UML과 소프트웨어개발

강사명: 손재현 -넥스트리소프트
-jhsohn@nextree.co.kr

과정 개요

❑ 교육 목표 & 특징

- UML2.x의 이해
- 유스케이스 작성
- 객체모델링 이해
- UML2.x의 다양한 다이어그램 이해 및 활용
- 모델링 도구 사용법 습득

강의 요구 기술

- **본 강의는 아래 기술에 대한 이해를 필요로 합니다.**
  - 객체지향 언어(Java) 기초
  - 개발프로세스 이해

교육 일정표

❑ 교육은 매 회 4 시간씩 총 5회에 걸쳐 진행합니다.

1 일차	2 일차	3 일차	4 일차	5 일차
- UML 개요 UML 소개 UML 다이어그램분류	- 유스케이스 유스케이스 개요 유스케이스 다이어그램 유스케이스 명세	- 분석모델 I 개념모델 구조 다이어그램 (클래스, 객체 ...)	- 분석모델II 컴포넌트 식별 인터페이스 식별 행위 다이어그램 (액티비티, 시퀀스 ...)	- 설계모델 컴포넌트 동적 설계 컴포넌트 내부 설계

1. 컴포넌트(인터페이스) 식별
2. 행위 다이어그램

1. 컴포넌트(인터페이스) 식별

ONE STEP AHEAD

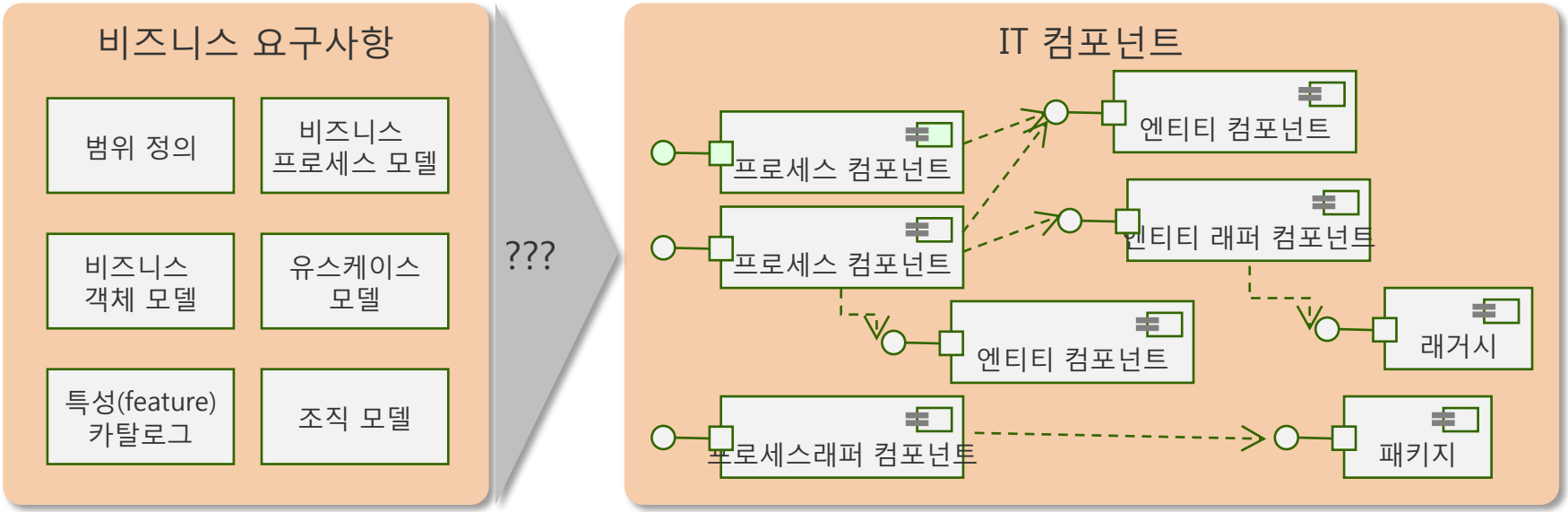
- The Bridge [CBDI Forum]

들어가기 전에...

- **SOA 기초**
  - ESOA 유연한 애플리케이션을 위한 기초를 제공
  - CBSE 기반 아키텍처는 B2B, B2C, 워크플로우, 레거시 래핑을 해결
- **Bridge**
  - 비즈니스 민첩성과 효과성을 위해 비즈니스 요구와 IT 구현 간의 추적성 필요
  - 비즈니스 요구 정의와 구현 간의 매듭없는 진행이 필요함
  - 비즈니스 지향 시스템 모델링(<-> 시스템 지향 비즈니스 모델링)
    - 비즈니스 모델과 IT 시스템 컴포넌트 간의 일대일 매핑 보장
    - 비즈니스로부터 코드로의 추적성 제공
    - 보다 정확한 비즈니스 요구 파악

개요 [1/2]

- 개발 프로세스 상의 임피던스 미스매치
- 문제점
  - 어떤 소프트웨어가 비즈니스의 민첩성과 응답성에 영향을 주는 지 알수 없음
  - Time-to-Market을 위한 IT 시스템의 생산성에 어떤 영향을 주는 지 알 수 없음
  - IT 시스템이 비즈니스 요구를 수용하지 못하는 경우 있음



개요 [2/2]

- **문제점(계속)**
  - 비즈니스 명세가 IT 구조와 알고리즘으로 건너가는 동안 정보를 유지
  - IT 알고리즘은 코드 모듈 속에 잠겨버려서 비즈니스 구조를 잃어버림
  - 실질적이고 효과적인 추적성은 매우 어려움
- **Bridge**
  - 서비스, 비즈니스 구조와 규칙으로부터 IT 컴포넌트와 알고리즘으로 일대일 매핑
  - 비즈니스 민첩성을 위한 기초
  - 서비스를 제공하는 컴포넌트에 대한 적절한 식별이 IT 민첩성의 핵심
  - 비즈니스 프로세스나 규칙의 변화가 하나 이상의 특정 컴포넌트로 추적되어야.
  - 새로운 비즈니스 구조와 알고리즘은 분석되고, 그 결과 요구 활동에 정의된 구조가 IT 설계 모델의 출발점이 됨

요구사항

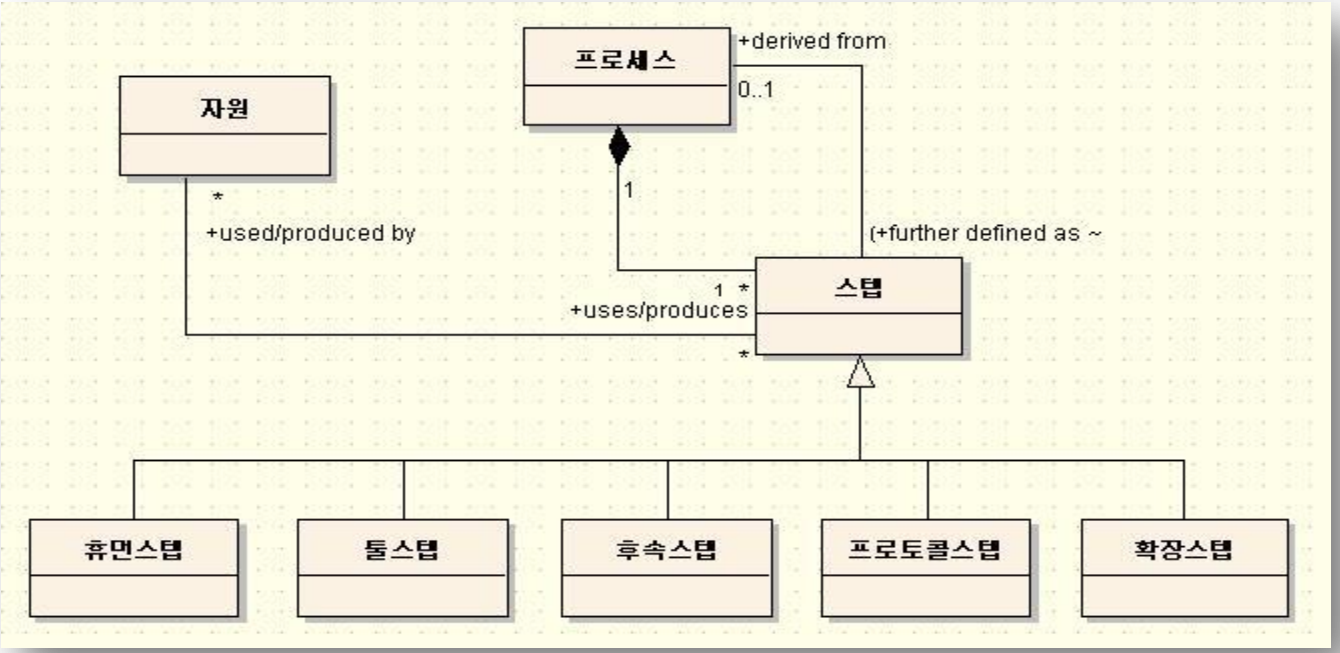
- **요구사항 파악의 목표**
  - 계획된 시스템이 제공해야 하는 서비스 정의
  - 서비스 수행 방법 정의
  - 제시되어야 할 비즈니스 특성 정의
- **요구사항 파악은 높은 수준의 범위 설정 활동이 아니라 시스템 개발의 일부분**
- **요구사항 파악은 IT 시스템 설계는 포함되지 않음**
- **요구사항 파악을 위한 접근방법**
  - 범위, 비전, 목표 정의
  - 비즈니스 프로세스와 이 프로세스에 의해 생성되고 사용되는 자원이라는 관점에서 비즈니스 요구 정의("자원"은 비즈니스 객체 또는 엔티티)
  - IT 시스템으로 구현할 "비즈니스 요소(business element)" 정의

비즈니스 요소

- "비즈니스 요소 (Business Element)"
  - 프로세스, 자원, 조직
  - 이 세 가지는 상호 의존적
    - 조직은 명세된 프로세스를 수행
    - 프로세스는 수행할 조직과 입출력으로서의 자원을 요구
    - 자원은 조직의 자산(asset)이며, 프로세스에 의해 생산되거나 입력 됨
- 비공식 정의
  - 비즈니스 구성원에게 중요한 것으로 간주되는 식별 가능한 비즈니스 "조각"
  - 비즈니스 요소는 프로세스 지향이거나 자원 지향
  - 예, "판매 주문 서비스", "주문서 관리", "고객", "예약", "구매 주문"
- 비즈니스 요소의 특징
  - 경험을 기반으로 요구사항 단계 초기에 식별됨
  - 프로세스와 자원 모델을 기반으로 후보를 식별할 수 있는 경험법칙 존재

프로세스와 자원 [1/2]

- **비즈니스 프로세스**
  - 각 프로세스는 하나 이상의 스텝(step)으로 구성됨
  - 프로세스는 더 작게 쪼개질 수 있음
  - 프로세스는 서비스를 제공하고, 단계는 서비스 제공 방법을 정의
- **자원 또는 엔티티는 스텝에서 사용되고 생산됨**
- **스텝은 유스케이스 모델에서 텍스트로 서술될 수 있음.**



프로세스와 자원 [2/2]

- 스텝의 종류
  - 후속 스텝(immediate step)
  - 확장 스텝(extended step)
  - 휴먼 스텝(human step)
  - 툴 스텝(tool step)
  - 프로토콜 스텝(protocol step)
- 자원은 “비즈니스 객체 모델”이나 “엔티티 모델”에서 파악됨
  - 예, “고객”, “주소”, “주문 항목”
- 프로세스와 자원 모델이 있으면, 프로젝트 범위 안에 있는 “비즈니스 요소”의 집합을 정의할 수 있음

비즈니스 요소 분석

- **전통적인 자원과 프로세스 모델로부터 끌어 낼 수 있는 비즈니스 요소**
  - 자원 비즈니스 요소 (Resource Business Element)
    - 비즈니스에서 중요한 자원
  - 서비스 비즈니스 요소(Service Business Element)
    - 비즈니스 에 의해 제공되고 비즈니스 프로세스에 의해 구현되는 연관 서비스의 집합
  - 인도 비즈니스 요소(Delivery Business Element)
    - 조직과 관련된 서비스 집합을 제공하는 서비스와 자원 비즈니스 그룹에 대한 정의된 그룹

자원 비즈니스 요소 [1/4]

