노동부 Ministry of Labor 중소기업직업훈련컨소시엄

Korea Software Technology Association

UML2.x 기초 다루기

훈련기간: 2010.01.25 ~ 02.05

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

Copyright © 2009 Nextree Soft, All rights reserved

과정 개요

❑ 교육 목표 & 특징

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

Copyright © 2009 Nextree Soft, All rights reserved

강의 요구 기술

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

Copyright © 2009 Nextree Soft, All rights reserved

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 역사 UML 다이어그램분류	- 구조 다이어그램 클래스 객체 컴포넌트 배치	- 행위 다이어그램 유스케이스 액티비티 상태기계	- 상호작용 다이어그램 상호작용 Overview 시퀀스 커뮤니케이션 타이밍	- 유스케이스 I 유스케이스 개요 유스케이스 내용 유스케이스 다이어그램
6 일차	**7 일차**	**8 일차**	**9 일차**	**10 일차**
- 유스케이스 II 유스케이스 목표수준 유스케이스 명세 유스케이스 패턴	- 유스케이스 III 유스케이스 분석기법 분석클래스 제어클래스 실체클래스	- 요구사항 모델실습 I 유스케이스 사용자 시나리오 핵심개념 모델	- 요구사항 모델실습 II 인터페이스 추출 유스케이스 분석 컴포넌트 식별	- 설계모델 실습 컴포넌트 설계 유스케이스 설계 도메인 모델

Copyright © 2009 Nextree Soft, All rights reserved

2일차 – 구조 다이어그램

1. 클래스 다이어그램(기본)
2. 클래스 다이어그램(고급)
3. 컴포넌트
4. 배치
5. 패키지

Copyright © 2009 Nextree Soft, All rights reserved

1. 클래스 다이어그램(기본)

- 개요
- 클래스 다이어그램과 객체 다이어그램
- 클래스 란?
- 프로퍼티
- 다중성 및 가시성
- 오퍼레이션
- 일반화
- 노트와 주석
- 의존성

Copyright © 2009 Nextree Soft, All rights reserved

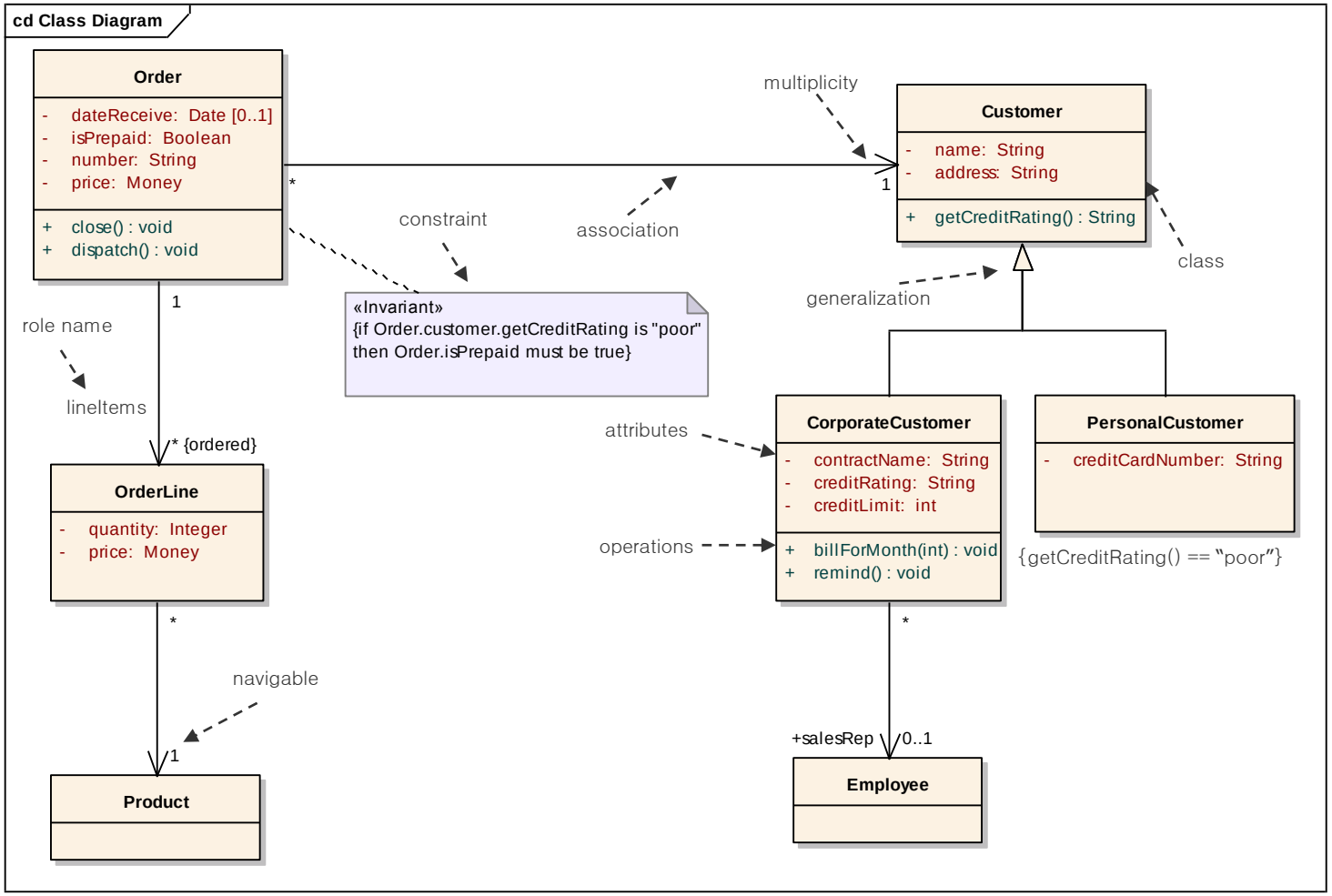
❑ UML의 두 가지 정적 다이어그램

- 클래스 다이어그램
  - UML 다이어그램의 핵심, UML 도구의 코드 생성은 기본적으로 클래스 다이어그램을 기반
  - 클래스, 연관관계, 집합관계, 복합관계, 일반관계를 보여줌
  - 널리 사용되며 그 모델링 개념의 폭이 넓음
    - 기본 개념 : 모든 사람들에게 필요
    - 고급 개념 : 많이 사용 되지 않음
  - 객체 타입인 클래스를 표현하는 다이어그램
  - 클래스의 프로퍼티와 오퍼레이션 및 제약사항을 보여줌
    - 클래스의 특징(feature) = 클래스의 프로퍼티 + 오퍼레이션
  - 클래스 다이어그램에서 클래스는 클래스 개념을 지원하는 언어에서 직접 구현 될 수 있음
    - 예) Java, C++
- 객체 다이어그램
  - 특정 시점에서 객체들의 스냅샷
  - 클래스의 인스턴스와 인스턴스 사이의 링크를 보여줌
  - 클래스 다이어그램을 설명하는 간단한 예를 보여줌

Copyright © 2009 Nextree Soft, All rights reserved

클래스 다이어그램

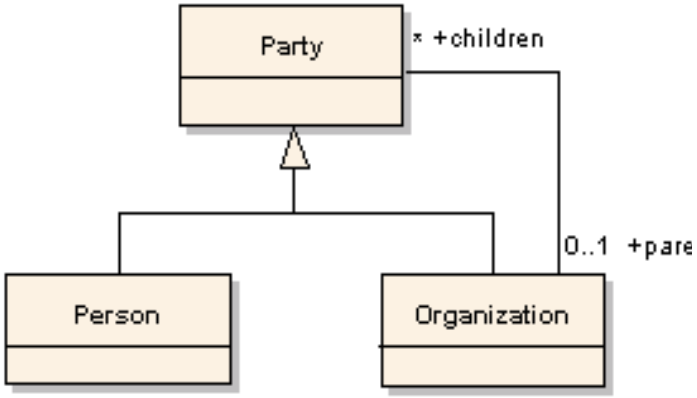
❑ 클래스 다이어그램의 예



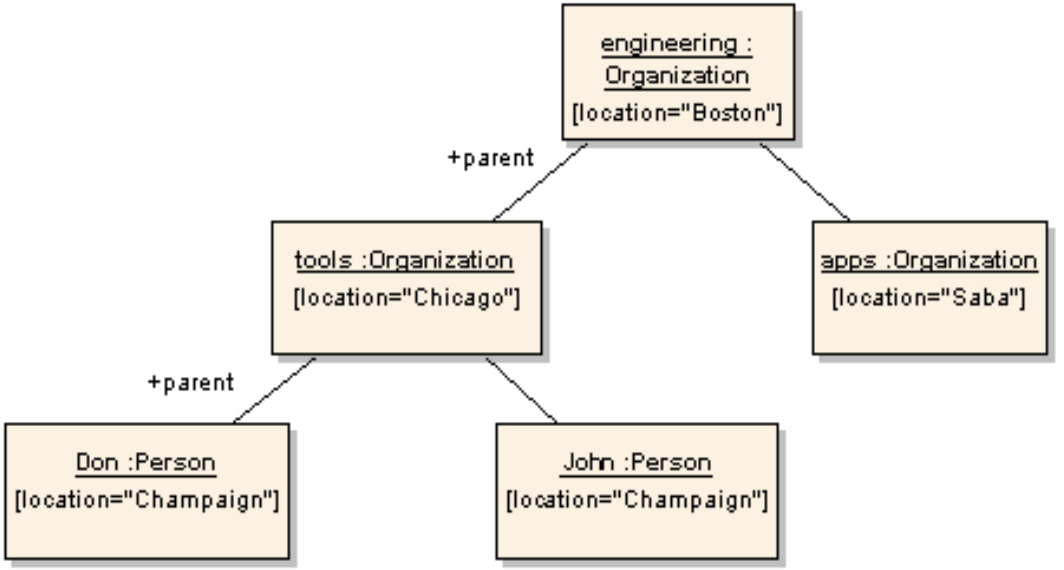
Copyright © 2009 Nextree Soft, All rights reserved

객체 다이어그램

❑ 객체 다이어그램의 예



파티 복합 구조
(Party Composition structure)
클래스 다이어그램



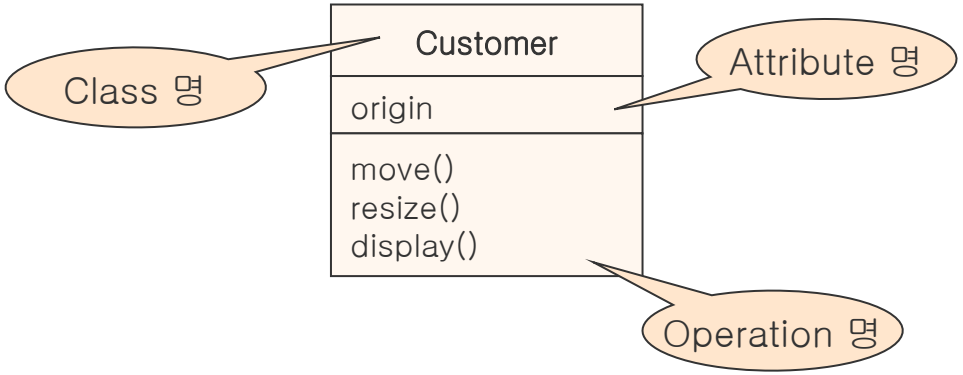
파티의 예제 인스턴스들을 보이는 객체 다이어그램

Copyright © 2009 Nextree Soft, All rights reserved

클래스 란?

❑ 클래스란 무엇인가?

