

객체 모델링 (Object Modeling)

객체 모델링(Object Modeling)

1. Object Modeling ?

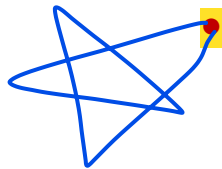
1) 개요

- Definition
 - 논리적 관점 및 정적인 관점에서 존재하는 객체들과 객체간의 관계를 표현하는 것
- Characteristics
 - 개발중인 시스템의 개념 도출
 - Class, Attribute, Operation 등 도출
 - Component Modeling의 기초
 - 실행시스템을 구축하는데 필수
- Modeling 유형
 - Concept(Domain) Modeling
 - Initial Modeling
 - Detail Modeling

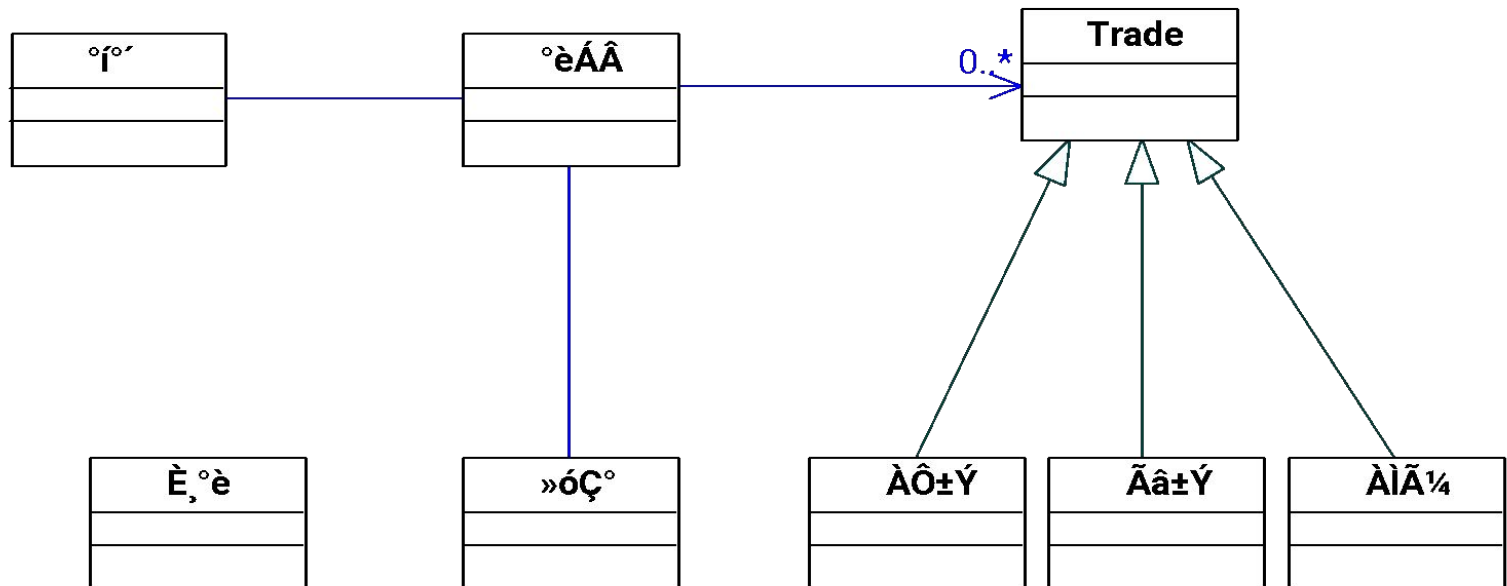
객체 모델링(Object Modeling)

2) Concept(Domain) Modeling

- 개발범위를 확정할 목적으로 업무관점으로 작성
- Requirement와의 접점으로 사용자와의 의사소통을 위해 개략적 업무범위 표기
- 관련 도메인의 개념 Class 또는 실세계 객체들의 시각적 표현