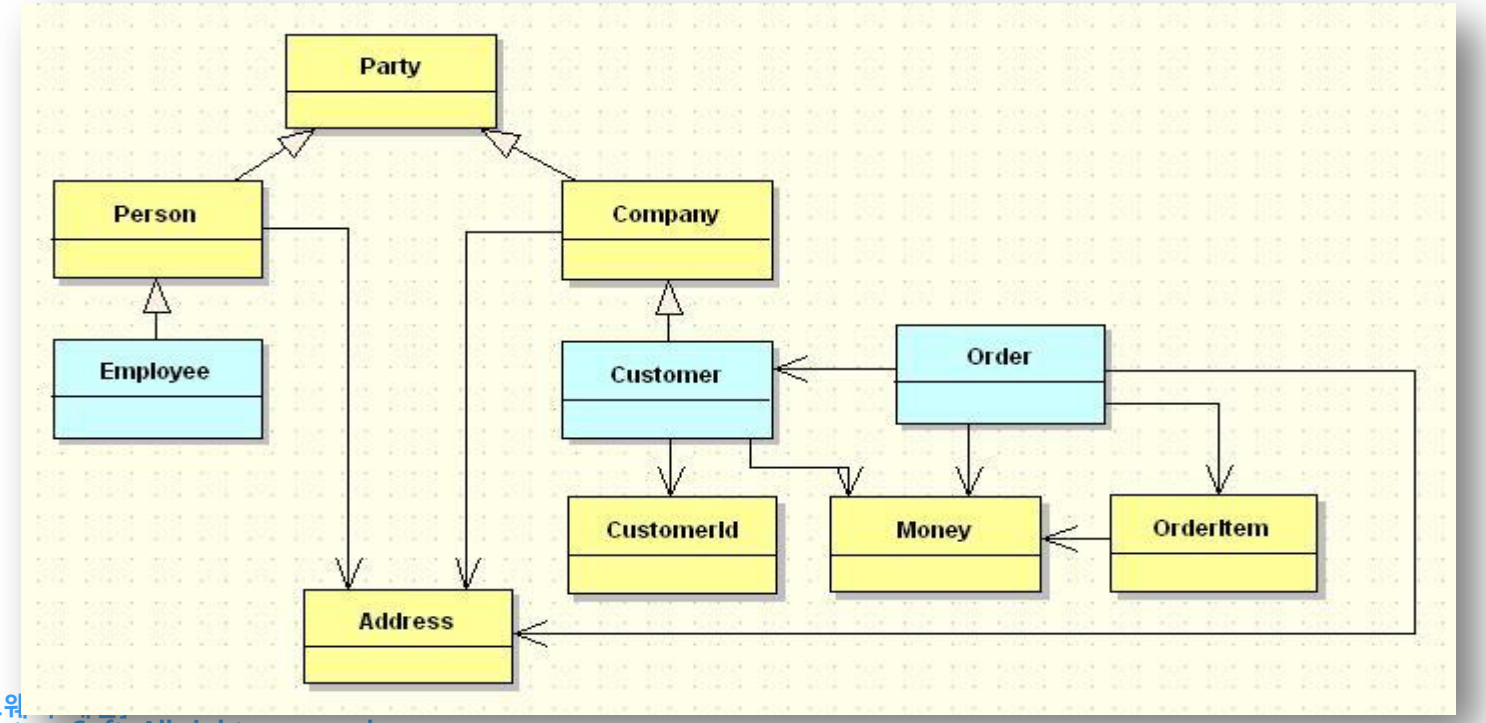
❑ 자원 비즈니스 요소(Resource Business Element)

- 자원 그룹을 캡슐화 함
- RBE는 비즈니스 도메인에서 "실재하고 독립적인(real and independent)" 특별한 자원 주위에 초점이 맞추어진 자원 그룹
- 실재적인(real)
  - 주제 분야 전문가에 의해 사용되고 이해되는 실재 자원
  - 구체적임
  - 올바른 예, "고객", "주소", "주문 항목"
  - 잘못된 예, "법률 주체", "위치", "집합 구성원"
- 독립적인(independent)
  - 주제 분야 전문가가 그것이 소속된 곳을 말하지 않고 말할 수 있는 것
  - 범위가 이해되는 것으로 비즈니스 개체이거나 조직 개체
  - 올바른 예, "고객"
  - 잘못된 예, "(고객) 주소", "(공급자) 주소"
- 실재적이고 독립적인 자원을 "초점(focus)" 자원이라 하고, 나머지는 "보조(auxiliary)" 자원이라 함

자원 비즈니스 요소 [2/4]

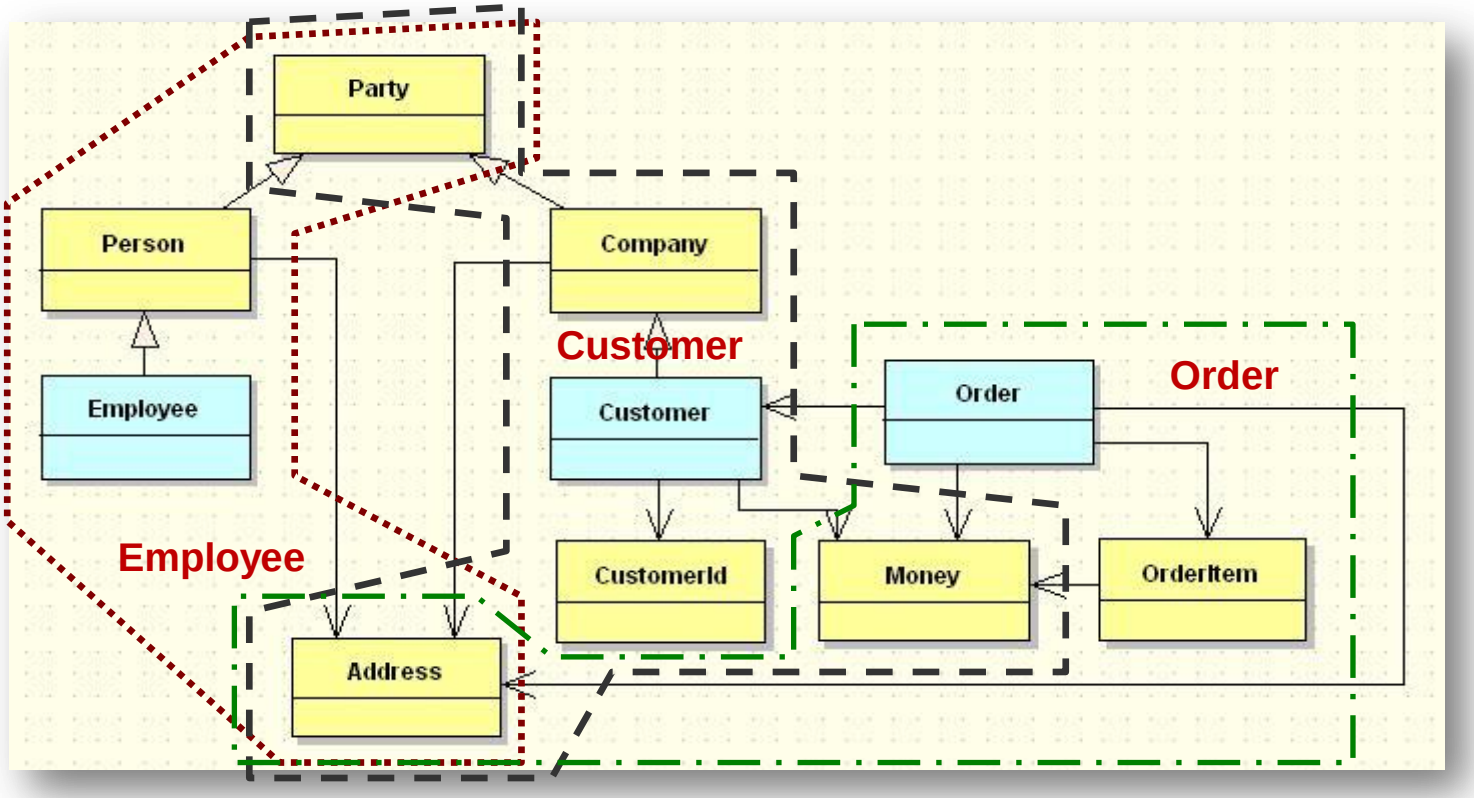
❑ RBE 식별 경험 법칙

- 비즈니스 자원 모델에서 각 자원이 실재적인지 독립적인지 확인
- 초점 자원으로부터 관계를 따라서 보조 자원으로 가다가 또 다른 초점 자원을 만나면 멈춘다. 마주친 보조 자원에 주목한다. 이러한 보조 자원은 초점 자원과 더하여 "초점 그룹"을 형성함
- RBE 예제는 비즈니스 자원 모델(비즈니스 객체 모델)로부터 출발함



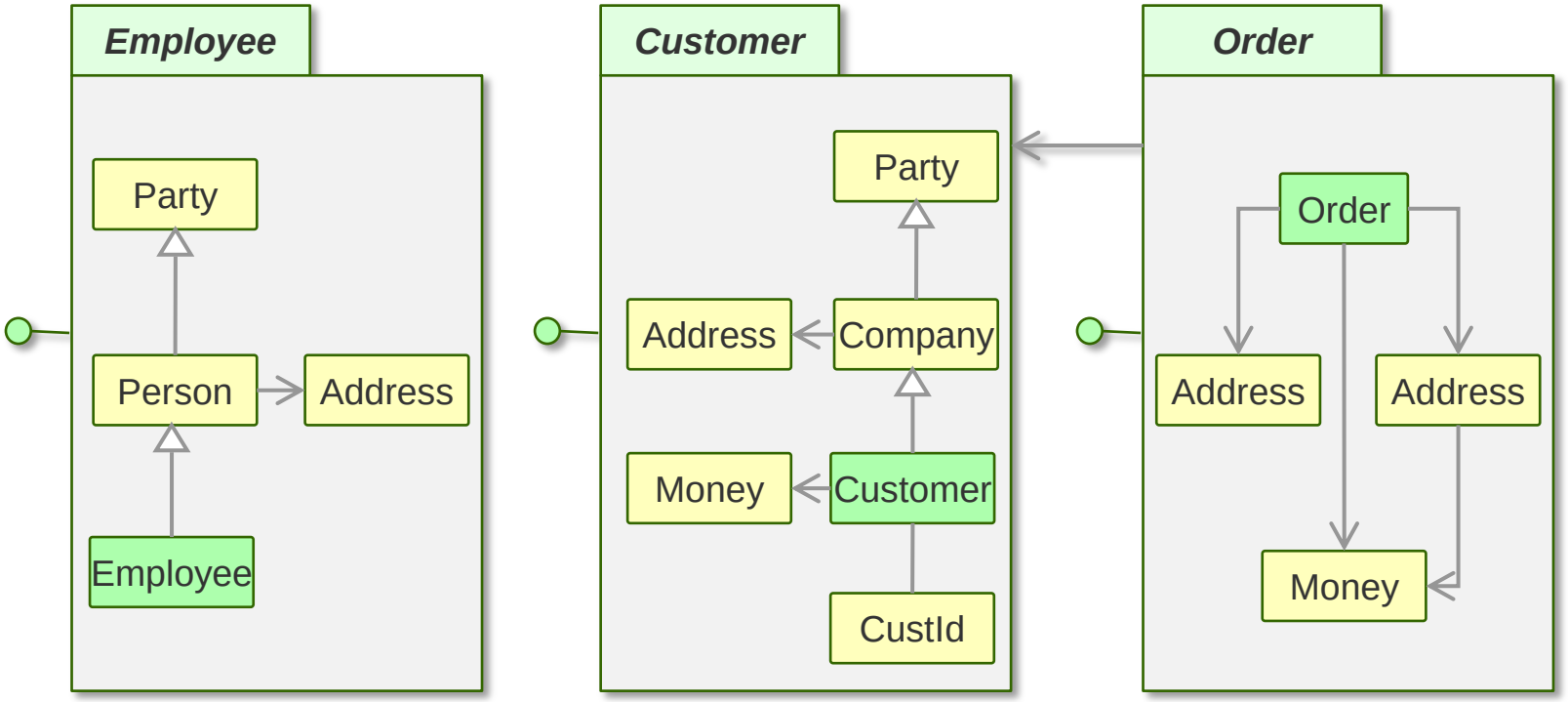
자원 비즈니스 요소 [3/4]

- Employee, Customer, Order를 초점 자원으로 가정
- 보조 자원을 묶기 위해 식별 경험 법칙을 따라가면 다음과 같은 세 개의 초점 그룹에 이름



자원 비즈니스 요소 [4/4]

- 정리하면 세 개의 자원 비즈니스 요소(RBE)가 나옴
- 원래 자원 모델에서 정보를 잃지 않고 아주 단순화 함
- 여러 RBE에 나타나는 보조 자원도 있음
- 초점 자원은 하나의 RBE에만 나타남



