[찾아보기]

왜?

찾고자하는 내용

예) 로그인, 게시글등.

[개념적 이해].

Use Case Modeling

Prof. Kim, Hyeon Soo

충남대학교
CHUNGNAM NATIONAL UNIVERSITY

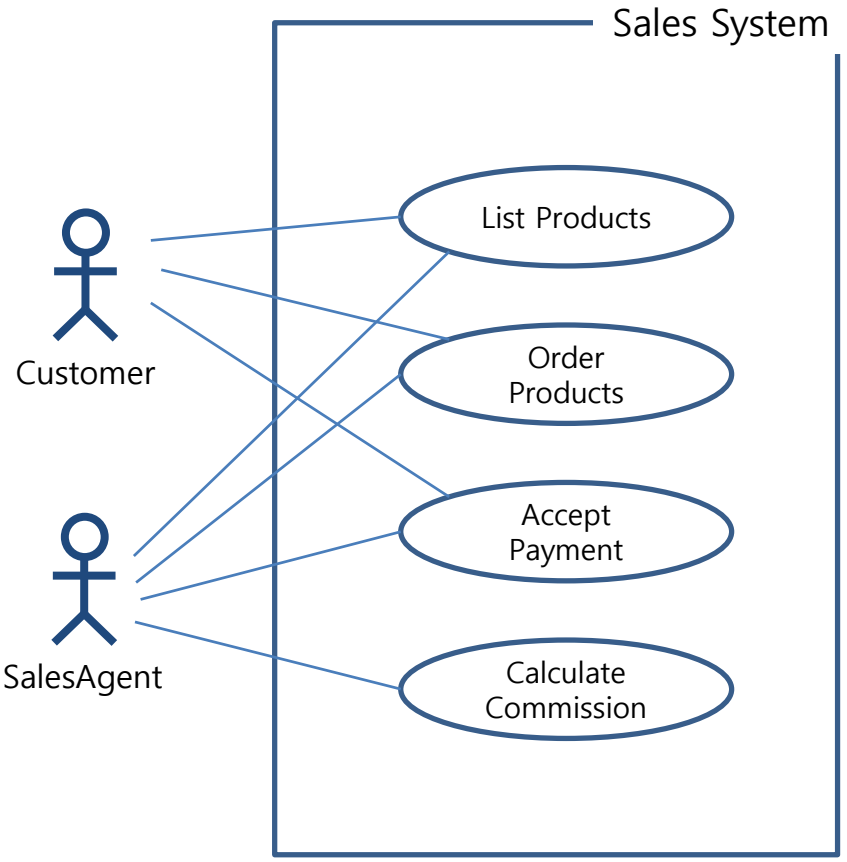
SEAL Software Engineering & Application Lab.
소프트웨어 공학 및 응용 연구실

Use Case Modeling

- **Use Case Modeling**
  - Another form of requirements engineering
    - ≫ Find the subject
    - ≫ Find actors
    - ≫ Find use cases
  - Subject
    - ≫ what is part of the system, what is external to the system
  - Actor
    - ≫ roles played by things external to the system that interact directly with the system
  - Use cases
    - ≫ functions that the system performs on behalf of specific actors
      - ✓ considering what functions the system offers to the actors
      - ✓ always started by an actor
      - ✓ always written from the point of view of the actors

Use Case Modeling

- **Use Case Diagram**
  - The subject
  - Actors
  - Use cases
  - Interactions



Use Case Modeling

- **Use Case Specification**
  - Preconditions
    - ≫ System constraints that affect the execution of a use case
  - Postconditions
    - ≫ System constraints arising from the execution of a use case
  - Find key alternatives flows
    - ≫ Examine the main flow and look for
      - ✓ Alternatives
      - ✓ Error situations
      - ✓ Interrupts

Use case: PaySalesTax
ID: 1
Brief description: Pay sales tax to the Tax Authority at the end of ….
Primary actors: Time
Secondary actors: TaxAuthority
Preconditions: 1. It is the end of the business quarter.
Main flow: 1. The use case starts …. 2. The system determines the amount …. 3. The system sends an electronic payment ….
Postconditions: 1. The Tax Authority receives the correct amount ….
Alternative flows: None.