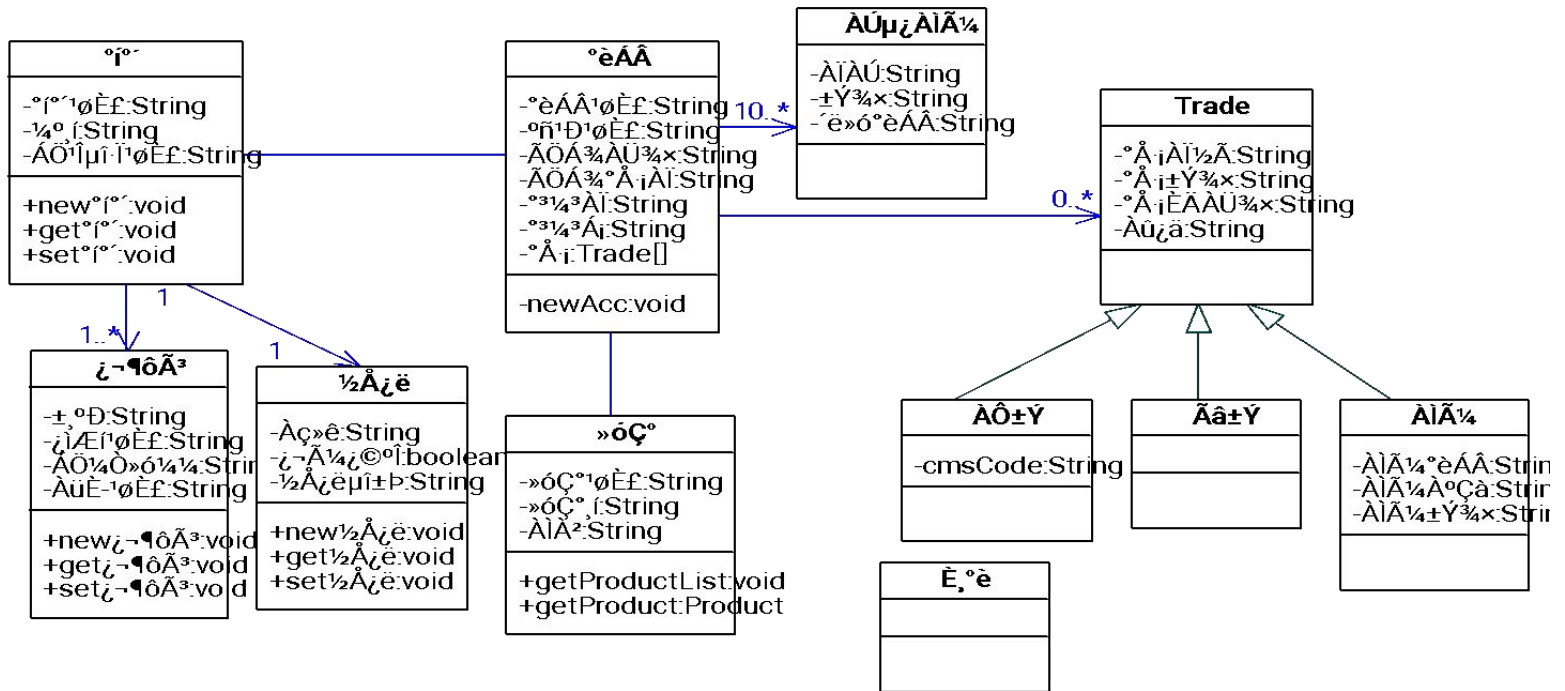
요구사항명세서, 도메인모델, Usecase모델, Usecase 및 시나리오 분석을 통하여 개념 Class를 도출한 후, Attribute과 관계를 식별하여 기술



객체 모델링(Object Modeling)

3) Initial Modeling

- 주요 Class 식별
- 식별가능한 Attribute, Operation 포함

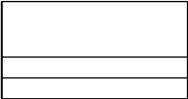


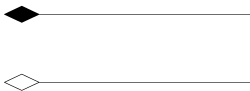




4) Detail Modeling

- 명확한 Class, Attribute, Operation 식별

객체 모델링(Object Modeling)

2. Elements & Relationships

Classification	Construct	Description	Syntax
Elements	class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
	interface	a named set of operations that characterize the behavior of an element.	
Relationships	association	a relationship between two or more classifiers that involves connections among their instances.	
	aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
	Generalization	a taxonomic relationship between a more general and a more specific element.	
	dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

객체 모델링(Object Modeling)

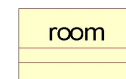
3. Class

1) Definition

- Class는 공통의 Structure(Attribute)와 Behavior(Operation), Relationship, Syntax를 갖는 객체들을 기술한 실체

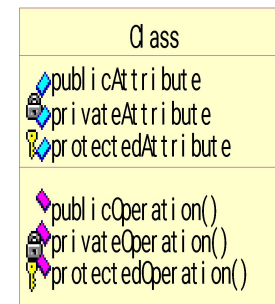
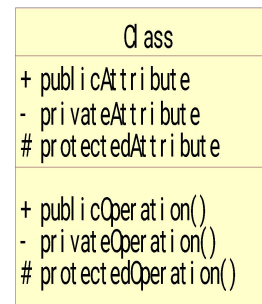
2) Notation

- 3 부분으로 된 직사각형으로 표시
 - 순서대로 Name, Attribute, Operation을 표기
 - 관점에 따라 Attribute, Operation 생략 가능



3) Characteristics

- Class는 Domain 용어를 사용하여 명명표준에 의해 명명
- Class는 Sequence Diagram과 Collaboration Diagram에 있는 객체들을 조사하여 보완
- Visibility
 - Class는 물론, 보유하고 있는 Attribute, Operation도 나타낼 수 있는 가시성
 - UML 표준 Notation
 - » +(Public), -(Private), #(Protected)
 - » CASE Tool에 따라 Icon화 하여 사용하기도 함

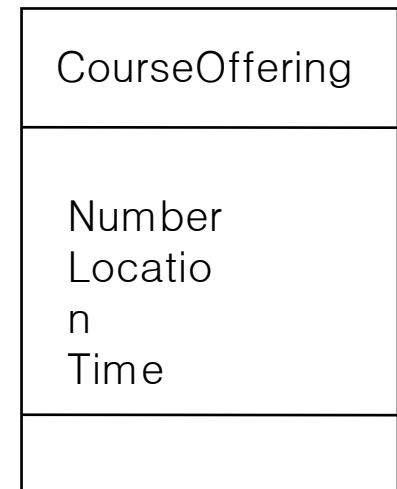
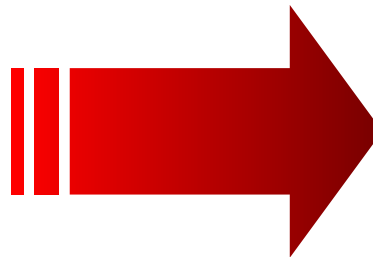


객체 모델링(Object Modeling)

