



UML

목차

- ✓ Chap01. UML 개념
- ✓ Chap02. 유스케이스 다이어그램
- ✓ Chap03. 클래스 다이어그램
- ✓ Chap04. 시퀀스 다이어그램

Chap01.

UML 개념

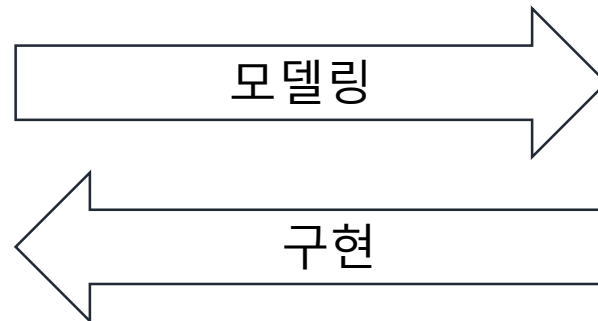
▶ 모델링과 UML

✓ 모델링

말 그대로 모델을 만드는 작업을 뜻함. 즉, 현실 세계를 단순화 시켜 표현하는 기법



실체
(Reality)



모델
(Model)

▶ 모델링의 필요성

1. 의사소통 하기 좋다

다양한 형태들의 모델을 가시화 시켜 **고객-개발자, 개발자-개발자 간의 의사소통을 원활히** 하며
요구사항에 부합한 시스템을 개발할 수 있도록 해줌

2. 대규모 프로젝트 구조의 로드맵(길잡이)을 만들 때 유용하다

로드맵을 통해 클래스와 클래스 간에 의존하는 관계 등을 개발자가 빨리 파악 가능

3. 개발할 시스템 구축에 대한 기초를 마련할 수 있다

프로세스 과정 상 분석/설계 후 구현단계로 진행하게 되는데

모델링 단계에서 만들어낸 산출물을 통해 CASE도구에서 소스코드 생성 기능 같은 것들 제공

4. 백엔드 문서용으로 제격이다

프로젝트를 다른 팀에게 넘기거나 이어서 맡는 경우 그 팀에게 유용

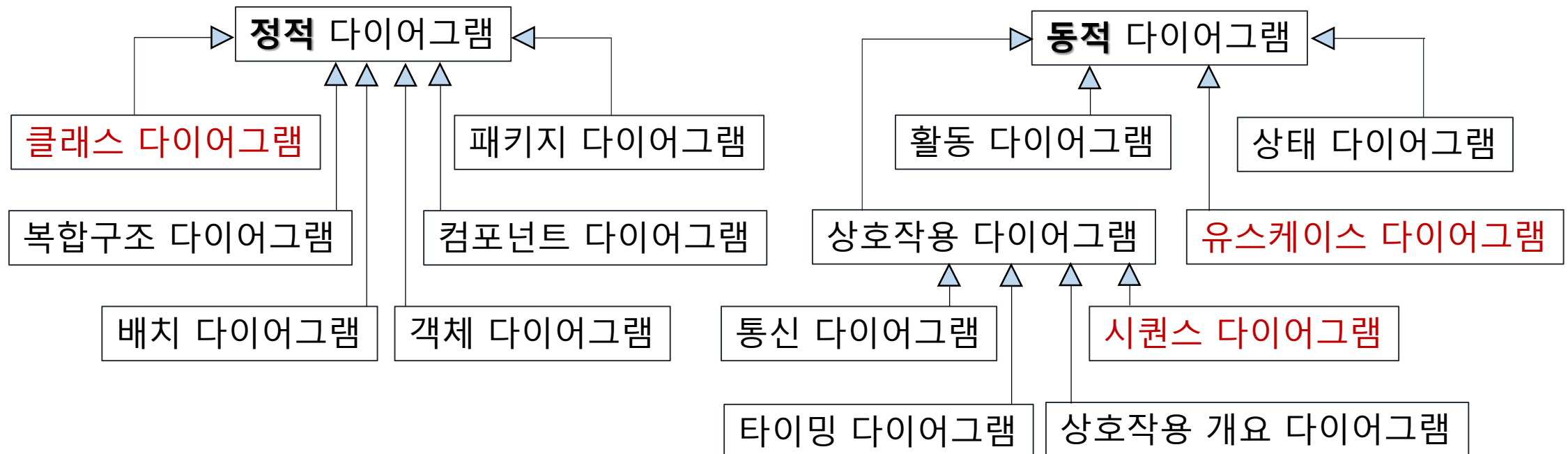
UML을 통해 의사 소통을 활발히 하고 소프트웨어 구조의 재사용 문제 해결

▶ 모델링과 UML

✓ UML

통합 모델링 언어(UML, Unified Modeling Language)는 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어로 소프트웨어 개념을 다이어그램으로 그리기 위해 사용하는 시각적인 표기법

✓ UML 다이어그램 종류

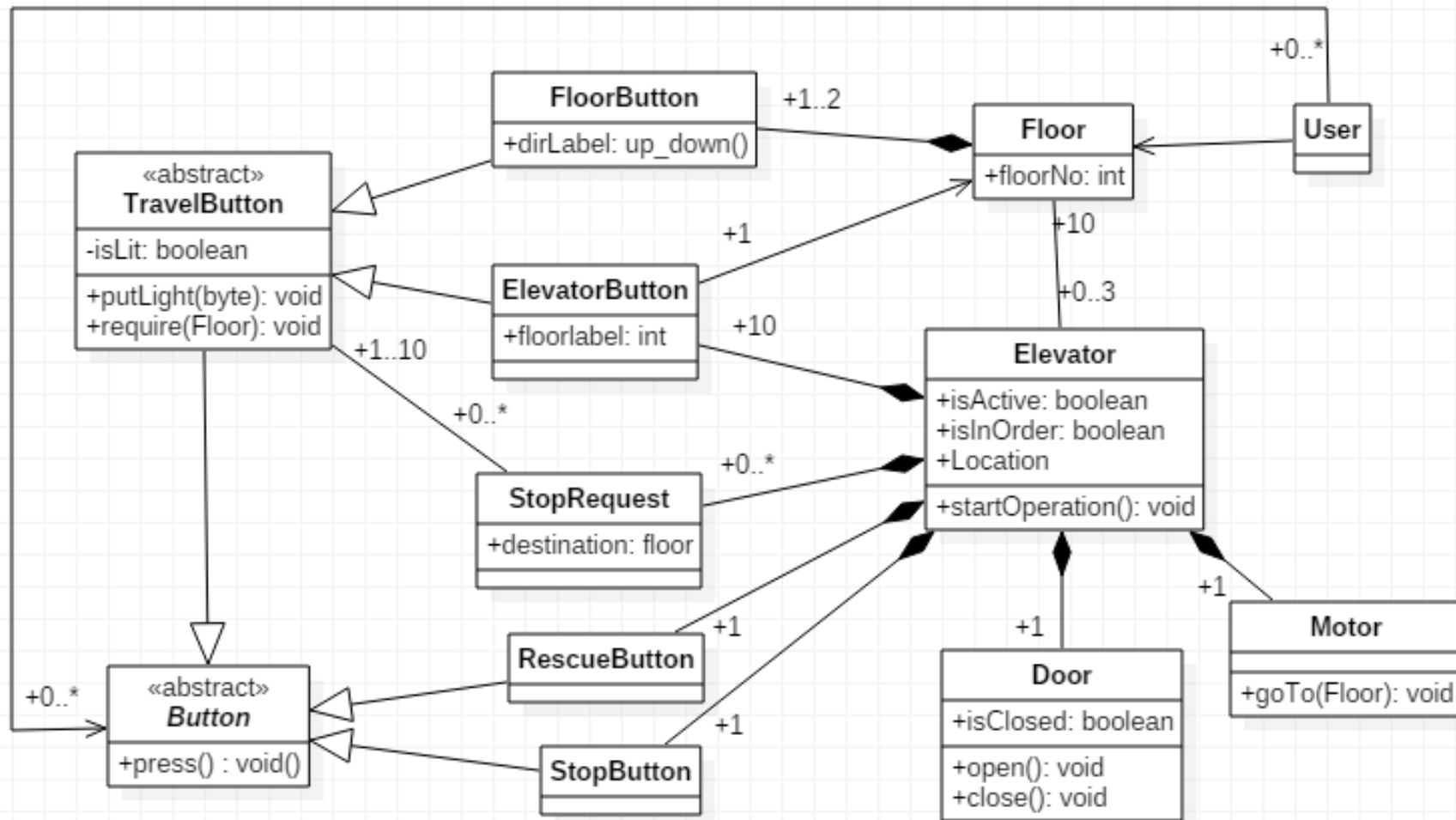


▶ UML

분류	다이어그램 유형		목적
정적	클래스		프로그램 안의 주요 클래스와 주요 관계를 보여줌
	객체		시스템 실행 중 어느 순간의 객체와 관계를 포착해서 보여줌
	복합구조		내부 구조를 표현하는 다이어그램
	배치		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조 표현
	컴포넌트		컴포넌트 사이의 의존관계 묘사. 컴포넌트를 구성하는 요소들과 그것들을 구현하는 요소들도 모두 표현 가능
	패키지		대규모 시스템에서 주요 요소간의 종속성을 나타내거나 여러 클래스들의 그룹화된 매커니즘을 나타낼 때 쓰임
동적	활동		플로우 차트가 uml에 접목된 개념, 여러가지 행위들과 제어구조 등을 모두 포함
	상태		한 객체의 상태 변화를 다이어그램으로 표현한 것
	유스케이스		시스템과 사용자가 상호작용하는 경우를 나타내는 기능 위주의 다이어그램
	상호 작용	시퀀스	시간 흐름에 따른 객체 사이의 상호작용 표현
		상호작용 개요	여러 상호작용 다이어그램 사이의 제어흐름을 표현
		통신	객체 사이의 관계를 중심으로 표현
		타이밍	객체 상태 변화와 시간 제약을 명시적으로 표현

