# CSE053 Software Engineering
# Lecture: Use case diagram (I)

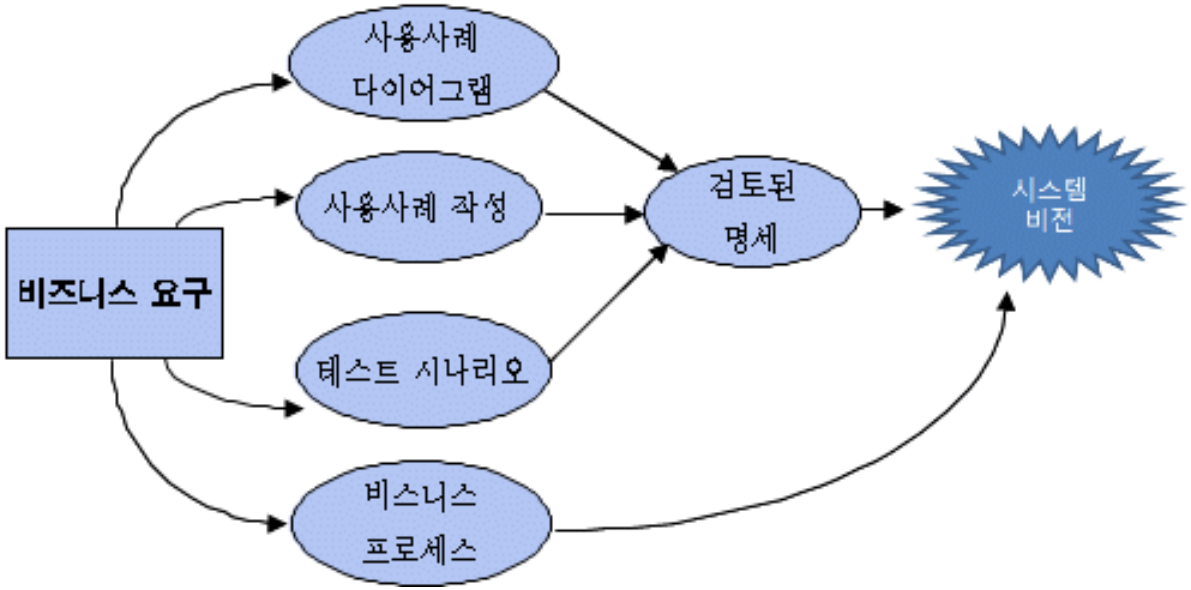

**Software engineering laboratory**
**Yeungnam university**


영남대학교 Yeungnam University

YEUNGNAM UNIVERSITY 1947 영남대학교

# SW Project 착수 이후 개념화 작업

- 기능을 잘 정리하고 표현하는 방법으로 기술
  - Use case
  - Prototype
  - Natural language
- Use case 기반의 분석 작업 과정



# Use case

- **Use case**
  - **외부에서 본 시스템의 view, behavior**
  - **시스템에 무슨 서비스가 있는지 사용자의 관점으로 본 것 (기능 중심)**
  - **개발 대상이 되는 시스템이 제공하는 개별적인 기능**
  - **시스템의 범위에 해당되어 개발될 시스템의 단위기능**
  - **시스템(system)과 외부 액터(actor) 사이의 관계**

  - Example) 쇼핑몰

  → 고객이 책 번호와 수량을 선택하여 주문한다.

  → 고객이 신용카드 번호와 배달 정보를 입력하여 주문을 결재한다.

  → 신용카드 회사에서 결재를 승인하거나 거부한다.

  → 고객의 신용카드 사용이 승인되었다면 요청한 상품을 포장하여 배송한다.

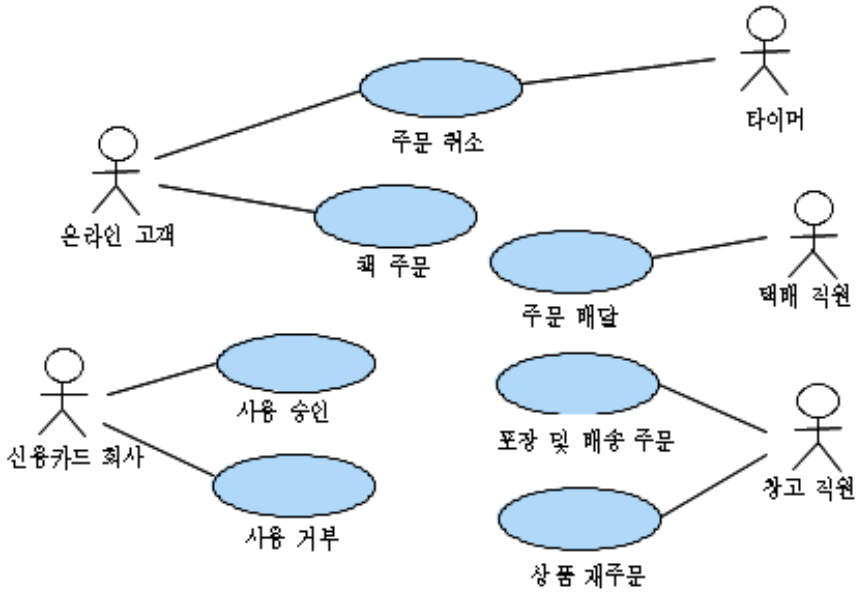
# Use case

- **Use case 작성 목적**
  - 시스템의 범위를 정하는데 도움이 됨
  - 개발 과정을 계획하는데도 사용됨
  - 요구를 개발하고 검증하는데 사용 됨
  - 테스트케이스를 정의하는데 기초가 됨
  - 사용자 매뉴얼 구성하는데 사용될 수 있음