Advanced Features 사용

- **Actor Generalization**
  - factor out into a parent actor behaviors that are common to two or more actors
  - can simplify use case diagrams

- **Use Case Generalization**
  - factor out features that are common to two or more use cases into a parent use case

- **<<include>> Relationship**
  - factor out steps repeated in several use case flows into a separate use case
  - **include(UseCaseName)** is used to include the behavior of another use case
  - Including use case is the base use case
  - Included use case is the inclusion use case
  - Base use case is not complete without all of its inclusion use cases

Advanced Features 사용

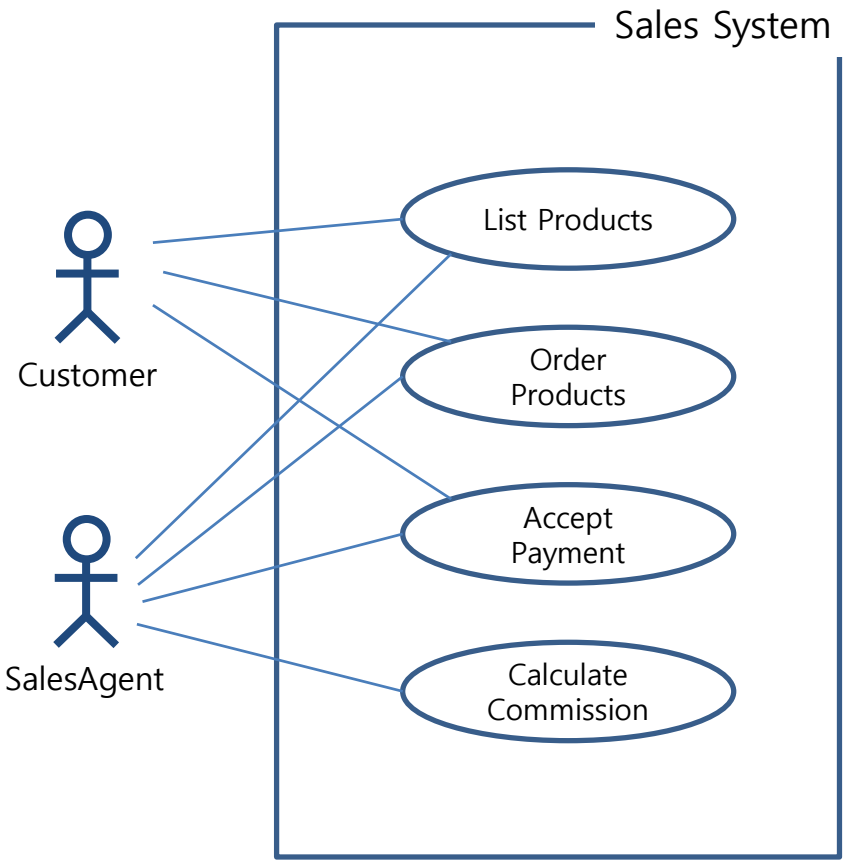
- **<<extend>> Relationship**
  - Adds new behavior to an existing (base) use case
  - Base use case has extension points in an overlay on its flow of events
    - ≫ Extension points occur between the steps in the flow of events
  - Extension use case provide insertion segments
    - ≫ These are behavior fragments that may be "plugged into" extension points
  - Base use case is complete without the insertion segments
  - Extension use case is generally not complete
    - ≫ just consists of one or more insertion segments

Advanced Features 사용

- **Use advanced features as follows:**
  - Actor generalization
    - ≫ Use only where it simplifies the model
  - Use case generalization
    - ≫ Consider *not* using, or only using with abstract parents
  - <<include>>
    - ≫ Use only where it simplifies the model
    - ≫ Beware of overuse, as this makes a use case model turn into a functional decomposition
  - <<extend>>
    - ≫ Consider *not* using
    - ≫ But if you do use it, be careful to ensure that all modelers and stakeholders understand and agree on its semantics

