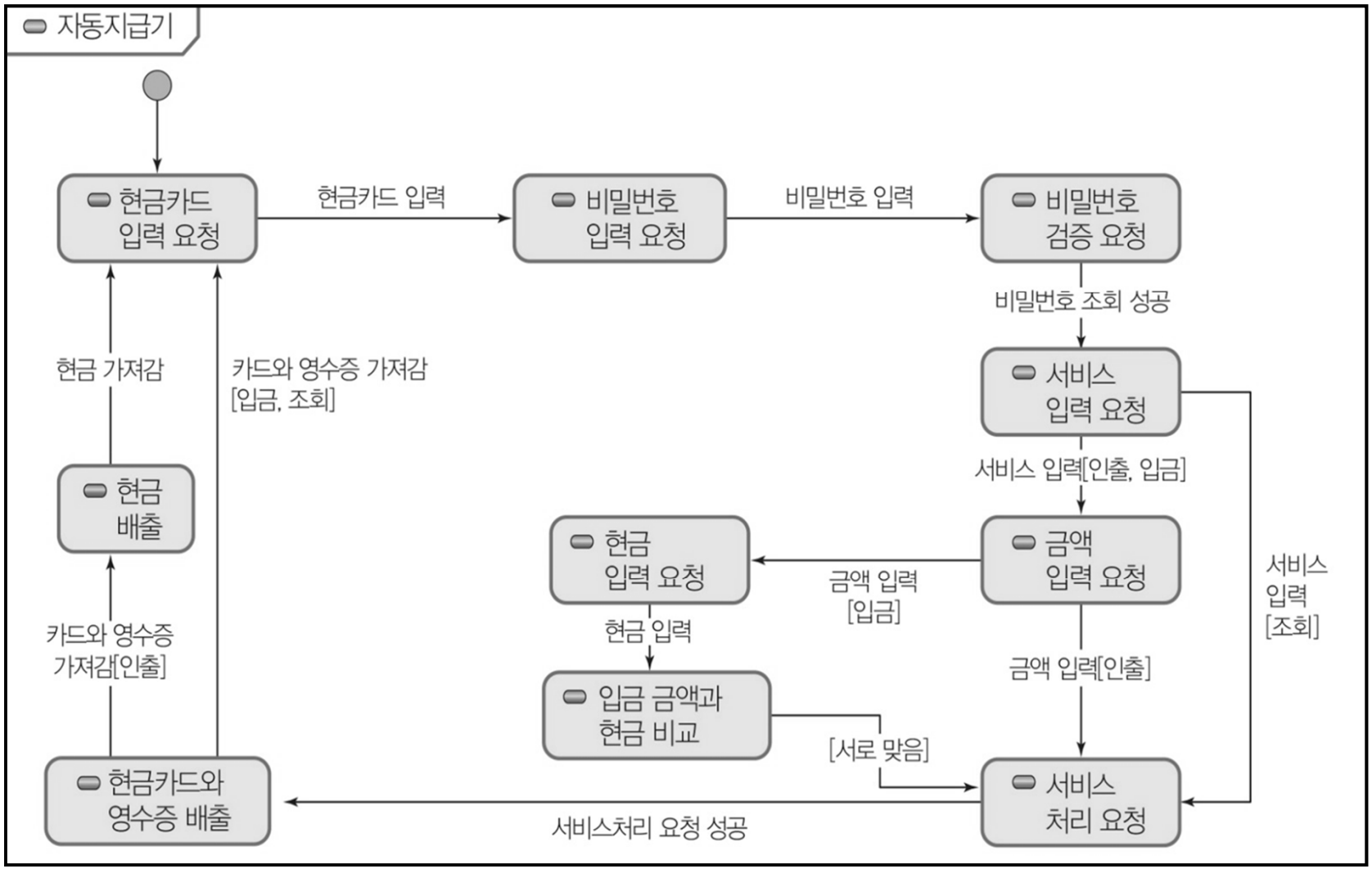
- 자동 지급기가 현금 카드를 입력할 것을 요구한다.
- 사용자가 현금 카드를 자동 지급기의 카드 입구에 넣는다.
- 자동 지급기는 현금 카드로부터 계좌 번호와 카드 번호를 읽고 사용 자에게 비밀 번호를 요구한다.
- 사용자가 비밀번호를 입력한다.
- 자동 지급기는 현금 카드 소속 은행에게 비밀 번호 대조를 요청한다.
- 은행은 현금 카드에게 비밀 번호 대조를 요청한다.
- 현금 카드는 은행에게 비밀 번호가 일치함을 알린다.
- 은행은 자동 지급기에게 비밀 번호가 일치함을 알린다.
- 자동 지급기는 사용자에게 가능한 서비스를 보여준다.
- 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.

