

## 5. 객체지향 분석

---

# 학습목표

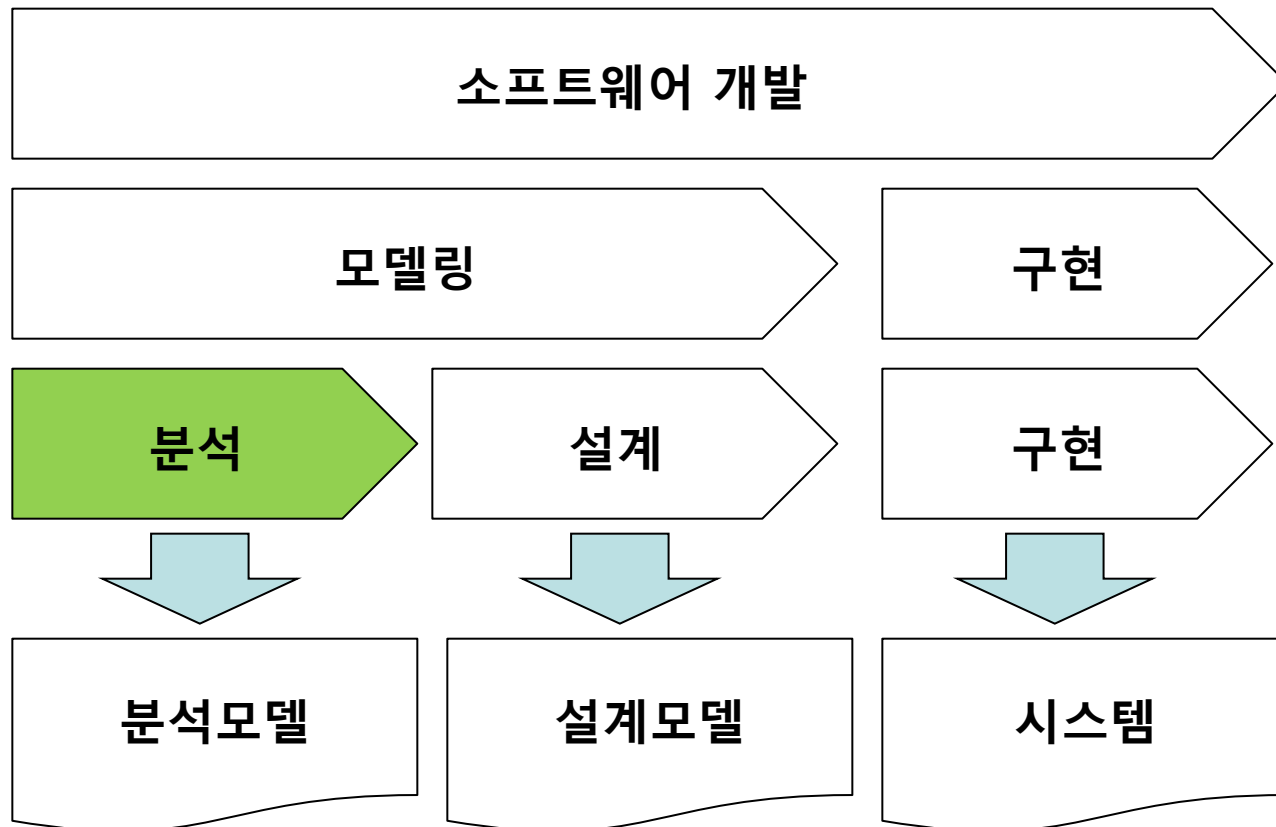
---

- ❖ 객체지향 분석 개요
- ❖ 객체 모델
- ❖ 객체지향 분석 프로세스

# 모델링과 소프트웨어 개발

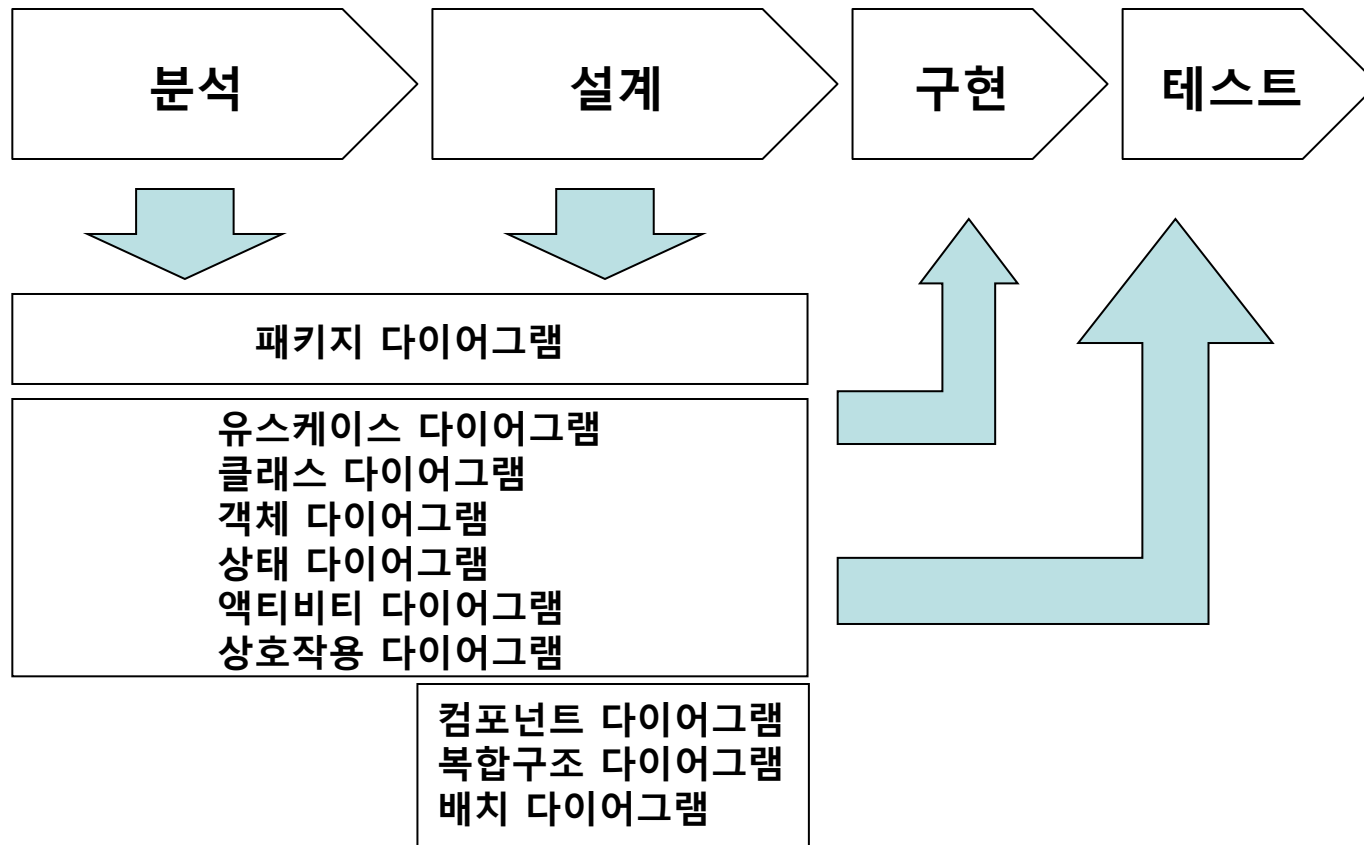
## ❖ 분석

- 문제 영역에 대한 연구
- 외부적으로 관찰 가능한 행위에 대한 명세서 작성



# UML과 소프트웨어 프로세스

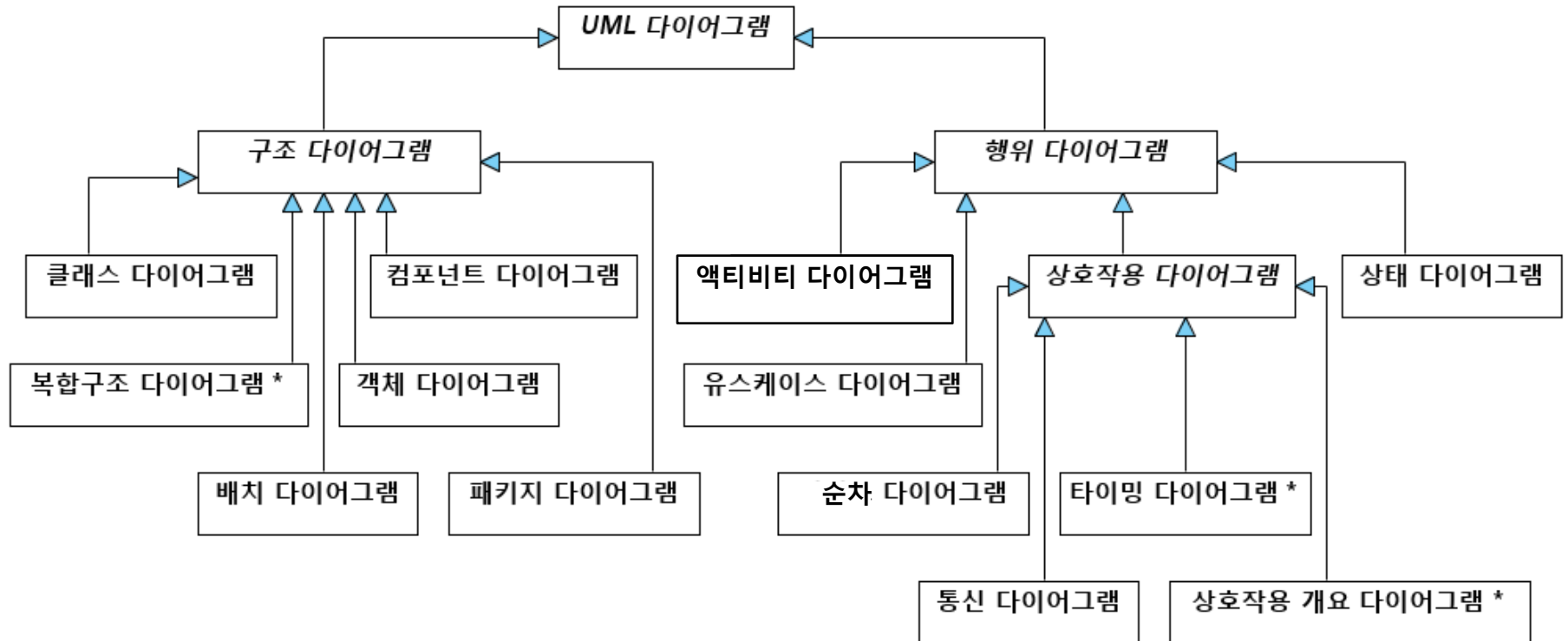
---



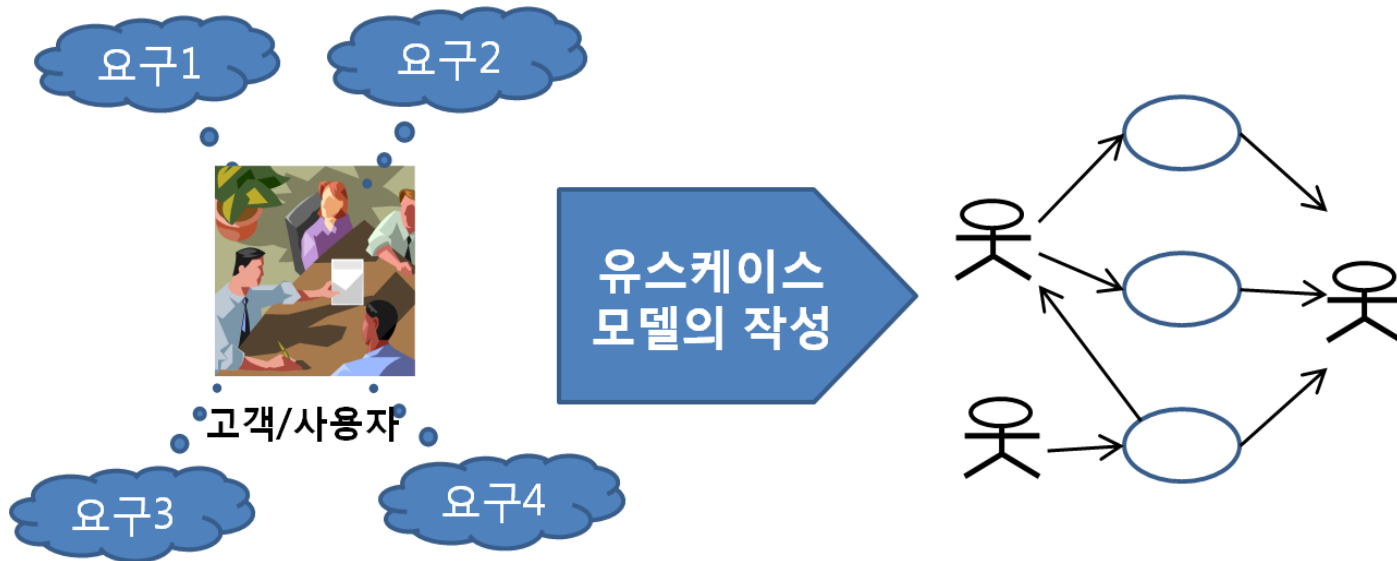
# UML 다이어그램 (1/2)

| 관점 | 다이어그램         | 설명                             |
|----|---------------|--------------------------------|
| 구조 | 클래스 다이어그램     | 클래스의 구조(속성과 연산)와 클래스 사이의 정적 관계 |
|    | 객체 다이어그램      | 특정 시점에서 객체의 상태와 객체 사이의 관계      |
|    | 패키지 다이어그램     | 패키지의 구성과 패키지 사이의 의존 관계         |
|    | 복합 구조 다이어그램   | 실행 시의 클래스 내부 구조                |
|    | 컴포넌트 다이어그램    | 컴포넌트의 구조와 의존 관계                |
|    | 배치 다이어그램      | 시스템에서의 물리적 배치                  |
| 행위 | 유스케이스 다이어그램   | 시스템에서 제공하는 기능과 사용자 사이의 관계      |
|    | 액티비티 다이어그램    | 작업의 순서와 병행성                    |
|    | 상태 다이어그램      | 객체의 상태와 이벤트에 의한 상태 전이          |
|    | 순차 다이어그램      | 객체 사이의 상호작용(시간 흐름 중심)          |
|    | 통신 다이어그램      | 객체 사이의 상호작용(객체 사이의 관계 중심)      |
|    | 타이밍 다이어그램     | 객체의 상호작용 타이밍                   |
|    | 상호작용 개요 다이어그램 | 순차 다이어그램과 액티비티 다이어그램의 개요       |

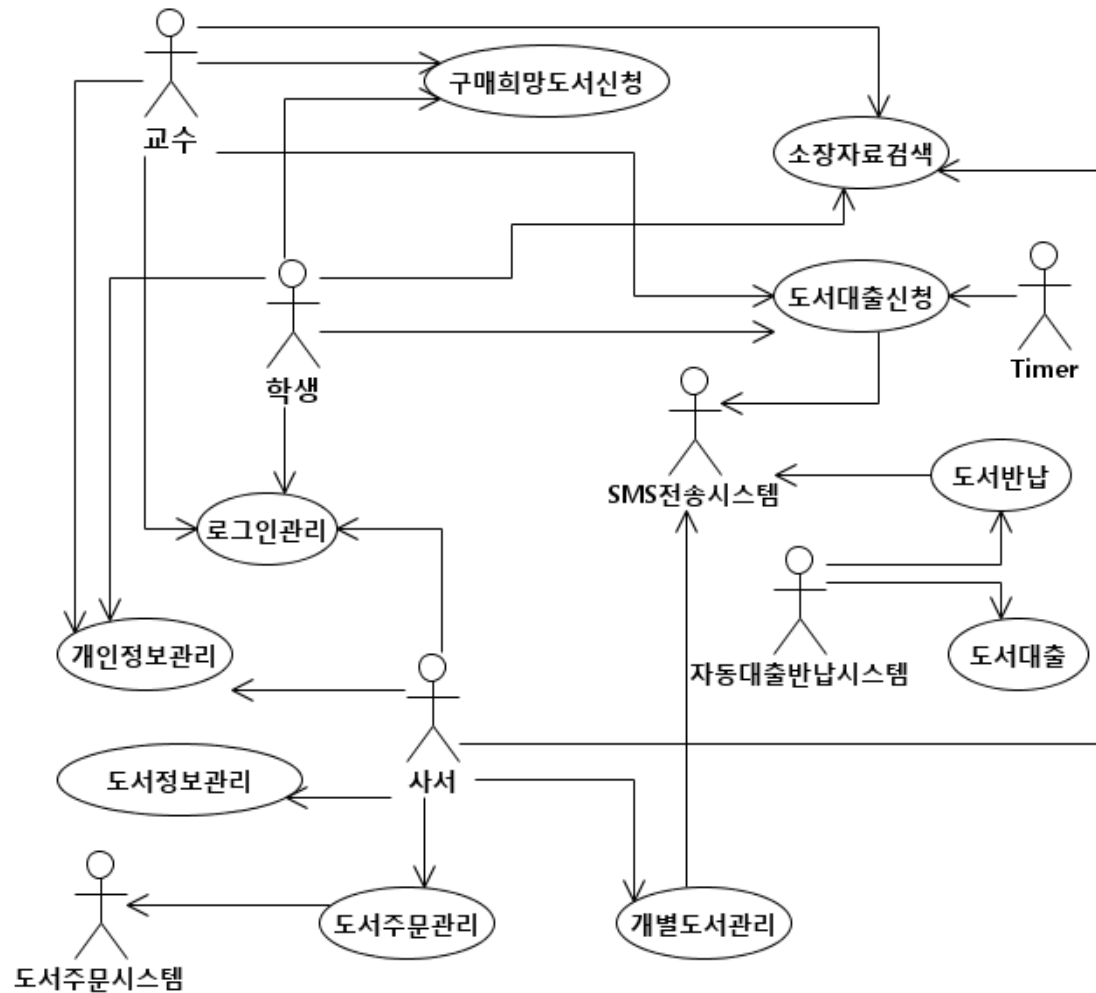
# UML 다이어그램 (2/2)



# 유스케이스 모델의 작성



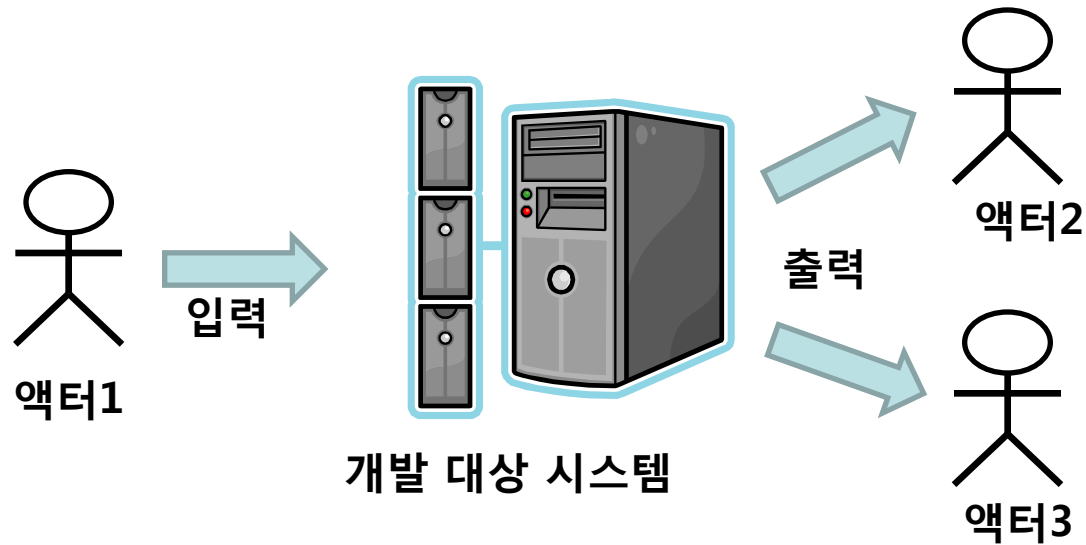
# 유스케이스 모델





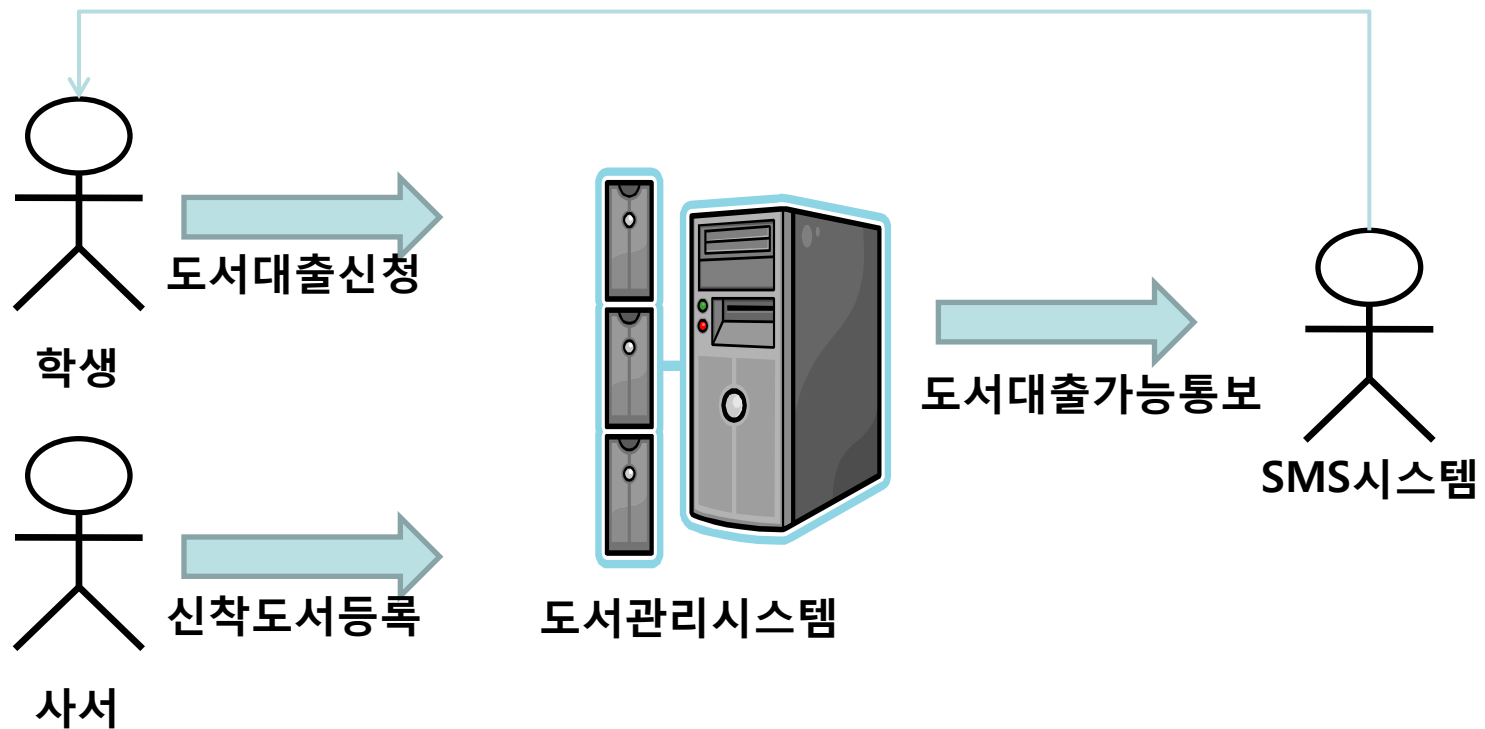
# 액터

- ❖ 액터는 시스템과 상호작용(interaction)을 하는 시스템 외부의 존재이다.