서비스 비즈니스 요소 [1/3]

❑ 서비스 비즈니스 요소(Service Business Element)

- 서비스 비즈니스 요소는 "후속(immediate)" 스텝의 집합
- 스텝을 묶는 이유는 해당 조직이 RBE 주변의 프로세스(스텝)의 집합에 대한 책임이 있음
- 이러한 스텝들은 응집된 서비스를 제공함
- 비즈니스 단위에 대한 높은 응집도와 낮은 결합도의 모듈화 원리를 적용

❑ SBE 식별 경험법칙

- 비즈니스 프로세스 모델에서 가장 높은 후속 스텝을 식별 -> 서비스
- 각 스텝에 "동사형"의 이름을 부여, 명사형 이름이 적절한 것은 RBE
- RBE를 기준으로 묶음, 각 그룹은 RBE에 대한 CRUD 를 포함
- 비즈니스 프로세스 모델에서 다음 수준의 후속 스텝을 반복함,

❑ 정리

- 각 SBE는 조직으로 매핑
- 프로세스 컴포넌트 구현을 위한 후보
- 각 후속 스텝의 이름에 있는 동사는 컴포넌트 인터페이스의 후보 오퍼레이션

서비스 비즈니스 요소 [2/3]

❑ 프로세스 모델로부터 식별된 서비스

- 고객 기록 변경
- 주문 처리
- 직원 퇴사
- 신규 고객 기록
- 기존 주문 변경
- 주문 취소
- 신규 직원 채용

서비스	후속(immediate) 스텝
고객기록 변경	고객 상세 정보 검증
	신용 체크를 통해 거래 한계 확인
	고객 상세 정보 기록
	타 고객과의 관계 확인 후 있으면 갱신
	표준 "고객 정보변경 통지"를 보냄
주문처리	제출 데이터 검증
	고객 검증
	주문 총액 계산
	신용 확인
	재고 할당
	필요 시 대기 주문 생성
	주문 생성
	주문 승인을 위한 전송

서비스 비즈니스 요소 [3/3]

❑ 결과 서비스 비즈니스 요소(SBE)

SBE	서비스	후속(immediate) 스텝
고객 서비스	고객기록 변경	고객 상세 정보 검증
		신용 체크를 통해 거래 한계 확인
		고객 상세 정보 기록
		타 고객과의 관계 확인 후 있으면 갱신
		표준 "고객 정보변경 통지"를 보냄
	신규고객 처리	...
주문 서비스	주문처리	제출 데이터 검증
		고객 검증
		주문 총액 계산
		신용 확인
		재고 할당
		필요 시 대기 주문 생성
		주문 생성
		주문 승인을 위한 전송
	기존주문 변경	...
	주문 취소	...
직원관리 서비스	직원 채용	
	직원 퇴사처리	...

인도 비즈니스 요소 [1/2]

❑ 개요

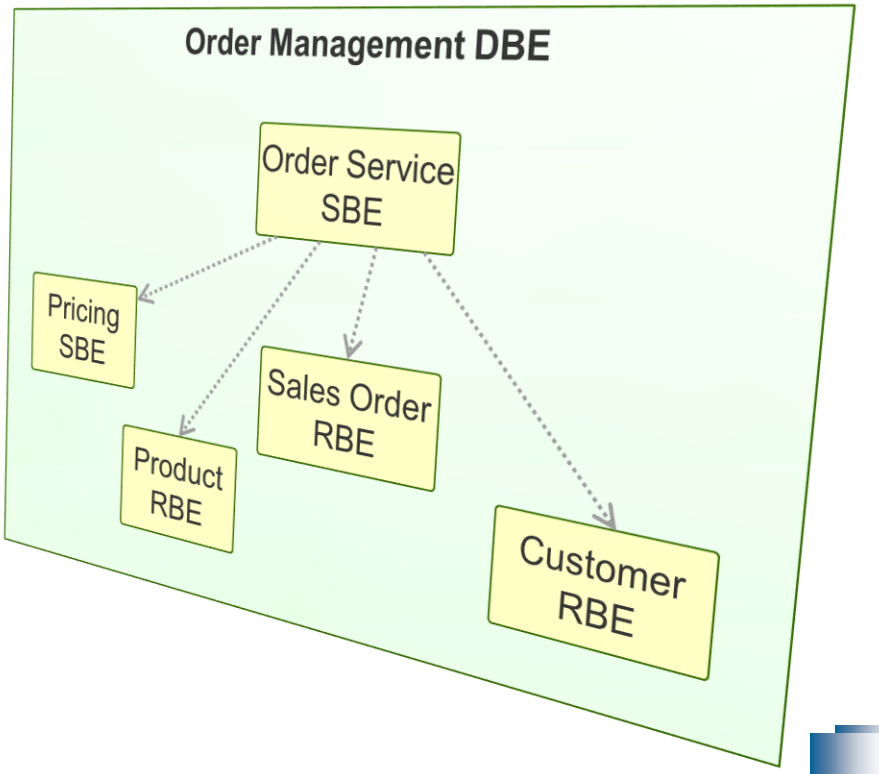
- 비즈니스에서 SBE와 RBE는 조직에 인도될 때 사용됨
- 예, 비즈니스에서 고객, 제품, 구매 주문 등의 RBE는 주문 서비스 SBE와 더불어 구매 주문 처리 부서의 주문관리 역량 인도에 사용
- 이러한 역량은 부서 뿐 아니라 고객, 공급자 등에도 정의된 서비스를 제공

❑ 인도 비즈니스 요소(Delivery Business Element)

- 묶음 자체가 제 3의 비즈니스 요소 = 인도 비즈니스 요소
- RBE 자체는 서비스와 프로세스를 가지지 못함
- SBE 자체로는 자원이 없음
- DBE는 가치를 생산하는 자산의 집합
- 정의 : DBE는 SBE와 RBE의 묶음으로 비즈니스 문제에 대한 솔루션을 제공, 즉 요청자에게 서비스를 제공함
- DBE는 부서나 큰 조직의 주요 책무와 일치함
  - 예, "구매 주문 관리" DBE에 포함된 서비스와 자원은 부서의 책임을 반영

인도 비즈니스 요소 [2/2]

- **DBE 식별**
  - 최상위 수준의 SBE를 식별함으로써 DBE가 식별됨
  - 각 최상위 SBE에 대해, BE의 집합을 식별
  - 결과의 집합이 DBE를 구성함
- **DBE의 예**
  - 두 개의 SBE와 세 개의 RBE 간의 의존 관계를 보여줌
  - 주문 관리 DBE



컴포넌트 매핑 [1/2]

❑ 매핑 개요

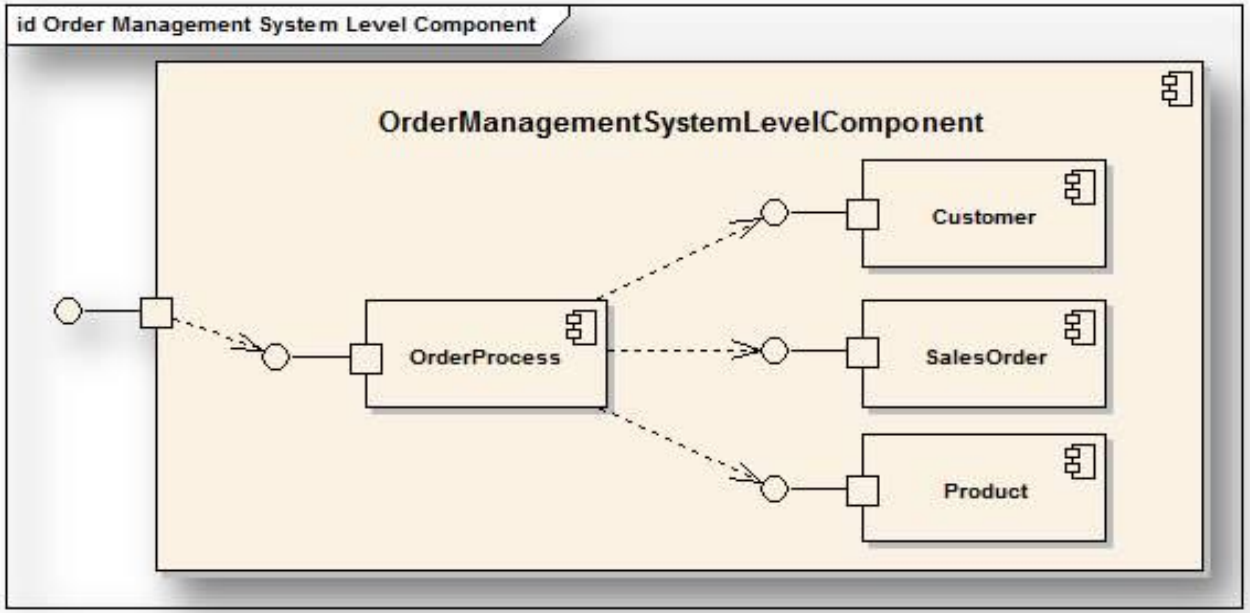
- BE는 주요 비즈니스 개념을 내포
- CBD는 IT 시스템에서 BE 구현을 위한 이상적인 구조화 개념 제공
- BE와 IT설계 모델의 변환이 필요 없음

❑ 매핑

- SBE에서 프로세스 컴포넌트
  - SBE에서 제공하는 각 후속(immediate) 스텝은 IT 시스템 설계에서 프로세스 컴포넌트의 오퍼레이션이 되므로, 후속 스텝은 시스템이 제공하는 서비스가 됨
- RBE에서 엔티티 컴포넌트
  - 직접 매핑이 되며, 초점 클래스와 보조 클래스가 있음
- DBE에서 시스템 수준 컴포넌트
  - DBE는 프로세스 컴포넌트와 엔티티 컴포넌트 간의 협업
  - 협업 자체도 컴포넌트로 볼 수 있음, 즉, "시스템-수준 컴포넌트"로 볼 수 있음

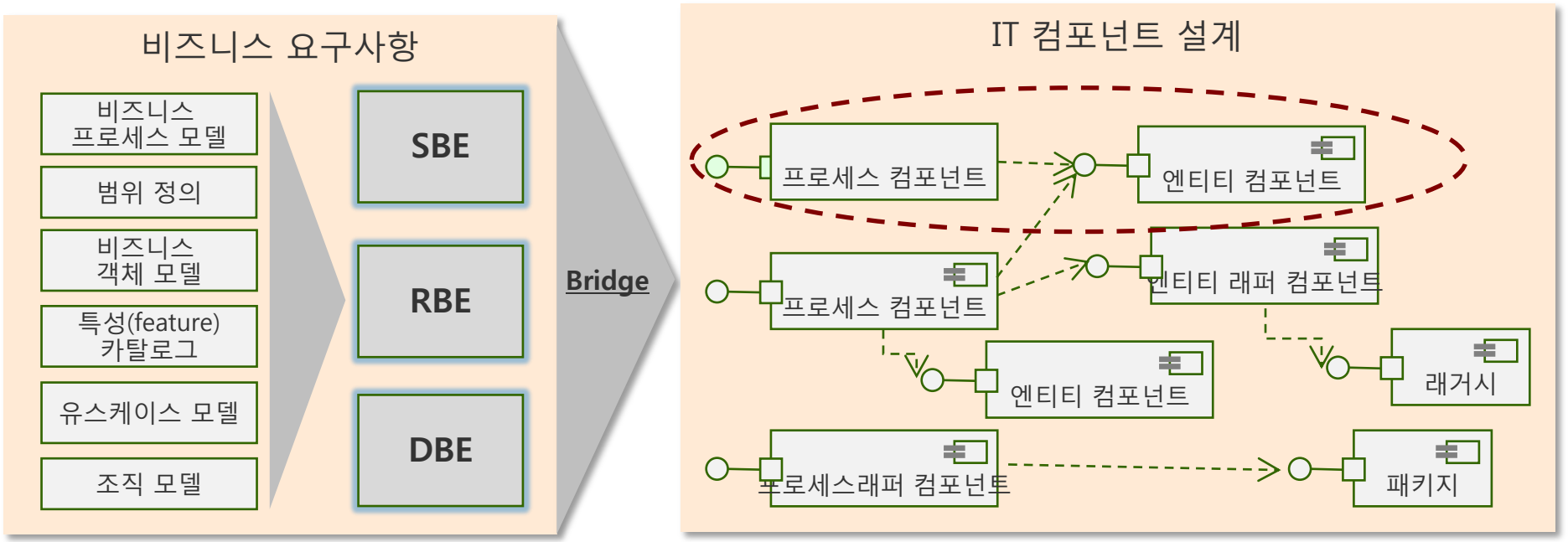
컴포넌트 매핑 [2/2]