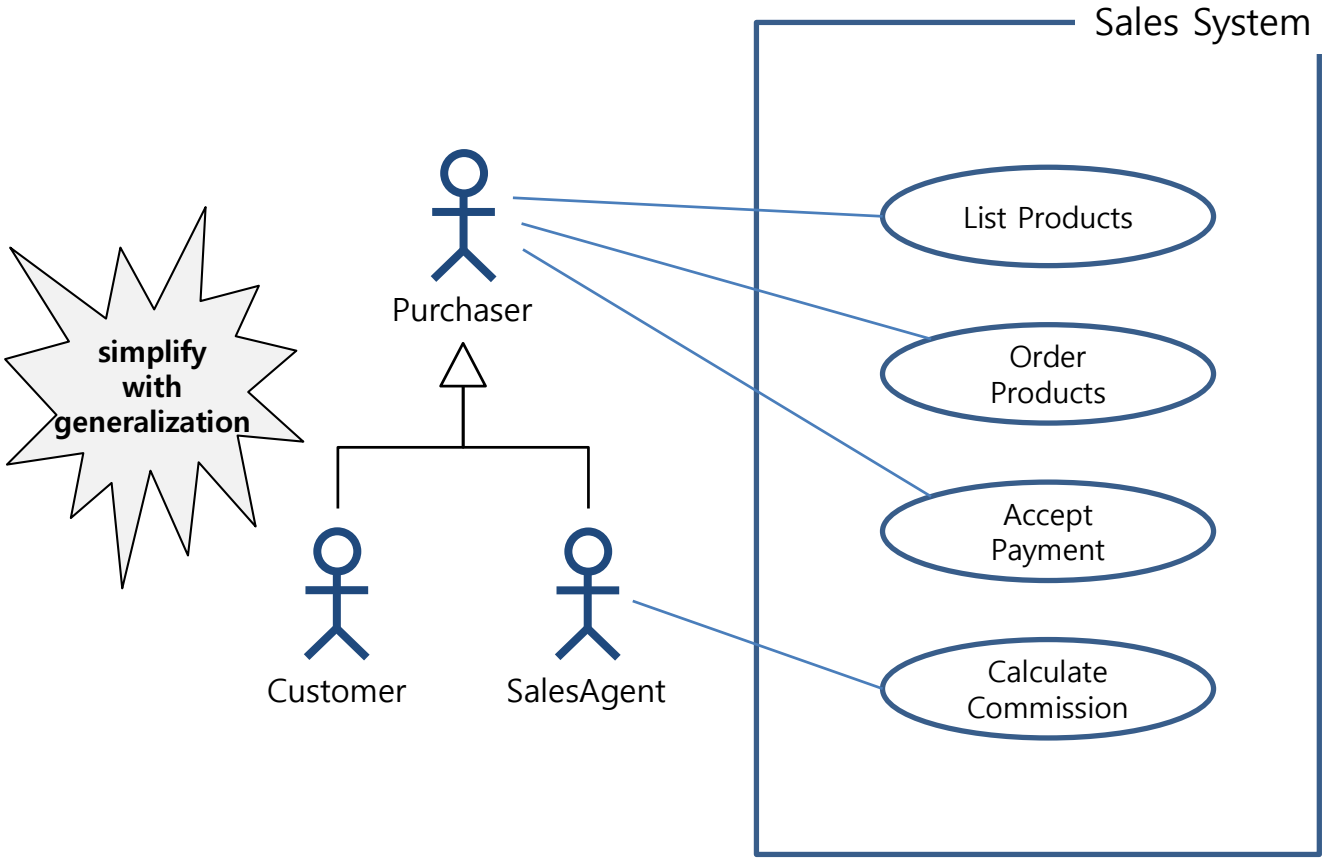
Actor Generalization

- **Use Case Diagram for Sales System – v.1**



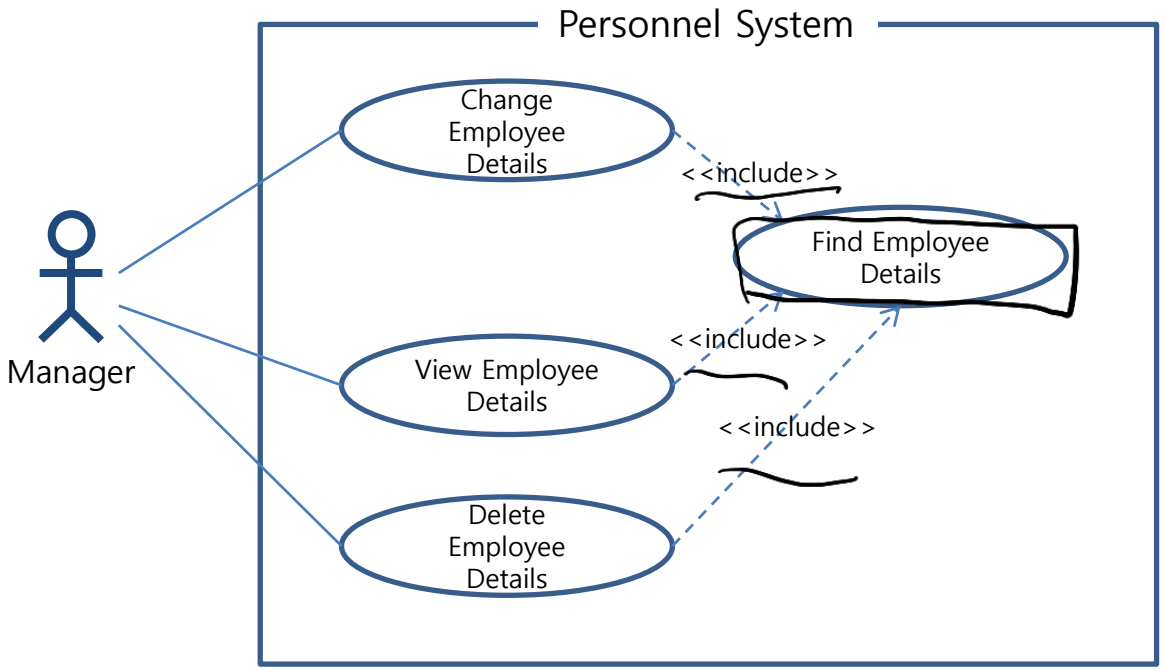
Actor Generalization

- **Use Case Diagram for Sales System – v.2**
  - Actor Generalization



<<include>> Relationship

- **Use Case Diagram – <<include>>활용**



<<include>> Relationship

- **Use Case Diagram – <<include>>활용**

Use case: ChangeEmployeeDetails
ID: 1
Brief description: The Manager changes the employee details.
Primary actors: Manager
Secondary actors: None.
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. include(FindEmployeeDetails). 2. The system displays the employee details. 3. The Manager changes the employee details. ...
Postconditions: 1. The employee details have been changed.
Alternative flows: None.

Use case: ViewEmployeeDetails
ID: 2
Brief description: The Manager views the employee details.
Primary actors: Manager
Secondary actors: None.
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. include(FindEmployeeDetails). 2. The system displays the employee details. ...
Postconditions: 1. The system has displayed the employee details.
Alternative flows: None.