10.3 동적 모델링 : 현금 인출에 대한 시퀀스 다이어그램

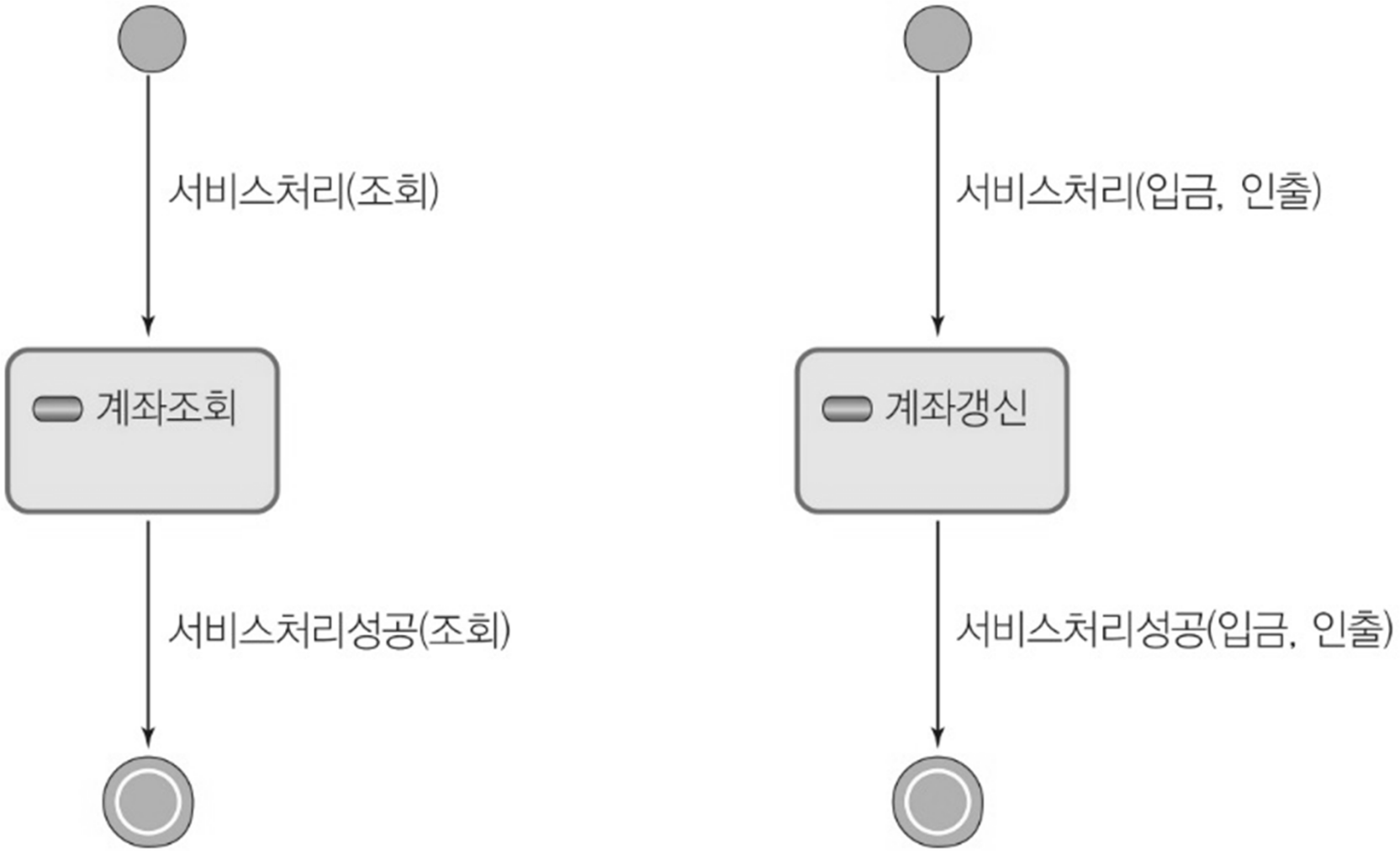


10.3 동적 모델링(Dynamic Modeling): 상태변화도(STD)

- ATM 자동지급기의 STD(State Transition Diagram))



10.3 동적 모델링(Dynamic Modeling) : 계좌의 STD

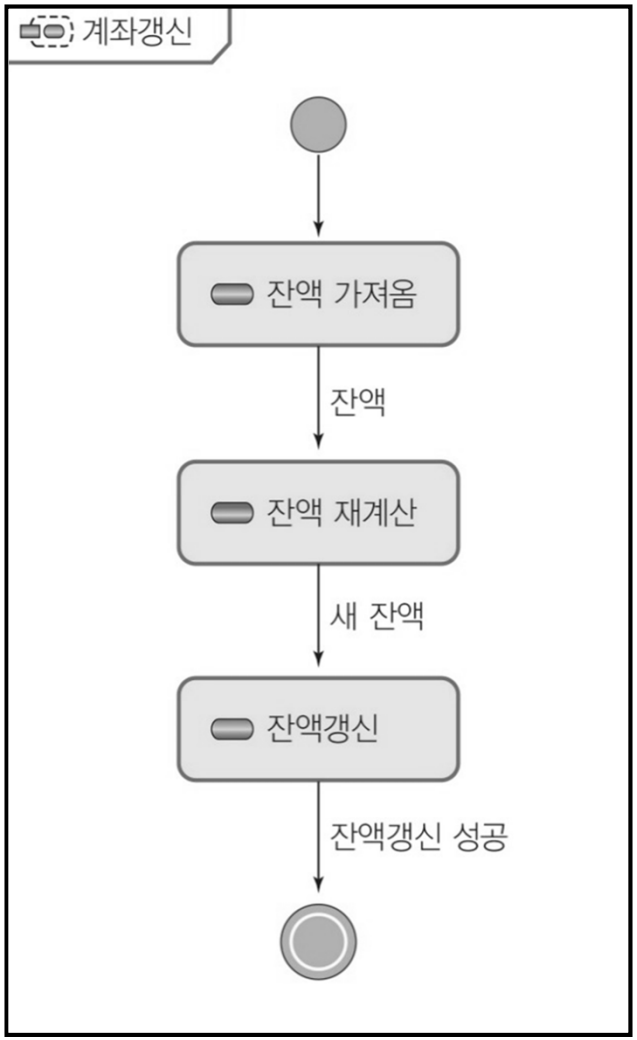


10.4 기능 모델링(Functional Modeling)

- 객체지향 분석기법에서 객체 모델링, 동적 모델링에 이어 시스템을 기술하는 3번째 단계
- 입력값부터 계산을 거쳐서 어떤 결과가 나타나는지 보여주며, 이것이 어떻게(how to) 유도되었는지 구현방법은 고려치 않음
- 구조적 방법론과 마찬가지로 자료흐름도(DFD) 사용 가능
- DFD의 경우 입력 흐름은 프로세스를 통해 변환되어 출력 흐름으로 바뀌고, 다른 프로세스의 입력이나 외부로의 출력으로 작용
- 동적 모델링에서 나타난 동작이 어떠한 기능으로 이루어져 있는지 보여주며 자료흐름도의 정보 흐름은 객체에 해당