▶ 모델링과 UML

✓ 클래스 다이어그램

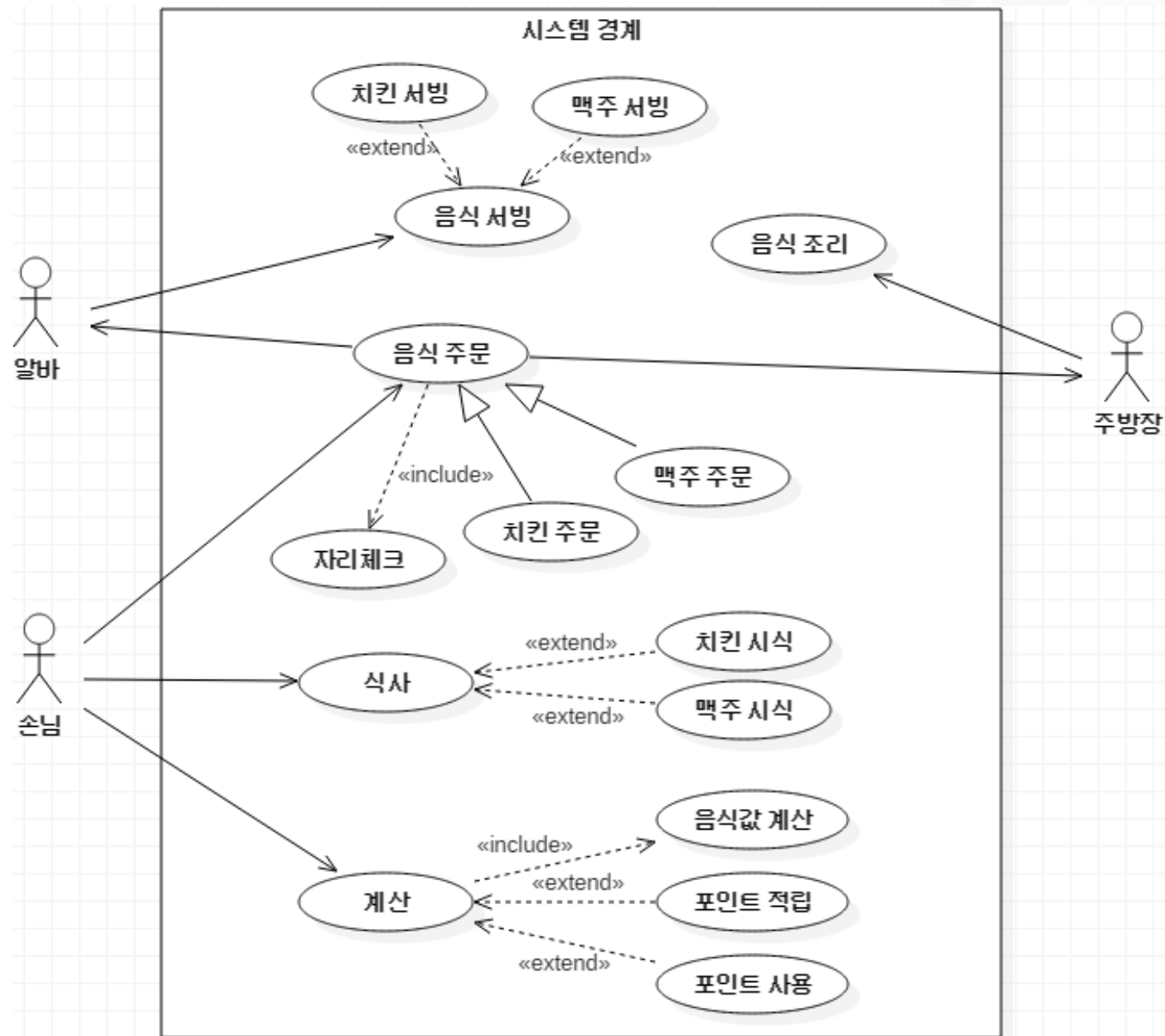


▶ UML

분류	다이어그램 유형		목적
정적	클래스		프로그램 안의 주요 클래스와 주요 관계를 보여줌
	객체		시스템 실행 중 어느 순간의 객체와 관계를 포착해서 보여줌
	복합구조		내부 구조를 표현하는 다이어그램
	배치		소프트웨어, 하드웨어, 네트워크를 포함한 실행 시스템의 물리 구조 표현
	컴포넌트		컴포넌트 사이의 의존관계 묘사. 컴포넌트를 구성하는 요소들과 그것들을 구현하는 요소들도 모두 표현 가능
	패키지		대규모 시스템에서 주요 요소간의 종속성을 나타내거나 여러 클래스들의 그룹화된 매커니즘을 나타낼 때 쓰임
동적	활동		플로우 차트가 uml에 접목된 개념, 여러가지 행위들과 제어구조 등을 모두 포함
	상태		한 객체의 상태 변화를 다이어그램으로 표현한 것
	유스케이스		시스템과 사용자가 상호작용하는 경우를 나타내는 기능 위주의 다이어그램
	상호 작용	시퀀스	시간 흐름에 따른 객체 사이의 상호작용 표현
		상호작용 개요	여러 상호작용 다이어그램 사이의 제어흐름을 표현
		통신	객체 사이의 관계를 중심으로 표현
		타이밍	객체 상태 변화와 시간 제약을 명시적으로 표현

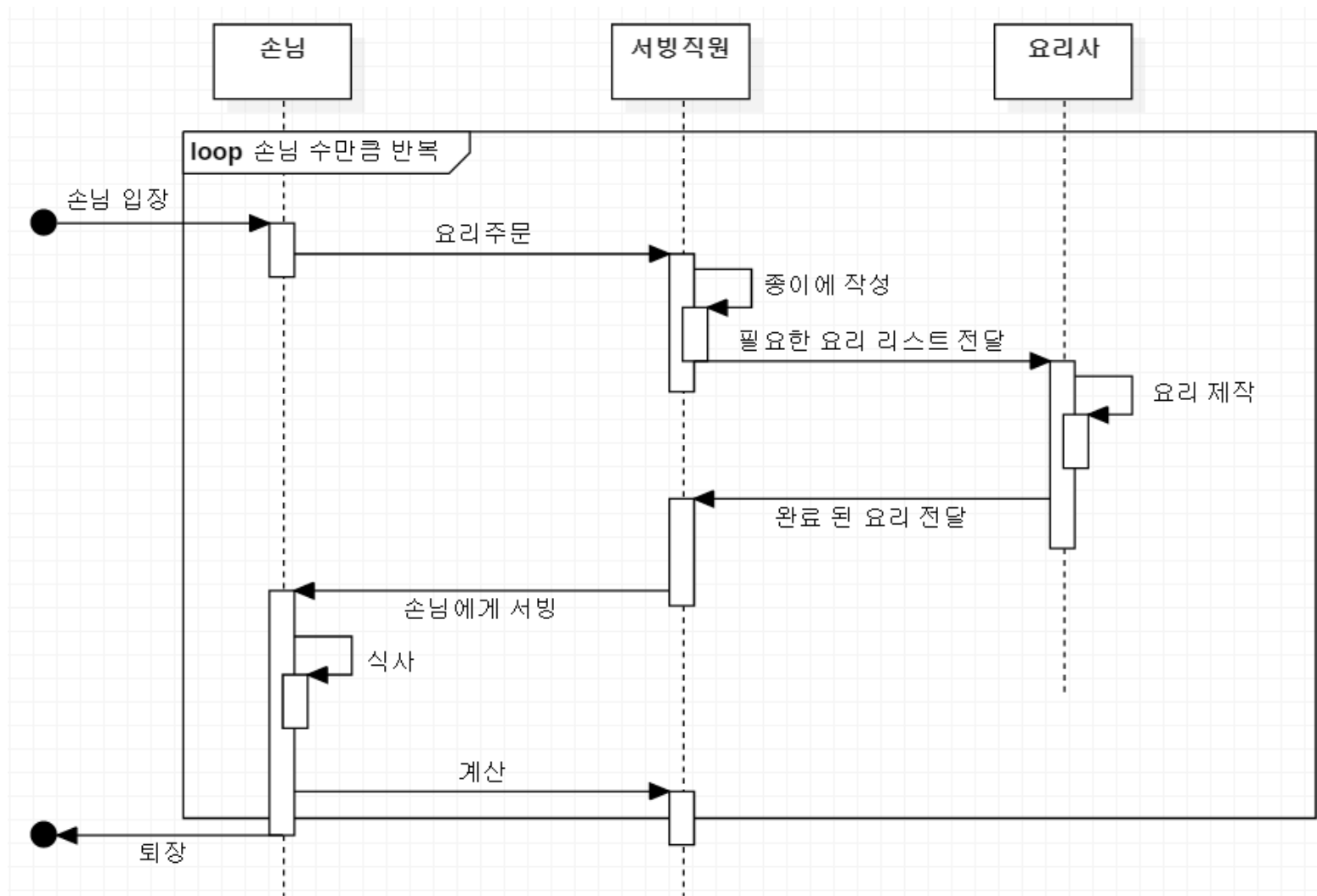
▶ 모델링과 UML

✓ 유스케이스 다이어그램



▶ 모델링과 UML

✓ 시퀀스 다이어그램



▶ 모델링과 UML

✓ 소프트웨어(프로그램) 개발 프로세스



패키지 다이어그램

유스케이스
다이어그램

클래스 다이어그램
객체 다이어그램

상태 다이어그램
활동 다이어그램
상호작용 다이어그램

컴포넌트
복합구조
배치
다이어그램

▶ UML의 V 프로세스

✓ 기능모델링

✓ 동적모델링

✓ 정보모델링

행위자

유스케이스

유스케이스
시나리오

유스케이스
시나리오의
정보

블랙박스 분석

요구사항 명세서

액티비티
다이어그램

시퀀스
다이어그램

클래스
다이어그램

화이트박스 분석



▶ 모델링 시 주의할 점

1. 핵심적인 기능 위주로 작업하라

모든 기능을 상세히 표현하기 위하여 모델링 작업을 진행하면
작성자도 힘들고, 상대방도 알아보기 어려워짐 → **시간 낭비!**

2. 모델링 작업은 개발 프로세스 분석/설계 단계에서만 하는 것이 아니다

반복적으로 모델을 수정하여 **다듬어 나가는 것이 중요**

3. 모델을 보고 개발하려는 기능의 코드를 떠올릴 수 있어야 한다

모델링 된 산출물을 보고 코드가 떠오르지 않으면 모델링 과정에 문제가 있을 가능성이 높음

▶ UML 툴

✓ StarUML 5.0 설치

UML 툴 중에서 가장 보편화 되어 있는 프로그램

1. <https://sourceforge.net/projects/staruml/files/staruml/5.0/>
2. Download Lastet Version 클릭
3. exe 설치
4. Next> 경로 설정하고 Next> ...
5. 사용자 가이드
[http://staruml.sourceforge.net/docs/user-guide\(ko\)/toc.html](http://staruml.sourceforge.net/docs/user-guide(ko)/toc.html)



▶ UML 툴

✓ StarUML 5.0 활용

1. New Project By Approach

Empty Project – Ok

2. 우측 Untitled 우클릭 – Add - Model

(Default로 Model1이라는 Model이 만들어짐)

3. Model1 우클릭 – Add Diagram 후

원하는 다이어그램 선택

4. 좌측에 생기는 기호를 끌어다가 쓰면서 다이어그램 완성

Chap02.

유스케이스 다이아그램

▶ 요구사항

고객 및 소프트웨어 개발에 관계된 사람들이 시스템 개발에 앞서 개발되는 프로그램에 필요한 조건이나 능력을 말함

✓ 요구사항 프로세스



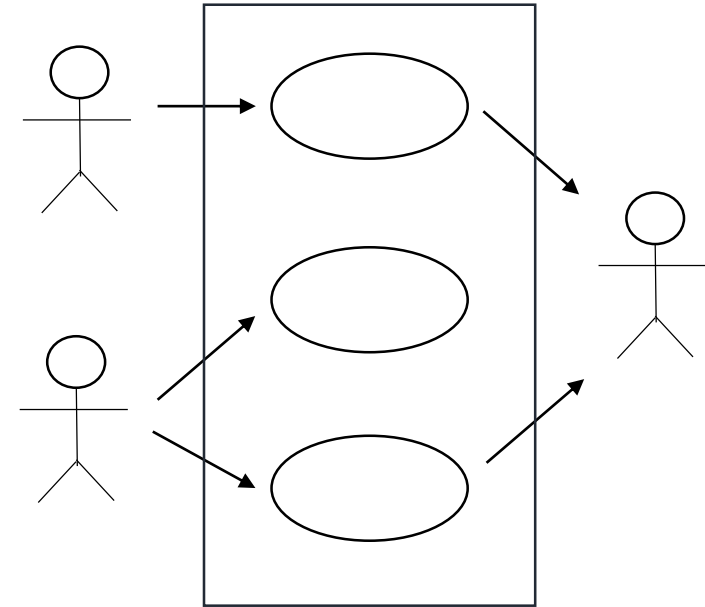
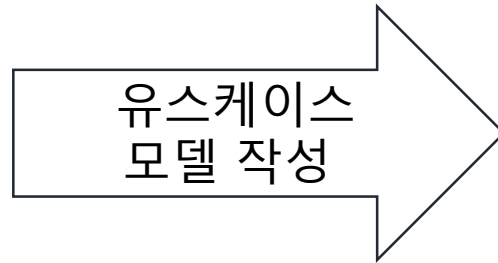
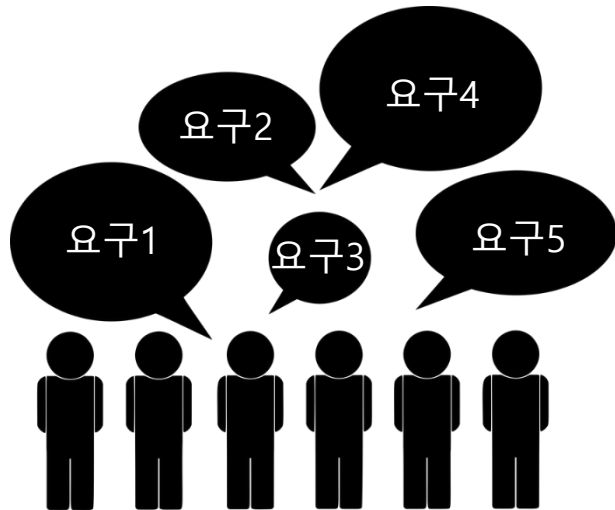
✓ 요구사항 조건