# Use case diagram

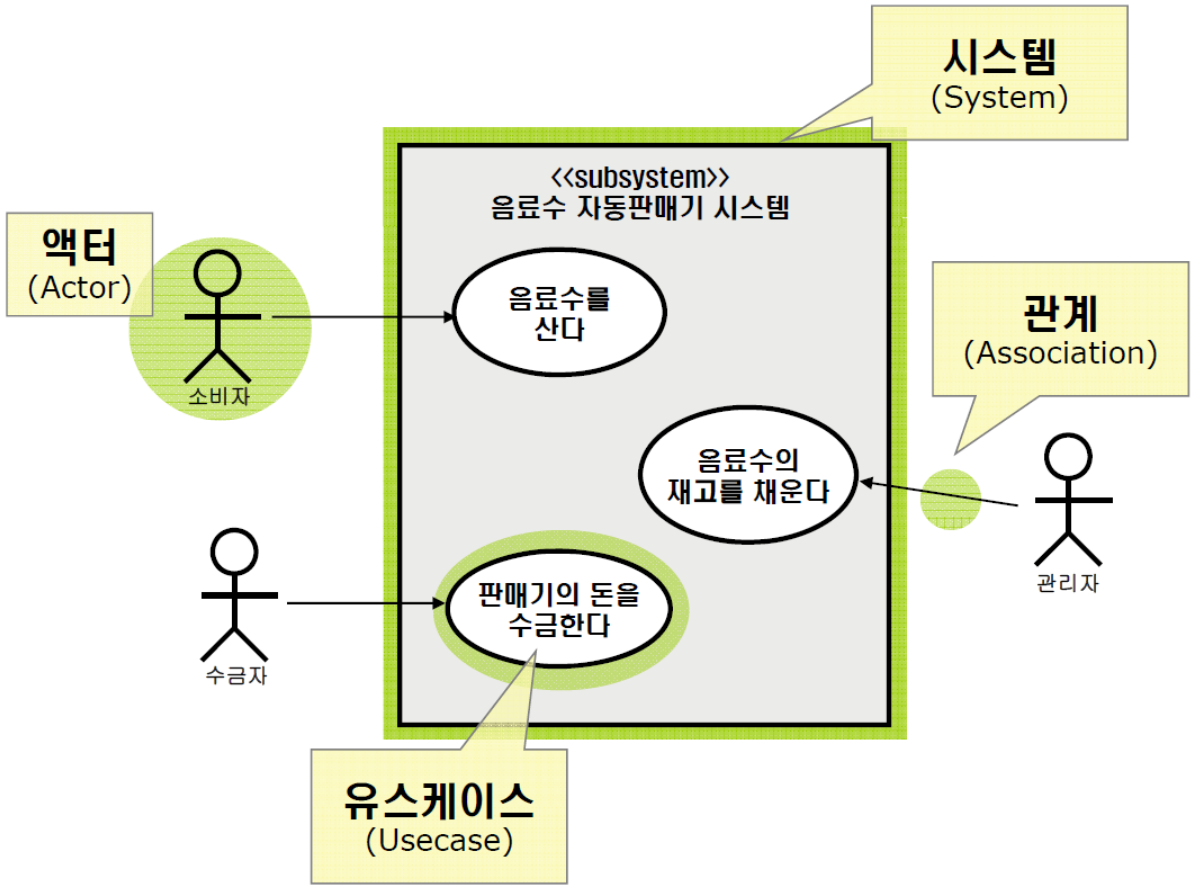
## ▪ 개요

- 사용자의 관점에서 시스템의 서비스 혹은 기능 및 그와 관련한 외부 요소를 보여주는 다이어그램
- Use case에 대한 목차 성격의 그림
- 고객과 PM이 함께 보며 요구사항에 대한 의견을 조율할 수 있음



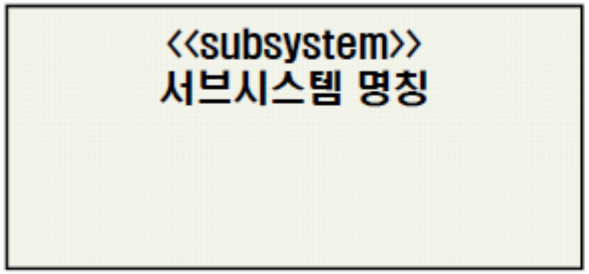
# Use case diagram

- **구성요소**
  - (1) System, (2) Actor, (3) Use case, (4) Association

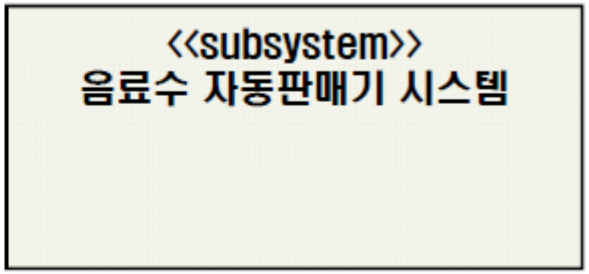


# System

- 의미
  - 만들고자 하는 시스템의 범위
- 표기법(Notation)
  - Use case를 포함한 사각형틀로 표현
  - 시스템이나 모델의 명칭을 사각형 안쪽 상단에 기술
  - 서브시스템일 경우<<subsystem>>이라 기술



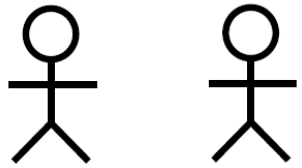

<시스템의 표현 방법>



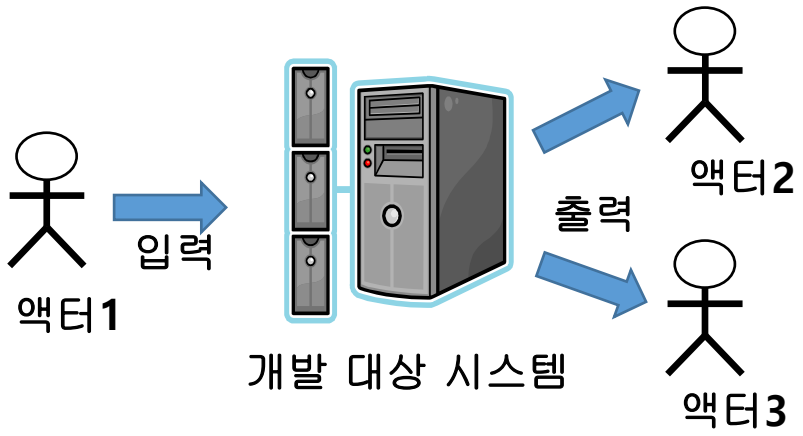

<예시>

# Actor (1/2)

- **의미**
  - 시스템과 상호작용하는 시스템 외부의 존재
  - 시스템으로부터 서비스를 받을 필요가 있는 외부 요소
  - 시스템의 외부에 있으면서 시스템과 상호 작용을 하는 “사람” 또는 “다른 시스템”
- **표기법(Notation)**
  - 원과 선을 조합하여 사람 모양으로 표현
  - 그 아래에 Actor명 표시
  - Actor명은 Actor의 역할로 정함

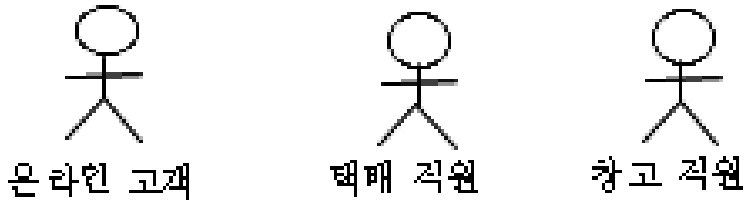



<액터의 표현 방법> <예시>

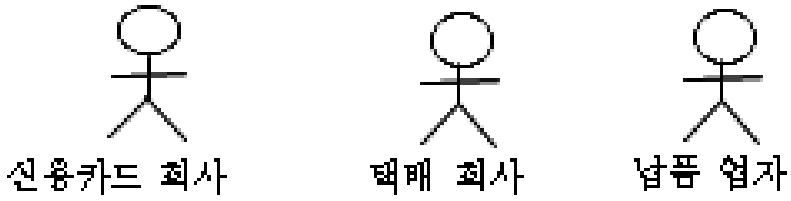


# Actor (2/2)

- **Actor를 찾는 요령**
  - 시스템의 주된 사용자는 누구인가?
  - 시스템과 상호작용하는 다른 시스템이 있는가?
  - 시스템의 출력에 관심이 있는 사람이나 다른 시스템은 무엇인가?
- **Example**
  - 사람, 역할

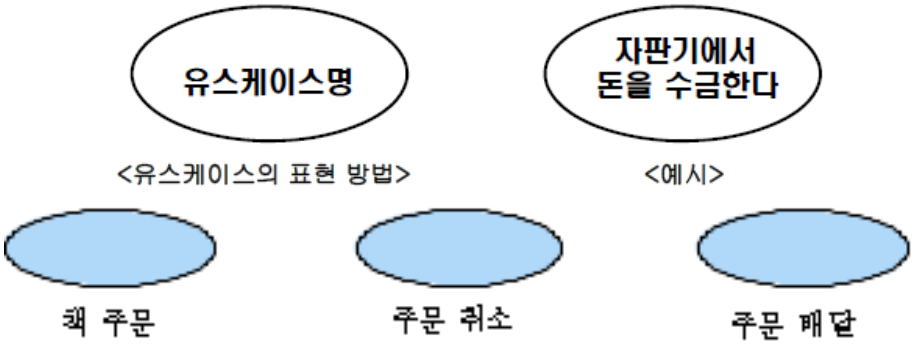



  - 시스템, 조직



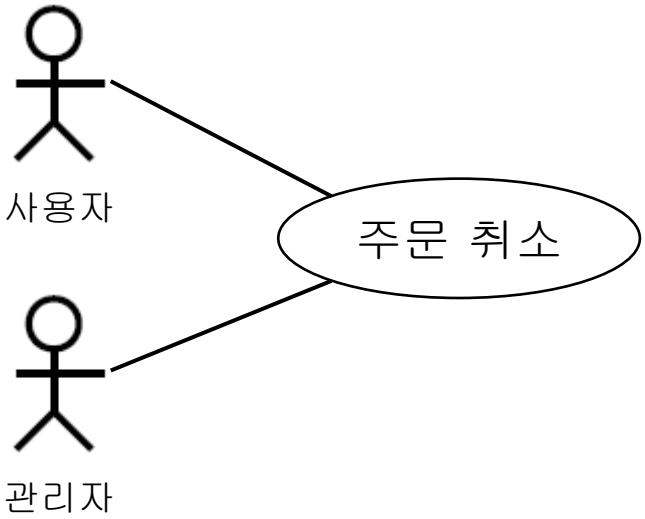
# Use case (1/2)

