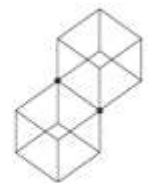


4. 디자인패턴



JAVA 개체 지향 디자인 패턴

UML과 GoF 디자인 패턴 핵심 10가지로 배우는



학습목표

학습목표

- 디자인 패턴을 만든 동기 이해하기
- 합동과 디자인 패턴 관계 이해하기
- 디자인 패턴 분류하기

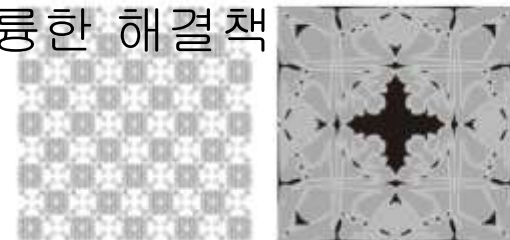
4.1 디자인패턴의 이해

❖ 크리스토퍼 알렉산더

- Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.
- 바퀴를 다시 발명하지 마라 (Dont reinvent the wheel)
- 패턴은 비슷하거나 동일한 양식 또는 유형들이 반복되어 나타난다는 의미이며, 문제와 해결책도 동일한 유형이나 양식을 통해 쉽게 찾을 수 있다.

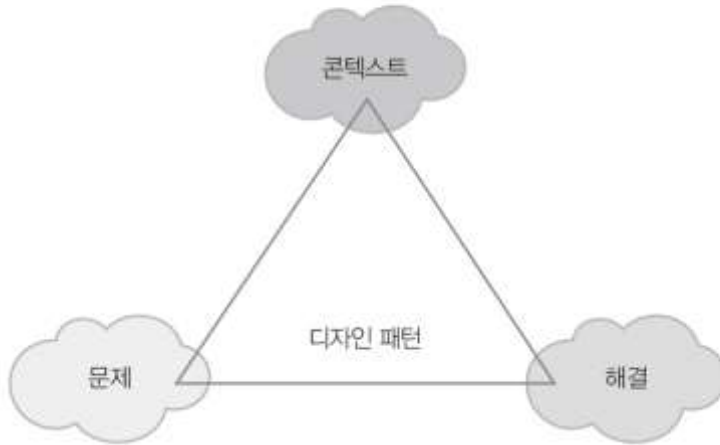
- ❖ 소프트웨어를 설계할 때 특정 맥락에서 자주 발생하는 고질적인 문제들이 발생했을 때 재사용할 수 있는 훌륭한 해결책

그림 4-1 패턴



디자인 패턴 구조

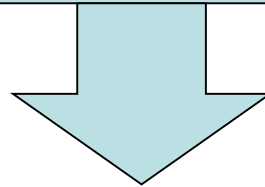
그림 4-3 디자인 패턴의 구성 요소



- ❖ 컨텍스트: 문제가 발생하는 여러 상황을 기술한다. 즉, 패턴이 적용될 수 있는 상황을 나타낸다. 경우에 따라서는 패턴이 유용하지 못하는 상황을 나타내기도 한다.
- ❖ 문제: 패턴이 적용되어 해결될 필요가 있는 여러 디자인 이슈들을 기술한다. 이때 여러 제약 사항과 영향력도 문제 해결을 위해 고려해야 한다
- ❖ 해결: 문제를 해결하도록 설계를 구성하는 요소들과 그 요소들 사이의 관계, 책임, 협력 관계를 기술한다. 해결은 반드시 구체적인 구현 방법이나 언어에 의존적이지 않으며 다양한 상황에 적용할 수 있는 일종의 템플릿이다.

디자인 패턴의 구조

- ❖ 콘텍스트: 클래스가 객체를 생성하는 과정을 제어해야 하는 상황
- ❖ 문제: 애플리케이션이 전역적으로 접근하고 관리할 필요가 있는 데이터를 포함한다. 동시에 이러한 데이터는 시스템에 유일하다. 어떤 방식으로 클래스에서 생성되는 객체의 수를 제어하고 클래스의 인터페이스에 접근하는 것을 제어해야 하는가?
- ❖ 해결: 클래스의 생성자를 `public`으로 정의하지 말고 `private`이나 `protected`로 선언해 외부에서 생성자를 이용해 객체를 일단 생성할 수 없게 만들고 (이하 생략).



싱글톤 패턴

GoF 디자인 패턴

❖ GoF(Gang of Four)

- 감마/리차드 헬름/랄프 존슨/존 블리시디시
- 생성 패턴: 객체의 생성에 관련된 패턴
- 구조 패턴: 클래스를 조합해 더 큰 구조를 만드는 패턴
- 행위 패턴: 알고리즘이나 책임의 분배에 관한 패턴

표 4-1 GoF 디자인 패턴의 분류

	생성 패턴	구조 패턴	행위 패턴
패턴 이름	추상 팩토리(Abstract Factory)	어댑터(Adapter)	책임 연쇄
	빌더(Builder)	브리지(Bridge)	(Chain of Responsibility)
	팩토리 메서드(Factory Method)	컴퍼지트(Composite)	커맨드(Command)
	프로토타입(Prototype)	데커레이터(Decorator)	인터프리터(Interpreter)
	싱글톤(Singleton)	퍼사드(façade)	이터레이터(Iterator)
		플라이웨이트(Flyweight)	미디어이터(Mediator)
		프록시(Proxy)	메멘토(Memento)
			옵서버(Observer)
			스테이트(State)
			스트래티지(Strategy)
			템플릿 메서드(Template Method)
			비지터(Visitor)

