Introduction to UML

Team 2
200911371 김민철
200911381 김진현
200911417 정명권

Contents

1. UML이란?

UML(Unified Modeling Language)

- 요구분석, 시스템설계, 시스템 구현 등의 시스템 개발 과정에서 개발자간의 의사소통을 원활하게 이루어지게 하기 위하여 표준화한 모델링 언어로서 기존의 객체 지향 방법론과 함께 제안되어 모델링 언어 표기법의 표준화를 목적으로 한 것이다.

- 프로그램 개발 과정에서 생길 수 있는 개발자간의 의사 소통의 불일치를 해소할 수 있으며, 모델링에 대한 표현력이 강하고 비교적 모순이 적은 논리적인 표기법을 가진 언어라는 장점이 있다.

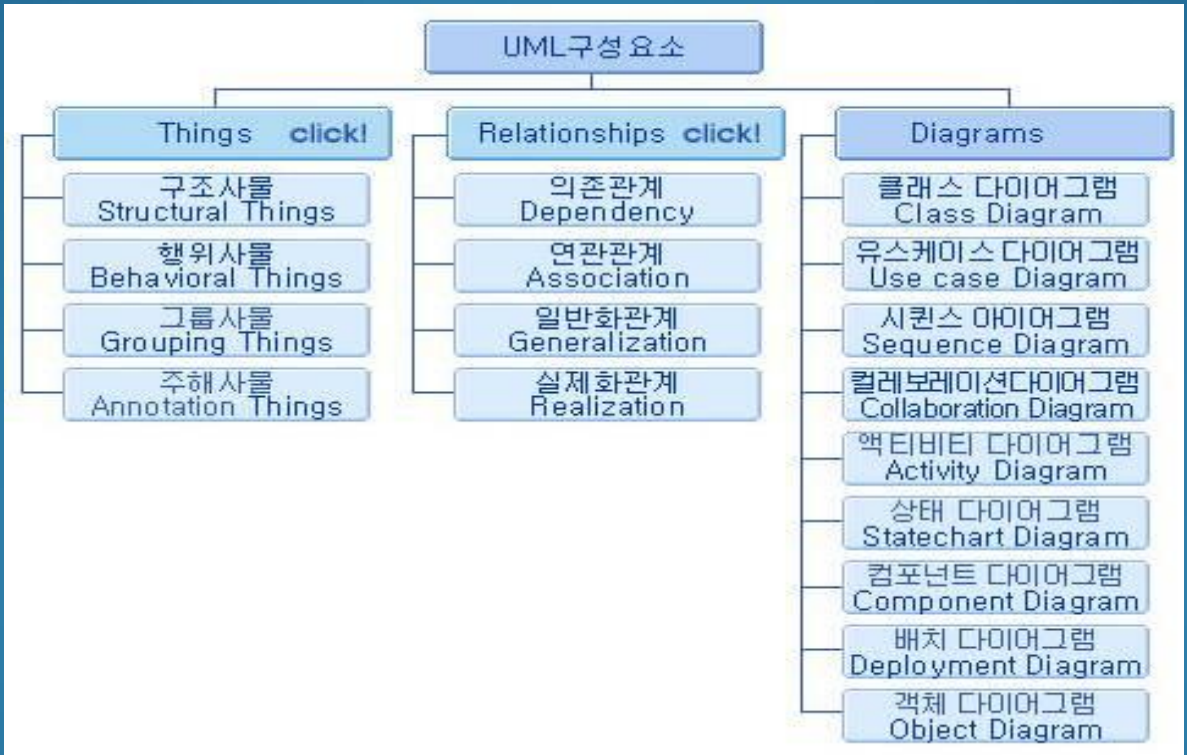
1. UML이란?

UML의 특징

1) UML은 가시화 언어이다.
   - 그래픽 형태, 심볼에 명확한 정의가 존재하고, 오류 없는 의사 소통을 가능하게 한다.
2) UML은 명세화 언어이다.
   - 분석, 설계, 구현 단계의 각 과정에서 필요한 모델을 정확하고 완전하게 명세화 할 수 있다.
3) UML은 문서화 언어이다.
   - 시스템 아키텍쳐 등 상세 내역에 대한 문서화를 다루며, 요구 사항을 표현하고 시스템을 테스트하는 언어를 제공한다.
4) UML은 구축 언어이다.
   - 다양한 프로그래밍 언어로 표현하고, UML을 소스코드로 변환 할 수 있다.

2. UML의 구성

UML은 사물과 사물과의 관계, 그리고 관계를 표현하는 다이어그램으로 구성되어 있다.



2. UML의 구성

사물(Things)

- 사물은 추상적 개념으로 모델에서 가장 중요하다.

1. 구조 사물 : 시스템의 구조 표현(클래스,인터페이스,컴포넌트,노드 등)
2. 행동 사물 : 시스템의 행위 표현
3. 그룹 사물 : 개념을 그룹화 하는 사물
4. 주해 사물 : 부가적으로 개념을 설명하는 사물

2. UML의 구성

관계(Relationships)

1) 의존(Dependency)

- 의존은 두 사물간의 의미적 관계를 나타내는 것으로 한 사물의 명세가 바뀌면 의존 관계에 있는 사물에 영향을 미치는 것
- 의존은 점선으로 나타내고, 의존하는 사물을 향하고 있다.
- 한 클래스가 다른 클래스를 Operation의 매개변수로 사용하는 경우에 주로 나타난다.