10.4 기능 모델링 : 계좌 변경 동작의 기능

- 계좌 갱신의 액티비티 다이어그램 예



10.5 객체지향 설계

- 분석 단계에서는 논의된 3가지 모델이 순서 대로 개발
- 3가지 모델을 하나로 통합하는 작업이 이루어져야 하며, **객체지향 설계단계**에 해당
- 3가지 모델 통합은 객체의 정적 구조와 오퍼레이션을 함께 포함하여 객체를 정의하는 것
- 동적 모델의 **사건, 동작** 및 **활동**을 객체의 오퍼레이션에 매핑하고, 기능 모델의 **프로세스**를 객체 모델의 **오퍼레이션**에 통합시키는 것!
- 객체 수준의 상태변화도는 한 객체가 생명주기 동안 가질 수 있는 상태를 기술하며, 상태의 변환은 객체의 오퍼레이션으로 매핑
- 유사한 방법으로 객체에 주어진 사건은 다른 객체의 동작으로 나타냄
- 하나의 사건은 이전 사건에 의하여 초기화된 활동의 결과이며, 동작과 이에 반응하는 동작으로 매핑

10.5 객체지향 설계 : 모델의 통합

- 자료흐름도는 다른 두 모델과 연관성을 가지고 있으며, 상태 변환에 의하여 이루어진 동작은 기능 모델의 자료 흐름도와 연관
- 서로 연결되어 있는 프로세스 집합들은 동작의 구체적인 목적(기능) 표시
- 자료흐름도의 프로세스들은 상태변화도의 동작에 의하여 수행되는 하위 기능으로 나타냄
- 이들 동작의 일부는 자신에게 작용하는 동작일 수도 있고 다른 객체에 보내지는 동작일 수 있음
- **대상 객체**는 오퍼레이션 정의되고 속하여 있는 객체 의미
- 만약 프로세스가 입력 흐름을 가진다면 입력 흐름이 대상 객체가 될 수 있고, 이 프로세스는 대상 객체의 오퍼레이션이 됨

10.5 객체지향 설계 : 모델의 통합

- 객체 모델에 동적 모델과 기능 모델 통합 가이드라인
  - 만약 프로세스가 입력 흐름을 사용하고 입력 흐름을 고쳐 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 저장 하는 경우
    - 자료저장소나 외부 객체가 이 오퍼레이션의 대상 객체가 되며,
    - 이 결과는 객체에 속한 모든 가능한 정보, 객체들 사이의 관계나 객체의 오퍼레이션들을 나타낼 수 있게 되어 시스템에 대한 완벽한 기술 가능
  - 객체들 사이의 **관계** ⇒ 상대 객체를 나타내는 포인터 변수를 객체 속성으로 표현

10.5 객체지향 설계 : 계좌 클래스 설계 예

계 좌(Account)
계좌번호 /* <그림 9.14> */ 잔고 /* <그림 9.14> */ 계좌타입 /* <그림 9.14> */ 신용 한계 /* <그림 9.14> */ 접근가능 현금카드 /* <그림 9.14>의 계좌와 현금 카드 사이의 연관성 */ 보유 은행 /* <그림 9.14>의 계좌와 은행사이의 연관성 */ 소유주 /* <그림 9.14>의 계좌와 고객사이의 연관성 */
계좌_조회() /* <그림 9.18> */ 계좌_갱신(int 트랜잭션_종류, int 금액) /*<그림 9.18> */ 잔액_가져옴() /* <그림 9.19> */ 잔액_재_계산(int 트랜잭션_종류, int 금액, int 잔액) /* <그림 9.19> */ 잔액_갱신(int 새_잔액) /* <그림 9.19> */

계좌_조회()

{ /* 이 오퍼레이션은 해당 계좌의 잔액을 돌려준다. 이 동작은 계좌의 상태변화도 중 계좌 조회 동작으로 부터 유도되었다. */}

계좌_갱신 오퍼레이션의 매개변수 입력은 계좌 갱신 상태에 대한 <그림 12.19>의 DFD로부터 유도되었다.