# 액터

## ❖ 예) 도서관리 시스템의 액터



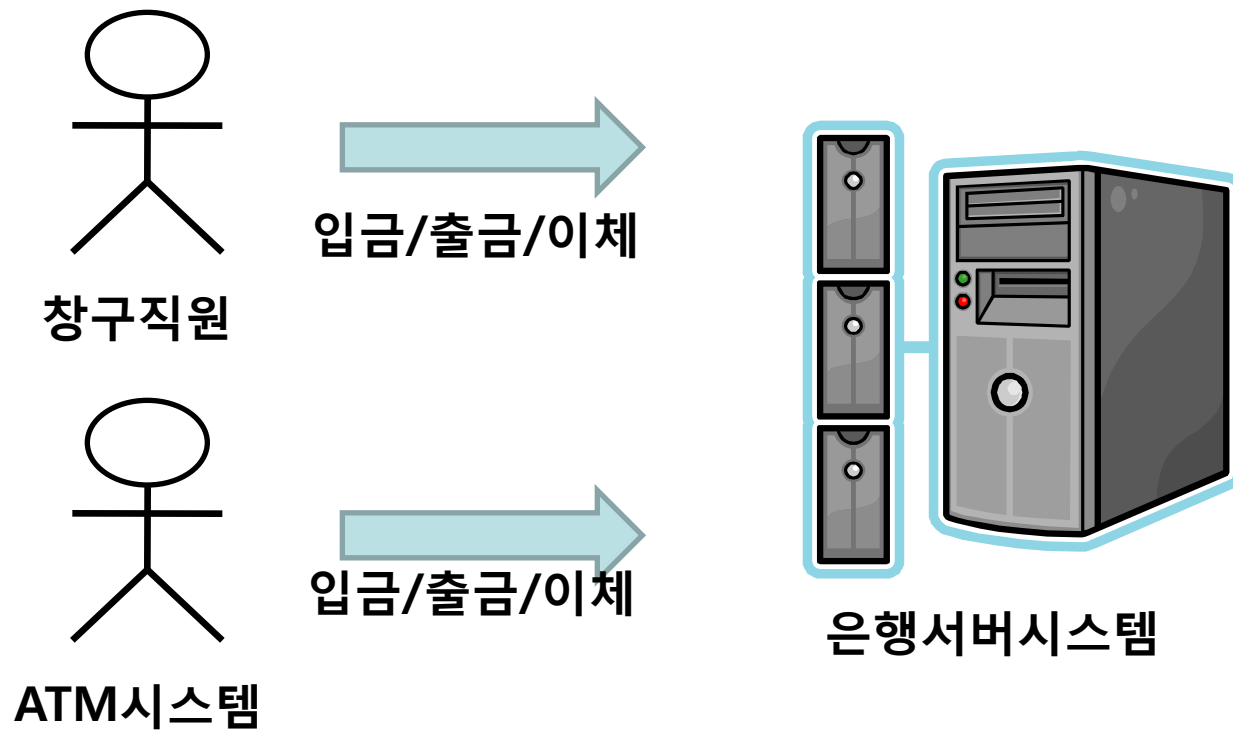
# 액터

- ❖ 액터는 개발 대상이 되는 시스템에 따라서 달라질 수 있다.
- ❖ 예) ATM시스템의 액터

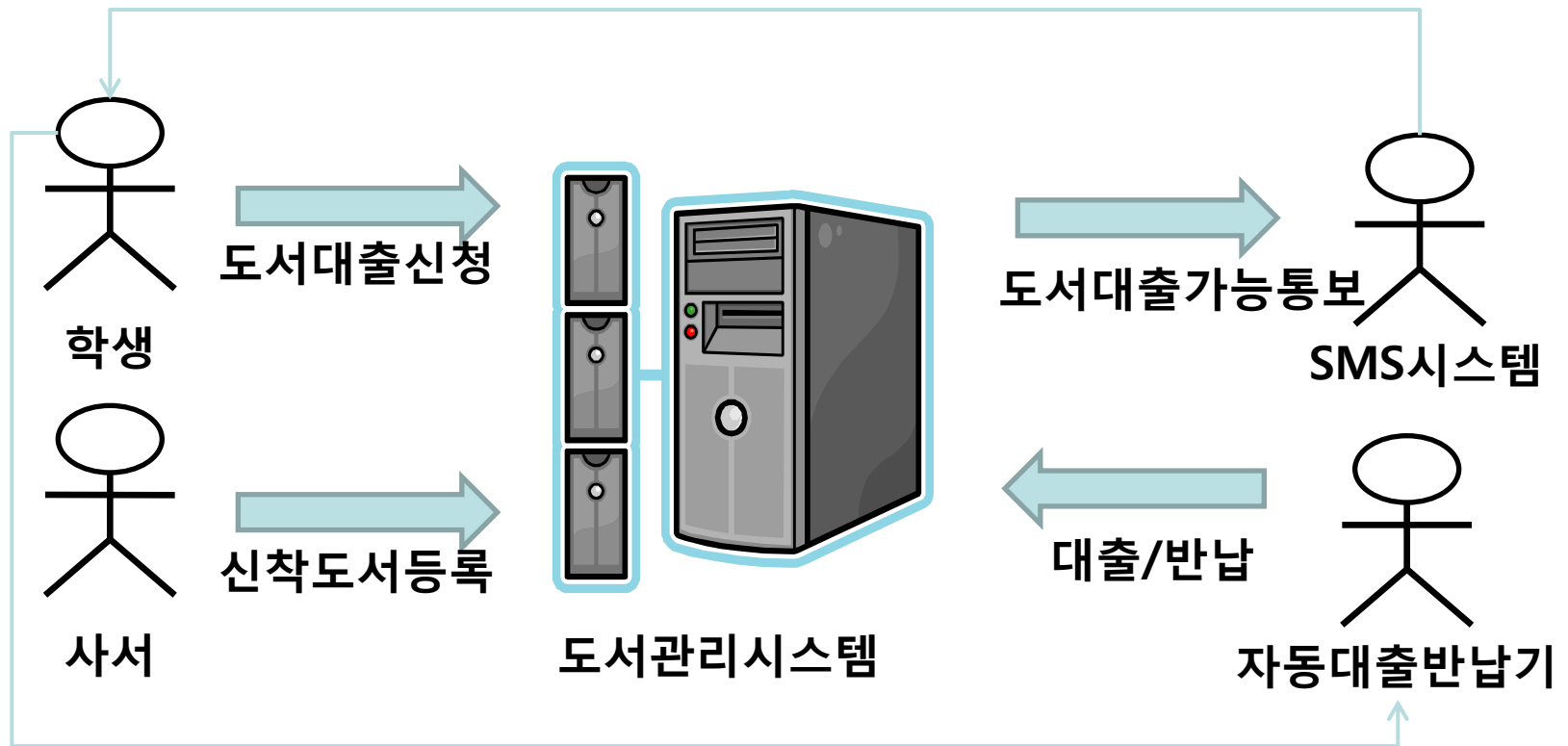


# 액터

❖ 예) 은행서버시스템의 액터

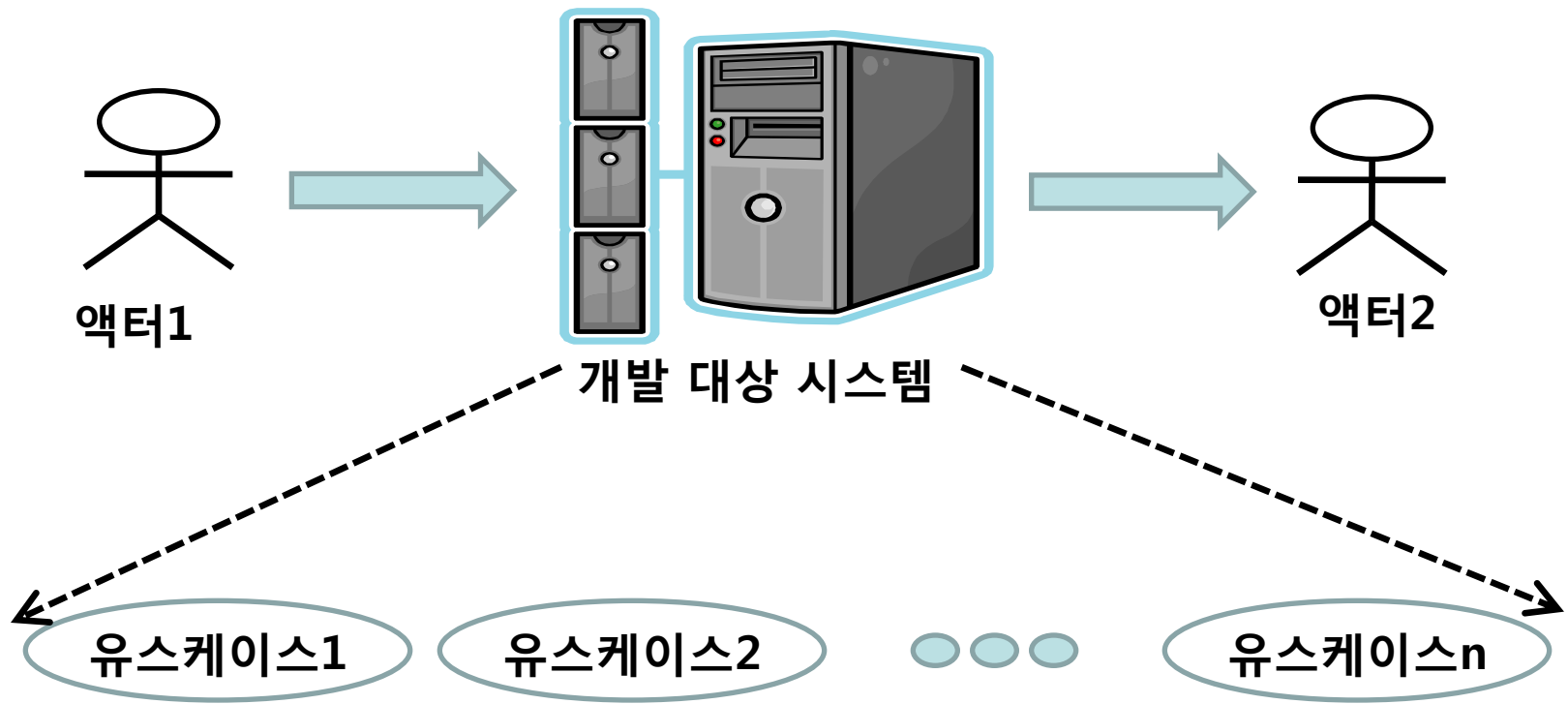


# 액터의 유형



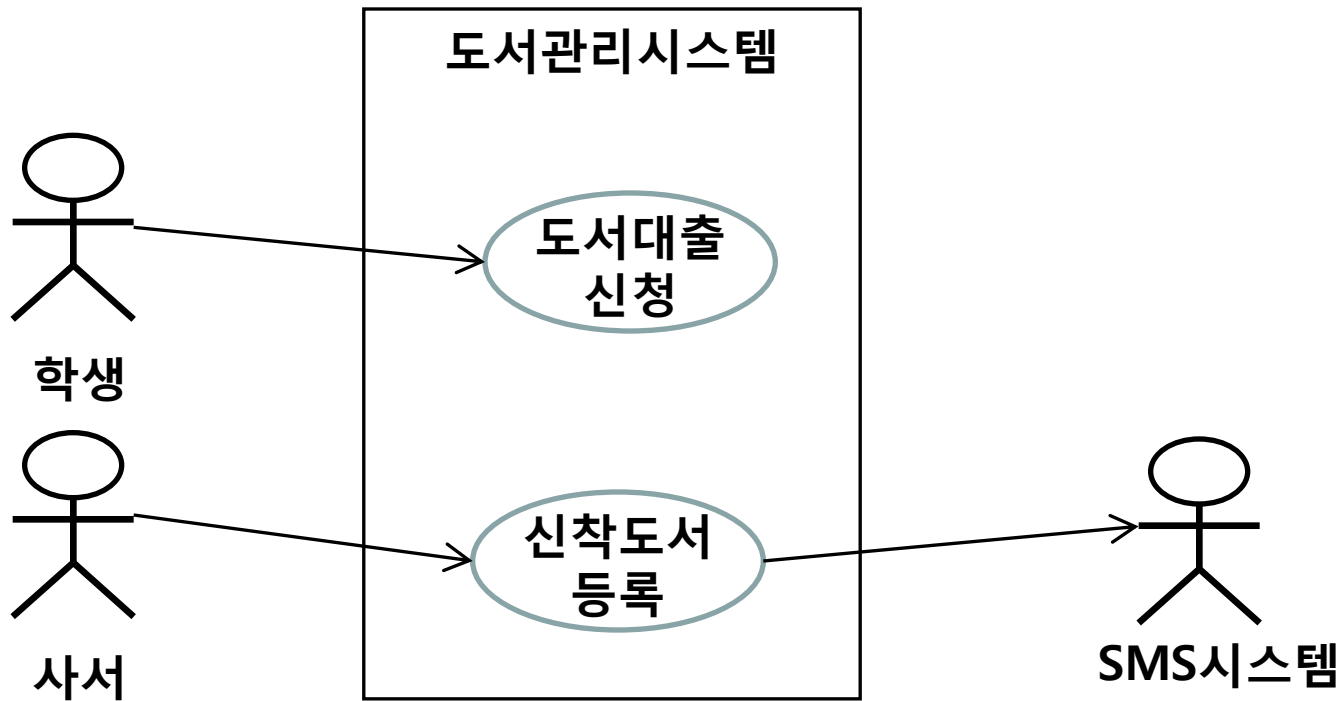
# 유스케이스

- ❖ 유스케이스 (usecase)는 개발 대상이 되는 시스템이 제공하는 개별적인 기능을 뜻한다



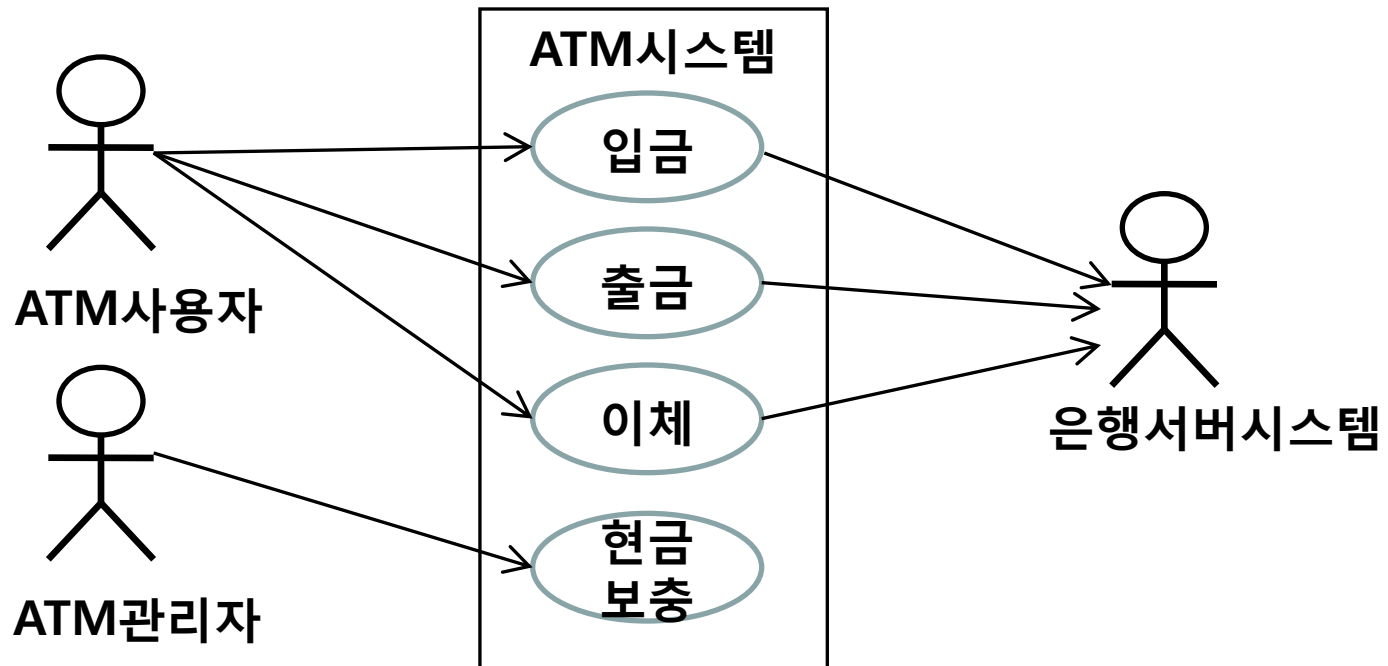
# 유스케이스

- ❖ 유스케이스로 표현된 기능은 시스템의 사용자가 이용한다.



# 유스케이스

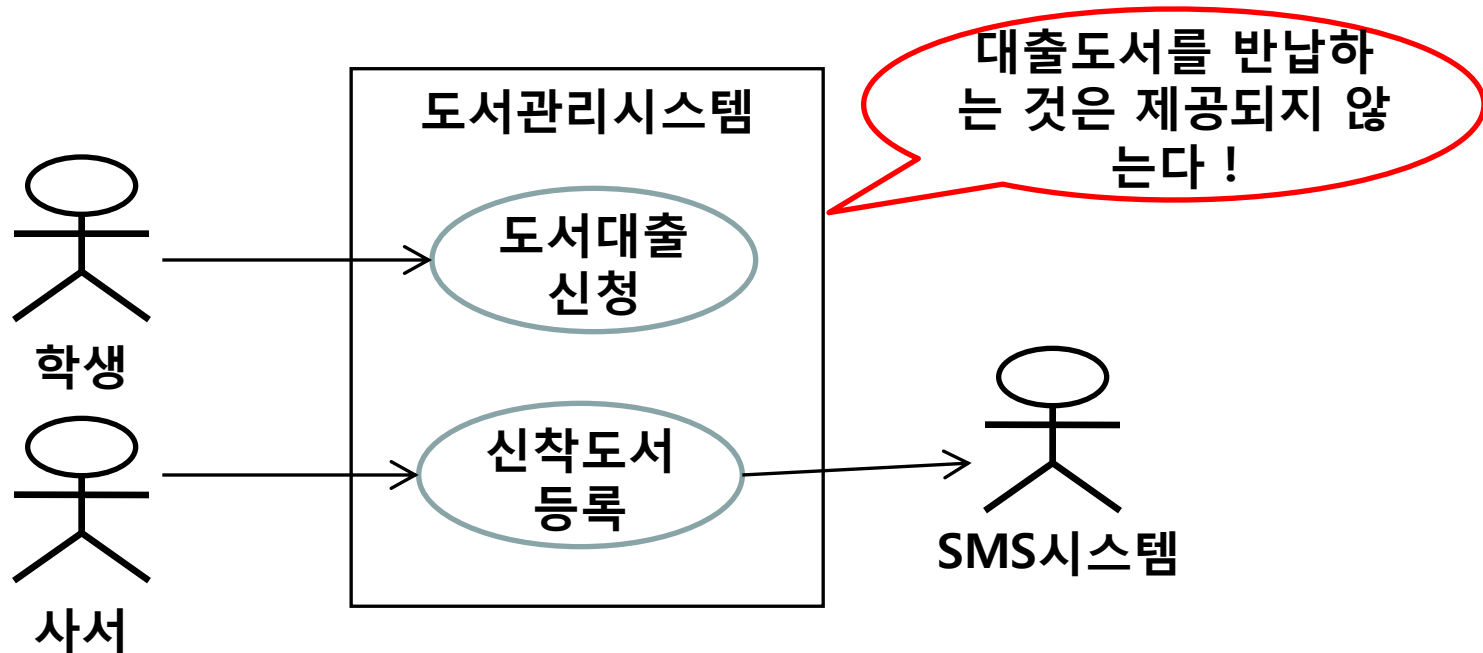
- ❖ 유스케이스 다이어그램에서는 유스케이스의 기능과 이를 이용하는 액터를 연관관계(association)를 이용하여 명시적으로 표현한다.





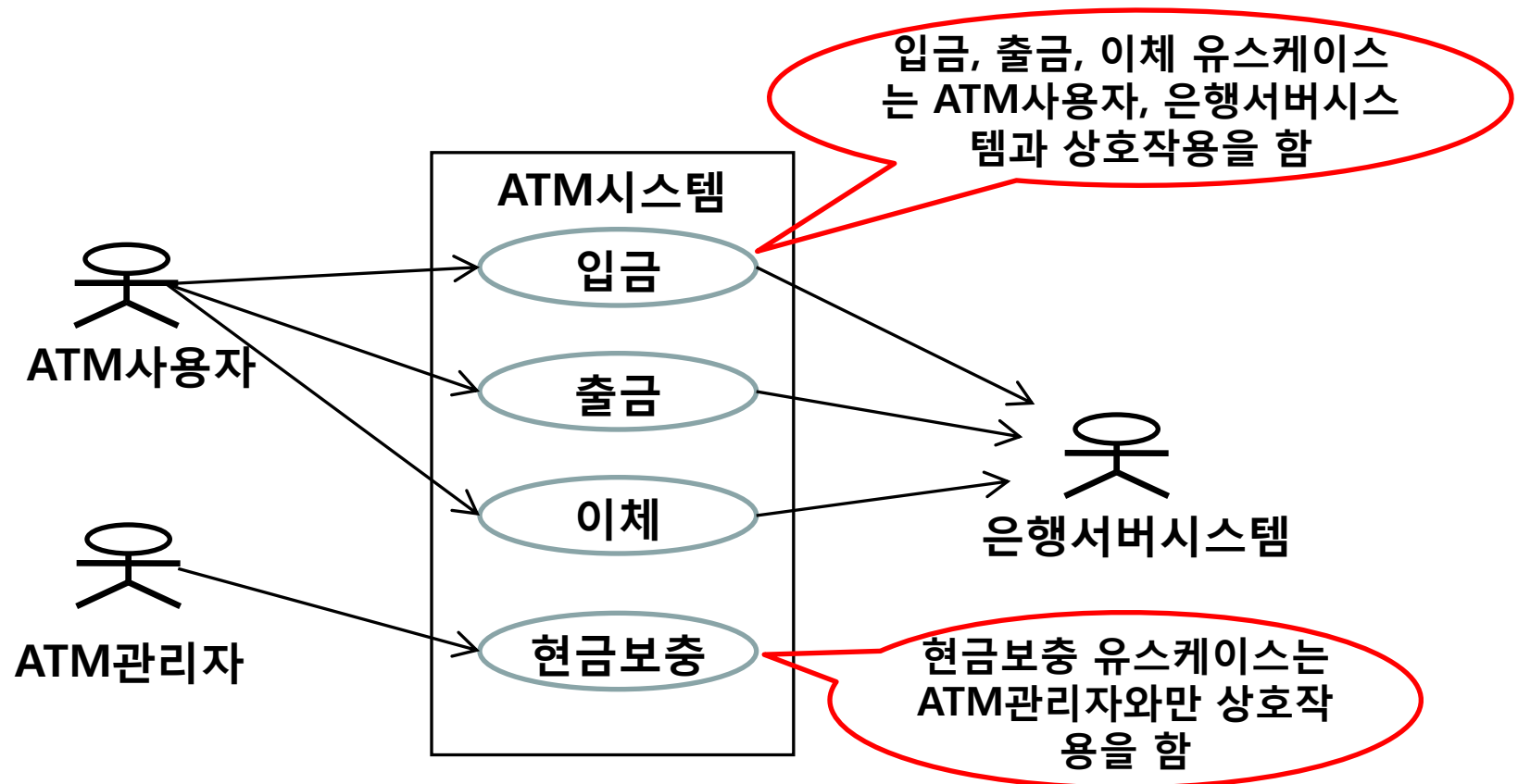
# 유스케이스

- ❖ 시스템의 전체 기능적 요구사항은 표현된 유스케이스로 구성된다.