2. UML의 구성

관계(Relationships)

2) 연관(Association)

- 연관은 구조적 관계로 사물간의 연결을 의미, 한 객체가 다른 객체로 옮겨갈 수 있으며 한 연관이 한 클래스에서 순환하는 것도 가능하다.

- 연관은 클래스들을 연결하는 실선으로 표현하며, 이름, 역할, 다중성 등을 추가할 수 있다.

- 클래스가 수행하는 역할을 클래스 옆에 명시적으로 적어줄 수 있다.

2. UML의 구성

관계(Relationships)

2) 연관(Association)

① 집합 연관 : 전체와 부분 관계(has-a 관계)를 나타낸다.
상위 한 객체가 하위 객체들을 소유한다.
빈 다이아몬드로 표기한다.(◇———)
구현 시 반드시 객체를 소유하는 개념은 아니다.

② 복합 연관 : 한 객체가 다른 객체들을 필수적으로 소유한다.
하위 객체들은 생성/소멸을 상위 객체와 함께 한다.
속이 칠해진 다이아몬드로 표기한다.(◆———)

2. UML의 구성

관계(Relationships)

3) 일반화(Generalization)

- 사물과 더 특수화된 사물 사이의 관계(is-a 관계)를 나타낸다.
- 하위 객체가 상위 객체를 대신할 수 있다.
- 속이 빈 실선 화살표로 나타낸다.()
- 주로 상속 관계를 나타낼 때 사용한다.

2. UML의 구성

관계(Relationships)

4) 실체화(Realization)

- 객체들 사이의 의미적 관계로 한 객체가 다른 객체의 계약을 지정한다.
- 의존(Dependencu)과 일반화(Generalization)의 혼합된 형태이다.
- 속이 빈 점선 화살표로 나타낸다.()
- 주로 인터페이스를 구현하는 클래스와의 관계를 나타낸다.

2. UML의 구성

다이어그램(Diagram)

- 다이어그램은 요소들을 그림으로 표현한 것으로 서로 다른 관점에서 시스템을 가시화하기 위하여 사용한다.

다이어그램의 종류

1) 정적 다이어그램
 ① 클래스 다이어그램
 ② 객체 다이어그램
 ③ 컴포넌트 다이어그램
 ④ 배치 다이어그램

2) 동적 다이어그램
 ① Use-Case 다이어그램
 ② 순차 다이어그램
 ③ 통신 다이어그램
 ④ 상태 다이어그램
 ⑤ 활동 다이어그램

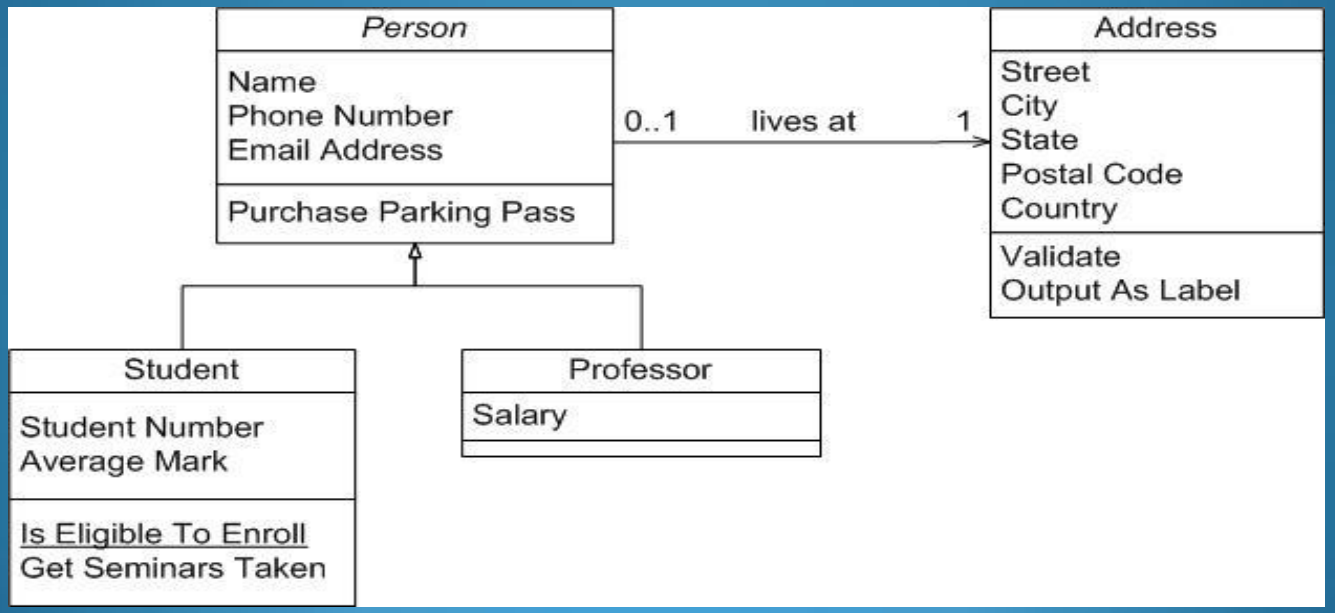
2. UML의 구성

다이어그램(Diagram)

1) 정적 다이어그램

① 클래스 다이어그램

- 클래스, 인터페이스, 통신 간의 관계를 나타내며 가장 공통적으로 쓰이는 다이어그램이다.