<<include>> Relationship

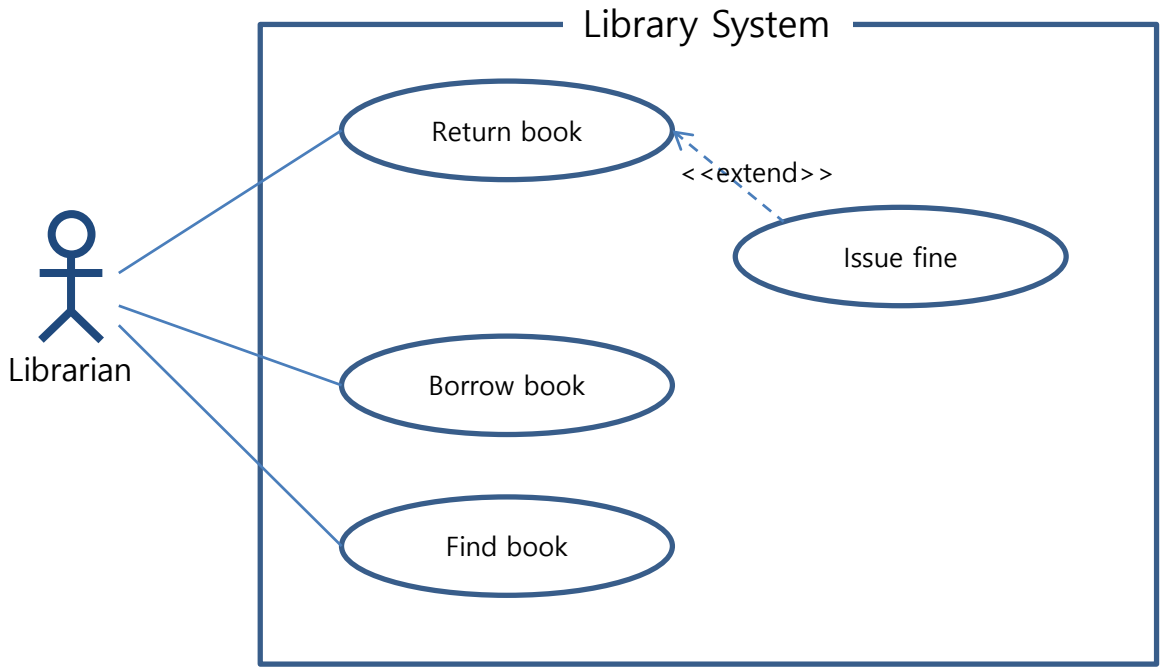
- **Use Case Diagram – <<include>>활용**

Use case: DeleteEmployeeDetails
ID: 3
Brief description: The Manager deletes the employee details.
Primary actors: Manager
Secondary actors: None.
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. include(FindEmployeeDetails). 2. The system displays the employee details. 3. The Manager deletes the employee details. ...
Postconditions: 1. The employee details have been deleted.
Alternative flows: None.

Use case: FindEmployeeDetails
ID: 4
Brief description: The Manager finds the employee details.
Primary actors: Manager
Secondary actors: None.
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. The Manager enters the employee's ID. 2. The system finds the employee details.
Postconditions: 1. The system has found the employee details.
Alternative flows: None.

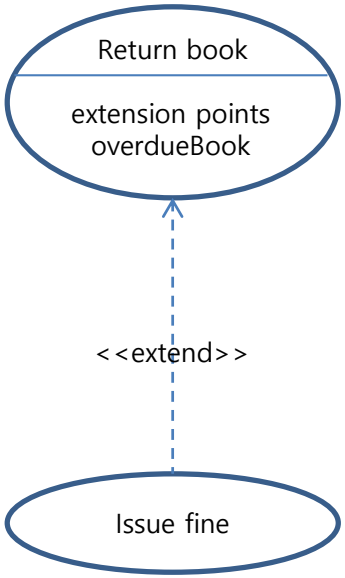
<<extend>> Relationship

- **Use Case Diagram – <<extend>>활용**



<<extend>> Relationship

- **Use Case Diagram – <<extend>>활용**



Use case: ReturnBook
ID: 9
Brief description: The Librarian returns a borrowed book.
Primary actors: Librarian
Secondary actors: None.
Preconditions: 1. The Librarian is logged on to the system.
Main flow: 1. The Librarian enters the borrower's ID number. 2. The system displays the borrower's details including the list of borrowed books. 3. The Librarian finds the book to be returned in the list of books. extension point: overdueBook 4. The Librarian returns the book. ...
Postconditions: 1. The book has been returned.
Alternative flows: None.