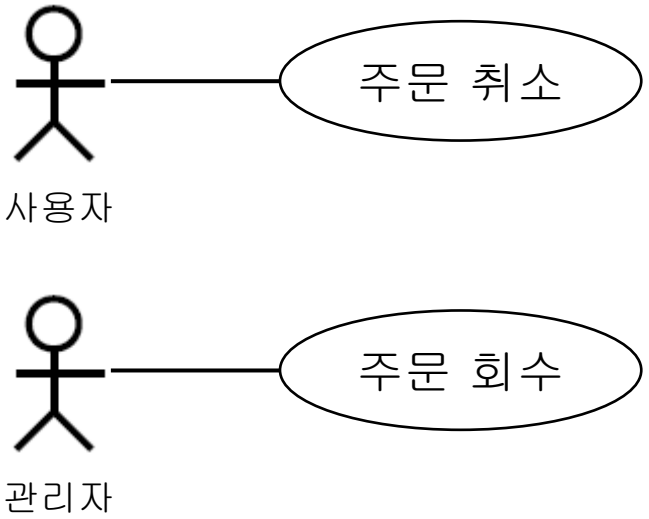
- **의미**
  - Actor에게 서비스를 제공하기 위하여 시스템이 수행하는 중요 기능
  - 시스템의 요구사항을 보여줌
- **표기법**
  - 타원으로 표시하고 그 안쪽이나 아래쪽에 Use case명을 기술
  - Use case의 이름은 "~한다"와 같이 동사로부터 추출
  - 각 Use case가 개발될 기능 하나와 연결될 수 있도록 해야 함



# Use case (2/2)

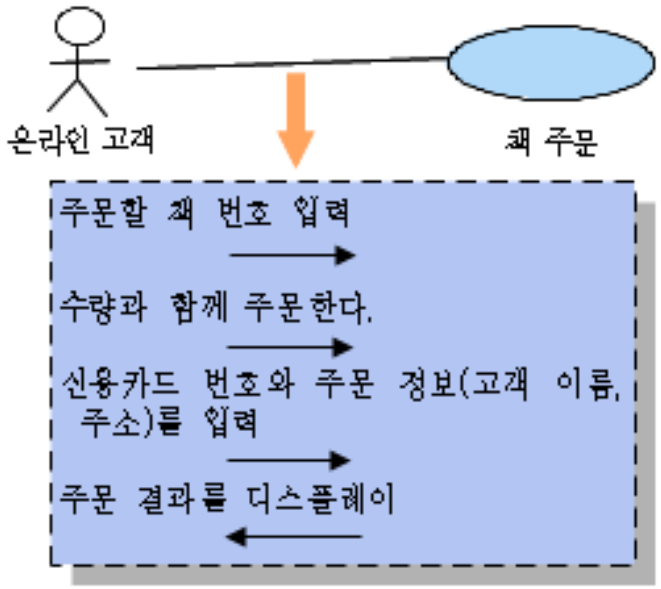
- **유의사항**
  - 단일 Use case가 여러 Actor에 의하여 구동 될 때
    → 작업이 동일하면 같은 사용 사례
    → 이벤트 흐름이 다르면 다른 사용 사례



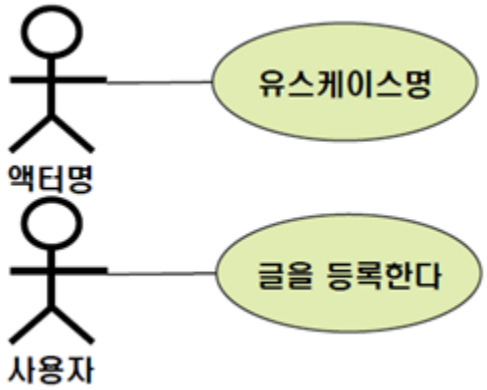
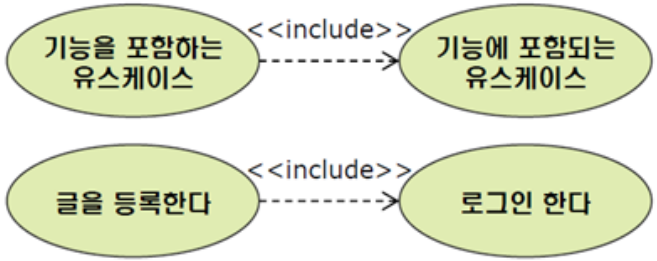



# Association (1/3)

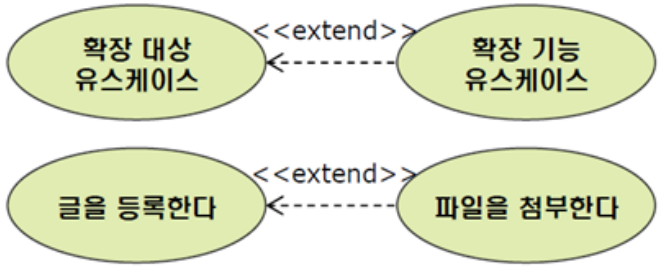
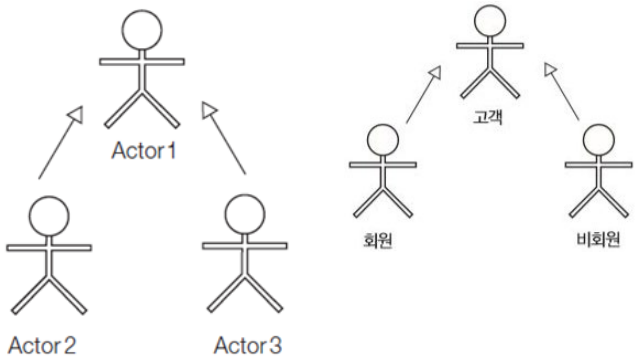
- 의미
  - Actor와 Use case 사이의 관계
- 표기법
  - Actor와 Use case사이의 실선으로 표시
  - 관계 특성에 따라 적절한 모양의 선으로 표시해야 함
    - → Association (연관관계)
    - → Dependency (의존관계)
    - → Generalization (일반화 관계)



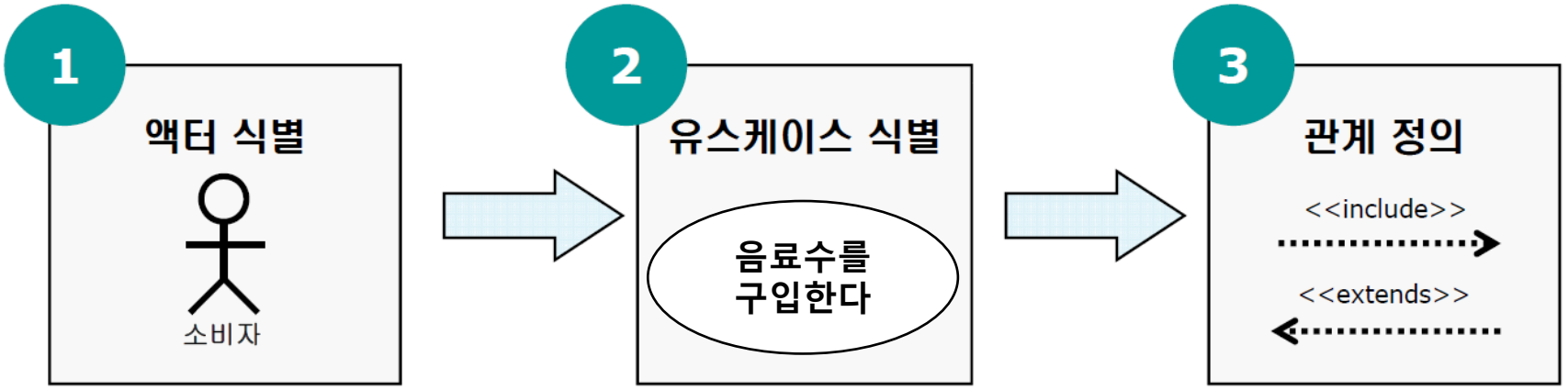
# Association (2/3)

| 관계 종류 | 설명 | 표기법 |
|---|---|---|
| **연관 관계 (association)** | • 유스케이스와 액터간의 상호작용이 있음을 표현<br>• 유스케이스와 액터를 실선으로 연결함 |  |
| **포함 관계 (include)** | • 하나의 유스케이스가 다른 유스케이스의 실행을 전제로 할때 형성되는 관계<br>• 포함되는 유스케이스는 포함하는 유스케이스를 실행하기 위해 반드시 실행되어야 하는 경우에 적용<br>• '포함하는 유스케이스 ' 에서 '포함되는 유스케이스 ' 방향으로 화살표를 점선으로 연결하여 표현하고 <<include>>라고 표기 |  |