이름을 일일이
와로 필하는 법

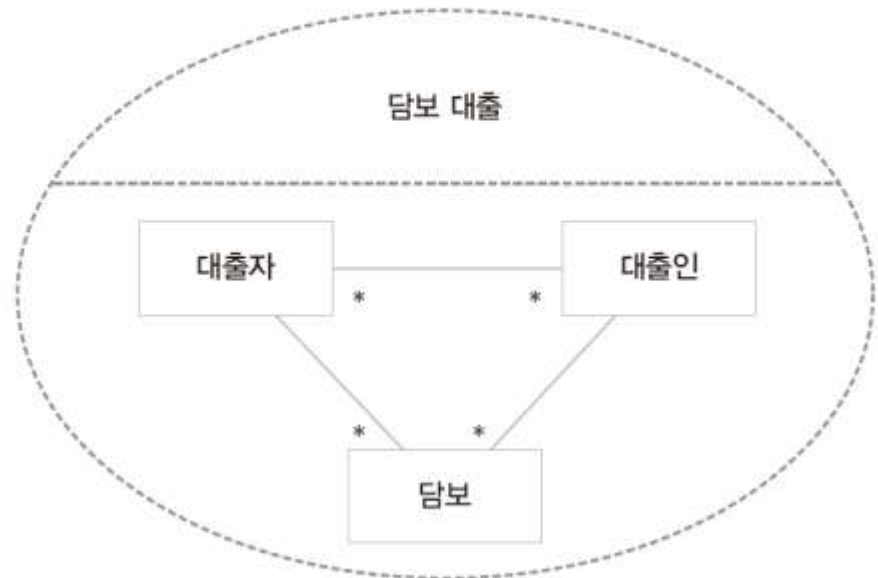
4.3 UML과 디자인 패턴

- ❖ 컬레보레이션을 통해 디자인 패턴 기술
- ❖ 역할들의 상호작용을 추상화

그림 4-4 객체와 역할



그림 4-5 컬레보레이션



컬레보레이션 어커런스

그림 4-5 컬레보레이션

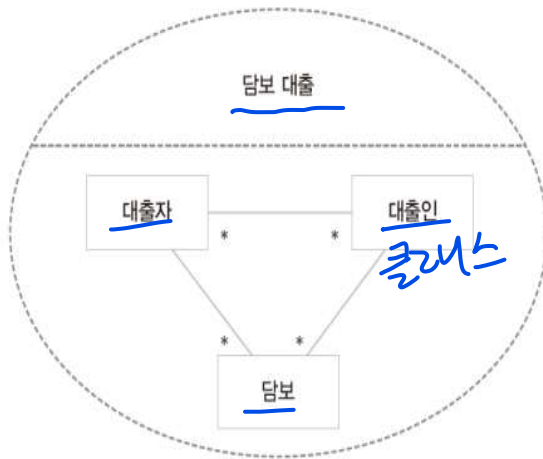
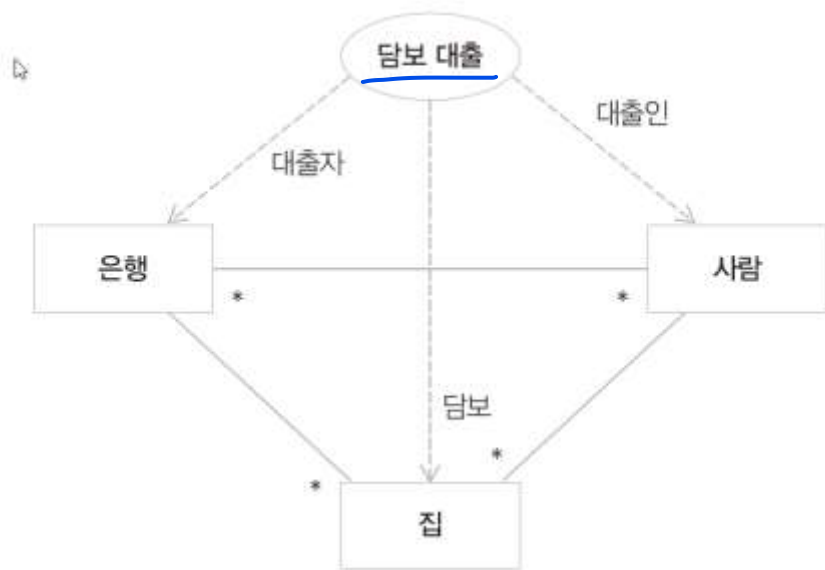


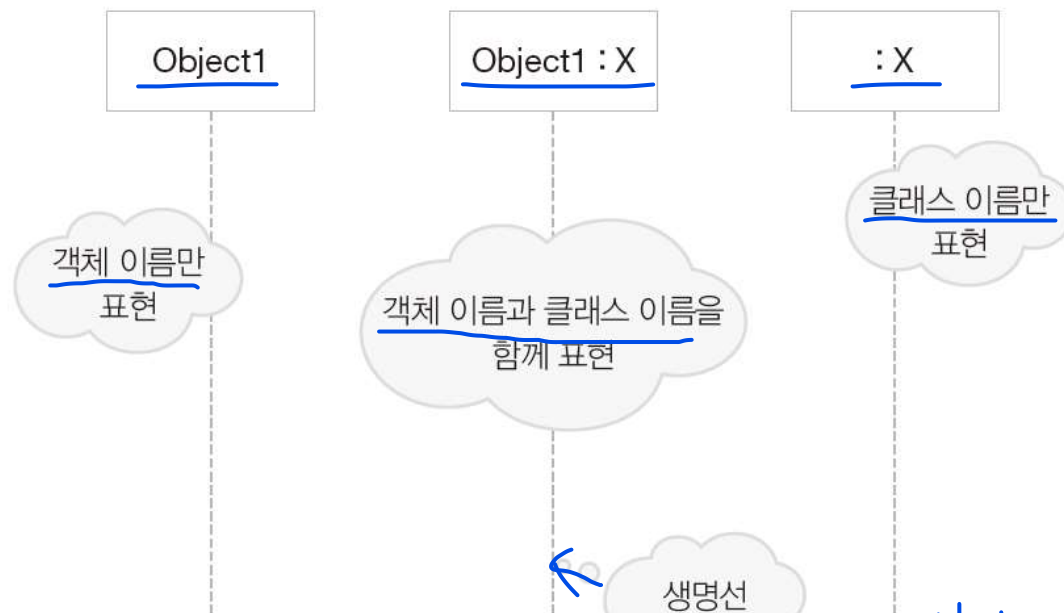
그림 4-6 컬레보레이션 어커런스



순차 다이어그램

❖ 객체들 사이의 메시지 송신과 그들의 순서

그림 4-7 객체의 3가지 표현



라이프라인 : 객체의 시작부터 끝까지 존재

Keypoint_ UML 1.X에서는 객체 이름 아래에 밑줄이 표시되지만 UML 2.0에서는 밑줄을 생략할 수 있다. 그런데 대표적 UML 도구인 starUML을 사용하면 객체 이름에 여전히 밑줄이 표시된다.