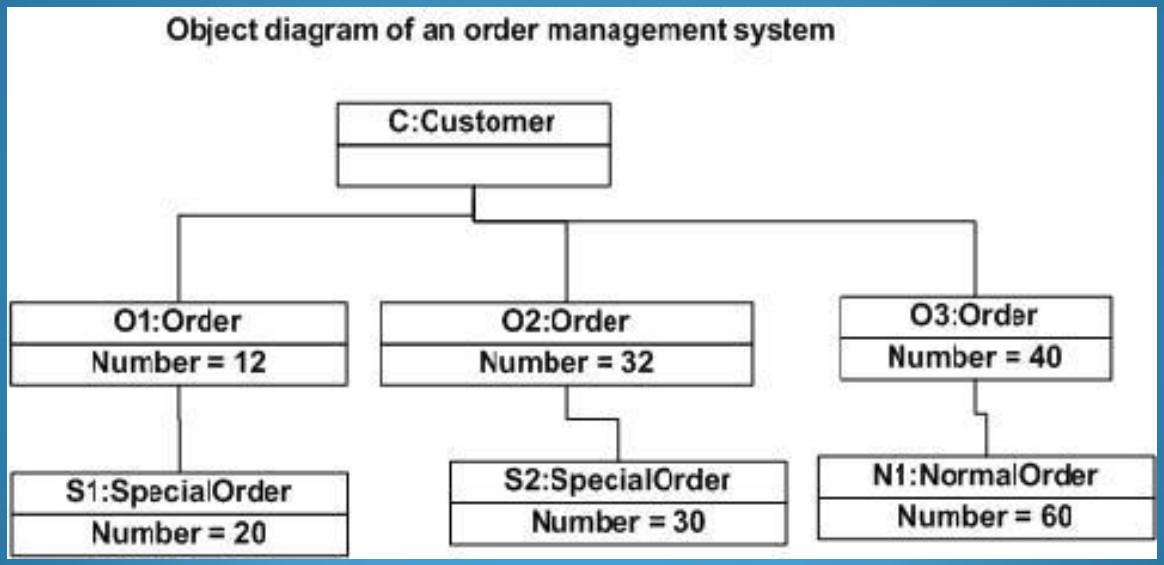
2. UML의 구성

다이어그램(Diagram)

1) 정적 다이어그램

② 객체 다이어그램

- 객체와 객체들 사이의 관계를 나타낸다.
- 특정 시점의 객체들의 구조적 상태를 표현한다.
- 클래스 다이어그램과 비슷하지만 실제 사례의 시각에서 구현한다.



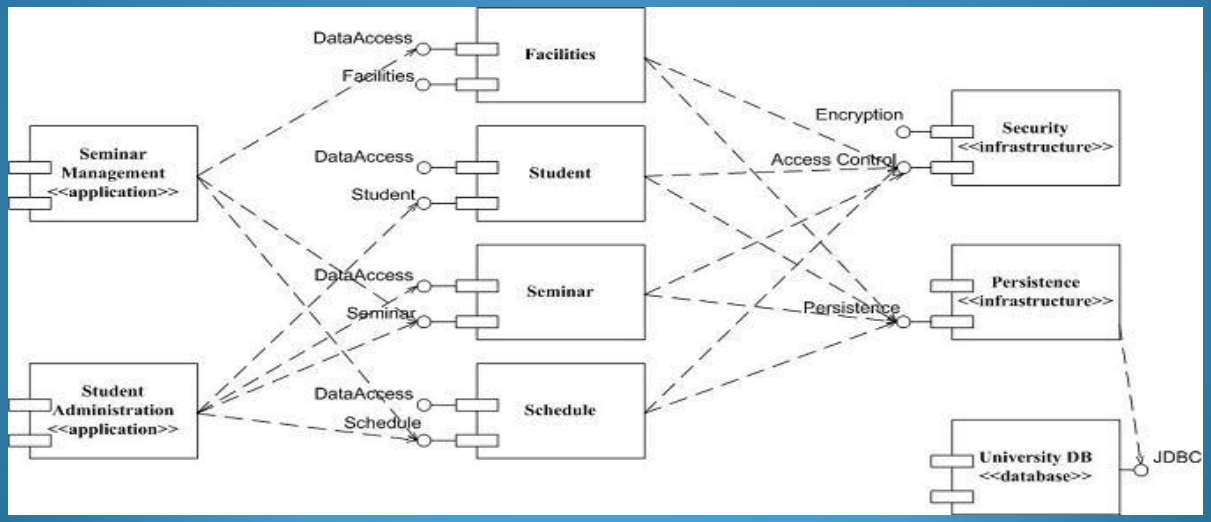
2. UML의 구성

다이어그램(Diagram)

1) 정적 다이어그램

③ 컴포넌트 다이어그램

- 컴포넌트 사이의 구성과 의존을 나타내며 시스템의 정적 구현 뷰를 다룬다.
- 하나의 컴포넌트는 클래스 다이어그램에 있는 하나 이상의 클래스, 인터페이스, 통신들과 대응한다.