- 명확성 : 기술된 요구사항은 항상 동일한 의미로 해석되어야 함 → 모호하지 않아야 함
- 완전성 : 사용자가 기대하는 모든 요구사항이 기술되어야 함 → 누락되어서는 안 됨
- 일관성 : 서로 상충되는 요구사항이 있어서는 안 됨
- 검증 가능성 : 객관적으로 검증할 수 있도록 구체적이어야 함

▶ 유스케이스 다이어그램

동적(행위) 다이어그램으로 사용자 관점의 시스템 동작 요구사항 표현

여러 업무 프로세스를 설명하는데 자주 활용

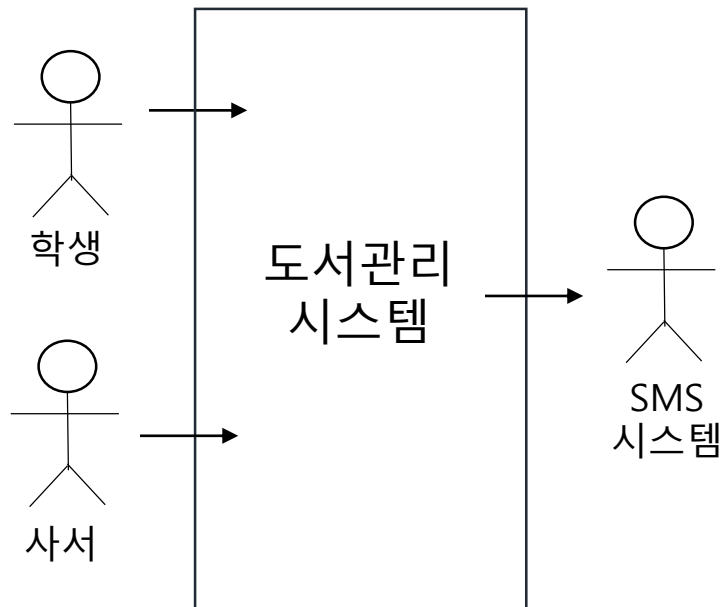


▶ 유스케이스 다이어그램

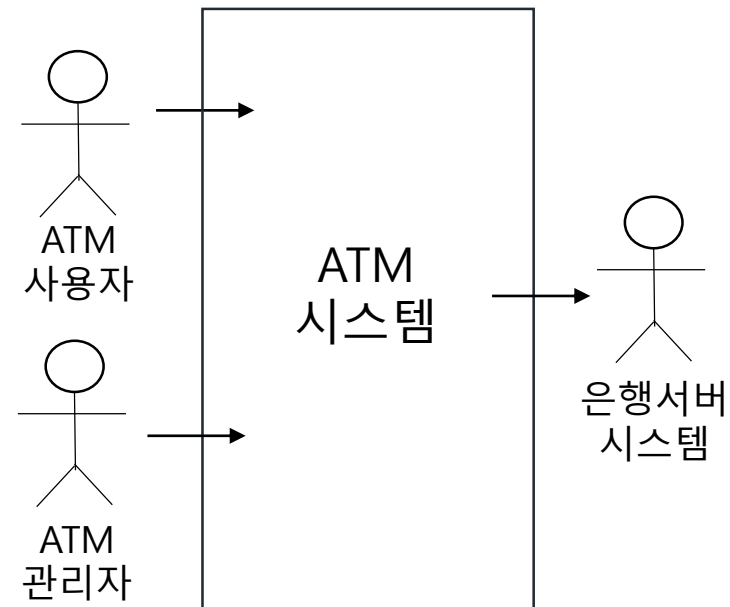
✓ 액터

시스템과 상호작용을 하는 **시스템 외부의 존재**로 개발 대상에 따라 달라질 수 있음
시스템 관점에서 바라본 사용자, 타 시스템의 역할을 뜻해야 함

ex) 도서관리 시스템의 액터



ex) ATM 시스템의 액터

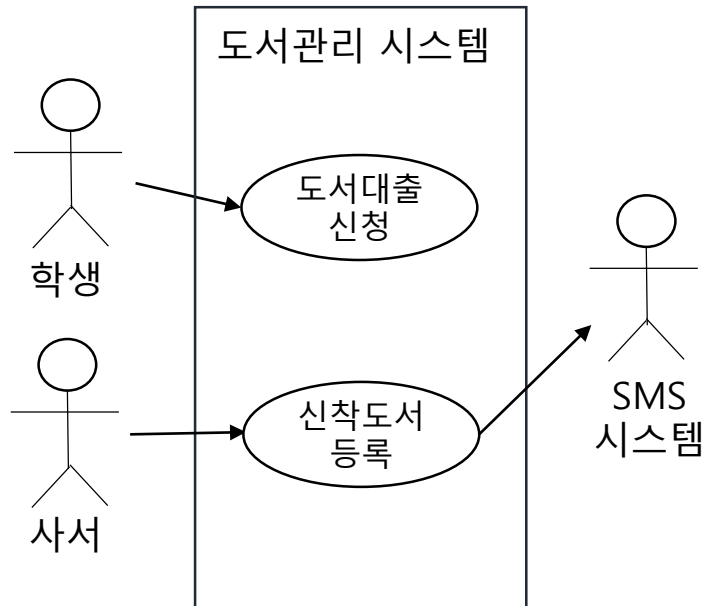


▶ 유스케이스 다이어그램

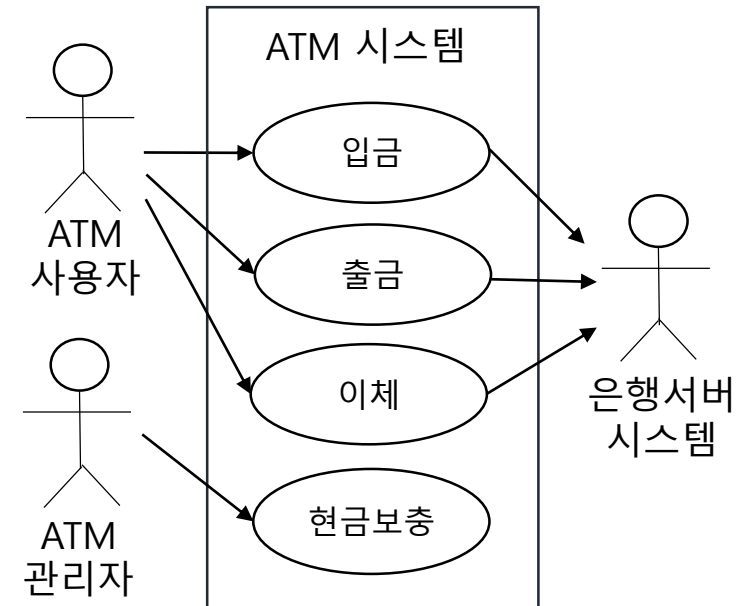
✓ 유스케이스

개발 대상이 되는 시스템이 제공하는 **개별적인 기능**
사용자가 인지할 수 있는(눈에 보이는) 하나의 시스템 기능 단위

ex) 도서관리 시스템의 유스케이스

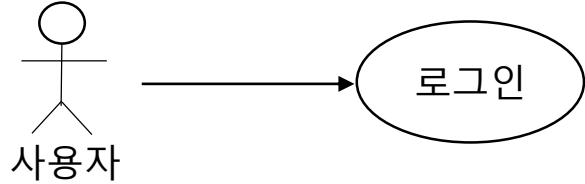
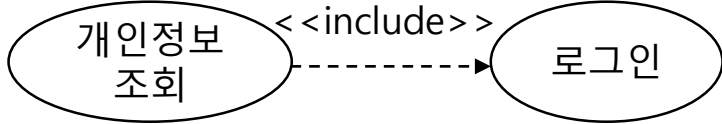
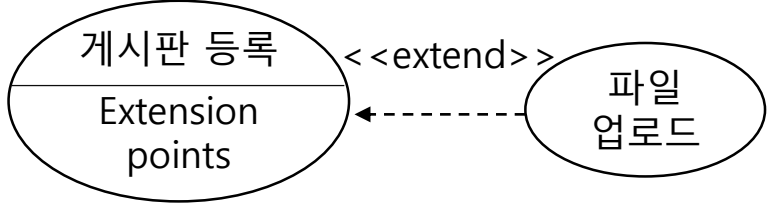
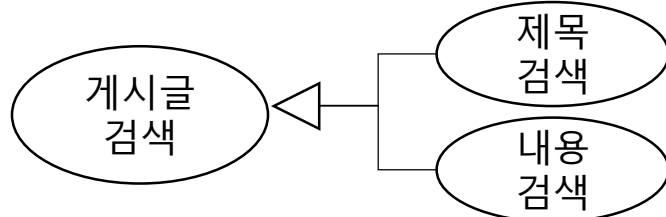


ex) ATM 시스템의 유스케이스



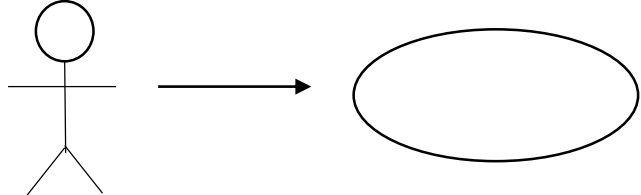
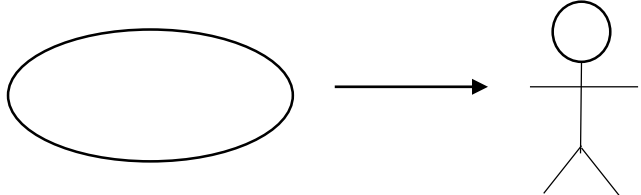
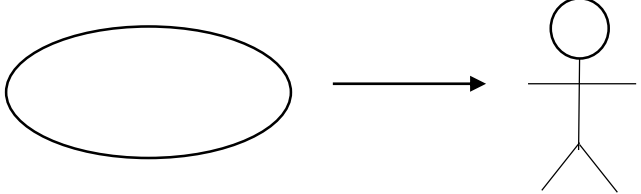
▶ 유스케이스 다이어그램

✓ 관계 종류

유형	설명	관계 방향
연관 관계	유스케이스와 액터 간 상호작용을 의미하는 관계	
포함 관계	한 유스케이스가 다른 유스케이스의 기능을 포함하는 관계 (반드시 해야만 하는 관계)	
확장 관계	기본 유스케이스에서 특정 조건이나 액터의 선택에 따라 발생하는 유스케이스 (선택적 으로 할 수 있는 관계)	
일반화 관계	유사한 유스케이스들 또는 액터들을 추상화 한 하나의 유스케이스로 그룹핑 하여 이해도를 높인 관계	

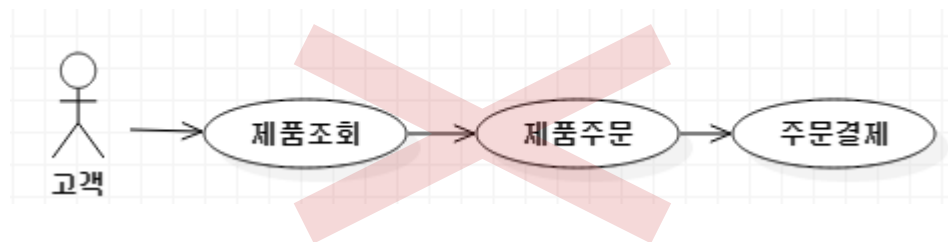
▶ 유스케이스 다이어그램

✓ 액터와 유스케이스 간의 연관 관계 방향

유형	설명	연관 관계 방향
활성화	액터가 유스케이스를 활성화 시킴	
수행결과 통보	유스케이스 결과가 액터에게 통보 됨	
외부 서비스 요청	외부 시스템에 서비스 실행을 요청함	

