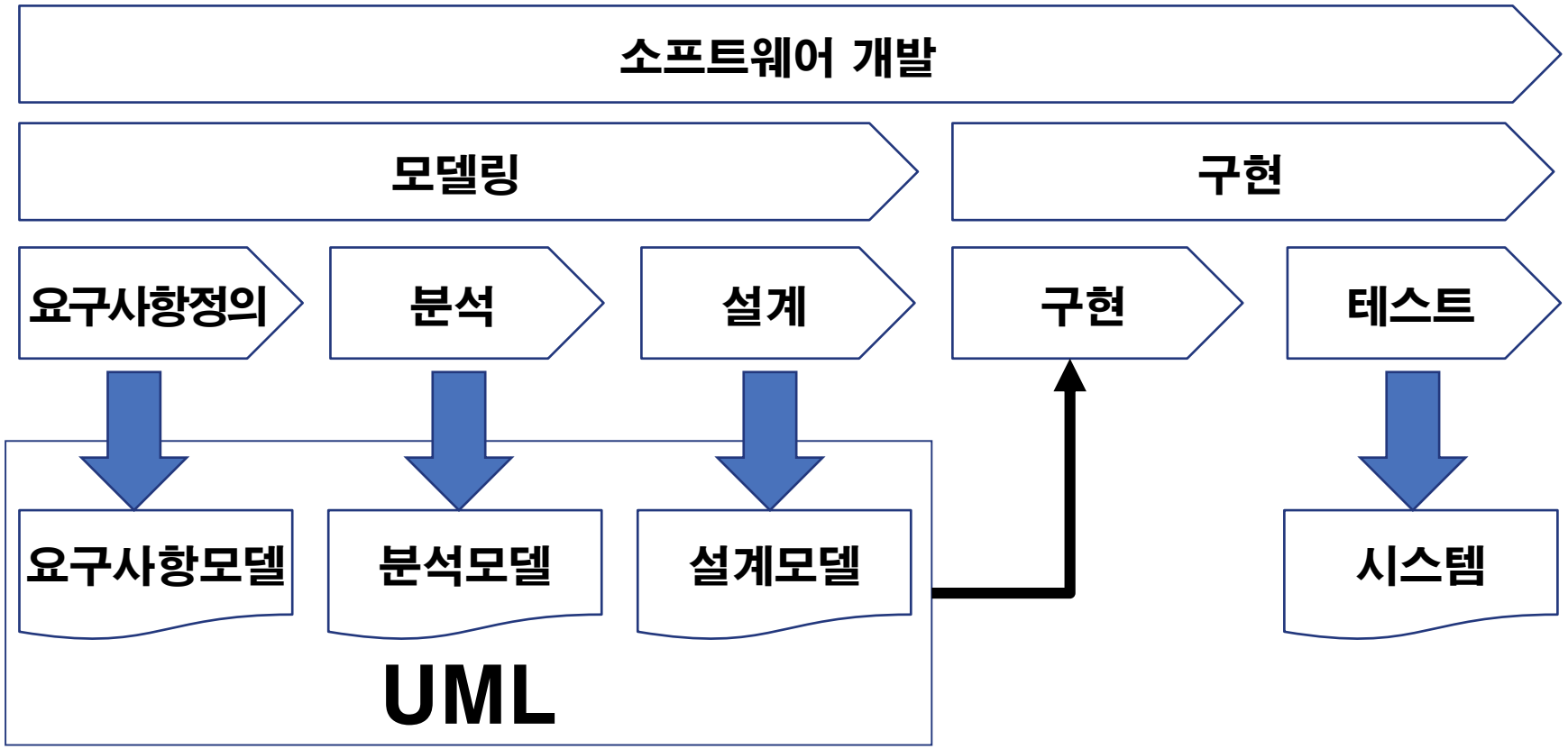
UML Diagrams

- **Structural Diagrams**
  - Class Diagram
  - Object Diagram
  - Component Diagram
  - Composite Structure Diagram
  - Deployment Diagram
  - Package Diagram
- **Behavioral Diagrams**
  - Use case Diagram
  - Sequence Diagram
  - Communication Diagram
  - State Diagram
  - Activity Diagram
  - Timing Diagram
  - Interaction Overview Diagram

Computer & Software Engineering
Lee Jong Yeol
soyeum@kumoh.ac.kr

UML Diagram

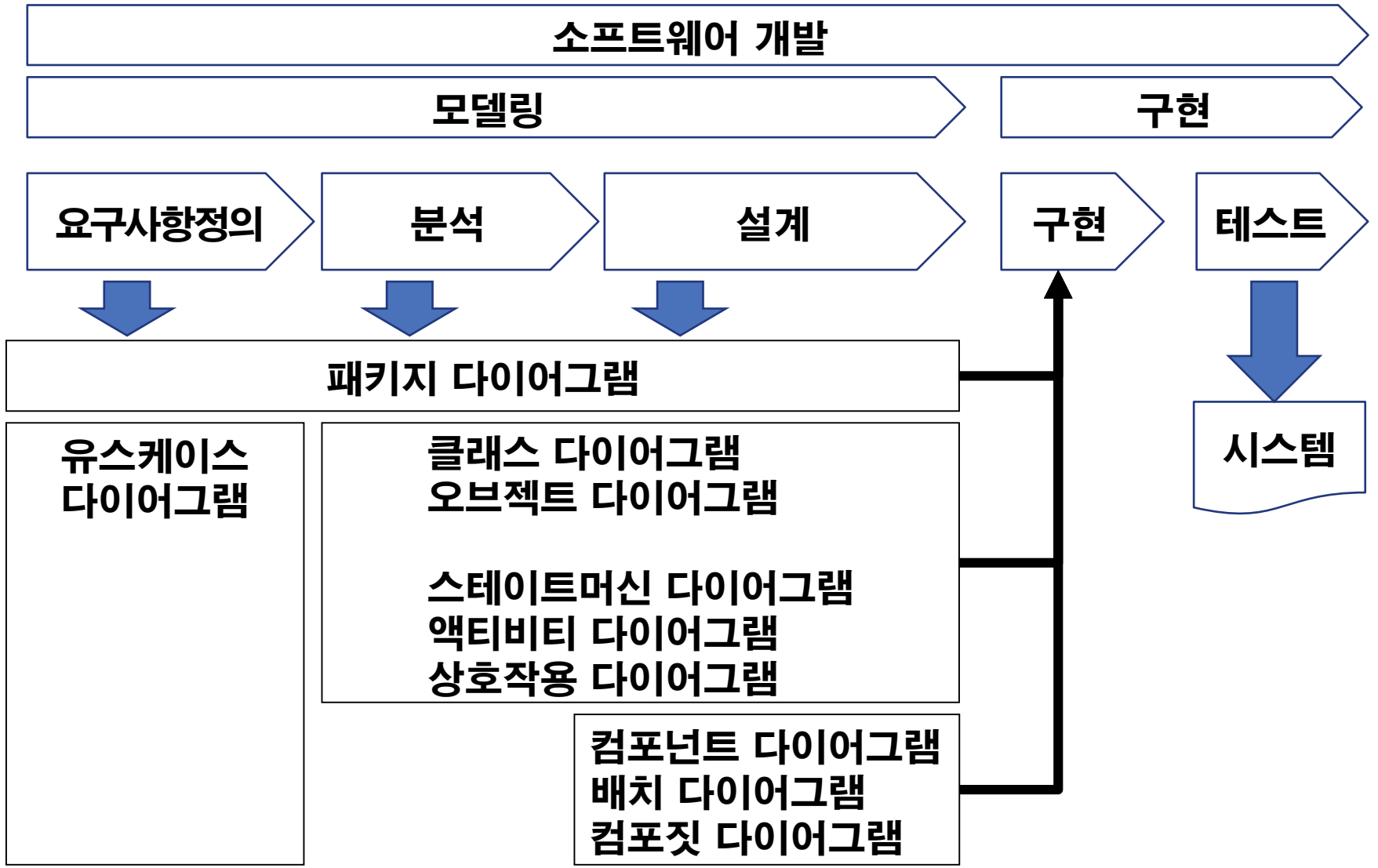
❑ Software Modeling과 UML



※ UML은 개발 전 과정에서 사용된다.

UML Diagram

❑ UML Diagram positions



UML Diagram의 분류

- Structural Diagrams
  - 시스템의 정적인 측면을 표현하는 다이어그램
  - 6개의 기본 다이어그램
    - Class Diagram, Object Diagram
    - Component Diagram , Composite Structure Diagram
    - Deployment Diagram, Package Diagram

- Behavioral Diagrams
  - 시스템의 동적인 측면을 표현하는 다이어그램
  - 7개의 기본 다이어그램
    - Use case Diagram, Sequence Diagram
    - Communication Diagram, State Diagram, Activity Diagram
    - Timing Diagram, Interaction Overview Diagram

Class Diagram(1)

- 클래스 다이어그램이란?
  - 클래스, 인터페이스, 협동체와 이들 사이의 관계를 표현
  - 가장 널리 사용되는 필수 다이어그램
    - 컴포넌트 다이어그램 및 배치 다이어그램의 기반

- 일반적인 용도
  - 시스템의 정적(static) 구조를 표현
    - 액티브(active) 클래스는 프로세스를 생성시킬 수 있으므로 동적인 측면을 포함하고 있음
  - 단순한 협동체(collaboration)의 구조를 표현
  - DB schema 설계에 사용 -> ER 다이어그램을 대체

Class Diagram(2)

- 클래스 다이어그램의 구성 요소
  - 중요한 구성 요소
    - 클래스 -> 추상 클래스는 *italic* 체로 표기
    - 인터페이스
    - 협동체
    - 종속 관계, 일반화 관계, 연관 관계
  - 부수적인 구성 요소
    - 주석(notes)
    - 제약사항(constraints)
    - packages & subsystems -> 구성 요소의 그룹화에 사용
    - 클래스에 속하는 객체 (설명을 위해 특별히 필요한 경우에 포함)

Class Diagram(3)

❑ 클래스 다이어그램 작성법

단계1: 모델링 대상을 선정

※ 대상: 시스템이 제공하는 기능 또는 특정 함수

단계2: 모델링 대상에 대해서 참여하는 클래스, 인터페이스,
다른 협동체를 찾는다.

단계3: 클래스들 사이에 존재하는 관계를 찾는다.