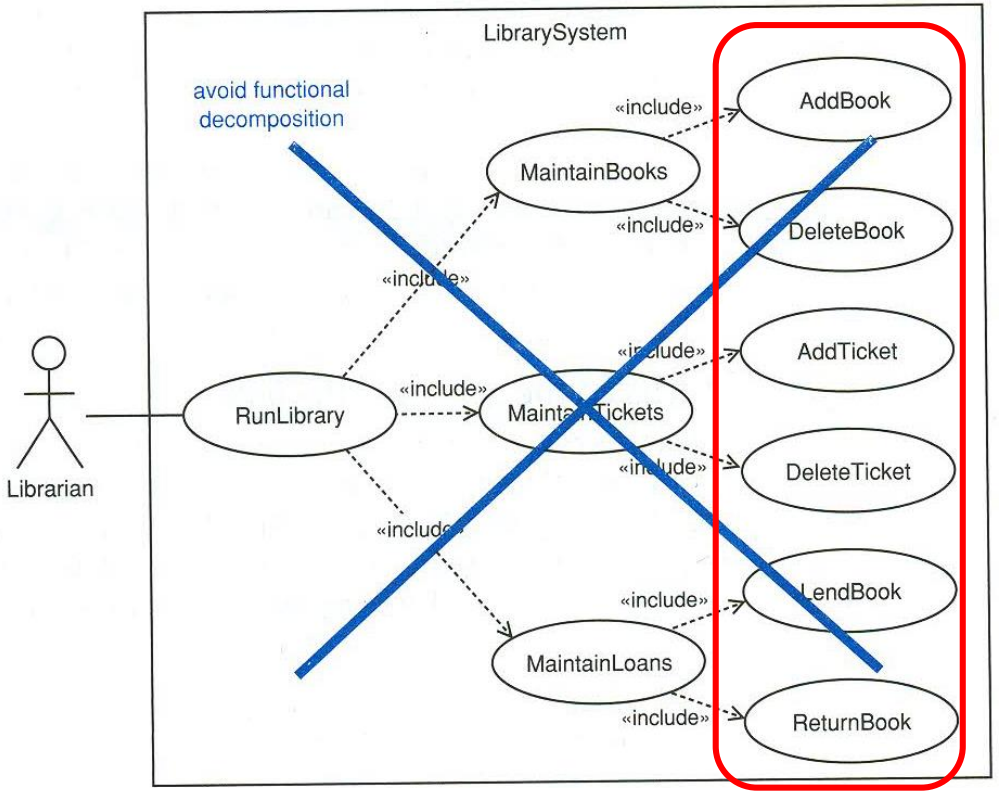
Extension Use case: IssueFine
ID: 10
Brief description: Segment 1: The Librarian records and prints out a fine.
Primary actors: Librarian
Secondary actors: None.
Segment 1 preconditions: 1. The returned book is overdue.
Segment 1 flow: 1. The Librarian enters details of the fine into the system. 2. The system prints out the fine.
Segment 1 postconditions: 1. The fine has been recorded in the system. 2. The system has printed out the fine.