- 사물들의 추상화
- 클래스는 가장 중요한 구성 요소
- 동일한 속성(attribute), 오퍼레이션(operation), 연관(relation), 그리고 의미를 공유하는 객체 집합을 표현
- 하나 또는 그 이상의 Interface를 구현



Copyright © 2009 Nextree Soft, All rights reserved

프로퍼티(Properties)(1/2)

❑ 프로퍼티 개요

- 프로퍼티는 클래스의 구조적인 측면을 나타내는 것
- 속성(attribute)이나 연관(association)으로 표현

❑ 속성(Attributes)

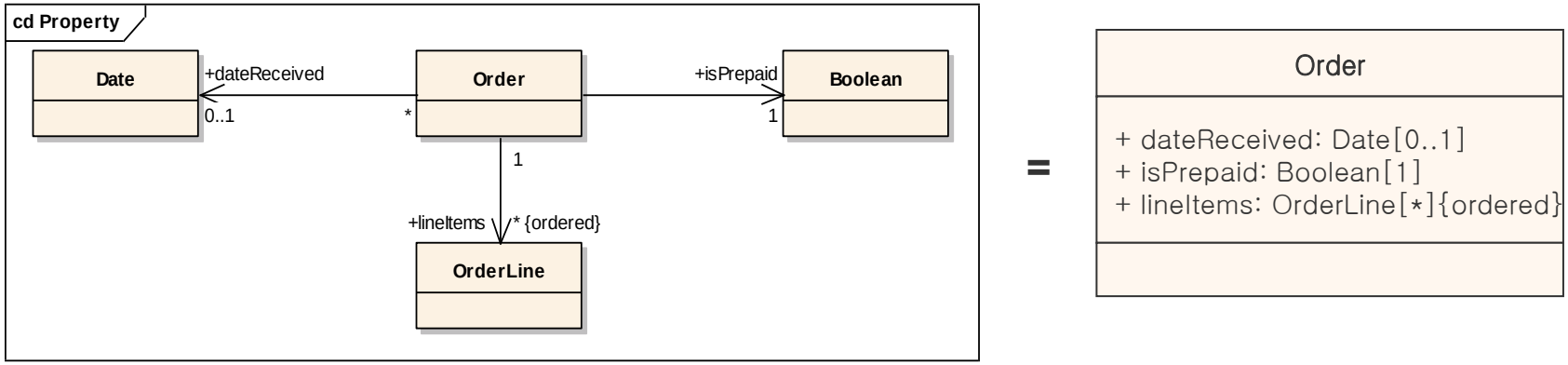
- 클래스 내부에 정의되어 프로퍼티를 설명
- visibility name : type multiplicity = default {property-string}
  ex) - name : String[1] = “Untitled” {readonly}
  - visibility: 가시성, public(+), private(-)
  - name: 속성의 이름
  - type: 속성의 종류
  - multiplicity: 다중성
  - default: 초기 값
  - {property-string}: 속성의 추가 특성을 표현

Copyright © 2009 Nextree Soft, All rights reserved

프로퍼티(Properties)(2/2)

❑ 연관(Associations)

- 연관관계를 이용한 프로퍼티 표현



- 연관관계는 소스 클래스와 타겟 클래스를 연결
- 프로퍼티의 이름은 역할이름으로 표현
- 연관관계의 양끝에 개수를 표현(참여하는 속성의 개수 표현)
- 값 객체와 같은 것들은 속성으로 표현, 참조 객체들은 연관으로 표현

Copyright © 2009 Nextree Soft, All rights reserved

다중성(Multiplicity)

❑ 다중성 개요

- 얼마나 많은 객체들이 존재하는지를 표현
- 속성에 대한 다중성 표현
  - attributeName: AttributeType [Multiplicity]

❑ 다중성의 의미

UML 다중성	의미
1	정확히 1
0..1	0 이거나, 혹은 1
*	무제한(0 포함)
1..*	적어도 하나 이상
2..6	2 부터 6 까지
2, 4	2 이거나, 혹은 4

❑ 다중성 예제

- name: String [1..2] = “Michael”

Copyright © 2009 Nextree Soft, All rights reserved

가시성(Visibility)

❑ 가시성 개요

- 클래스는 공용(public) 요소와 전용(private) 요소 보유
  - 공용(public) 요소는 다른 클래스에 의해 사용 가능
  - 전용(private) 요소는 소유 클래스에 의해서만 사용 가능
- 각각의 프로그래밍 언어는 자신만의 가시성 규칙 보유
  - 여러 프로그래밍 언어 사용자에게 혼동
- UML의 가시성 지시자(visibility indicator)
  - +(public), -(private), ~(package), #(protected)
- 가시성을 사용할 때는 사용중인 언어의 규칙을 적용

Copyright © 2009 Nextree Soft, All rights reserved

오퍼레이션(Operation)(1/2)

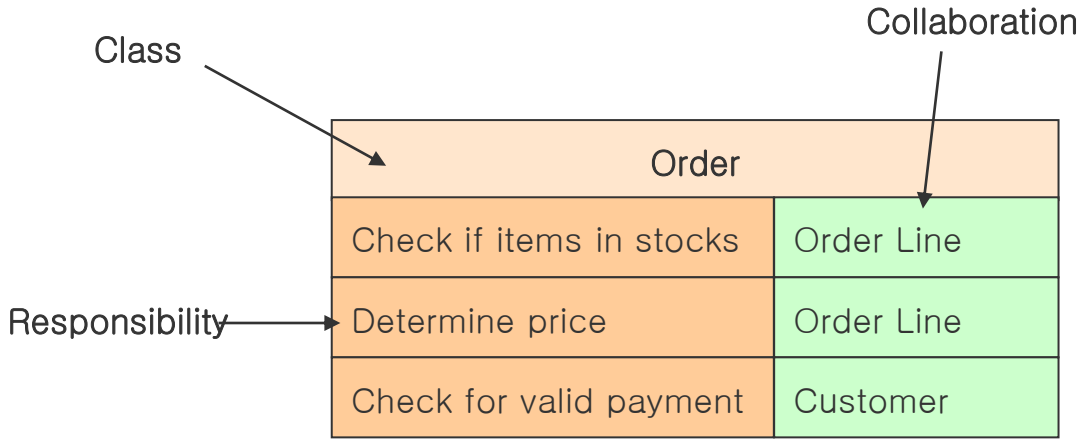
- ❑ 오퍼레이션 개요
  - ▪ 클래스가 수행할 행위
- ❑ 오퍼레이션 문법
  - ▪ visibility name(parameter-list) : return- type{property-string}
    - visibility: 가시성, public(+) 또는 private(-)
    - name: 오퍼레이션의 이름(문자열)
    - parameter list: 파라메터 목록, 오퍼레이션을 위한 parameter list
    - return-type: 존재한다면 반환되는 값의 type
    - property-string : 연산에 적용되는 특성 값
  - ▪ parameter list의 파라메터 표현
    - direction name : type = default value
      - ➢ name, type, default value는 속성)의 표현과 동일
      - ➢ direction: 파라메터 입력(in), 출력(out), 입출력(inout)을 표기

Copyright © 2009 Nextree Soft, All rights reserved

오퍼레이션(Operation)(2/2)

❑ CRC(Class Responsibility Collaboration) 카드

- 개념 모델상에서는 클래스에 오퍼레이션을 사용하지 않고, CRC 카드를 사용

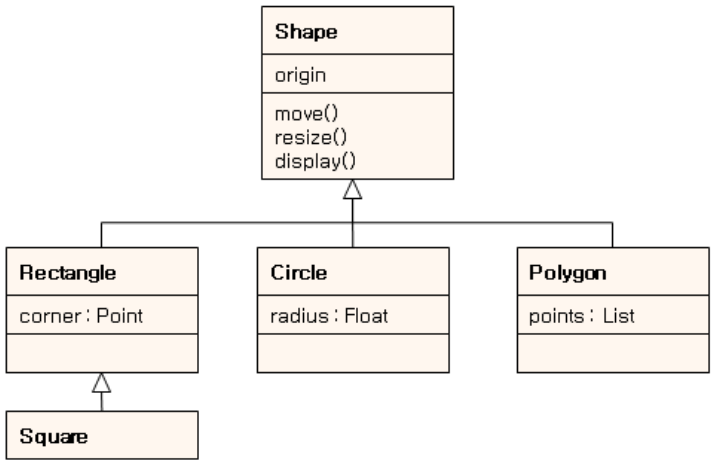


Copyright © 2009 Nextree Soft, All rights reserved

일반화(Generalization)

❑ 일반화 개요

- "**is-a-kind-of**" 관계
- 일반화된 사물(supertype)과 보다 특수화된 사물(subtype)들 사이의 관계를 표현
- 하위타입의 인스턴스의 특징은 상위타입의 인스턴스가 가지는 모든 특징을 가짐
  - 속성, 연관, 오퍼레이션
- 하위타입의 인스턴스는 상위타입의 인스턴스를 대체(substitutability) 가능

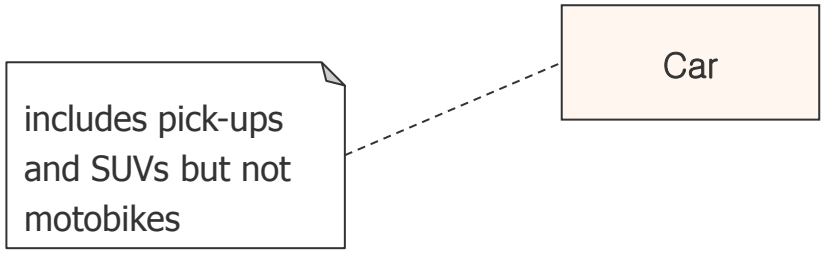


Copyright © 2009 Nextree Soft, All rights reserved

노트(Notes)와 주석(Comments)

❑ 노트 개요

- 노트는 다이어그램을 설명하기 위한 주석
- 다이어그램과 연결되거나 그 자체로 사용될 수 있음

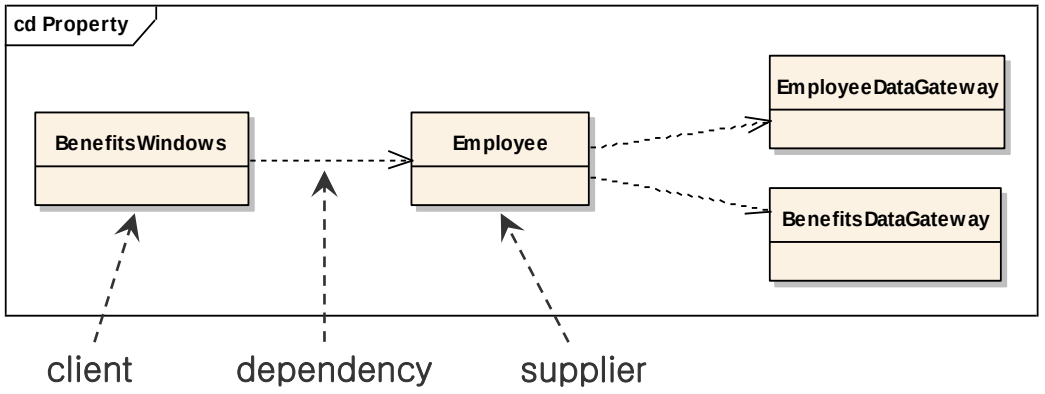


Copyright © 2009 Nextree Soft, All rights reserved

의존성(dependency)

❑ 의존성 개요

- 한 요소(supplier)의 변화가 다른 요소(client)에 영향을 미칠 경우 의존성이 존재
  - 어떤 클래스가 다른 클래스로 메시지를 전송할 경우
  - 어떤 클래스가 다른 클래스를 데이터의 일부로 소유하고 있을 경우
  - 어떤 클래스가 다른 클래스를 오퍼레이션의 파라메터로 언급하고 있을 경우
- 시스템의 대형화로 의존성 제어가 더욱 중요
- 의존성을 통제하지 못할 경우, 특정 요소의 변화에 대한 파급효과는 커짐
- UML에서 모든 요소들의 의존성을 표현할 수 있음
- 기본적으로 의존성은 이행성(transitive)이 없음
- 의존성을 최소화 하는 것이 매우 중요, 특히 패키지 레벨의 순환 의존성을 제거해야 함



Copyright © 2009 Nextree Soft, All rights reserved

의존성(dependency)

실습) 전자 상거래 (EC는) 개발한다.