4) Attribute

- Class 구조는 Attribute에 의하여 표현
- 객체의 논리적인 자료 값
- 요구사항이 정보를 기억해야 할 필요가 있을 때 식별
- Class 정의서와, 문제 요구사항 조사, Domain 지식을 기반으로 파악
- Syntax
 - Visibility Name : Type = DefaultValue
 - (ex) – xPosition : int = 0;

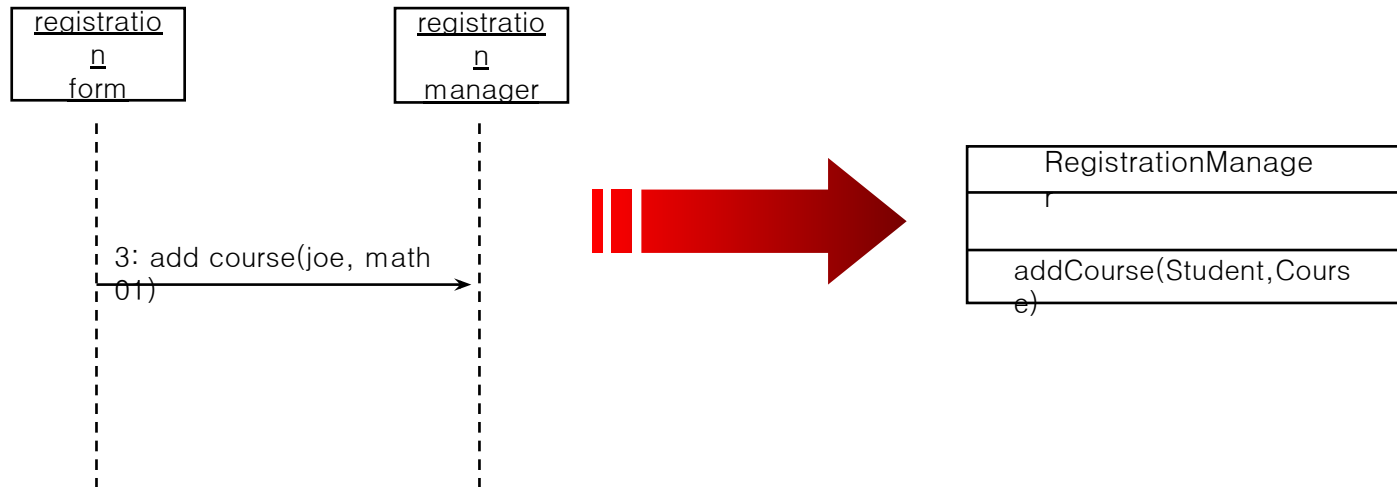
각 Course는 번호, 강의장,
시간이 정의되어진다



객체 모델링(Object Modeling)

5) Operation

- Class의 행위는 Operation에 의하여 표현
- Operation들은 Interaction Diagram을 조사하여 보완
- Syntax
 - Visibility Name (ParameterList) : Return Type {Property}
 - (ex) + changeColor (ColorType c) : Boolean
 - Return Type : optional.



4. Relationship

1) Characteristics

- Relationship은 객체들간의 의사소통을 위한 통로를 제공
- 두 객체가 서로 대화할 필요가 있다면, 그들 간에는 연결선이 있어야 함
- 어떤 행위를 달성하기 위하여 객체들간에 무슨 연결선 들이 존재할 필요가 있는지를 결정하기 위하여 Interaction(Sequence, Collaboration) Diagram 활용

2) Relationship 식별시 고려사항

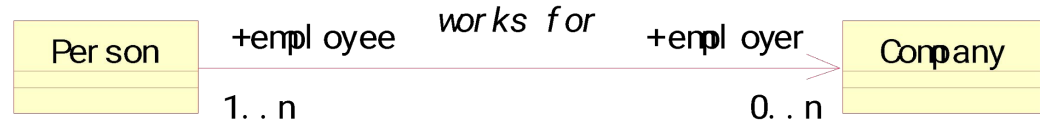
- 연관관계에 대한 지식을 일정기간 동안 보존할 필요가 있는 연관관계에 초점을 맞추어 식별
- 너무 많은 연관관계는 Domain 모델을 오히려 혼란스럽게 만들 수 있음

3) Association 유형

- Association
 - Role, Multiplicity, Navigability
- Aggregation & Composition
- Generalization
- Dependency
- Realization

객체 모델링(Object Modeling)

4) Association (1/3)



- Definition

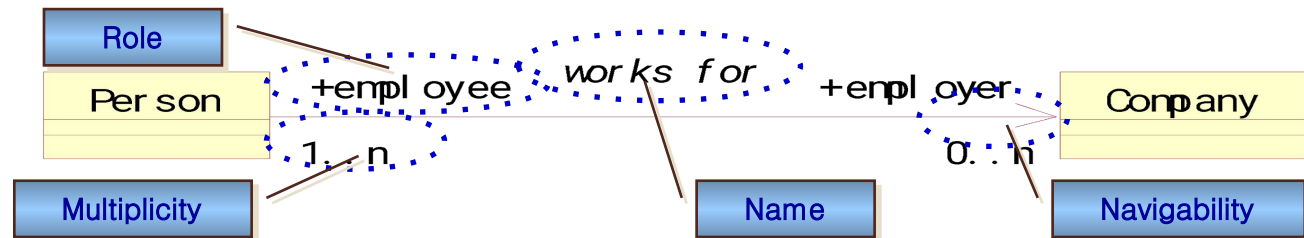
- 구조적 관계로서, 객체와 객체간 물리적 연결이 존재함을 의미
- Class들간의 양방향 연결