순차 다이어그램 개념 알기

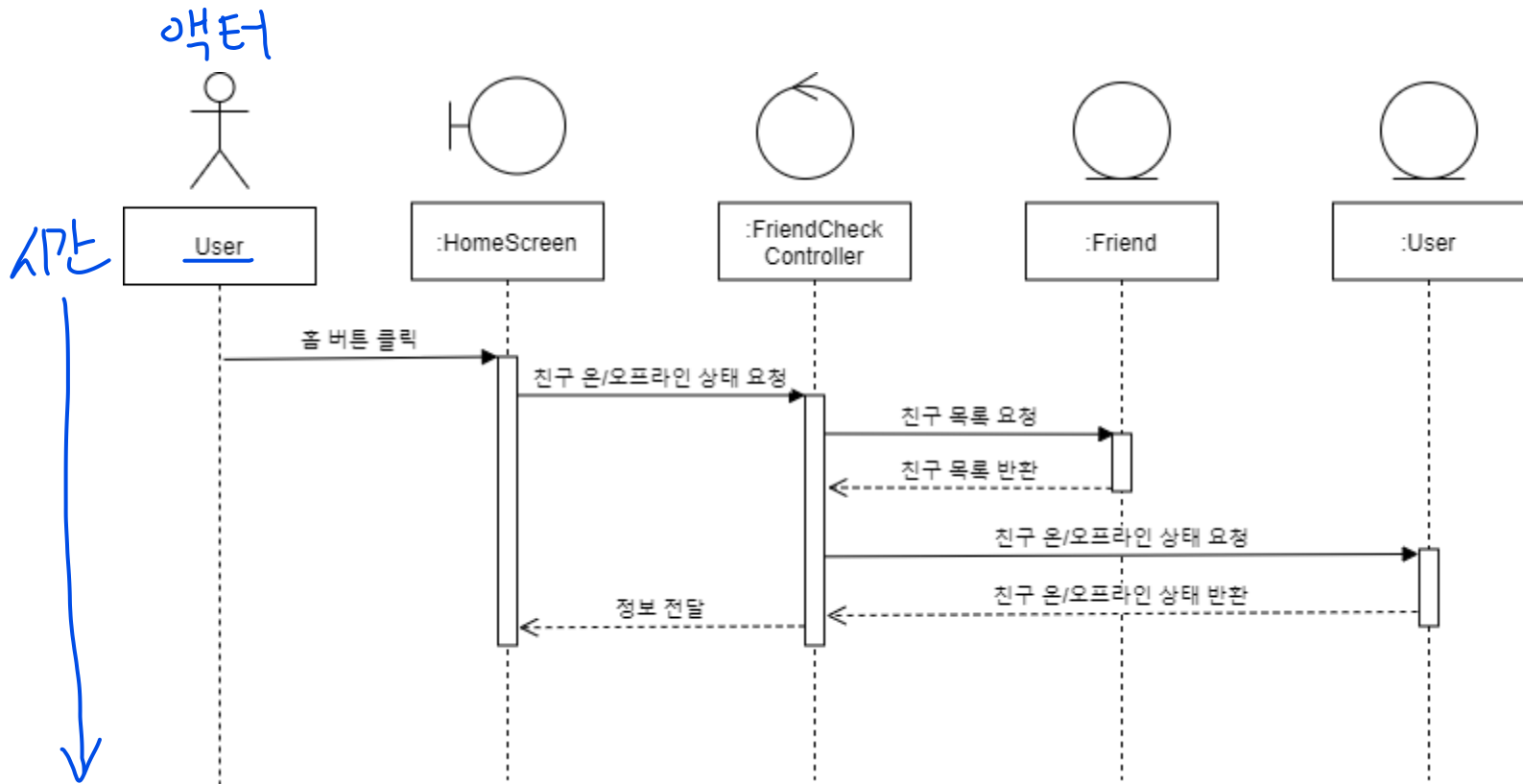
❖ 순차 다이어그램(Sequence Diagram)이란?

- 시스템 세부사항들의 진행 과정과 시간적 흐름을 표현할 수 있는 다이어그램임
- 즉 팀에서 탐색한 주제에 맞게 프로젝트가 잘 진행되고 있는지 활동의 흐름을 확인할 수 있으며, 활동을 시간 순서에 따라 자세히 표현할 수 있음

❖ 순차 다이어그램(Sequence Diagram) 구성요소

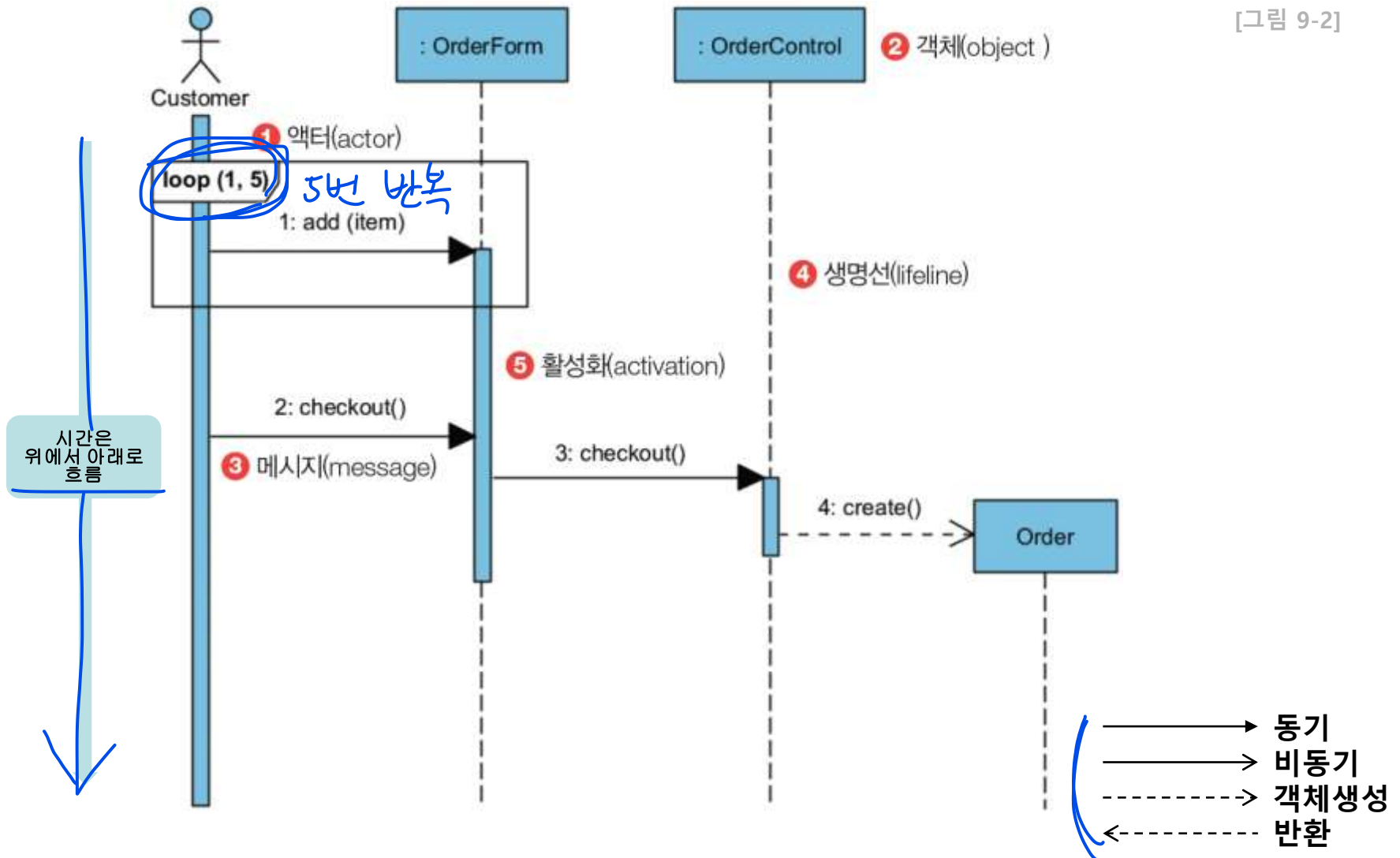
- ① 액터(actor) : 시스템을 사용하는 사람이나 외부 시스템을 의미한다.
- ② 객체(object) : 시스템 내에서 객체와 객체, 객체와 액터가 상호작용하는 개체를 의미한다.
- ③ 메시지(message) : 객체 간의 상호작용을 의미한다. 수평의 화살표 방향으로 진행되는 것을 의미한다.
- ④ 생명선(lifeline) : 객체가 존재할 수 있는 시간을 의미한다. 객체가 소멸되면 생명선도 함께 소멸된다.
- ⑤ 활성화(activation) : 객체가 활성화된 기간을 의미한다.