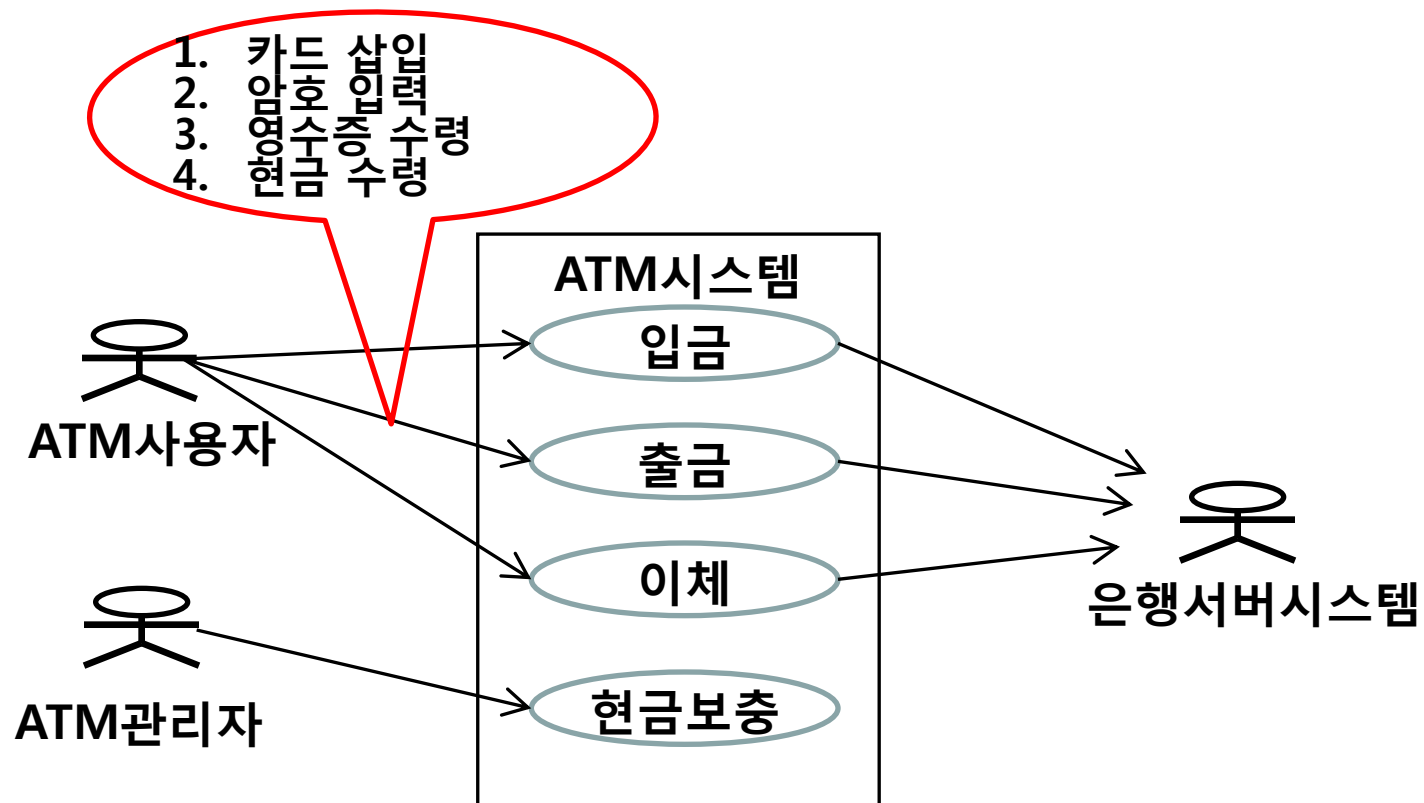
# 액터와 유스케이스 간의 관계

- ❖ 액터와 유스케이스 간의 연관 관계는 둘 간의 상호작용을 뜻한다.



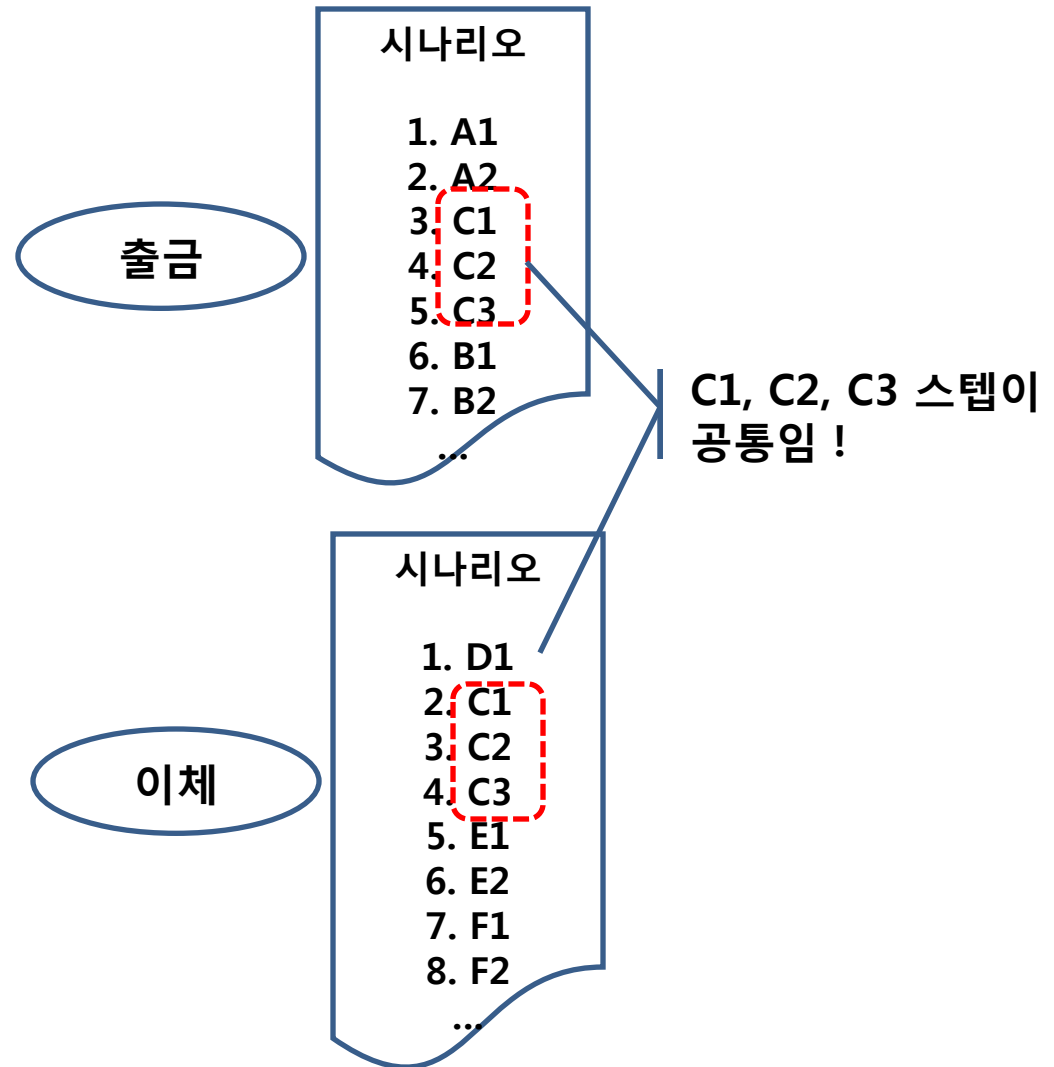
# 액터와 유스케이스 간의 관계

- ❖ 액터는 유스케이스와의 연관관계를 통하여 시스템과 다양한 상호작용을 한다.



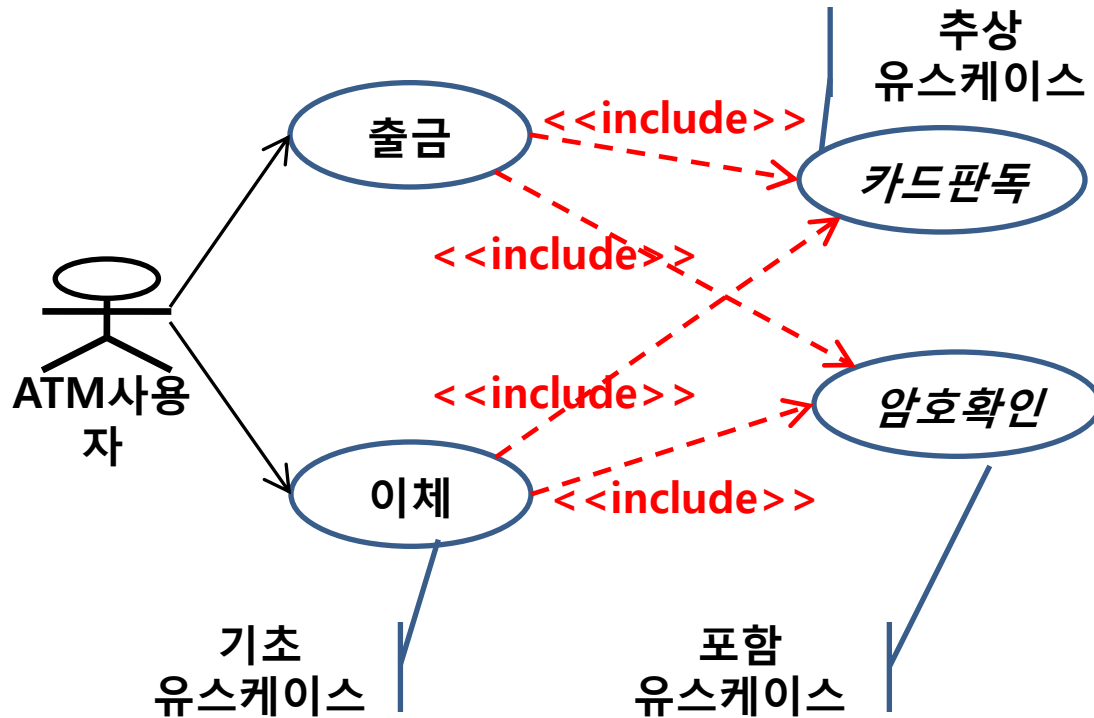
# 유스케이스 포함

- ❖ 유스케이스 포함이 필요한 상황



# 유스케이스 포함

- ❖ 포함 유스케이스는 두 개 이상의 유스케이스의 부분적인 공통 시나리오를 표현한다.



## 출금 유스케이스

1. ATM사용자는 카드입력 장치에 카드를 삽입한다.
2. 시스템은 은행서버시스템에게 카드판독을 요청한다.
3. 은행서버시스템은 카드판독 결과를 시스템에게 전달한다.
4. 시스템은 메뉴 화면을 출력한다.
5. ATM사용자는 "출금"을 선택한다.
6. 시스템은 암호 입력 화면을 출력한다.
7. ATM사용자는 암호를 입력한다.
8. 시스템은 은행서버시스템에게 암호 확인을 요청한다.
9. 은행서버시스템은 암호 확인 결과를 시스템에게 전달한다.
10. 시스템은 출금 금액 입력 화면을 출력한다.
11. ATM사용자는 인출금액을 입력한다.
12. 시스템은 은행서버시스템에게 출금요청을 한다.
13. 은행서버시스템은 요청된 출금에 대한 처리 결과를 시스템에게 통보한다.
14. 시스템은 카드와 지폐를 배출하고, 영수증은 인쇄한다.
15. ATM사용자는 카드, 지폐, 영수증을 수령한다.
16. 시스템은 지폐 배출 문을 닫는다.

## 이체 유스케이스

1. ATM사용자는 카드입력 장치에 카드를 삽입한다.
2. 시스템은 은행서버시스템에게 카드판독을 요청한다.
3. 은행서버시스템은 카드판독 결과를 시스템에게 전달한다.
4. 시스템은 메뉴 화면을 출력한다.
5. ATM사용자는 "이체"를 선택한다.
6. 시스템은 암호 입력 화면을 출력한다.
7. ATM사용자는 암호를 입력한다.
8. 시스템은 은행서버시스템에게 암호 확인을 요청한다.
9. 은행서버시스템은 암호 확인 결과를 시스템에게 전달한다.
10. 시스템은 이체 계좌 입력 화면을 출력한다.
11. ATM사용자는 이체 계좌 입력한다.
12. 시스템은 이체 금액 입력 화면을 출력한다.
13. ATM사용자는 이체금액을 입력한다.
14. 시스템은 은행서버시스템에게 이체요청을 한다.
15. 은행서버시스템은 요청된 이체에 대한 처리 결과를 시스템에게 통보한다.
16. 시스템은 카드와, 영수증은 인쇄한다.
17. ATM사용자는 카드, 영수증을 수령한다.

# 포함 유스케이스의 시나리오

---

## 카드판독 유스케이스

1. ATM사용자는 카드입력 장치에 카드를 삽입한다.
2. 시스템은 은행서버시스템에게 카드판독을 요청한다.
3. 은행서버시스템은 카드판독 결과를 시스템에게 전달한다.

## 암호확인 유스케이스

1. 시스템은 암호 입력 화면을 출력한다.
2. ATM사용자는 암호를 입력한다.
3. 시스템은 은행서버시스템에게 암호 확인을 요청한다.
4. 은행서버시스템은 암호 확인 결과를 시스템에게 전달한다.

## 출금 유스케이스

1. 카드판독 유스케이스를 포함한다.
2. 시스템은 메뉴 화면을 출력한다.
3. ATM사용자는 "출금"을 선택한다.
4. 암호확인 유스케이스를 포함한다.
5. 시스템은 출금 금액 입력 화면을 출력한다.
6. ATM사용자는 인출금액을 입력한다.
7. 시스템은 은행서버시스템에게 출금요청을 한다.
8. 은행서버시스템은 요청된 출금에 대한 처리 결과를 시스템에게 통보한다.
9. 시스템은 카드와 지폐를 배출하고, 영수증은 인쇄한다.
10. ATM사용자는 카드, 지폐, 영수증을 수령한다.
11. 시스템은 지폐 배출 문을 닫는다.

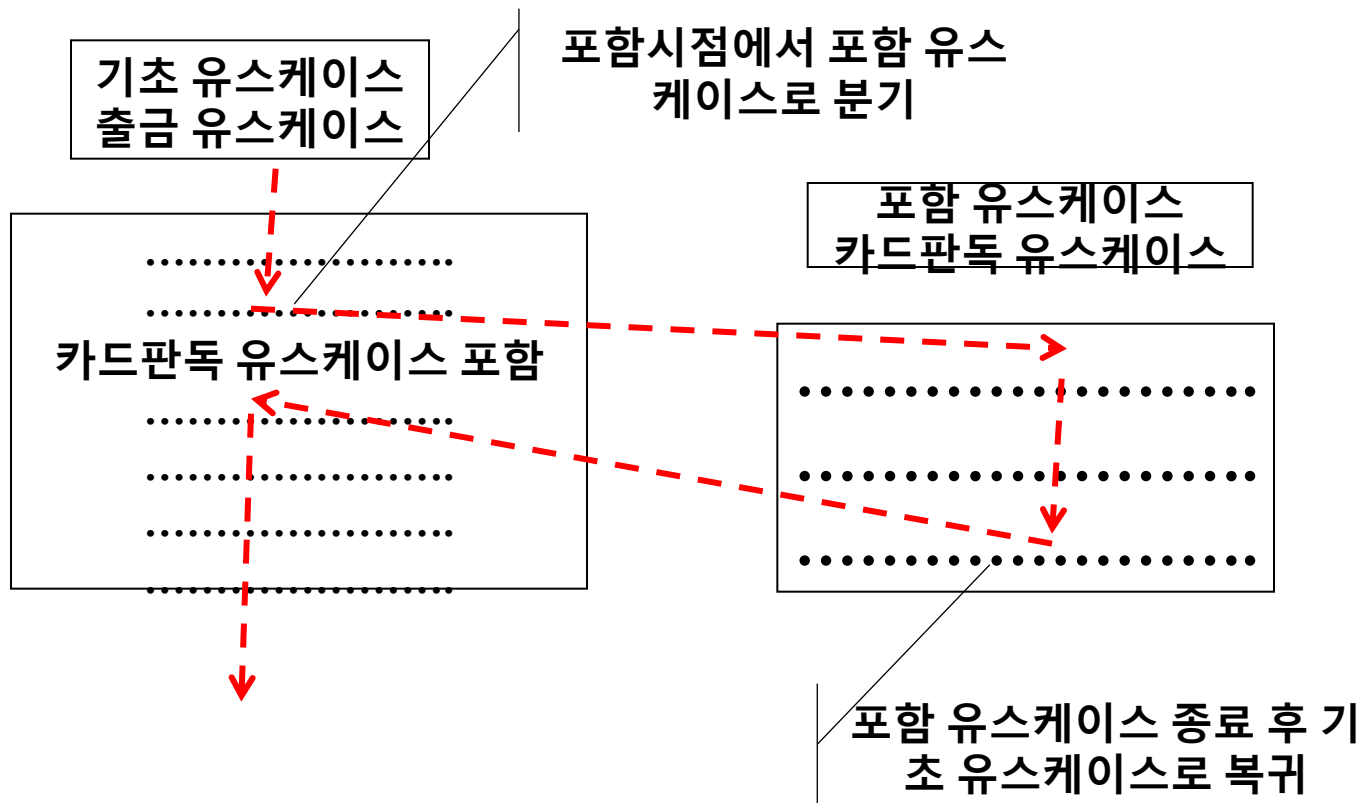
## 이체 유스케이스

1. 카드판독 유스케이스를 포함한다.
2. 시스템은 메뉴 화면을 출력한다.
3. ATM사용자는 "이체"을 선택한다.
4. 암호확인 유스케이스를 포함한다.
5. 시스템은 이체 계좌 입력 화면을 출력한다.
6. ATM사용자는 이체 계좌 입력한다.
7. 시스템은 이체 금액 입력 화면을 출력한다.
8. ATM사용자는 이체금액을 입력한다.
9. 시스템은 은행서버시스템에게 이체요청을 한다.
10. 은행서버시스템은 요청된 이체에 대한 처리 결과를 시스템에게 통보한다.
11. 시스템은 카드와, 영수증은 인쇄한다.
12. ATM사용자는 카드, 영수증을 수령한다.