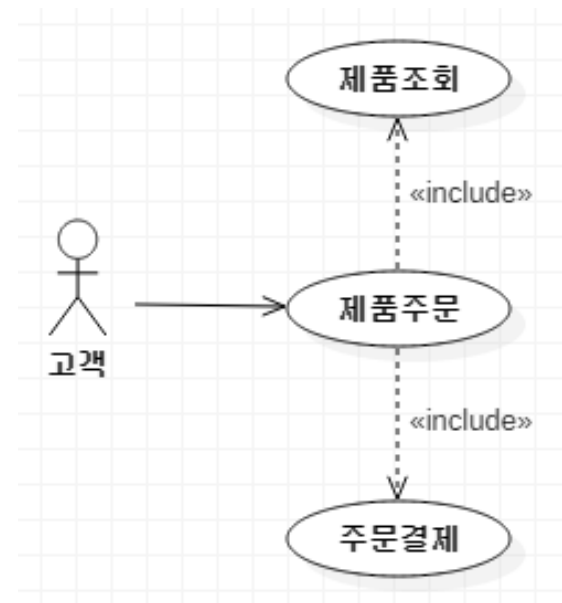
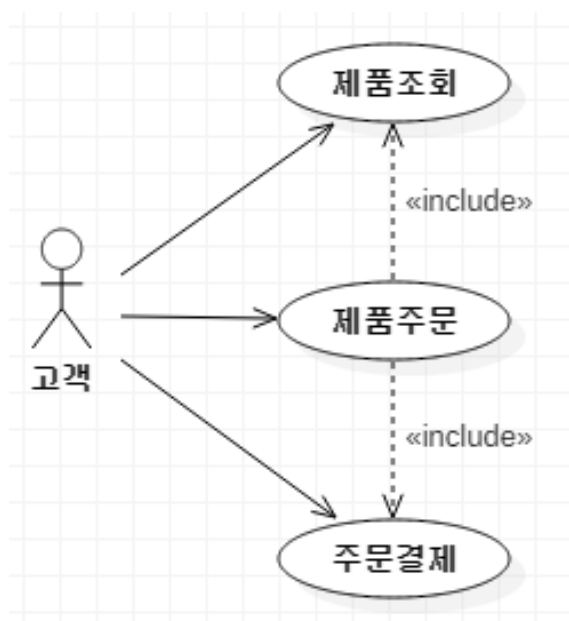
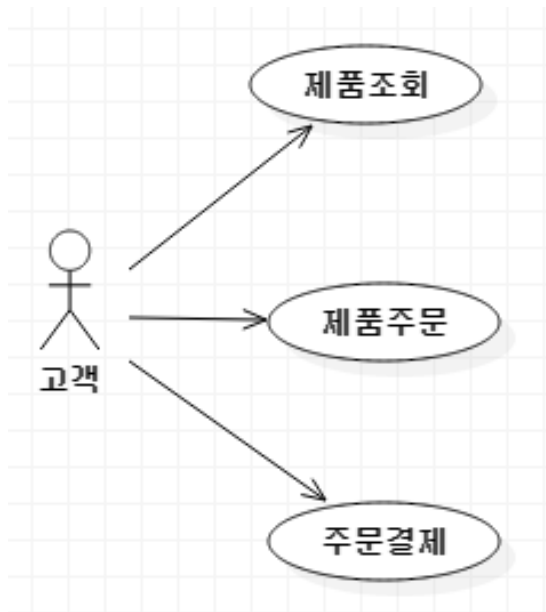
▶ 유스케이스 다이어그램 상황 별 예시

✓ 시나리오상 다음과 같은 흐름 인식



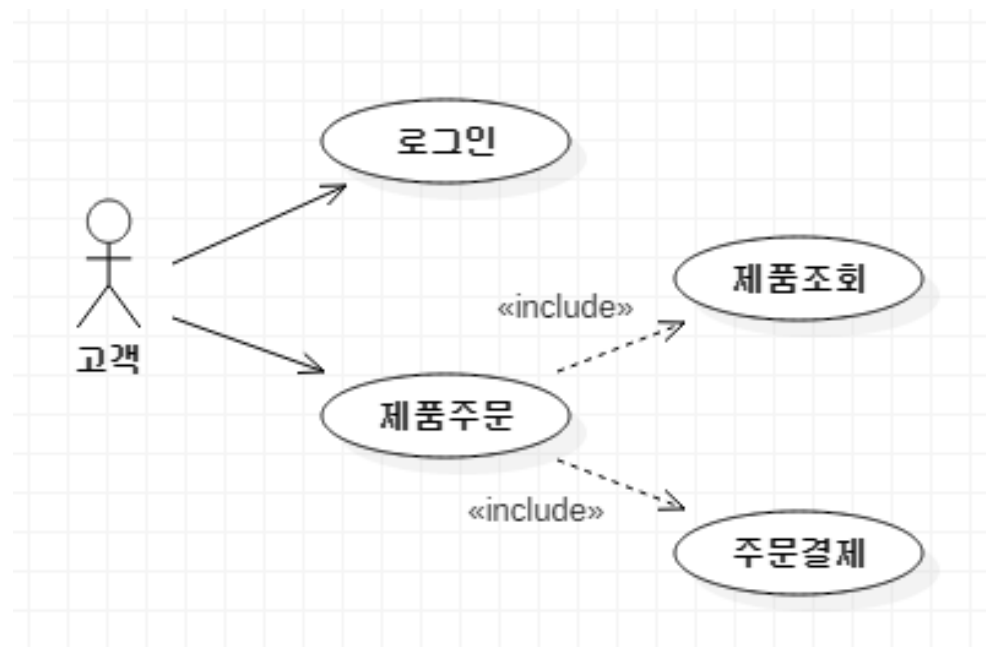
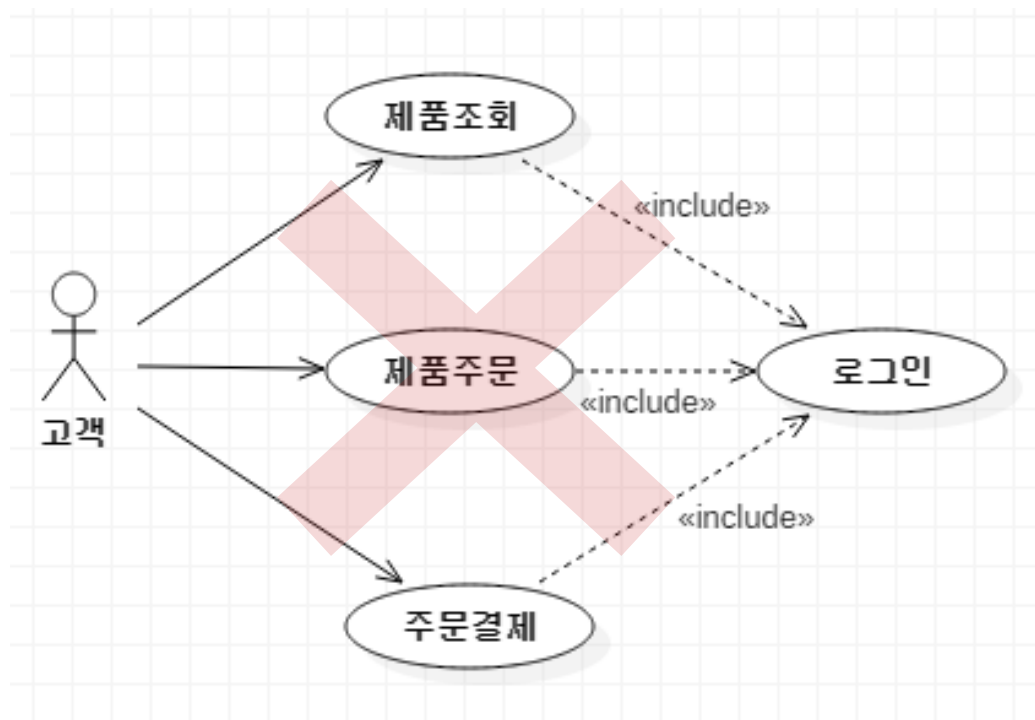
연관관계(실선)는 액터와 유스케이스 간만 사용 가능

✓ 의미에 따른 모델링



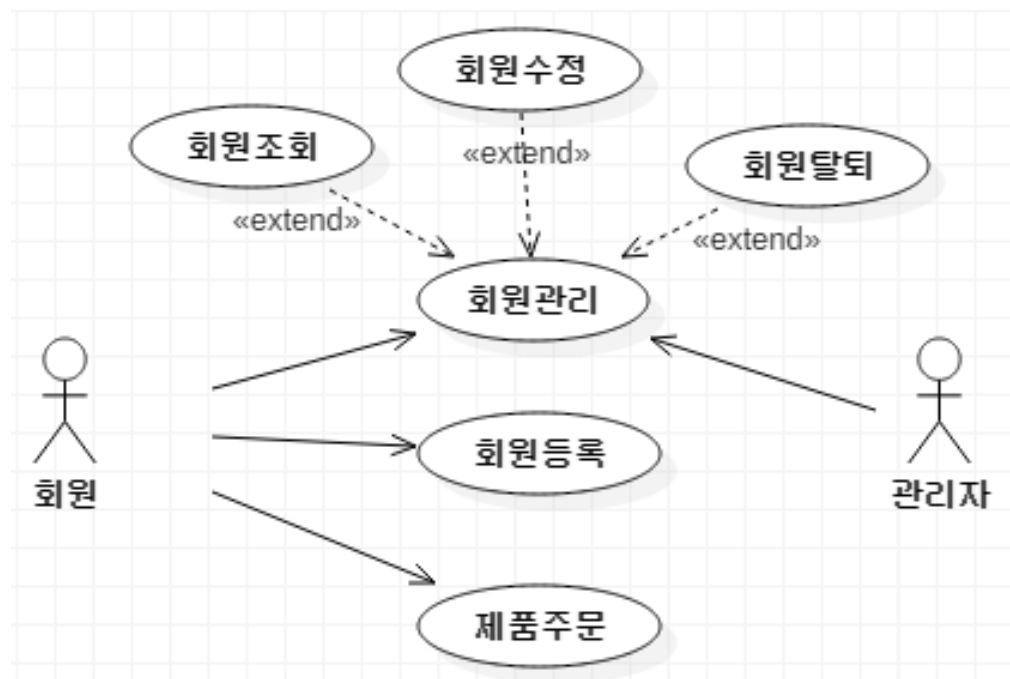
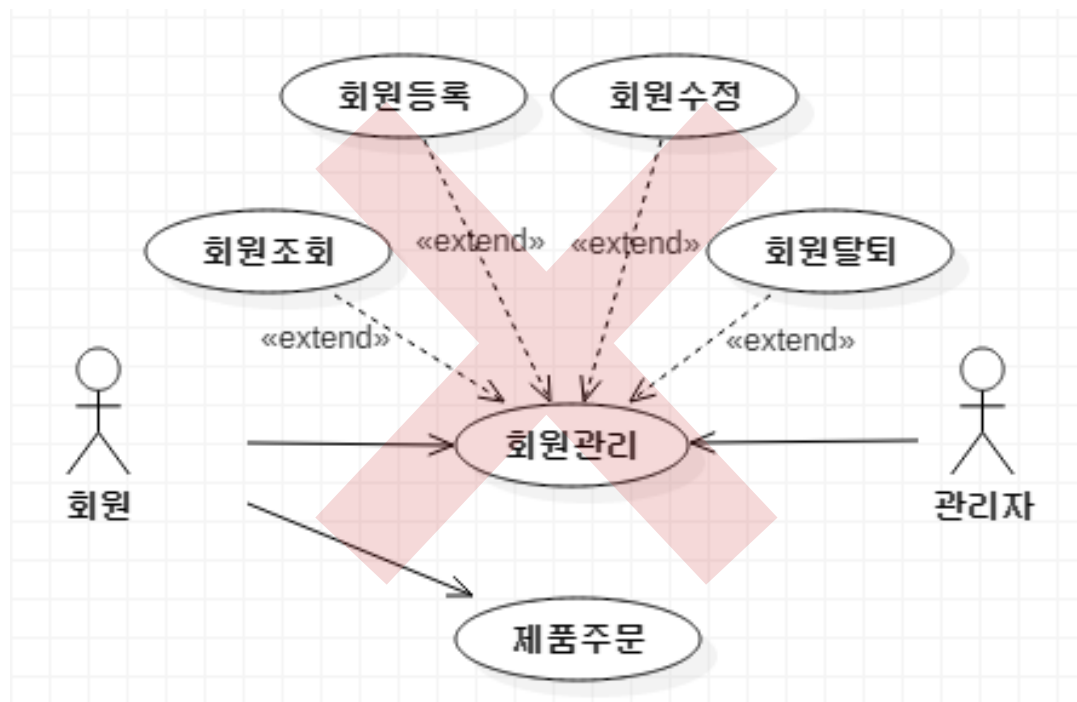
▶ 유스케이스 다이어그램 상황 별 예시