계좌_갱신(int 트랜젝션_종류, int 금액)

{/* 이 오퍼레이션은 트랜젝션 종류, 금액을 입력으로 받아 해당 계좌 의 잔액을 갱신한다. 이 오퍼레이션은 계좌의 상태변화도 중 계좌 갱신 동작으로부터 유도되었다. */

int 잔액, 새_잔액;

잔액 := 잔액_가져옴();

새_잔액 := 잔액_재_계산(트랜젝션_종류, 금액, 잔액);

return (잔액_갱신(새_잔액));

}

10.5 객체지향 설계 : 계좌 클래스 오퍼레이션

잔액_가져옴 ()

{/* 해당 계좌의 잔액을 돌려준다. */}

잔액_재_계산 (int 트랜젝션_종류, int 금액, int 잔액)

{/* 트랜젝션 종류에 따라 금액만큼 잔액을 재 계산한다. */}

잔액_갱신 (int 새_잔액)

{/* 해당 계좌의 잔액을 새_잔액으로 갱신한 후 오퍼레이션이 성공적 으로 끝났음을 알린다. */}

서비스_처리(계좌 해당_계좌, int 트랙젝션_종류, int 금액)

{/* ATM으로부터 넘어온 매개변수를 바탕으로 해당 계좌에게 적절한 오퍼레이션을 요청한다. 매개변수로 입력받은 계좌 객체인 해당_계좌는 ATM을 통해입력된 계좌 번호를 바탕으로 은행에서 찾아 놓은 것이다. */

```
if (트랜젝션_종류 = 조회)
    return (해당_계좌.계좌_조회());
else if ((트랜젝션_종류 = 입금) or (트랜젝션_종류 = 인출))
    return (해당_계좌.계좌_갱신(트랜젝션_종류, 금액));
}
```

10장 요약

- 객체지향 개발 방법
  - 시스템 확장이나 변경을 쉽게 할 수 있는 뛰어난 기법
  - 기존 분석기법이 가지고 있는 한계점을 극복하고 시스템의 3가지 관점을 체계적으로 통합하여 기술하고, 확장성과 적응력이 뛰어난 시스템 설계 목적
  - 상향식 접근방법이며, 구조적 기법의 하향식 접근방법과 대조
  - 상향식 기법에 따른 많은 장점 보유
    - 시스템의 데이터에 초점을 맞추기 때문에 변화에 잘 견디는 우수한 품질의 소프트웨어 생산
    - 객체지향 분석 결과는 설계로 넘어가 큰 변화 없이 사용가능
    - 3가지 모델의 결과가 통합되어 완벽한 객체와 클래스 구현 가능