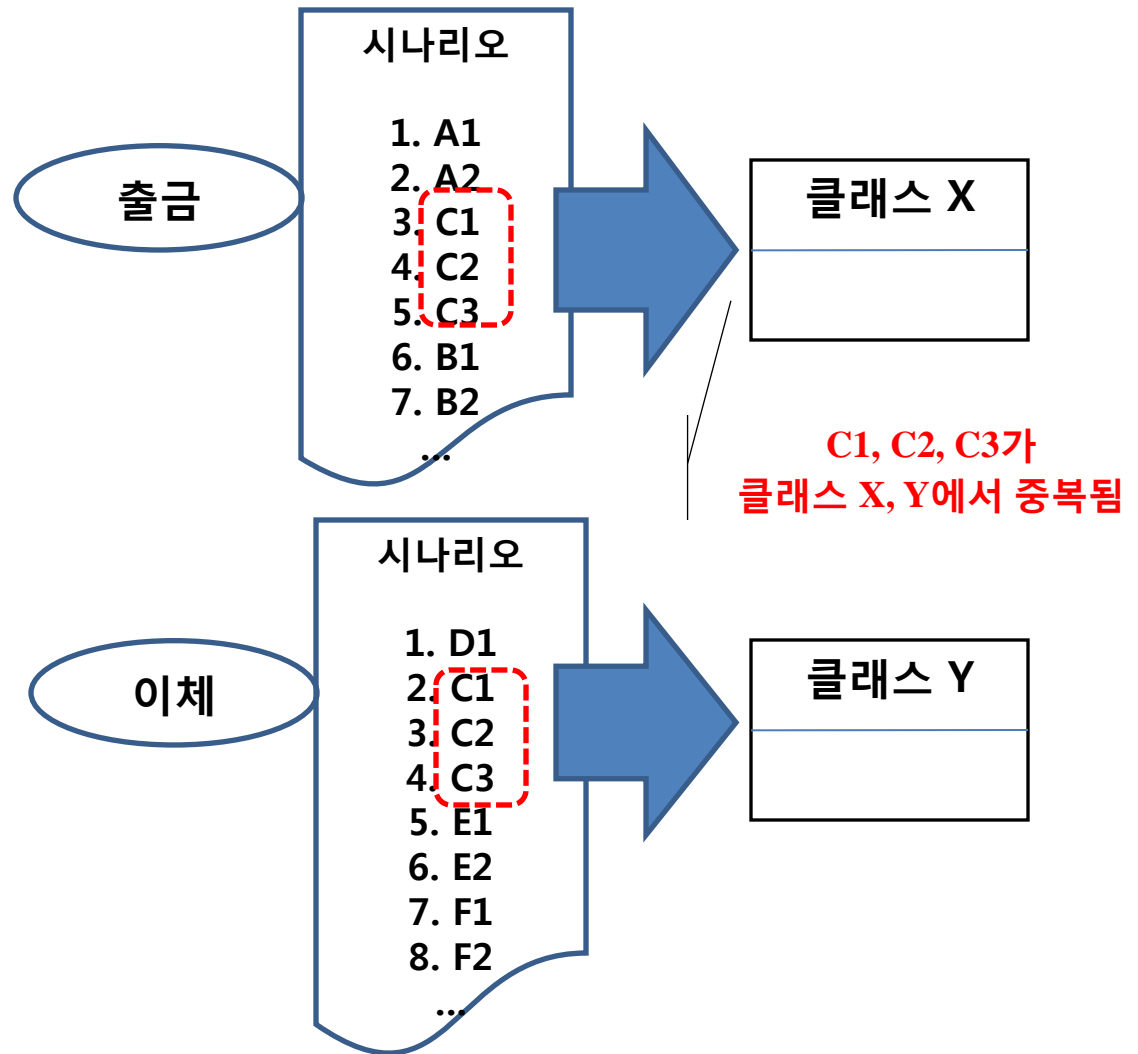
# 유스케이스 포함

## ❖ 유스케이스 포함의 시나리오 관계



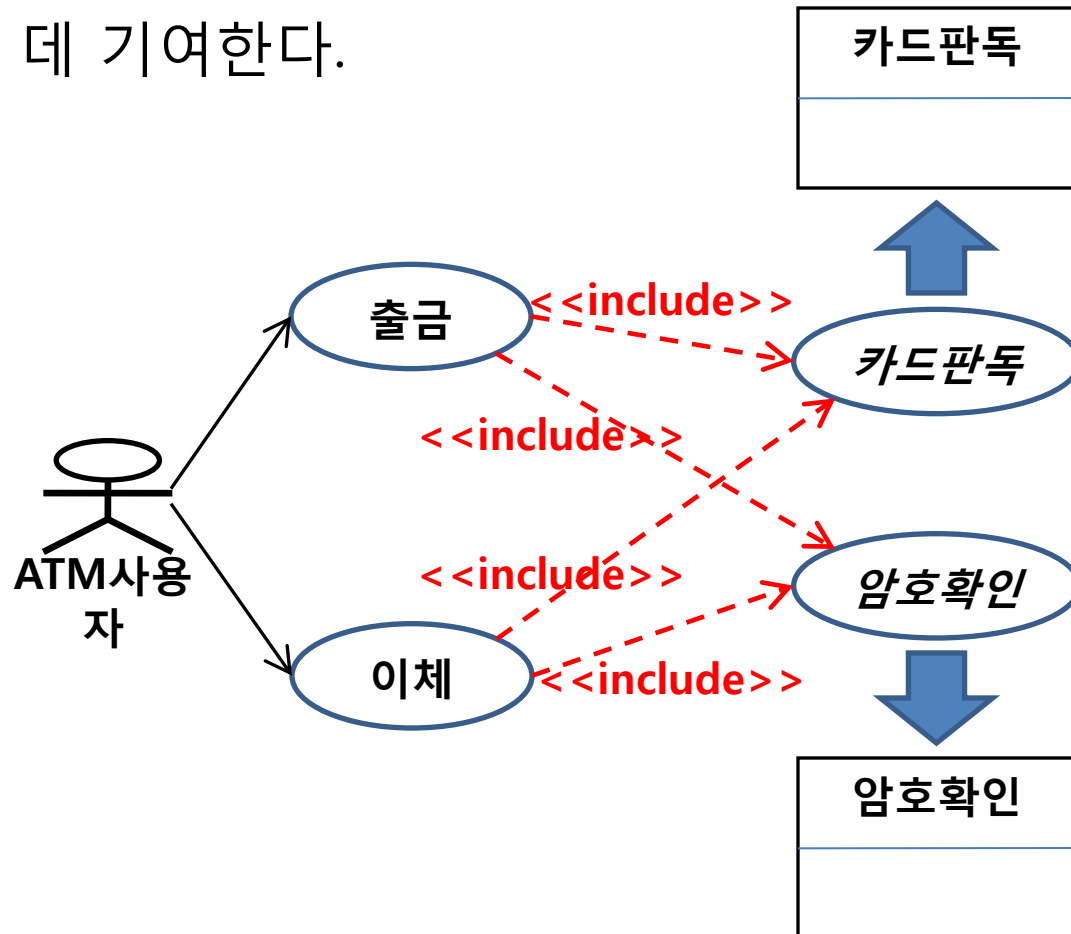
# 유스케이스 포함

- ❖ 포함 유스케이스를 정의하지 않을 때의 중복 문제

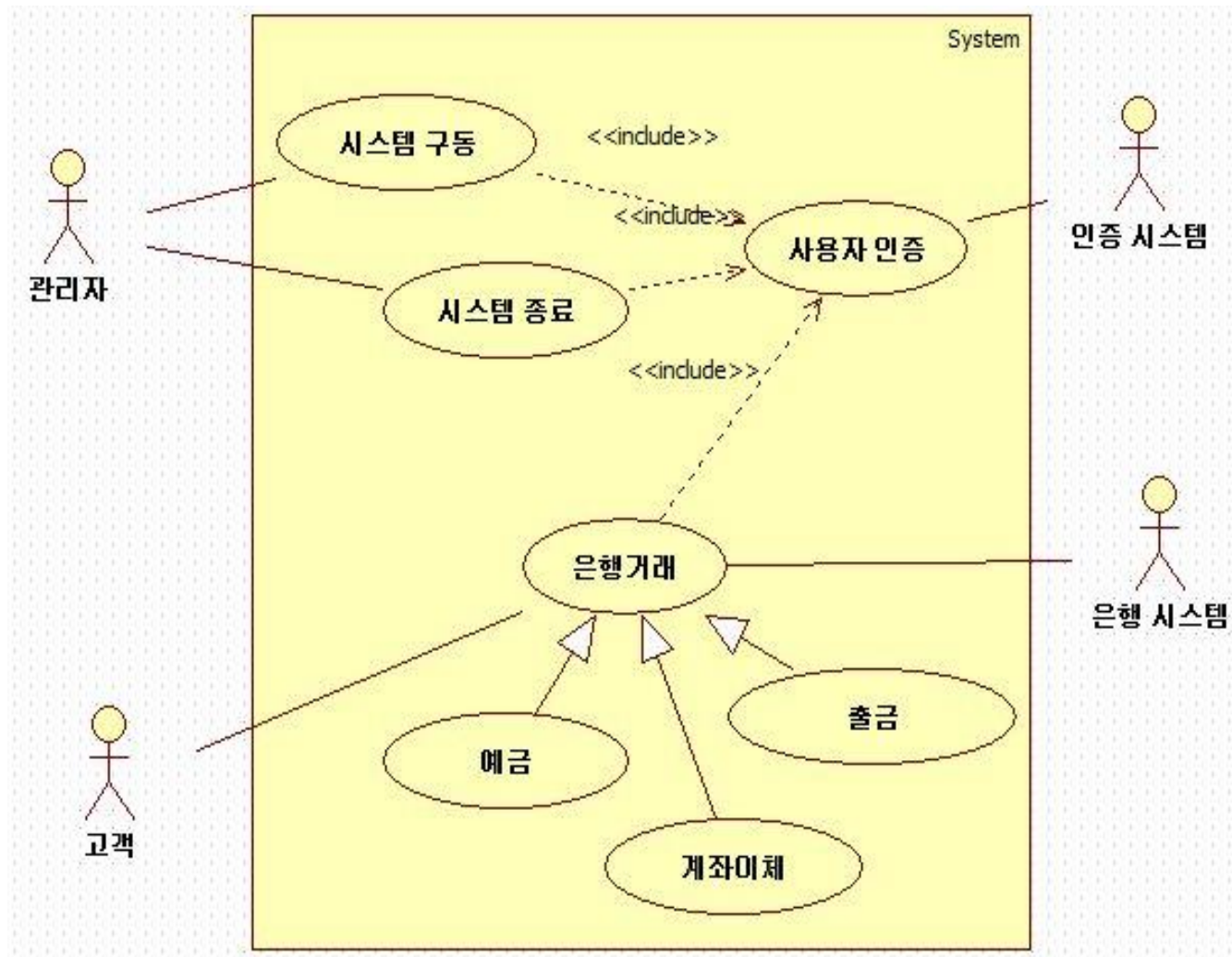


# 유스케이스 포함

- ❖ 포함 유스케이스는 유스케이스간의 모듈화를 통하여 재사용성을 높이는 데 기여한다.



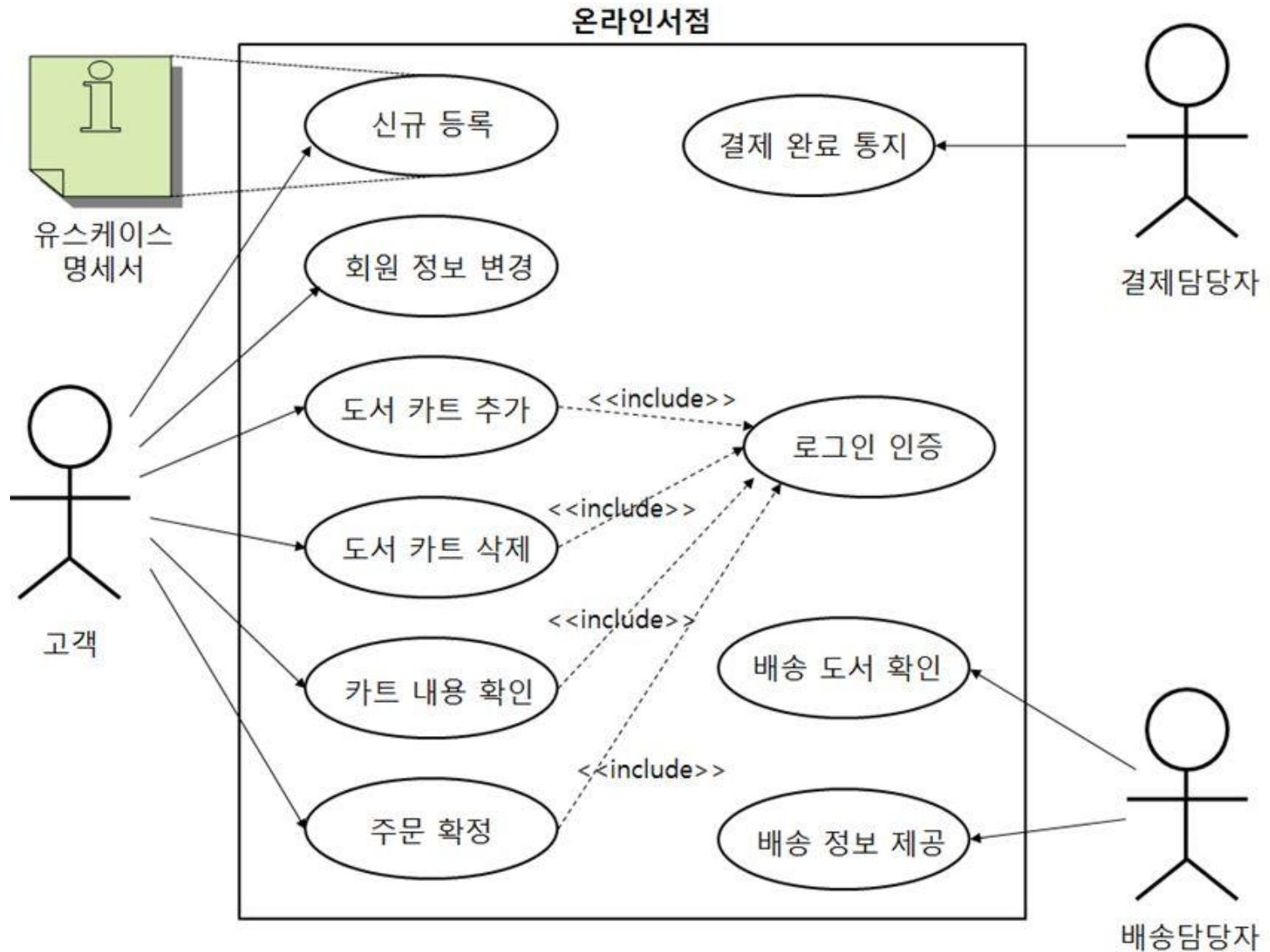
# 유스케이스 다이어그램의 예



# 유스케이스의 관계

| 관계     | 설명   |
|--------|--|
| 포함관계   | 여러 유스케이스에서 중복되는 단계가 있을 경우 이를 떼어내어 새로운 유스케이스를 만든다. 원래의 유스케이스에서 새로 만들어진 유스케이스 쪽으로 점선 화살표로 연결하고 <<include>>라고 표시한다.   |
| 확장관계   | 유스케이스가 특정 조건이 만족되는 경우에만 실행되는 단계가 있는 경우, 이를 단계를 독립시켜 별도의 유스케이스로 만들고, 원래의 유스케이스 방향으로 점선 화살표로 연결하고 <<extend>>라고 표시한다. |
| 일반화 관계 | 일반적인 유스케이스와 구체적인 유스케이스 사이에 존재하는 관계이다. 구체적인 유스케이스에서 일반적인 유스케이스 방향으로 속이 빈 삼각 화살표로 연결한다.                              |

# 유스케이스 다이어그램



# 유스케이스 명세서

명칭      주문 확정

액터      고객

목적      카트에 담겨있는 도서를 구입한다.

사전조건    고객은 서점에 재고로 남아 있는 도서를 구입한다.

## 정상처리 시나리오

1. 고객이 주문확정버튼을 누른다.
2. 시스템은 카트 속의 도서들에 대한 합계금액을 계산한다.
3. 시스템은 카트의 내용과 합계금액을 표시한다.
4. 고객이 신용카드번호를 입력하고, 구입버튼을 누른다.
5. 시스템은 결제담당자에게 고객의 결제상황을 확인한다.
6. 결제가 성공하면, 결제 완료로 통지한다.
7. 시스템은 배송담당자에게 도서 배송을 의뢰한다.
8. 시스템은 카트를 비운다.
9. 시스템은 도서 재고를 갱신한다.

## 예외처리

5에서 결제가 실패한다.

- a. 시스템은 결제 실패 메시지를 표시한다.

# 유스케이스 명세서: 양식

| 항목           | 설명                          |                            |                              |
|--------------|-----------------------------|----------------------------|------------------------------|
| 개요           | 유스케이스에 대한 간략한 설명            |                            |                              |
| 관련 액터        | 주 액터                        | 유스케이스를 활성화시키는 액터           |                              |
|              | 보조 액터                       | 유스케이스와 관련된 주 액터를 제외한 모든 액터 |                              |
| 우선 순위        | 개발의<br>우선순위                 | 중요도                        | 이 기능이 얼마나 사용자에게 중요한가?        |
|              |                             | 난이도                        | 개발자가 이 기능을 개발하는 것이 얼마나 어려운가? |
| 선행 조건        | 유스케이스의 수행 이전에 만족이 되어야 하는 조건 |                            |                              |
| 후행 조건        | 유스케이스의 수행 후에 만족이 되어야 하는 조건  |                            |                              |
| 시나리오         | 기본 시나리오                     | 액터와 시스템 간의 기본/정상 시나리오      |                              |
|              | 대안 시나리오                     | 액터와 시스템 간의 예외/선택 시나리오      |                              |
| 비기능적<br>요구사항 | 유스케이스와 관련된 비기능적 요구사항        |                            |                              |



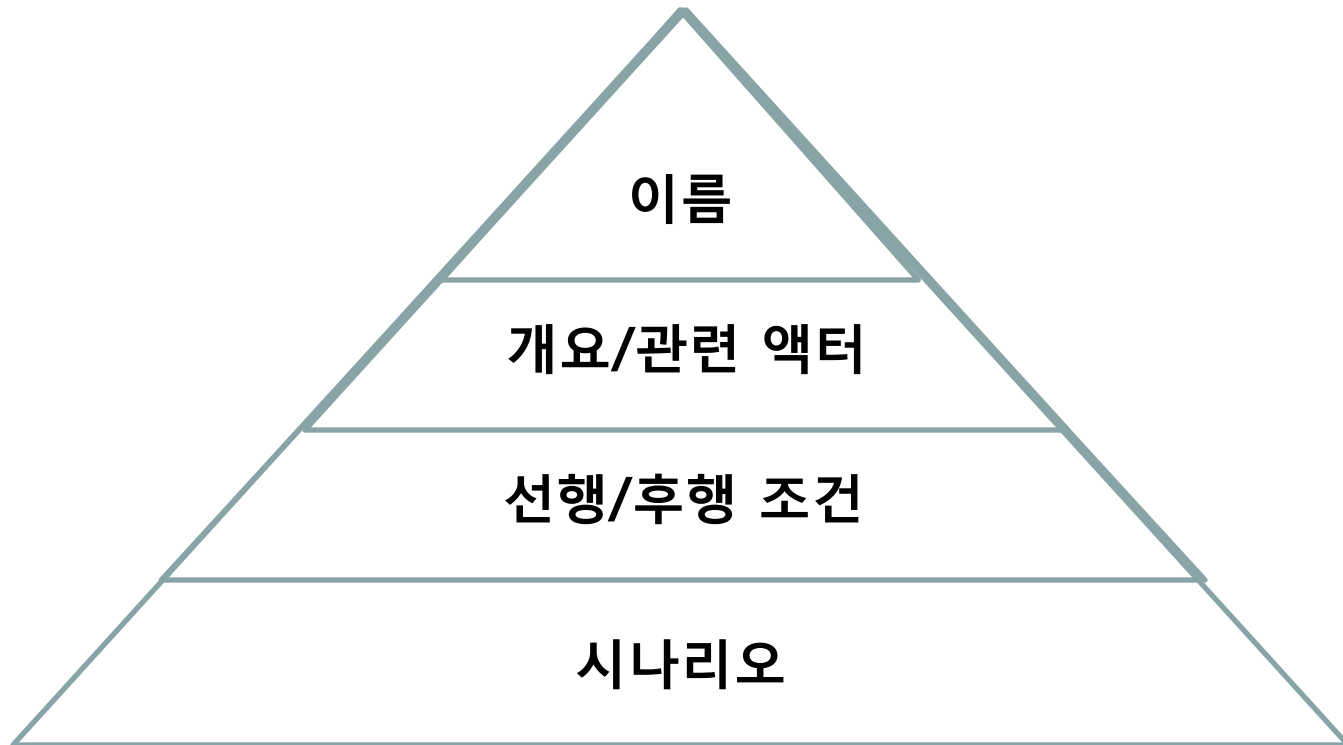
# 유스케이스 명세서: 항목

---

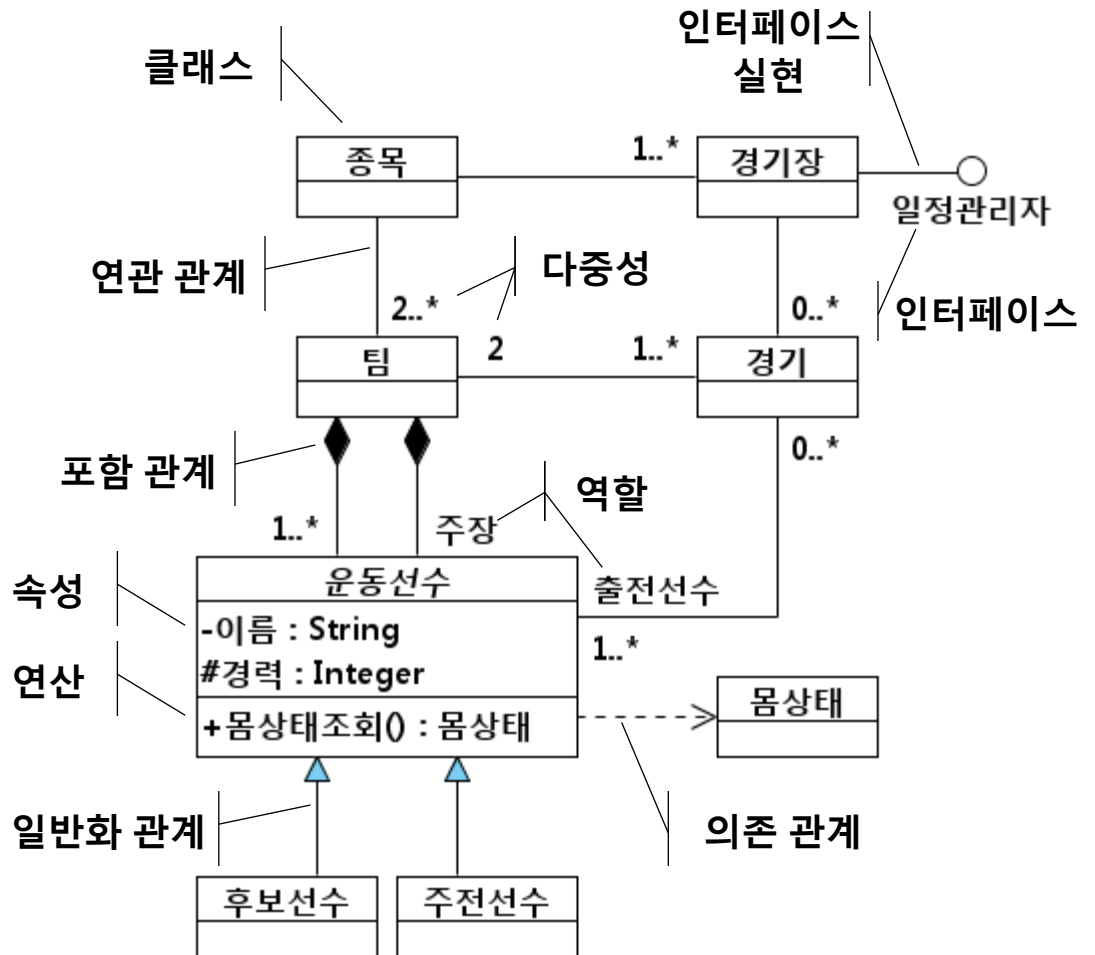
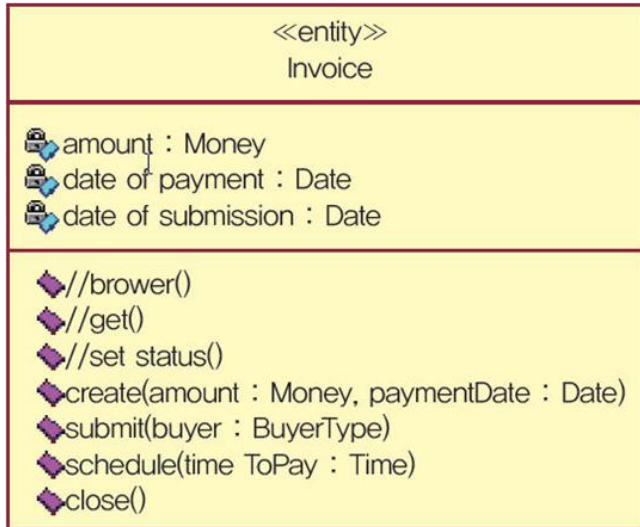
- ❖ 유스케이스 이름: 유스케이스에 대한 가장 간결한 명세로서 유스케이스를 통하여 제공되는 시스템의 기능을 명확한 동사구 형태로 표현해야 한다.
- ❖ 개요: 모든 이해관계자는 가장 먼저 유스케이스 명세서의 개요 항목을 통해서 유스케이스를 이해한다.
- ❖ 관련 액터 항목: 해당 유스케이스가 동작할 때 필요한 주변 액터를 이해할 수가 있다. 주 액터와 보조 액터를 구분하여 기록 한다.
- ❖ 우선 순위: 기능의 중요도와 개발 난이도를 바탕으로 결정된다. 개발 순서, 자원 투입 등과 같이 프로젝트 관리 측면에서 활용된다.
- ❖ 선행조건: 유스케이스의 시작 시 만족되어야 할 조건으로서 만족되지 않으면 유스케이스는 시작되지 않는다.
- ❖ 후행 조건: 유스케이스의 종료 시 만족해야 하는 조건으로 유스케이스의 정상 동작 여부에 대한 최소한의 판단 기준으로 사용될 수 있다.
- ❖ 시나리오: 유스케이스의 관련 액터와 시스템 간의 상호작용에 대한 구체적인 정의를 담고 있다. 하나의 유스케이스에는 기본 시나리오와 대안 시나리오로 구분하여 기술한다.
- ❖ 비기능적 요구사항: 성능, 신뢰도, 보안 등 이 유스케이스와 관련된 비기능적 요구사항을 기술한다.