간단한 순차 다이어그램 예시



순차 다이어그램 개념 알기

[그림 9-2]


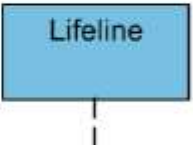
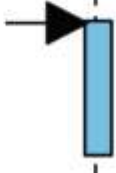


순차 다이어그램 개념 알기

❖ 순차 다이어그램의 구성요소 : 사물(Things)

- 생명선 또는 Lifeline

[표 9-1] 사물을 표현하는 요소들

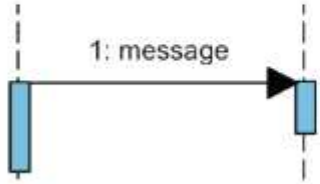

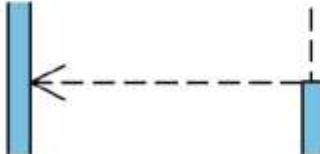
이름	그림	역할
액터 (Actor)	 Actor	<ul style="list-style-type: none">• 시스템의 외부에서 시스템을 작동시키는 주체• 사람뿐만 아니라 외부 하드웨어도 가능
생명선 (Lifeline)	 Lifeline	<ul style="list-style-type: none">• 사각형 모양의 객체(object)와 점선의 결합• 객체 : 상호작용의 개별 참여자인 객체• 점선의 길이 : 객체의 생존 기간 의미
활성화 (Activation)		<ul style="list-style-type: none">• 생명선 위의 얇은 직사각형• 객체가 활성화되어 있는 기간• 참여자가 작업을 실행하는 기간

순차 다이어그램 개념 알기

❖ 순차 다이어그램의 구성요소 : 관계(Relationships)

- 메시지(Message)가 가장 많이 쓰임
- 이는 프로그래밍 코드의 호출(call)과 밀접한 관계가 있음

[표 9-2] 관계를 표현하는 요소들 (계속)

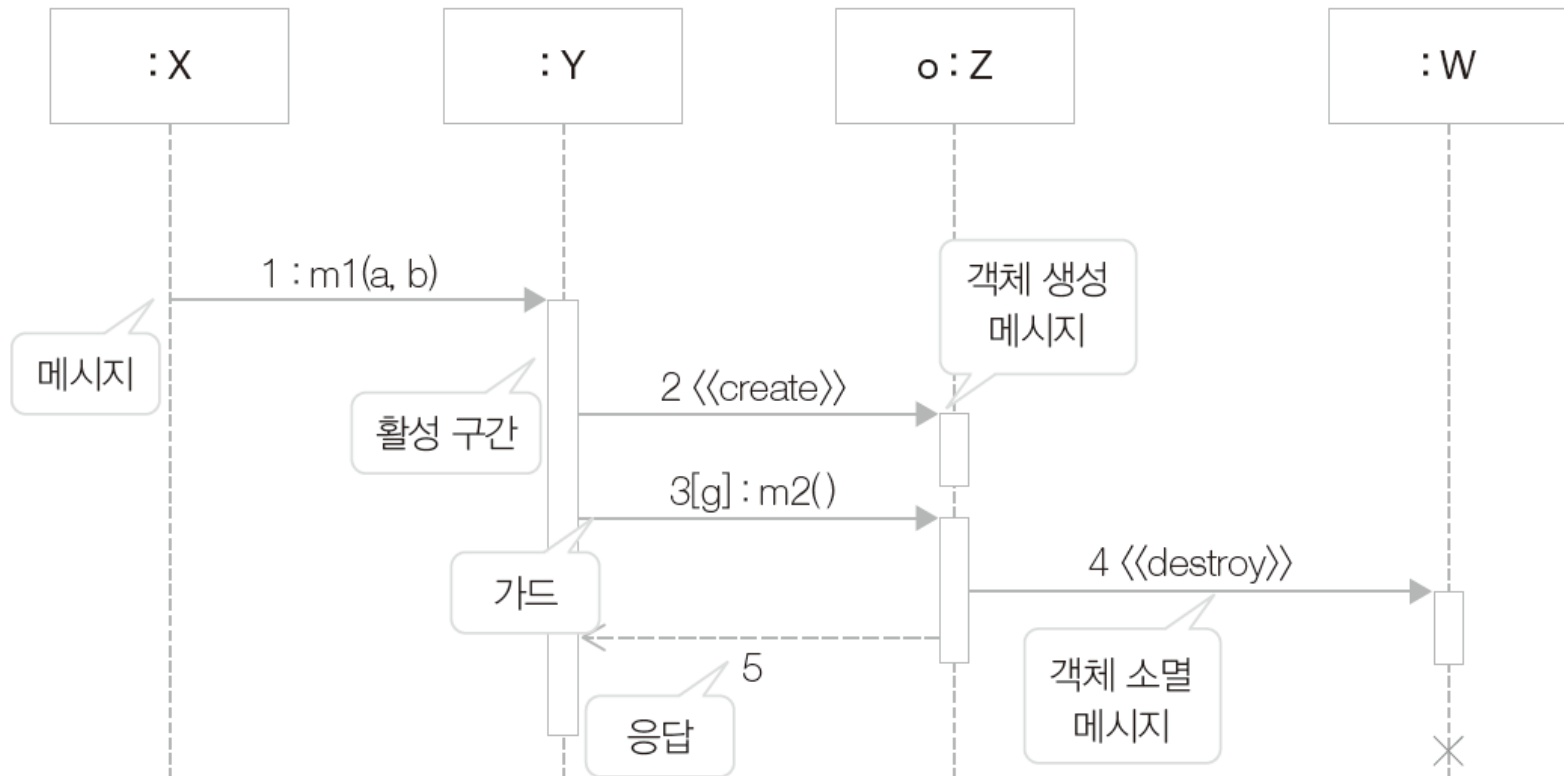
이름	그림	역할
메시지 (Message)		<ul style="list-style-type: none">• 시스템 사이의 활동 전달을 의미• 화살표로 표현(흐름의 방향)
시퀀스 메시지 (Sequence message)		<ul style="list-style-type: none">• 화살표 방향을 따라 순차적으로 수행되는 메시지
반환 메시지 (Return message)		<ul style="list-style-type: none">• 이전 호출의 반환을 기다리는 객체에게 다시 돌아오도록 하는 메시지