- Characteristics

- Name, Role, Multiplicity, Navigability와 같이 구체적으로 설명하기 위한 장식을 가짐
- 구현 시 Association이 Attribute화 될 가능성이 높은 물리적 관계
- Whole 또는 Part와 같은 개념을 상세화 하기 위해 Aggregation 또는 Composition 으로 확장될 수 있음

- Elements

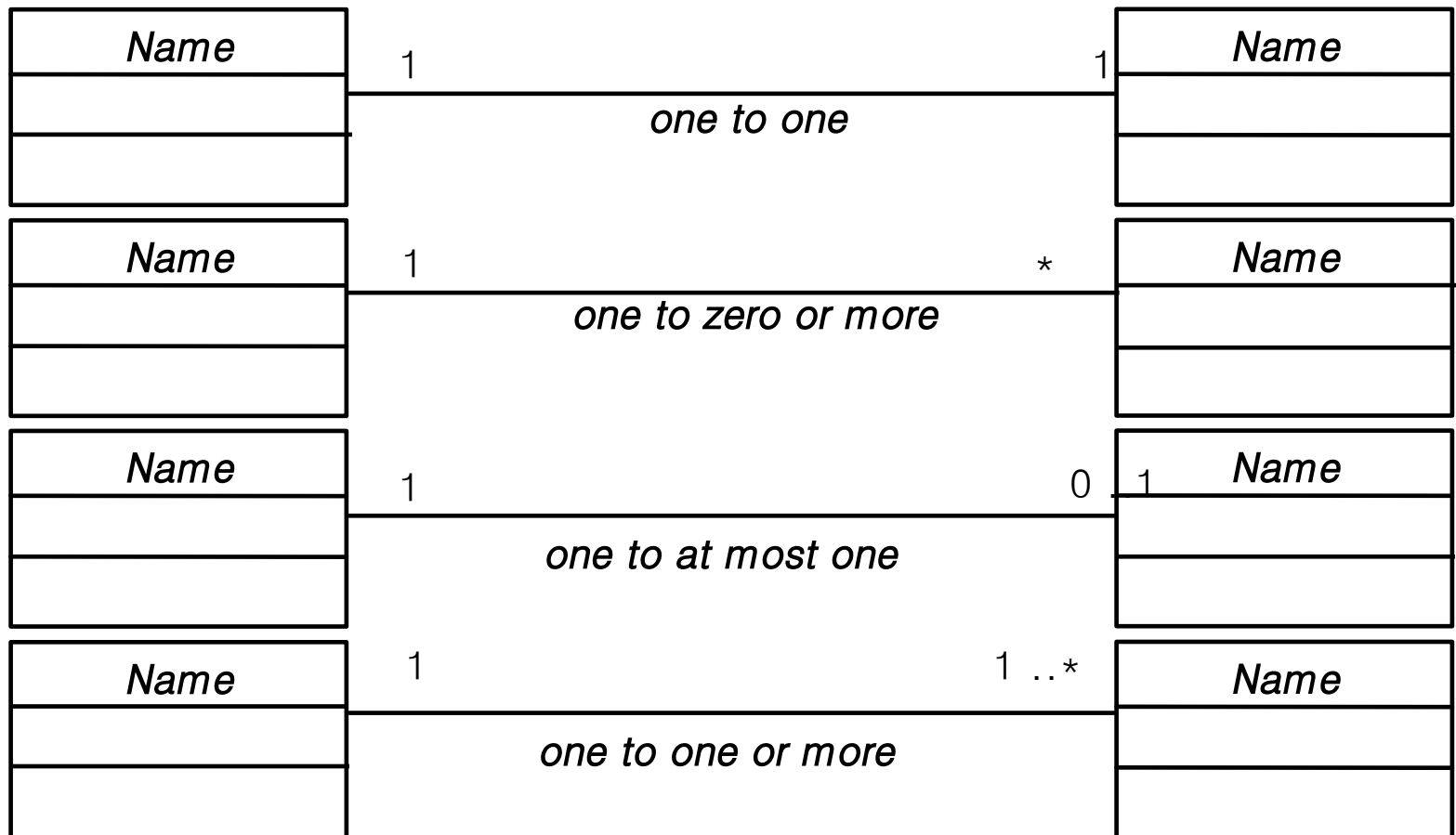
- Name : Association이 의미하는 뜻
- Role : Association에서 각 객체가 가지는 역할
- Multiplicity
 - . 관계에 얼마나 많은 객체들이 참여하는지를 정의하는 것
 - . 각 객체가 상대 객체와 몇 개의 인스턴스와 관련되어 있을 지의 다중성
- Navigability : 메시지의 흐름 및 객체간의 방향성 및 종속관계



객체 모델링(Object Modeling)

4) Association (2/3)