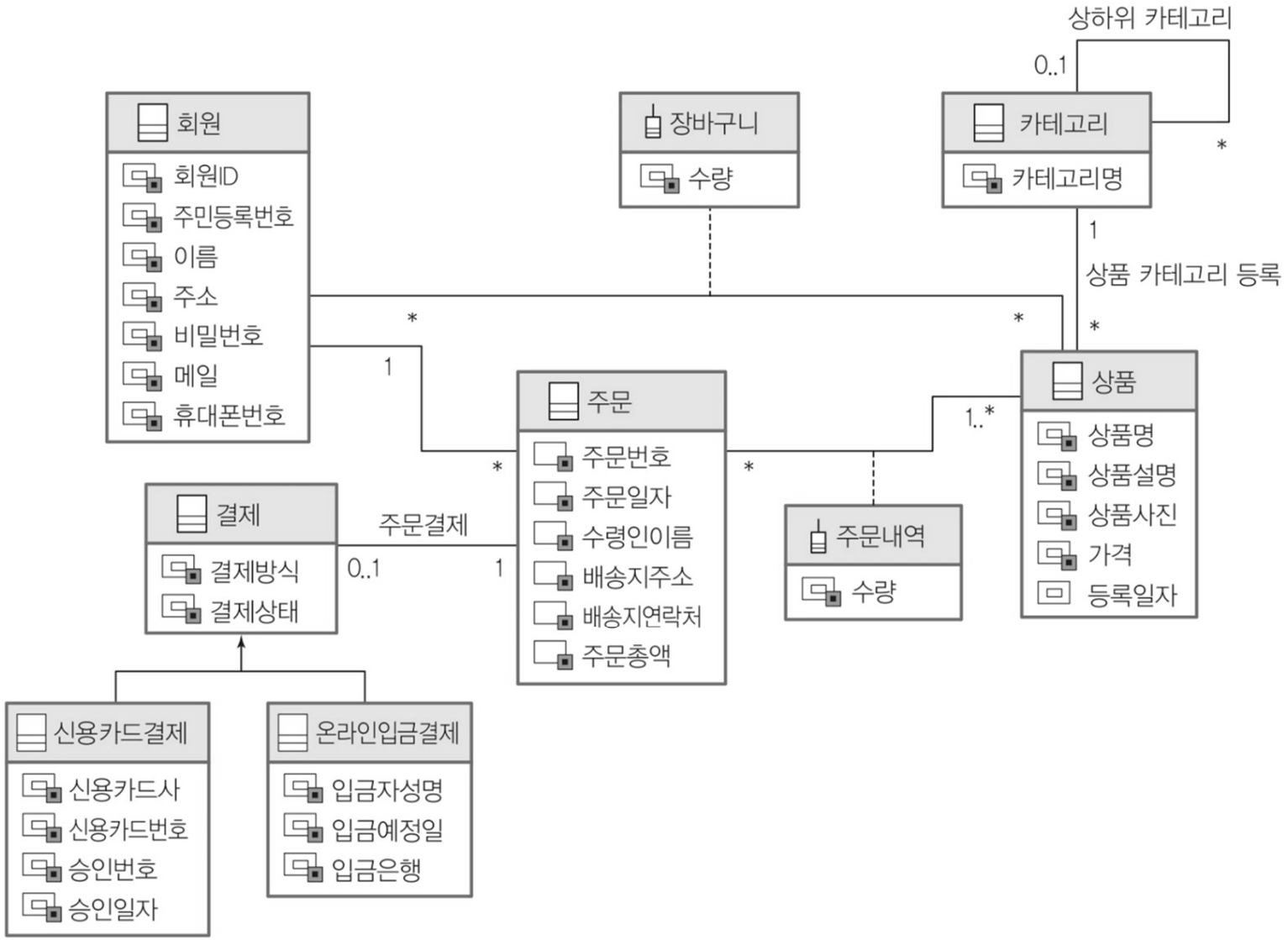
10장 요약

- 객체지향 개발 방법(계속)
  - 시스템 확장성이 용이하고 엔지니어링 결과를 재사용할 수 있어 원형 패러다임이나 나선형 패러다임을 적용한 시스템 개발에 많은 도움
  - 객체지향 개발방법 적용 시 단점 및 고려사항
    - 튼튼하고 견고한 시스템은 분석과정에서 집중적으로 이루어지는 철저한 검증과정과 이에 따른 수정 없이는 불가능
    - 기존의 구조적 기법 등 다른 접근방법에서 객체지향으로의 변화에 요구되는 교육과 훈련 필요
    - 객체지향 개념과 설계기법, 객체지향 언어를 배우는 데 비용 요구
    - 많은 프로그래머가 객체 중심보다는 기능 중심으로 시스템을 개발해 왔으며 객체지향 사고로의 전환 및 문화의 변화 필요
    - 깊은 클래스 계층 구조가 있는 경우와 시스템 작동 중 객체에 대한 동적 할당과 제거 시 성능에 대한 문제가 제기