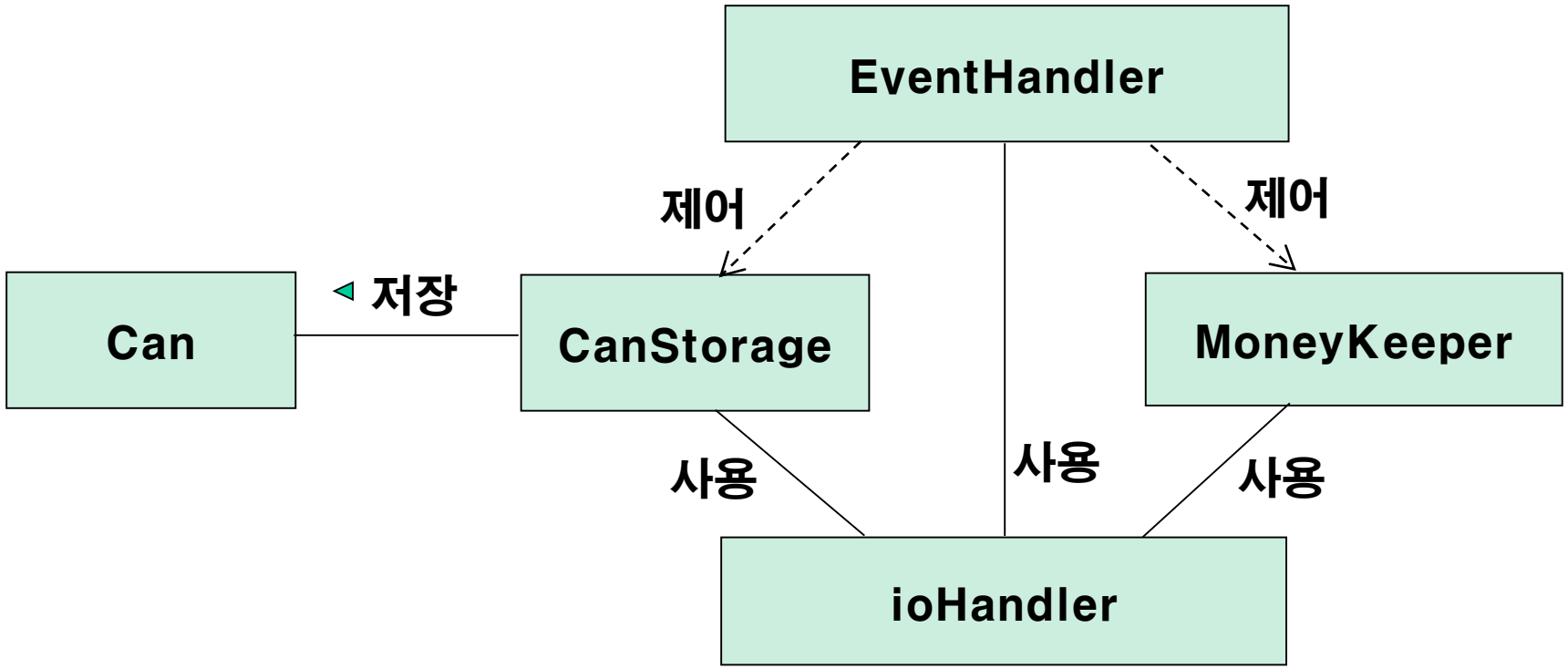
단계4: 전형적인 시나리오를 따라 가면서 누락된 클래스 등이 있으면 추가

단계5: 클래스 등에 대한 상세 내역을 추가

※ 클래스 -> 책임, 속성, 오퍼레이션

Class Diagram 예시

❑ 음료수 자판기 사례



Object Diagram(1)

- 객체 다이어그램이란?
  - 실행 중에 나타나는 객체와 이들 사이의 관계를 표현
  - 클래스 다이어그램의 실제 모습 또는 전형적인 경우를 보여 주기 위해 작성
  - 실제로 생성되는 모든 객체를 표현할 수는 없으며,
    통상적으로 특별히 관심의 대상이 되는 전형적인 객체들만 표현

- 일반적인 용도
  - 시스템의 정적인 구조를 표현
  - 시스템이 작동하고 있는 동안에 나타날 수 있는 특정 순간의 모습을 포착
  - 정적인 자료, 특히 복잡한 자료 구조를 표현하는데 유용

Object Diagram(2)

- 객체 다이어그램의 구성 요소
  - 중요한 구성 요소
    - 객체
    - links
    - 주석(notes)
    - 제약사항(constraints)
  - 부수적인 구성 요소
    - packages & subsystems -> 구성 요소의 그룹화에 사용
    - 클래스에 속하는 객체 -> 설명을 위해 특별히 필요한 경우에 포함
    - 클래스 -> 특정 객체가 어느 클래스에 속하는 것인가를 보여줄 필요가 있을 때 포함

Object Diagram(3)

❑ 객체 다이어그램 작성법

단계1: 모델링 대상을 선정

※ 대상: 시스템이 제공하는 기능 또는 특정 함수

단계2: 모델링 대상에 대해서 관련된 클래스, 인터페이스,
그 외의 요소를 찾고, 이들 사이의 관계를 파악

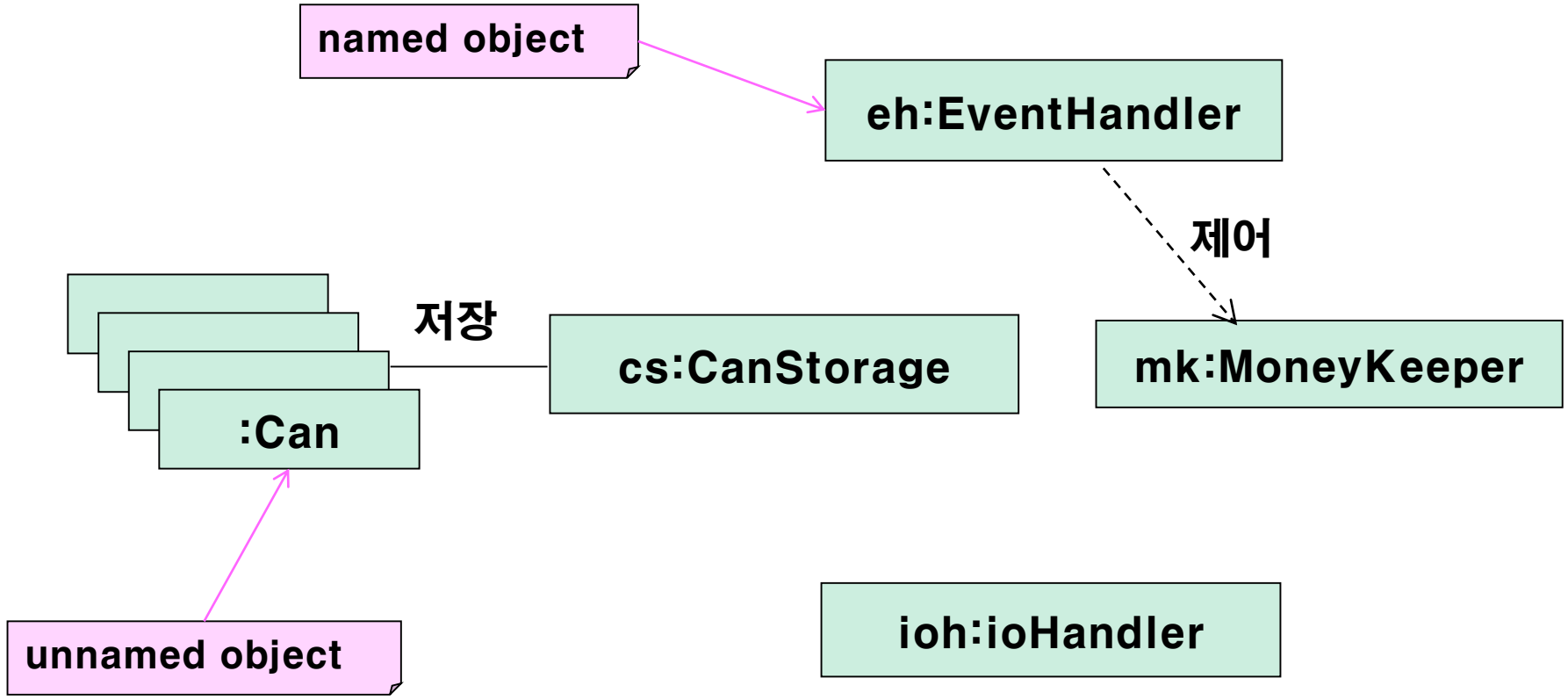
단계3: 모델링 대상에 관한 시나리오의 한 순간을 포착하고,
포착된 장면을 상세히 기술한다.

단계4: 중요한 객체에 대해서 현재 상태, 속성 값 등
필요한 정보를 다이어그램에 포함시킨다.

단계5: 객체 사이의 link를 찾아 다이어그램에 표현

Object Diagram 예시

❑ 예시: 자판기에 동전을 투입하는 순간을 포착



Component Diagram(1)

- 컴포넌트 다이어그램이란?
  - 시스템 또는 컴포넌트를 구성하는 캡슐화된 클래스, 인터페이스, 포트(port), 하위 컴포넌트와 이들 사이의 관계를 보여주는 다이어그램
  - 클래스 다이어그램과 엄격하게 구분하기 어려움
  - 복합 구조 다이어그램
    - 시스템을 실행할 때의 구조를 표현할 수 있음
    - 컴포넌트 다이어그램으로 대치하는 경우가 많으므로, 사용 빈도가 낮음

- 일반적인 용도
  - 시스템의 정적인 구조를 표현
  - 컴포넌트의 구성 내역을 표현

Component Diagram(2)

- 컴포넌트 다이어그램의 구성 요소
  - 중요한 구성 요소
    - 컴포넌트 or 캡슐화된 클래스
    - 인터페이스
    - 외부와의 연결 창구인 포트(port)
    - 컴포넌트 사이의 관계 -> 종속, 일반화, 연관, 실현 관계
  - 부수적인 구성 요소
    - 주석
    - 제약사항
    - packages

Component Diagram(3)

❑ 컴포넌트 다이어그램 작성법

단계1: 모델링 대상을 선정

※ 대상: 시스템, 컴포넌트

단계2: 모델링 대상을 구성하는 클래스, 인터페이스, 하위 컴포넌트를 찾는다.