고객은 시스템을 통해 다양한 컴퓨터 제품을 검색하고

하드웨어 및 소프트웨어를 주문한다.

고객은 신용 카드 또는 온라인 송금을 통해 지불할 수 있습니다.

일단 고객이 자신의 주문에 대한 지불하고 시스템은 온라인 또는 오프라인으로 주문한 제품을 제공합니다.

소프트웨어를 온라인으로 다운로드 받을 수 있고, 나머지는 택배 회사에 의해 배달된다.

이 시스템은 모든 판매 및 거래를 추적합니다.

Copyright © 2009 Nextree Soft, All rights reserved

2. 클래스 다이어그램(고급)

- 키워드(Keyword)
- 책임성(Responsibilities)
- 정적 오퍼레이션과 속성
- 집합과 복합
- 파생 프로퍼티
- 인터페이스와 추상클래스
- 참조 객체와 값 객체
- 한정(Qualified) 연관
- 연관(Association) 클래스

Copyright © 2009 Nextree Soft, All rights reserved

키워드(Keyword)

❑ 키워드 개요

- 기존의 모든 UML 심볼의 의미를 기억하기 매우 곤란하여 키워드를 사용
- UML에 정의되지 않은 심볼이지만 의미가 비슷한 경우 기존 UML 심볼을 사용하고 차이를 키워드로 명시
- UML 인터페이스(interface)
  - 키워드의 대표적인 예
  - 메소드 몸체가 없고 오직 공용 오퍼레이션만을 가진 클래스
  - 클래스 아이콘에 <<interface>> 키워드를 사용하여 표시

❑ 키워드 표기법

- 이중 꺽쇠(<<~>>) 내에 텍스트 표시
  - <<interface>>
- 중괄호({~}) 내에 텍스트 표시
  - {abstract}
- 이중 꺽쇠 내에 표기할 내용과 중괄호 내에 표기할 내용 구분은 불명확

❑ 스테레오타입(stereotype)

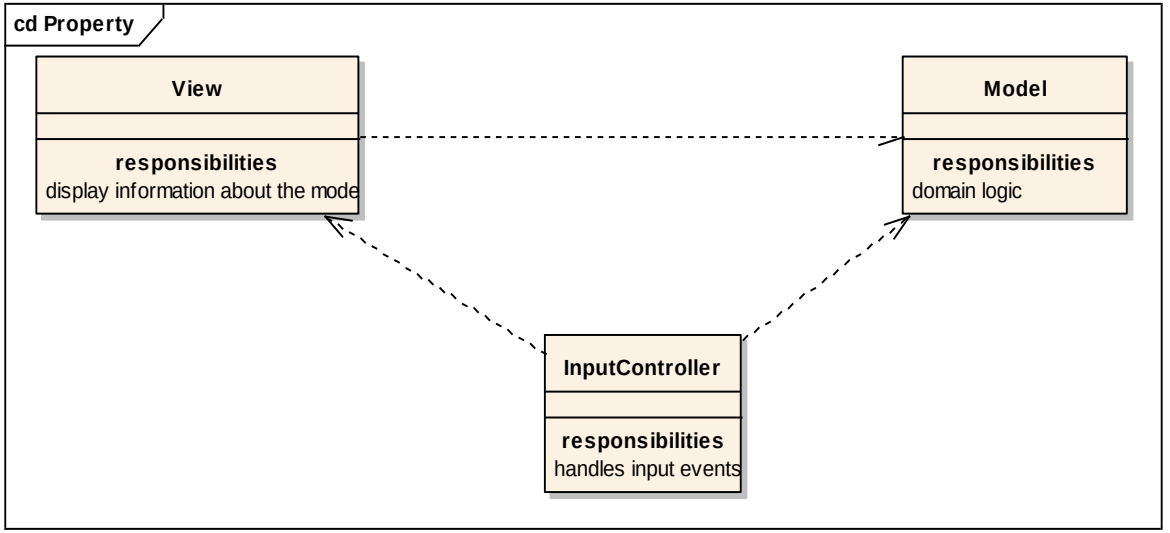
- UML 1에서 이중 꺽쇠(<<~>>)로 표현된 키워드

Copyright © 2009 Nextree Soft, All rights reserved

책임성(Responsibilities)

❑ 클래스에 책임 표시

- 클래스 내의 별도 구획(compartment) 내에 문자열을 사용하여 표시
- 구획에 명칭 부여 가능

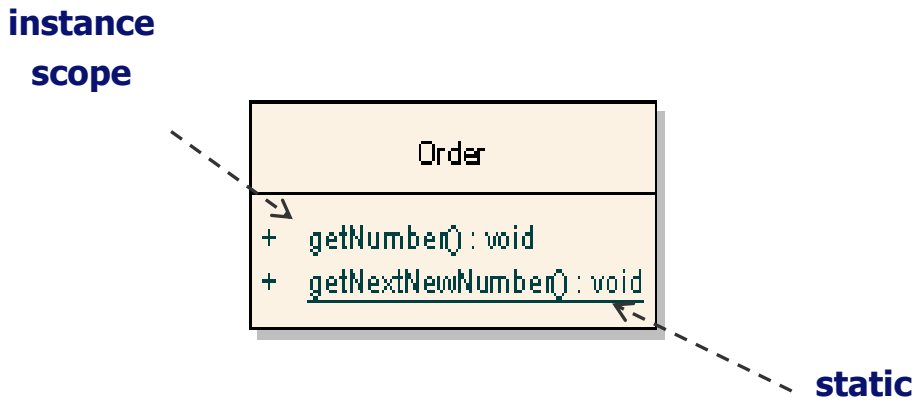


Copyright © 2009 Nextree Soft, All rights reserved

정적 오퍼레이션과 속성(Static Operations and Attributes)

❑ 정적 오퍼레이션과 속성 개요

- 인스턴스가 아니라 클래스에 적용되는 오퍼레이션 또는 속성
- 클래스 다이어그램 상에 밑줄을 그어 표현

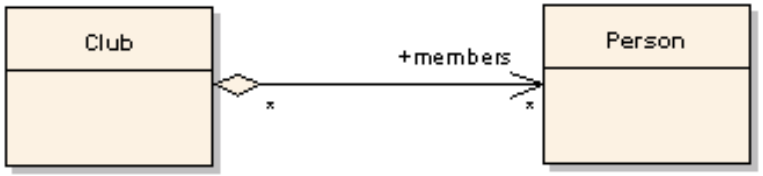


Copyright © 2009 Nextree Soft, All rights reserved

집합(Aggregation)과 복합(Composition)

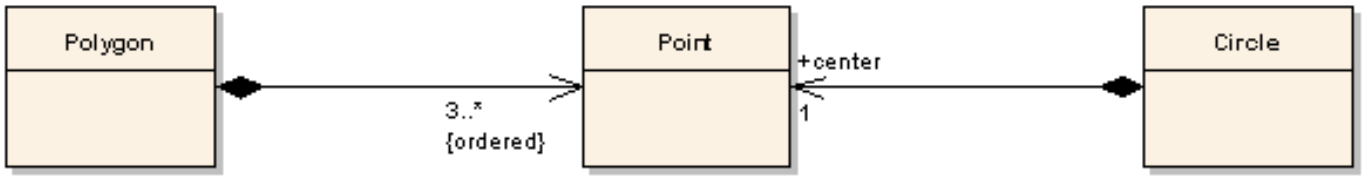
❑ 집합

- part-of 관계
- 연관관계(association)와의 차이가 모호
  - UML은 집합을 포함하지만 특별한 의미를 부여하지는 않음



❑ 복합

- 비 공유 규칙(“no sharing” rule)
  - 인스턴스는 오직 하나의 소유 클래스만을 가짐
    - ➢ 클래스 다이어그램 상에는 여러 개의 소유 클래스 표현 가능
  - 포함하는 클래스 쪽의 다중성은 0..1 또는 1
  - 포함하는 인스턴스 삭제 시 자동적으로 포함되는 인스턴스 삭제
- 용도 : 값 객체(value object), 강한 배타적인 소유관계 표현

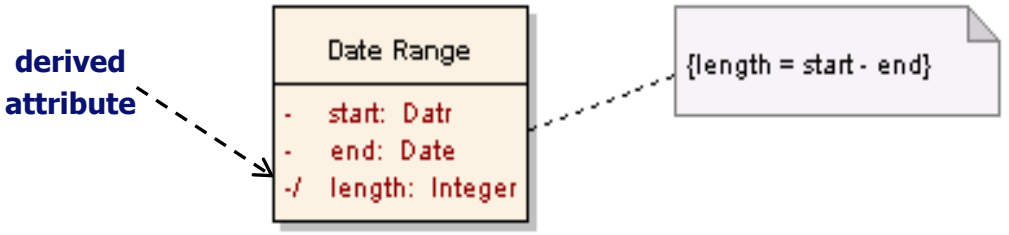


Copyright © 2009 Nextree Soft, All rights reserved

파생 프로퍼티(Derived Properties)

❑ 파생 프로퍼티 개요

- 다른 값에 의해 계산 가능한 프로퍼티



- 두 가지 해석 방법
  - 계산된 값과 저장된 값
    - start와 end는 저장된 값(stored value)
    - length는 계산된 값(calculated value)
    - 일반적이지만 Date Range 내부에 대한 정보를 너무 많이 도출
  - 값 간의 제약사항
    - 세가지 값 사이에 제약사항이 존재함을 표현
    - 세가지 값 중 어떤 것이 파생인지는 중요하지 않음

❑ 파생 연관관계

- 연관관계 표기법을 사용하여 프로퍼티에 적용
- 연관관계 이름에 '/'를 추가

Copyright © 2009 Nextree Soft, All rights reserved

인터페이스(Interface)와 추상(Abstract) 클래스(1/3)