# 유스케이스 명세서 항목의 상세화 수준

---



# 클래스 다이어그램



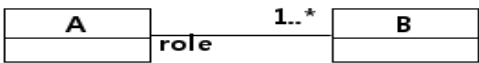
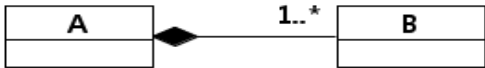

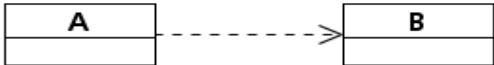
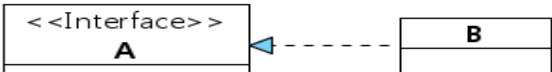
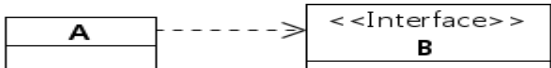
# 클래스의 접근 제어자

---

| 접근 제어자    | 기호 | 설명                                   |
|-----------|----|--------------------------------------|
| public    | +  | 모든 클래스에서 접근 가능                       |
| protected | #  | 동일 패키지에 있거나 상속관계의 하위 클래스의 객체들만 접근 가능 |
| package   | ~  | 동일 패키지에 있는 클래스의 객체들만 접근 가능           |
| private   | -  | 해당 클래스로부터 생성된 객체들만 접근 가능             |

# 클래스 다이어그램의 요소

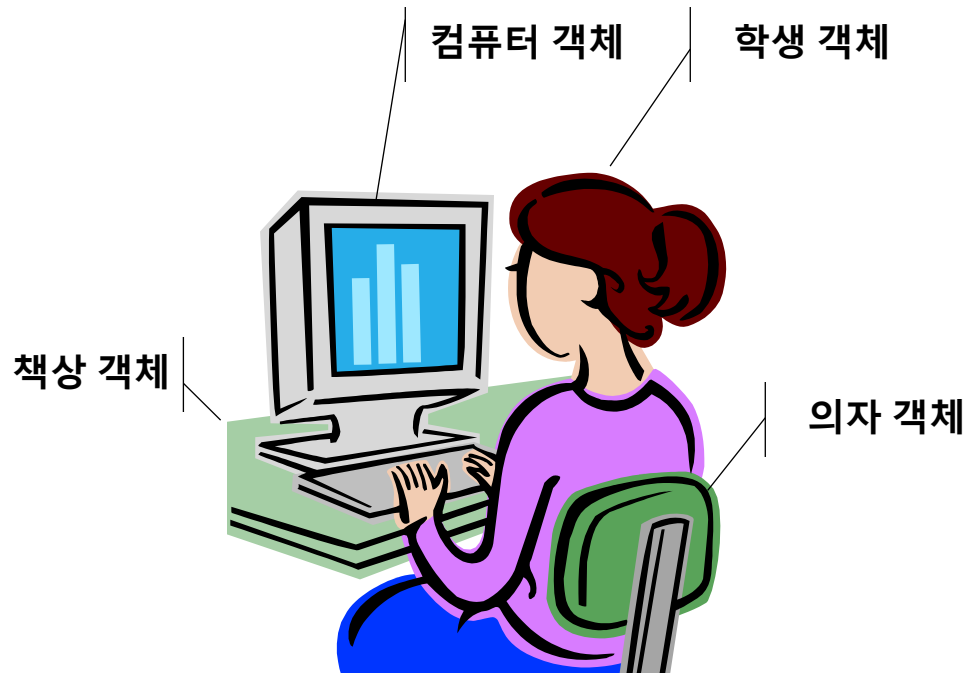
## ❖ 클래스 다이어그램의 관계

| 관계          | 표기법  | 의미                           |
|-------------|--|------------------------------|
| 연관 관계       |     | 클래스 A와 클래스 B는 연관 관계를 가지고 있다. |
| 포함 관계       |     | 클래스 B는 클래스 A의 부분이다.          |
| 일반화 관계      |    | 클래스 B는 클래스 A의 하위 클래스이다.      |
| 의존 관계       |    | 클래스 A는 클래스 B에 의존한다.          |
| 인터페이스 실현 관계 |  | 클래스 B는 인터페이스 A를 실현한다.        |
| 인터페이스 의존 관계 |  | 클래스 A는 인터페이스 B에 의존한다.        |

# 객체는 상태와 행동을 가지는 개별적인 실체이다.

---

- ❖ 객체는 우리가 실 세계에서 인지할 수 있는 개별적인 실체(instance)이다.



# 객체는 상태와 행동을 가지는 개별적인 실체이다.

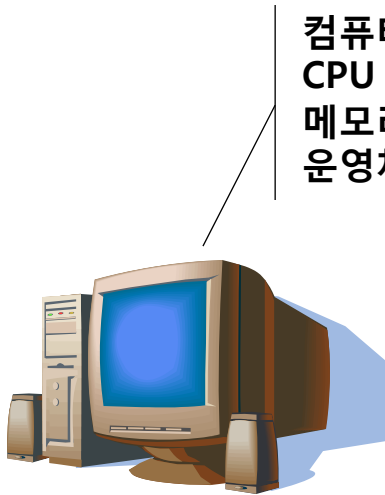
- ❖ 객체는 자신의 특성과 정보를 상태(state)로서 가지며 이를 바탕으로 한 행위(behavior)를 외부에 제공한다.
  - 상태(state)는 객체의 특성을 표현하는 개별적인 객체에 대한 정보이다.
  - 행위(behavior)는 객체가 자신의 상태를 바탕으로 외부에 제공하는 기능을 뜻한다.

| 객체 | 컴퓨터 객체  | 학생 객체  |
|----|---|--|
| 상태 | CPU ="Intel Core 2 Duo"<br>메모리크기=2046MB<br>운영체제="Windows Vista" | 이름="박혜자"<br>학교="강릉원주대"<br>학과="컴퓨터공학과"<br>학년=4    |
| 행위 | 컨다()<br>실행()<br>끝다()  | 시험을본다()<br>보고서를제출한다()<br>먹는다()<br>잔다()<br>운동한다() |

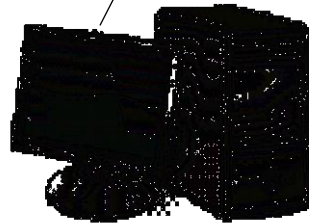
# 객체는 상태와 행동을 가지는 개별적인 실체이다.

## ❖ 객체는 개별적인 존재이다.

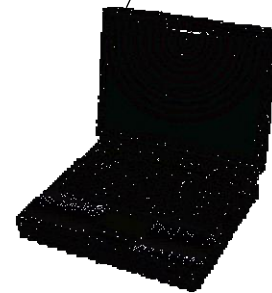
- 각 객체는 개별적인 존재이므로 서로 다른 상태를 가질 수가 있다
- 상태가 동일하다고 해서 동일한 하나의 객체는 아니다.



컴퓨터1 객체  
CPU ="Intel Core 2 Duo"  
메모리크기=4GB  
운영체제="Windows Vista"



컴퓨터2 객체  
CPU ="Intel Core 2 Duo"  
메모리크기=4GB  
운영체제="Windows Vista"



컴퓨터3 객체  
CPU ="Intel Core 2 Duo"  
메모리크기=1GB  
운영체제="Windows XP"


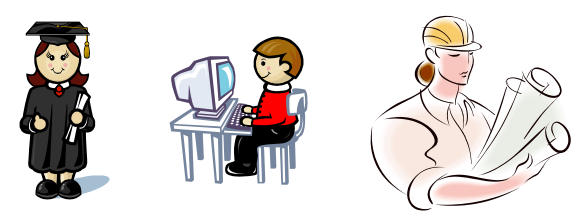



# 클래스는 유사한 객체들의 묶음이다

---



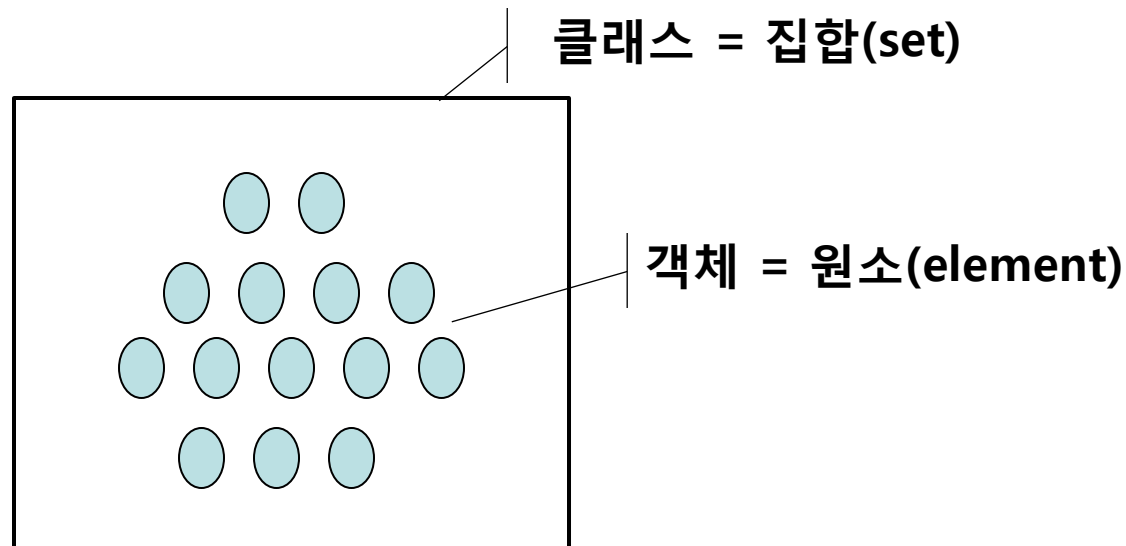
# 클래스는 유사한 객체들의 묶음이다

| 해당 객체   | 공통점                       |   | 클래스  |
|---|---------------------------|---|------|
|    | 사람이 이동을 위해서 탈 수 있는 교통 수단들 | ➡ | 교통수단 |
|   | 다양한 유형의 사람들               | ➡ | 사람   |
|  | 살아 있는 생물들                 | ➡ | 생물   |

# 클래스는 유사한 객체들의 묶음이다

---

- ❖ 클래스는 유사한 특성 즉 유사한 상태와 행위를 가지는 객체들을 한꺼번에 부르는 용어이다.
- ❖ 클래스는 마치 집합과 유사하다.



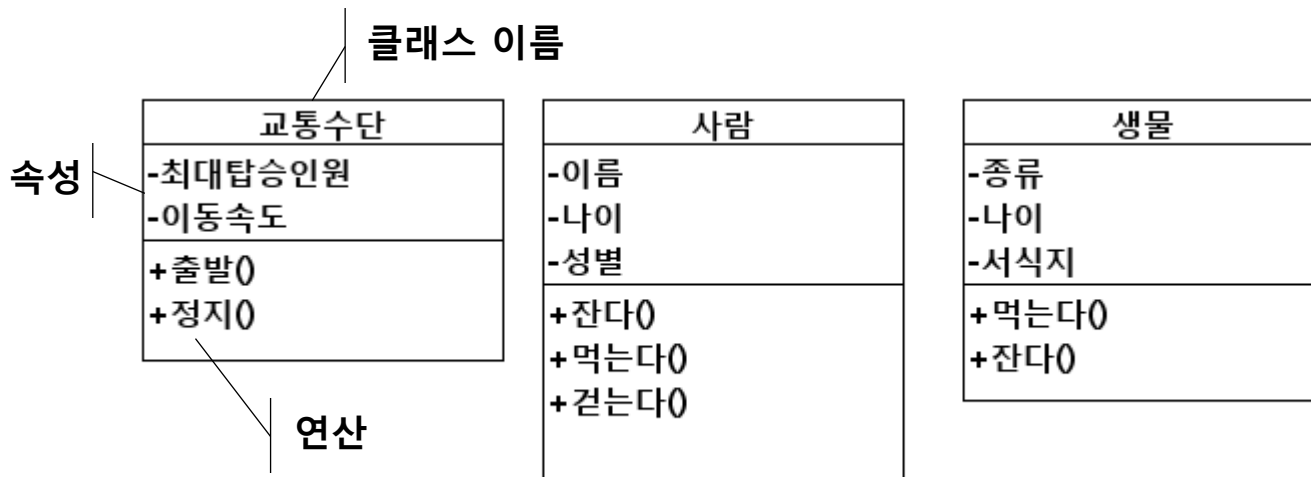
# 클래스는 유사한 객체들의 묶음이다

---

| 객체      |   | UML | Java | C++    |
|---------|---|-----|------|--------|
| 유사한 객체들 |  | 클래스 | 클래스  | 클래스    |
| 상태      |  | 속성  | 필드   | 데이터 멤버 |
| 행위      |  | 연산  | 메소드  | 멤버 함수  |

# 클래스는 유사한 객체들의 묶음이다

## ❖ 클래스의 표현 - 클래스 다이어그램



# 클래스는 유사한 객체들의 묶음이다

---

## ❖ 클래스의 표현 – Java

```
class 교통수단 {  
    private int 최대탑승인원 ;  
    private float 이동속도 ;  
    public void 출발() { ... }  
    public void 정지() { ... }  
}
```

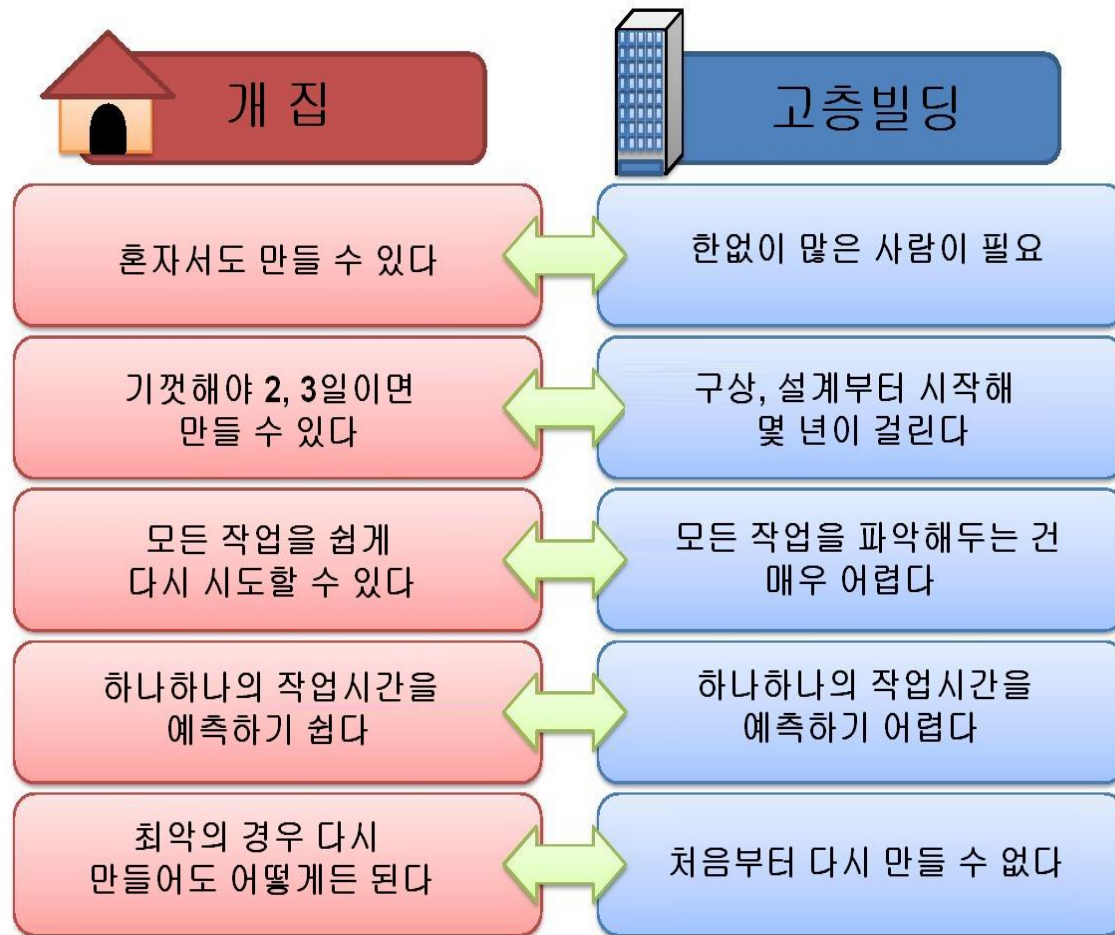
# 객체는 특정 클래스의 실체이다.

- ❖ 모든 객체는 특정한 하나의 클래스의 실체로서 정의된다.
- ❖ 클래스로부터 객체를 생성하는 행위를 실체화(instantiation)라고 부른다.

| 교통수단    |
|---------|
| -최대탑승인원 |
| -이동속도   |
| +출발0    |
| +정지0    |



# 분석과 설계의 필요성



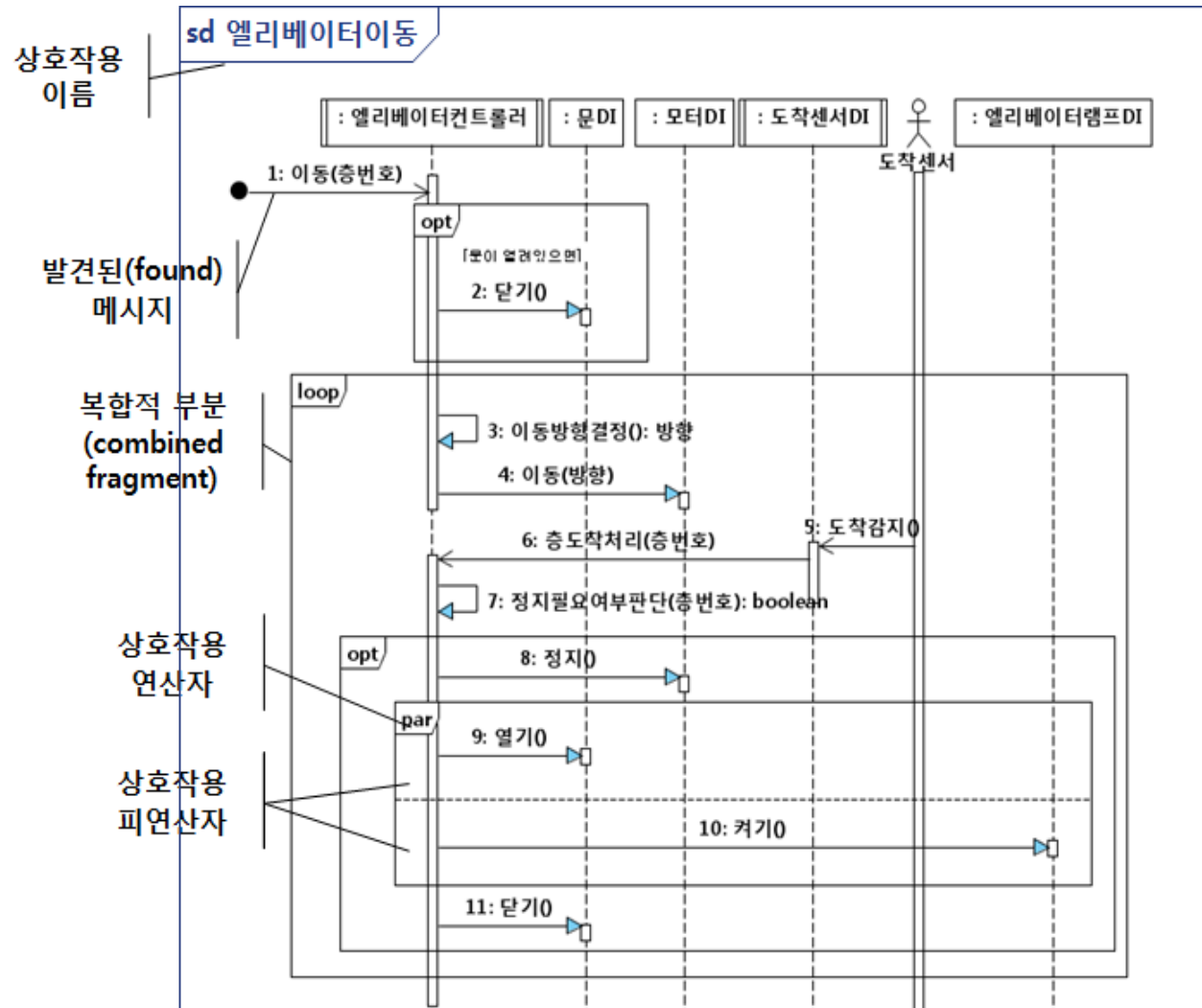


# 클래스 추출 과정 (예: pp. 146-147)

---

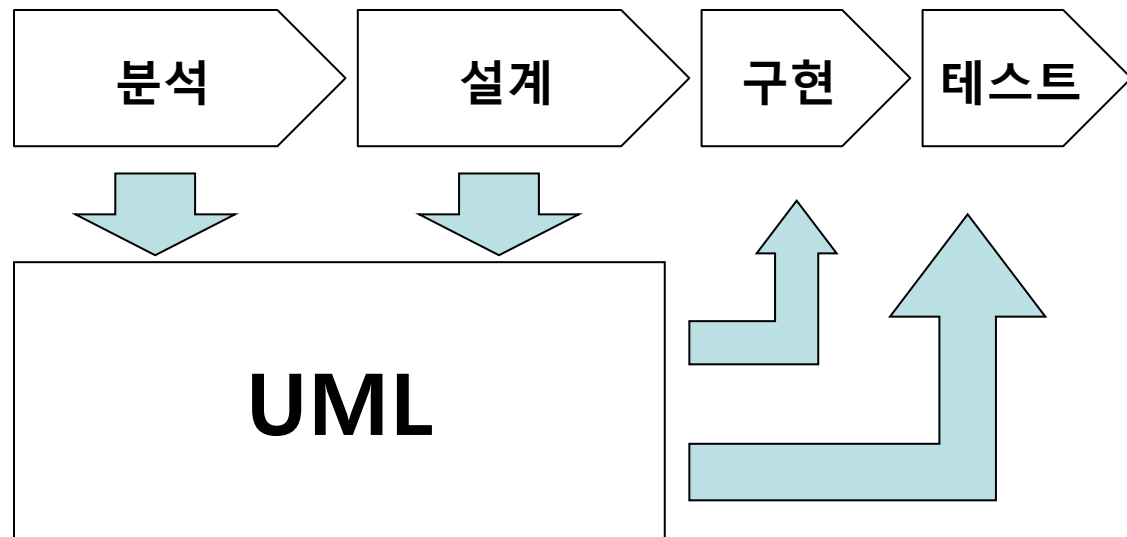
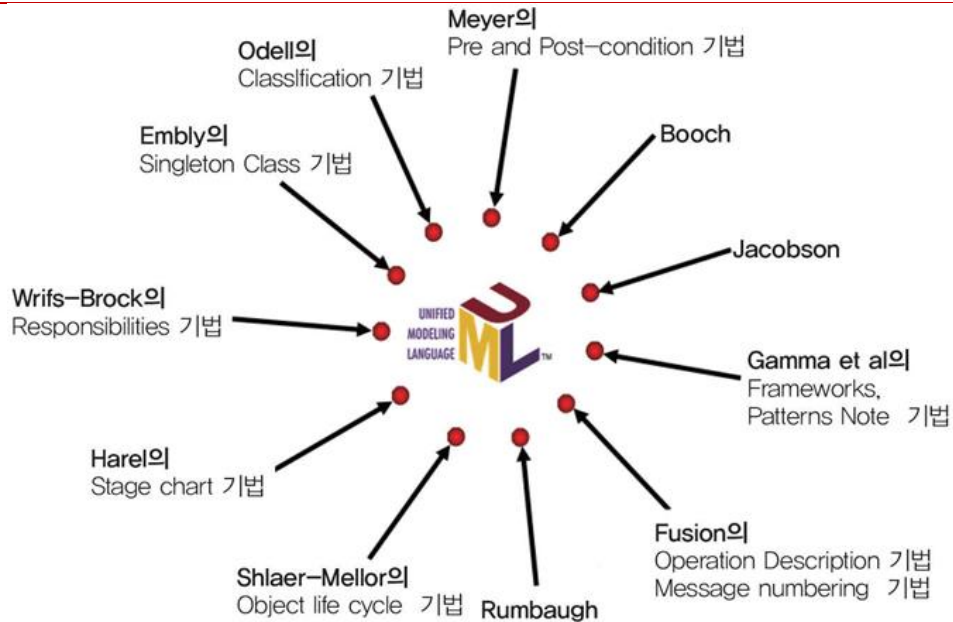
- ① 요구사항으로부터 명사나 명사구 추출
- ② 클래스가 되기에는 부적절한 명사 제거
- ③ 클래스의 속성이나 메소드 제거
- ④ 요구사항에 정의되어 있지 않지만, 필요한 클래스 추가
- ⑤ 최종적으로 추출된 클래스