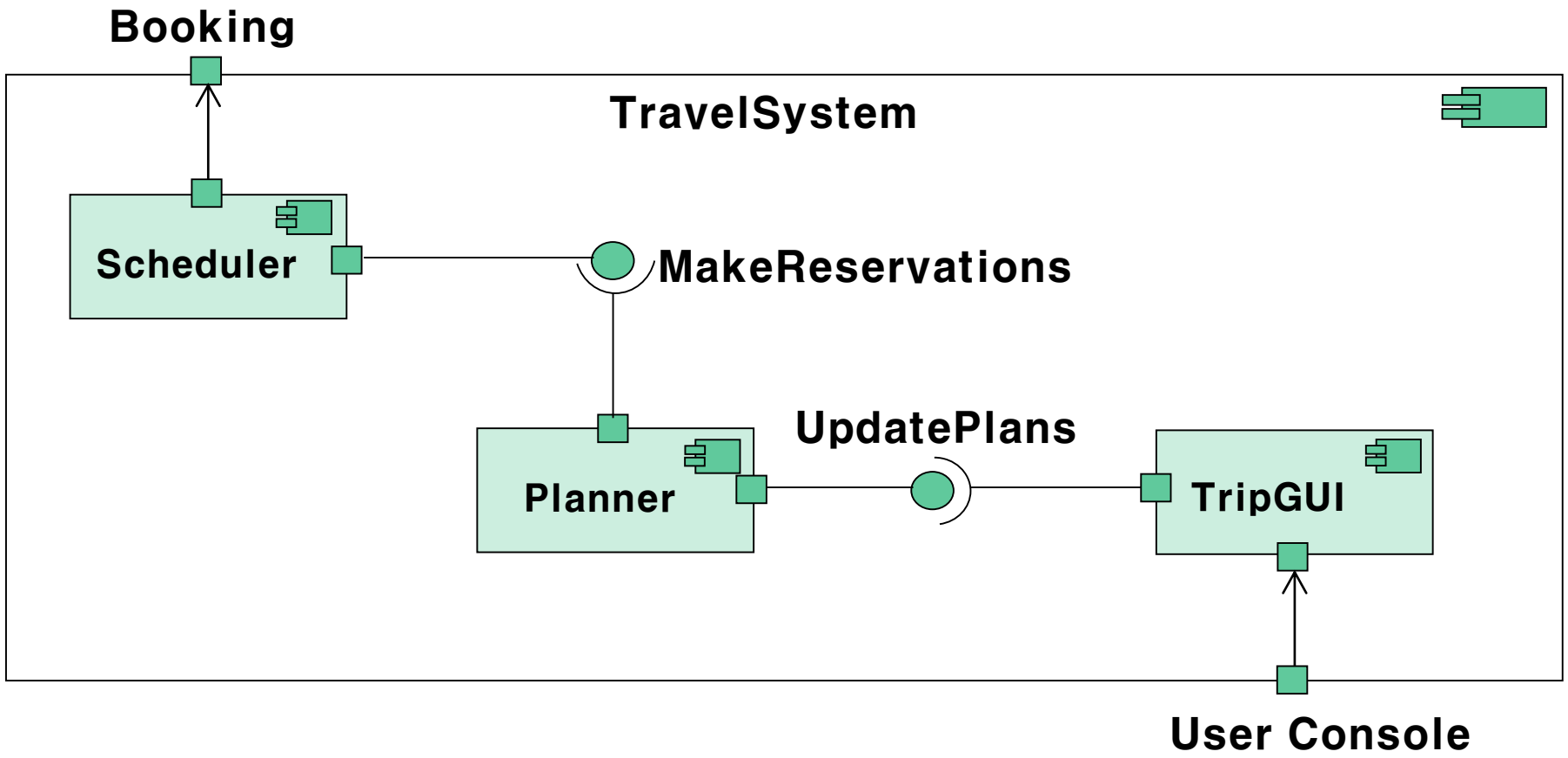
단계3: 구성 요소 사이에 존재하는 관계를 찾는다.

단계4: 모델링 대상과 외부와의 창구인 포트를 찾는다

단계5: 포트를 통하여 소통하는 구성 요소를 기술

Component Diagram 예시

❑ 여행관리 시스템 사례



Artifact Diagram(1)

- 산출물 다이어그램이란?
  - 시스템 개발 과정에서 생산된 산출물의 구성과 이들 사이의 관계를 표현
  - 시스템의 산출물에 초점: 실행 파일, 라이브러리, DB 테이블, 각종 파일, HTML 문서, XML 문서 , …
  - 산출물 사이의 관계: 종속, 일반화, 연관, 실현

- 일반적인 용도
  - 시스템의 정적인 구현 뷰(view)를 모델링
    - 소스 코드의 형상(Configuration)
    - 고객에게 배포하는 실행 파일
    - DB 스키마 및 테이블
    - 시스템의 실행에 참여하는 라이브러리

Artifact Diagram(2)

- Artifact 다이어그램의 구성 요소
  - 중요한 구성 요소
    - 산출물(Artifacts)
    - 산출물 사이의 관계: 종속, 일반화, 연관, 실현 관계
  - 부수적인 구성 요소
    - 주석
    - 제약 사항

Artifact Diagram(3)

❑ 산출물 다이어그램 작성법(소스 코드의 형상)

단계1: 표현 대상에 포함시킬 소스 코드 파일을 선정

단계2: 선정된 파일을 해당하는 스테레오 타입으로 표현

단계3: 시스템이 대형인 경우에는 패키지를 사용하여 소스코드 파일을 그룹화

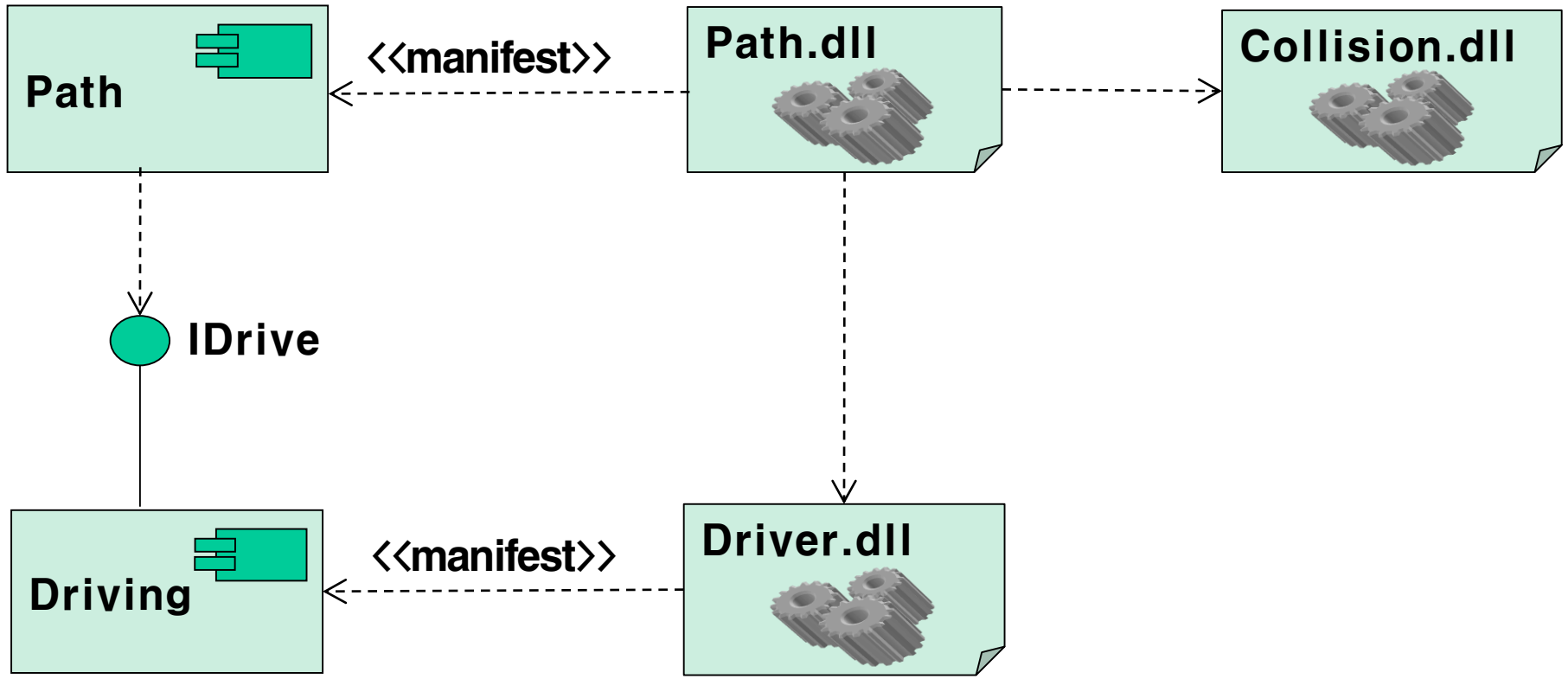
단계4: 소스 코드 파일의 버전 번호, 저자, 최근 변동일 등의
자료를 태그로 표시

※ 태그에 표기할 값은 도구를 사용하여 자동화

단계5: 컴파일 우선 순위를 소스 코드 파일 사이의 종속 관계로 표현

Artifact Diagram 예시

❑ 로봇의 경로 탐색 부품 사례



Use Case Diagram(1)

- 유즈 케이스 다이어그램이란?
  - 유즈 케이스, 액터, 그리고 이들 사이의 관계를 표현하는 다이어그램
    - 시나리오: 실제로 발생하는 일을 기술한 유즈 케이스의 instance
    - 유즈 케이스와 액터 사이의 관계: communication
  - 시스템의 기능을 표현하는 핵심적 다이어그램
    - 시스템의 기능적 요구를 추출하고 분석하는데 중요한 역할
    - 시스템 테스트에 자주 사용

- 일반적인 용도
  - 시스템의 범위를 설정
  - 시스템의 요구사항을 표현

Use Case Diagram(2)

- 유즈 케이스 다이어그램의 구성 요소
  - 중요한 구성 요소
    - subject(기술 대상의 명칭)
    - Use Cases(유즈 케이스)
    - Actor(액터)
    - 관계: communicate, include, extend, generalization
  - 부수적인 구성 요소
    - 주석
    - 제약 사항
    - packages
    - use case의 instance

Use Case Diagram(3)

❑ 유즈 케이스 다이어그램 작성법

단계1: 시스템의 범위를 결정

단계2: 시스템과 관련된 액터를 파악하고, 이들 사이의 관계를 조사하여 기술

단계3: 각각의 액터에게 시스템이 제공해야 하는 서비스를 파악하여
유즈 케이스로 표현

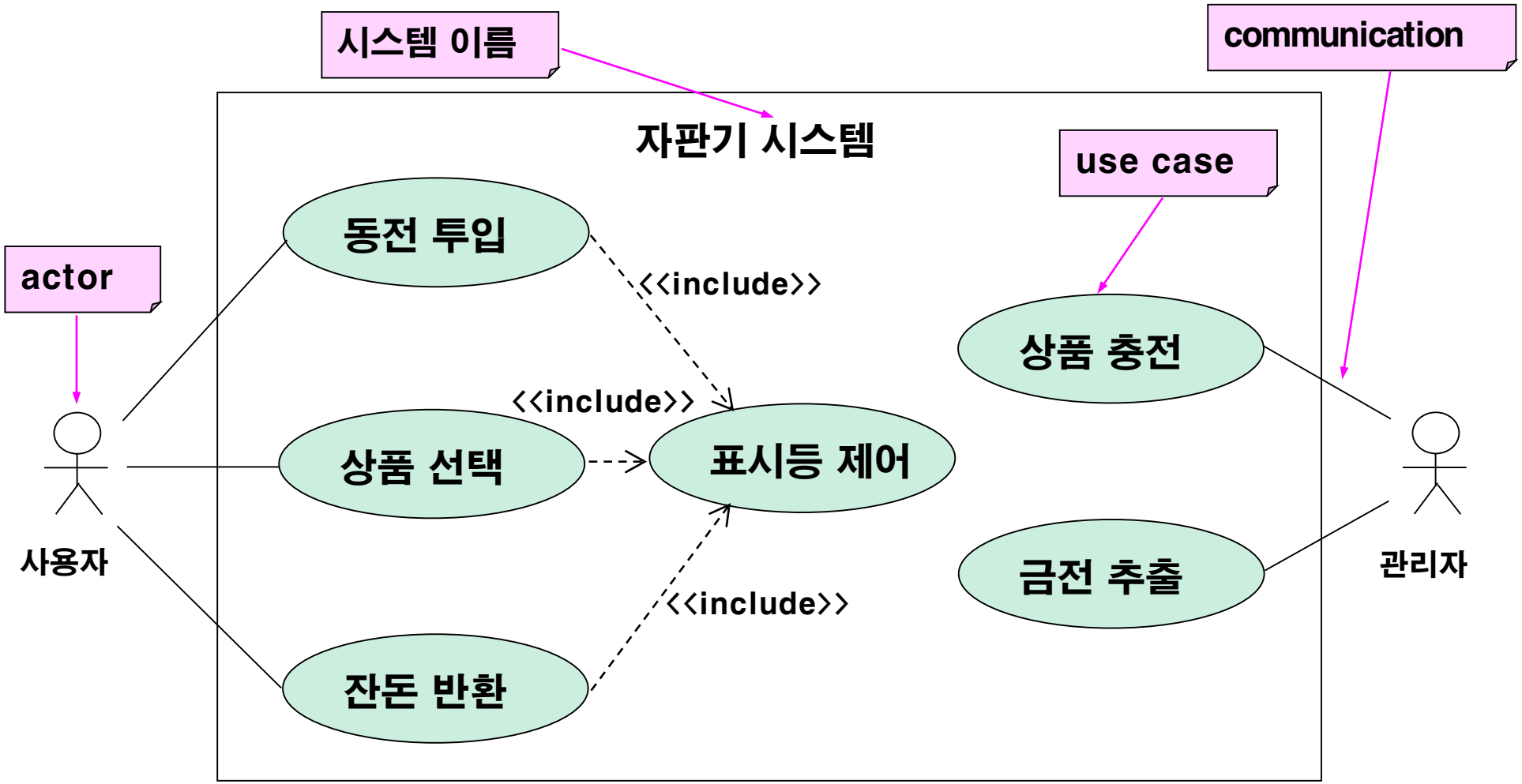
단계4: 공통된 기능(inclusion)과 확장 기능(extension)을
추출하여 별도의 유즈 케이스로 작성