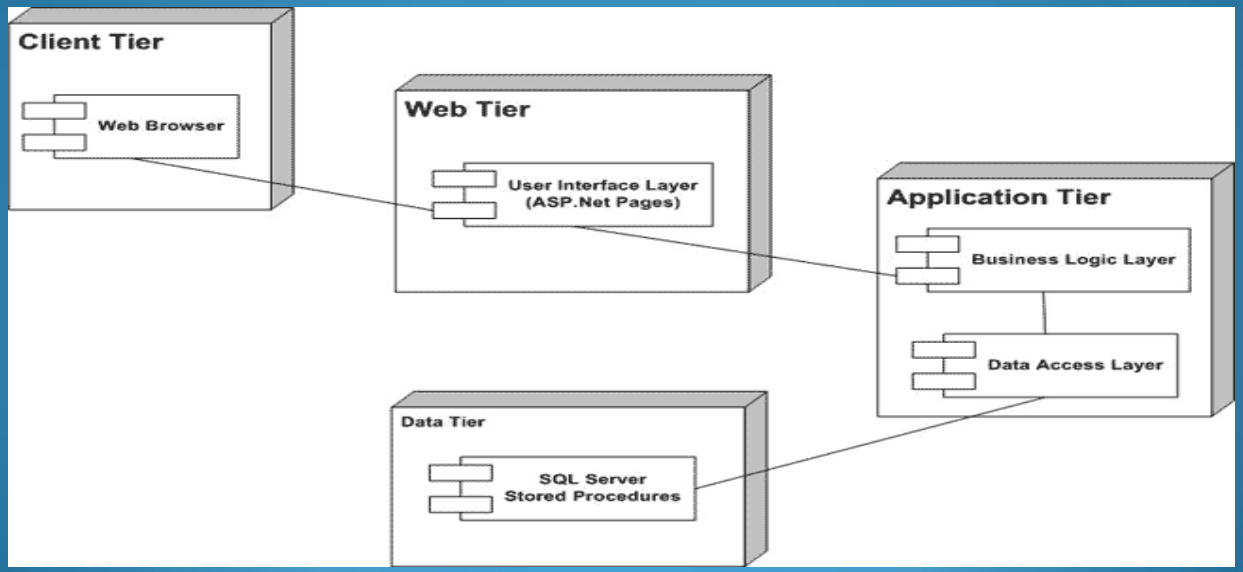
2. UML의 구성

다이어그램(Diagram)

1) 정적 다이어그램

④ 배치 다이어그램

- 실행 시 처리하는 노드와 그 노드의 컴포넌트들의 구성을 나타낸다.
- 하나의 노드가 컴포넌트를 수용하기 때문에 컴포넌트 다이어그램과 관련이 있다.



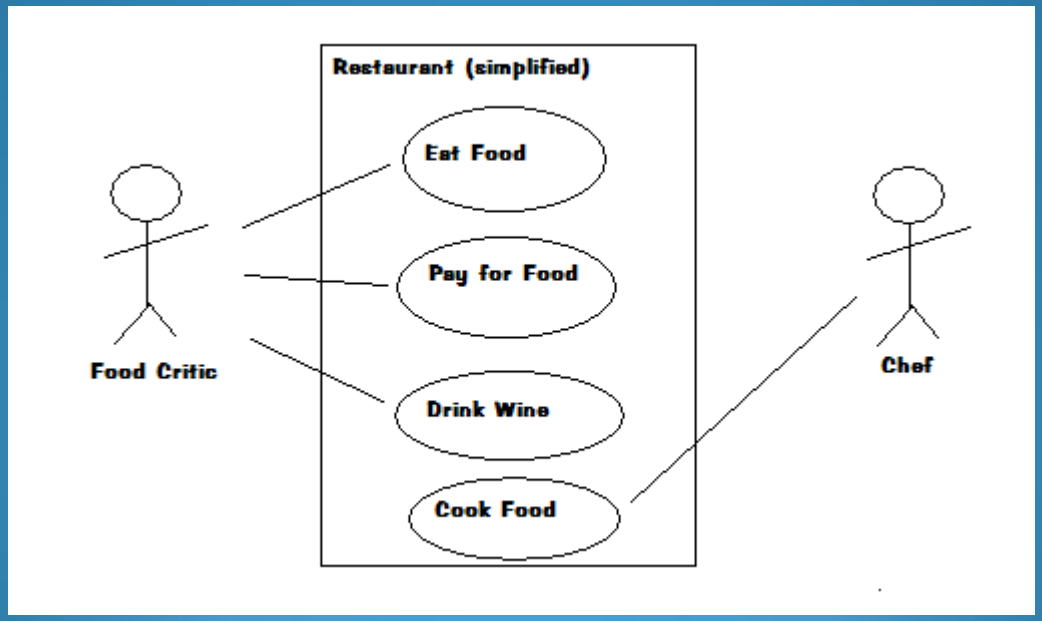
2. UML의 구성

다이어그램(Diagram)

2) 동적 다이어그램

① Use-Case 다이어그램

- Use-Case와 행위자 사이의 관계를 구조적으로 나타낸다.
- 시스템의 행동을 조직화하고 모델링 하는 데 중요하다.