# 순차 다이어그램



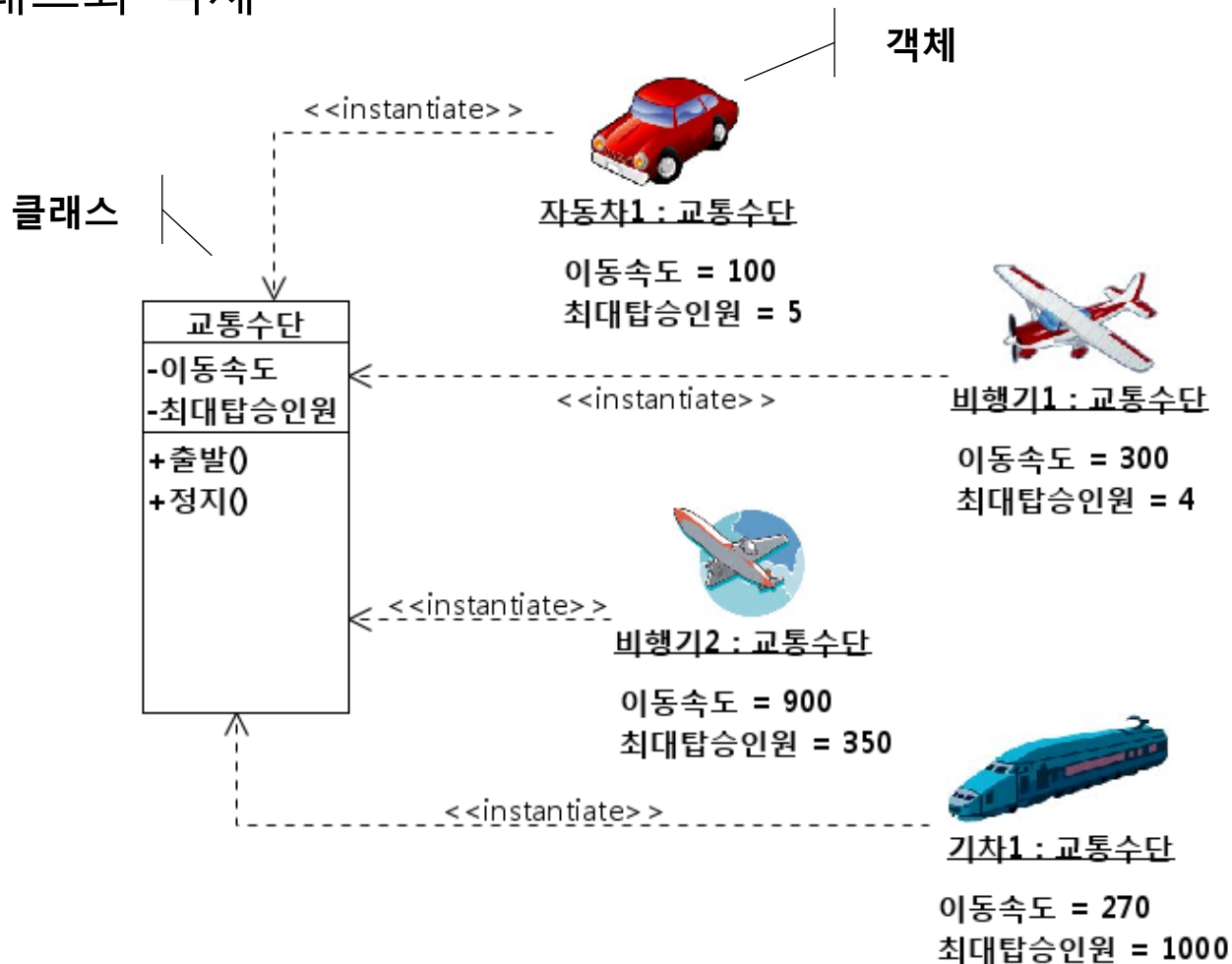
# 객체지향 분석 개요

## ❖ UML



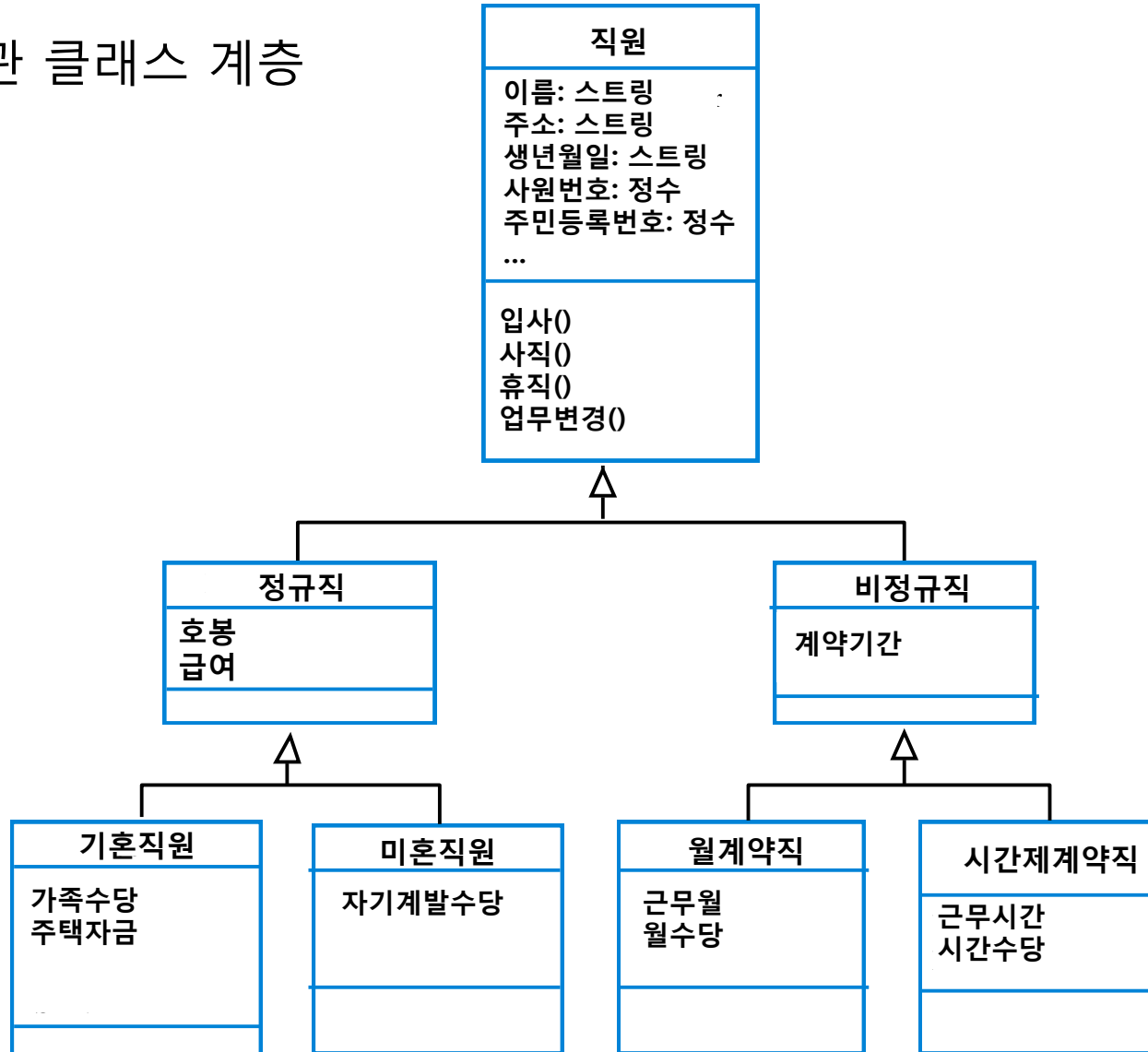
# 분석 모델

## ❖ 클래스와 객체



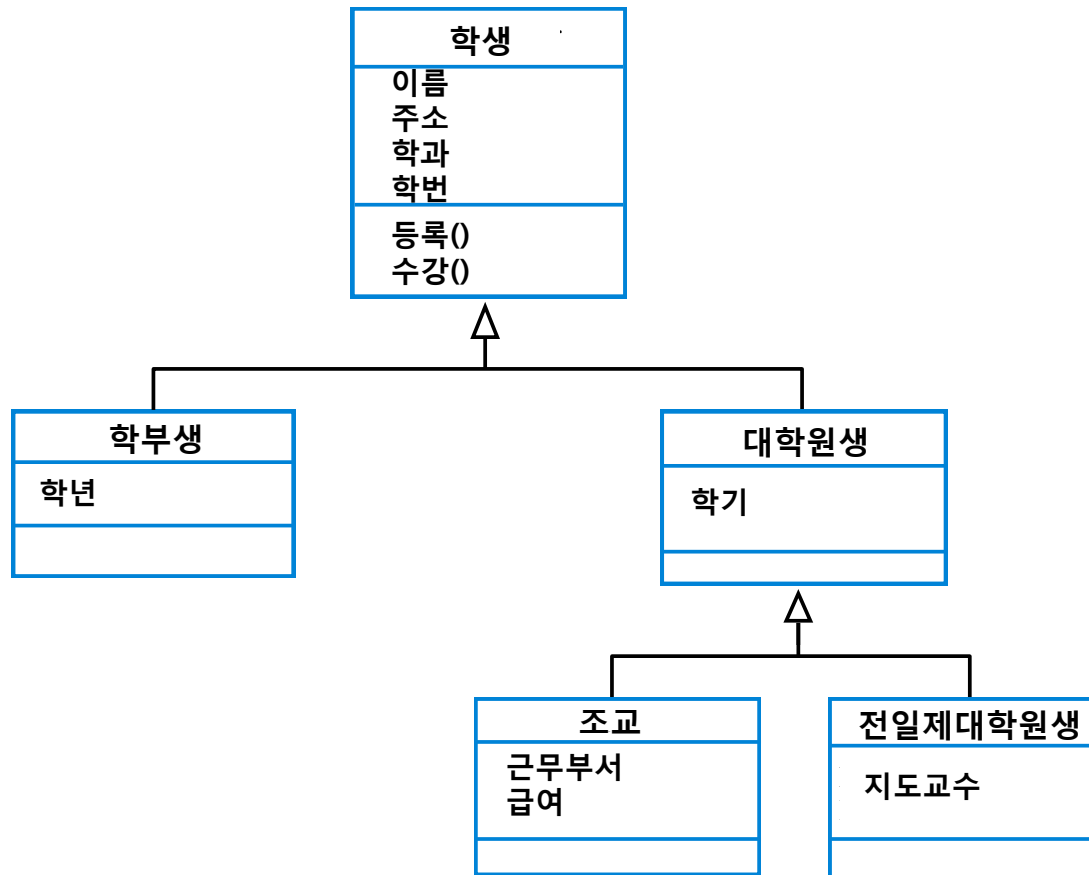
# 상속 모델 (1/2)

## ❖ 도서관 클래스 계층



## 상속 모델 (2/2)

### ❖ 도서관 클래스 사용자 계층



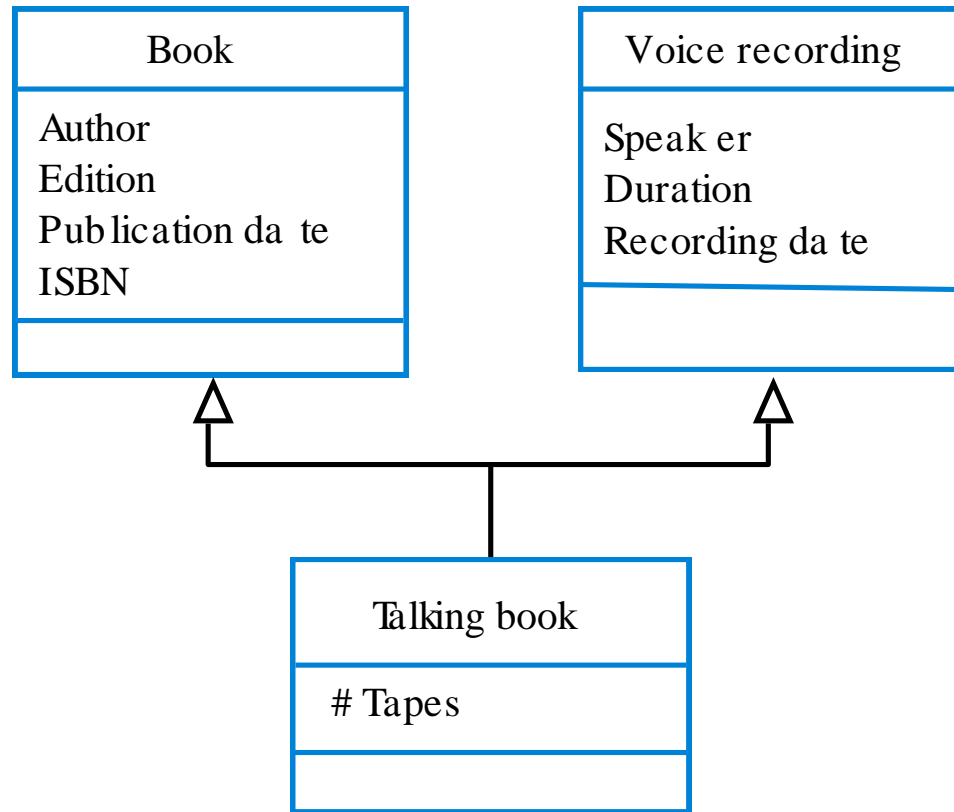
## 다중 상속 (1/2)

---

- ❖ 시스템은 하나의 상위 클래스로부터 속성과 연산을 상속 받는 것이 아니라, 여러 상위 클래스로부터의 상속을 가능하게 하는 다중 상속을 지원
- ❖ 불필요한 속성까지 상속 가능
- ❖ 상이한 의미를 가지는 상이한 수퍼 클래스에서 동일한 이름을 가지는 속성과 연산에서 발생하는 이름 충돌을 유발할 수 있음
- ❖ 다중 상속은 클래스 계층의 재구성을 더욱 복잡하게 만듦

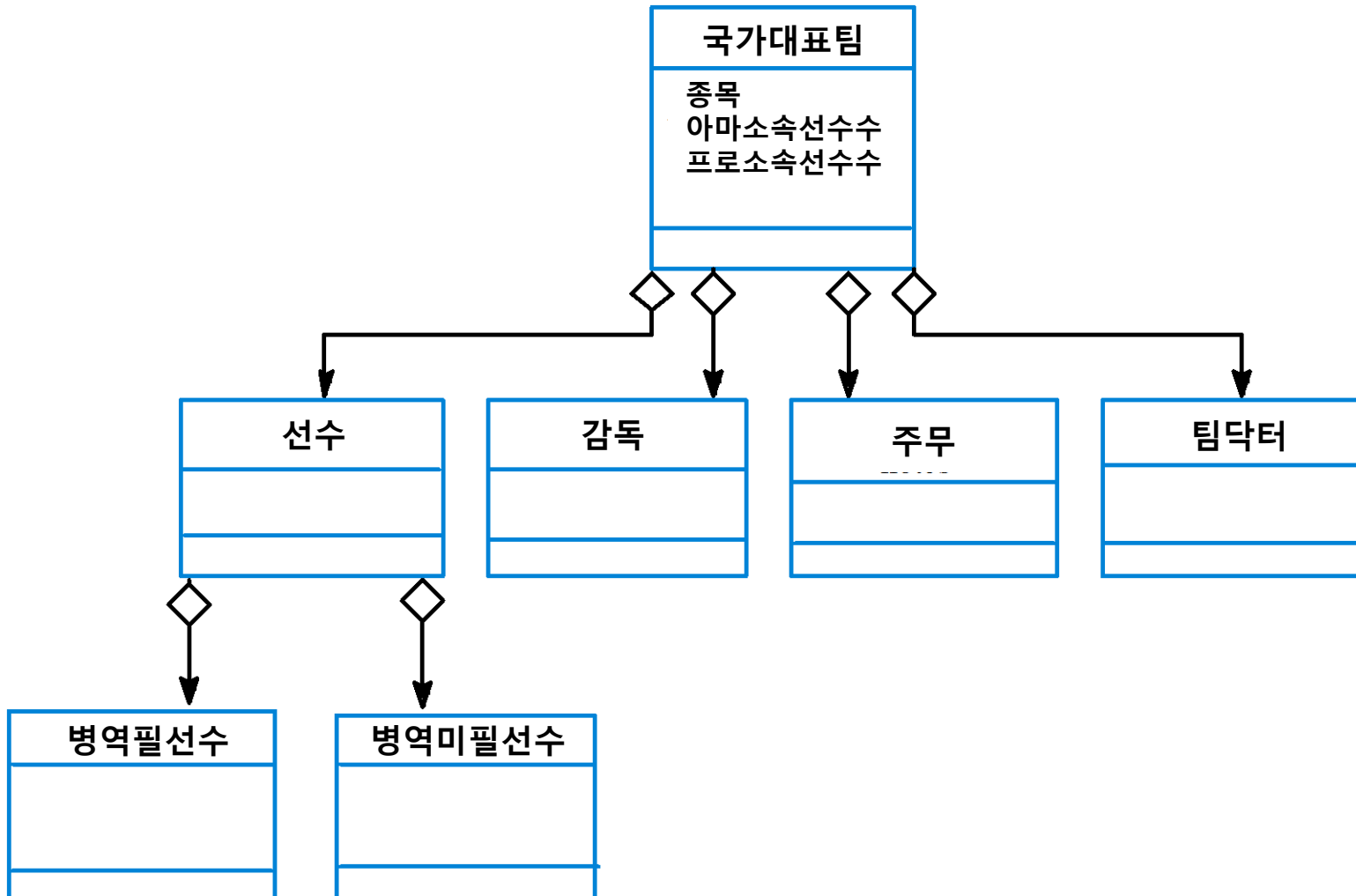
## 다중 상속 (2/2)

---

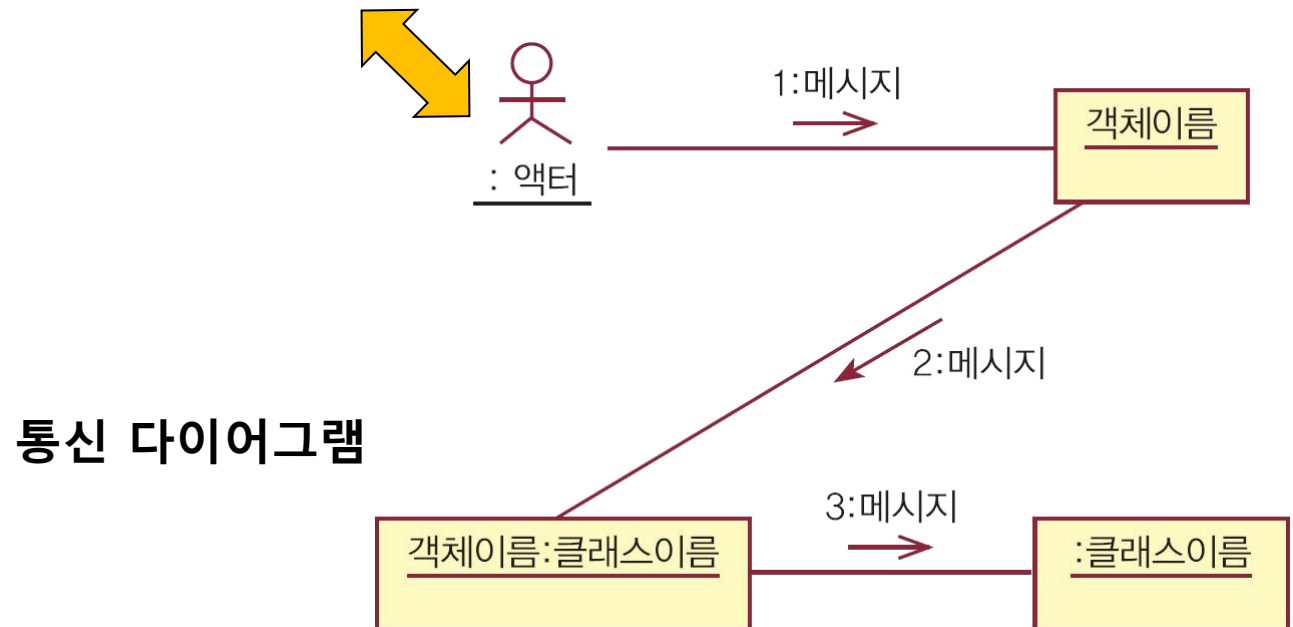
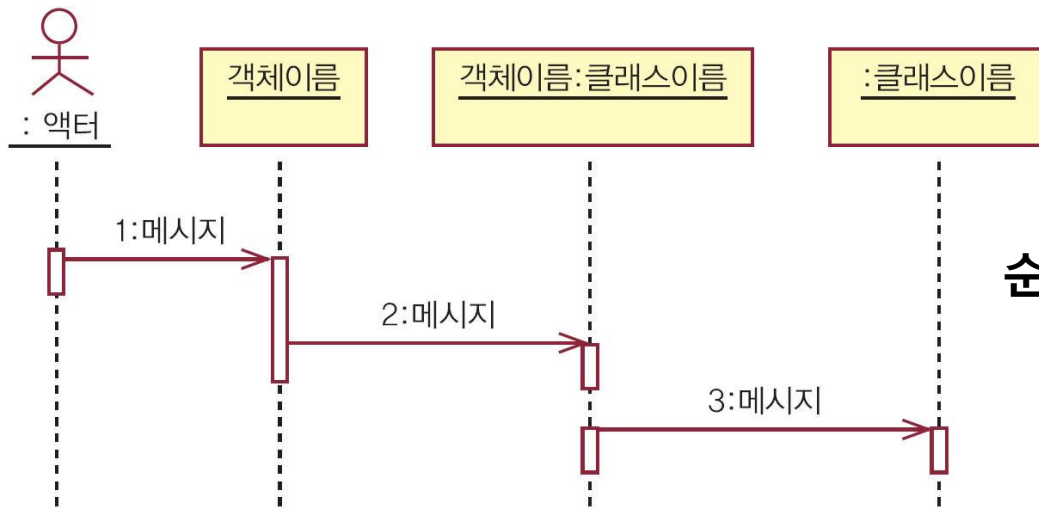