# Association (3/3)

| 관계 종류 | 설명 | 표기법 |
|---|---|---|
| **확장 관계 (extend)** | • 확장 기능(Extending) 유스케이스와 확장 대상(Extended) 유스케이스 사이에 형성되는 관계<br>• 확장 대상 유스케이스를 수행할 때에 특정 조건에 따라 확장 기능 유스케이스를 수행하기도 하는 경우에 적용<br>• '확장 기능 유스케이스 ' 에서 '확장 대상 유스케이스 ' 방향으로 화살표를 점선으로 연결하여 표현하고 <<extend>> 표기 |  |
| **일반화 관계 (generalization)** | • 액터들이 유스케이스와 중복하여 관계가 나타나면 액터들을 통합하여 일반화 관계로 표현<br>• 추상적인 액터와 좀 더 구체적인 액터 사이에 관계를 맺어줌 |  |

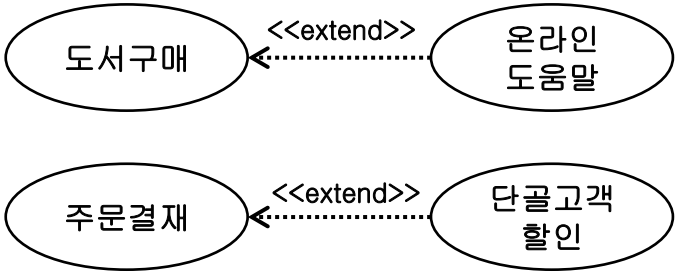
# Use case diagram 작성 순서




- 사용자 식별
- 외부 시스템 식별

- 액터가 요구하는 서비스 식별
- 액터가 요구하는 정보 식별
- 액터가 시스템과 상호작용하는 행위 식별

- 액터와 유스케이스간의 연관관계 정의
- 유스케이스간 포함, 확장관계 정의
- 액터간, 유스케이스간의 일반화 정의

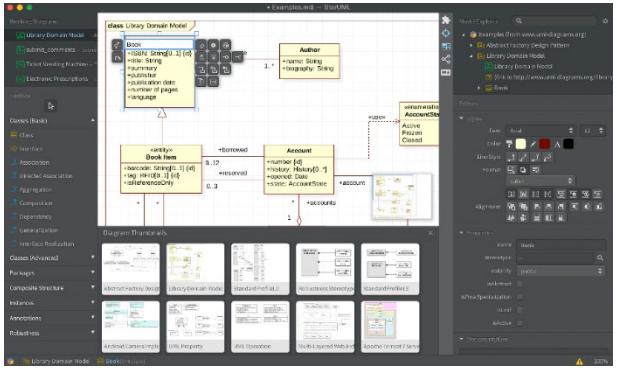


# Use case 작성 Tip

- 소프트웨어가 제공할 기능에 집중하라.
- 사용자 관점으로 작성하라.
- 목표 지향적으로 작성하라.
- 시스템 흐름도와는 다름.
- 읽기 쉽게 작성하라.

# StarUML

- **StarUML**
  - 상업적 도구에 준하는 기능을 갖춘 소프트웨어 모델링 도구
  - UML 표준 spec에 기반한 모델 작성 가능
  - 뛰어난 확장성과 유연성을 제공
  - Reverse engineering 가능
  - 제공하는 모든 기능 사용을 위해서는 License가 필요
- **StarUML - Install**
  - http://staruml.io/
  - OS에 맞는 버전으로 선택하여 설치해야 함

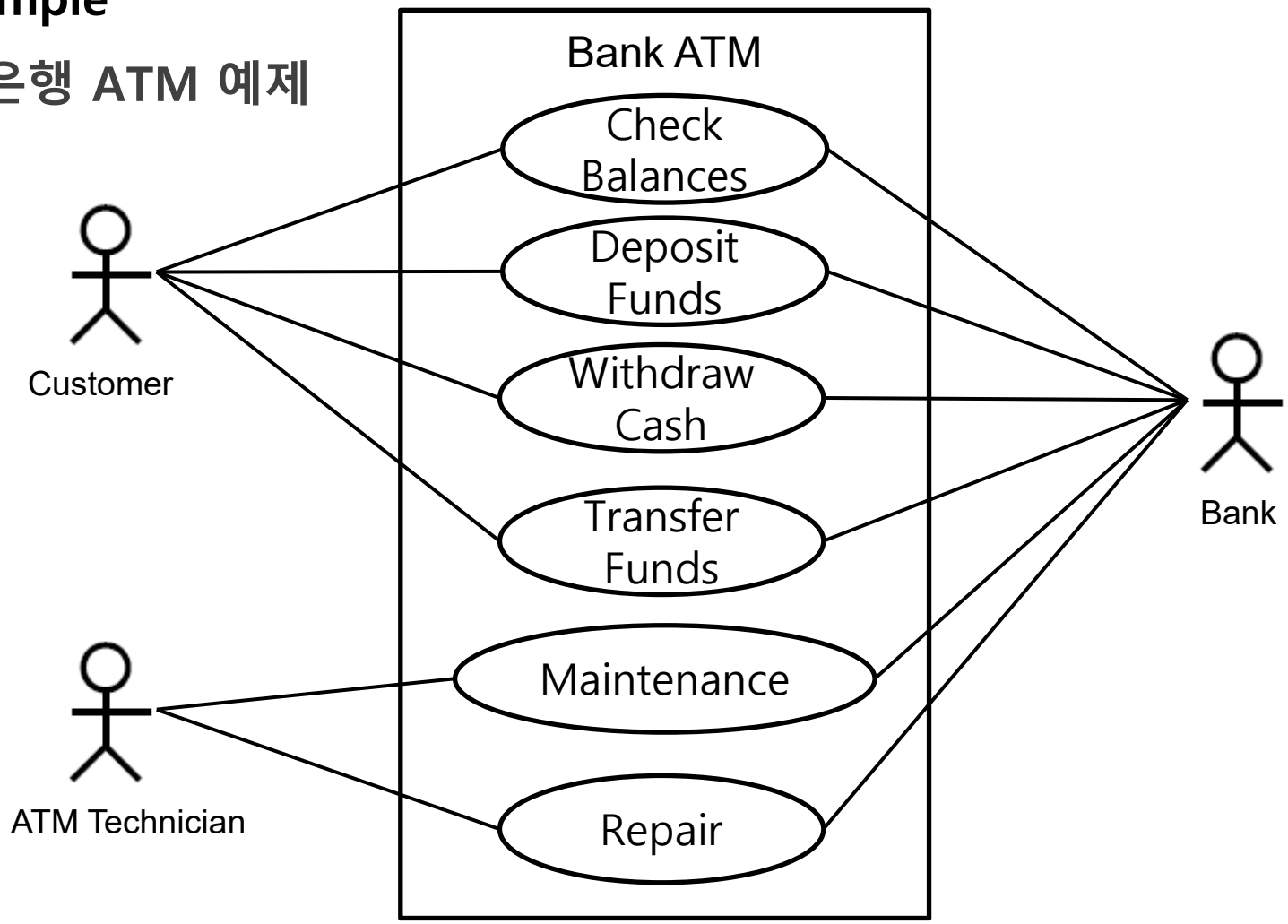


StarUML™
The Open Source UML/MDA Platform

참고: 각 단계별 SW 개발도구 모음
https://www.swbank.kr/helper/tool/toolMain.do

# 실습

- **Example**
  - 은행 ATM 예제



# Use case diagram의 구성요소 식별법

- **Actor를 찾기 위한 질문들**
  - 누가 정보를 제공하고, 사용하고, 삭제하는가?
  - 누가 또는 어떤 조직에서 개발될 시스템을 사용할 것인가?
  - 누가 요구사항에 대해 관심을 가지고, 시스템이 만들어낸 결과에 관심이 있는가?
  - 누가 시스템이 잘 운영될 수 있도록 유지보수 및 관리를 하는가?
  - 개발될 시스템과 상호작용하는 하드웨어나 소프트웨어 시스템은 무엇인가?