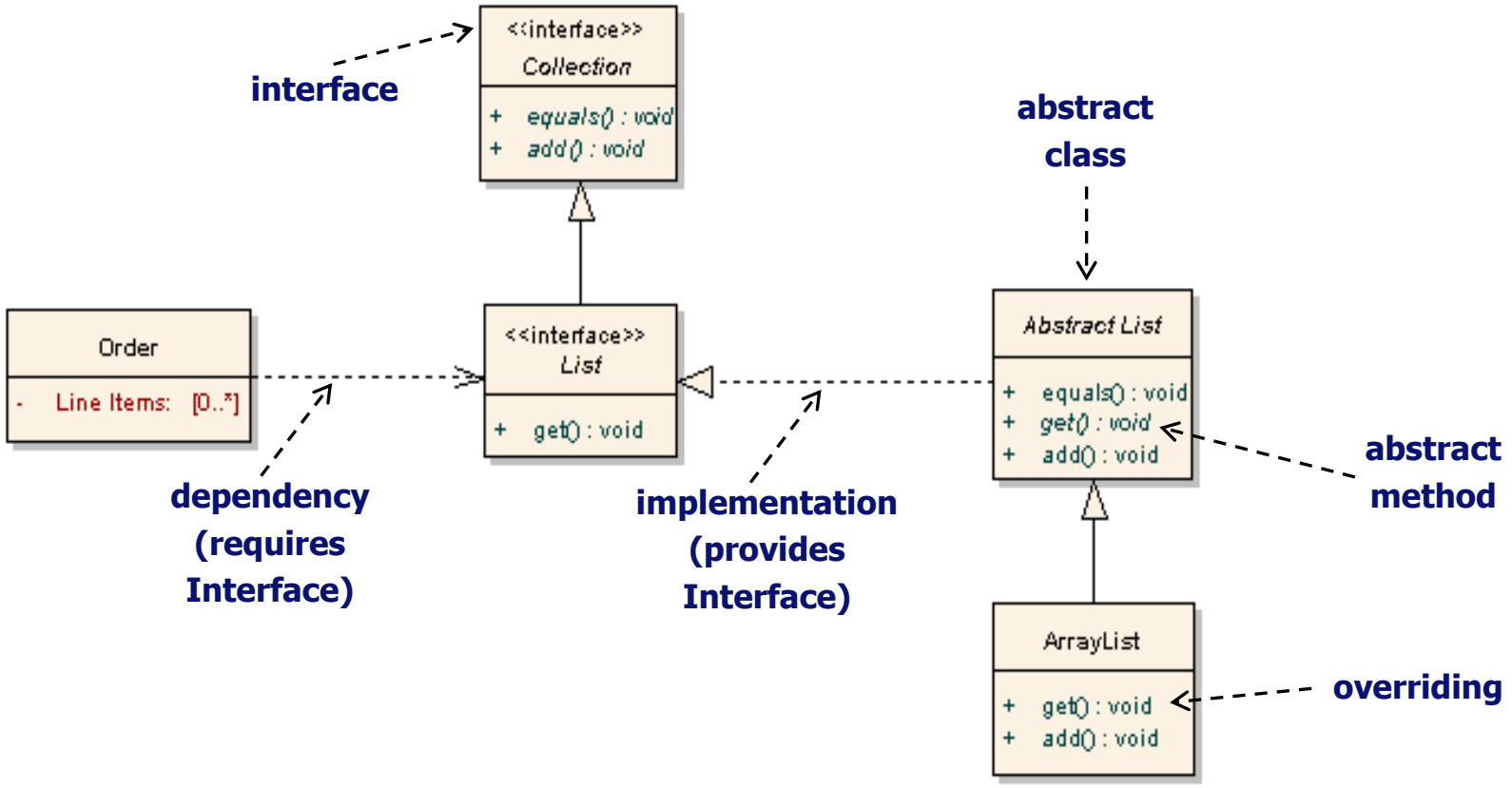
- **추상 클래스**
  - 직접 인스턴스를 생성할 수 없는 클래스
  - 추상 오퍼레이션(abstract operation)
    - 구현이 없이 순수한 정의(pure declaration)만을 가진 오퍼레이션
    - 추상 클래스는 하나 이상의 추상 오퍼레이션 보유
  - 추상 클래스와 추상 오퍼레이션은 이탤릭 체로 표현
- **인터페이스**
  - 어떤 구현도 가지지 않는 클래스
    - 모든 특성이 추상(abstract)
    - 키워드 <<interface>>를 사용하여 표현
- **클래스와 인터페이스 간의 관계**
  - 인터페이스 제공(provides interface)
    - 클래스가 인터페이스를 치환(substitutable) 가능
    - 인터페이스를 구현하거나 인터페이스의 서브타입을 구현
  - 인터페이스 요구(requires interface)
    - 작업을 수행하기 위해 인터페이스의 인스턴스를 요구
    - 인터페이스에 대한 의존(dependency) 관계

Copyright © 2009 Nextree Soft, All rights reserved

인터페이스(Interface)와 추상(Abstract) 클래스(2/3)

❑ 인터페이스의 장점

- 구현의 변경이 용이
- 구현이 아닌 인터페이스에 대해 프로그래밍
  - 변경에 의해 영향 받는 부분을 최소화

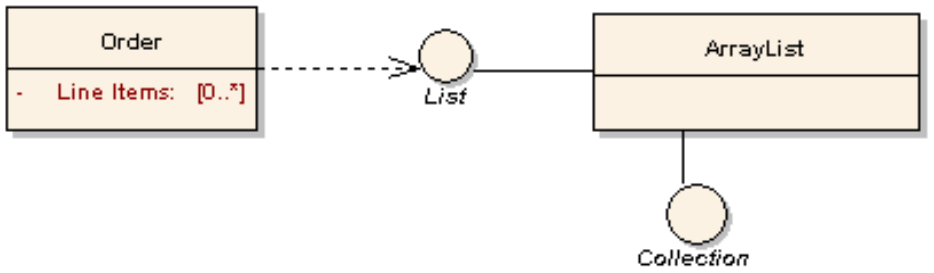


Copyright © 2009 Nextree Soft, All rights reserved

인터페이스(Interface)와 추상(Abstract) 클래스(3/3)

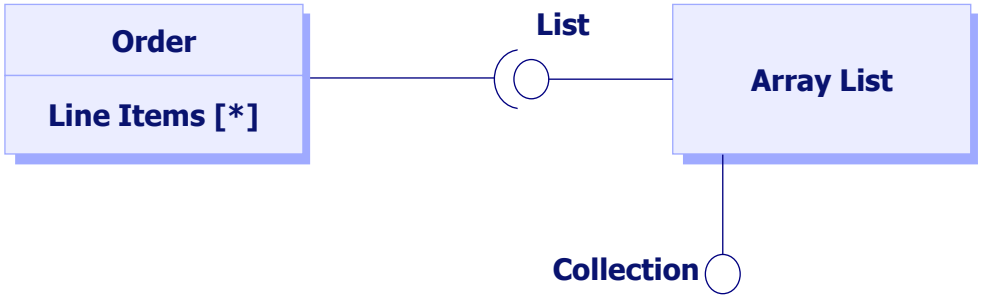
❑ 인터페이스 - UML 1 표기법

- 인터페이스를 롤리팝(lollipop)으로 표현
- 의존 관계 사용



❑ 인터페이스 - UML 2 표기법

- 의존관계를 소켓(socket) 표기법으로 대체



Copyright © 2009 Nextree Soft, All rights reserved

읽기 전용(Read-Only)와 동결(Frozen)

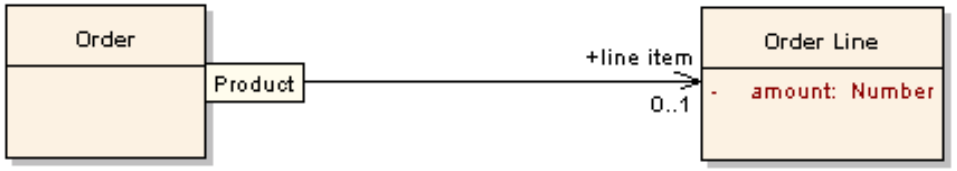
- **읽기 전용**
  - 클라이언트에 의해 읽기만 가능하고 갱신은 불가능한 프로퍼티
  - {readOnly} 키워드 사용
- **동결**
  - 객체 생명주기 동안 변경 불가(immutable)
  - {frozen} 키워드 사용
  - UML 2에서는 누락
  - 클래스에 적용 가능
    - 인스턴스의 모든 프로퍼티가 동결임을 표시

Copyright © 2009 Nextree Soft, All rights reserved

한정 연관(Qualified Association)

❑ 한정 연관 개요

- 객체의 그룹이 속성값에 의하여 그룹화 될 경우 고려
- 클래스 간 연관에 속성이 중요하게 부각될 필요가 있을 경우 한정자로 이를 나타냄
- Order와 Order Line 간의 한정 연관 예
  - Order는 각 Product 인스턴스에 대해 하나의 Order Line 인스턴스와 연관



  - 한정 연관관계는 소프트웨어 관점에서의 인터페이스 표현

```
class Order . . .
   public OrderLine getLineItem(Product aProduct);
   public void addLineItem(Number amount, Product forProduct)
```

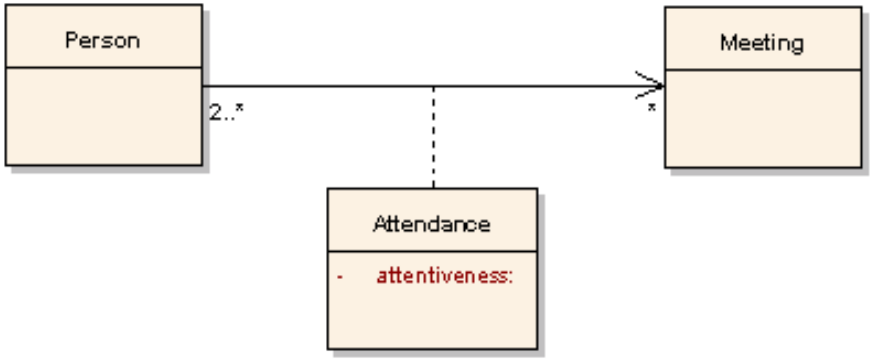
  - 한정 연관관계에서의 다중성(Multiplicity)
    - ➢ 실제로 하나의 Order는 여러 개의 Order Line과 연관
    - ➢ 다중성은 한정자의 문맥에서 명시

Copyright © 2009 Nextree Soft, All rights reserved

연관 클래스(Association Class)

❑ 연관 클래스 개요

- 연관관계에 속성, 오퍼레이션, 다른 특성 추가 가능
- 참여 객체간에 제약사항 부가
  - 참여하는 두 객체 간에 오직 하나의 연관 클래스 인스턴스 만이 존재 가능



연관 클래스

연관 클래스를 완전한 클래스로 만들기

Copyright © 2009 Nextree Soft, All rights reserved

연관 관계(1/3)

- **연관 관계**
  - 한 클래스가 다른 클래스를 인지하는 것
  - 연관은 매우 광범위한 의미를 갖는다.
  - 연관 관계의 의미를 명확하게 표현해야 한다.
- **모호한 연관 관계**
  - 예)
    - 사람이 도서를 대출한다. 사람은 학생 또는 교수를 나타낸다.
    - 사람이 도서를 관리한다. 사서가 도서의 등록/삭제 등을 한다.