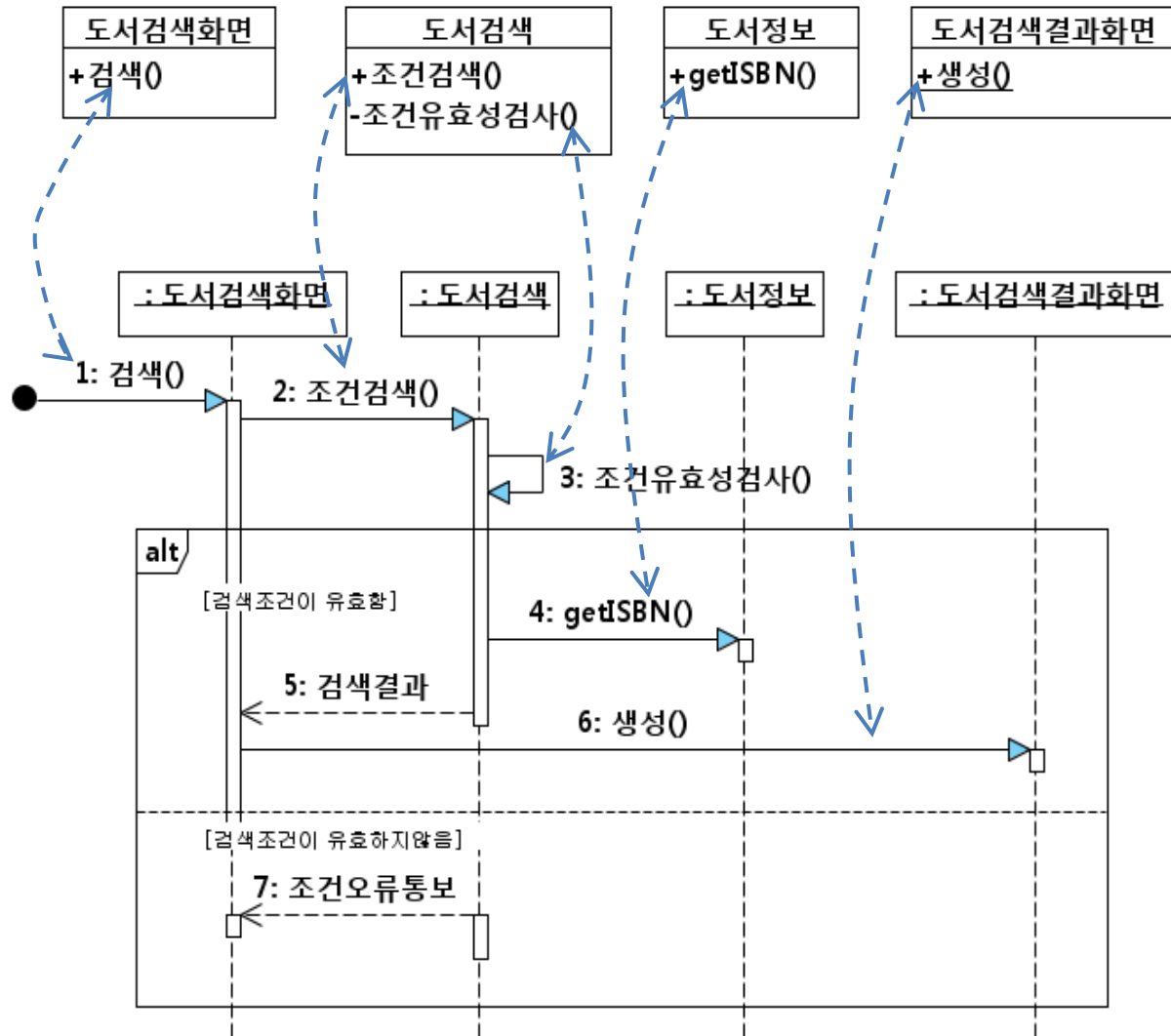
# 집합 관계



# 동적 모델링



# 순차 다이어그램과 클래스의 연산

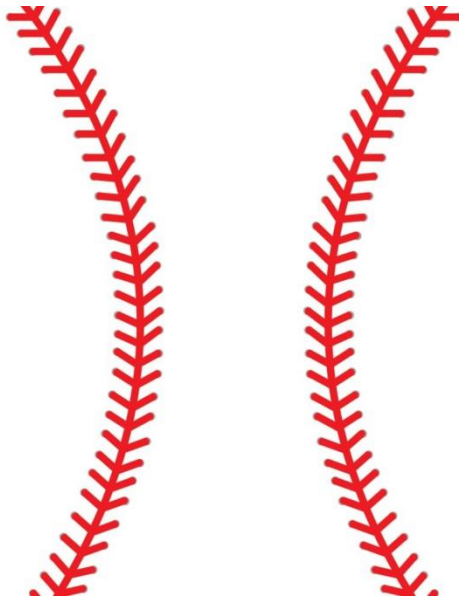


# 객체지향 분석 프로세스

---

❖ 객체지향 개발은 **seam**less transition

❖ Seam



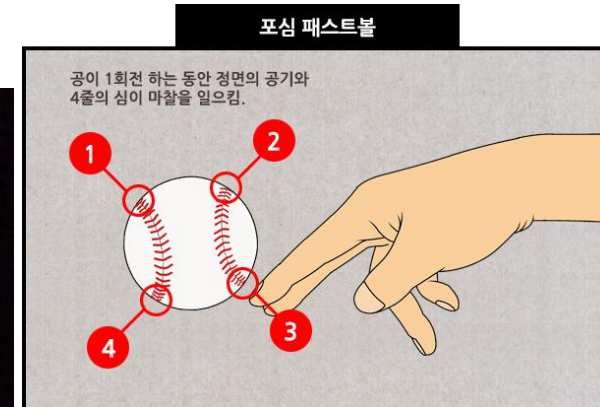
<http://blog.naver.com/com2001v>

# 투심과 포심 패스트 볼

## ❖ 투심 그립



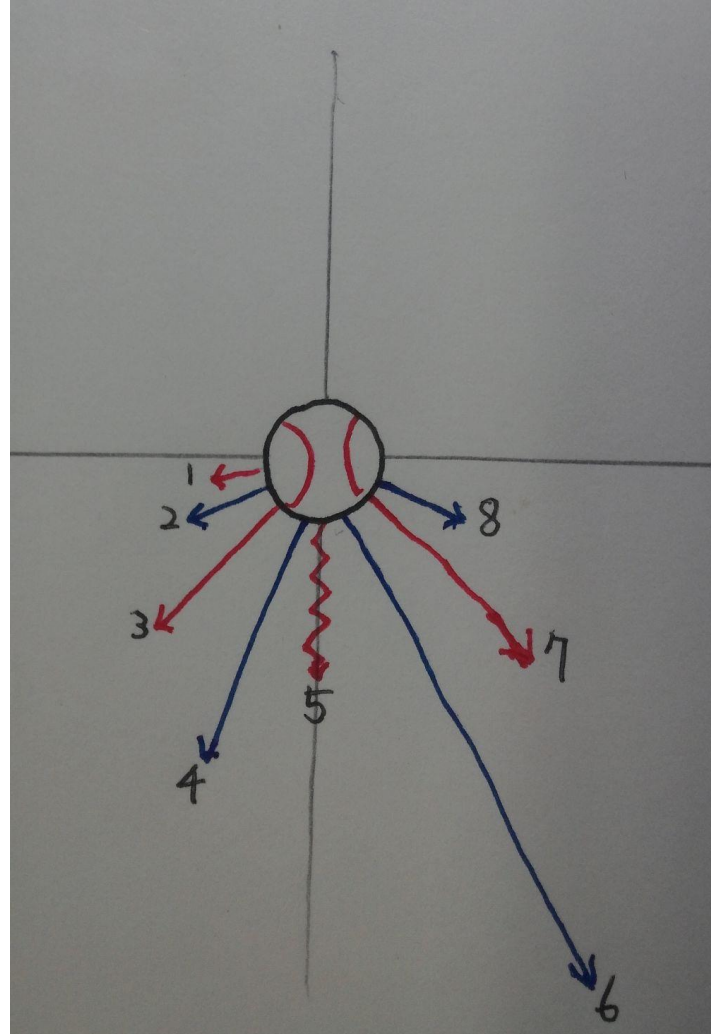
## ❖ 포심 그립



# 투수의 궤적

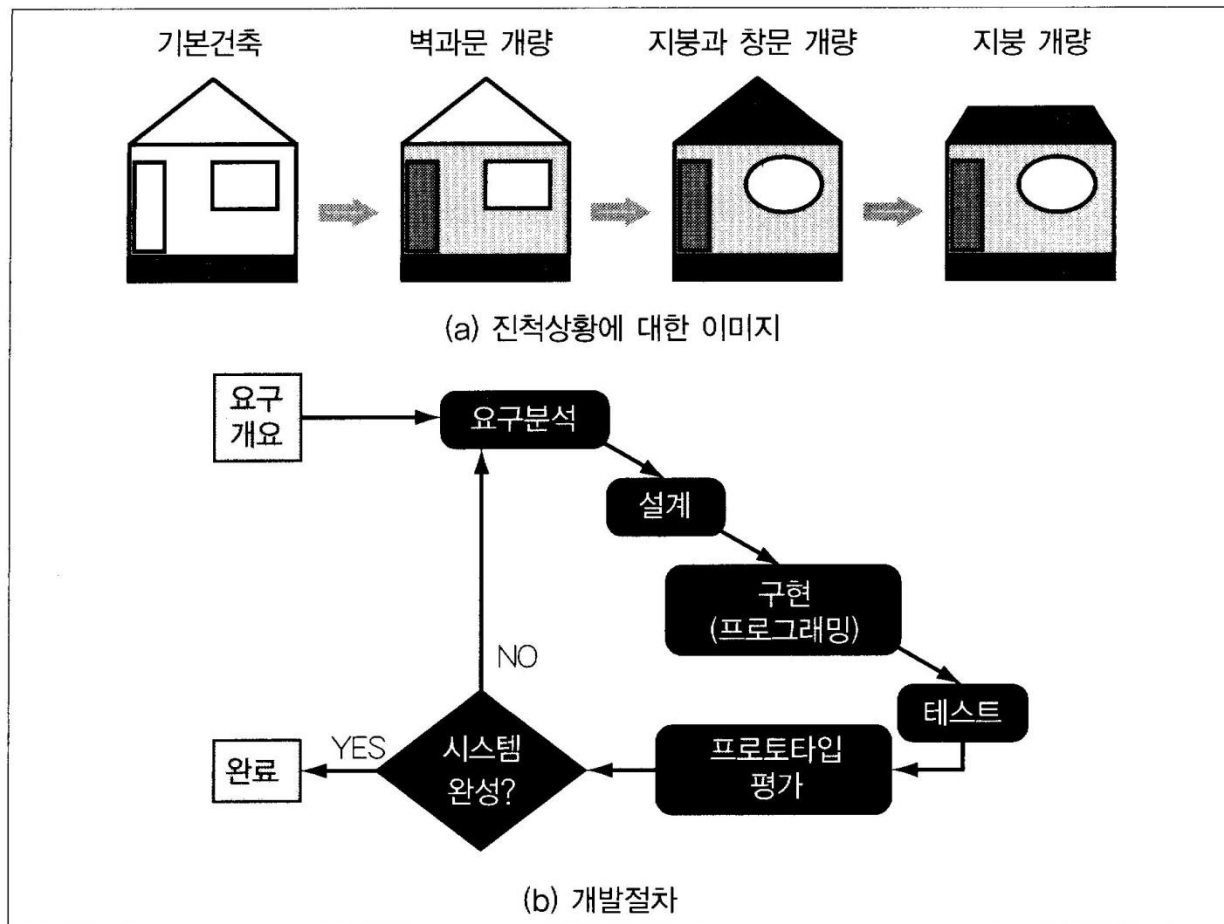
## ❖ <좌완투수의 궤적>

- (1) 포심패스트볼
- (2) 투심패스트볼
- (3) 체인지업
- (4) 스플리터
- (5) 너클볼
- (6) 커브
- (7) 슬라이더
- (8) 커터



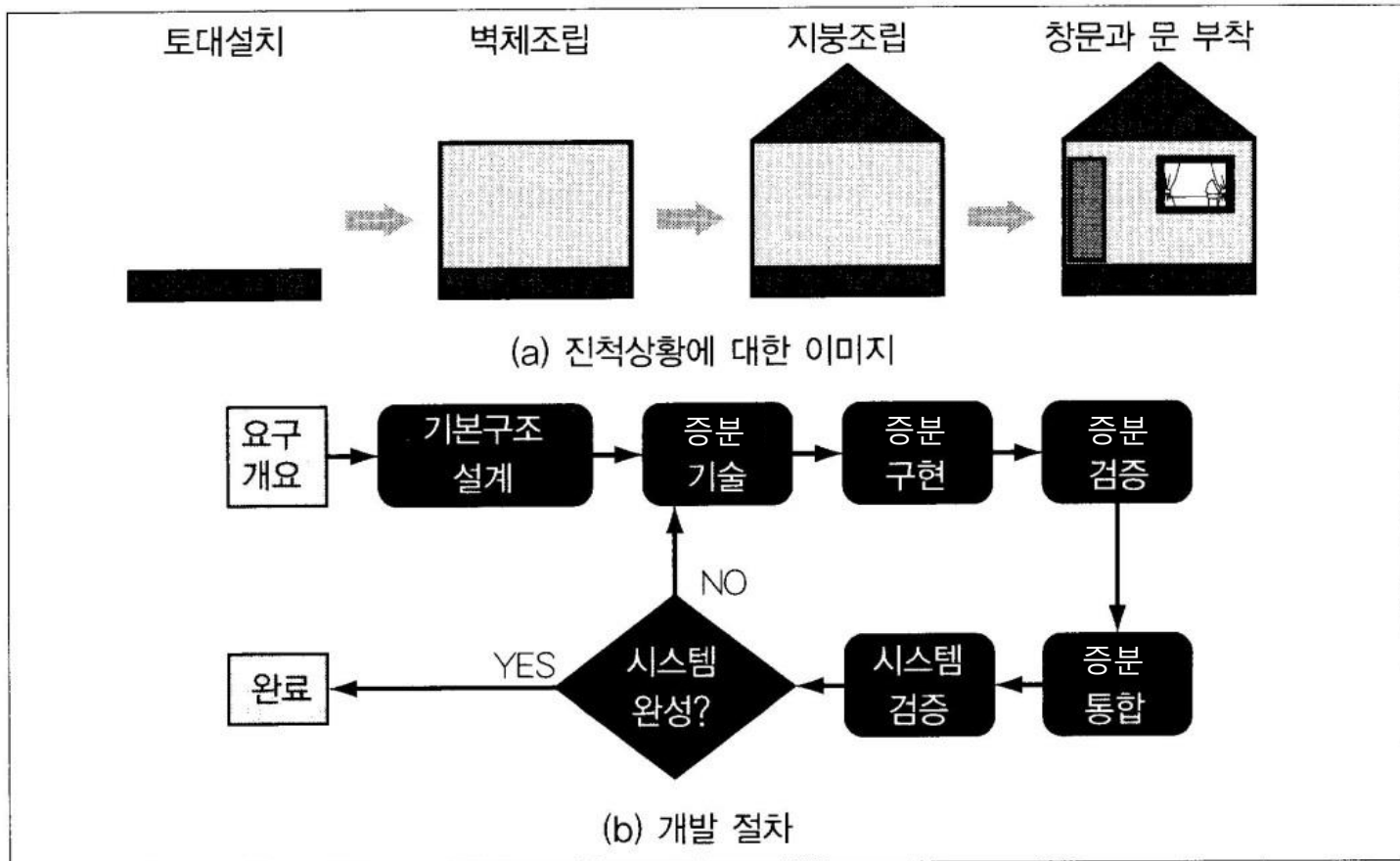
# 반복적 소프트웨어 개발

- ❖ 분석, 설계, 구현, 테스트의 각 단계를 연속적으로 반복하여 수행함으로써 개발을 진행하는 기법



# 점증적 소프트웨어 개발

- ❖ 소프트웨어를 여러 개의 증분으로 나누어서 조금씩 단계적으로 개발해가는 방법

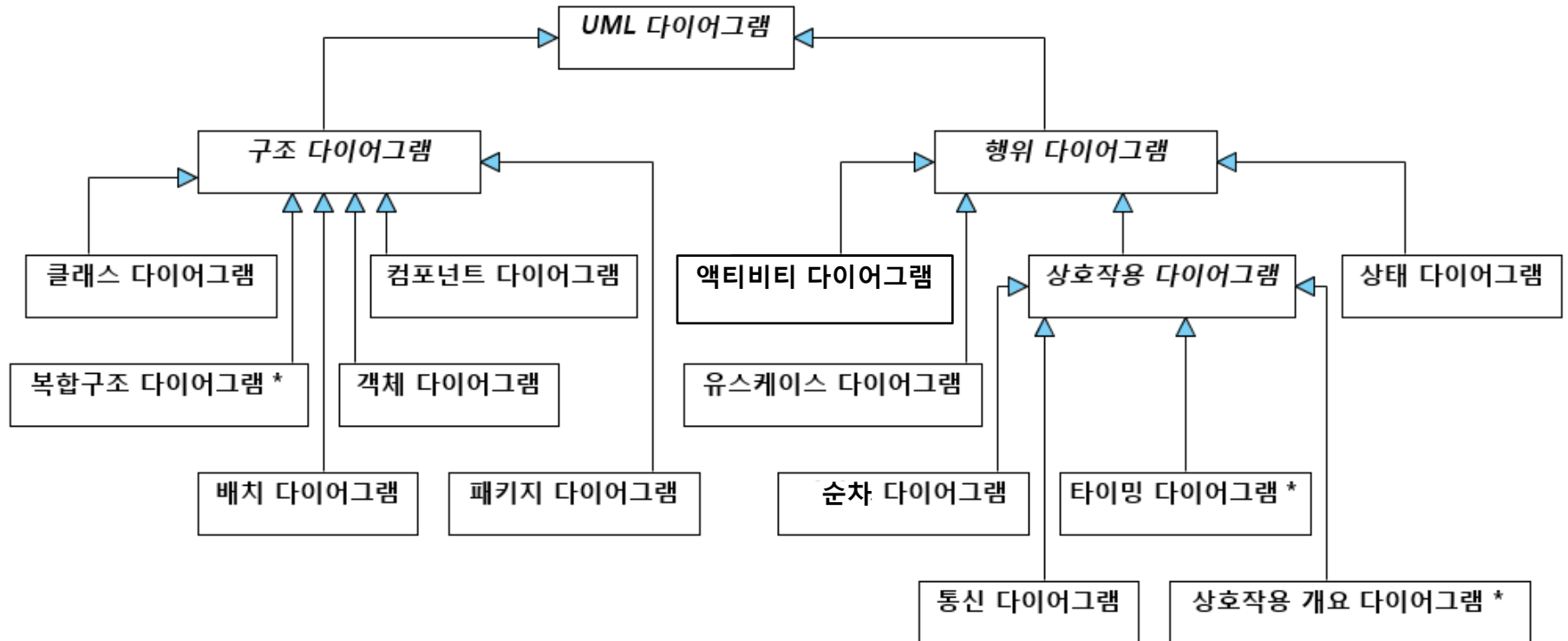




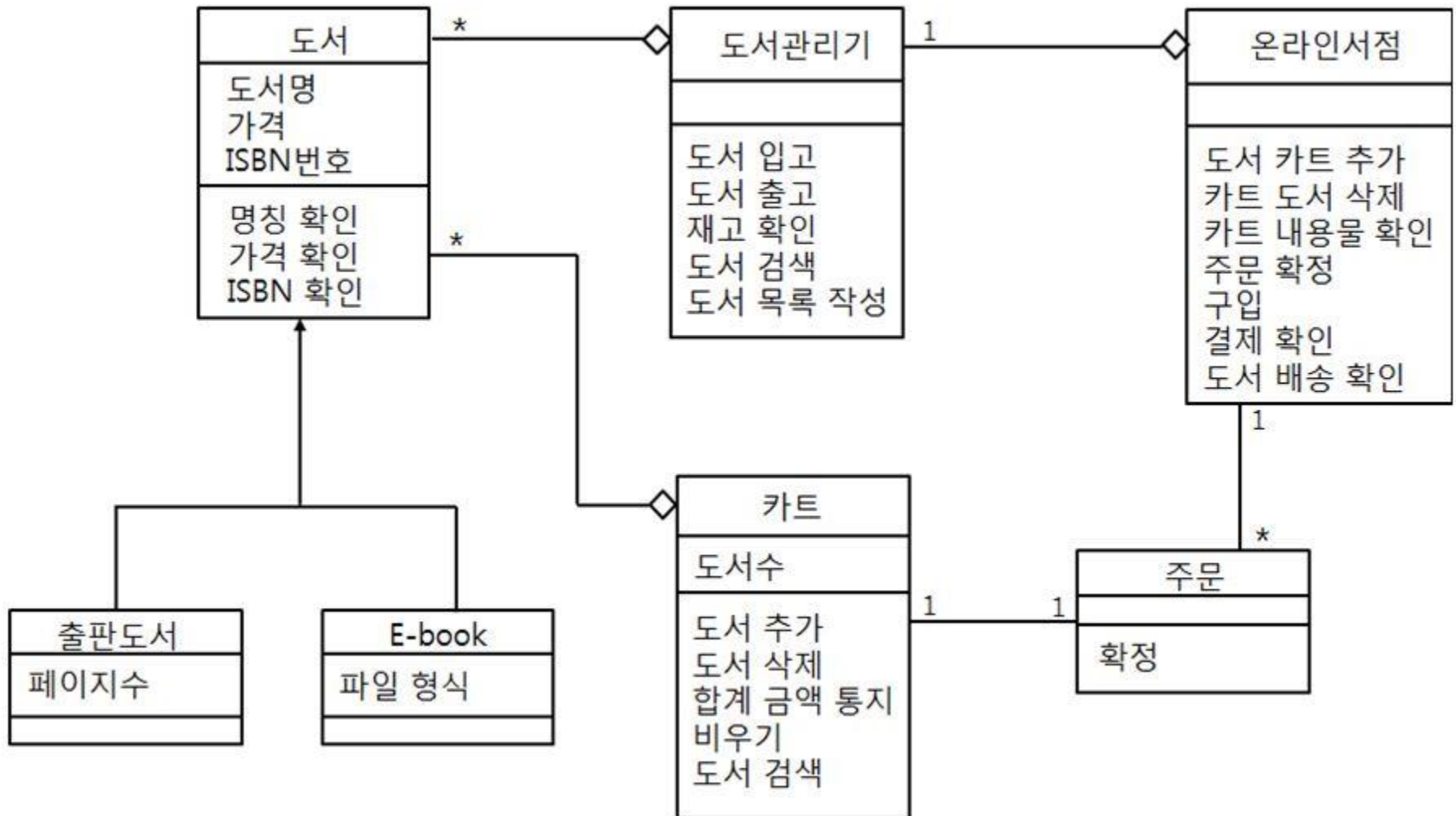
# UML 다이어그램 (1/2)

| 관점 | 다이어그램         | 설명                             |
|----|---------------|--------------------------------|
| 구조 | 클래스 다이어그램     | 클래스의 구조(속성과 연산)와 클래스 사이의 정적 관계 |
|    | 객체 다이어그램      | 특정 시점에서 객체의 상태와 객체 사이의 관계      |
|    | 패키지 다이어그램     | 패키지의 구성과 패키지 사이의 의존 관계         |
|    | 복합 구조 다이어그램   | 실행 시의 클래스 내부 구조                |
|    | 컴포넌트 다이어그램    | 컴포넌트의 구조와 의존 관계                |
|    | 배치 다이어그램      | 시스템에서의 물리적 배치                  |
| 행위 | 유스케이스 다이어그램   | 시스템에서 제공하는 기능과 사용자 사이의 관계      |
|    | 액티비티 다이어그램    | 작업의 순서와 병행성                    |
|    | 상태 다이어그램      | 객체의 상태와 이벤트에 의한 상태 전이          |
|    | 순차 다이어그램      | 객체 사이의 상호작용(시간 흐름 중심)          |
|    | 통신 다이어그램      | 객체 사이의 상호작용(객체 사이의 관계 중심)      |
|    | 타이밍 다이어그램     | 객체의 상호작용 타이밍                   |
|    | 상호작용 개요 다이어그램 | 순차 다이어그램과 액티비티 다이어그램의 개요       |