- **MDA 뷰**
  - BE는 CIM(Computaion-Independent Model)을 구성하며 컴포넌트 PIM으로 전환
  - 구조 변경이 없으므로, 변환은 단순하며, 정보 유실이 없음
  - 이 후 IT 시스템 개발 주기내에서 컴포넌트는 지속적으로 개선됨
- **구매 주문 관리 시스템과 컴포넌트 들**

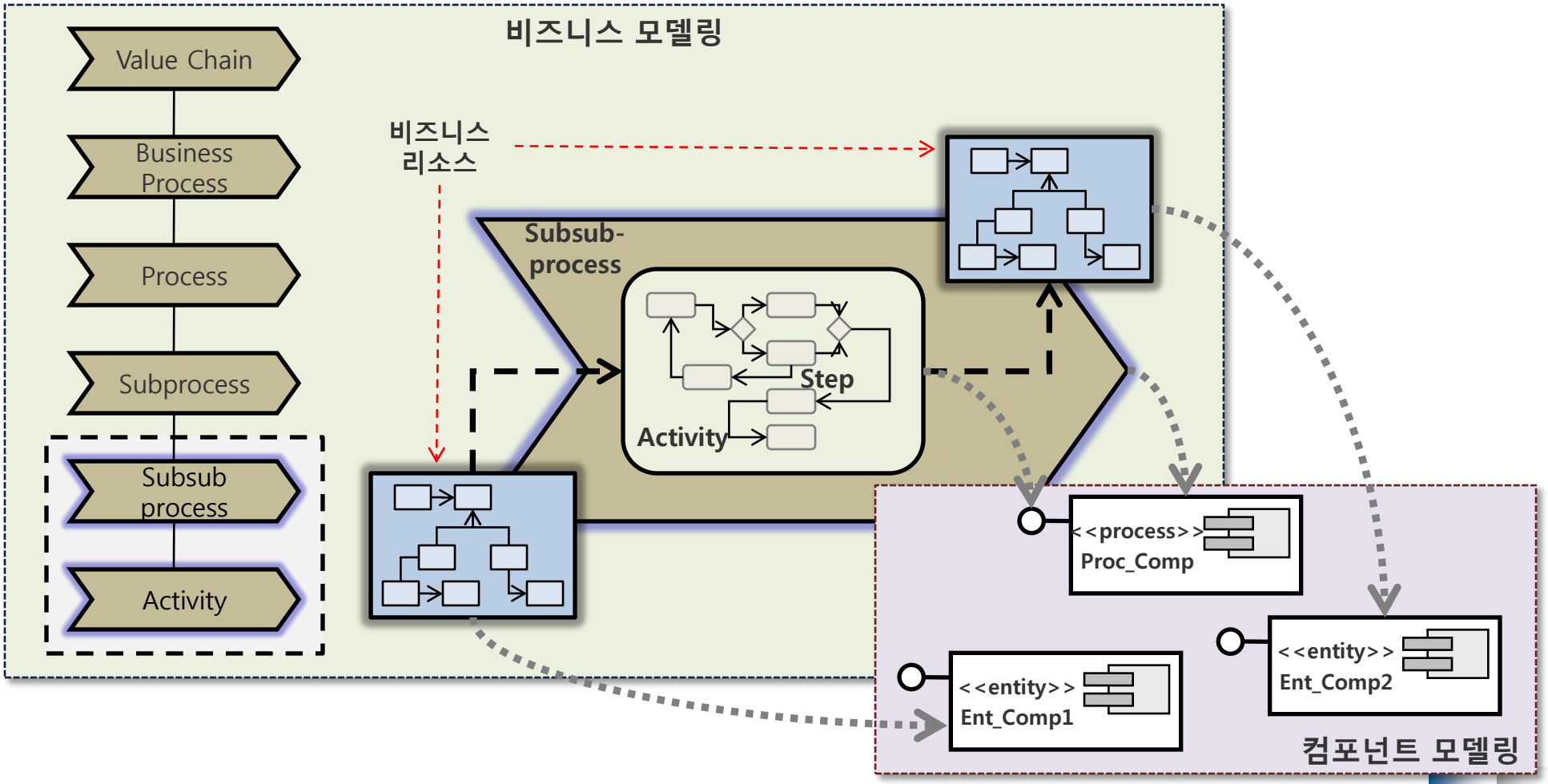


The Bridge [1/2]

- **비즈니스와 IT 시스템 간의 Bridge**
  - 서비스 비즈니스 요소(SBE) -> 프로세스 컴포넌트
  - 자원 비즈니스 요소(RBE) -> 엔티티 컴포넌트
  - 인도 비즈니스 요소(DBE) -> 시스템 수준 컴포넌트
- **IT 시스템과 비즈니스 컴포넌트 간의 추적성 확보**



❑ 비즈니스 모델링과 비즈니스 컴포넌트 매핑



요약

- SBE는 RBE나 다른 SBE를 조율함
- 프로세스 컴포넌트는 다른 컴포넌트를 조율함
- 각 컴포넌트는 명확한 서비스 제공(엔티티 컴포넌트의 엔티티 서비스)
- 각 컴포넌트는 자율적인 비즈니스 요소를 내포하며 자율적으로 설계되고 구축되므로, 비즈니스 요소가 비즈니스에서 재사용되듯이 컴포넌트 또한 그러함
- BE는 비즈니스 “모듈”이며, CBD는 BE를 일대일로 매핑할 수 있는 기술 기반을 제공함, 따라서 비즈니스를 지원하는 IT 시스템은 비즈니스를 따라 구조화 될 수 있음

2. 행위 다이어그램

- 액티비티 다이어그램
- 시퀀스 다이어그램

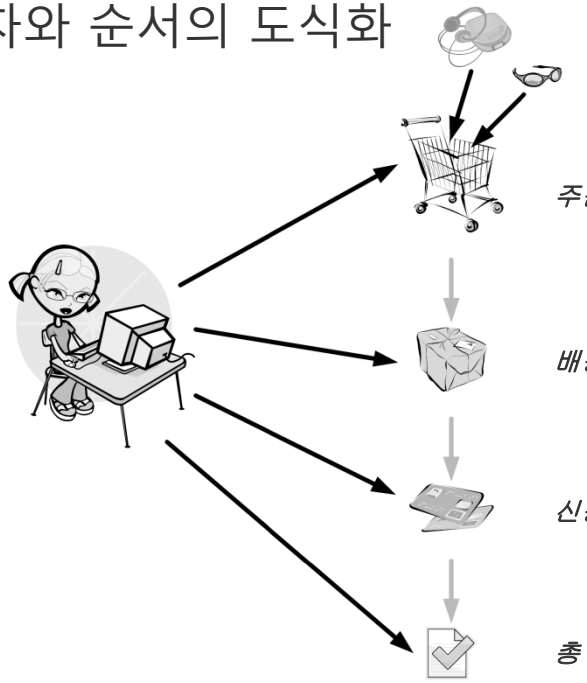
액티비티 다이어그램

- ❑ 개요
- ❑ 구성요소
- ❑ 작성 및 주의사항

개요 (1/4)

❑ 절차와 순서의 메타포*Metaphor*

- 물품구매 시 장바구니 기능
  - 주문할 품목을 추가하고 이 단계를 수행하여 품목을 계속 추가 가능
  - 배송방법, 목적지 주소, 보증 등에 대한 배송정보 작성
  - 신용카드 정보 및 청구서 주소 등의 결재정보 입력
  - 총 결재 금액 등을 포함한 주문완료 및 주문내역을 재검토
- 절차와 순서의 도식화



주문할 품목을 추가한다. 사용자는 이 단계를 수행하여 품목을 계속 추가할 수 있다.

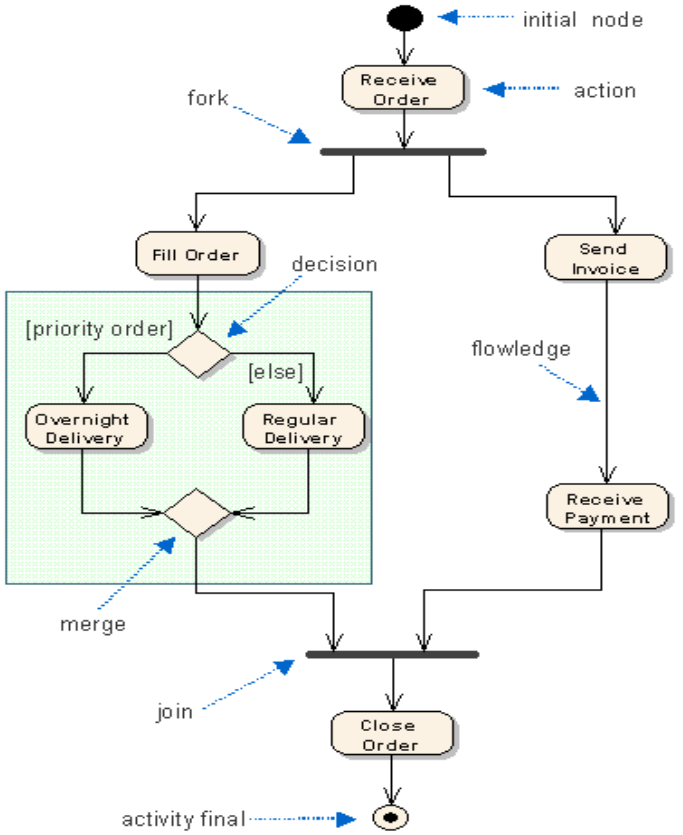
배송 방법, 목적지 주소, 보증 등에 대한 배송정보를 작성한다.

신용카드 정보 및 청구서 주소 등의 결재정보를 입력한다.

총 결재 금액 등을 포함한 주문 완료 및 주문내역을 재 검토한다.

개요 (2/4)

- 업무영역이나 시스템영역에서 다양하게 존재하는 각종 처리로직
- 조건에 따른 처리흐름을 순서에 입각하여 정의한 모델



개요 (3/4)

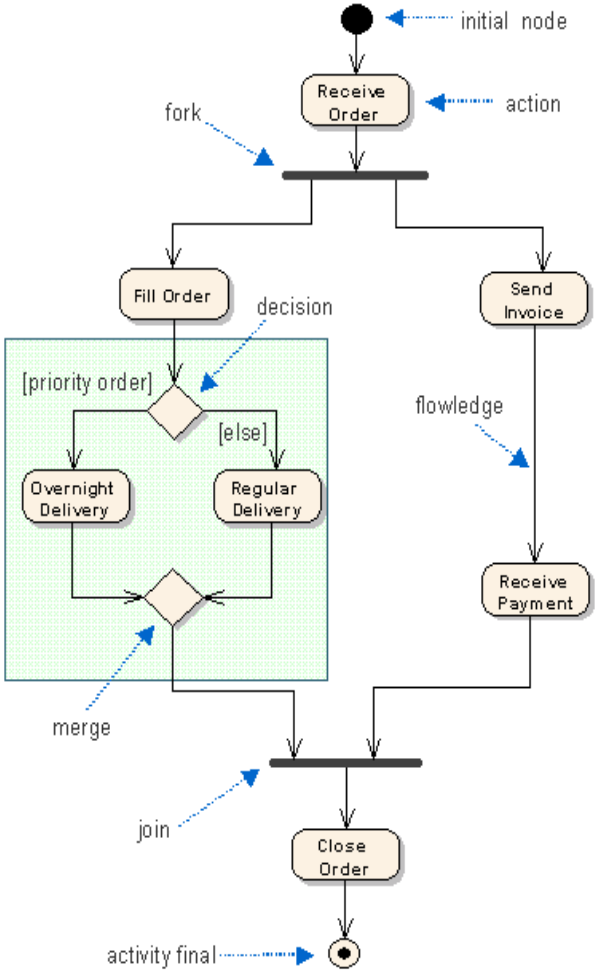
- **작성목적**
  - 대상에 상관없이 처리 순서를 표현하기 위해 작성
  - 비즈니스 프로세스를 정의
  - 프로그램 로직을 정의
  - 유스케이스 실현*Realization*
- **작성시기**
  - 업무 프로세스 정의 시점
  - 유스케이스 정의 작성 시점
  - 오퍼레이션 사양 정의 시점
  - 기타 처리흐름이나 처리절차가 필요한 시점

개요 (4/4)