단계5: 유즈 케이스 사이의 관계를 파악하여 기술

단계6: 비기능적 요구사항을 주석이나 제약사항으로 표시

Use Case Diagram 예시

❑ 음료수 자판기 사례



Interaction Diagram

❑ 기본 개념
- 객체 사이의 상호작용을 표현하는 다이어그램
  - storyboards를 표현할 수 있는 강력한 도구
  - 작동 환경 , 객체의 역할, 객체 사이의 link, 메시지와 그 순서
- Sequence Diagram과 Communication Diagram
  - Sequence Diagram -> 메시지 발생 순서에 초점
  - Communication Diagram -> 객체의 구성을 강조
  - 2종류의 다이어그램은 isomorphic

❑ 일반적인 용도
- 시스템의 동작을 개괄적으로 기술
  - 업무수행절차를 기술하는데도 매우 유용
- 유즈 케이스가 수행되는 과정을 표현
- 오퍼레이션의 알고리즘

Sequence Diagram(1)

- 시퀀스 다이어그램의 중요한 구성 요소
  - 객체
  - 메시지: 수신 객체로 가는 화살표. 시간 축을 따라 진행
    - 비동기적 메시지 -> 굵은 실선 화살표
    - 동기적 메시지 -> 채워진 삼각형 헤드를 갖는 화살표
    - 동기적 메시지에 대한 응답 -> 점선 화살표
  - 생명선(life line): 수직 점선. 객체의 존속 기간을 표시
    - 처음부터 끝까지 존속하거나, create() 메시지를 받는 순간부터 destroy() 메시지를 받을 때까지 존속
  - focus of control: 길고 가는 직사각형. 객체가 어떤 작업을 수행하는 기간을 표시
    - nested focus of control -> 재귀, 다른 객체로부터의 호출

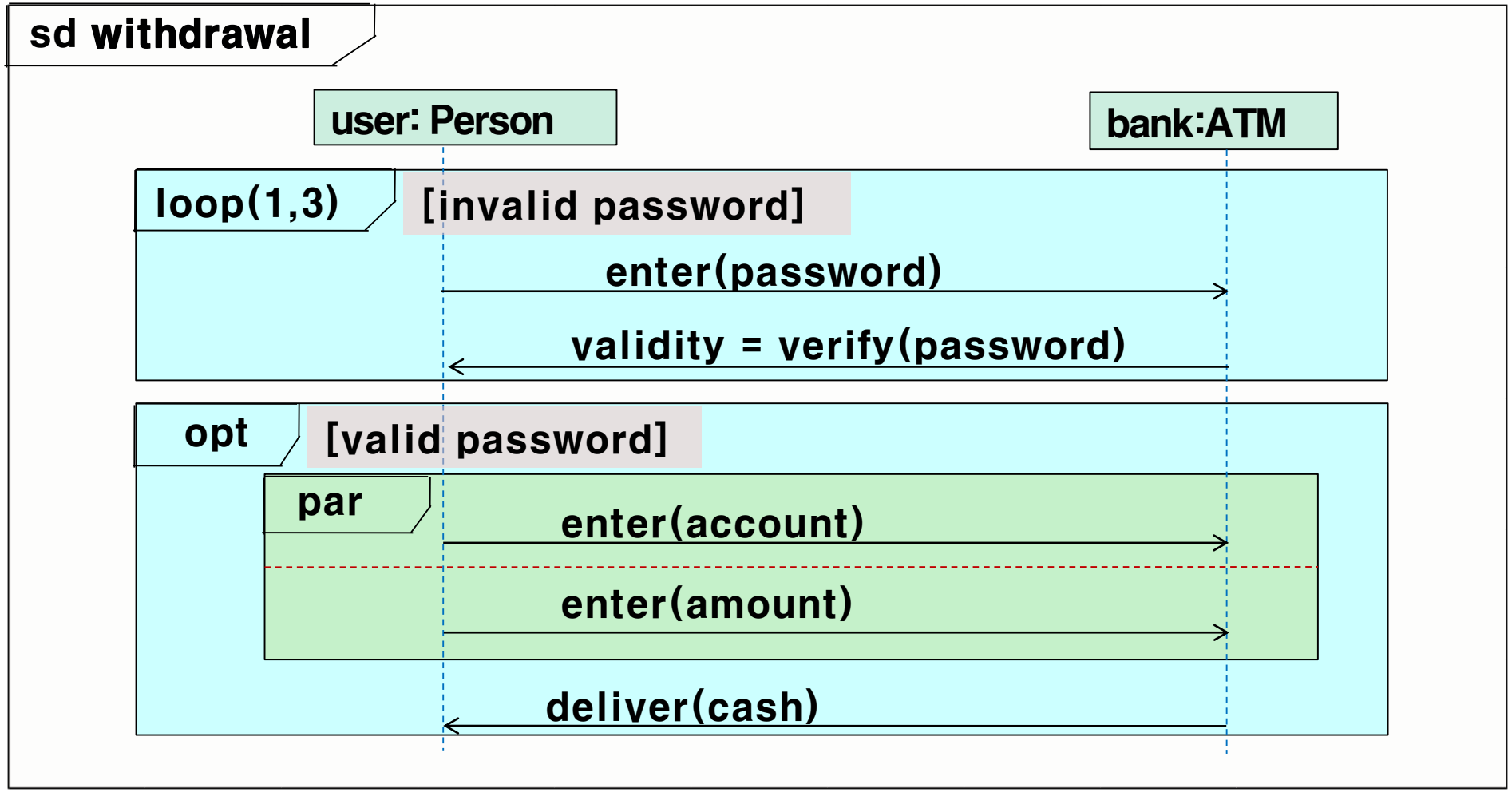
Sequence Diagram(2)

- 제어 연산자(control operator): 직사각형으로 표기
  - 태그(tag): 좌측 상단의 작은 오각형. 제어의 종류 표시
  - 선택적 수행(opt) -> guard condition이 참일 때 수행
  - 조건부 수행(alt) -> 제어 연산자를 수직 영역으로 구분하고, 각 영역은 guard condition이 참일 때 수행
  - 병렬 수행(par) -> 제어 연산자를 수평 영역으로 구분
  - 반복 수행(loop) -> guard condition이 참인 동안 수행
  - 시퀀스 다이어그램(sd) -> 시퀀스 다이어그램을 표시
  - 하위 작업(ref) -> activity, state, sequence diagram 등으로 표현한 하위 작업을 참조

❑ 시퀀스 다이어그램의 부수적인 구성 요소
- 주석, 제약사항
- packages

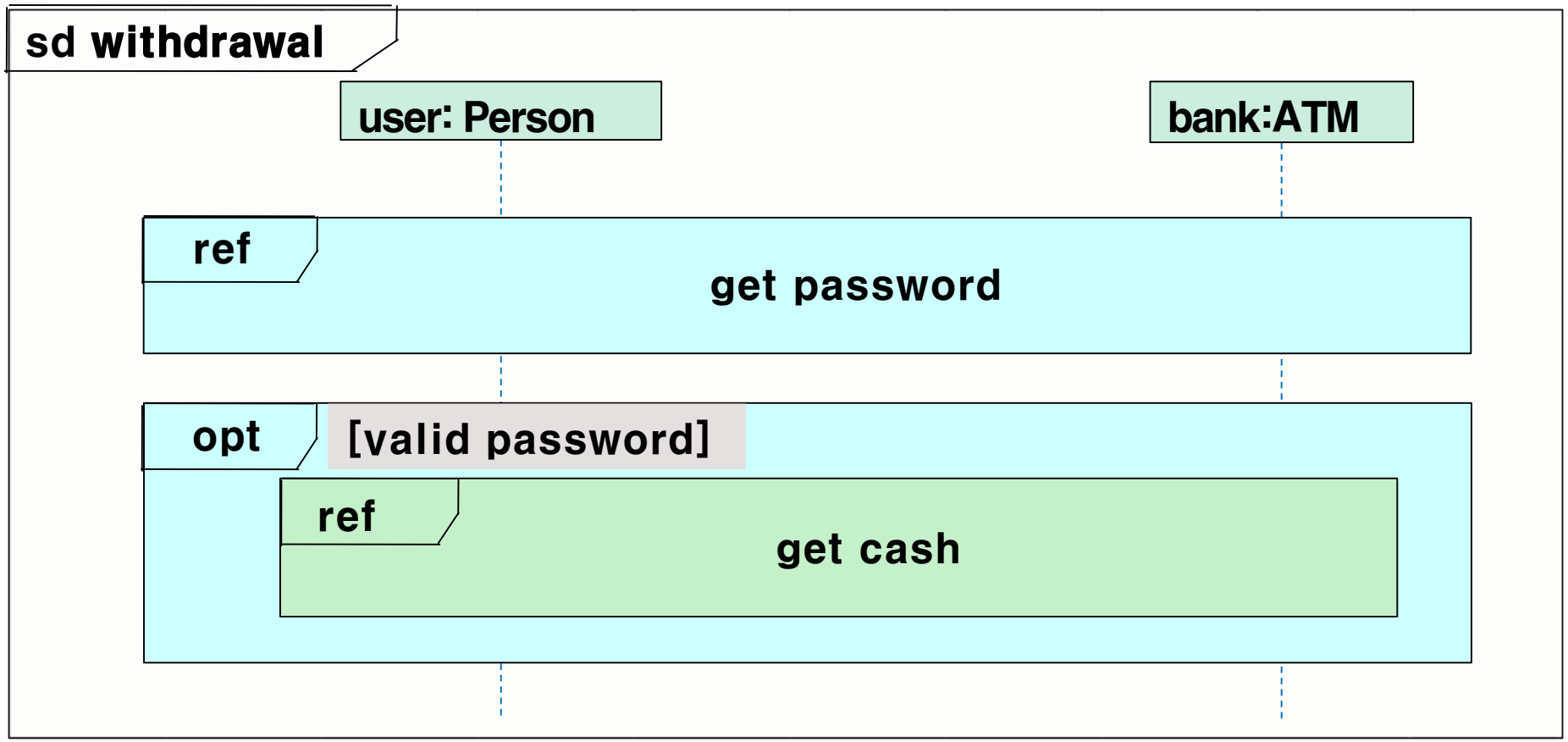
Sequence Diagram(3)

❑ 제어 연산자 작성 예1



Sequence Diagram(4)

❑ 제어 연산자 작성 예2



Sequence Diagram(5)

❑ 시퀀스 다이어그램 작성법

단계1: 모델링 대상을 선정

※ 대상: 유즈 케이스 또는 협동체

단계2: 참여 객체를 상단에 나열

※ 중요한 객체부터 왼쪽에서 오른쪽으로!