# Use case diagram의 구성요소 식별법

- **Use case를 찾기 위한 질문들**
  - Actor가 원하는 System 제공 기능은 무엇인가?
  - Actor는 System에 어떤 정보를 생성, 수정, 조회, 삭제하고 싶어 하는가?
  - Actor는 System의 갑작스러운 외부 변화에 대해 어떤 정보를 필요로 하는가?
  - System이 어떤 기능을 제공하면 Actor의 일상 작업이 효율적이고 편리해지는가?
  - 모든 기능 요구사항들을 만족할 수 있도록 Use case가 모두 식별되었는가?

# Use case diagram의 구성요소 식별법

- **Association을 판단하기 위한 질문들**
  - 연관 관계(Association)

    → Actor와 Use case 간에 상호 작용이 존재하는가?
  - 포함 관계(Include)

    → 이 Use case를 실행하기 위하여 반드시 실행되어야 하는 Use case가 존재하는가?
  - 확장 관계(Extend)

    → 이 Use case를 실행함으로써 선택적으로 실행되는 Use case가 있는가?
  - 일반화 관계(Generalization)

    → Actor or Use case가 구체화된 다른 여러 Actor나 Use case를 가지고 있는가?

# 실습 예제 (I)

- **예제 요구사항**
  - **SE사는 고객으로부터 다음의 요구사항을 전달받았다.**

**사용자 요구사항 (SRS)**
**: 글을 등록, 수정, 삭제할 수 있는 게시판을 개발한다.**
**: 단, 관리자 모드는 개발하지 않는다.**

*** 조건**
- 글을 등록할 때에는 파일을 첨부할 수 있다.
- 글을 조회하여 읽을 수 있다.
- 등록된 글은 글쓴이 혹은 날짜 별로 검색할 수 있다.
- 게시판의 글 등록, 수정, 조회, 삭제 기능은 사용자 로그인 후에 사용할 수 있다.

# 실습 예제 (I)

- **(1) System 식별**
  - 요구사항을 통해 만들고자 하는 시스템은 '게시판'임

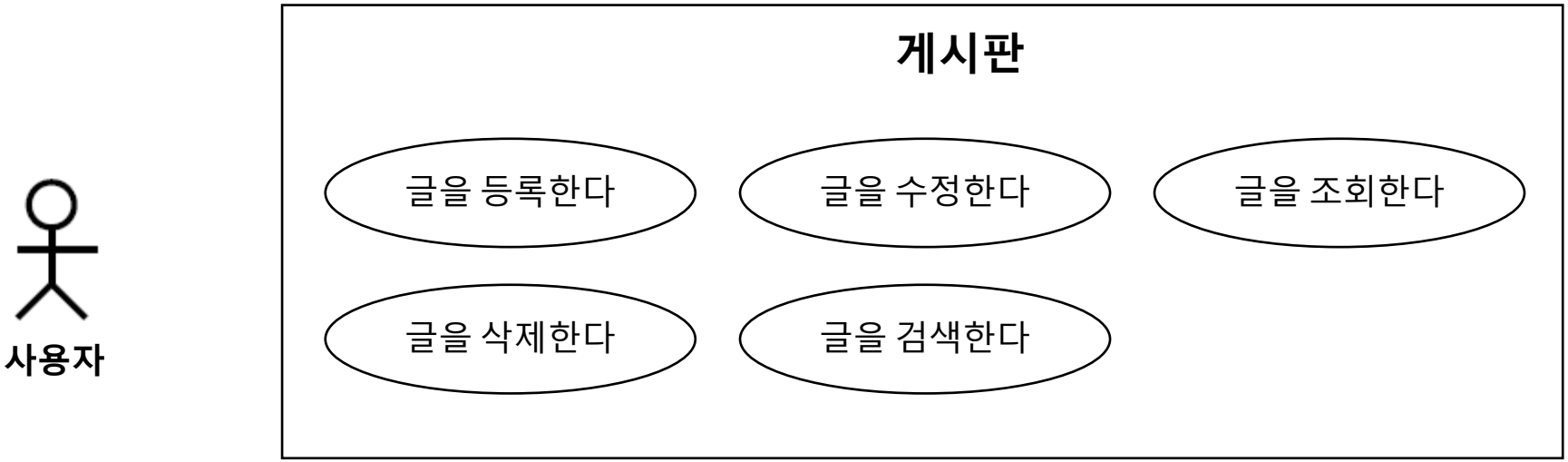
# 실습 예제 (I)

- **(2) Actor 식별**
  - 개발할 '게시판' 외부에서 상호작용하는 Actor로 글을 등록하고 삭제하는 등의 역할을 하는 '사용자'가 식별됨

사용자

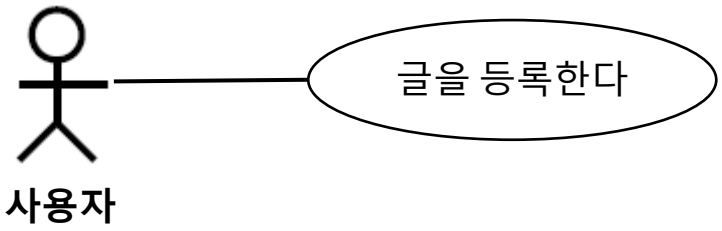
# 실습 예제 (I)