✓ 로그인 유스케이스



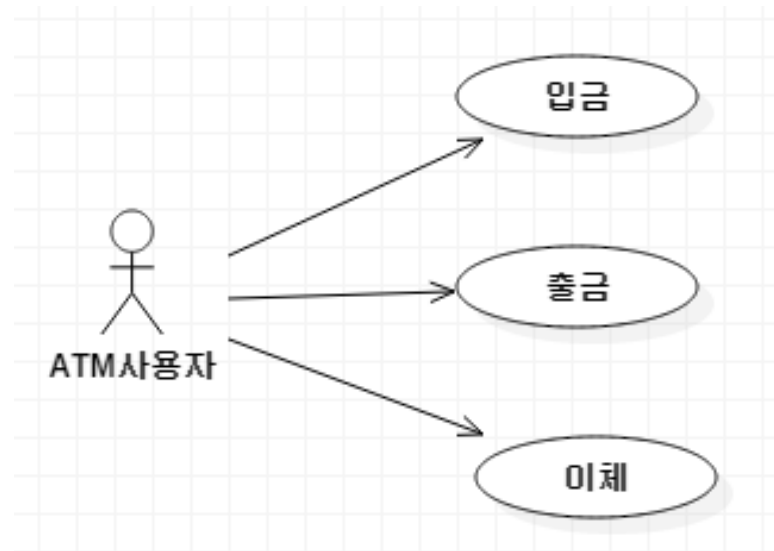
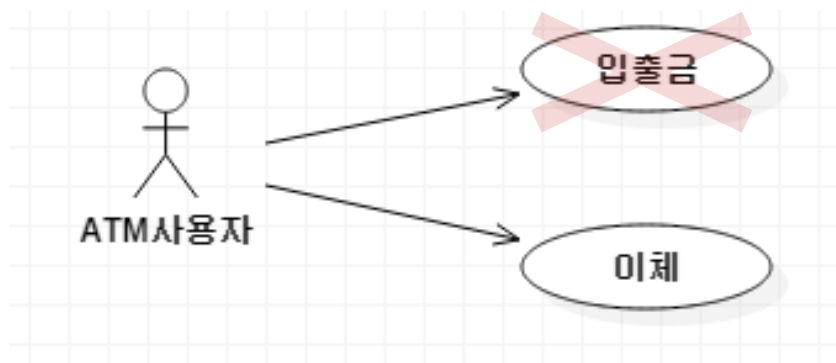
▶ 유스케이스 다이어그램 상황 별 예시

✓ 유스케이스의 동일한 기능 제공

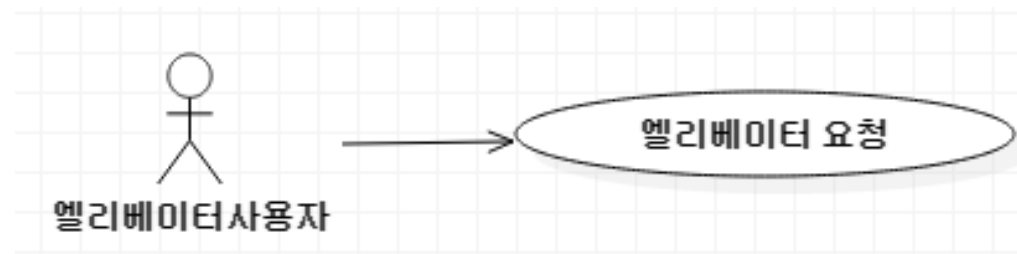
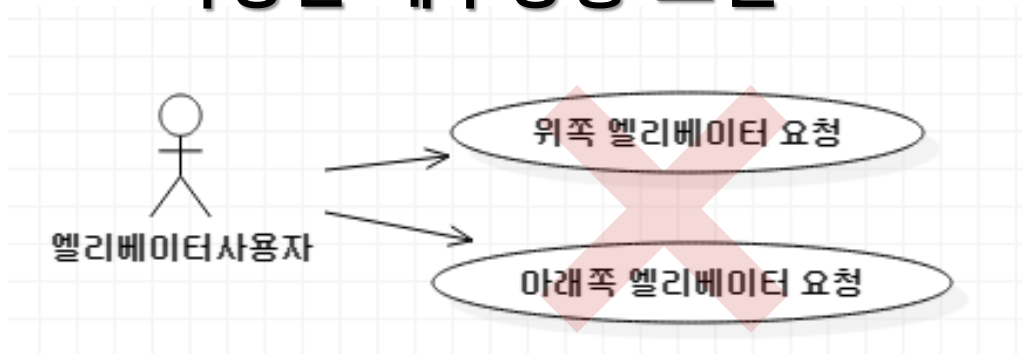


▶ 유스케이스 다이어그램 상황 별 예시

✓ 유스케이스의 구체화



✓ 다양한 세부상황 표현



▶ 유스케이스 다이어그램 수업 실습

✓ 유스케이스 다이어그램 작업 과정

1. 요구사항 기술서 → 액터, 유스케이스 추출
2. 추출한 액터, 유스케이스 무작위 단순 배치 (UML 툴 사용)
3. 중복 의미의 유스케이스 제거 및 불필요한 유스케이스 제거, 그룹핑 작업
4. 액터, 유스케이스 간 관계 설정

중고나라는 아나바다의 정신을 온라인으로 이은 대표적인 사이트이다.

일반 쇼핑몰처럼 관리자가 개입하여 판매나 구매가 이루어지는 것이 아니라 사용자끼리 게시판을 통해 각자의 물건을 사거나 팔 수 있다.

중고나라 사용자는 게시글을 등록해 상품을 판매할 수 있다. 게시글 등록 시 반드시 카테고리 선택을 해야 하며 사진도 첨부할 수 있다.

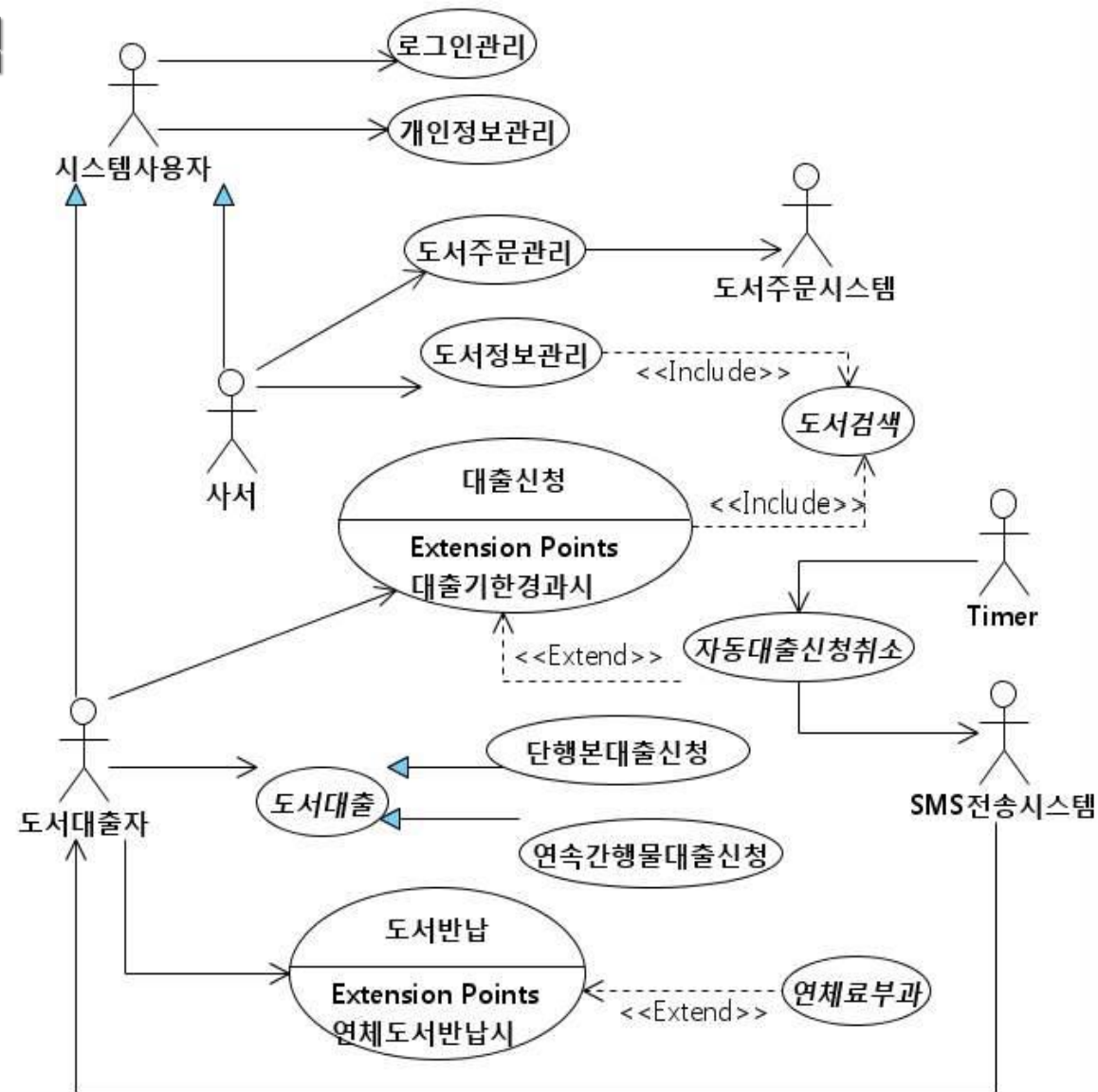
또한 사용자는 본인이 올린 게시글과 다른 사용자가 올린 게시글의 목록을 조회할 수 있고 게시글 목록 조회는 반드시 카테고리를 선택하여야 조회 가능하다. 게시글 목록 조회를 통해 게시글 상세 조회가 가능하다. 게시글 상세 조회 시 댓글을 달 수 있고 상품도 구매할 수 있다.

사용자는 다른 사용자가 올린 게시글이나 댓글을 통해서 해당 사용자를 신고할 수 있고 신고 내용은 관리자에게 넘어간다.

사용자는 본인이 올린 게시글에 한해 수정과 삭제를 할 수 있는데 본 게시글의 비밀번호를 인증했을 시 가능하다.