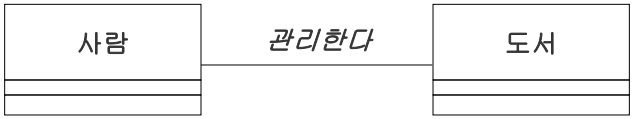
모호한 연관 관계

Copyright © 2009 Nextree Soft, All rights reserved

연관 관계(2/3)

❑ 연관의 이름과 역할

- 연관의 이름은 실선 위에 표시 – 동사 또는 동사구
- 역할의 이름은 연관 관계를 속성으로 표현할 때 상대 객체에 대한 이름으로 사용

(1) 연관 이름의 사용	**(2)** 역할의 사용
	

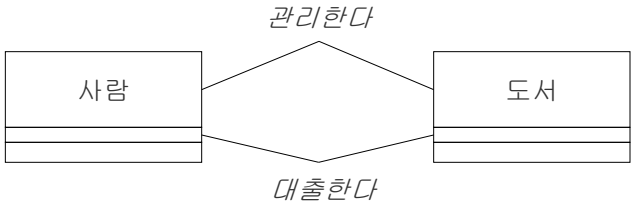
class 사람 {
private 도서 the_도서;
}
	class 도서 {
	private 사람 사서;
	}

Copyright © 2009 Nextree Soft, All rights reserved

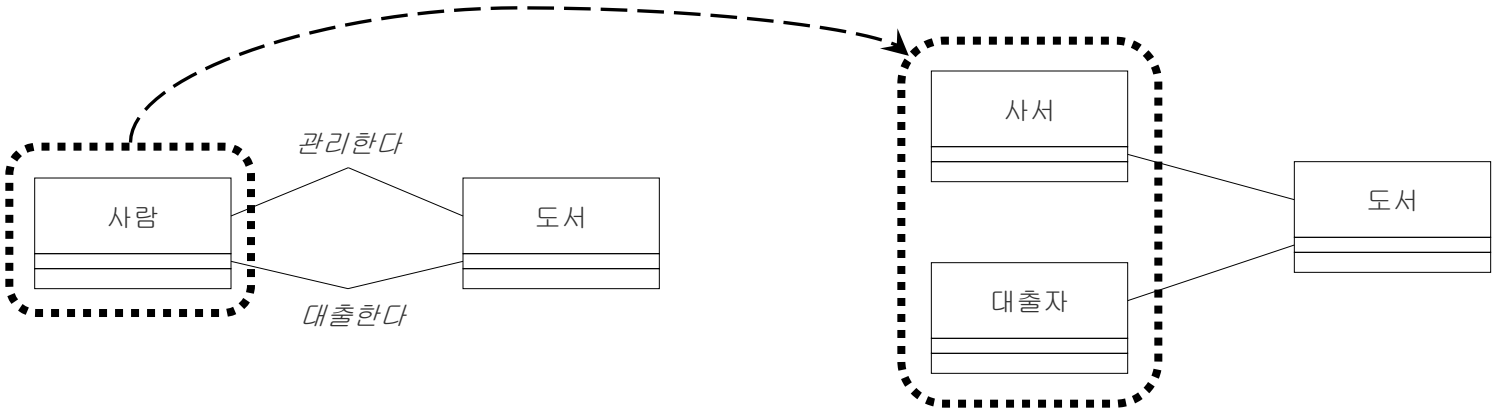
연관 관계(3/3)

❑ 복수 연관

- 동일한 두 클래스 사이에 두 개 이상의 연관 관계가 맺어지는 것
- 두 클래스가 명확하게 다른 의미의 관계를 맺는 경우 사용



- 복수 연관 관계의 신중한 사용



복수 연관 사용의 대안

Copyright © 2009 Nextree Soft, All rights reserved

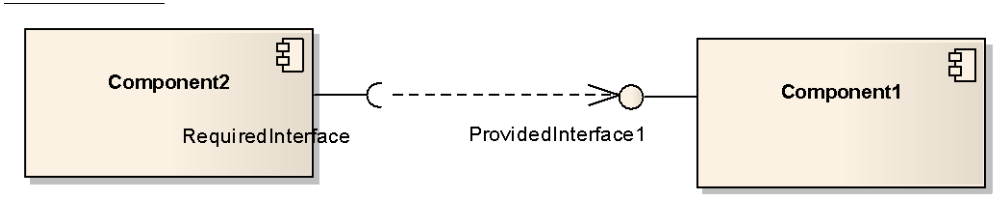
3. 컴포넌트 다이어그램

- ❑ 개요
- ❑ 구성요소
- ❑ 작성 및 주의사항

Copyright © 2009 Nextree Soft, All rights reserved

개요 (1/3)

- **시스템의 구현과정**
  - 실행모듈(컴포넌트)을 정의하고 실행모듈 간의 정적 상호작용을 정의한 모델
  - 물리적 구성요소들로 실행모듈(컴포넌트) 구성되고 그들간의 연관성을 정의
- **SW 분야에서 사용되는 컴포넌트의 광역적 의미**
  - 시스템의 재사용 가능한 구성요소
  - 시스템의 교체단위이자 업그레이드 단위
  - 인터페이스를 통한 기능 사용 및 독립적으로 인도되는 기능조각



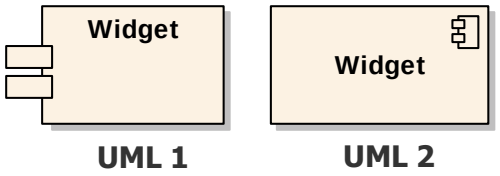
Copyright © 2009 Nextree Soft, All rights reserved

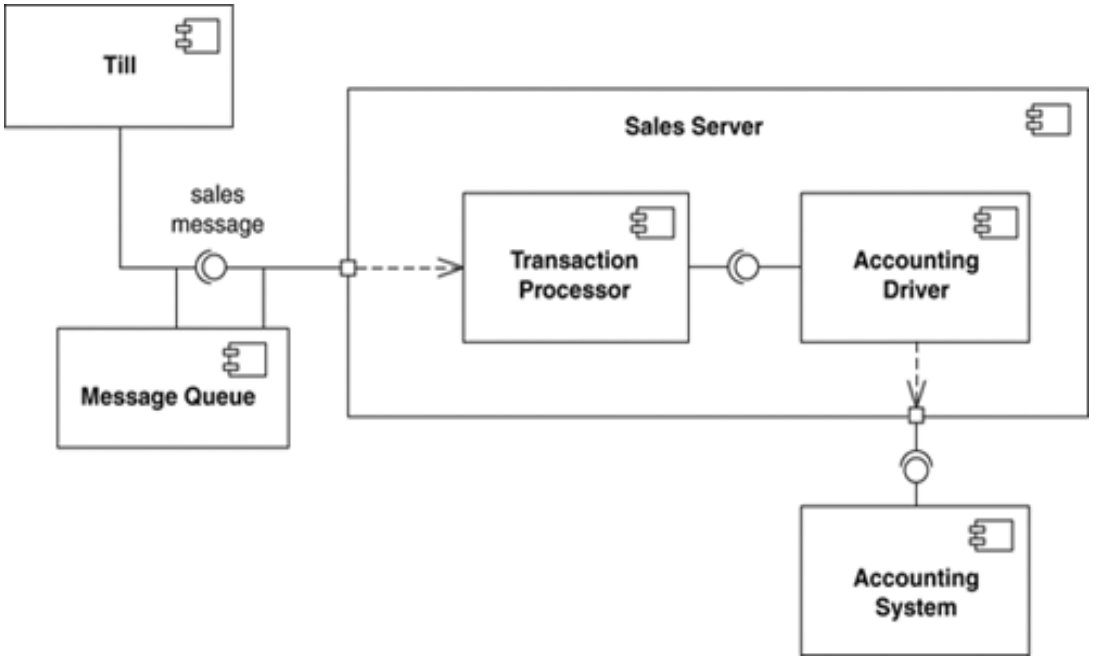
개요 (2/3)

❑ 표기법 변경

- UML1.x의 표기법이 UML2.x에서 변경

표기형식의 변경





Copyright © 2009 Nextree Soft, All rights reserved

개요 (3/3)

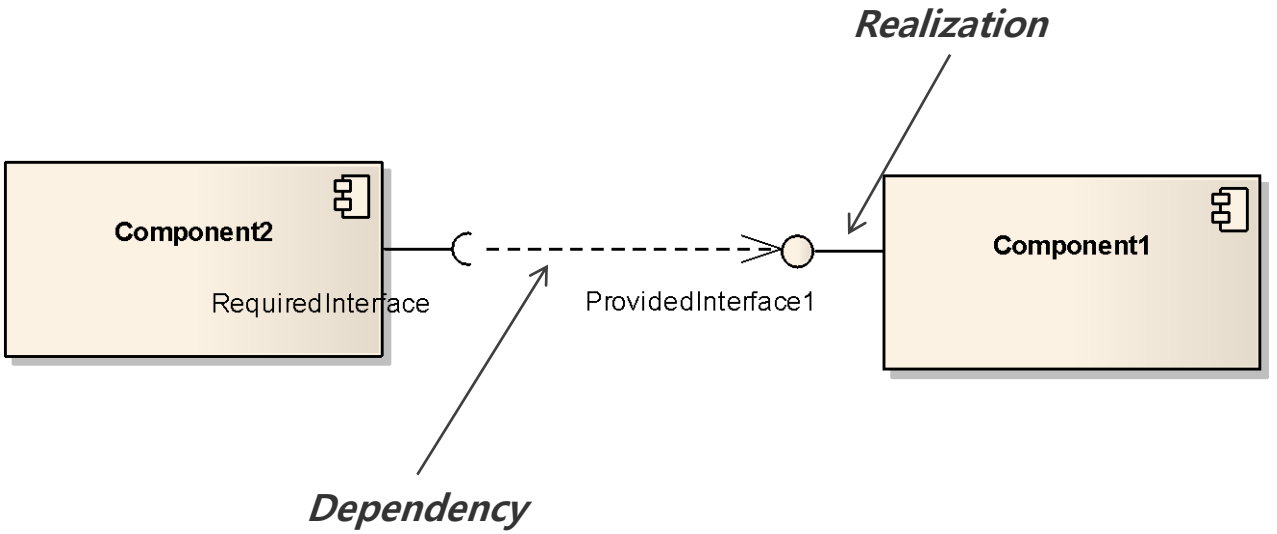
- **작성목적**
  - 시스템의 실행모듈(컴포넌트)들을 정의
  - 컴포넌트 간 의존성*Dependency* 정의
  - 실행모듈뿐 아니라 소스코드, 데이터베이스 등의 상호작용 모델링
- **작성시기**
  - 모든 클래스가 물리적으로 완전히 정의되고 상호관계도 정의된 후 작성
  - 시스템 설계단계 후반에 주로 작성

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (1/5)

❑ 컴포넌트 다이어그램의 구성요소

- 요소: 컴포넌트*Component*, 인터페이스*Interface*
- 관계: 의존성*Dependency*, 실현*Realization*



Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (2/5)

❑ 컴포넌트*Component*

- 독립적으로 배포되고 교체되며 재사용될 수 있는 SW 조각
- 소스코드, User Interface, 분석, 설계 산출물 등을 포함한 광의적 정의
- 용어를 표현하는 사람의 의도에 따라 달라지므로 컨텍스트 파악이 중요

컴포넌트 예)

보험시스템	고객, 사원, 계약, 상품 등
오픈 마켓 시스템	신용카드 결재, 상품, 우편번호 검색 등

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (3/5)

❑ 인터페이스*Interface*

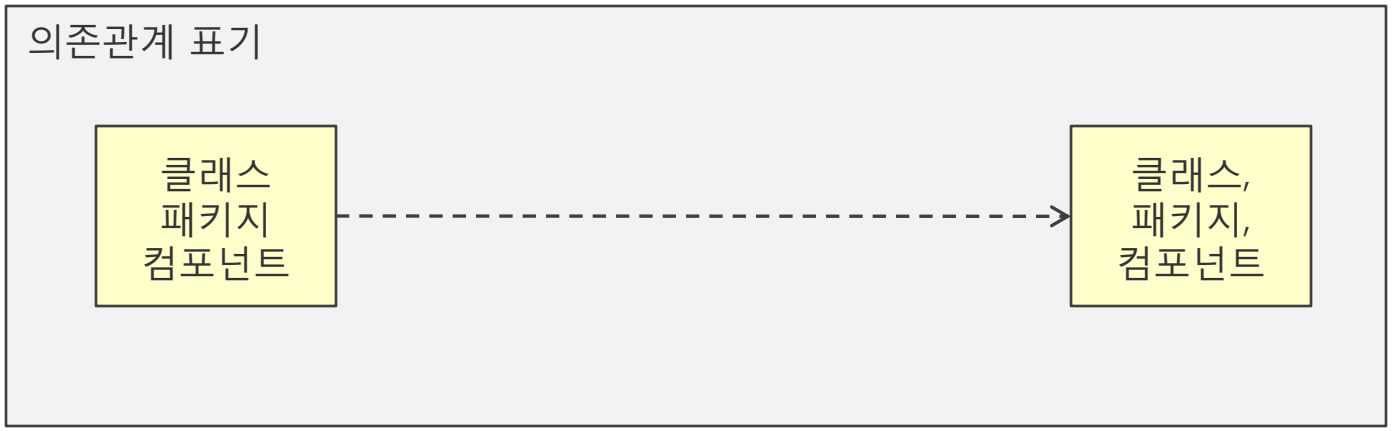
- 클래스의 일종
- 컴포넌트의 기능을 외부에 공개할 목적으로 사용
- 클래스나 컴포넌트는 선언뿐인 인터페이스의 구현을 담당