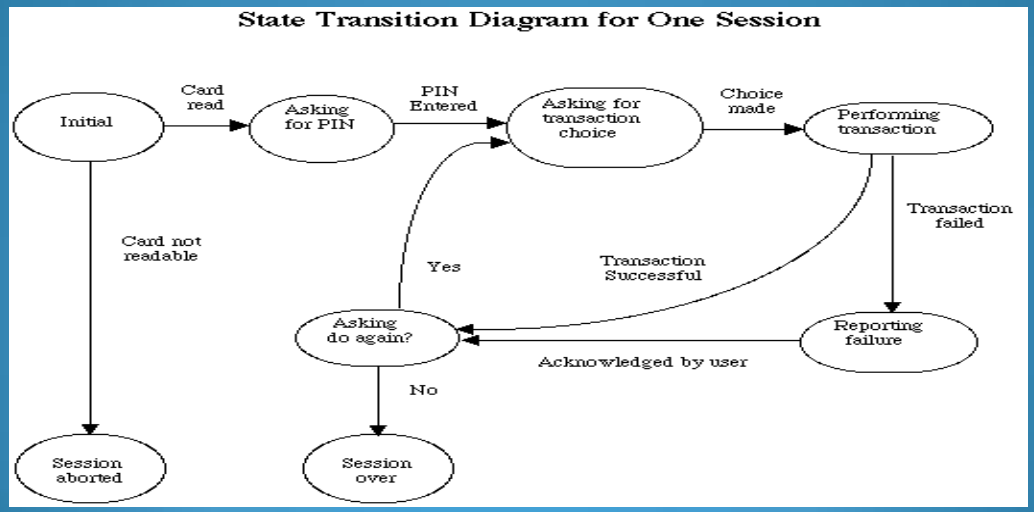
2. UML의 구성

다이어그램(Diagram)

2) 동적 다이어그램

② 상태 다이어그램

- 상태 머신을 나타내며, 시스템의 내부 전이를 표현한다.
- 상태, 전이, 사건, 활동으로 구성된다.
- 인터페이스, 클래스, 통신의 행동을 모델링 하는 데 중요하다.
- 빠른 반응이 필요한 시스템을 모델링 할 때 유용하다.



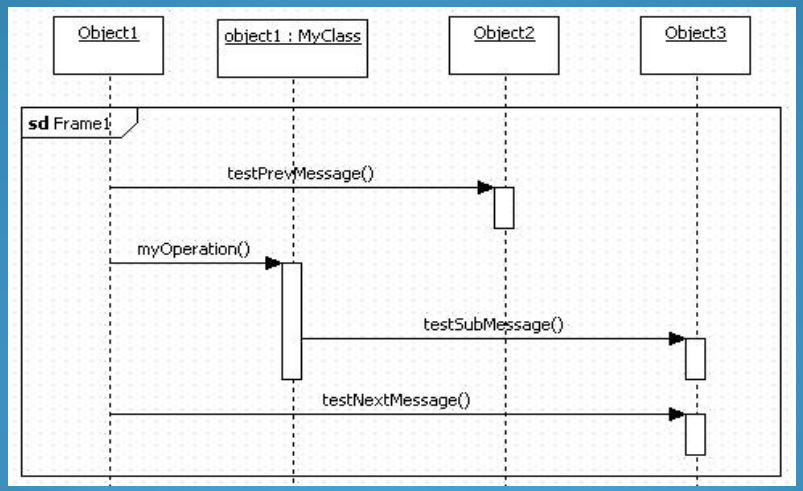
2. UML의 구성

다이어그램(Diagram)

2) 동적 다이어그램

③ 순차 다이어그램 / ④ 통신 다이어그램

- 이 둘은 교류도(객체와 관계,메시지로 구성되는 교류)의 한 종류이다.
- 순차 다이어그램은 시스템의 순서를 강조하고, 통신 다이어그램은 메시지를 주고 받는 객체의 구조적 구성을 강조한다.
- 이 둘은 같은 구조이기 때문에 상호 변환 가능하다.