단계3: 각각의 객체에 대하여 생명선을 그린다.

단계4: 상호작용을 시작하는 메시지부터 Y축을 따라 차례로 추가.
메시지는 송신 객체에서 수신 객체로!

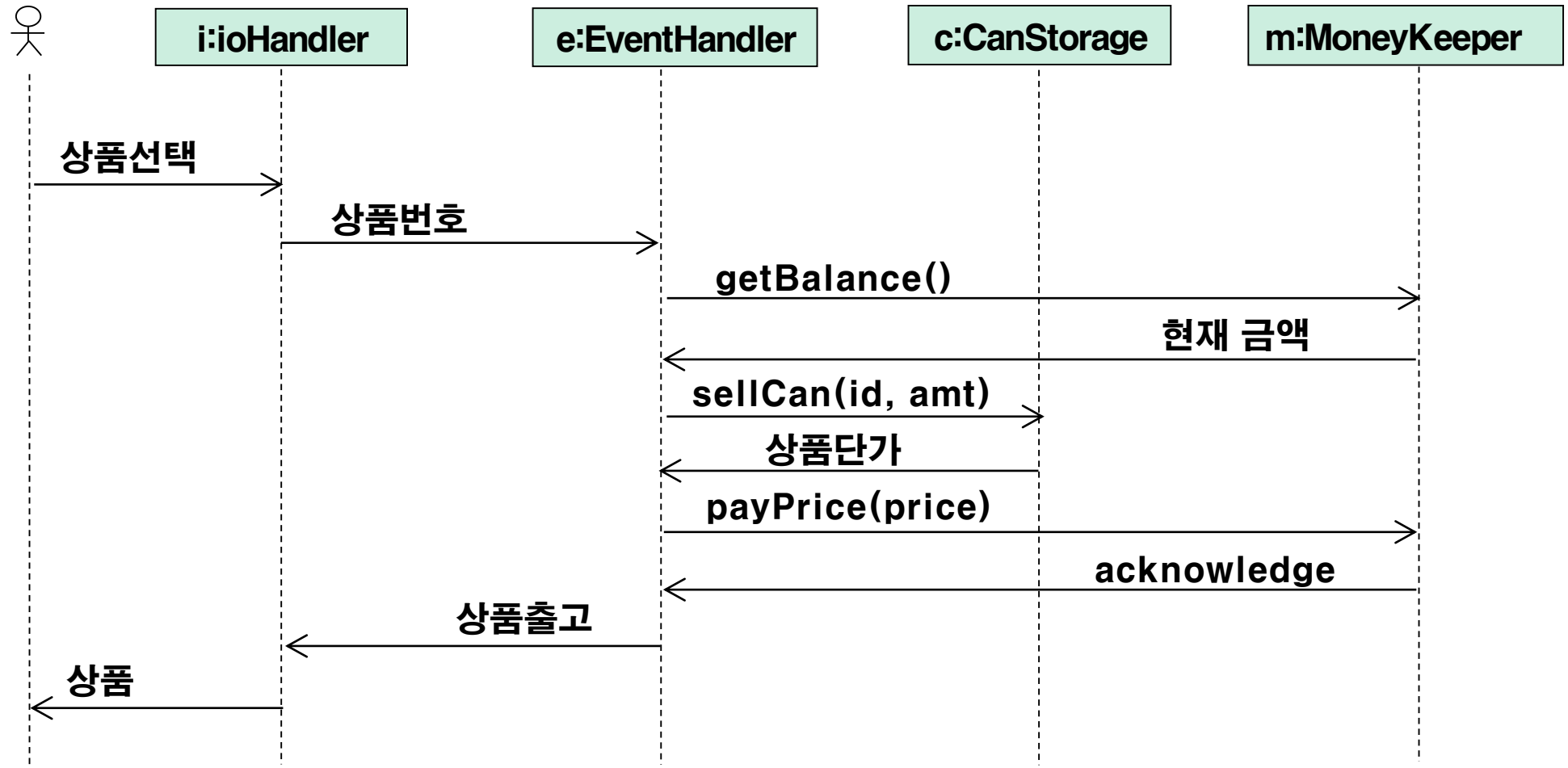
단계5: 객체가 작업을 수행하는 기간을 focus of control로 표시

단계6: 메시지가 시간 또는 공간상의 제약사항을 받는 경우에는
주석 또는 제약사항을 첨부

단계7: 복잡한 제어 흐름은 control operator로 표현

Sequence Diagram 예시

❑ 음료수 자판기의 “상품 선택” 사례



Communication Diagram(1)

- 커뮤니케이션 다이어그램의 중요한 구성 요소
  - 객체
  - link: 관련된 객체 사이를 연결하는 실선
    - 관계에 참여하는 객체의 역할을 표기 가능
    - 제약사항: 'local', 'parameter', 'global', ‘self’
  - 메시지
    - 채워진 삼각형 헤드를 갖는 화살표로 표기
    - 반환 값은 메시지의 일부로 처리
  - sequence number: 메시지의 발생 순서를 표현
    - Dewey 십진분류법에 따라 메시지 별로 고유번호를 부여. 단, 분기의 경우에는 동일 번호 부여 가능
    - 분기 -> 순번 앞에 조건절(예: [x>0])을 표기
    - 반복 -> 순번 앞에 *[i := 1..n]으로 표기

Communication Diagram(2)

- 커뮤니케이션 다이어그램의 부수적인 구성 요소
  - 주석 및 제약사항
  - packages

Communication Diagram(3)

❑ 커뮤니케이션 다이어그램 작성법

단계1: 모델링 대상을 선정

※ 대상: 유즈 케이스 또는 협동체

단계2: 참여 객체를 배치

※ 중요한 객체를 중앙에 배치하고, 관련도에 따라 가깝게 배치

단계3: 관련된 객체를 link로 연결

※ 연관 관계를 먼저 표현

※ 그 외의 관계는 필요에 따라 연결하되, 그 사유를 주석으로 설명

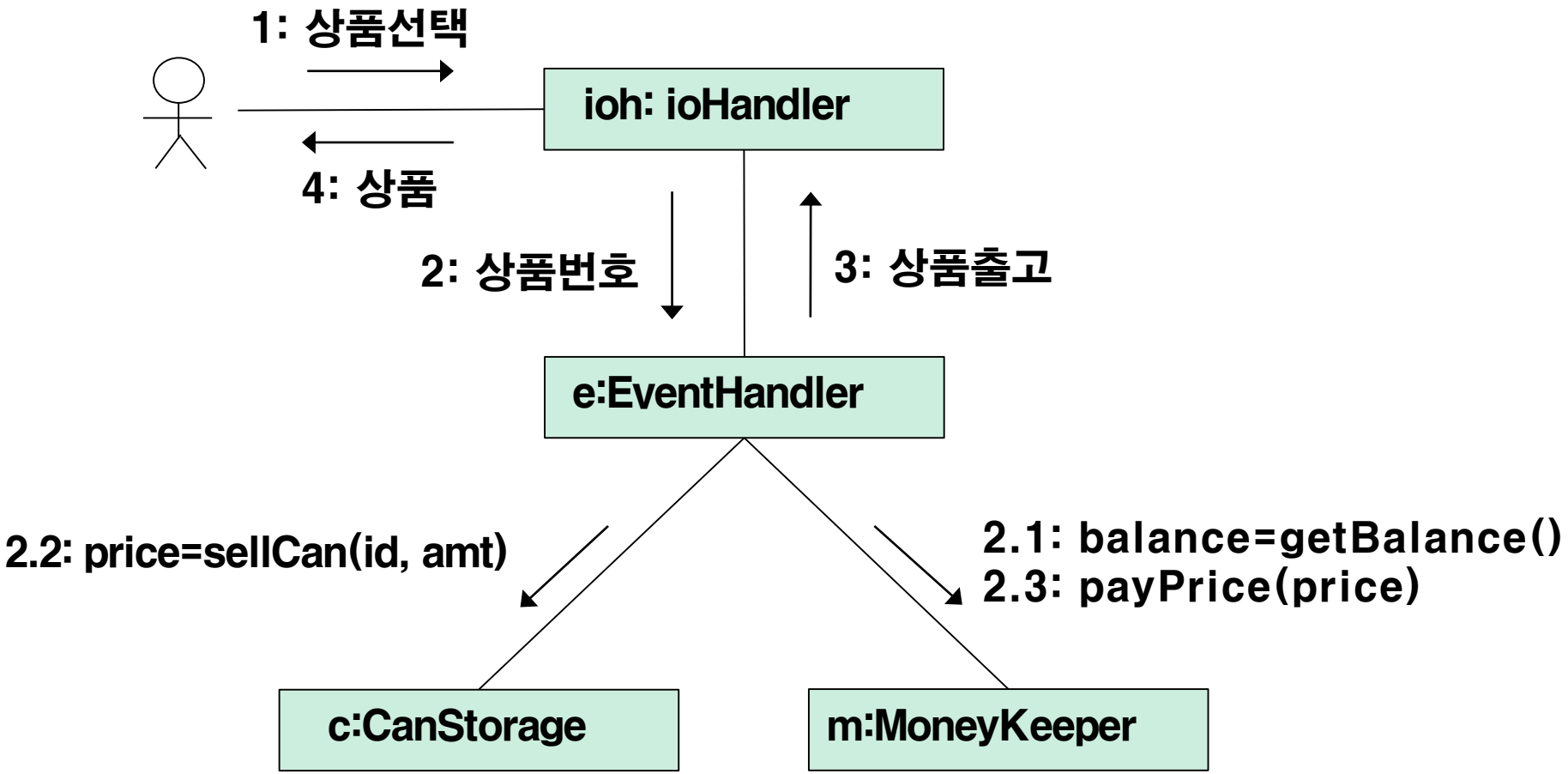
단계4: 상호작용을 시작하는 메시지부터 차례로 link에 메시지를 추가 하고, 고유한 sequence number 부여

※ 메시지는 송신 객체에서 수신 객체로!