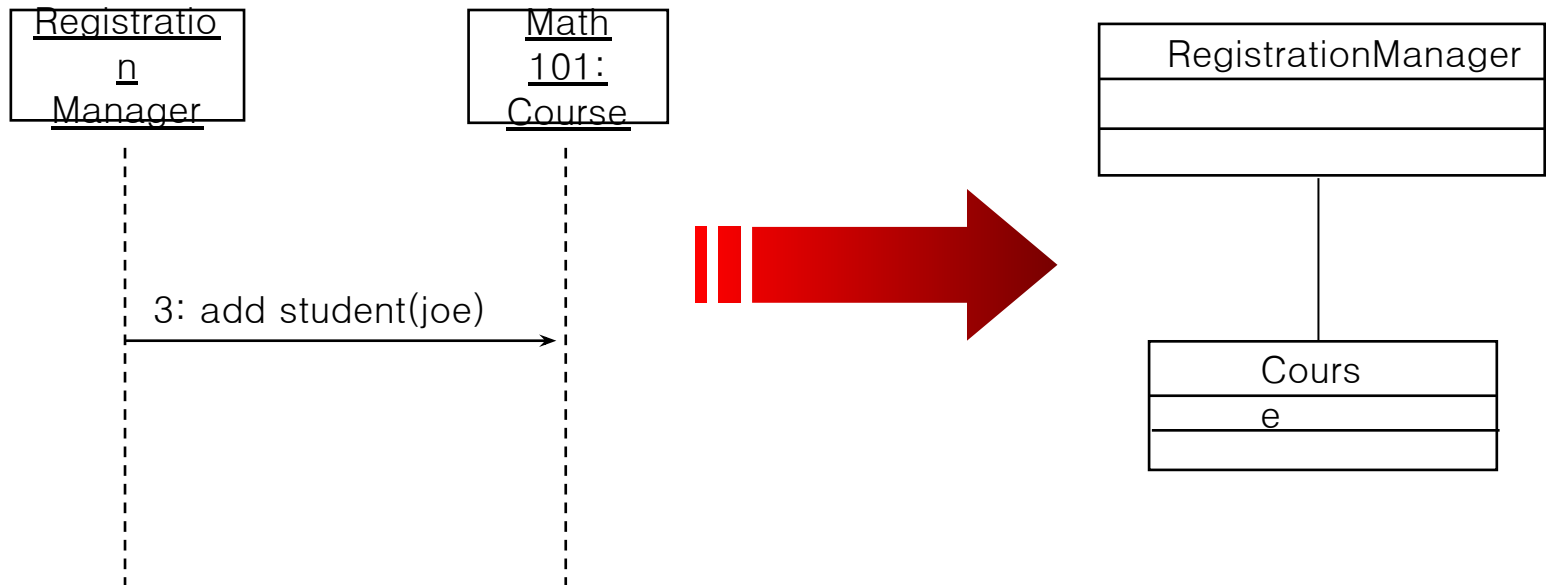
- Multiplicity



객체 모델링(Object Modeling)

4) Association (3/3)

- Interaction Diagram 조사함으로써 보완 검출
- 만약 두 객체가 서로 대화할 필요가 있다면 그들 간에는 연결선이 존재



객체 모델링(Object Modeling)

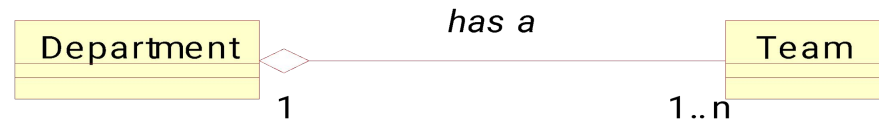
5) Aggregation

- Definition

- Association에서 확장된 개념으로, 객체와 객체가 부분의 관계로 연결 되었음을 의미
- Aggregation은 관계가 부분으로 맺어질 때 사용되는 강력한 형태의 관계
- 개념적인 의미로, 소멸에 대한 권한관계 없음

- Characteristics

- 소멸의 권한 등을 기술하여야 할 필요가 있을 경우에는 Composition 관계를 사용
- 전체를 나타내는 Class에 다이아몬드 형태를 가진 한 Line으로 표현하고, “has a” 관계로 사용될 수 있음



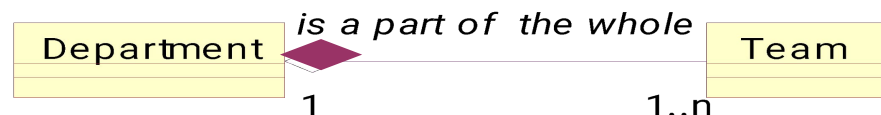
7) Composition

- Definition

- Association과 Aggregation에서 확장된 개념으로, 객체와 객체가 전체의 관계로 연결 되었음을 의미
- 물리적인 의미로, 소멸에 관한 권한관계 있음

- Characteristics

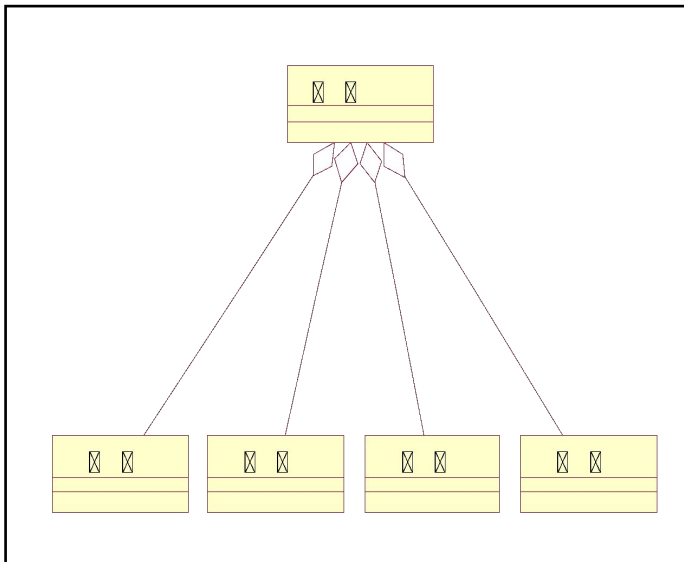
- 전체가 소멸될 경우 부분이 명시적으로 소멸되어야 할 정도로 Aggregation보다는 강력한 소유권을 가지고 있다는 의미
- “is a part of” 관계로 사용될 수 있음