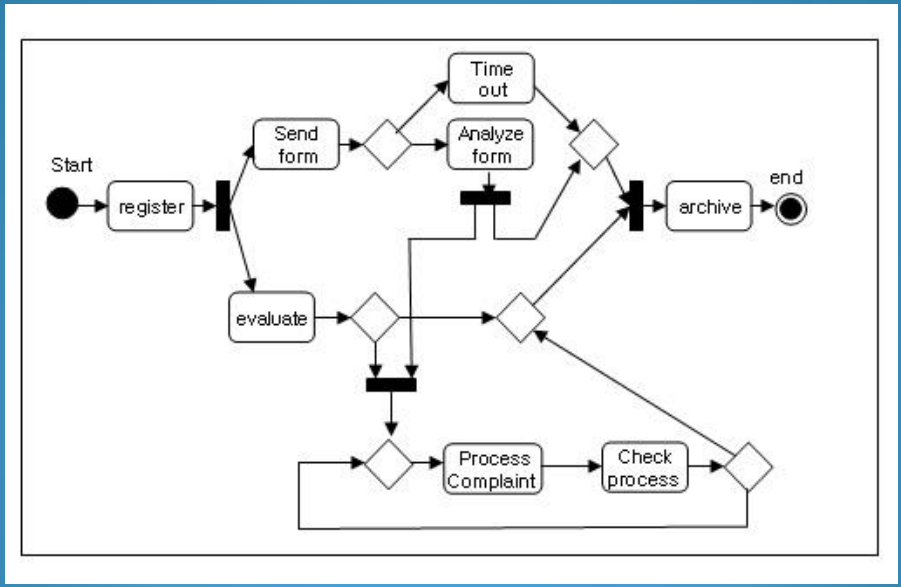
2. UML의 구성

다이어그램(Diagram)

2) 동적 다이어그램

⑤ 활동 다이어그램

- 시스템 내부에 있는 활동의 흐름을 기술한다.
- 시스템의 기능을 모델링 할 때 중요하고, 객체 간의 제어 흐름에 중점을 둔다.



3. UML 사용법

※ 본 자료는 StarUML을 기준으로 작성되었습니다.

1) Use - Case 다이어그램

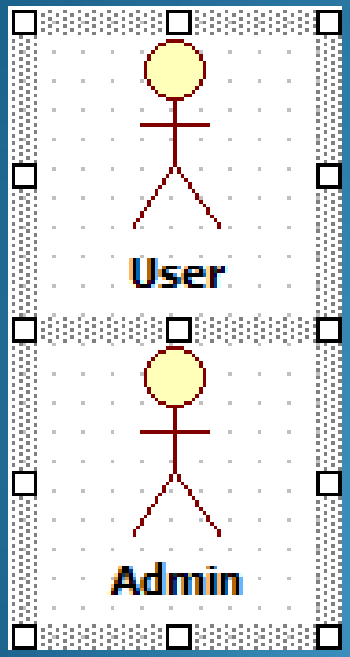
- 주요 구성 요소

Actor(시스템과 상호 작용하는 개체)
UseCase(시스템의 행위를 정의)
Association(UseCase 사이의 관계)
Include(한 UseCase가 다른 쪽을 포함)
Extend(한 UseCase가 다른 UseCase로 추가 확장)