단계5: 메시지가 시간 또는 공간상의 제약사항을 받는 경우에는 주석 또는 제약사항을 첨부

단계6: 복잡한 제어 흐름은 분기 또는 반복 표기법 사용

Communication Diagram 예시



State Diagram(1)

- 상태 다이어그램이란?
  - 상태 기계를 표현하는 다이어그램으로서, 시스템의 동적 측면을 표현
    - Reactive system, Embedded home security system
  - 상태와 상태 사이의 제어흐름을 강조
  - 이벤트의 발생 순서에 따라 객체가 어떻게 대응하는가를 기술

- 일반적인 용도
  - 시스템 전체의 개괄적인 동작을 기술
  - 객체의 생성에서 소멸까지의 생애를 표현
  - 유즈 케이스의 시나리오를 기술

State Diagram(2)

- 상태 다이어그램의 중요한 구성 요소
  - 단순/복합 상태
    - action state, activity state
    - 초기 상태, 종결 상태, history state
  - 상태 전이
  - 사건
  - 동작(action)

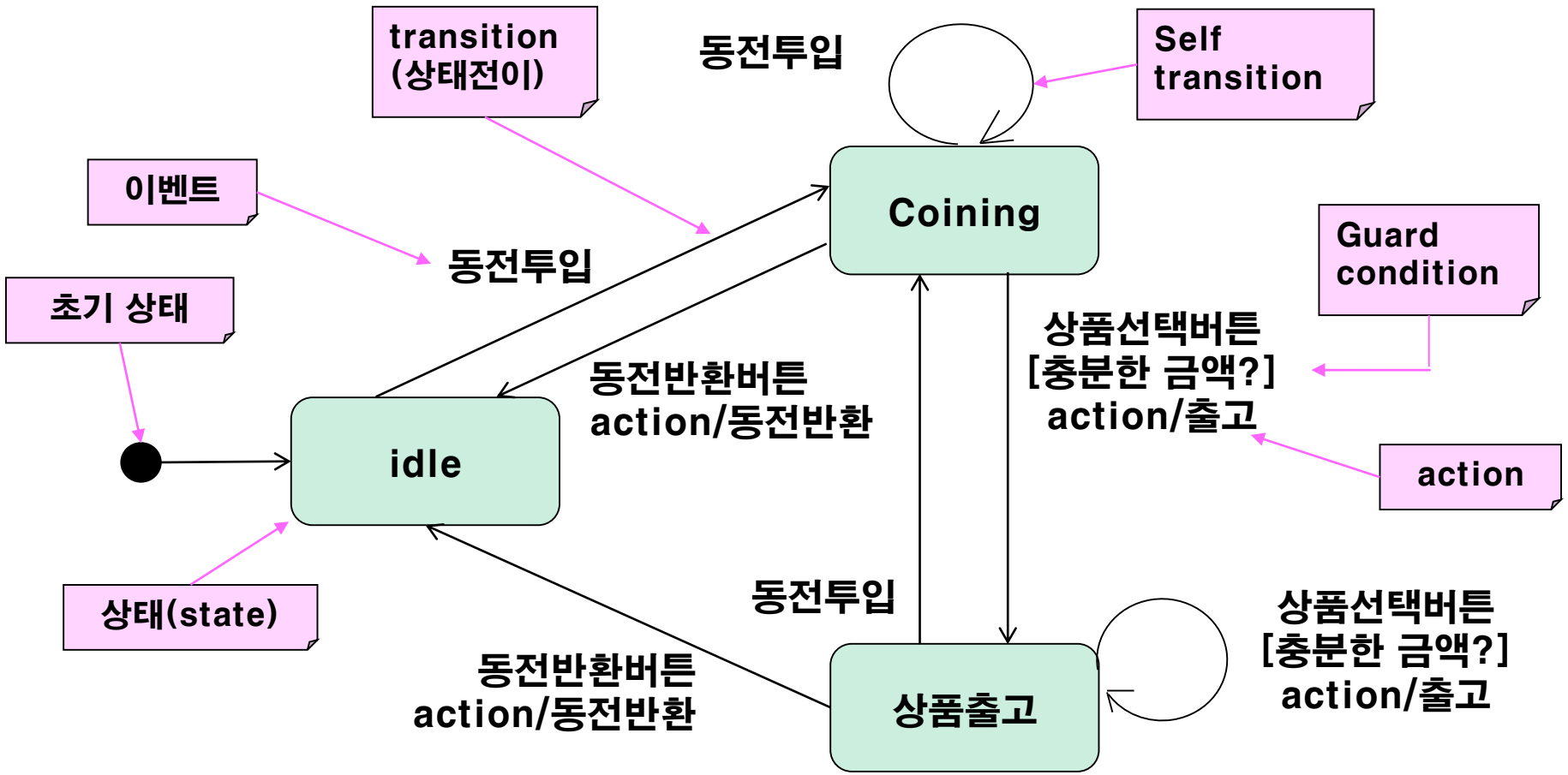
- 상태 다이어그램의 부수적인 구성 요소
  - 주석 및 제약사항

State Diagram(3)

❑ 상태 다이어그램 작성법(reactive object 모델링)

단계1: 모델링 대상 객체를 선정
단계2: 객체의 초기 상태 및 종결 상태를 선택
단계3: 객체가 생명주기 동안에 거쳐가는 상태를 결정
단계4: 결정된 상태의 부분 순서를 결정
단계5: 각각의 상태에 대해서 전이를 유발시키는 사건을 열거
단계6: 상태와 상태로부터의 전이에 수행 가능한 동작을 부가
단계7: 하위 상태, 분기, fork, join, history state를 사용하여 다이어그램을 단순한 형태로 변환
단계8: 가상의 시나리오를 적용하여 도달 불가능한 상태가 있는지 확인
단계9: 어떠한 사건이 발생해도 탈출 불가능한 상태가 있는지 확인
단계10: 예상되는 일련의 사건에 대해서 다이어그램이 제대로 작동하는지 확인

State Diagram 예시



Activity Diagram(1)

- Activity 다이어그램이란?
  - 객체 내부 또는 객체 사이에 이루어지는 작업의 순서를 표현하는 다이어그램
    - vertex -> activity or action
    - edge -> control flow
  - 본질적으로 flowchart와 동일하나, 병렬처리 가능
  - 상태기계의 일종이므로, 상태 기계의 모든 속성을 상속

- 일반적인 용도
  - 시스템의 기능을 기술
    - 오퍼레이션의 알고리즘
    - 유즈 케이스의 시나리오나 협동체의 개략적인 동작
  - 업무처리절차를 기술

Activity Diagram(2)

❑ Activity diagrams vs. Interaction diagrams

Activity diagram	Interaction diagram
시스템의 동적 측면을 기술	시스템의 동적 측면을 기술
Pert chart와 유사	Gantt chart와 유사
객체의 오퍼레이션이 어떻게 수행되는가에 초점	객체 사이에 주고 받는 메시지에 초점

※ 2개의 다이어그램이 개념적인 차이는 미미하지만,
시스템을 바라보는 관점의 차이는 엄청나게 크다!!

Activity Diagram(3)

- Activity 다이어그램의 중요한 구성 요소
  - Actions Node
    - atomic execution which can not be interrupted
    - 동작의 유형: 상태 변경, 메시지 교환, 수식 계산, …
  - Activity Node
    - actions으로 구성되는 비원자적인 작업
    - Entry/Exit action 포함 가능
  - Flows
    - 초기상태에서 시작하여 종결상태에서 종료
    - Triggerless transition: 어떤 node에서의 작업이 끝나면 trigger 없이 바로 다음으로 진행
    - Sequential transition: 제어흐름을 변경시키지 않으면 순차적으로 진행

Activity Diagram(4)

- 분기
  - 1개의 입력, 2개 이상의 출력을 갖는 다이아몬드로 표기
  - guard expressions이 겹치면 안됨
  - 분기와 반복자(iterator)를 사용하여 작업의 반복이 가능
- fork & join
  - fork: 하나의 흐름을 2개 이상의 병렬처리 흐름으로 분리
  - join: 2개 이상의 병렬처리 흐름을 1개의 흐름으로 병합
  - 병렬처리 흐름 안에 있는 Activity에서 signal를 주고 받음으로써 소통 가능

▪ 객체
- 객체를 생성/소멸/변경 시키는 action에 연결하여 배치 가능
- object flow = flow 상에 배치된 객체
- 객체의 상태 및 속성 값 -> 대괄호 안에 표기

Activity Diagram(5)

- Activity 다이어그램의 부수적인 구성 요소
  - Swim lane
    - 정의: Activity node를 그룹으로 묶어 배치하는 영역
    - 여러 부서에 걸쳐 수행되는 업무의 처리절차를 기술할 때 편리하게 사용 가능
    - 각각의 swim lane은 고유한 이름을 가지며, 1개 이상의 클래스로 구현 가능
    - Flow는 swim lane를 건너갈 수 있으나, activity는 불가
  - Expansion region
    - 다수의 대상에 대한 반복 작업을 기술하는 표기법
    - 점선으로 된 둥근 사각형으로 반복 작업 영역을 표시
    - 처리 대상 -> input array, 처리 결과 -> output array
  - 주석 및 제약사항

Activity Diagram(6)

❑ Activity 다이어그램 작성법1 (operation 모델링)

단계1: 대상 오퍼레이션에 관한 다음의 자료를 수집

※ 매개변수

※ 대상 오퍼레이션이 속한 클래스 및 주변 클래스의 속성

단계2: 초기상태에서의 사전 조건, 종결상태에서의 사후 조건, 실행 간의 불변 조건(invariants)을 파악

단계3: 초기상태부터 시작하여 시간이 지남에 따라 수행해야 할 작업 (activity & action)을 파악하여 다이어그램에 상태로 표현

단계4: 선택 및 반복 구조를 나타낼 필요가 있으면 분기를 사용

단계5: 병렬 처리 구조를 나타낼 필요가 있으면 fork와 join을 사용

Activity Diagram(7)

❑ Activity 다이어그램 작성법2 (workflow 모델링)