10장 요약

- 객체지향 개발방법(계속)
  - 소프트웨어공학의 목적인 고품질 소프트웨어를 얻을 수 있는 좋은 기법으로 수정, 이해, 재사용이 용이하여 많은 개발분야에 확산
  - 객체지향 설계는 응용분야의 관점에서 컴퓨터 관점으로, 사용자 관점에서 개발자 관점으로 이동하여 객체지향 분석 결과에 살을 붙여가는 과정
  - 객체지향 개발방법이 적합한 응용분야
    - 응용분야의 객체들이 논리적으로 잘 분리되며, 각 객체가 독립적인 자료구조와 오퍼레이션을 가질 때 효과적
    - 통신분야, 의료진단, 일기예보, 명령제어 분야 등
  - 객체지향 기법은 데이터보다는 성능을 중요시하며, 많은 계산을 요구하는 시스템에는 부적합

10장 요약 : 인터넷 쇼핑몰의 클래스 다이어그램



강의 계획 피드백 (4주차-3)

주차	강의주제	강의내용	과제	평가
1주차	객체지향 패러다임	과목 소개 및 객체지향 방법론의 전반적인 개요		
2주차	프로젝트 관리1	프로젝트 계획 및 팀 편성/프로젝트 과제 제시		
3주차	소프트웨어 개발방법론과 UML	기존의 소프트웨어 개발방법론과 객체지향방법론 차이점 이해	과제1 : 프로젝트 [illegible] 제출(5)	
4주차	Use Case와 UML	UML 특성 이해		
5주차	UP(Unified Process) 방법론	UP 방법론 이해		
6주차	비즈니스 모델링 및 요구사항 정의	사례를 통한 비즈니스 모델링 및 요구사항 정의 방법 이해	과제2 : 요구사항 정의 결과 제출(5)	
7주차	분석 모델링 및 UML 다이어그램(분석)	객체지향 분석 방법 이해 및 분석용 UML 다이어그램 작성 방법 이해		
8주차	분석 결과 문서화 및 설계 모델링	분석 산출물 작성 방법 및 객체지향 설계 방법 이해	과제3 : 분석 결과 제출(10)	
9주차	UML 다이어그램(설계)	설계용 UML 다이어그램 작성 방법 이해		
10주차	객체 설계	객체설계 및 세분화		
11주차	설계 결과의 문서화 및 프로젝트 관리 2	시스템 설계 결과의 문서화 방법 이해 및 형상관리/검증과 확인 방법 이해	과제4 : 설계 결과 제출(10)	
12주차	시스템 구현	객체지향 프로그래밍의 기본 개념 및 기법		
13주차	시스템 테스트 및 구현/시험 결과의 문서화	객체지향 테스팅 기법 및 구현/시험 산출물의 문서화 방법 이해	과제5 : 구현/시험 결과 및 유지보수 계획 제출(20)	
14주차	프로젝트 관리3	소프트웨어 품질관리와 프로세스 개선 방법 이해		
15주차	최종 결과 문서화 및 발표	최종 산출물 문서화 방법 이해 및 개발 결과 발표	과제6 : 최종보고서 제출 및 발표(10)	

다음 주(5주차) 강의계획

- 교재#1의 11장, 교재#2의 1~2장 꼭 읽어오기
- 팀 프로젝트 진행 : 계획단계-분석단계
  - 프로젝트 헌장(PC) 수정제출 : 최종 제출(2016.09.28, 18:00)
  - 프로젝트 관리 계획서(PMP) :
    - 1차 제출(2016.09.28, 18:00)
    - 최종 제출(2016.10.05, 18:00)
  - 요구사항 수집 및 요구사항 정의 : 1차 제출(2016.10.05, 18:00)

☞ 다음 주(5주차)는 **교재#1의 11장 및 교재#2의 1~2장 강의**