- **(3) Use case 식별**
  - '사용자'는 게시판을 통해 글을 등록, 수정, 조회하는 등의 작업을 함

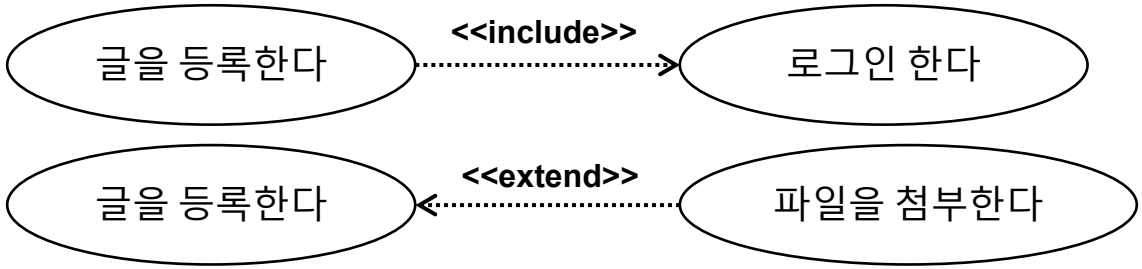


# 실습 예제 (I)

- **(4) Association 정의**
  - **Association (Actor와 Use case 사이)**

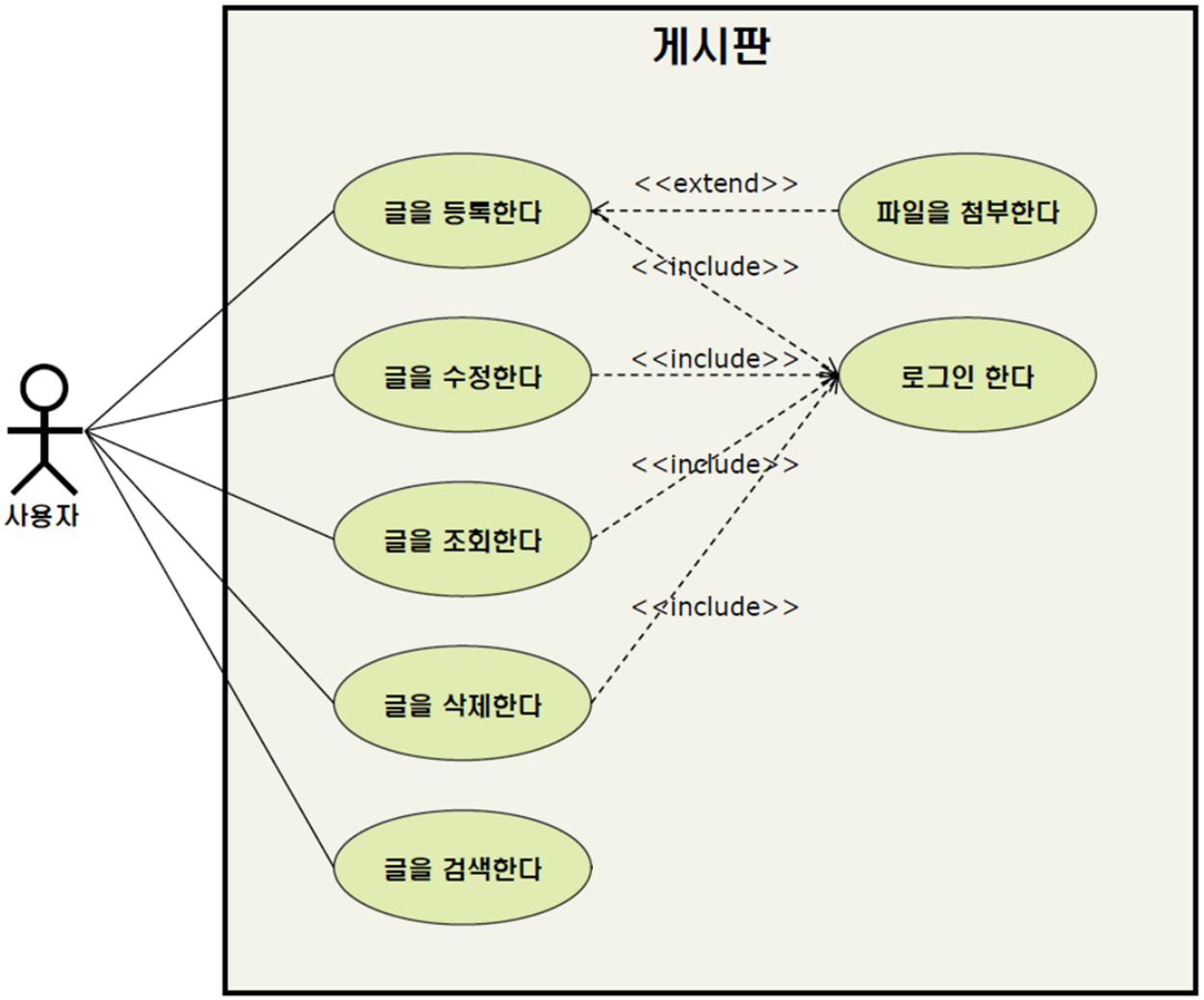



  - **Include, Extend (Use case와 Use case 사이)**



# 실습 예제 (I)

- 결과 예


게시판
글을 등록한다
파일을 첨부한다
<<extend>>
<<include>>
글을 수정한다
<<include>>
로그인 한다
<<include>>
글을 조회한다
<<include>>
글을 삭제한다
글을 검색한다
사용자

# 실습 예제 (II)

## ▪ 도서 관리 시스템 (SRS)

- 도서 관리 시스템
  - 고객은 반드시 회원으로 가입해야 도서를 대여가능
  - 대여, 반납, 연체 관리 기능이 있음
- 관리자
  - 이름과 전화번호로 회원을 확인
  - 연체 관리 기능을 통해 현재 연체 중인 회원과 연체된 도서를 확인
  - 연체금 표시 기능을 사용해 오늘 날짜에 해당하는 연체금을 표시
  - 반납 기능을 통해 반납한 도서 코드를 입력하여 대여 목록에서 삭제
  - 새로운 도서의 등록 및 삭제를 관리할 수 있음
  - 대여할 때는 고객이 도서를 선택하면 도서 코드를 확인하여 시스템에 입력
- 대여
  - 해당 고객이 현재 대여 중인 도서가 있으면 표시하고, 대여 기간이 지났으면 연체료를 계산하여 보여줌
  - 연체 고객은 연체료를 납부하면 도서를 대여할 수 있음
  - 대여료와 연체료는 현금이나 신용카드 결제를 통해 이루어짐
  - 대여된 도서는 대여 목록에 도서 코드와 고객명으로 등록

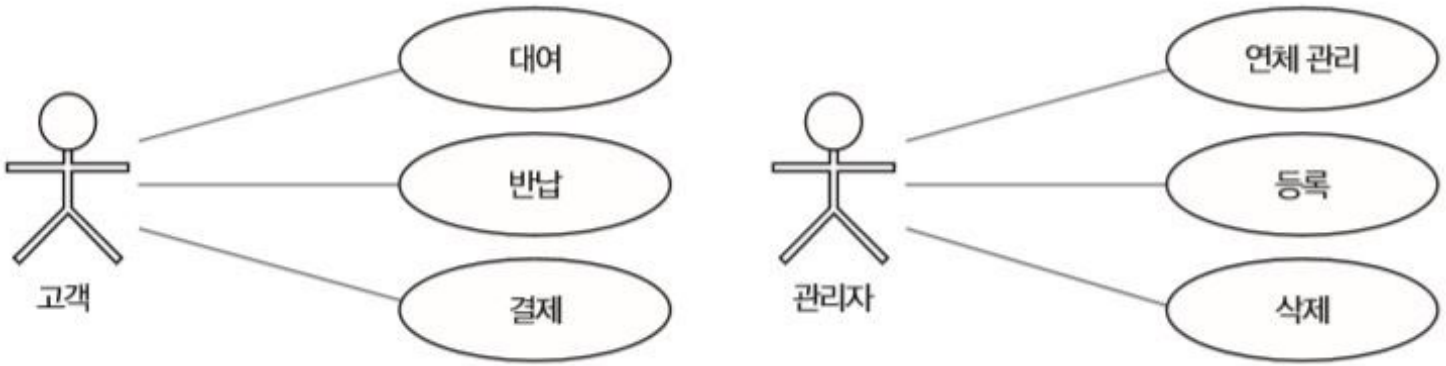
# 실습 예제 (II)

- **도서 관리 시스템**
  - (1) Actor 식별


  - (2) Use case 식별

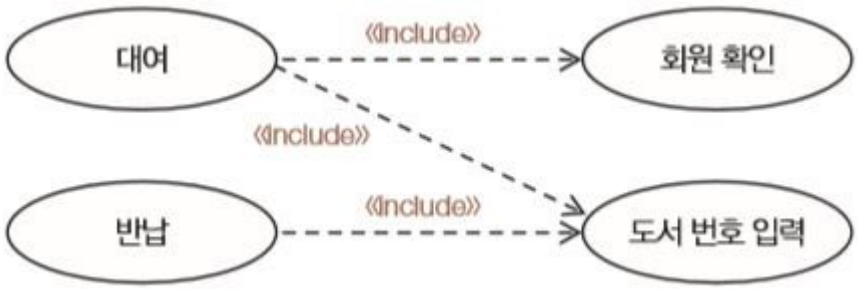
# 실습 예제 (II)

- **도서 관리 시스템**
  - (3) Use case diagram 작성
    - Actor와 Use case, Use case와 Use case간의 관계를 설정해 표현
    - 고객 : 대여,반납,결제
    - 관리자 : 연체관리,등록,삭제



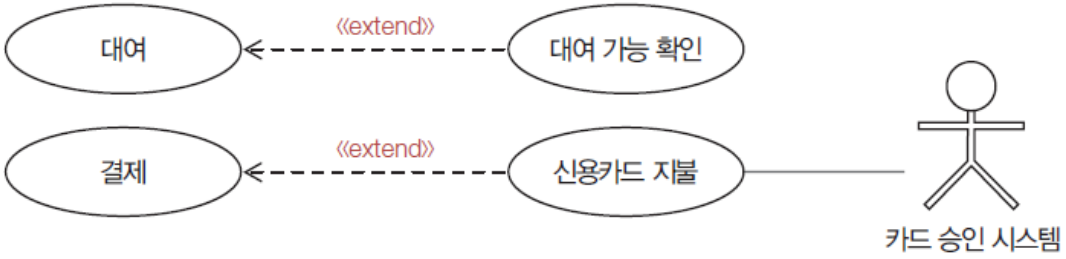
# 실습 예제 (II)