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (4/5)

❑ 의존관계*Dependency*

- 객체나 컴포넌트가 다른 객체나 컴포넌트의 실행을 요구할 경우 실행 또는 참조관계 표현
- 클래스, 패키지, 컴포넌트 사이에 주로 사용되는 관계
- 때로는 클래스 – 패키지 – 컴포넌트 상호 간에도 사용

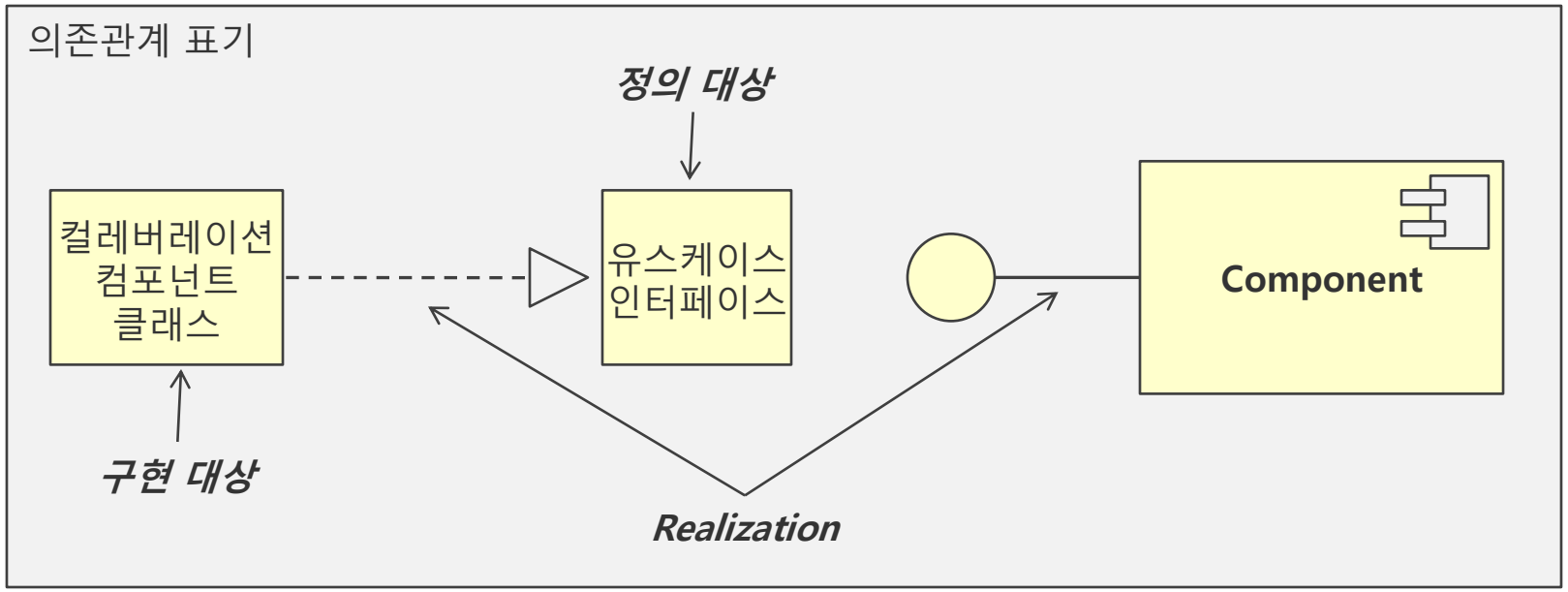


Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (6/5)

❑ 실체화 관계*Realization*

- 정의하는 인터페이스와 이를 구현하는 클래스 간의 표현하는 관계
- 실체화 유형
  - 유스케이스(정의) – 컬레버레이션(구현)
  - 인터페이스(정의) – 컴포넌트(구현)
  - 인터페이스(정의) – 클래스(구현)



Copyright © 2009 Nextree Soft, All rights reserved

작성 및 주의사항 (1/2)

❑ 컴포넌트 다이어그램 작성 단계

- 컴포넌트 대상 정의
- 컴포넌트 식별
- 컴포넌트 배치
- 의존관계 정의

단계	내용
컴포넌트 대상 정의	컴포넌트 다이어그램을 그리기 전에 무엇을 컴포넌트로 표현할 지 클래스를 구성요소로 하는 실행모듈로 할지, 소스코드를 정의할 지 기타 무엇을 컴포넌트로 표현할 지를 정해야 한다.
컴포넌트 식별	컴포넌트 다이어그램에 등장할 컴포넌트를 정한다. 소스파일일 경우 그 대상은 쉽게 식별되지만 실행모듈일 경우 간단하지 않으므로 여러 가지 방법으로 컴포넌트를 식별해 내는 작업을 수행해야 한다.
컴포넌트 배치	컴포넌트 다이어그램에 컴포넌트를 배치하고 이름을 정의한다. 인터페이스가 필요한 경우 인터페이스를 정의하고 컴포넌트와 실체화 관계로 연결한다.
의존관계 정의	컴포넌트 간 의존관계를 분석하여 Dependency를 정의한다.

Copyright © 2009 Nextree Soft, All rights reserved

작성 및 주의사항 (2/2)

❑ 컴포넌트 다이어그램 작성 시 주의사항

- 컴포넌트는 응집도는 높고 결합도는 낮은 단위로 정의
- 컴포넌트 크기*Granularity* 일관성 유지
- 추상화 수준에 맞는 상세성을 일관되게 제공
- 목적을 전달할 수 있는 명칭 부여

순서	내용
응집도와 결합도	실행모듈로서의 컴포넌트를 식별할 때, 컴포넌트는 다른 컴포넌트와 독립적이고 기능 차별성을 갖추는 단위로 정의되어야 합니다. 즉, 기능 측면에서 컴포넌트 내부는 강한 유사성을 갖는 단위들로 구성되어야 하고(높은 응집도), 다른 컴포넌트에 강하게 종속되지는 않는(낮은 결합도) 단위로 정의되어야 한다.
크기	한 시스템에서 컴포넌트의 크기에 너무 차이가 나면 바람직하지 않으므로 컴포넌트의 크기는 기술구조와 시스템 특성들이 고려되어 적절한 크기로 정의해야 하며, 그 크기도 되도록 많이 차이 나지 않도록 해야 한다.
상세성	모든 모델이 마찬가지 이지만 한 장의 모델에는 동일한 상세화 레벨이 유지하고 서로 다른 추상화 레벨의 컴포넌트가 섞여 있으면 의미 파악이 어려우며 소스와 실행모듈을 한 장에 정의한 컴포넌트는 좋은 예가 아니다.
명칭	모호한 명칭으로 정의하면 혼란만 야기 시키는 결과가 나오므로 명확한 표현을 사용해야 한다.

Copyright © 2009 Nextree Soft, All rights reserved

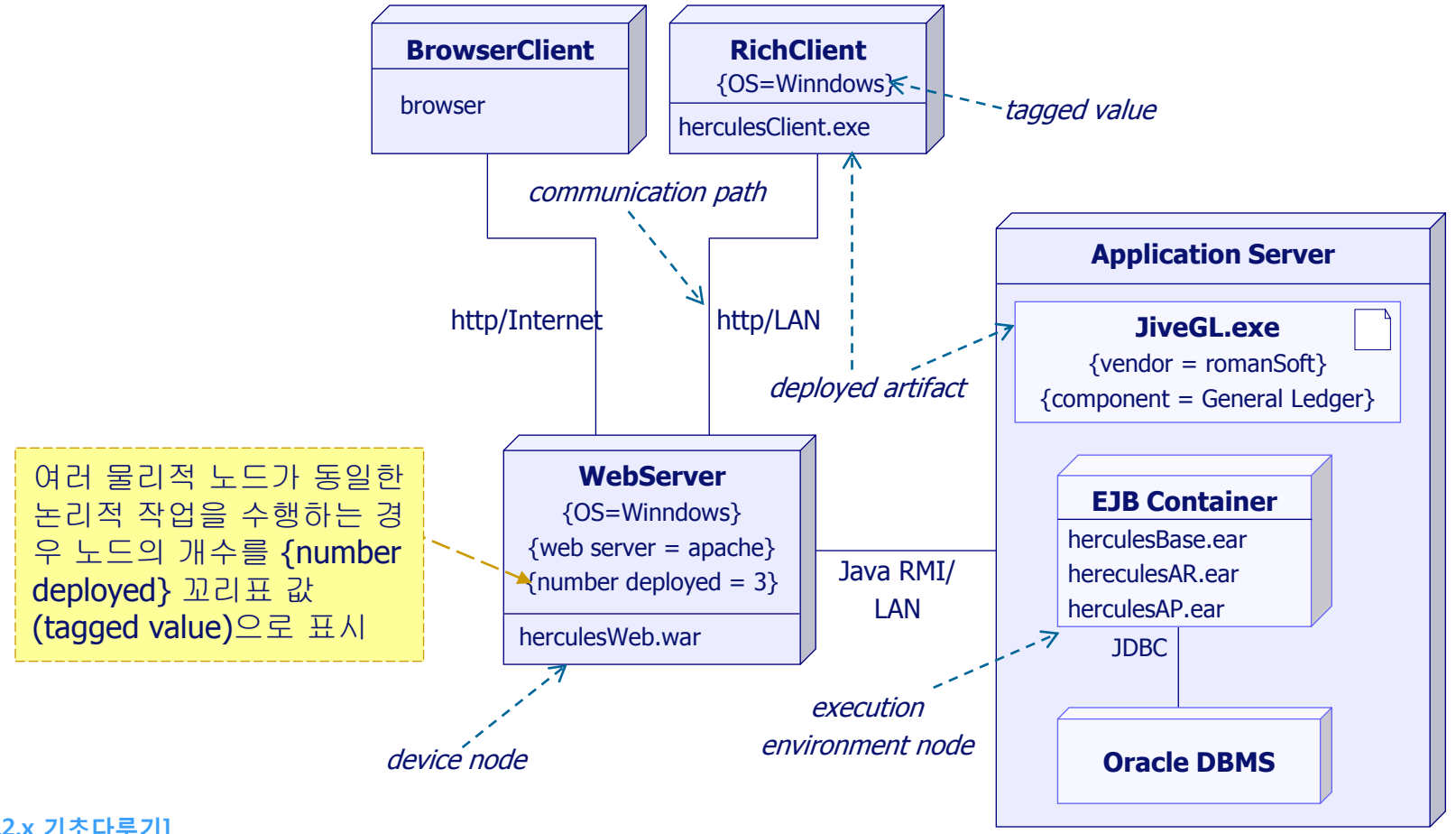
4. 배치 다이어그램

- ❑ 개요
- ❑ 구성요소
- ❑ 작성 및 주의사항

Copyright © 2009 Nextree Soft, All rights reserved

개요 (1/2)