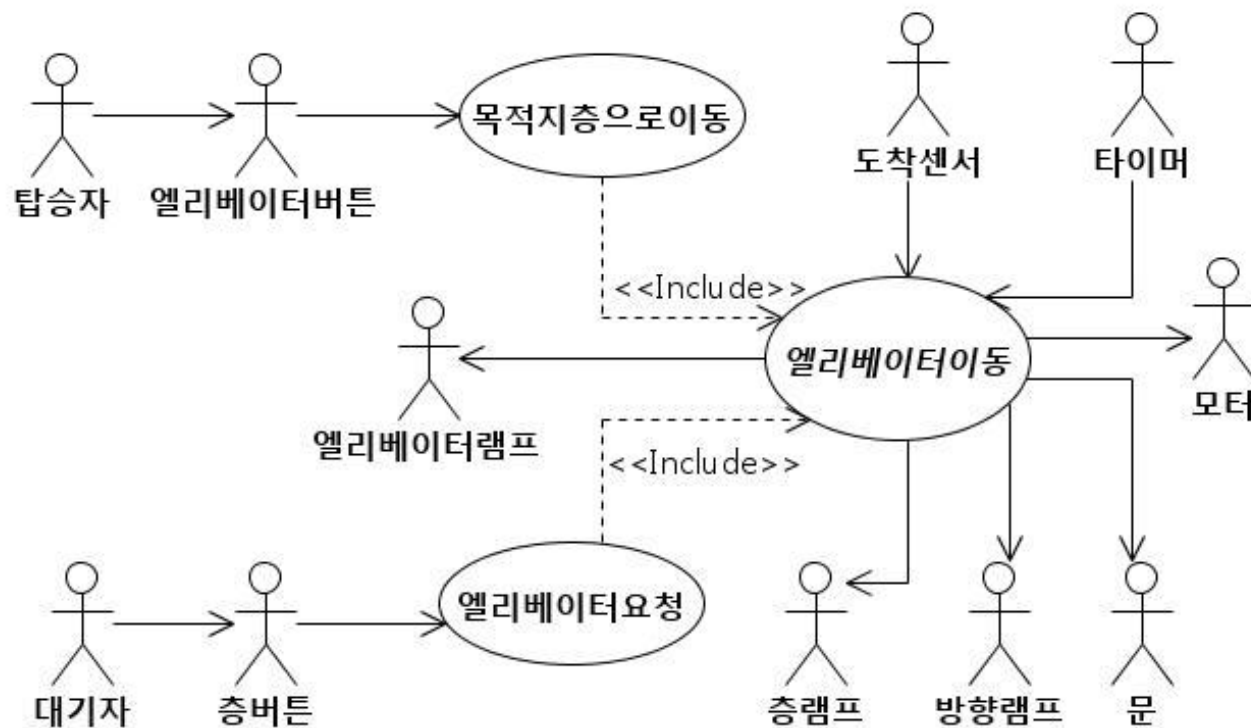
▶ 유스케이스 다이어그램 예시

✓ 도서관리 시스템



▶ 유스케이스 다이어그램 예시

✓ 엘리베이터 시스템



▶ 이벤트 흐름

✓ 기본 흐름

아무것도 잘 못 되지 않았다는 가정 하에 사용자의 자극에 시스템이 어떻게 반응하는지 기술

✓ 대안 흐름

세부 상황 중 일부가 일이 잘 못 되었을 경우를 고려한 흐름으로 선택흐름과 예외흐름이 존재

- 선택흐름 : 사용자 혹은 시스템에 의해 선택적으로 수행되는 흐름
- 예외흐름 : 시스템에서 발생하는 에러 등을 처리하기 위해 수행되는 흐름

▶ 유스케이스 시나리오

✓ 예시

항목	설명		
개요	마이크는 주변 소리를 SOS시스템에 전달한다. SOS시스템은 입력받은 소리 중 분석 대상인 음성과 음향을 선별한다. 선별된 정보는 음성인식기로 보내 의미 분석을 요청한다.		
관련 액터	주 액터	마이크	
	보조 액터	음성인식기	
우선 순위	개발의 우선 순위	중요도	상
		난이도	상
선행 조건	마이크와 음성인식기는 사용 가능한 상태이어야 한다.		
후행 조건	인식이 완료된 소리는 파일로 추출되어 음성인식기로 보내야 한다.		
시나리오	기본 시나리오	1. 마이크는 주변 <u>소리</u> 를 iPhone 앱에 전송한다. 2. iPhone 앱은 전송 받은 소리를 판별한다. 3. iPhone 앱은 판별 결과 <u>의미가 있는 음성</u> 인 경우 음성인식기로 전송하여 의미 분석을 요청한다.	
	대안 시나리오	A1 : 소리 판별 불가 시 (2)에서 일치 유형이 없는 경우 1. 음성인 경우 : iPhone 앱은 'key word' 검색 과정을 반복한다. 2. 음향인 경우 : iPhone 앱은 결과를 저장하지 않는다.	
비기능적 요구사항	가용성: 소리 인식 기능은 언제나 활성화가 가능해야 한다.		

1. 개요

회원은 장바구니에 상품을 담은 후 주문한다. 장바구니에 담긴 상품들의 목록 중에서 주문하고자 하는 상품을 선택한 후 주문서를 작성한다.

2. Relationships

- Initiator : 회원
- Supporters :
- Pre-condition : 장바구니에 담긴 상품의 목록을 조회한다.
- Post-condition : 선택한 상품에 대한 주문 정보가 저장된다.

3. 기본 흐름

- ① 장바구니에 담긴 상품목록에서 주문하고자 하는 상품들을 선택하고, '주문하기' 기능을 실행한다.
- ② 회원이 장바구니에서 선택한 각 상품의 목록(상품명, 가격, 수량, 금액, 가격) 수량과 주문 총액을 보여주며, 주문정보(수령인 이름, 배송지주소, 배송지연락처)를 입력받기 위한 화면이 나타난다.
- ③ 주문정보를 입력하고 '저장' 기능을 실행한다.
- ④ 이때 주문번호는 시스템에 의해 자동으로 생성되며, 주문일자는 시스템 날짜로 저장된다.
- ⑤ 주문정보에 대한 주문결제 화면을 보여준다(JC-MCM 주문결제)
- ⑥ 주문이 이루어지면 주문된 상품들은 장바구니에서 삭제된다.

4. 대안 흐름

- A1. 주문 정보 입력 시 '취소' 기능을 요청한 경우
주문서 작성을 취소하고 장바구니목록 화면을 출력한다.

5. 예외 흐름

- E1. 장바구니에서 상품을 선택하지 않고 주문하기 기능을 요청한 경우
선택한 상품이 없다는 메시지를 출력한다.
- E2. 입력되지 않은 주문정보(수령인 이름, 배송지주소, 배송지연락처)가 있는 상태로 '저장' 기능을 요청한 경우
입력하지 않은 주문정보가 있다는 메시지를 출력하고 해당 필드로 커서를 이동한다.

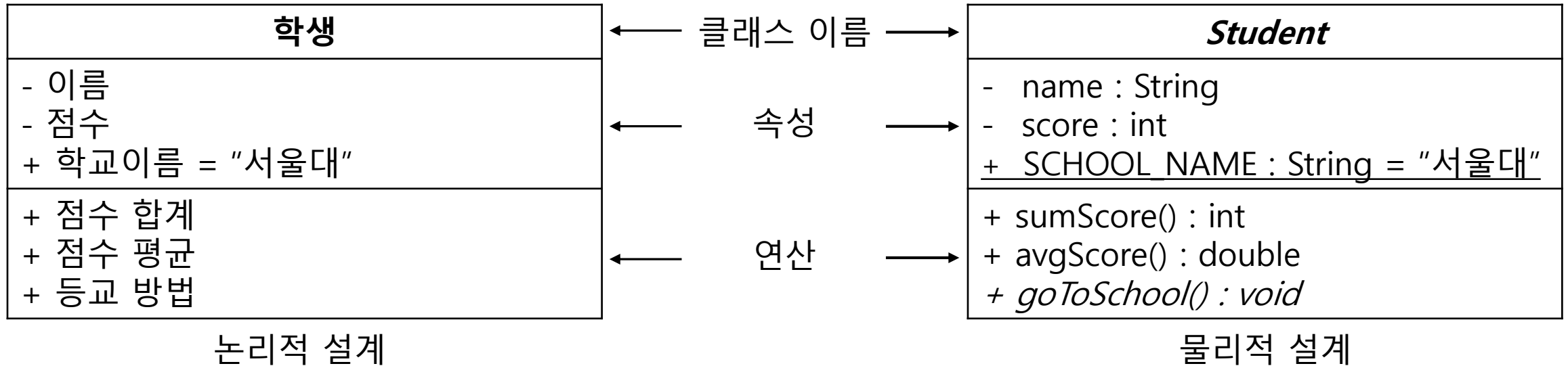
Chap03.

클래스 다이어그램

▶ 클래스 다이어그램

정적(구조) 다이어그램으로 UML모델링에서 가장 일반적으로 사용
시스템의 구조와 구조 간 상호 관계를 나타내며
시스템의 논리적 및 물리적 구성요소 설계 시 주로 활용

✓ 클래스의 표현



▶ 클래스 다이어그램


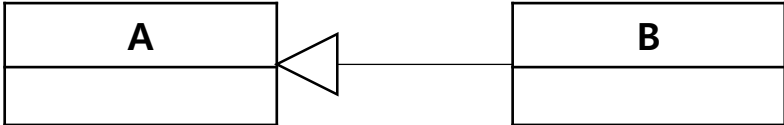
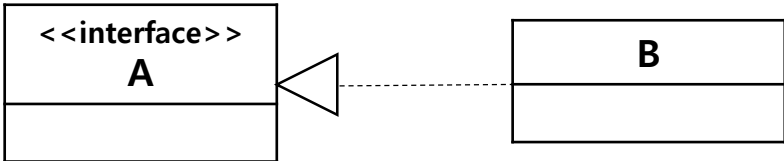

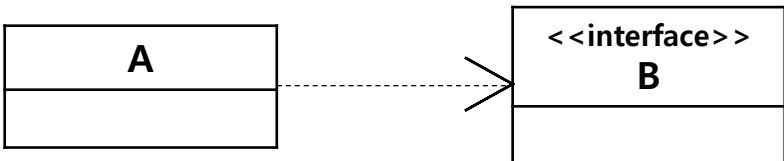
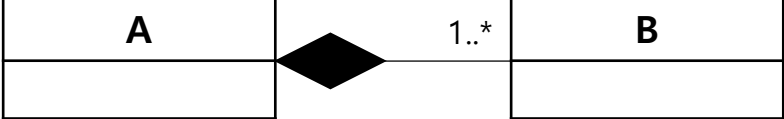
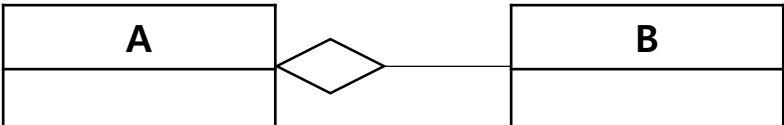
✓ 접근제한자

기호	예약어	적용 범위
+	public	전체
#	protected	같은 패키지 + 상속 관계
~	default	같은 패키지
-	private	같은 클래스

✓ 클래스 다이어그램 표기 방법

표현법	적용 범위	예약어
<u>attribute</u> / <u>method</u> (밑줄)	속성, 연산	static
FIELD (대문자)	속성	final
<i>Class</i> / <i>method</i> (기울임)	클래스 명, 연산	abstract

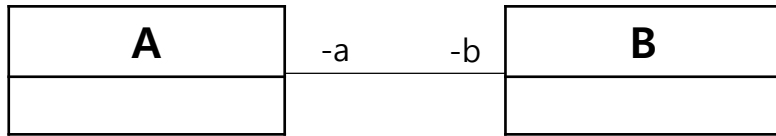
▶ 클래스 다이어그램의 관계

관계	표기법	의미
연관 관계		클래스 A와 클래스 B는 연결되어 있다.
일반화 관계		클래스 B는 클래스 A의 하위 클래스이다.
실체화 관계 (인터페이스 실현 관계)		클래스 B는 인터페이스 A를 실현한다.
의존 관계		클래스 A는 클래스 B에 의존한다.
인터페이스 의존 관계		클래스 A는 인터페이스 B에 의존한다.
합성관계		클래스 A는 클래스 B를 한 개 이상 포함하고 있다.
집합 관계		클래스 B는 클래스 A의 부분이다.

▶ 연관 관계

한 클래스가 **필드**로 다른 클래스를 참조할 때를 의미.