- 도서 관리 시스템
  - (3) Use case diagram 작성
    - <<include>>
    - 하나의 유스케이스를 수행 할 때 같은 기능을 가진 또 하나의 유스케이스를 반드시 수행해야 할 경우를 의미
    - 대여와 반납 유스케이스가 수행될 때는 반드시 도서 번호 입력 유스케이스가 선행(포함)필요
    - 대여할 때는 회원 확인 유스케이스도 포함되어야 함



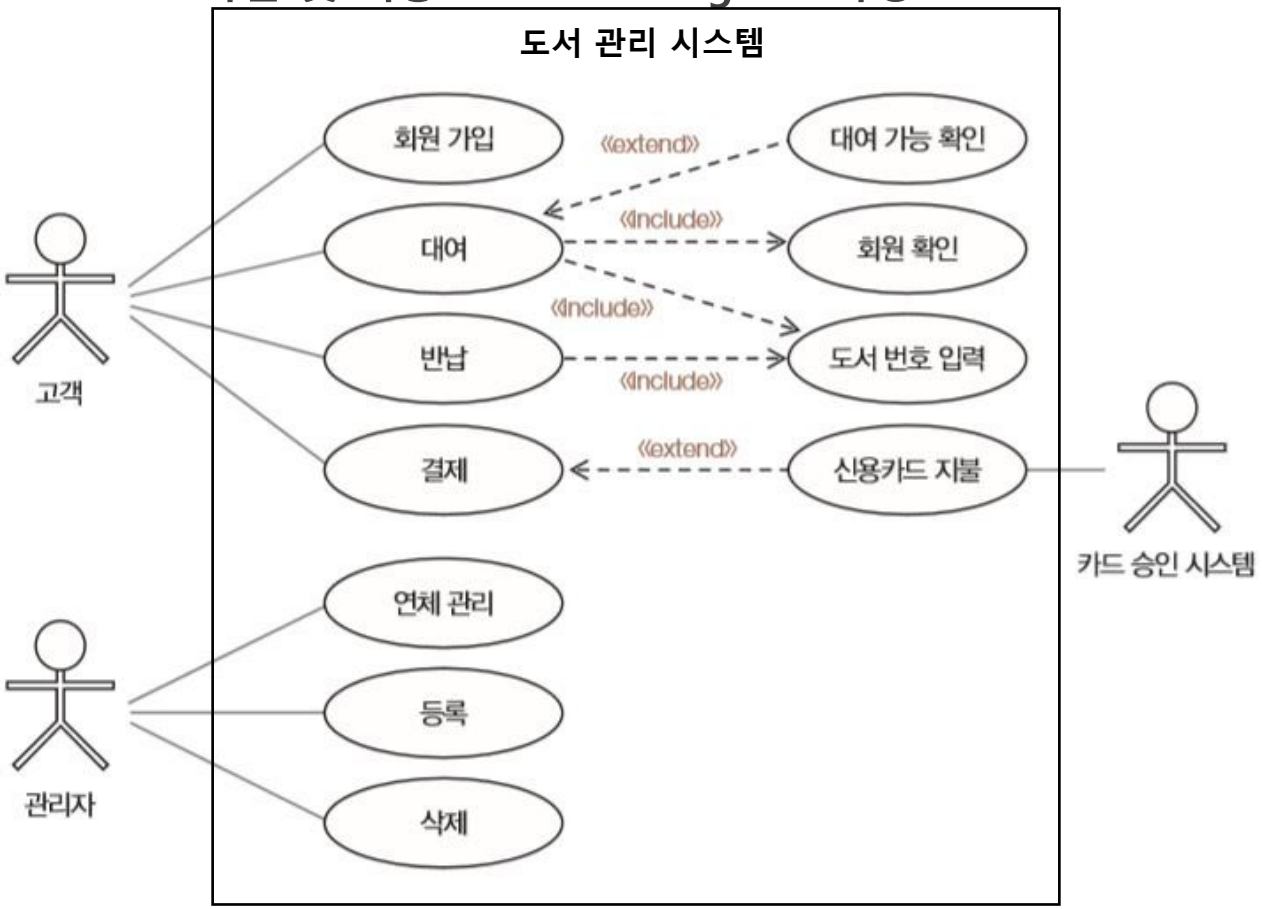
# 실습 예제 (II)

- 도서 관리 시스템
  - (3) Use case diagram 작성
    - <<extend>>
    - 여러 유스케이스에 걸쳐 중복 사용되지 않고, 특정 조건에서 한 유스케이스로만 확장되는 것을 의미
    - 상위 유스케이스로부터 어떠한 특정 조건에 의해 수행됨
    - 결제를 수행할 때는 결제 유스케이스로부터 신용카드 지불 유스케이스가 확장되는 형태로 이루어져야 함
    - 신용카드 지불 시 카드 승인사에 카드 승인을 요청해야 하므로, 카드 승인 시스템 액터와도 관계를 설정해야 함



# 실습 예제 (II)

- 도서 관리 시스템
  - (4) Association 식별 및 최종 Use case diagram 작성



# 실습 예제 (III)

- **카카오 택시 (SRS)**
  - 카카오택시 운영을 위한 간단한 SW시스템을 구축하고자 한다.
    - 택시 호출 :
      - 승객이 출발지와 도착지를 설정해서 택시를 호출
      - 택시 기사는 출발지와 도착지를 확인한 후 택시 콜을 받음
      - 승객은 택시 호출을 취소할 수 있고 택시 기사도 배차를 취소할 수 있음
      - 택시 결제는 현금이나 신용카드를 통해 이루어짐
    - 이용 기록 조회 :
      - 승객은 이용 기록을 조회할 수 있음
      - 승객은 이용 기록을 통해 택시 기사명과 택시 정보에 대해 확인할 수 있음

# 실습 예제 (III)

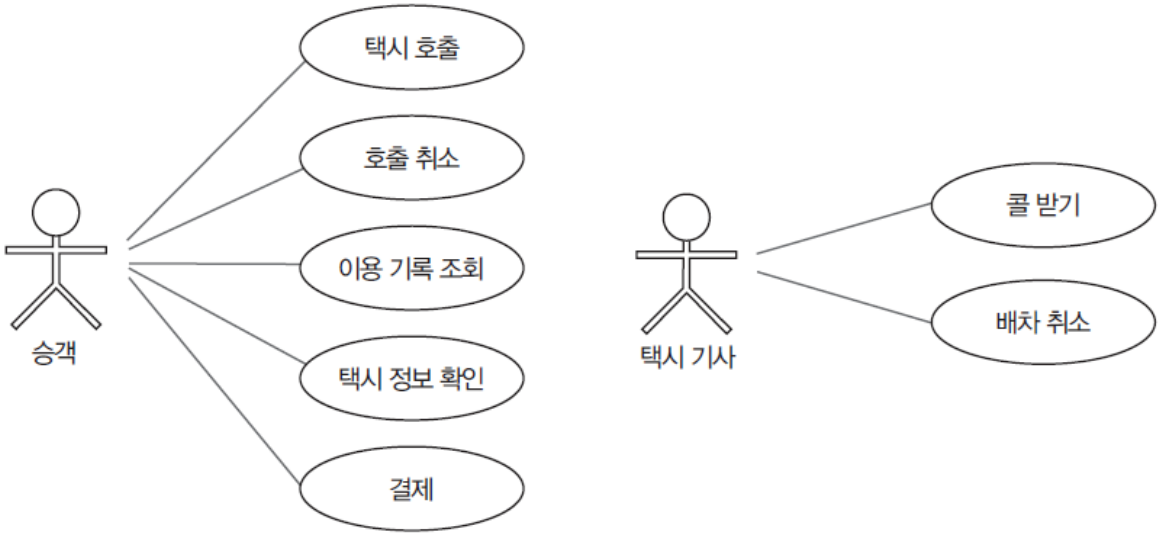
- **카카오 택시**
  - (1) Actor 식별


  - (2) Use case 식별

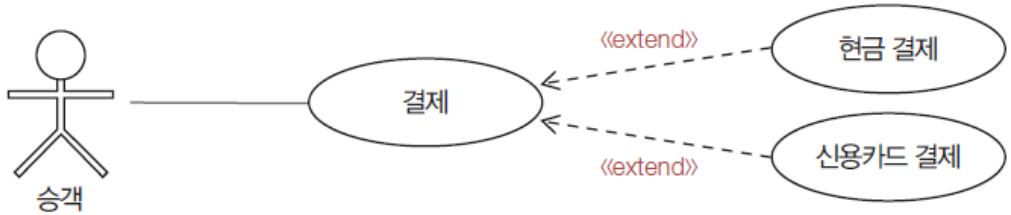
# 실습 예제 (III)