- **UML 1.X**
  - 상태 다이어그램의 특별한 형태로서 액티비티 다이어그램을 취급
  - 절차적인 로직이나 비즈니스 프로세스, 워크 플로우를 표현하기 위한 기법
- **UML 2 : fork와 join이 반드시 매칭 되어야 한다는 룰 제거**
  - 상태의 변화보다 flow에 의해 액티비티 다이어그램을 이해
  - 새로운 표기법으로 표현
  - 타임 시그널(time signals), 수신 시그널(accept signals), 매개변수(parameter), 조인 명세서(join spec.), 서브다이어그램 표기(subdiagram rakes)
- **다차원을 나타내는 swim lanes는 파티션(partition)으로 불림**

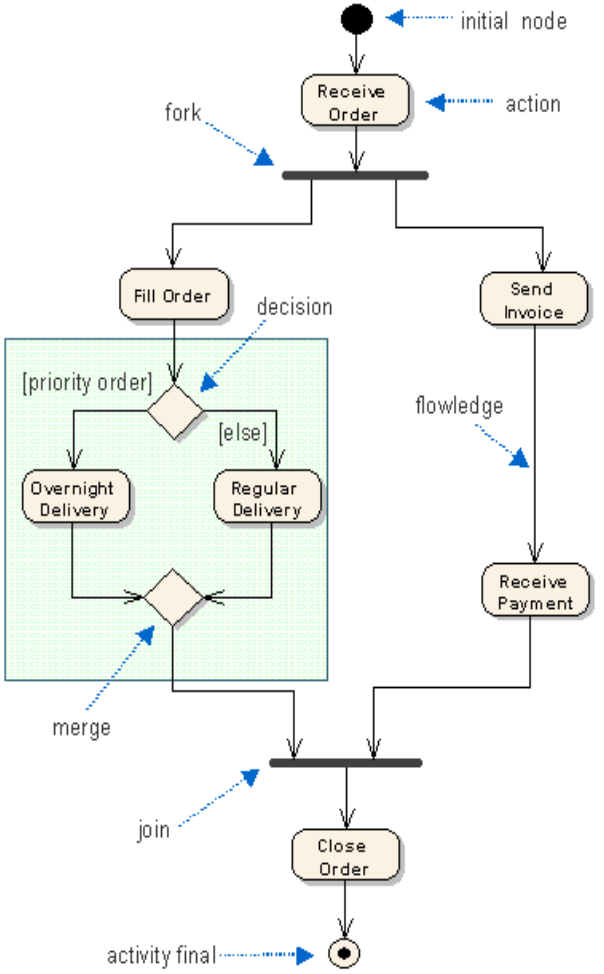
구성요소 (1/4)

- **액티비티 다이어그램의 구성요소**
  - UML 1: 액티비티(activity), 분기(branch)
  - UML 2: 액션(action), 결정(decision)
- **상호작용 다이어그램과 액티비티 다이어그램**
  - 상호작용 다이어그램은 객체에서 객체로의 제어흐름 강조
  - 액티비티 다이어그램은 액션에서 액션으로의 제어흐름 강조
- **액티비티 다이어그램과 플로우차트(Flowchart)**
  - 플로우차트: 순차처리만 표현
  - 액티비티 다이어그램: 병렬처리 역시 표현



구성요소 (2/4)

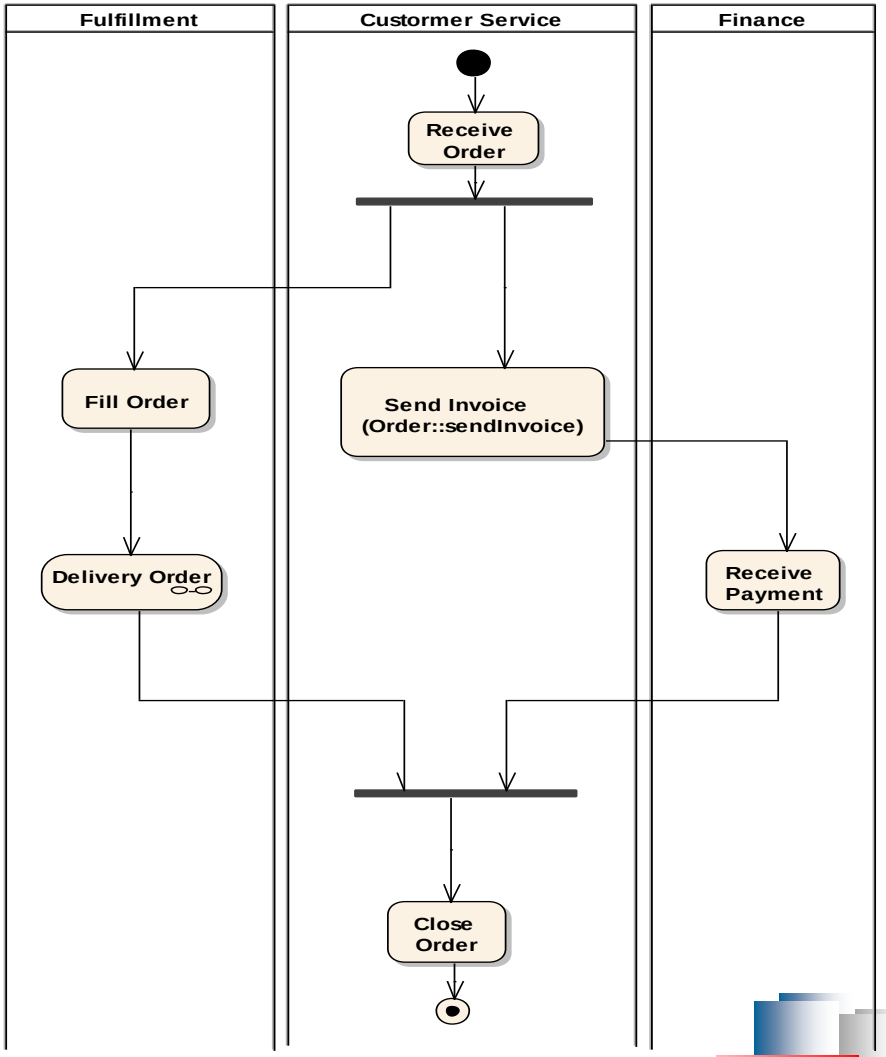
- **조건행동**
  - 결정(decision): 하나의 입력과 조건에 의한 다수의 출력전이로 구성
  - 병합(merge) : 여러 입력과 하나의 출력전이로 구성
- **병렬행동**
  - 포크(Fork): 하나의 입력전이와 여러 출력전이를 동시에 처리
  - 조인(Join): 여러 입력전이와 하나의 출력전이, 포크의 마무리



구성요소 (3/4)

❑ 파티션 *Partition*

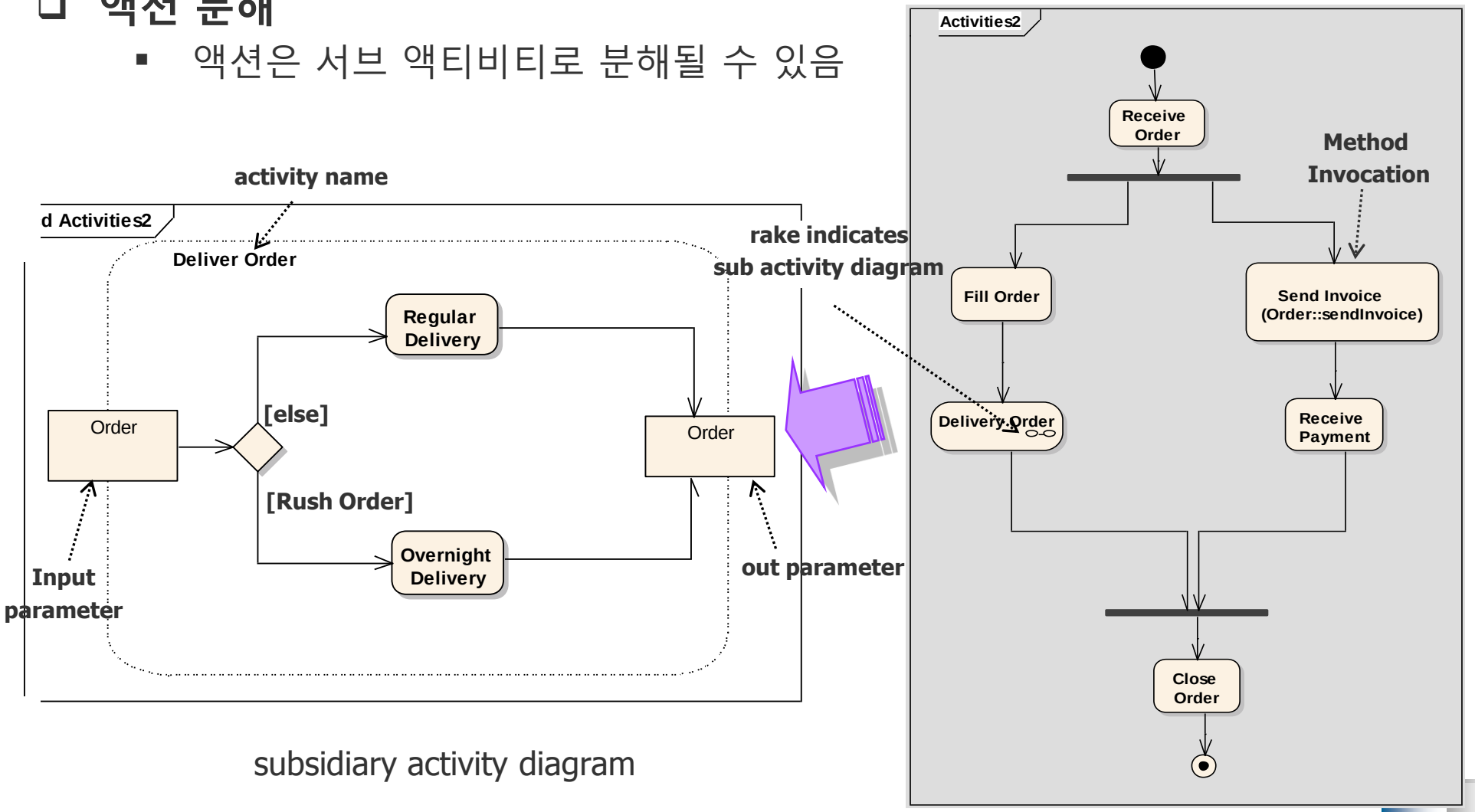
- 일반적 이슈
  - 액티비티 다이어그램은 어떠한 액션을 수행하는지 알 수 있지만 누가 수행하는지 모호
  - 프로그래밍 관점에서 어느 클래스가 어떤 액션을 수행하는지 그 책임을 알 수 없음
  - 비즈니스 프로세스 모델링 관점에서 각 액션을 책임지는 사람이나 부서를 알 수 없음
- 해결책
  - 각 액션의 책임 클래스 또는 사람을 표기
  - 파티션을 사용하여 책임성 부여
- 표기법
  - UML 1.X : swimlanes
  - UML 2 : partitions



구성요소 (4/4)

❑ 액션 분해

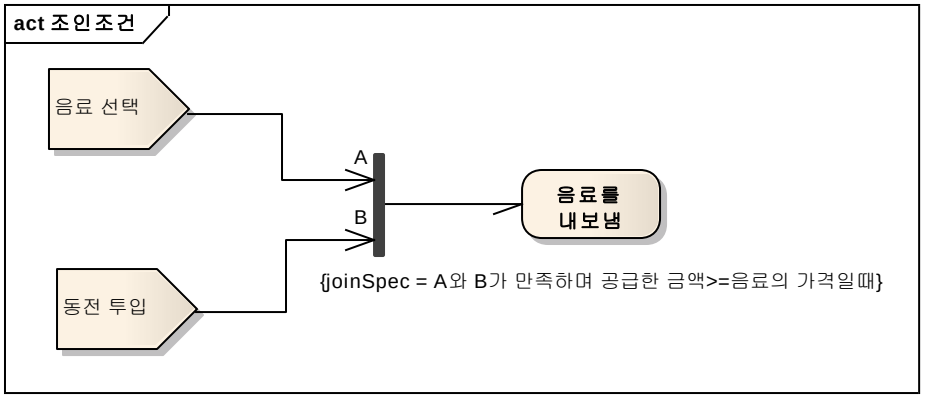
- 액션은 서브 액티비티로 분해될 수 있음



subsidiary activity diagram

❑ 조인 조건*Join specification*

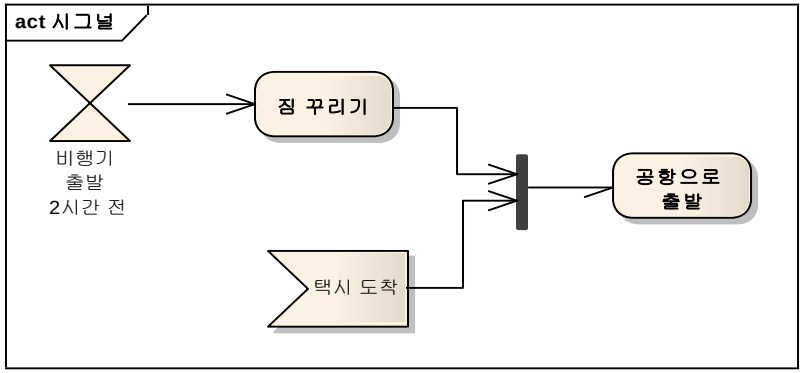
- 조인은 모든 입력 플로우가 조인에 도착했을 때 나가는 방향의 플로우 실행
- 토큰이 조인에 도착할때마다 조인 조건을 검사