클래스 간의 관련성을 뜻하는 것으로 메시지 전달의 통로 역할을 함

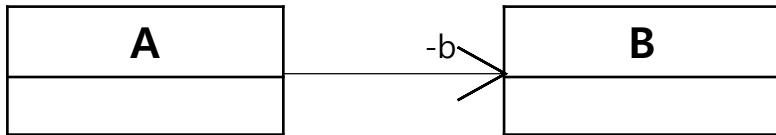


```
public class A{
    private B b;
}
```

```
public class B{
    private A a;
}
```

✓ 방향성이 있는 연관 관계

방향성은 메시지 전달의 방향을 뜻하며 반대 방향은 불가능



```
public class A{
    private B b;
}
```

```
public class B{
    private A a;
}
```

▶ 연관 관계

✓ 연관 관계의 다중성

관계를 맺을 수 있는 실제 상대 객체의 수를 다중성을 통하여 지정 가능
동일한 의미/역할의 **복수 개의 객체**와의 관계

A	-a	-b	B
	1	0..*	

```
public class A{  
    private Collection<B> b;  
}
```

```
public class B{  
    private A a;  
}
```

✓ 다중 연관

동일한 클래스 간의 존재하는 **복수 개의 연관 관계**를 뜻 함
다른 의미/역할의 복수 개의 객체와의 관계

A	-a1	-b1	B
	1	1..*	
	-a2	-b2	
	1..*	0..*	

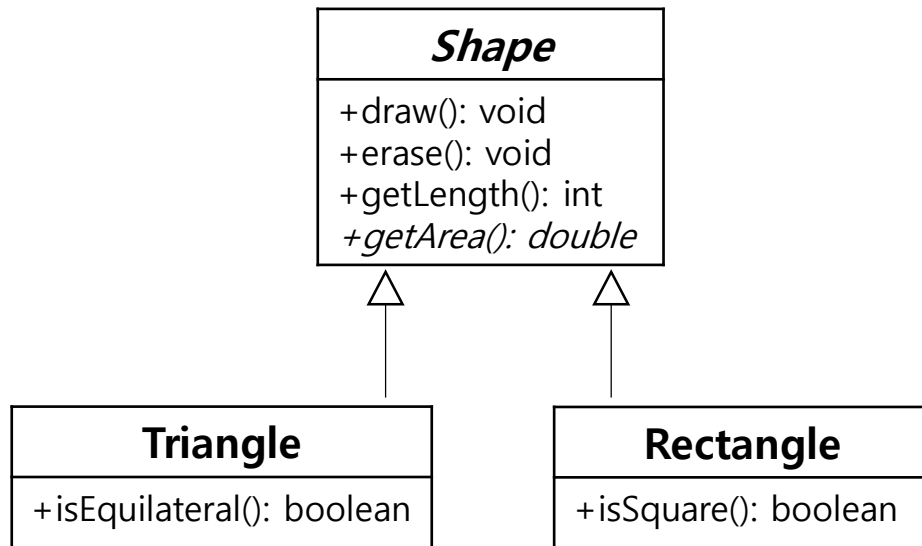
```
public class A{  
    private Collection<B> b1;  
    private Collection<B> b2;  
}
```

```
public class B{  
    private A a1;  
    private Collection<A> a2;  
}
```

▶ 일반화 관계와 실체화(인터페이스 실현) 관계

✓ 일반화 관계

보다 일반적인 클래스와 보다 구체적인 클래스 간의 관계를 뜻하는 것으로
한 클래스(상위 클래스)가 다른 클래스(하위 클래스)보다 일반적인 개념/대상 임을 의미하는 관계



```
public abstract class Shape {
    public void draw() {...}
    public void erase() {...}
    public int getLength() {...}
    public abstract double getArea();
}
```

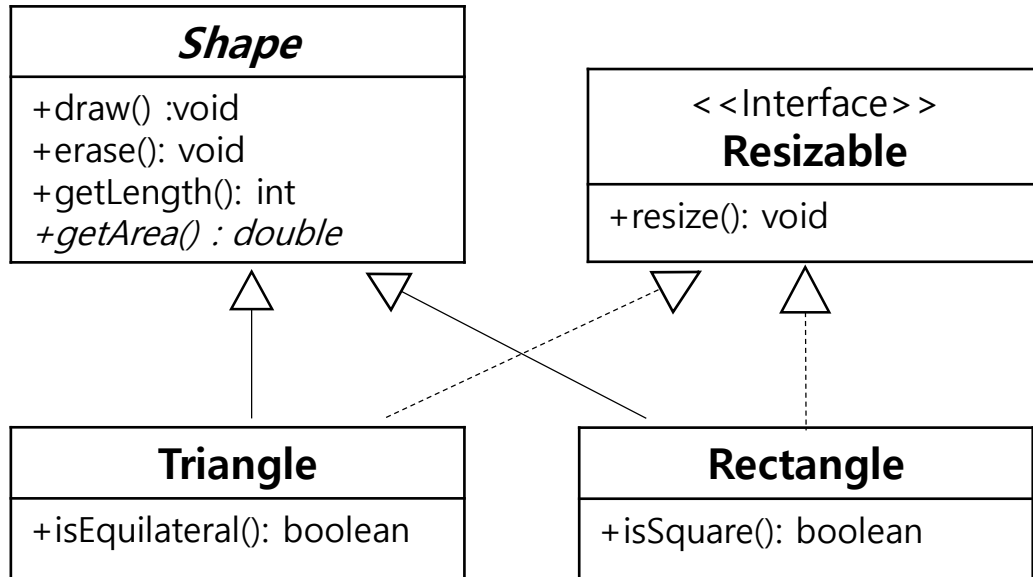
```
public class Triangle extends Shape {
    public boolean isEquilateral() {...}
    public double getArea() {...}
}
```

```
public class Rectangle extends Shape {
    public boolean isSquare() {...}
    public double getArea() {...}
}
```


▶ 일반화 관계와 실체화(인터페이스 실현) 관계

✓ 실체화(인터페이스 실현) 관계

인터페이스에 명시된 기능을 클래스에 의해서 구현한 관계 의미



```
public interface Resizable {
    void resize();
}

public class Triangle extends Shape
    implements Resizable {
    public boolean isEquilateral() {...}
    public double getArea() {...}
    public void resize() {...}
}

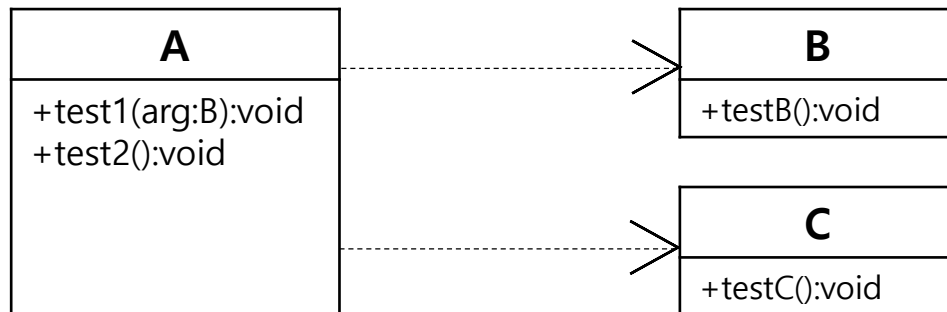
public class Rectangle extends Shape
    implements Resizable {
    public boolean isSquare() {...}
    public double getArea() {...}
    public void resize() {...}
}
```

▶ 의존 관계와 인터페이스 의존 관계

✓ 의존 관계

두 클래스의 **연산 간의 호출 관계**를 표현한 것으로 제공자의 변경이 이용자에 영향을 미칠 수 있음을 의미(제공자의 변경이 이용자의 변경 유발)

이용자는 의존 관계를 통해서 제공자의 연산을 호출할 수 있음

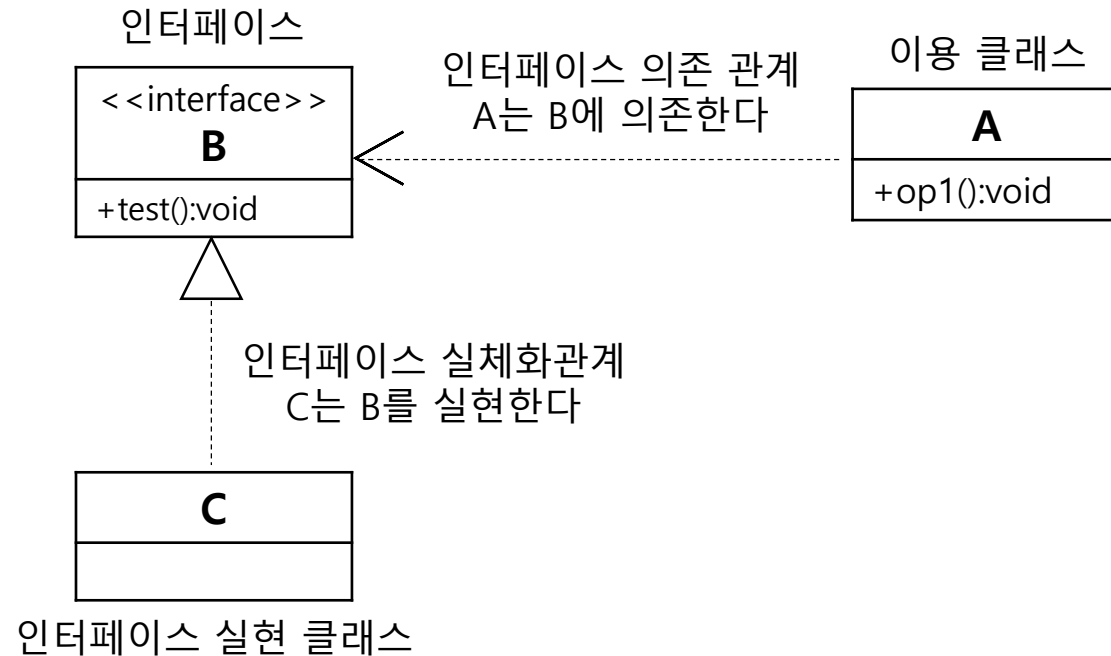


```
public class A{
    public void test1(B arg){
        arg.testB();
    }
    public void test2(){
        C = new C ();
        c.testC();
    }
}
```

▶ 의존 관계와 인터페이스 의존 관계

✓ 인터페이스 의존 관계

인터페이스와 인터페이스 이용자 간의 이용관계를 표현할 때 사용 될 수 있음



```
public interface B{
    void test();
}

public class C implements B{
    public void test() {...}
}

public class A{
    public void op1(){
        B = new C();
        b.test();
    }
}
```

▶ 연관 관계와 의존 관계

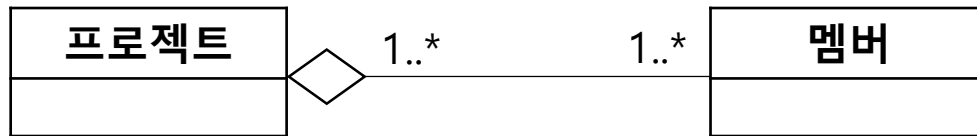
	연관 관계	의존 관계
역할	메시지 전달의 통로	
표현식	class A —→ class B	class A→ class B
관계의 발생 형태	A 클래스의 필드부 에 B 클래스 참조	A 클래스의 메소드 매개변수 또는 메소드 내부 에 B 클래스 참조
관계의 지속 범위	A 클래스의 생명주기	참조된 A 클래스 메소드의 생명주기
방향성	양방향 가능	단방향

▶ 집합 관계와 합성 관계

두 대상 간의 포함(소속) 표현으로 항상 Has-a 의미가 성립되어야 함

✓ 집합 관계

부분 객체가 다수의 전체 객체에 의해 공유 될 수 있음
→ 전체 객체가 사라져도 부분 객체는 존재한다.



프로젝트는 멤버로 구성된다
멤버는 프로젝트의 부분이다
멤버는 다른 프로젝트에도 공유된다

✓ 합성 관계

부분 객체가 오직 하나의 전체 객체에 포함될 수 있음
→ 전체 객체가 사라지면 부분 객체도 사라진다.