- 시스템의 물리적인 레이아웃 표현
- 어떤 소프트웨어 부분이 어떤 하드웨어 상에서 실행되는가를 표현



Copyright © 2009 Nextree Soft, All rights reserved

개요 (2/2)

❑ 작성목적

- SW 시스템이 배치 및 실행될 HW 자원 등을 정의
- SW 컴포넌트가 어떤 HW 자원에 탑재되어 실행될지 정의
- HW 자원의 물리적인 구성을 정의

❑ 작성시기

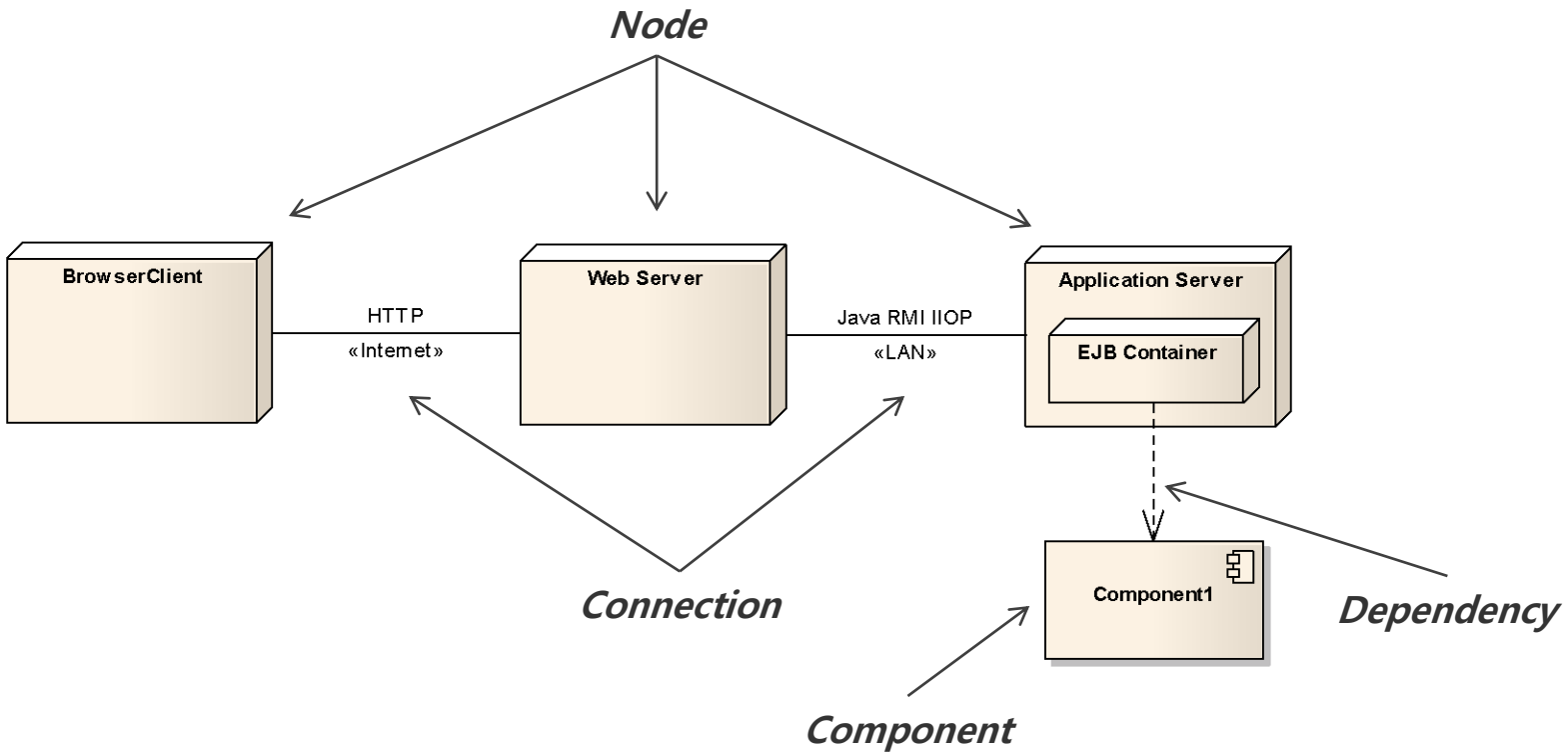
- 시스템의 설계 단계 마지막에 작성
- 모든 설계가 마무리되어 SW 컴포넌트가 정의되고 시스템의 HW 사양도 확정된 후 작성

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (1/5)

❑ 배치 다이어그램의 구성요소

- 요소: 노드*Node*, 컴포넌트*Component*
- 관계: 연결*Connection*, 의존성*Dependency*



Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (2/5)

❑ 노드*Node*

- 소프트웨어를 수용할 수 있는 것
- 디바이스 노드(device node)
  - 하드웨어, 혹은 시스템에 연결된 하드웨어의 간단한 부분
- 실행 환경 노드(execution environment node)
  - 그 자신이 다른 소프트웨어를 호스트하거나 포함하는 소프트웨어
  - 운영체제 또는 컨테이너 프로세스
- 배치되는 산출물(artifact) 포함
  - 소프트웨어의 물리적인 표현, 일반적으로 파일
  - 클래스 박스 또는 노드 내에 목록을 나열하여 표시
- 꼬리표 값(tagged value) : 노드(node) 또는 산출물(artifact)에 추가 정보 표시

HW 장비의 예)	노드의 예)
Printer, Card Reader Communication Device 등	Web Server, DB Server

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (3/5)

❑ 컴포넌트*Component*

- 독립적으로 배포되고 교체되며 재사용될 수 있는 SW 조각
- 소스코드, User Interface, 분석, 설계 산출물 등을 포함한 광의적 정의
- 용어를 표현하는 사람의 의도에 따라 달라지므로 컨텍스트 파악이 중요

컴포넌트 예)

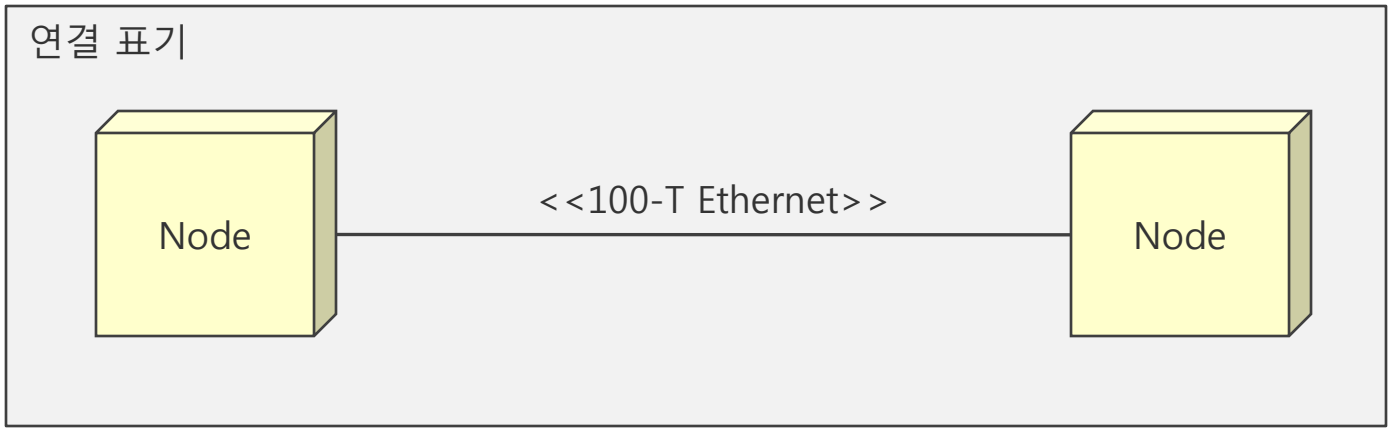
보험시스템	고객, 사원, 계약, 상품 등
오픈 마켓 시스템	신용카드 결재, 상품, 우편번호 검색 등

Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (4/5)

❑ 연결*Connection*

- 두 노드 사이의 물리적인 연결을 의미하고 특성을 설명
- 노드를 연결하는 실선으로 연결의 물리적 특성을 스테레오 타입으로 표기

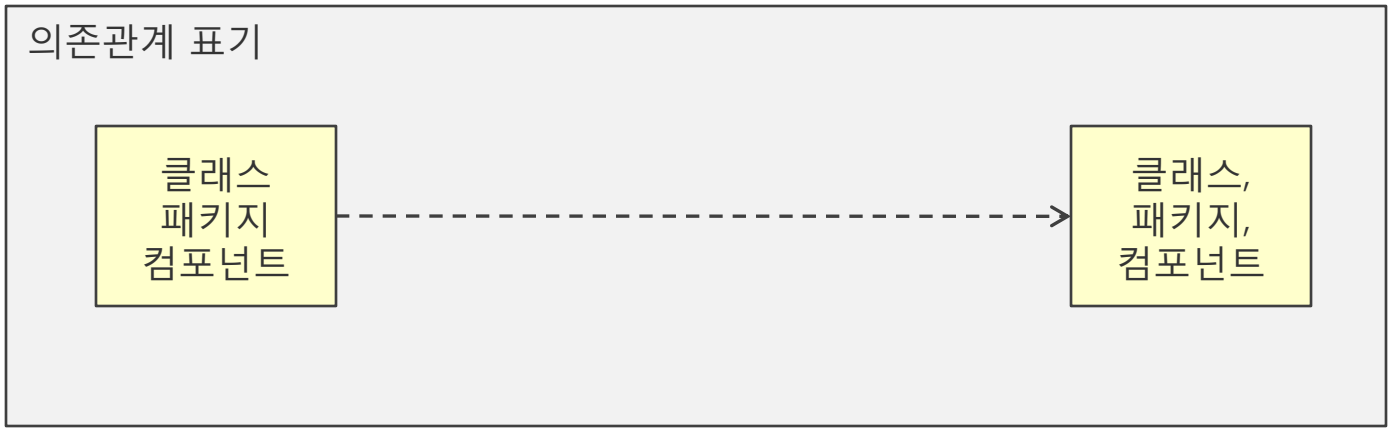


Copyright © 2009 Nextree Soft, All rights reserved

구성요소 (5/5)

❑ 의존관계*Dependency*

- 객체나 컴포넌트가 다른 객체나 컴포넌트의 실행을 요구할 경우 실행 또는 참조관계 표현
- 클래스, 패키지, 컴포넌트 사이에 주로 사용되는 관계
- 때로는 클래스 – 패키지 – 컴포넌트 상호 간에도 사용



Copyright © 2009 Nextree Soft, All rights reserved

작성 및 주의사항 (1/2)

❑ 컴포넌트 다이어그램 작성 단계

- 노드를 식별하여 정의
- 컴포넌트 식별
- 노드 간의 구성관계를 정의
- 노드에 컴포넌트 배치

단계	내용
노드 식별 정의	배치 다이어그램을 작성할 때 시스템의 운영을 위한 HW자원을 식별하고 그 사양을 확인하는 것을 가장 먼저 수행한다. 프로젝트 초기에 시스템 청사진을 작성하는 것을 기반으로 HW자원을 식별한다.
컴포넌트 식별	배치 다이어그램에 사용할 컴포넌트를 정의한다. 컴포넌트 다이어그램이 있을 경우 활용하여 수행한다.
노드 간 구성관계	배치 다이어그램에 노드를 배치하고 노드 간의 물리적 연결인 연결을 정의한다. 연결과 노드에는 스테레오 타입으로 하드웨어 특성을 표현한다.
컴포넌트 배치	정의된 노드와 연결을 고려하여 어떤 노드에서 컴포넌트를 실행하게 될 것인가를 정의한다.배치 다이어그램에 SW 컴포넌트의 배치 상황을 반영한다.