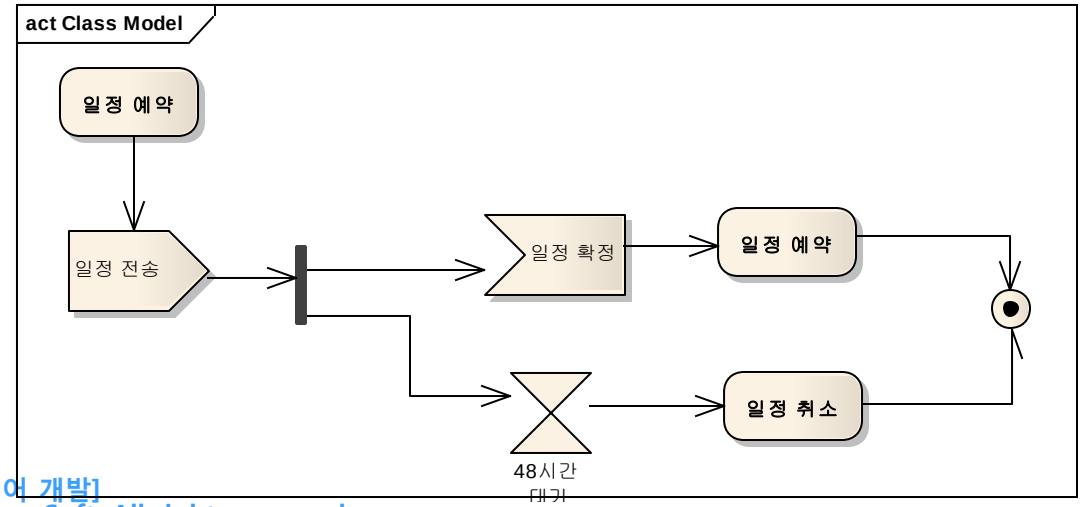
구성조건^시간 ^시그널(Time signal)

❑ 시간 시그널 *Time signal*

- 시간의 흐름에 따라서 발생



- 시그널은 액티비티가 외부의 프로세스로부터 이벤트를 수신함을 표현



작성 및 주의사항 (1/2)

❑ 액티비티 다이어그램 작성 단계

- 작성 대상을 선정
- 필요한 경우 파티션 정의
- 액티비티를 사용하여 처리절차 모델링

단계	내용
작성대상 선정	액티비티 다이어그램의 작성 대상을 선정한다. 대부분의 경우 액티비티 다이어그램은 업무 프로세스를 모델링 하거나, 오퍼레이션 사용을 정의하는 용도로 사용된다.
파티션 정의	대상영역에 명확한 역할을 정의할 수 있을 경우, 역할을 식별하여 파티션으로 표현한다. 파티션은 필수적으로 정의 해야하는 대상은 아니다.
처리절차 모델링	처리절차를 모델링 할 경우 시작점과 끝점이 표현되어야 하고 처리흐름이 도중에 끊겨 미아상태가 되지 않아야 한다.

작성 및 주의사항 (2/2)

❑ 액티비티 다이어그램의 작성 시 주의사항

- 해당 부분을 이해하는데 필수적인 요소들만 표현
- 추상화 수준에 맞는 상세성을 일관되게 제공
- 중요한 의미를 이해하기 적절한 단위로 표현
- 목적을 전달할 수 있는 명칭의 부여
- 주 흐름으로 부터 시작하여 전이, 분기, 동시성을 표현
- 교차선이 최소화하도록 요소를 배치
- 중요한 부분은 노트, 색 등을 이용하여 시각적 효과를 사용

시퀀스 다이어그램

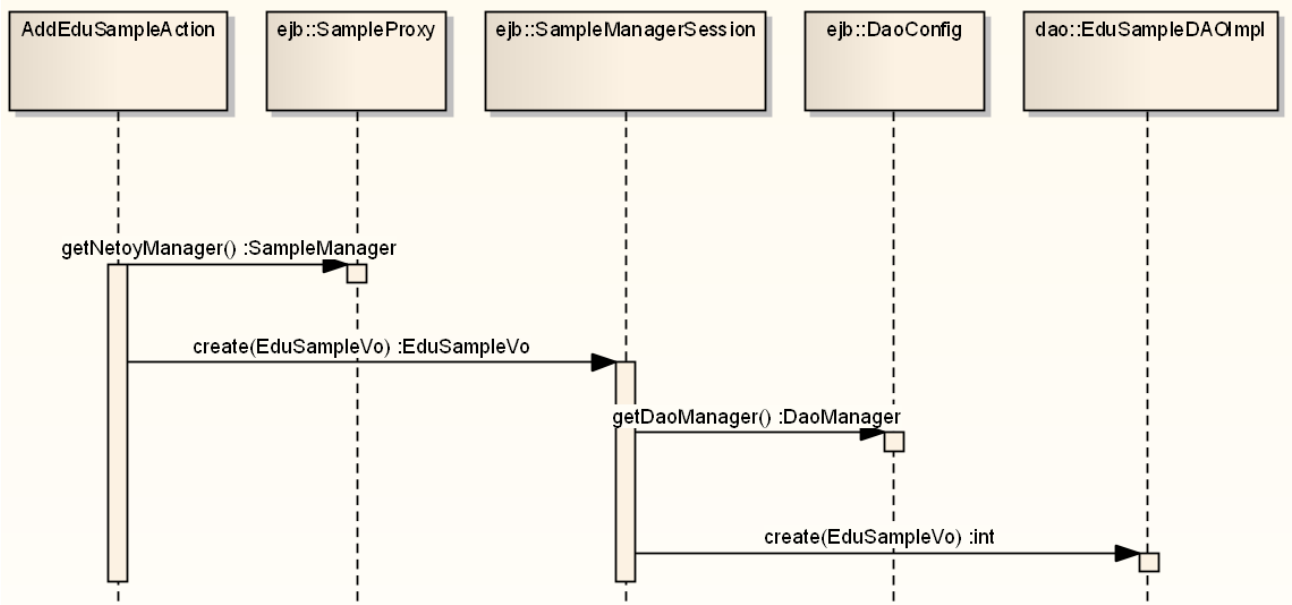
ONE STEP AHEAD

- ❑ 개요
- ❑ 구성요소
- ❑ 작성 및 주의사항

❑ 메타포*Metaphor*

- ATM 현금인출 흐름
  - ATM 기계에 카드 삽입
  - 현금인출 서비스 버튼 클릭
  - 현금인출 가능한 금액을 입력 후 확인
  - 인출 계좌 비밀번호 입력 후 확인
  - 현금과 영수증을 확인한 후 카드를 수령
- 시간의 순서를 중시하고 객체간의 메시지를 도식화

- **동적 모델링*Dynamic Modeling* 기법**
  - 객체와 객체 사이의 동적 상호관계를 정의한 모델
- **시간 개념**
  - 종축을 시간 축으로 하여 시간의 흐름을 나타내며 메시지의 순서에 초점을 두어 객체들 간에 주고 받는 메시지를 보여줌
  - 시스템 실행 시 생성되고 소멸되는 객체를 표기



❑ 작성목적

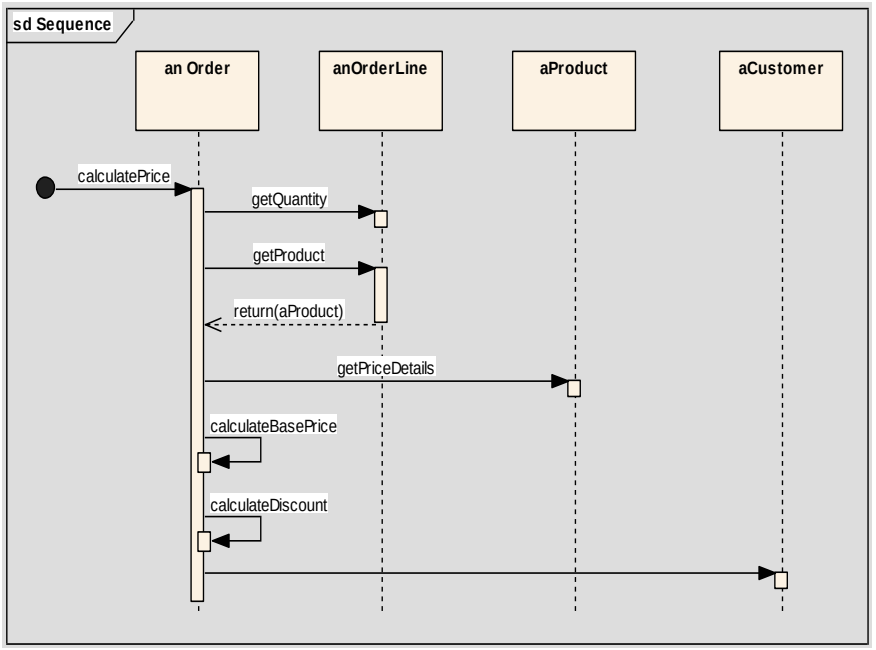
- 객체간의 동적 상호작용을 시간적 개념으로 모델링
- 객체의 오퍼레이션과 속성을 상세히 정의
- 유스케이스를 실현*Realization*
- 프로그램밍 사양을 정의

❑ 작성시기

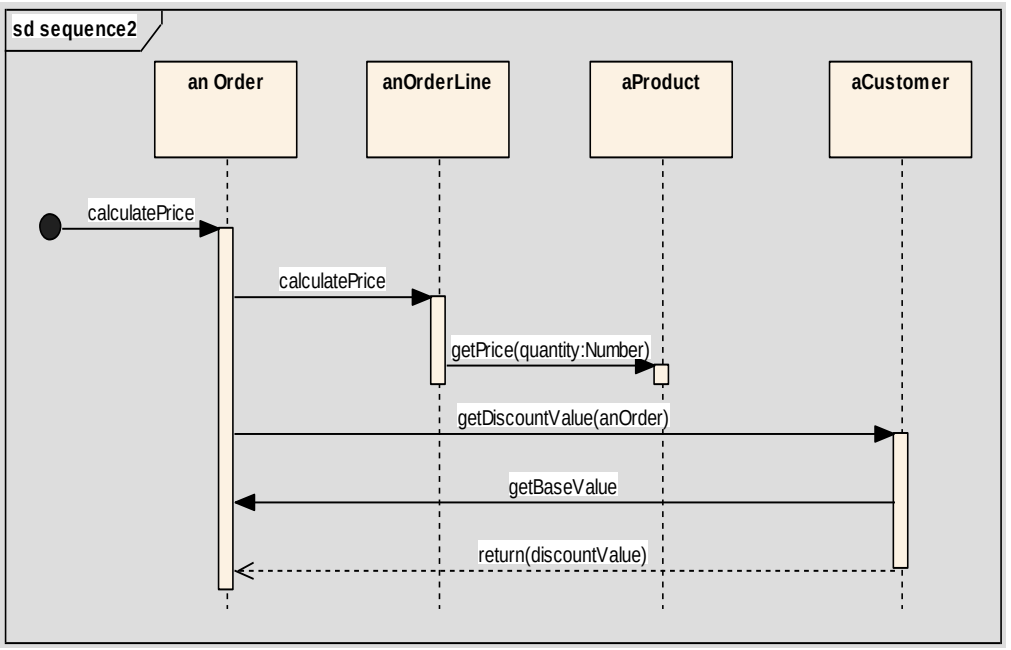
- 유스케이스 다이어그램이 정의된 후 부터 프로그램 코딩 전 까지 수행
- 시퀀스 다이어그램은 다른 여러 다이어그램처럼 여러 번의 정제과정을 거침
- 분석단계에서는 비즈니스 관점에서의 객체
- 설계단계에서는 구현관점의 객체