집은 방으로 구성된다
방은 집의 부분이다

▶ MVC패턴

디자인 패턴 중 하나

디자인 패턴은 역할을 분리시켜 유지보수성, 애플리케이션 확장성, 유연성에 대한 문제를 해결시켜 주는 장점이 있음

MVC패턴은 클라이언트가 보는 페이지, 데이터 처리, 그리고 이 사이를 중간에서 제어하는 컨트롤러, 이렇게 역할을 분담해놓게 되면 하나의 애플리케이션을 만들 때 디자인 패턴 자체가 갖는 장점과 더불어 각각 맡은 바에만 집중 가능

▶ MVC패턴 구조

✓ Model

데이터 처리 부분 담당

VO(Value Object) : 계층 간 데이터 교환을 위한 객체

DAO(Data Access Object) : DB에 접근하기 위한 용도

Service : DB에서 전달받은 데이터 혹은 DB에 전달할 데이터들을 가공처리(비즈니스 로직)

✓ View

전달 받은 데이터들을 클라이언트에게 전송하여 보여줌

✓ Controller

모델과 뷰를 연결하는 역할로 Service(로직) 처리 후 전달 받은 데이터를 뷰로 전송하거나 뷰에서 사용자가 입력한 값을 전달 받음

Chap04.

시퀀스 다이어그램

▶ 시퀀스 다이어그램

동적(행위) 다이어그램으로 상호작용 다이어그램의 일부분

시스템 내부에서 동작하는 객체들 사이의 주고 받는 메시지를 시간 순서를 강조하여 표현

✓ 생명선과 메시지

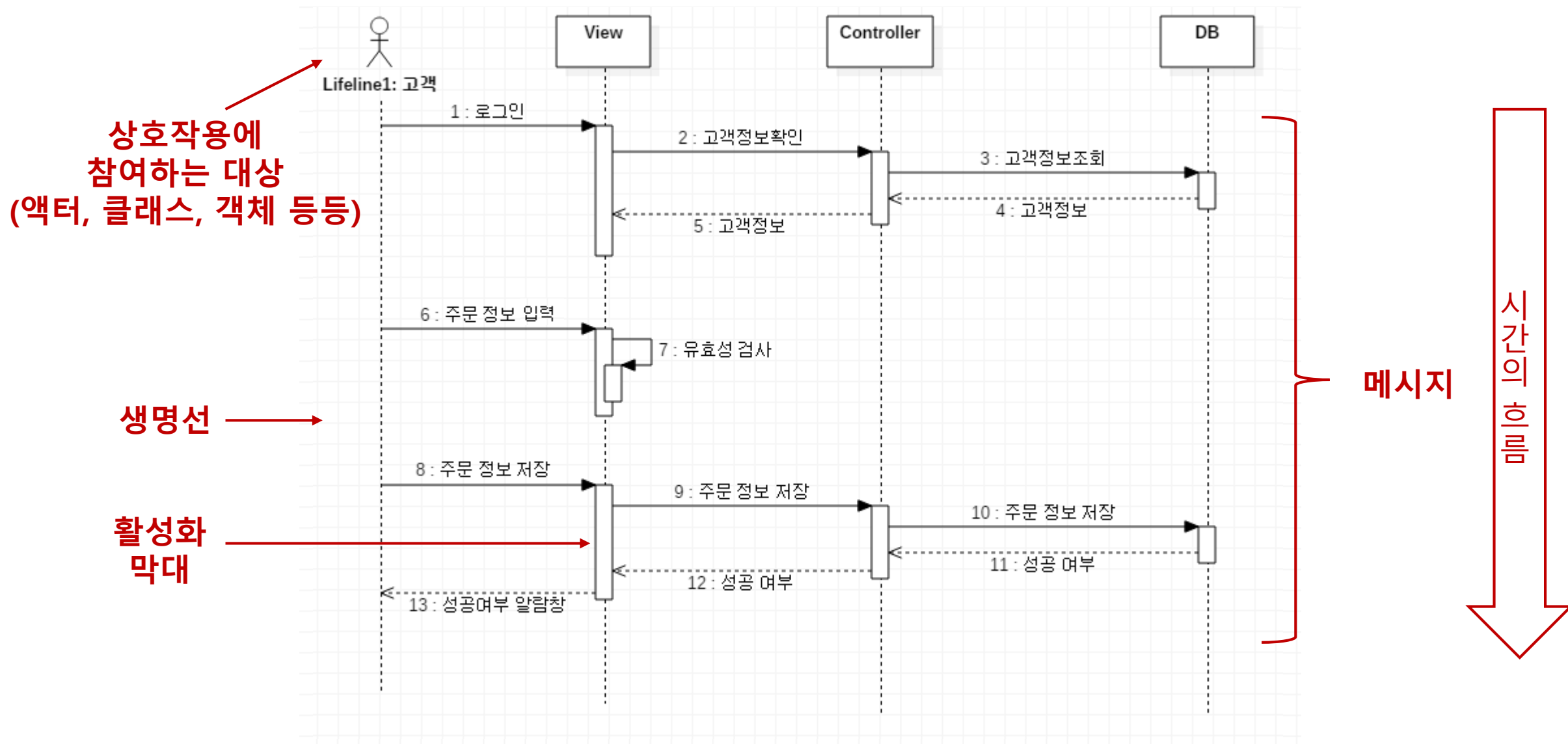
생명선 : 액터, 클래스 객체, 컴포넌트의 인스턴스 등, 상호작용에 참여하는 구체적인 대상 표현

생명선 끝에 X자로 끊겨 있으면 소멸

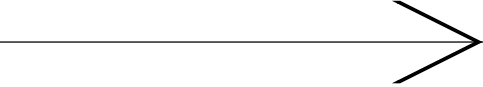

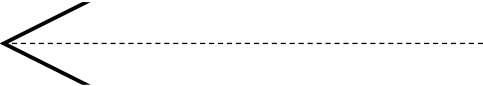
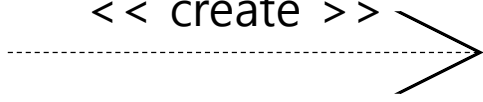

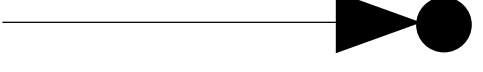
메시지 : 생명선 간에 전달되어 상태의 행위에 대한 호출

* 생명선의 소멸은 자바에서의 가비지 컬렉터에 넘기는 행동이 됨

시퀀스 다이어그램



▶ 시퀀스 다이어그램 메시지 종류

메시지 유형	표현법	설명
비동기적 메시지		송신자를 대기 시키지 않음
동기적 메시지		송신자를 대기시킴
대답 메시지		동기적 메시지의 수행 결과
생성 메시지		생명선 생성
발견된 메시지		모르는 송신자로부터의 메시지
유실된 메시지		모르는 수신자로부터의 메시지

시퀀스 다이어그램 제어흐름

Interaction Operator
(상호작용 연산자)

alt

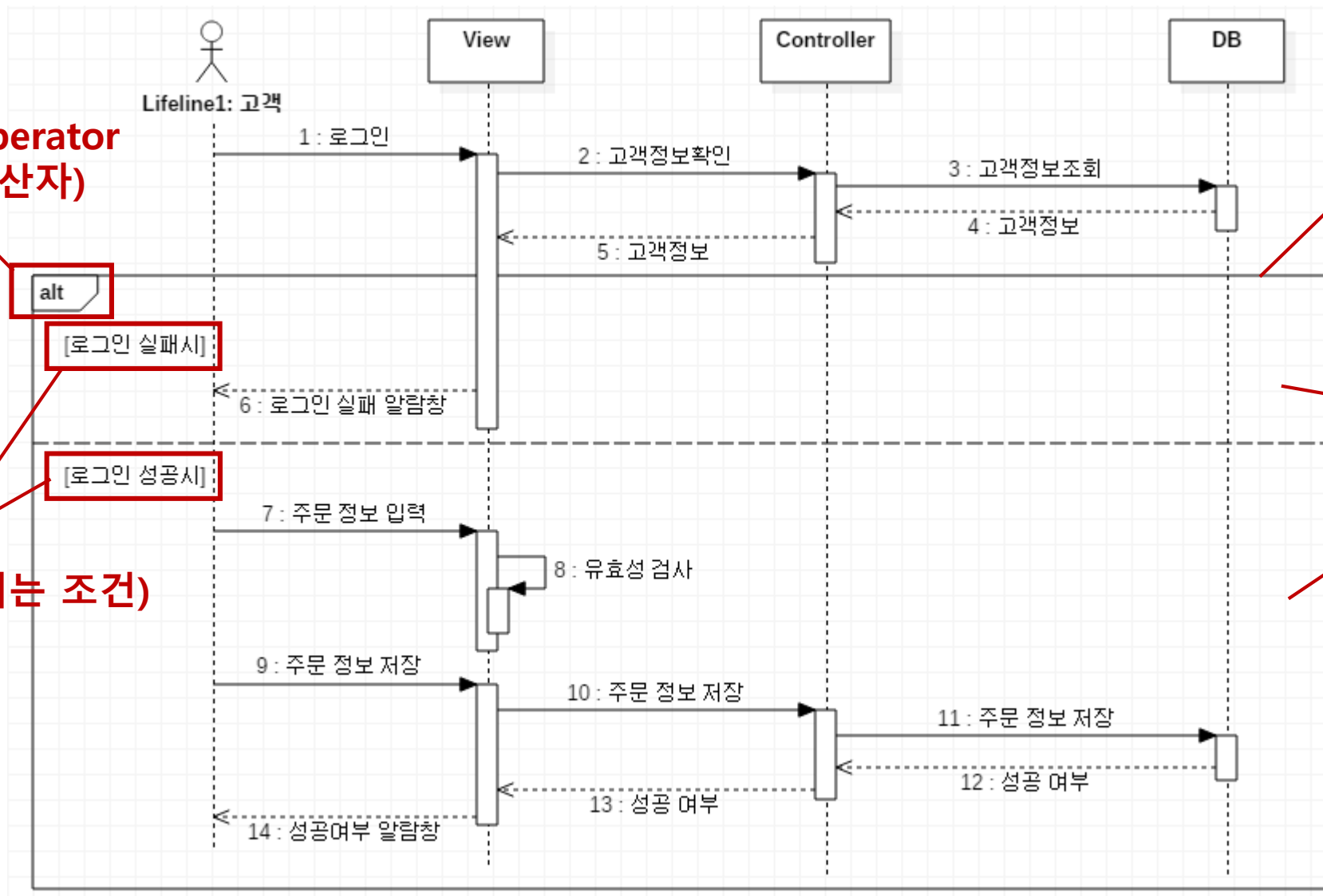
[로그인 실패시]

[로그인 성공시]

Guard
(메시지가 수행되는 조건)

Combined Fragment
(결합 조각)

Fragment
(조각)



▶ 시퀀스 다이어그램 상호작용 연산자 종류

연산자 종류		설명
alt	대체	메시지의 대체 시퀀스를 포함하는 연산자이며 어떤 상황에서도 하나의 시퀀스만 발생함 → if - else if 문에 해당하는 논리를 나타냄. 단, 모든 가드가 false이고 else가드가 존재하지 않으면 어떠한 조각도 실행되지 않음
opt	옵션	선택적 요소로 발생하거나 발생하지 않을 수 있는 시퀀스를 포함 → if 문에 해당하는 논리를 나타냄. 대안을 하나만 제공해야 하고 조건이 false일 경우 조각이 실행 되지 않음
loop	반복	루프 상호작용 연산자는 반복적으로 실행되는 부분을 나타냄 가드 안에 단편이 실행되는 횟수를 지정할 수 있음
break	중단	보통 loop 연산자와 함께 쓰이며 중단 상호작용 연산자는 기타 프로그래밍 언어의 중단 매커니즘과 유사함 조건이 true 일 경우 현재 실행을 포기하고 빠져나감

* 이 외에 critical, ignore, strict, seq 등이 존재한다.