Introduction to UML

Project Team

T4 Team

Latest update on:

2015-03-13

Team Information

박정민 컴퓨터공학부 201111354

박준한 컴퓨터공학부 201111356

정국빈 컴퓨터공학부 201111384

Table of Contents

1. UML 개요

UML은 Unified Modeling Language의 약자 이다. 그리고 개발자들이 Software System을 명시하고, 시각적으로 보이게 하며, 구조를 디자인하고, 내용을 기록하게 해주는 도구이다. 이 UML은 Software를 설계하는데 있어서, 확장 할 수 있게 해주며, 확실하며, 튼튼하게 할 수 있게 해준다. UML은 Object-oriented Software를 개발하는데 중요한 요소이며, 그래픽적으로 표현하여 Software의 시각적인 모델을 만들 수 있게 해준다.

또 UML은 시스템개발자가 자신의 비전을 구축하고 반영 하는데 있어서 표준적이고 이해하기 쉬운 방법으로 할 수 있도록 도와주며, 자신의 설계결과 물을 다른 사람과 효과적으로 주고받으며 공유할 수 있는 메커니즘을 제공한다고도 할 수 있다.

위와 같은 UML은 요즘같이, 복잡해지는 세상에 컴퓨터기반 시스템 역시 덩달아 복잡해지고 있다. 여러 대의 Hardware와 Software는 기본이고, Hardware를연결하기 위한 네트워크 시스템과 데이터베이스도 필요해졌다. 이렇게 복잡해지는데 있어서 결과물을 만들기 전의 청사진이라고 볼 수 있다. 더욱 거대하고, 튼튼하고 외관이 아름다운 건물을 만들기 위해서는 설계도부터 완벽해야 한다는 건 누구나 다 아는 사실인 것처럼, 더욱더 복합적이고, 군더더기 없으며, 결점이 없는 Software를 만들려면 이에 해당하는 청사진 또한 빼먹을 수 없는 부분이다.

2. UML의 구성 요소

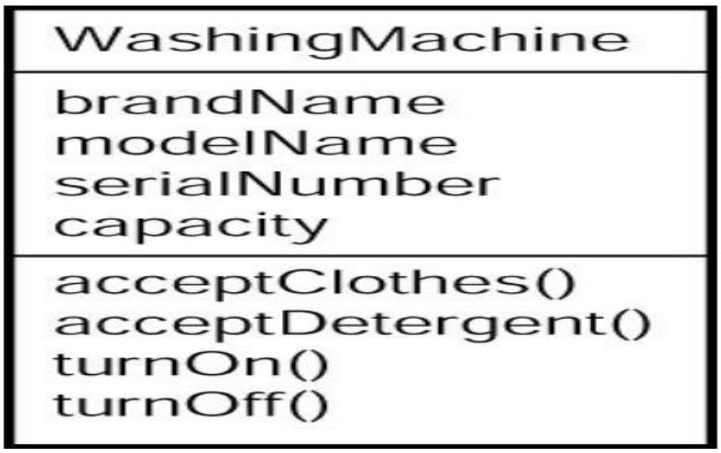
UML의 여러 가지 그래픽 요소는 하나의 큰 그림, 즉 다이어그램을 그리는데 사용된다. UML은 언어이기 때문에, 이들 그래픽 요소들을 서로 맞추는 데에는 규칙이 필요하다. 특히 기본 요소를 구성하는 건 '사물', '관계(Relation)', '다이어그램'으로 구성되어 있다. 먼저 다이어그램에 대해서 얘기해보자면 다이어그램이란?

다이어그램의 목적은 시스템을 여러 가지 시각에서 볼 수 있는 뷰를 제공하는 것이며, 이러한 뷰의 집합을 모델(Model)이라고 한다. UML 모델은 시스템자체의 목적 행동을 설명하는 언어이며 UML 모델은 시스템의 구현 방법을 설명하는 수단이 아니다. 일단 Diagram의 종류에 대해서 알아보자.