순차 다이어그램 개념 알기

이름	그림	역할
자체 메시지 (Self message)		<ul style="list-style-type: none"> • 동일한 생명선의 메시지 호출 • 재귀 메시지(Recursive message)라고도 불림 • 시작점 : 활성화의 상단
분실 메시지 (Lost message)		<ul style="list-style-type: none"> • 송신은 알 수 있지만, 수신은 알 수 없는 메시지
발견 메시지 (Found message)		<ul style="list-style-type: none"> • 수신은 알 수 있지만, 송신은 알 수 없는 메시지
종단 메시지 (Destroy message)		<ul style="list-style-type: none"> • 생명선(Lifeline) 전체의 실행을 중단시키는 메시지 <p>참고: X는 객체파괴</p>

메시지

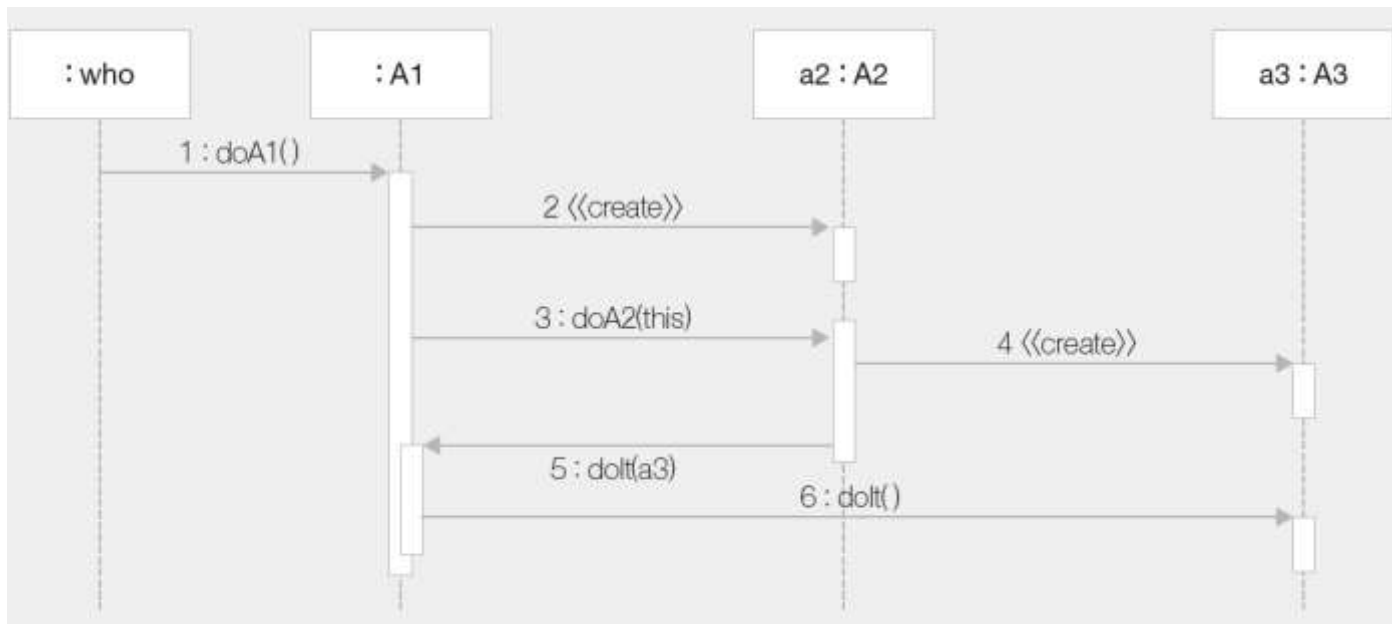
그림 4-8 여러 가지 형태의 메시지 표현



[g] : 만족시켜야 할 조건

체크포인트

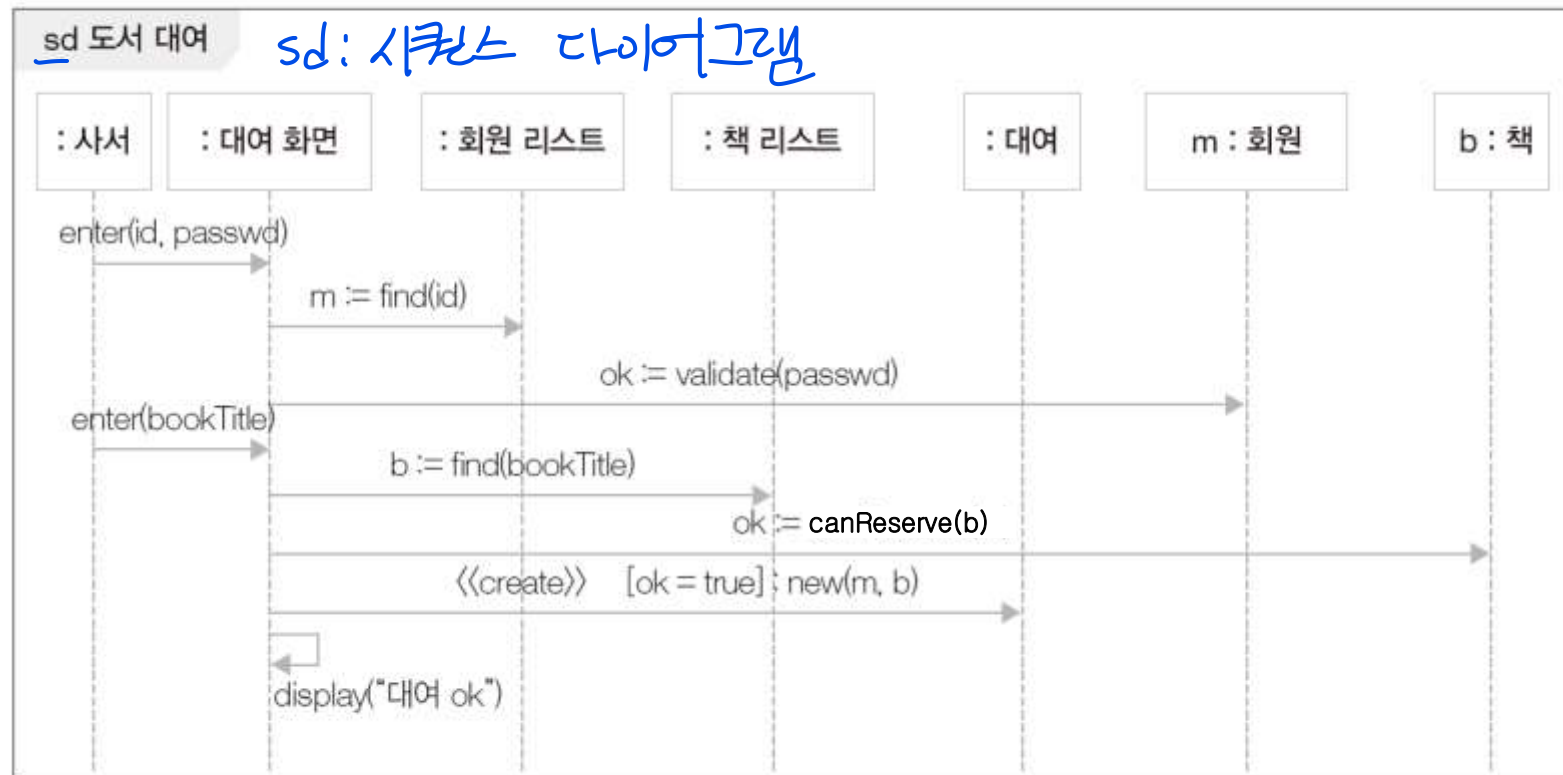
- ❖ 그림 4-8의 순차 다이어그램에 해당하는 코드를 작성하라.
- ❖ 다음 순차 다이어그램에 해당하는 코드를 작성하라.