Use Case Modeling

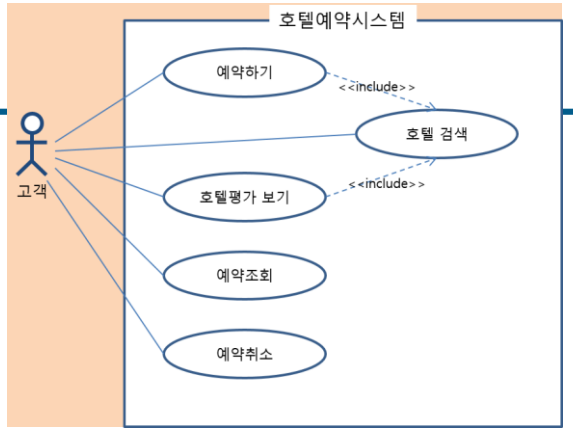
- **Hints and tips for writing Use Cases**
  - 유스케이스를 짧고 단순하게 유지하라
    - ≫ 요구사항을 포착하기에 충분한 세부 사항만 포함
    - ≫ 유스케이스의 주요 흐름은 종이의 한 면 이내로 작성
    - ≫ 텍스트 단순화부터 시작
    - ≫ 설계 세부 정보 제거
    - ≫ 시간이 너무 오래 걸리면 문제를 다시 분석하라
  - 어떻게(how)가 아니라 무엇(what)에 초점을 맞추어라
    - ≫ 시스템이 어떻게 수행해야 하는지가 아니라 시스템이 무엇을 수행할 것인가에 초점을 맞춰라
  - 기능 분할을 피하라
    - ≫ 일반적인 유스케이스 분석 오류
      - ✓ 고수준(high level) 유스케이스 집합을 만든 다음, 그것을 원시 유스케이스(primitive use cases)에 도달할 때까지 계속해서 하위 수준 유스케이스 집합으로 분할을 반복함

Use Case Modeling

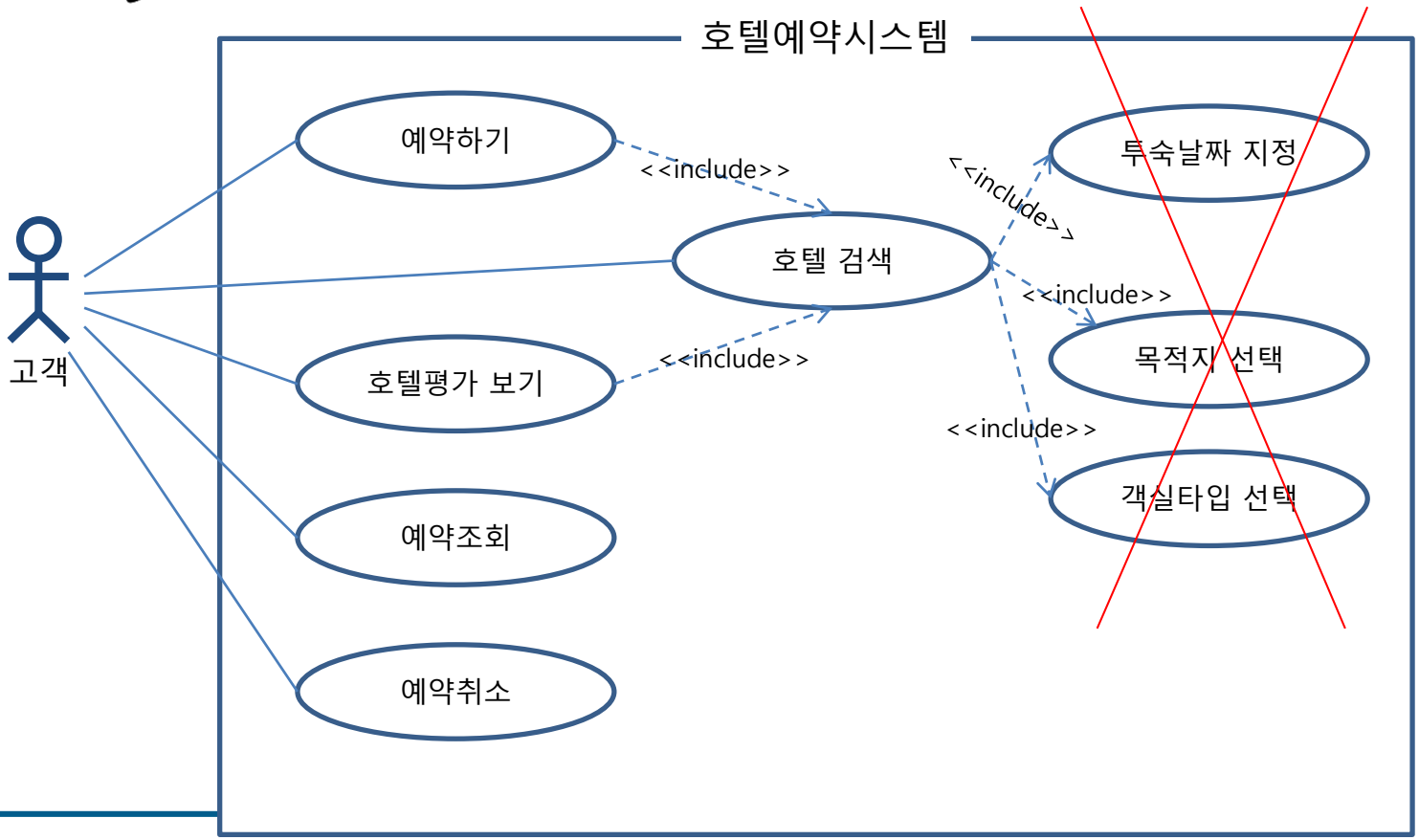
- **Hints and tips for writing Use Cases**
  - 기능 분할을 피하라
    - ≫ 계층 구조는 종종 유스케이스 모델링에서 자연스럽게 발생함
      - ✓ 이러한 계층 구조는 일반적으로 한 수준(또는 최대 두 수준) 이하여야 한다
      - ✓ 모델 전체가 단일 유스케이스에 뿌리를 두지 않아야 한다