## ▪ 카카오 택시

- (3) Use case diagram 작성



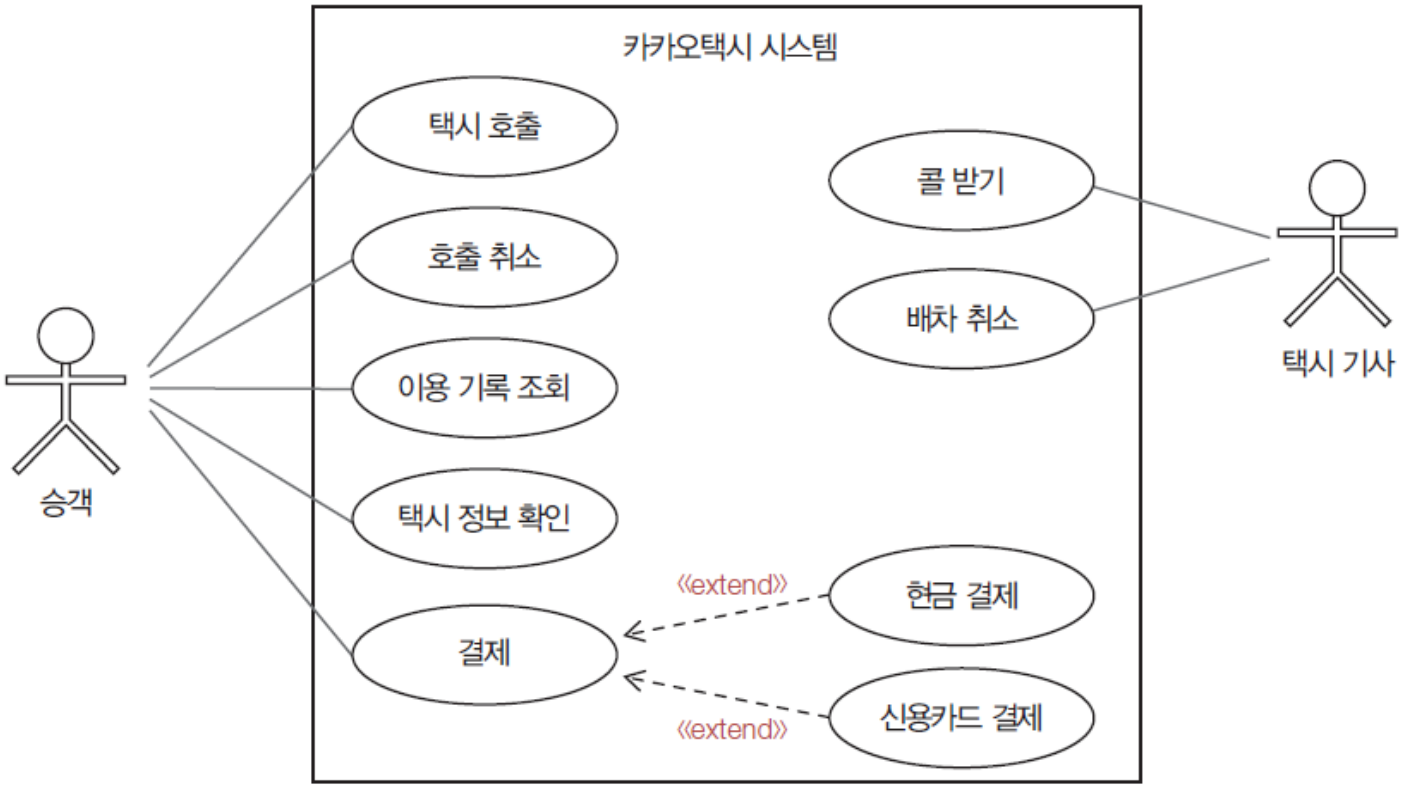

**[Actor와 Use case간의 관계 설정하기]**




**[Use case와 Use case간의 관계 설정하기]**

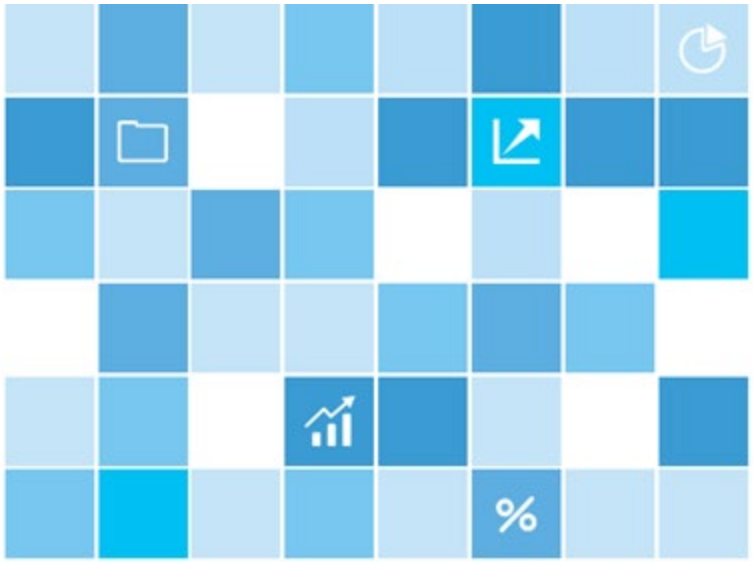
# 실습 예제 (III)

- **카카오 택시**
  - (4) 최종 Use case diagram 작성



# Summary: 검토 기준

| | |
|---|---|
| Actor | **Actor는 시스템과 상호작용을 하는 시스템 외부의 존재이다.** |
| | Actor는 시스템과 상호작용을 해야 한다. |
| | Actor는 시스템 관점에서 바라본 사용자의 역할을 뜻해야 한다. |
| | 시스템의 사용자와 외부 연동되는 시스템을 정확하게 파악할 수 있어야 한다. |
| | Actor의 이름만으로 해당 Actor의 역할을 명확하게 이해할 수 있어야 한다. |
| Use case | Use case는 개발 대상이 되는 시스템이 제공하는 개별적인 기능을 뜻한다. |
| | Use case로 표현된 기능은 시스템의 사용자가 이용한다. |
| | Use case의 기능과 이를 이용하는 Actor 사이의 연관관계를 명시적으로 표현한다. |
| | 시스템의 전체 기능적 요구사항은 표현된 use case로 구성된다. |
| | **Use case는 사용자가 인지할 수 있는 하나의 기능 단위이다.** |
| | Use case는 구체적이어야 한다. |
| | 하나의 독립적인 기능을 구성하는 다양한 세부 상황은 하나의 Use case로 표현되어야 한다. |
| | 반드시 한 개 이상의 활성화 상호작용을 하는 Actor가 있어야 한다. |
| | Use case는 모든 활성화 Actor에게 동일한 기능을 제공해야 한다. |
| | Use case는 트랜잭션 성격을 가져야 한다. |
| | Use case 이름으로부터 해당 Use case가 나타내는 시스템의 기능을 명확하게 이해할 수 있어야 한다. |
| Association | Actor와 Use case 간의 연관 관계는 둘 간의 상호작용을 뜻한다. |
| | **Association는 반드시 시스템이 제공하는 기능이어야 한다.** |
| | Association의 방향은 제어 흐름을 뜻해야 한다. |



# Thank You