프레임

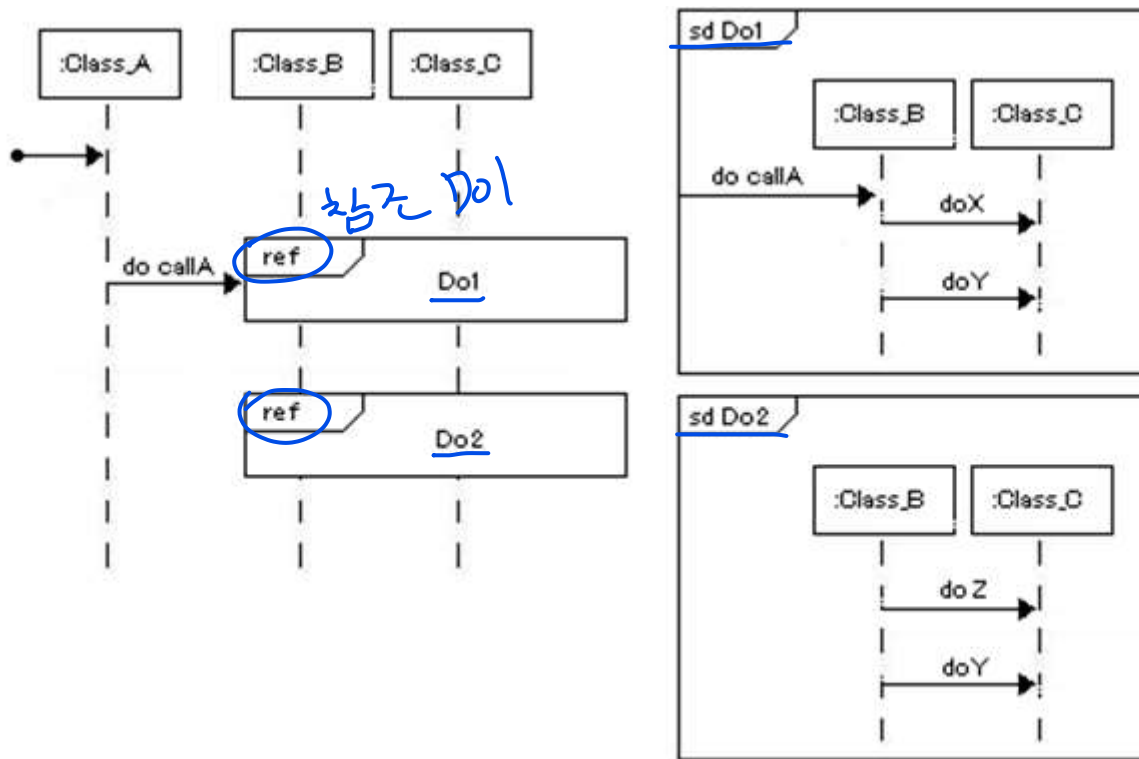
그림 4-9 프레임을 사용한 순차 다이어그램

ex. 12p



상호작용

18p 에 있던 프레임 사용, 저장해두었다가 필요한 때 마다 각다듬

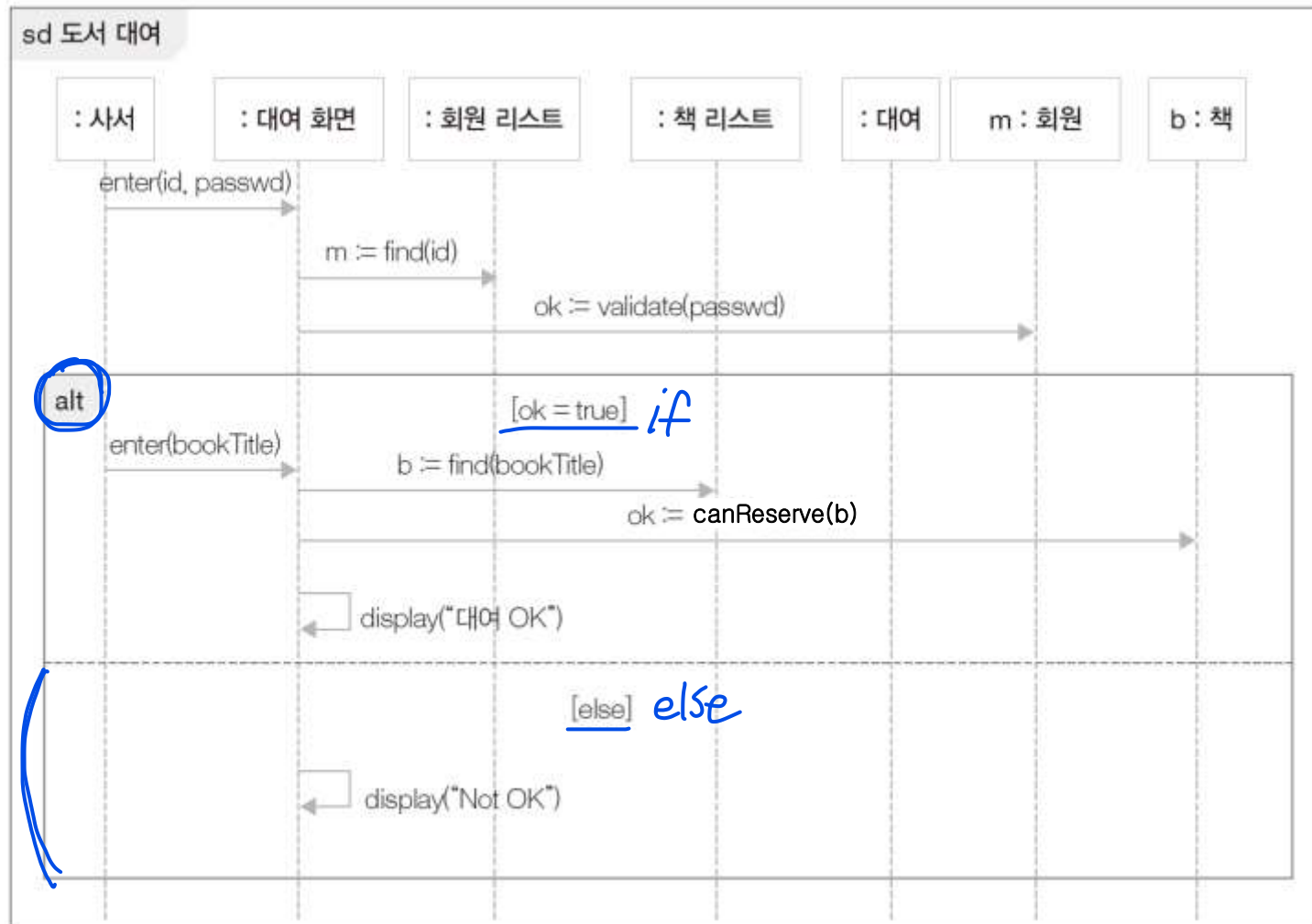


순차- 다이어그램이 복잡해지는 것을 방지

참고

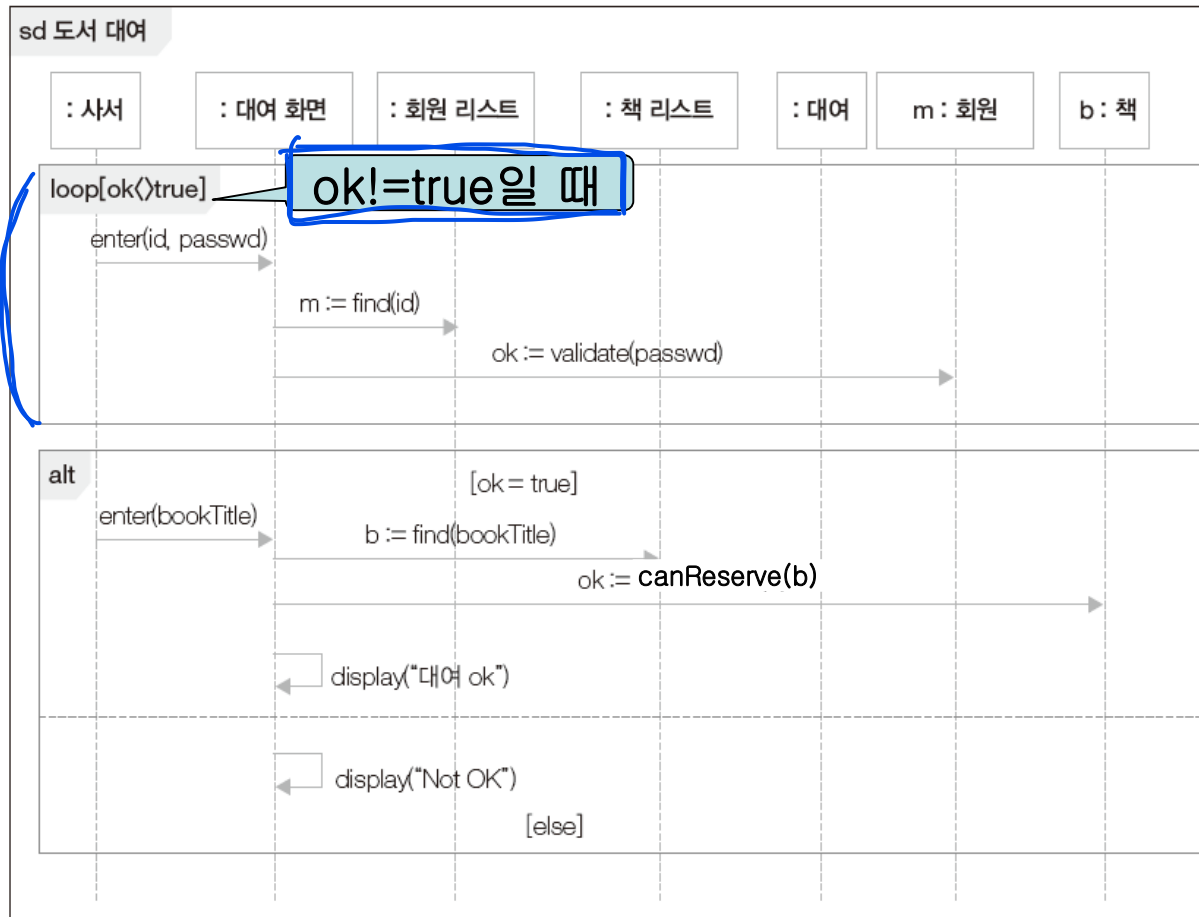
다이어그램	설명	연산자	설명
SD	순차 다이어그램	alt	여러 조건 중 하나만 수행 (alternative)
ACT	액티비티 다이어그램	opt	조건이 사실이면 수행 (optional)
CMP	컴포넌트 다이어그램	par	병행 수행(parallel)
PKG	패키지 다이어그램	loop	조건을 만족하면 반복 수행 (loop)
STM	상태 머신 다이어그램	ref	다른 상호작용 다이어그램 참조(reference)
UC	유즈케이스 다이어그램		