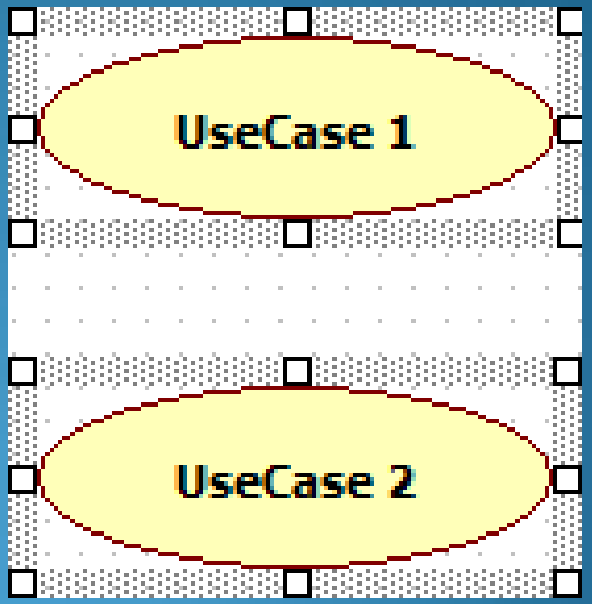
3. UML 사용법

1) Use - Case 다이어그램

① Actor 생성



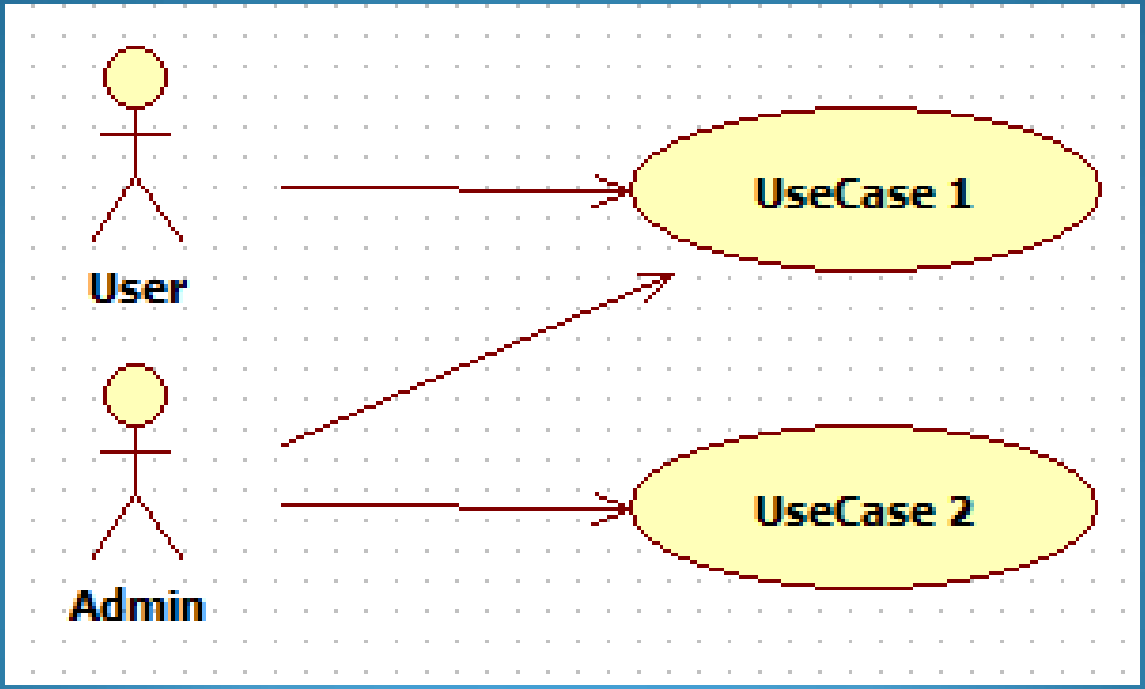
② UseCase 생성



3. UML 사용법

1) Use - Case 다이어그램

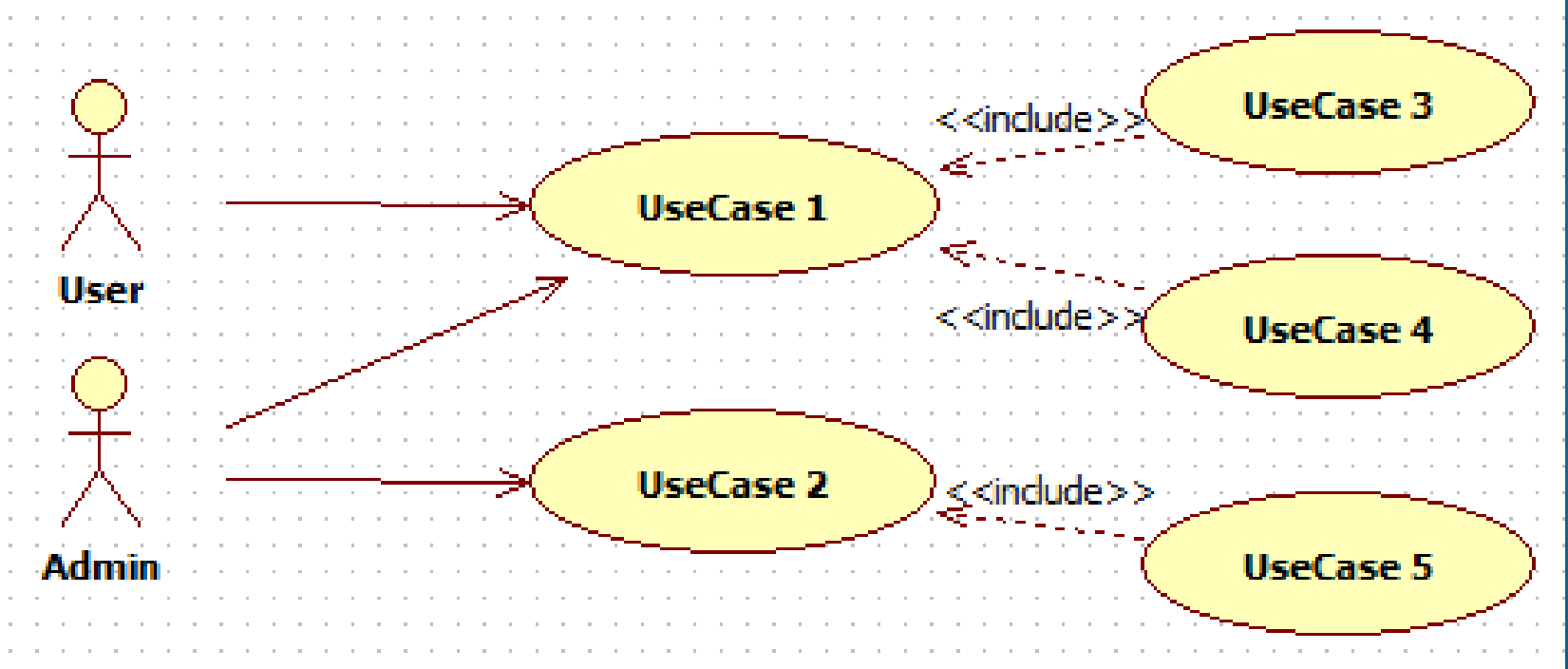
③ Actor와 UseCase 간의 Association 설정



3. UML 사용법

1) Use - Case 다이어그램

④ UseCase간의 Dependency를 스테레오 타입과 함께 설정



3. UML 사용법

2) Class 다이어그램

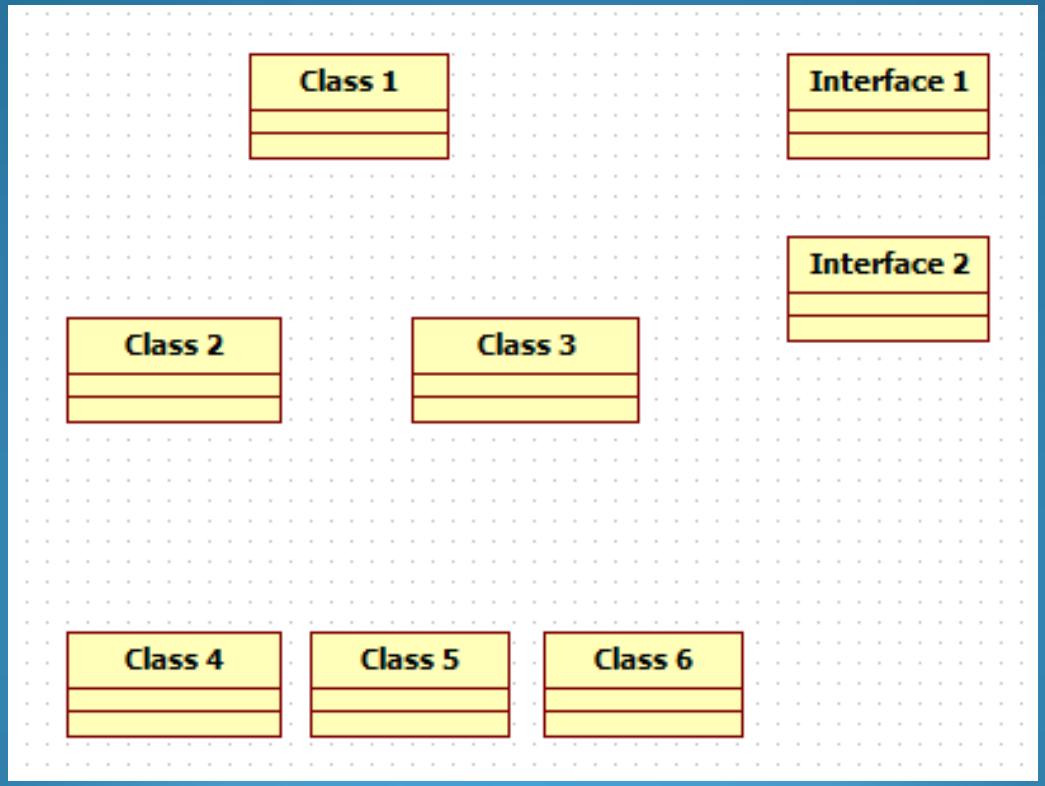