❑ 집중제어와 분산제어

- 집중 제어가 단순하고, 객체를 추적하는 감각이 보다 수월
- 좋은 설계는 변경에 대한 영향을 지역화 해야 한다는 목표달성을 위하여, 객체 전문가는 분산 제어를 선호



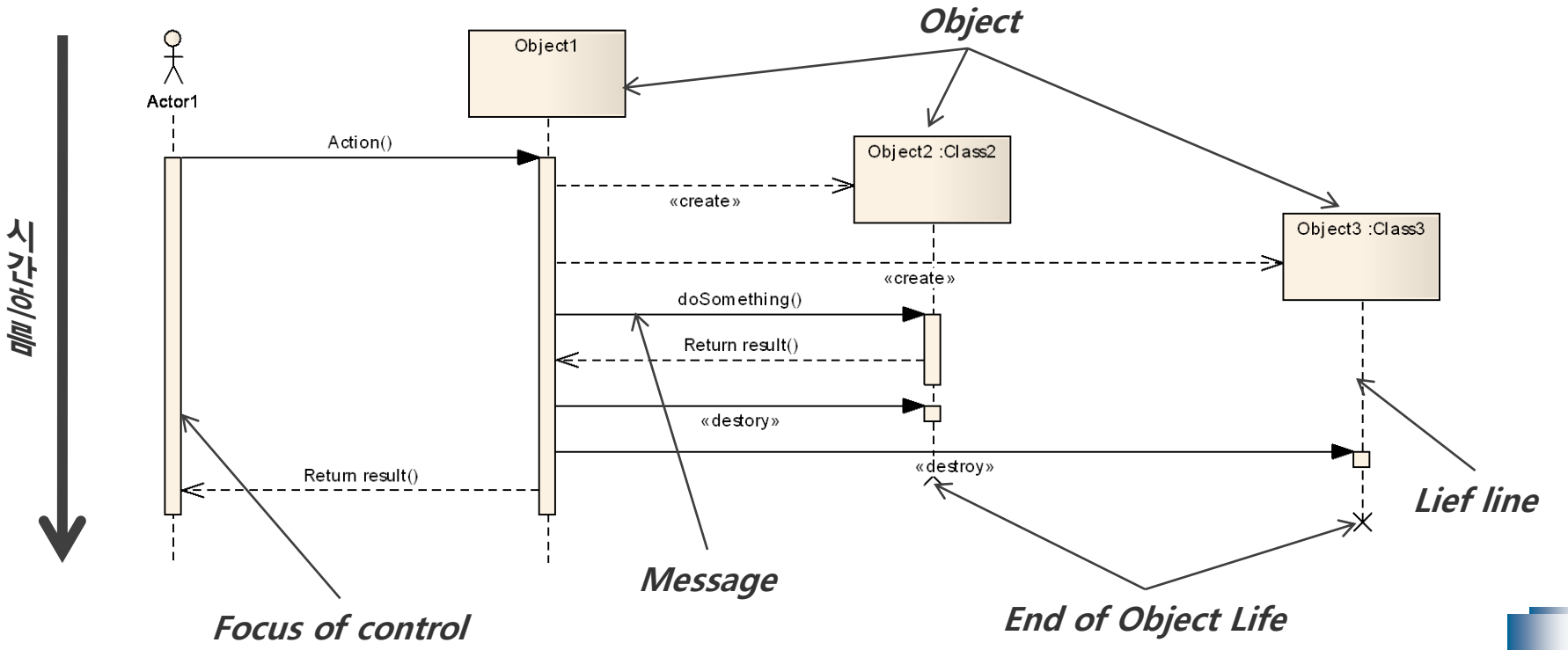
[집중 제어의 예]



[분산 제어의 예]

❑ 시퀀스 다이어그램의 구성요소

- 요소: 액터*Actor*, 객체*Object*
- 관계: 메시지*Message*
- 기타: Life Line, Focus of control



❑ 액터*Actor*

- 시스템의 외부에 존재하면서 시스템과 교류 혹은 상호작용하는 것
- 시스템이 서비스를 해 주기를 요청하는 존재
- 시스템에게 정보를 제공하는 대상

액터의 예)

보험시스템	고객, 사원, 관리자, 결재자, 환자 등
오픈 마켓 시스템	회원, 구매자, 배달자, 관리자, 배송시스템 등
병원관리 시스템	의사, 간호사, 환자, 수납책임자 등

❑ 객체 *Object*

- 클래스의 인스턴스
- 시스템에서는 해당 클래스 타입으로 선언된 변수의 형태로 존재
- 생성되고 소멸되기까지의 생명주기 동안 다양한 상태의 변화가 존재

표기의 예)

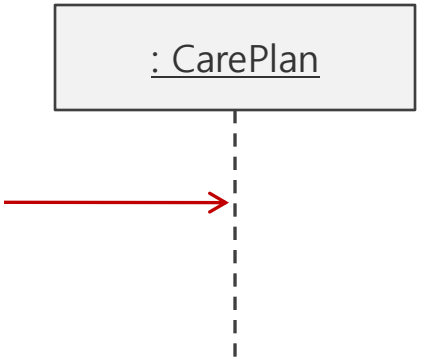
홍길동 : 사원	: 사원 클래스의 객체 중 홍길동 객체를 의미
홍길동	: 홍길동 객체로 클래스는 정의가 없음
: 사원	: 사원 클래스의 일반 객체 (일반적인 객체를 지칭)
홍길동 : 사원 사번=20391 이름=홍길동 급여=4000 성별=남 직위=과장	: 사원 클래스의 홍길동 객체로서 객체 속성을 모두 표현

❑ 메시지*Message*

- 객체지향 패러다임에서 객체와 객체가 통신하는 유일한 수단
- 객체 간의 협력작업과 상호작용
- 향후 클래스의 오퍼레이션으로 구현
- 메시지 유형
  - Flat flow of control
  - Nested flow of control
  - Asynchronous of control
  - Return flow

❑ Flat flow of control

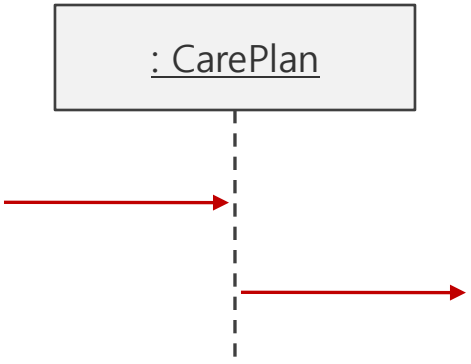
- 가장 일반적인 메시지 형태
- 객체에 메시지를 연결할 때 사용



Flat flow of control

❑ Nested flow of control

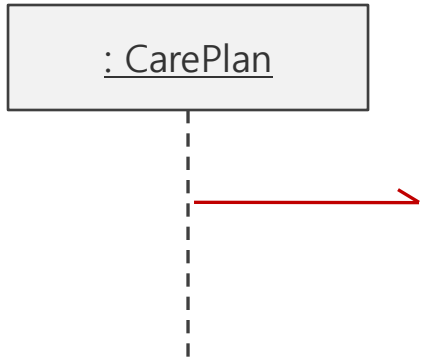
- Procedural call과 같은 의미로 사용
- 중첩*Nested*되는 경우 내부 메시지의 결과가 모두 돌아와야 가장 처음 시작한 메시지의 결과가 되돌려 보내짐
- 메시지 결과가 돌려지게 될 때까지 다음 처리를 진행하지 않는 동기화 메시지



Nested flow of control

❑ Asynchronous of control

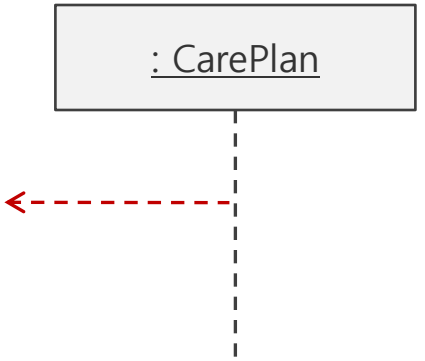
- 객체가 보낸 메시지의 결과를 기다리지 않고 다음 처리를 진행할 경우 사용



Asynchronous flow of control

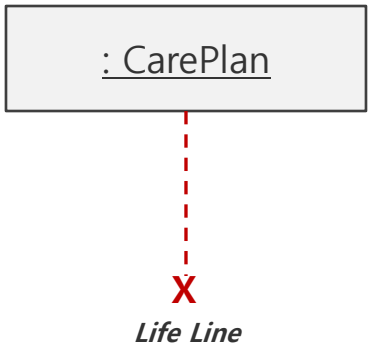
❑ Return flow

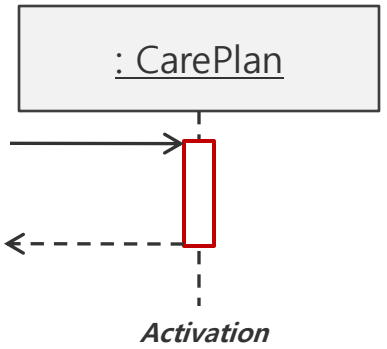
- 메시지를 처리한 결과 리턴을 의미
- 필요한 경우에만 표현



Return flow

- **Life line**
  - 객체의 생존 기간을 의미
  - 객체에 붙은 수직방향의 점선으로 표기
  - 점선 X 표시가 있을 경우 그 시점이 객체의 소멸시점
- **Activation**
  - 객체가 활성화 되어 있는 기간
  - 객체가 외부 메시지를 받고 다른 객체에 보낸 메시지에 대한 Return Flow를 기다리는 시간을 의미
  - Life line 위에 긴 사각형으로 표시





작성 및 주의사항

❑ 시퀀스 다이어그램 작성 단계

- 작성 대상 선정
- 유스케이스의 액터를 파악
- 유스케이스를 실현하기 위해 참여할 클래스(객체)들을 선정
- 시간 순서에 따른 객체 간 메시지 정의
- 필요한 객체를 추가로 정의

단계	내용
대상 선정	유스케이스 다이어그램을 이용하여 시퀀스 다이어그램의 작성 대상을 선정한 후 하나의 유스케이스를 선택하고 유스케이스 정의서를 분석한다.
액터 파악	액터가 둘 이상일 경우라도 모두 좌측부터 액터를 위치시켜야 한다. 순서는 중요하지 않고 메시지 선이 적게 교차하도록 배치하는 것이 좋다.
클래스(객체) 선정	정의된 클래스 중에 유스케이스의 처리에 참여하는 것들을 식별하고 시퀀스 다이어그램에 위치시켜야 하지만 순서는 중요하지 않다.
메시지 정의	유스케이스를 실현하기 위해 필요한 객체들 간의 메시지를 정의하되, 시간 순서에 유의하도록 한다. (시간흐름은 위에서 아래로 흐름)
객체 추가	정의되지 않은 객체가 있으면 시퀀스 다이어그램에 추가하고 객체 사이의 메시지도 정의하여 추가해야 한다.

작성 및 주의사항

❑ 시퀀스 다이어그램 작성 시 주의사항

- 동일한 상호작용을 여러 시퀀스 다이어그램에 중복되게 작성하는 것을 피함
- 중복을 최소화하기 위해 UI별 시퀀스 다이어그램을 작성해도 됨
- 시퀀스 다이어그램의 가독성이 좋도록 적당한 코멘트를 사용
- 메시지 흐름은 액터로부터 시작되게 작성
- 클래스 다이어그램에 표기된 클래스명과 매팅 가능하도록 객체이름을 표기

시퀀스 다이어그램은 유스케이스 별로 하나씩 작성해야 한다.
하지만 경우에 따라서 하나의 유스케이스에 여러 개의 시퀀스 다이어그램으로
또는 여러 유스케이스에서 공통으로 사용하는 상호작용을 하나의
시퀀스 다이어그램으로 작성할 수 있다.