alt



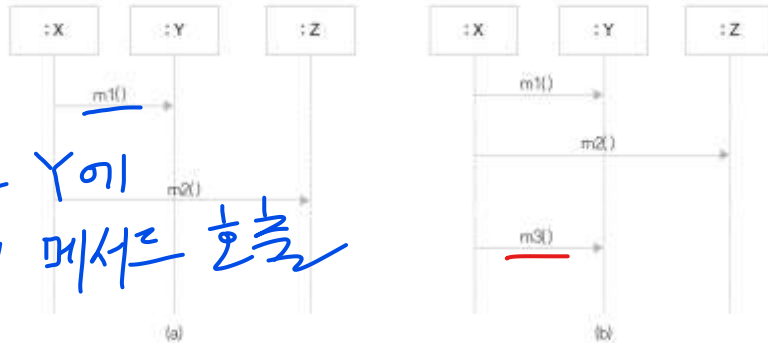
loop

그림 4-11 loop 키워드



순차 다이어그램과 클래스 다이어그램의 관계

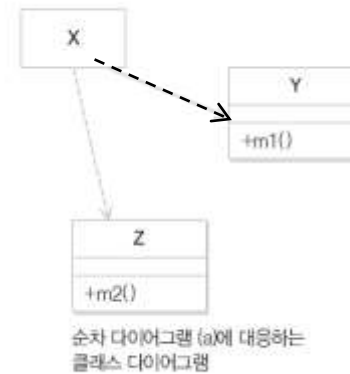
그림 4-12 연관이나 의존 관계가 존재하지 않는 순차 다이어그램



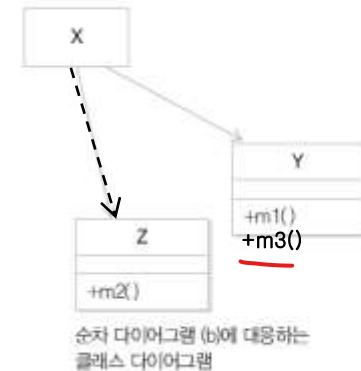
X 가 Y 의
미 메시지 흐름



그림 4-13 그림 4-12를 순차 다이어그램으로 수정

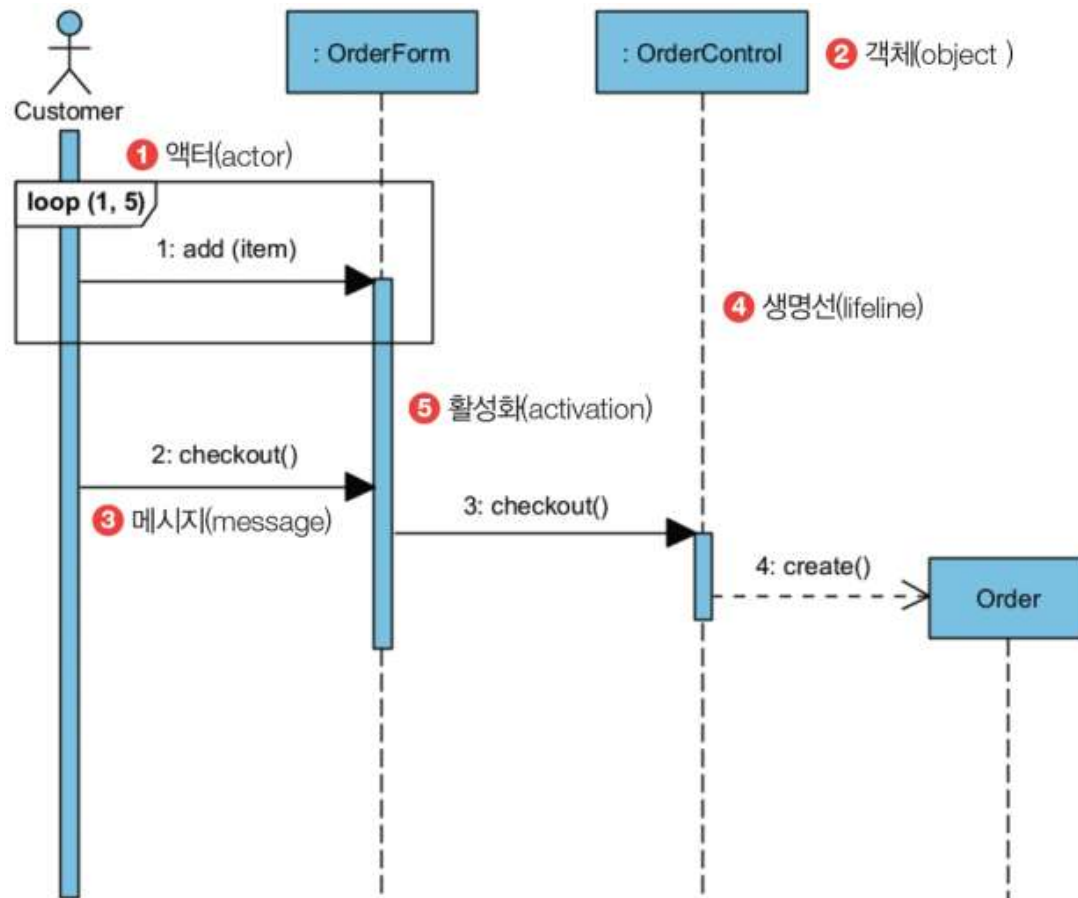


화살표 문양 기억



연습해보기

- ❖ 앞서 살펴보았던 순차 다이어그램에 대응되는 자바 코드를 구현해 봅시다.



참고 : Draw.io

- ❖ <https://app.diagrams.net/>
- ❖ 무료로 UML을 그릴 수 있는 사이트(무설치 사용 가능)
- ❖ UML 이외에도 ERD나 다른 비즈니스에 활용 가능한 다이어그램 다수 제공

참고 : Visual Paradigm

- ❖ <https://www.visual-paradigm.com/download/community.jsp>
- ❖ 유료/무료 버전 모두 제공하며, community는 비상업적 무료 사용 허용
- ❖ 설치파일 다운로드 받은 후, 설치를 완료하고 실행하면 최초 실행에 한해 이름과 메일주소를 입력하게 함
- ❖ 메일로 전송된 activation code를 프로그램에 입력하면 사용 가능
- ❖ UML뿐 아니라 ERD도 그릴 수 있음