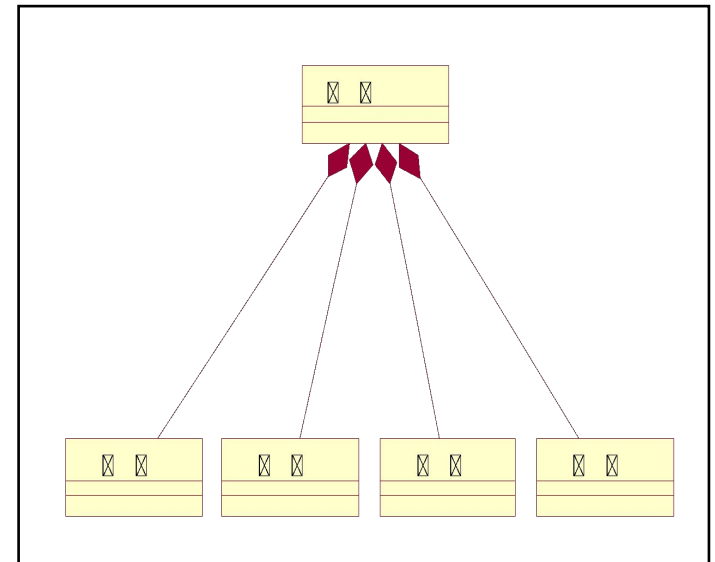
객체 모델링(Object Modeling)

7) Aggregation vs Composition

- Aggregation은 소유에 대한 권한이 없는데 반해 Composition은 소유에 대한 권한이 있음
- Aggregation은 Association(Reference)과 비슷하고 Composition은 Attribute(Value)와 비슷함
- 같은 사물도 Domain에 따라 다르게 표현될 수 있음
- 아래 Class Model은 관계의 유형에 따라 의미가 다르게 표현



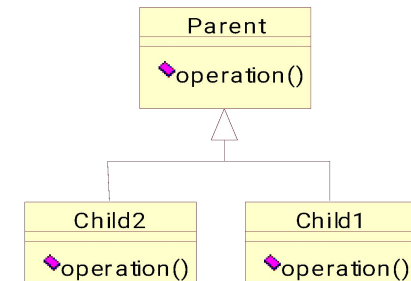
or



객체 모델링(Object Modeling)

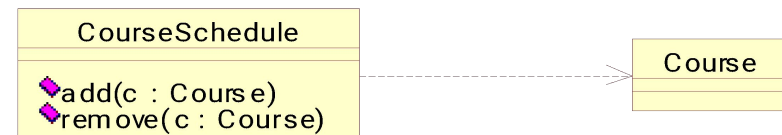
8) Generalization

- Definition
 - 일반화된 Class(SuperClass)와 좀 더 특수화된 Class(Sub Class)사이의 관계를 의미
- Characteristics
 - “is a kind of” 관계로 사용될 수 있음
 - Multiplicity, Role, Navigability는 무의미



9) Dependency

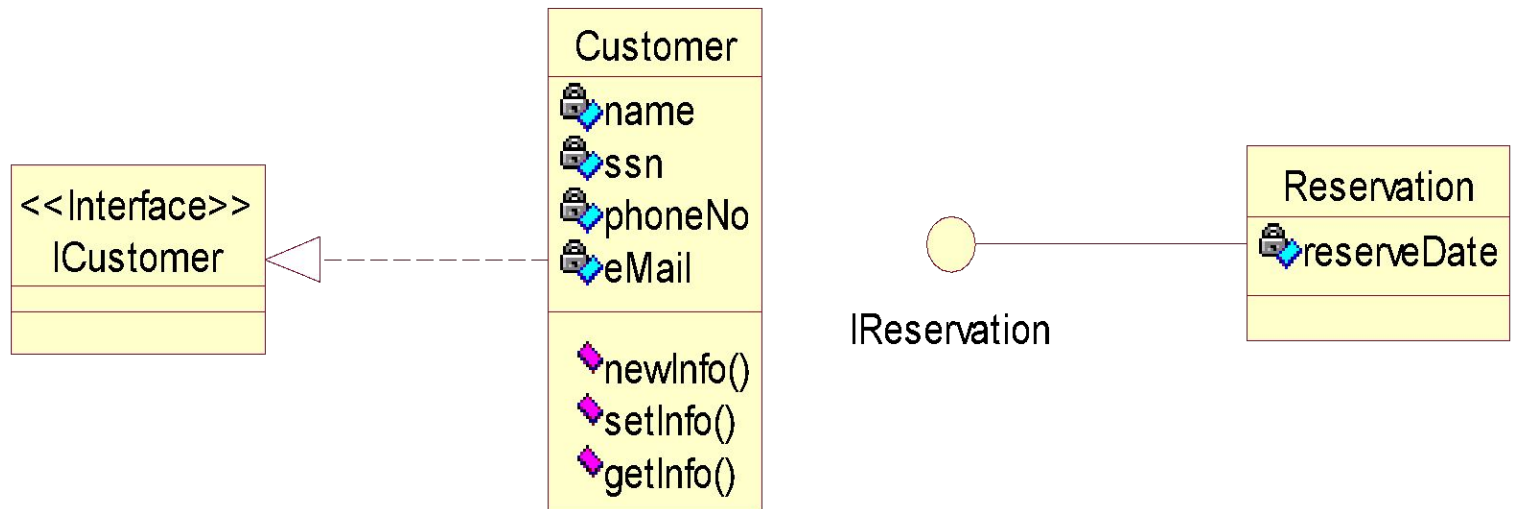
- Definition
 - 의미적 관계로서, 한 Class 명세서가 바뀌면 그것을 사용하는 다른 Class에게 영향을 끼치는 경우의 관계를 의미
 - Client가 Supplier에 대하여 의미적인 지식을 갖지 않는 약한 형태의 관계
- Characteristics
 - Association과 다른 점은 물리적인 관계가 없다는 점
 - 한 객체가 다른 객체를 Operation 인자로 사용하는 경우에 표현
 - 한 객체에서 다른 객체 Operation을 호출하는 경우에 표현
 - Client에서 공급자 방향으로 화살표를 갖는 점선으로 표현