2.1. Diagram

2.1.1 Class Diagram

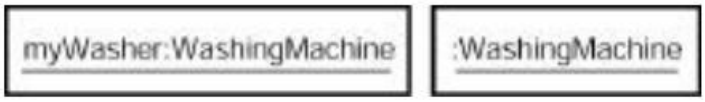
객체지향 기술은 여러분의 주변 상황과 가끔 흡사하다. 대부분의 사물은 자기만의 속성과 일정한 행동수단을 가지고 있다. 이러한 행동을 오퍼레이션의 집합으로 생각할 수 있을 것이다. 택시는 탈 것, 의자는 가구 범주에 넣을 수 있듯이, 사실 우리 주변에서 발견할 수 있는 대부 분의 것들은 어떠한 범주에 넣을 수 있다. 이러한 범주를 클래스라고 한다. 클래스란, 비슷한 속성과 공통적인 행동 수단을 지닌 것들의 범주 혹은 그룹을 일컫는다. 세탁기(washing machine)의 클래스가 있다고 가정하고 예를 들어보자. 이 클래스의 속성은 브랜드 이름, 모델, 일련 번호, 용량 등이고, 이 클래스의 행동은 옷을 넣는다, 세제를 뿌린다, 켜다, 끄다 등일 것이다. [그림 2.1.1]은 방금 이야기한 세탁기의 속성과 행동을 UML로그린 예로써, 사각형은 클래스를 나타내는 UML 아이콘이다. 이 사각형은 세 부분으로 쪼개져 있다. 가장 윗 영역에는 클래스 의 이름을 넣고, 둘째 영역은 속성을 넣는 공간 그리고 가장 마지막 영역은 오퍼레이션을 넣는 공간이다. 그림에서 클래스와 속성 그리고 행동의 이름을 각각 눈여겨보도록 하자. UML에서는 두 단어 이상으로 이루어진 클래스 이름은 단어 사이의 공백을 없애고, 각 단어의 처음 문자를 모두 대문자로 한다 속성과 행동의 이름 또한 마찬가지이지만, 가장 앞 단어의 처음 문자는 소문자로 한다



[그림 2.1.1]

2.1.2 Object Diagram

객체(Object)란, 클래스의 인스턴스 즉, 값이 매겨진 속성과 행동을 가지고 있는 개별적인 개체를 일컫는다. 세탁기를 예로 "들면 정보전자"라는 브랜드명과 발 빨래 세탁기라는 모델명, 그리고 "GL57774"라는 일련번호와 16파운드라는 용량을 가진 하나의 세탁기가 객체가 될 수 있다. [그림 2.1.2]는 객체를 UML로 나타낸 그림이다. 아이콘 자체는 클래스와 똑같이 사각형이지만, 이름에 밑줄이 그어져있다. 인스턴스의 이름은 콜론(:)의 왼편에 쓰며, 클래스의 이름은 콜론의 오른편에 쓴다. 인스턴스의 이름은 소문자로 시작하고, [그림 2.1.2]의 오른쪽아이콘처럼 익명의 객체도 가능하다. 객체가 속해 있는 클래스를 보여주는 것에만 중점을 둠으로써, 특정 이름을 지정해 주지 않은 것이다.

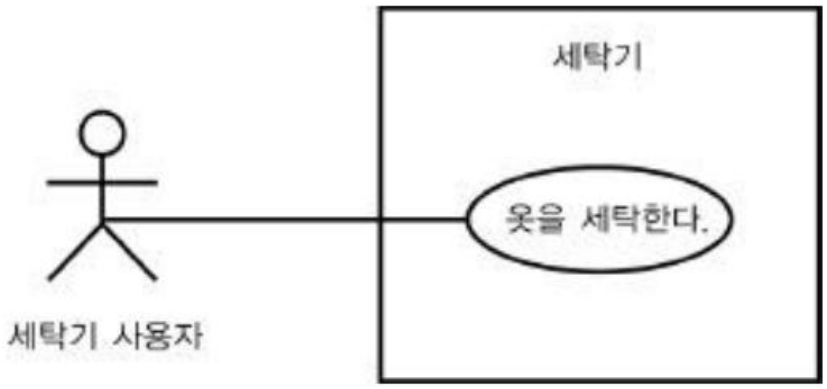


[그림 2.1.2]

2.1.3 Use case Diagram

유스 케이스(use case)는 사용자의 입장에서 본 시스템의 행동을 일컫는다. 시스템개발자에게 이 유스 케이스라는 것은 무척 값진 도구가 된다. 왜냐하면, 사용자가 원하는 시스템 사항을 얻어내는데 유용하게 쓰이기 때문이다. 목표가 사람들이 사용할 수 있는 시스템을 만드는 것이라면 더욱 중요하다.

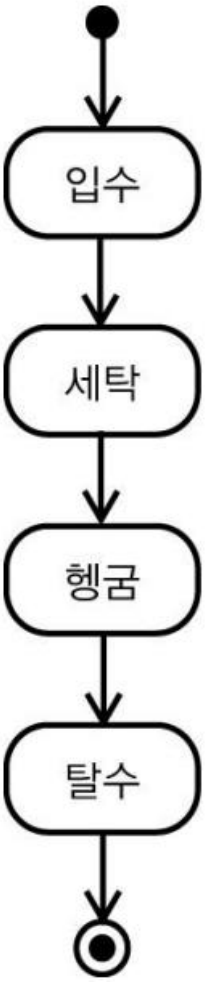
세탁기사용자를 나타내는 막대인간 그림을 행위자(actor)라고 한다. 타원은유스 케이스(use case)를 나타낸다. 행위자(유스 케이스와 대화를 시작하는 개체)-사람 혹은 다른 시스템이 될 수 있다. 또한, 유스 케이스가 시스템을 의미하는 사각형 내에 있고, 행위자는 사각형 바깥에 있음을 유의하여 보기 바란다.