Copyright © 2009 Nextree Soft, All rights reserved

작성 및 주의사항 (2/2)

❑ 컴포넌트 다이어그램 작성 시 주의사항

- 목적을 전달할 수 있는 명확한 의미의 명칭 부여
- 문제 영역의 HW에 대한 명쾌한 추상개념을 제공
- 모델을 만든 목적을 전달하기에 필요한 수준까지만 분해

순서	내용
명확한 의미의 명칭	노드 명과 스테레오 타입으로 정의하는 하드웨어 특성 등은 표현 방식에 기준이 없으므로 시스템과 관련이 없는 담당자가 보더라도 그 의미를 이해하기 쉽고 명확한 용어를 사용하여 명칭을 정의한다.
명쾌한 추상개념	SW 자원이 탑재되어 운영되는 보조적인 용도 뿐만 아니라 배치 다이어그램은 시스템의 하드웨어 구성을 개념적으로 보여주는 도구로 활용된다. 이러한 용도를 살려 HW 자원의 구성에 대한 좋은 모델이 되도록 정의한다.
목적에 맞는 수준으로 분해	배치 다이어그램에 모든 HW 장비가 나타날 필요는 없다. 모든 HW 장비가 표현된다면 오히려 다이어그램이 장황하고 복잡하게 만들어져 의미 파악이 어려워지므로 목적과 용도에 부합하는 요소들만 정의한다.

Copyright © 2009 Nextree Soft, All rights reserved

5. 패키지 다이어그램

- ❑ 패키지 다이어그램 개요
- ❑ 패키지와 의존성

Copyright © 2009 Nextree Soft, All rights reserved

패키지 다이어그램(1/3)

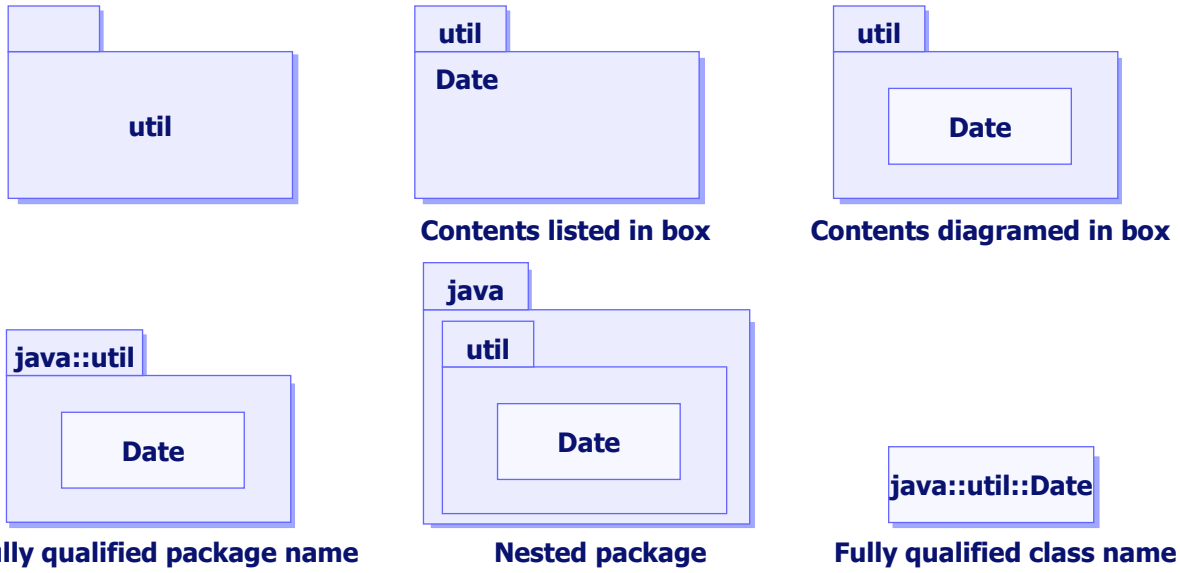
❑ 패키지 개요

- 그룹핑 구성체(Grouping Constructor)
  - 임의의 UML 요소를 취하여 더 상위 레벨 단위로 모으기 위한 그룹핑 요소
  - 클래스 모임을 구조화하기 위해 가장 많이 사용
  - 패키지는 다른 패키지 포함 가능
- 이름 영역(Namespace)
  - 모든 클래스는 포함된 패키지 내에서 유일한 이름 보유
  - 완전한 경로명(fully qualified name)
    - 포함 패키지의 구조도 함께 표시
    - System::Date, MartinFowler::Date
- Java의 패키지(package), C++과 .NET의 이름영역(namespace)과 대응

Copyright © 2009 Nextree Soft, All rights reserved

패키지 다이어그램(2/3)

❑ 패키지 표기법



❑ 패키지 가시성(Package Visibility)

- 패키지 내의 클래스 가시성은 public 또는 private
- public 클래스는 패키지 인터페이스의 일부
  - 다른 패키지의 클래스에 의해 사용 가능
- 패키지 가시성은 사용중인 언어의 규칙을 준수

Copyright © 2009 Nextree Soft, All rights reserved

패키지 다이어그램(3/3)

- **Façade[Gang of Four]**
  - 인터페이스 축소
    - 패키지 내의 public 클래스들과 연관된 오퍼레이션들의 작은 집합만을 반출
  - 방법
    - 모든 클래스에 private 가시성 부여
    - 공용 행동을 위해 별도의 public 클래스 선언
    - Façade 클래스는 요청을 내부의 클래스로 위임
- **패키지 구성 원칙**
  - CCP(Common Closure Principle)
    - 패키지 내의 클래스들은 비슷한 원인에 따라 변경되어야 함
  - CRP(Common Reuse Principle)
    - 패키지 내의 클래스들은 함께 재사용되어야 함

Copyright © 2009 Nextree Soft, All rights reserved

패키지와 의존성(1/3)

❑ 패키지 의존성과 패키지 다이어그램

- 패키지 다이어그램은 패키지와 패키지간의 의존성 표현
- 패키지간 의존성은 패키지 내부 요소간 의존성의 요약
  - 한 패키지의 클래스가 다른 패키지의 클래스에 의존할 경우 패키지간 의존성 존재
- 대형 시스템의 구조 제어를 위한 가치 있는 도구

❑ 패키지 의존성 제어 원칙

- ADP(Acyclic Dependency Principle)
  - 패키지간 의존성은 비순환적이어야 함
  - 절대적인 규칙은 아님
    - ➢ 의존성은 지역화
    - ➢ 레이어에 걸친 순환은 불가
- SDP(Stable Dependency Principle)
  - 의존성이 증가할 수록 인터페이스가 더 안정적이어야 함
  - 인터페이스에 대한 변경은 의존하고 있는 모든 클래스에 파급됨
- SAP(Stable Abstractions Principle)
  - 더 안정적인 패키지는 더 많은 비율의 인터페이스와 추상 클래스 보유

Copyright © 2009 Nextree Soft, All rights reserved

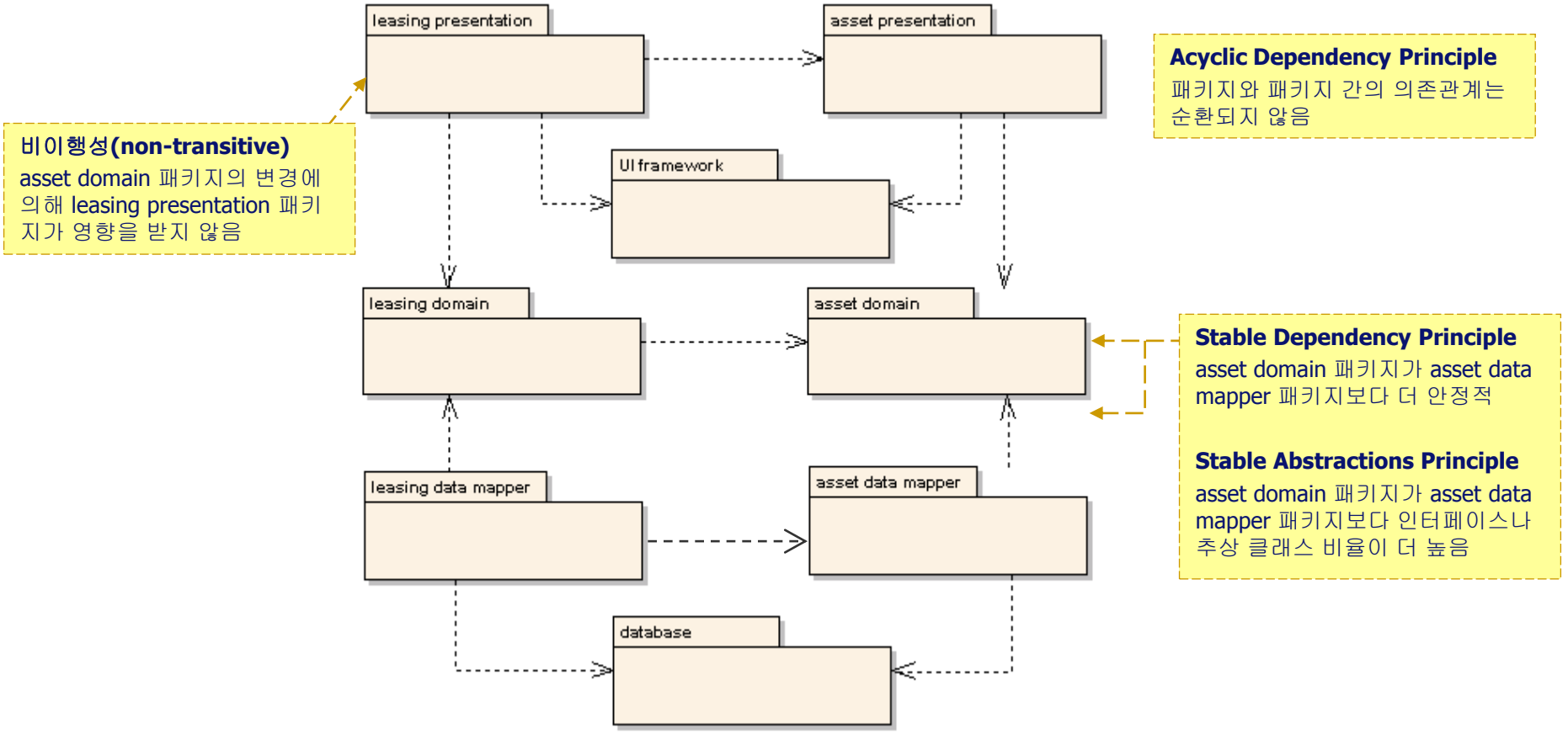
패키지와 의존성(2/3)

- **비이행적 패키지 의존성**
  - 패키지 의존성은 비이행적(non-transitive)
  - 의존하지 않는 패키지로의 변경 파급 방지
- **전역 패키지**
  - 거의 모든 패키지에서 사용되는 패키지
  - 다이어그램에 너무 많은 의존성이 표시되는 것을 방지
  - <<global>> 키워드 사용
- **패키지 다이어그램의 용도**
  - 대형 시스템의 주요 요소들 간의 의존성 관계를 표현
  - 일반적인 프로그래밍 구조 표현
  - 어플리케이션의 의존성 제어
  - 컴파일 시간 그룹핑 메커니즘 표현

Copyright © 2009 Nextree Soft, All rights reserved

패키지와 의존성(3/3)

❑ 패키지 다이어그램 예



Copyright © 2009 Nextree Soft, All rights reserved