객체 모델링(Object Modeling)

10) Realization

- Definition
 - 의미적 관계와 일반화/특수화 관계를 절충한 개념으로써, 한 Classifier가 반드시 수행해야 할 다른 Classifier의 계약을 지정한다는 의미
- Characteristics
 - Generalization 과 Dependency와의 혼합형
 - Multiplicity, Role, Navigability은 무의미



객체 모델링(Object Modeling)

5. Class Diagram 작성 (1/3)

- 1) Class Diagram은 Class의 존재여부 자체와 그들의 관계를 표현한 시스템의 논리적인 뷰
- 2) Class Diagram 에서의 UML 모델링 요소들
 - Class와 그들의 Structure (Attribute), Behavior (Operation)
 - Relationship
 - Association, Aggregation & Composition, Generalization, Dependency, Realization
- 3) Class 식별기법 : 명사 어구 이용 기법, Class Category List 사용
- 4) 명사 어구 이용기법
 - 문제 기술서로부터 후보 Class 식별
 - Class는 거의 명사임에 착안
 - 일부 Class는 물리적 Entity 이거나, 논리적, 개념적인 Entity
 - Domain knowledge나 현장경험을 토대로 추가 Class를 식별하여 임시 Class로 확정
 - 부적절한 Class 제거
 - 중복 및 파생 Class, 요구사항과 무관한 Class, Vague Class
 - 다른 Class의 Attribute인 명사
 - 명사이긴 하지만 의미하는 바가 Operation 인 경우
 - 객체가 수행하는 역할인 경우
 - 구현기술과 관련된 경우



객체 모델링(Object Modeling)

5. Class Diagram 작성 (2/3)

5) Class Category List 사용기법

Class Category	Sample
물리적, 유형 객체	Register, Airplane
사물의 명세, 설계, 기술	ProductSpec, FlightDescription
장소	Store, Airport
트랜잭션	Sale, Payment
트랜잭션 라인 아이템	SalesLineItem
사람의 역할	Cashier, Pilot
다른 사물의 컨테이너	Store, Bin, Airplane
시스템 외부의 컴퓨터	AirTrafficControl
추상 명사 개념	Hunger, Acrophobia
조직	SalesDepartment
이벤트	Sale, Meeting,

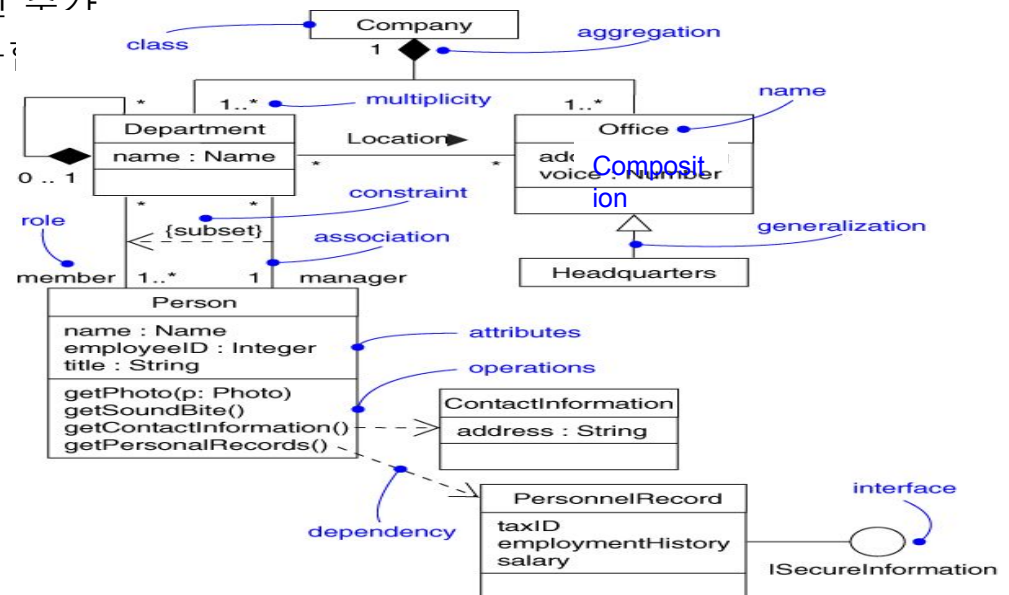
Class Category	Sample
프로세스	SellingAProduct
룰과 정책	RefundPolicy
카탈로그	ProductCatalog
재정, 작업, 계약, 법적 사건 기록	Receipt, Ledger
재정 도구 및 서비스	LineOfCredit, Stock
매뉴얼, 문서, 참조 문서, 책	DailyPriceChageList, RepairManual

객체 모델링(Object Modeling)