Use Case Modeling

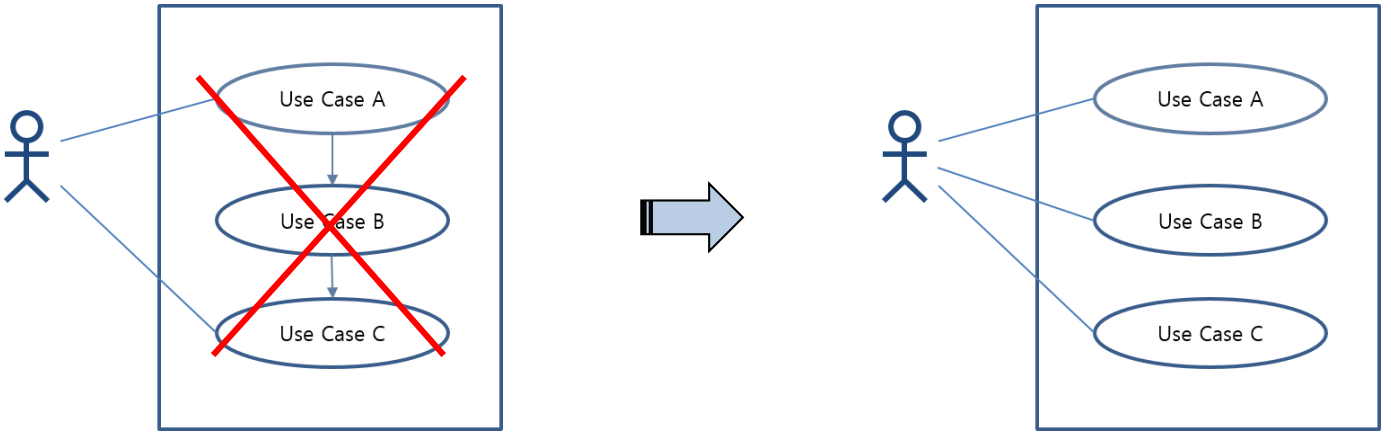


- **Hints and tips for writing Use Cases**
  - 기능 분할을 피하라
    - » 과도한 기능 분할을 피하라



Use Case Modeling

- **Hints and tips for writing Use Cases**
  - 유스케이스 순서화를 피하라
    - ≫ 유스케이스 간에 선후 관계에 대한 모델링이 필요한 경우, 유스케이스 다이어그램에 그것을 표현하지 말아야 함



Use case: B
ID: 0xx
Preconditions: 1. Use case A is done.
Main flow: 1. The use case starts …. 2. The system determines ….