커뮤니케이션 다이어그램

ONE STEP AHEAD

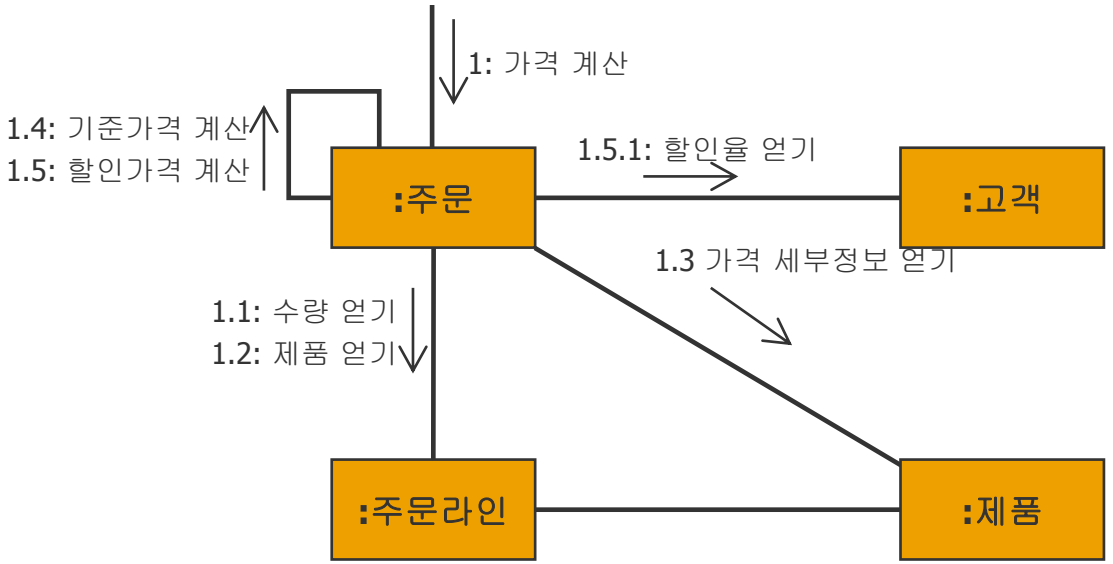
- 개요
- 구성요소
- 작성 및 주의사항

❑ 동적 모델링*Dynamic Modeling* 기법

- 객체와 객체 사이의 동적 상호관계를 정의한 모델

❑ 구조적인 측면 중시

- 해결해야 할 문제가 주어진 상황에서 그 문제를 해결하기 위해 필요한 객체를 정의하고 동적인 상호관계를 순서에 따라 정의
- 모델링 공간이 확보되어 같은 유형의 객체를 모아 놓아 모델링이 가능



❑ 작성목적

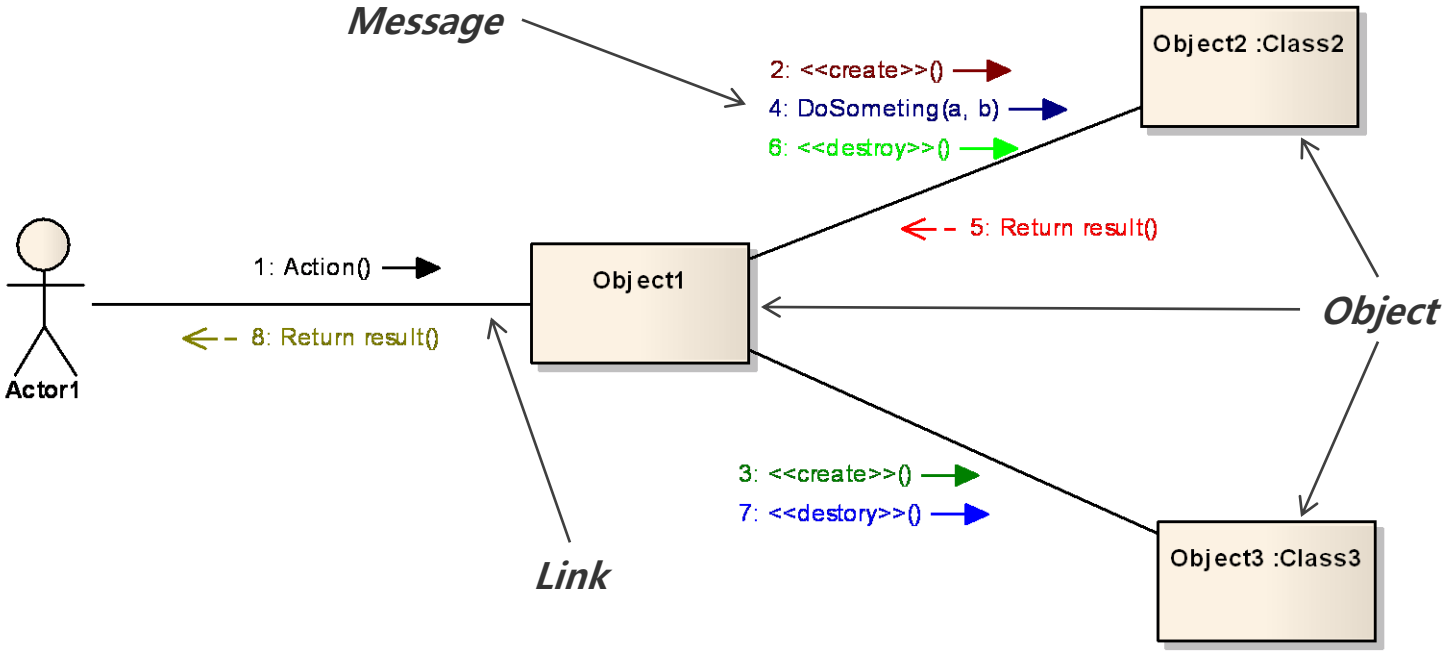
- 객체 간의 동적 상호작용을 구조적 측면을 중시하여 모델링
- 객체 간 상호작용을 정의하는 과정에서 객체를 더욱 상세하게 정의
- 유스케이스를 실현*Realization*
- 프로그래밍 사양을 정의

❑ 작성시기

- 시퀀스 다이어그램의 작성시기와 동일
- 분석단계에서는 비즈니스 관점에서 객체가 등장
- 설계 단계에서는 구현 관점의 객체들이 등장

❑ 커뮤니케이션 다이어그램 구성요소

- 요소: 액터*Actor*, 객체*Object*
- 관계: 메시지*Message*, 링크*Link*



구성요소 (2/7)

❑ 액터*Actor*

- 시스템의 외부에 존재하면서 시스템과 교류 혹은 상호작용하는 것
- 시스템이 서비스를 해 주기를 요청하는 존재
- 시스템에게 정보를 제공하는 대상

액터의 예)

보험시스템	고객, 사원, 관리자, 결재자, 환자 등
오픈 마켓 시스템	회원, 구매자, 배달자, 관리자, 배송시스템 등
병원관리 시스템	의사, 간호사, 환자, 수납책임자 등

❑ 객체 *Object*

- 클래스의 인스턴스
- 시스템에서는 해당 클래스 타입으로 선언된 변수의 형태로 존재
- 생성되고 소멸되기까지의 생명주기 동안 다양한 상태의 변화가 존재

표기의 예)

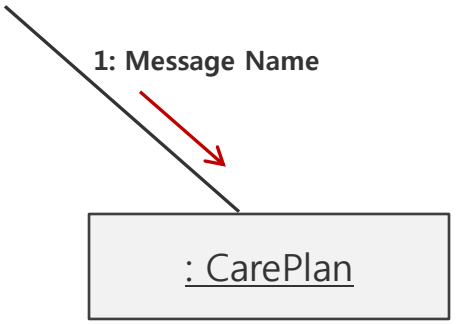
홍길동 : 사원	: 사원 클래스의 객체 중 홍길동 객체를 의미
홍길동	: 홍길동 객체로 클래스는 정의가 없음
: 사원	: 사원 클래스의 일반 객체 (일반적인 객체를 지칭)
홍길동 : 사원 사번=20391 이름=홍길동 급여=4000 성별=남 직위=과장	: 사원 클래스의 홍길동 객체로서 객체 속성을 모두 표현

❑ 메시지*Message*

- 객체지향 패러다임에서 객체와 객체가 통신하는 유일한 수단
- 객체 간의 협력작업과 상호작용
- 향후 클래스의 오퍼레이션으로 구현
- 메시지 유형
  - Flat flow of control
  - Nested flow of control
  - Asynchronous of control
  - Return flow

❑ Flat flow of control

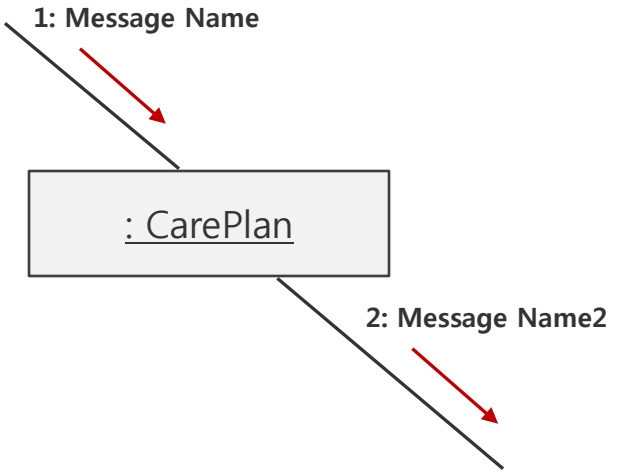
- 가장 일반적인 메시지 형태
- 객체에 메시지를 연결할 때 사용



Flat flow of control

❑ Nested flow of control

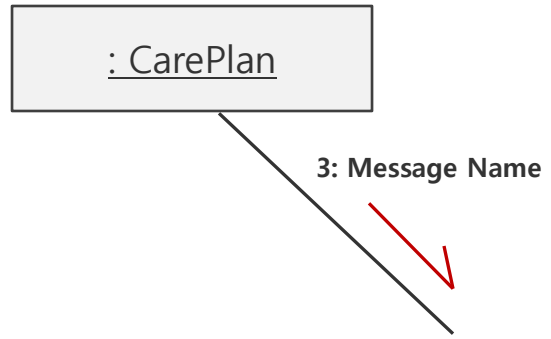
- Procedural call과 같은 의미로 사용
- 중첩*Nested*되는 경우 내부 메시지의 결과가 모두 돌아와야 가장 처음 시작한 메시지의 결과가 되돌려 보내짐
- 메시지 결과가 돌려지게 될 때까지 다음 처리를 진행하지 않는 동기화 메시지



Nested flow of control

❑ Asynchronous of control

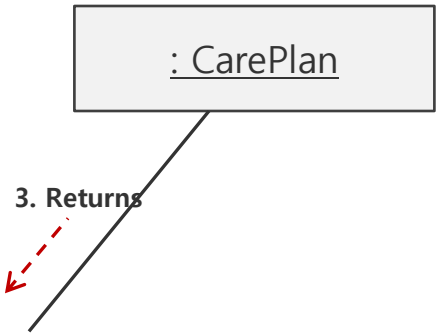
- 객체가 보낸 메시지의 결과를 기다리지 않고 다음 처리를 진행할 경우 사용



Asynchronous flow of control

❑ Return flow

- 메시지를 처리한 결과 리턴을 의미
- 필요한 경우에만 표현



Return flow

❑ 링크*Link*

- 객체와 객체 간의 연관관계를 표현
- 객체 간에 서로 통신을 하기 위해서는 링크로 연결
- 메시지는 링크를 따라 움직인다고 표현
- 링크에는 명명이 가능

작성 및 주의사항

❑ 커뮤니케이션 다이어그램 작성 단계

- 작성 대상을 선정
- 유스케이스의 액터를 파악
- 유스케이스를 실현하기 위해 참여할 클래스(객체)를 정의
- 시간 순서에 따른 객체 간 메시지를 정의
- 필요한 객체를 추가로 정의

단계	내용
작성 대상 선정	유스케이스 다이어그램을 이용하여 작성대상을 선정한 후 하나의 유스케이스를 선택하고, 유스케이스 정의서를 분석한다.
액터 파악	액터가 둘 이상일 경우라도 모두 좌측부터 액터를 위치시켜야 한다.
클래스(객체) 정의	정의된 클래스 중 유스케이스의 처리에 참여하는 것들을 식별한다.
메시지 정의	유스케이스 실현을 하기 위해 필요한 객체들 간의 메시지를 정의하되 시간 순서에 유의하도록 한다.
객체 추가	요구된 처리를 위해 필요하지만 아직 정의되지 않은 객체를 새로 정의하여 추가한다. 이때 추가된 객체 사이의 메시지도 정의하여 추가한다.

작성 및 주의사항

❑ 커뮤니케이션 다이어그램 작성 시 주의사항

- 유스케이스 별로 하나씩 작성
- 가독성이 좋도록 적당한 코멘트를 사용
- 메시지의 흐름은 액터로부터 시작되게 작성
- 클래스 다이어그램에 표기된 클래스 명과 매핑 가능하도록 객체 이름을 표기

동일한 상호작용을 여러 커뮤니케이션 다이어그램에 중복되게 작성하는 것을 피하고 중복을 최소화 시키기 위해서, UI별 커뮤니케이션 다이어그램을 작성해도 된다.