- 주요 구성 요소

Class (객체의 구조와 행위를 묘사)
Interface (Class에 의해 제공되는 서비스)
Association (Class 사이의 관계)
Generalization (2. UML의 구성 참조)
Realization (2. UML의 구성 참조)

3. UML 사용법

2) Class 다이어그램

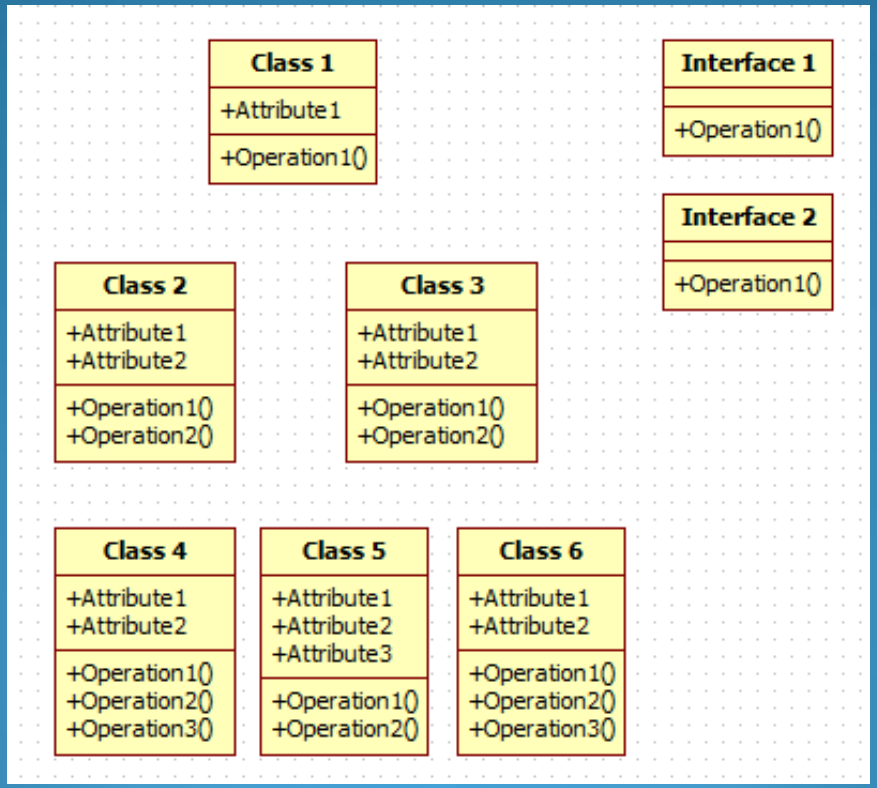
① Class와 Interface를 생성



3. UML 사용법

2) Class 다이어그램

② 각 Class들의 Attribute와 Operation들을 표기



3. UML 사용법

2) Class 다이어그램

③ Class들의 관계와 Interface와의 관계를 표기