5. Class Diagram 작성 (3/3)

6) 작성 절차

- Usecase Description 또는 Concept Model을 통해 Class 파악
- Association을 식별하고 Multiplicity & Navigability 정의
- State 정의가 필요할 경우 StateChart Diagram 작성
- Attribute & Operation, Visibility 정의
- Aggregation & Composition 관계 정의
- Generalization 정의, Dependency 관계 정의
- Derived Association & Attribute 식별 : '/' 사용, 중복 및 효율성 고려
- 일반적 비기능적 요구사항 조건 추가
- Class간의 유사성 등을 고려 통합
- Class Diagram 검증 및 확정



객체 모델링(Object Modeling)

6. StateChart Diagram 작성

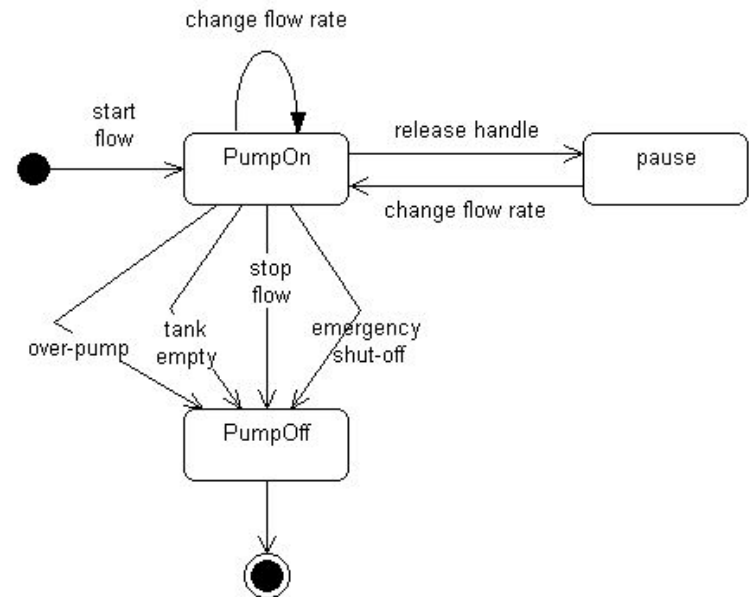
1) Characteristics

- 사건 요구에 따른 Object가 취할 수 있는 행동에 초점을 두고 상태들 사이의 전이를 강조
- 반응 시스템 모형 작성에 유용
- 중요한 동적 행위를 갖는 객체들을 위하여 작성 하므로 모든 객체들에 대하여 다 작성할 필요는 없음
- Operation의 행위는 수신 객체의 상태에 따라 달라짐

2) 다음과 같은 사항을 표현

- 하나의 상태에서 다른 상태로의 전이를 유발하는 Event
- 주어진 Class의 Life history
- 상태 변화의 결과로 인한 행동
- 도식방안

- 가능하면 Transition이 교차하지 않도록 작성
- 복합 상태를 확장할 때는 Diagram을 이해하기 쉽게 하는 범위로 한정



7. Check Points

1) Concept Model

- Domain에서 중시하는 개념 Class들이 잘 파악되었는가?
- 요구사항의 문맥에서 도메인을 이해하기 위해 필요한 핵심적인 추상화와 정보를 가지고 있는가?
- 개념 Class들간의 연관성이 잘 정의되었는가?
- 개념 Class들의 주요 Attribute이 파악되었는가?
- 프로젝트 참여자들이 개념 Class Model을 이해하는가?
- 사람들이 도메인(개념, 용어, 관계)을 쉽게 이해할 수 있는가?

2) Detail Model

- Concept(Domain)적 내용이 Class Diagram에 표현되었는가?
- Usecase Description에 표현된 내용이 Class Diagram에 반영되었는가?
- Class, Attribute, Operation, Association 등 묘사가 적절하고 이름이 충분히 의미를 전달하고 있는가?
- Class가 너무 작거나 크지 않은가?
- Class간의 통합 또는 추상화가 필요 없는가?

객체 모델링(Object Modeling)

8. 실습 (1/3)

1) Problem Statement

우리는 호텔의 Internet 예약시스템을 만들고자 한다.

이 시스템에서, 고객은 객실유형으로 객실을 예약한다.

예약 시 고객은 반드시 고객의 정보를 등록하여야 하고, 빠른 업무처리를 위하여 고객의 세부정보는 저장하도록 하여야 하며, 예약이 정상적으로 처리되면 예약번호를 고객에게 알린다.

고객은 객실예약을 취소하고자 하는 경우, 예약번호를 입력하여 객실예약을 취소한다.

호텔에는 예약담당자가 객실예약에 관한 사항을 담당하고, 객실예약 후 투숙하지 않는 건을 처리한다.

시스템은 호텔의 Check In, Check Out에 관한 업무를 처리하여야 한다.

Check In 시에 고객은 예약번호를 입력하고 객실을 배정 받는다.

Check Out 시에 고객은 객실번호를 입력하고 계산서를 발급 받는다.

직원은 고객이 투숙한 계산서에 의해 숙박비를 수납하며, 고객은 현금, 신용카드,

수표를 이용하여 숙박비를 지불 할 수 있다. 수납이 정상적으로 처리되면 직원은

영수증을 발급한다.

8. 실습(2/3)

2) Finding Classes

- Candidate Classes
- Add Classes by Domain Knowledge & Experience
- Eliminate Bad Classes
 - 중복 및 파생 Class
 - 다른 Class의 Attribute
 - Operation
 - 구현기술
 - etc
- Class 확정

8. 실습(3/3)

3) Concept Level Class Diagram

4) Detail Level Class Diagram