# UML 다이어그램 (2/2)

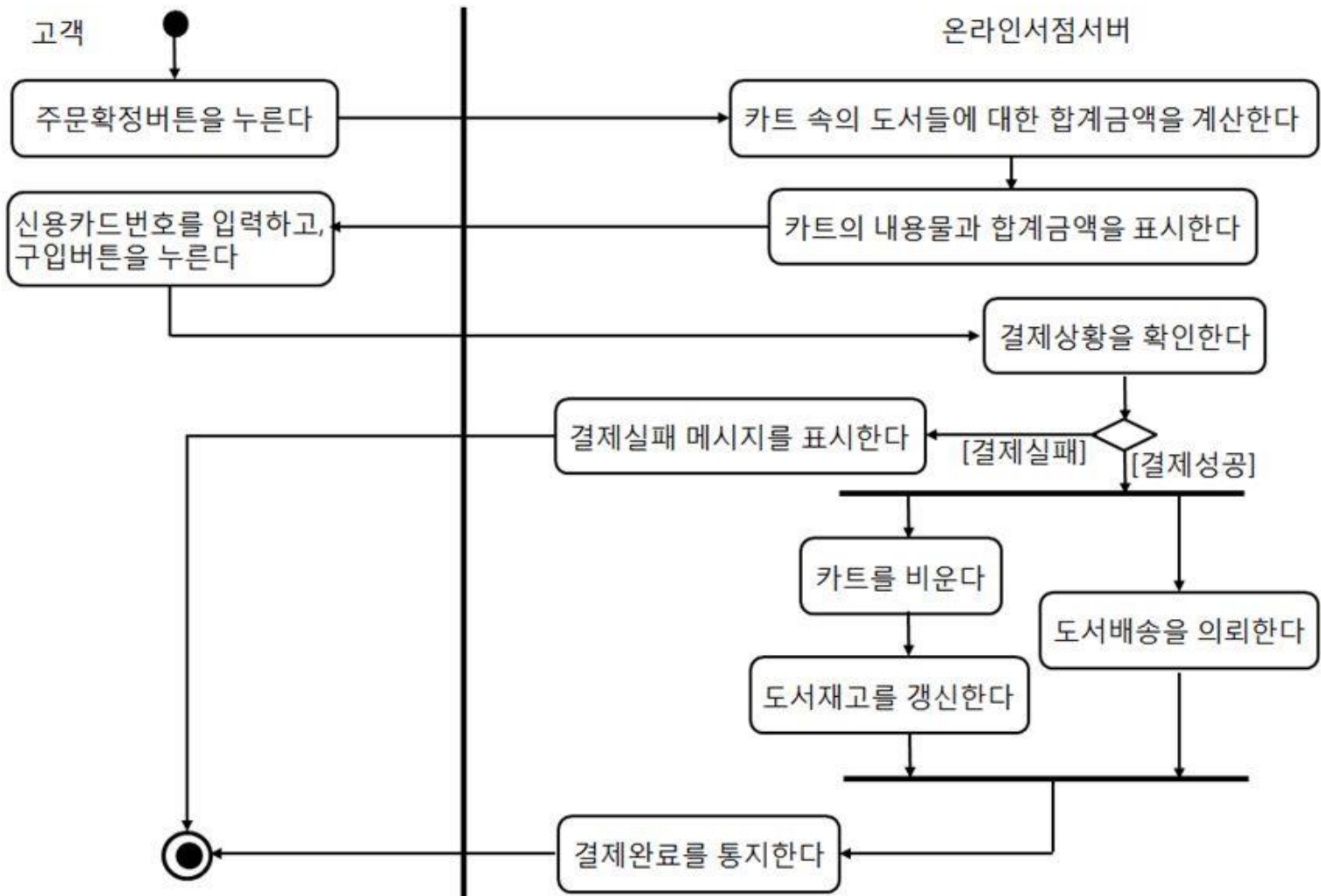


# 클래스 다이어그램



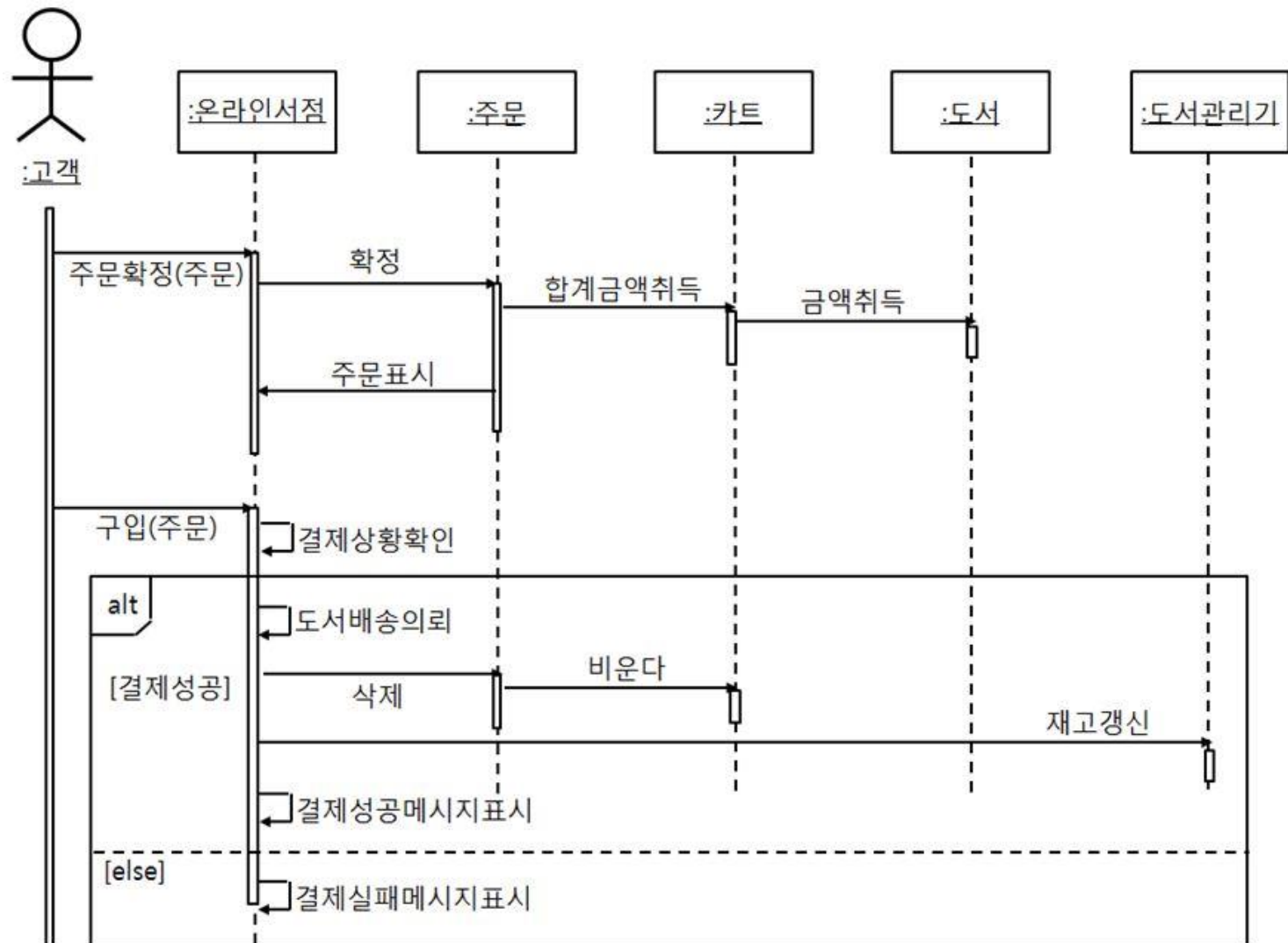
# 액티비티 다이어그램

❖ 시스템 전체의 작업 흐름을 표현하는 동적 모델



# 순차 다이어그램

- ❖ 각 객체들의 메시지 송수신을 시간 흐름에 따라 표현하는 동적 모델



# CRC 카드

| 클래스 이름<br>(Class)          |                         |
|----------------------------|-------------------------|
| 수행할 작업<br>(Responsibility) | 협력관계<br>(Collaboration) |
|                            |                         |

Chapter 4 Quiz: CS 2450 V. 3c Systems Analysis and Design

proquest.safaribooksonline.com.hal.weber.edu.2200/book/software-engineering-and-development/uml/9781118037423/firstchapter#X2ludGvybmFsX0h0bWxWaWV3

Apps Canvas Systems Analysis an... The Eclipse Guided ... The Eclipse Guided ... UIS email UIS BB

Front:

|  |       |                         |
|--|-------|-------------------------|
| Class Name: Old Patient  | ID: 3 | Type: Concrete, Domain  |
| Description: An individual that needs to receive or has received medical attention |       | Associated Use Cases: 2 |

| Responsibilities        | Collaborators   |
|-------------------------|-----------------|
| Make appointment        | Appointment     |
| Calculate last visit    |                 |
| Change status           |                 |
| Provide medical history | Medical history |
|                         |                 |
|                         |                 |
|                         |                 |

Back:

Attributes:

|                          |  |
|--------------------------|--|
| Amount (double)          |  |
| Insurance carrier (text) |  |
|                          |  |
|                          |  |

Relationships:

Generalization (a-kind-of): Person

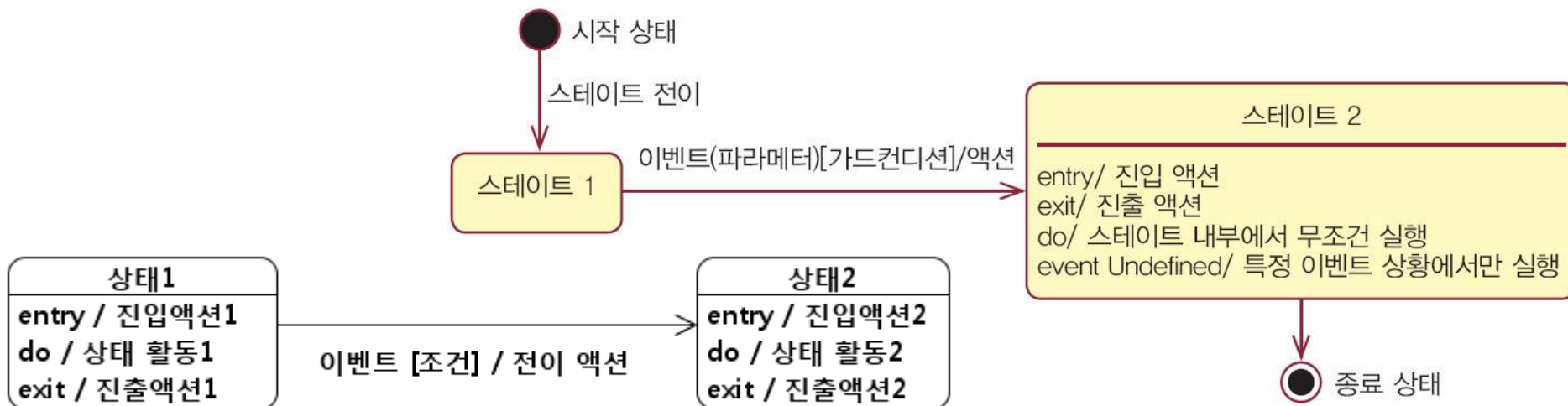
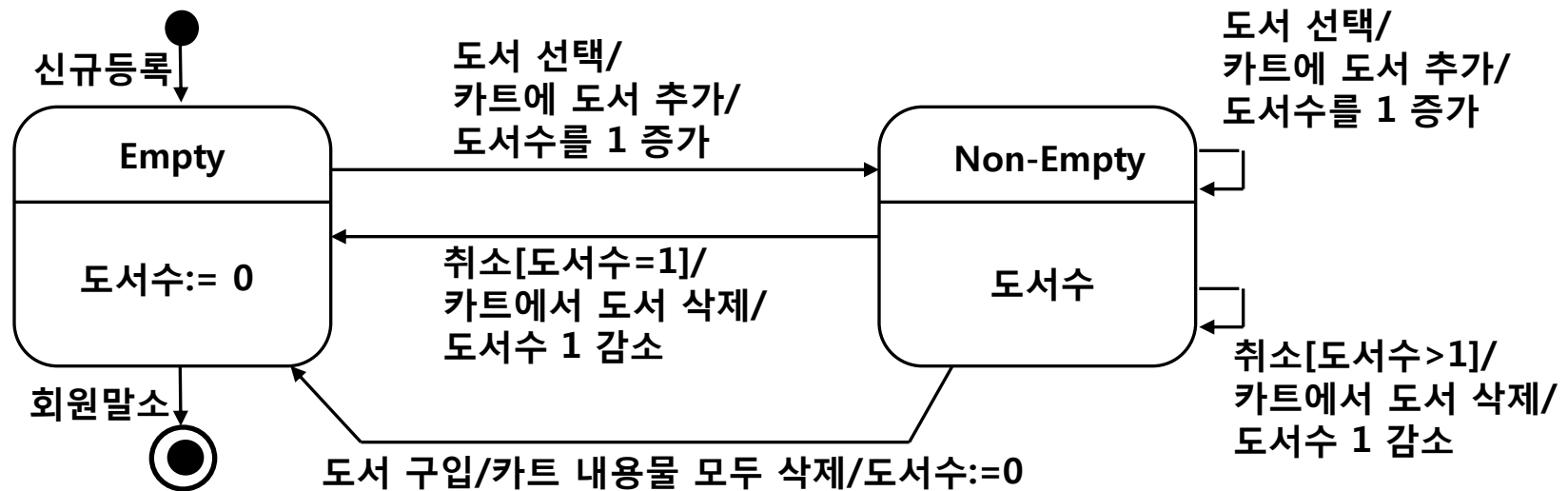
Aggregation (has-parts): Medical History

Other Associations: Appointment

00:07:25 10:40 AM 10/9/2014

# 상태 다이어그램

❖ 각 객체들의 동적인 움직임을 나타내는 동적 모델



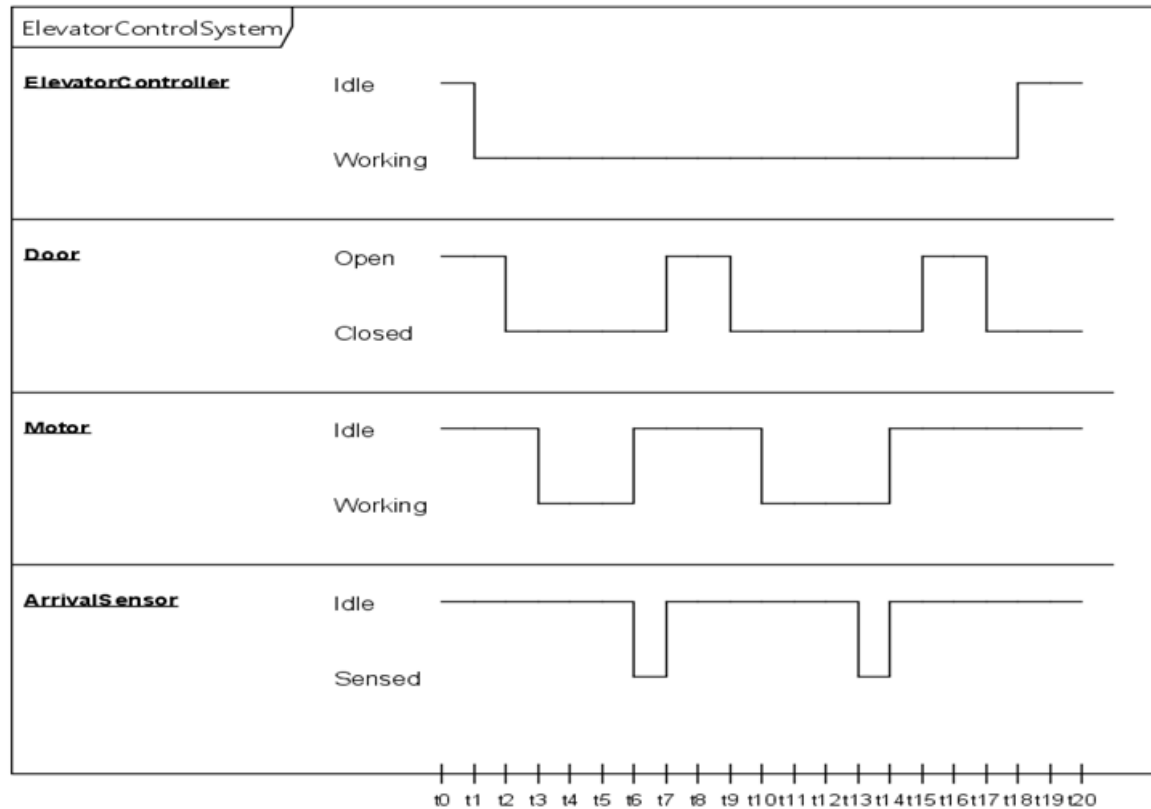
# 모델의 개선

---

- ❖ 타이밍 다이어그램
- ❖ 패키지 다이어그램
- ❖ 컴포넌트 다이어그램
- ❖ 배치 다이어그램
- ❖ 그 밖의 다이어그램

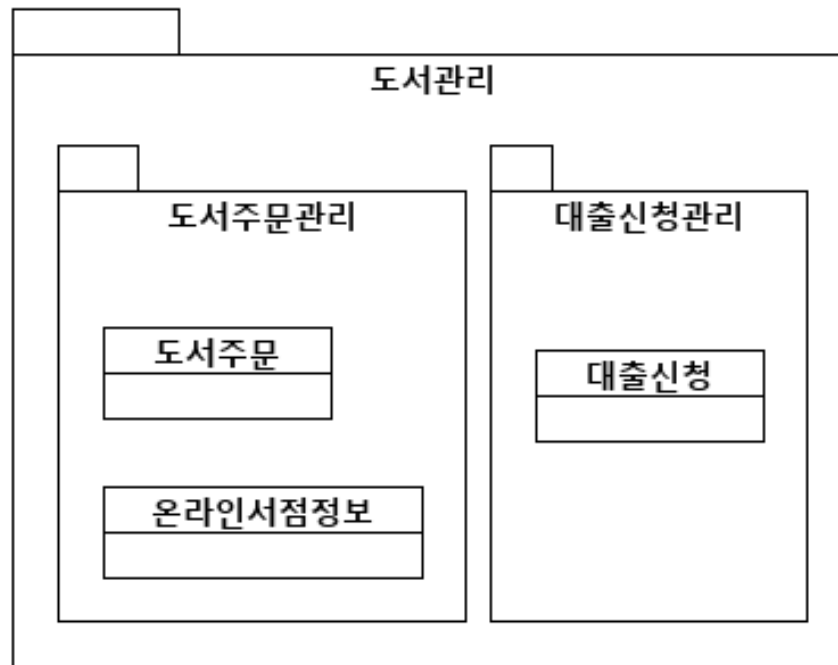


# 타이밍 다이어그램

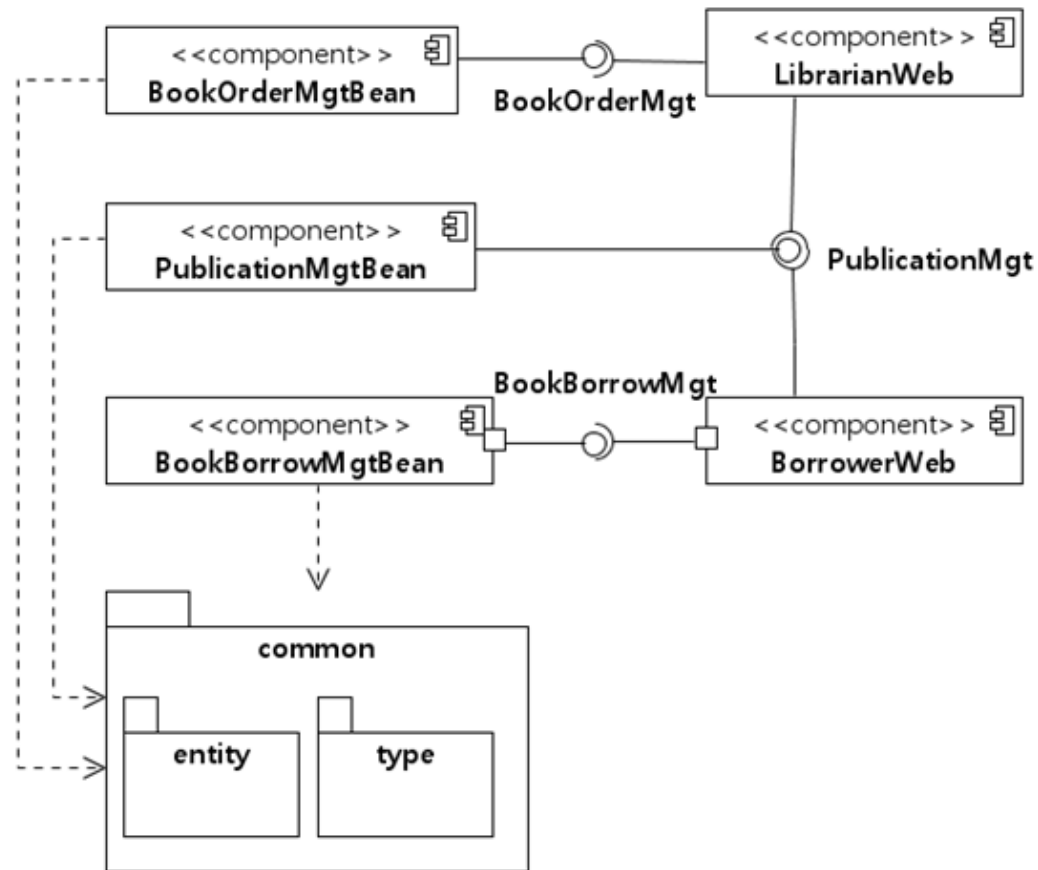


# 패키지 다이어그램

---



# 컴포넌트 다이어그램



# 배치 다이어그램