단계1: 모델링의 대상인 업무수행절차를 선정
※ 모든 업무를 대상으로 하는 것은 불가능

단계2: 업무 수행 주체를 파악하고, 각각에 대한 swimlane을 만든다

단계3: 초기 상태에서의 사전 조건과 종결 상태에서의 사후 조건을 파악

단계4: 초기 상태부터 시작하여 수행할 작업을 기술하고,
이를 다이어그램에 상태로 표현

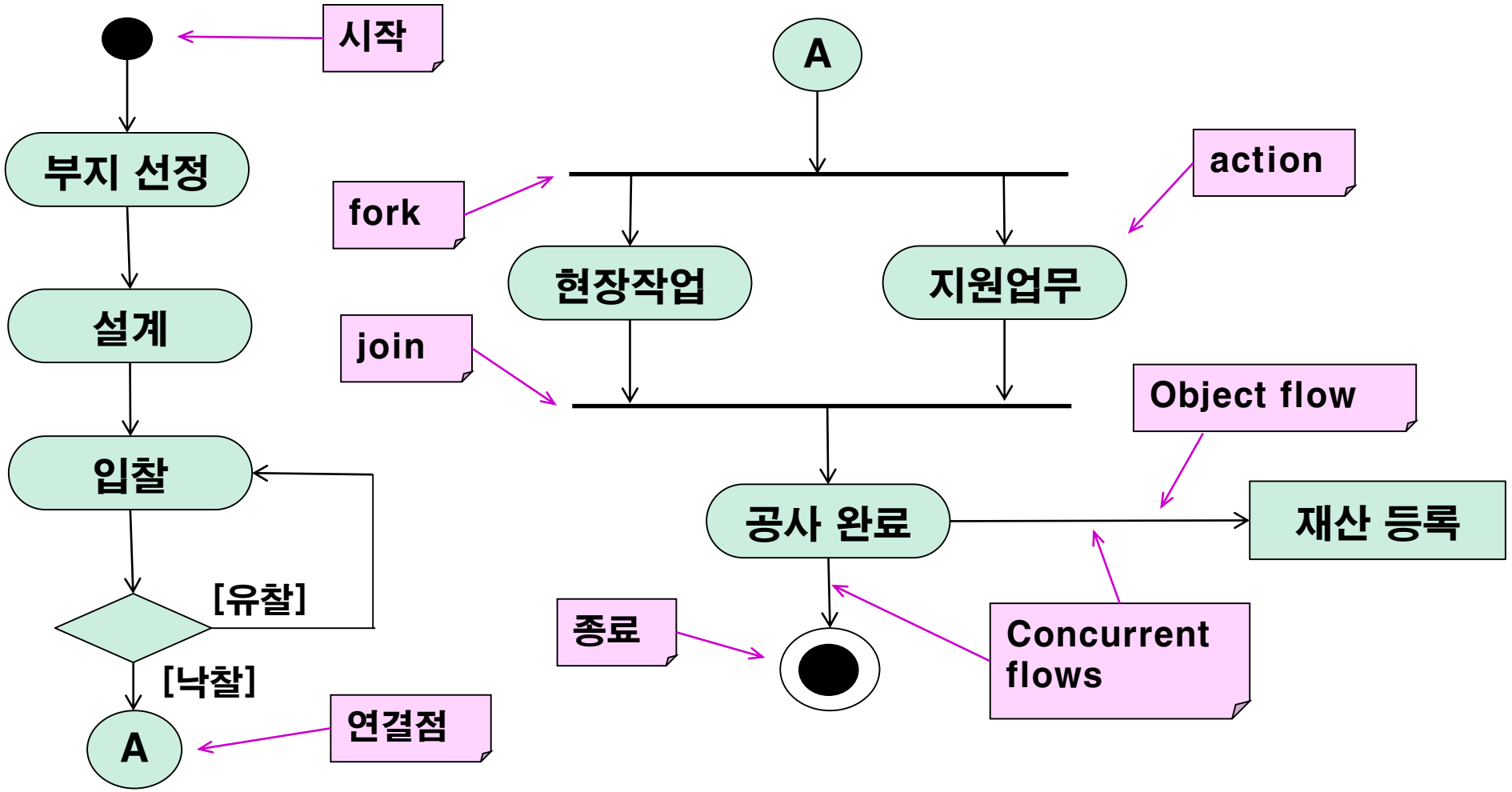
단계5: 반복해서 나타나는 작업은 전체를 1개의 상태로 나타내고,
상세 내역은 별도의 다이어그램으로 표현

단계6: 상태 사이의 제어흐름을 상태 전이로 나타낸다.

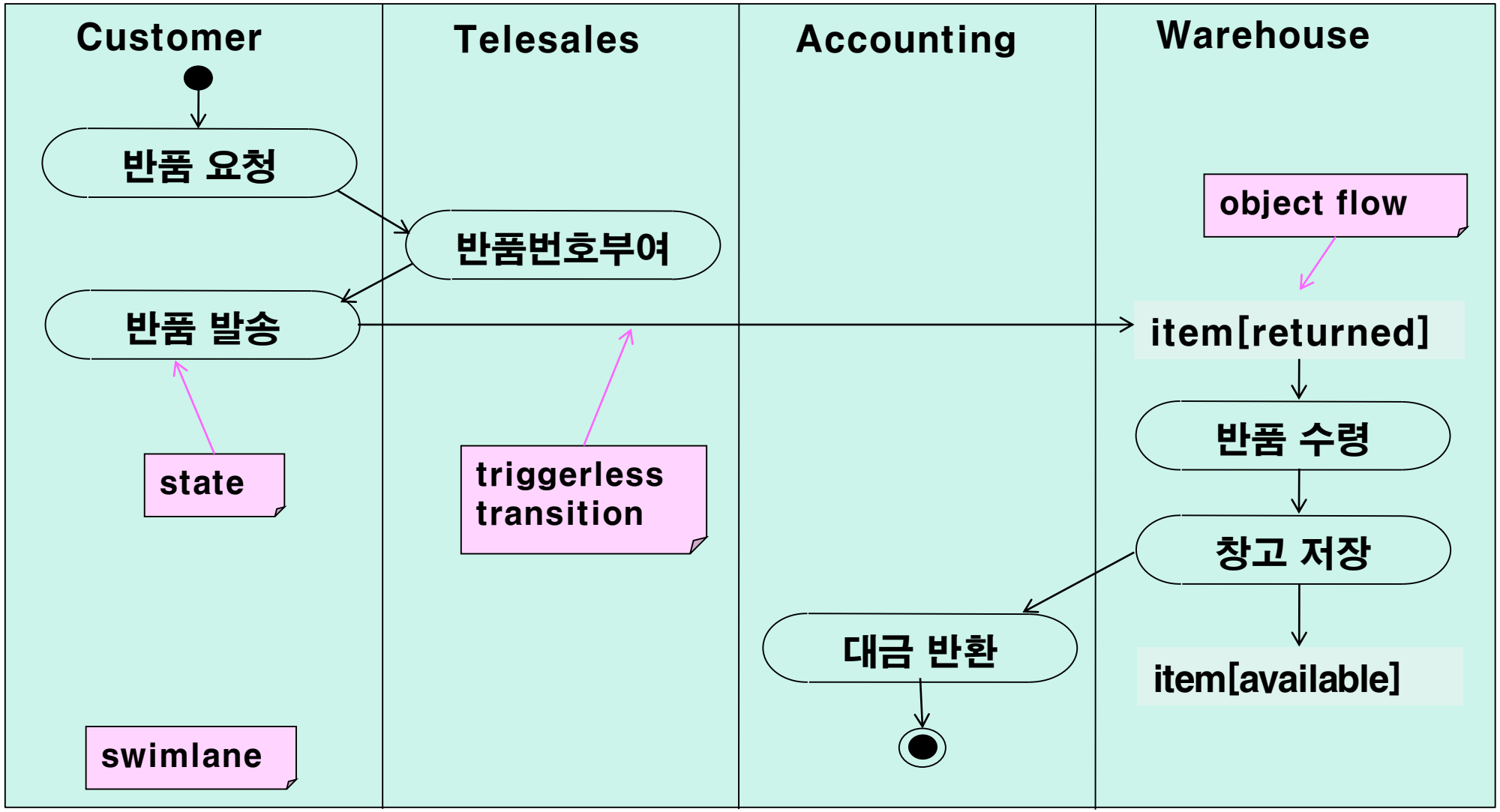
단계7: 업무절차에 포함된 중요한 객체가 있다면 object flow로 표현

Activity Diagram 예시(1)

❑ 주택건설 절차



Activity Diagram 예시(2)



Deployment Diagram(1)

- 배치 다이어그램이란?
  - 시스템을 실행시키는 노드와 이에 탑재된 산출물의 형상을 표현
    -> H/W 배치 및 네트워크 구성을 기술
  - 시스템을 구성하는 노드에 초점을 둔 클래스 다이어그램의 특수한 형태

- 일반적인 용도
  - 다음과 같은 시스템의 정적인 배치 뷰를 모델링
    - Embedded System
    - Client/Server System
    - Fully Distributed System

Deployment Diagram(2)

- 배치 다이어그램의 중요한 구성 요소
  - 노드(node)
    - 실행 시에 존재하는 h/w 장치로서, 메모리 및 CPU를 장착
    - 노드는 1개 이상의 산출물(*distribution unit*)을 탑재
    - 노드는 탑재된 객체 또는 컴포넌트를 실행
  - OS가 통상적으로 다루지 않는 특별한 장치
  - 관계: 노드 및 장치 사이의 관계
    - 종속
    - 연관: 노드/장치 사이의 물리적인 연결 관계를 표현
      ex> Ethernet, serial line, shared bus 등
  - 노드에 탑재된 산출물(예: 컴포넌트)

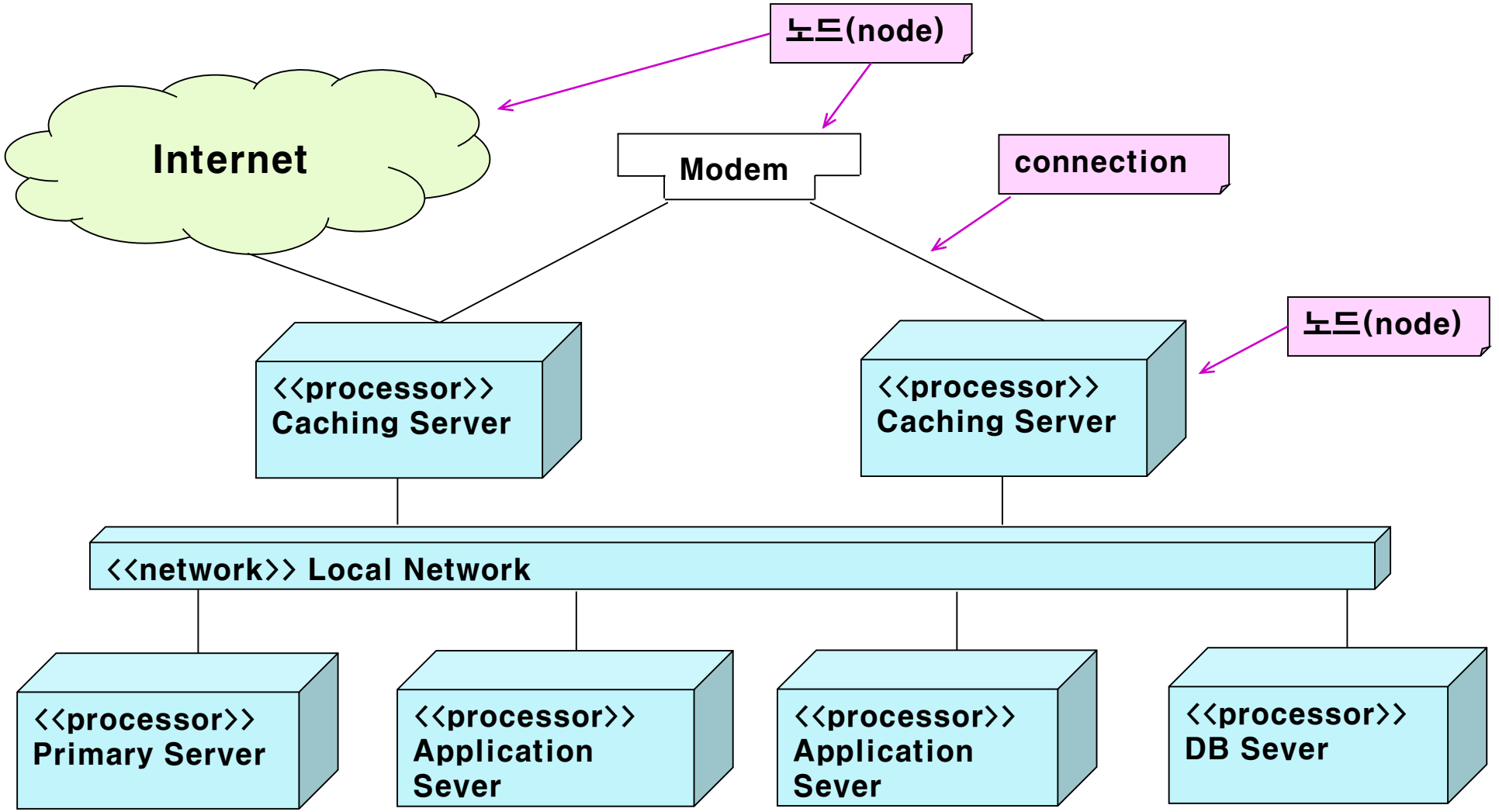
- 배치 다이어그램의 부수적인 구성 요소
  - 패키지 or 서브시스템 -> 구성 요소를 그룹화
  - 주석 및 제약사항

Deployment Diagram(3)

❑ 배치 다이어그램 작성법

단계1: 시스템에 존재하는 장치와 노드를 파악하여 다이어그램에 포함
※ 특별한 장치는 별도로 제작한 아이콘으로 표시
단계2: 노드 및 장치 사이의 연결 관계를 기술
단계3: 각각의 노드에 대해서 탑재된 산출물을 기술
단계4: 복잡한 장치 및 노드는 별도의 다이어그램으로 독립

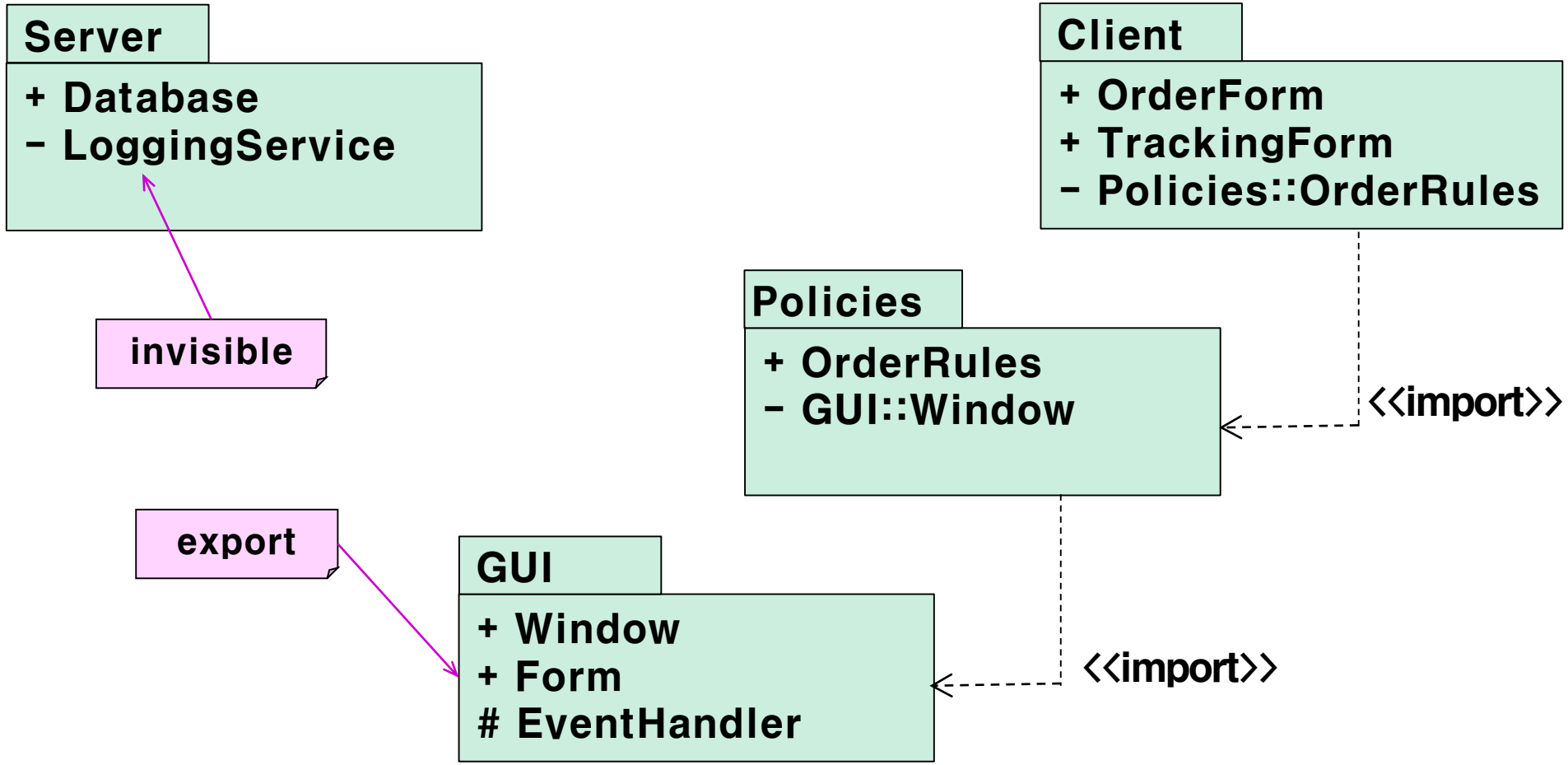
Deployment Diagram 예시



Package Diagram

- 패키지 다이어그램이란?
  - 패키지와 이들 사이의 관계를 표현
  - 패키지
    - 시스템의 구성 요소를 관련성에 따라 분류하여 계층 구조를 형성하는 도구
    - 패키지는 클래스, 인터페이스, 컴포넌트, 협력체, 노드, 유즈 케이스, 다이어그램, 다른 패키지를 원소로 가짐
  - 패키지 사이의 관계
    - <<export>> : 패키지의 원소를 외부에서 사용 가능토록 함
    - <<import>> : 다른 패키지의 원소를 자신의 것처럼 사용 가능
- 일반적인 용도
  - 시스템 구성 요소의 그룹을 표현
  - 시스템의 구조(Architecture)를 모델링

Package Diagram 예시

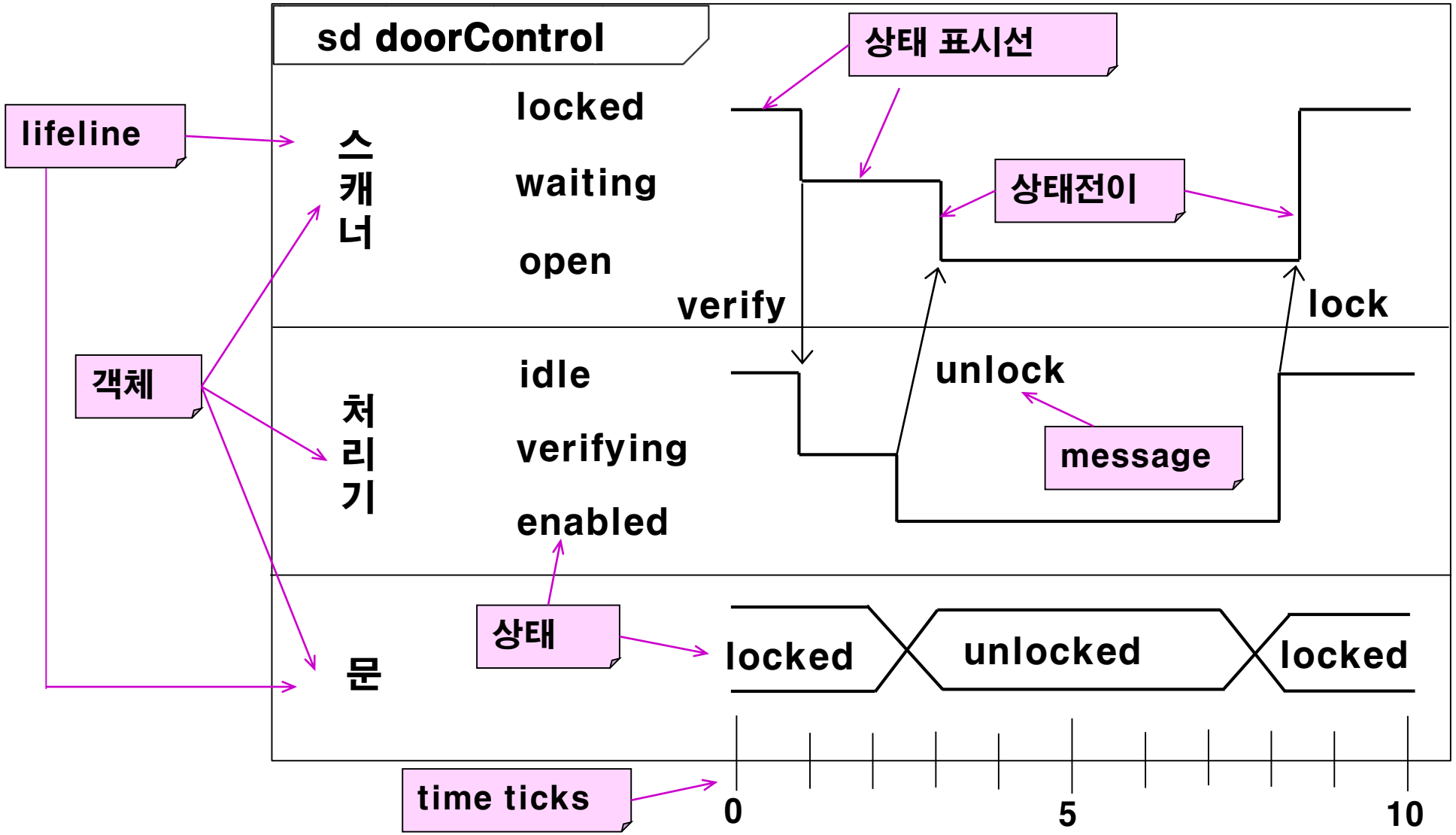


Timing Diagram

- 타이밍 다이어그램이란?
  - 시퀀스 다이어그램의 특수한 형태
    - 생명선(life-line)을 각각의 칸으로 표기
    - 시간 축은 좌측에서 우측으로 증가하도록 설정
    - 시간 경과에 따른 상태 변화를 추가적으로 표현

- 일반적인 용도
  - 실시간 시스템의 모델링에 유용

Timing Diagram 예시



Interaction Overview Diagram

- 기본 개념
  - Activity 다이어그램에 부분적으로 시퀀스 다이어그램을 혼합시킨 다이어그램
    - 상위 수준의 제어 흐름은 activity 다이어그램을 사용
    - 필요한 경우에는 동작의 세부 내역을 시퀀스 다이어그램으로 표현
  - 특별한 경우에만 사용

Interaction Overview Diagram 예시