[그림 2.1.3]

2.1.4 State Diagram

객체는 시간에 따라 각기 다른 상태에 있을 수 있다. 인간도 마찬가지이다. 갓난 아기로 태어나 영/유아가 되고, 어린이가 되고, 10대가 되고, 성인으로 자란다. 엘리베이터는 올라갔다가, 섰다가, 내려갈 수 있다. 세탁기는 물을 담고(soak), 세탁하고(wash), 헹구고(rinse), 돌리고(spin), 정지될 수 있다. UML 상태(state) 다이어그램은 [그림 2.1.4]에서보듯이, 현실 상황이 조금 많이 반영되어있다. 세탁기의 상태가 단계별로 변해감을 알 수 있다. [그림 2.1.4] UML 상태 다이어그램 이 그림의 상단에 있는 기호는 시작 상태(start state)를, 하단에 있는 기호는 종료 상태(end state)를 나타낸다.



[그림 2.1.4]

2.1.5 Sequence Diagram

클래스 다이어그램과 객체 다이어그램은 정적인 정보를 나타낸다. 하지만, 특정한 행동을 수 행하는 시스템에서는 여러 개의 객체들이 서로 메시지를 주고받으며 작업을 진행하는 것이 보통이다. UML 시퀀스(sequence) 다이어그램은 객체들끼리 주고받는 메시지의 순서를 시간 의 흐름에 따라 보여주는 그림이다. 역시 세탁기 예제를 가지고 계속 이야기하자. 세탁기는 타이머, 입수관(맑은 물을 넣기 위한 관) 그리고 드럼(빨래를 담는 통) 등으로 구성되어있다. 물론, 이것들은 모두 객체이다(곧이 해하겠지만, 객체는 다른 여러 개의 객체들로 구성될 수 있다). 이제, Wash clothes 유스 케이스를 작동시켰을 때 어떤 일이 일

어날까? 옷을 넣는다, 세제를 뿌린다, 그리고 켠다 오퍼레이션은 이미 마쳤다고 가정하면, 이 유스 케이스에서는 다 음의 행동이 단계별로 이루어질 것이다.

1. 입수 단계(Soaking)의 시작으로써 물이 입수관(water pipe)을 통해 드럼(drum)으로 들어간다.
2. 드럼은 5분 동안 정지된 상태를 유지한다.
3. 입수 단계(Soaking)의 끝으로써 물이 들어가다 멈춘다.
4. 세탁 단계(Washing)의 시작으로써 드럼이 앞 뒤로 15분간 회전한다.
5. 세탁 단계(Washing)의 끝으로써 세제와 때가 섞인 물이 배수관을 통해 나온다.
6. 드럼의 회전이 멈춘다.
7. 헹굼 단계(Rinsing)의 시작으로써 물이 다시 들어간다.
8. 드럼은 다시 앞뒤로 회전한다.
9. 15분 후에 물이 들어가다 멈춘다.
10. 헹굼 단계(Rinsing)의 끝으로써 물(rinse water)이 배수관을 통해 나온다. 11. 드럼의 회전이 멈춘다.
12. 탈수 단계(Spinning)의 시작으로 드럼이 시계 방향으로 회전하기 시작하면서 5분 동안 가속한다.
13. 탈수 단계(Spinning)의 끝으로써 드럼의 회전이 멈춘다.
14. 세탁이 종료된다.

타이머, 입수관, 드럼을 각각 객체라고 해보자. 각 객체는 하나 이상의 행동을 하면서 서로에게 메시지를 보내며 상호적으로 작동하게 된다. 메시지는 보내는 객체로부터 받는 객체로 전달되는 하나의 요청이라 볼 수 있다. 받는 객체 가 할 수 있는 행동 중에서 어느 하나를 수행해달라는 의미로 부탁되는 것이다. 그 행동들을 따져보자면, 우선 타이머는

▣ 입수의 시간을 잰다.

▣ 세탁의 시간을 잰다.

▣ 헹굼의 시간을 잰다.

▣ 탈수의 시간을 잰다.

를 할 수 있고,

입수관은

▣ 물을 흘려 보낸다.

▣ 물을 막는다.

를 행동할 수 있다.

또한, 드럼은

▣ 물을 저장한다.

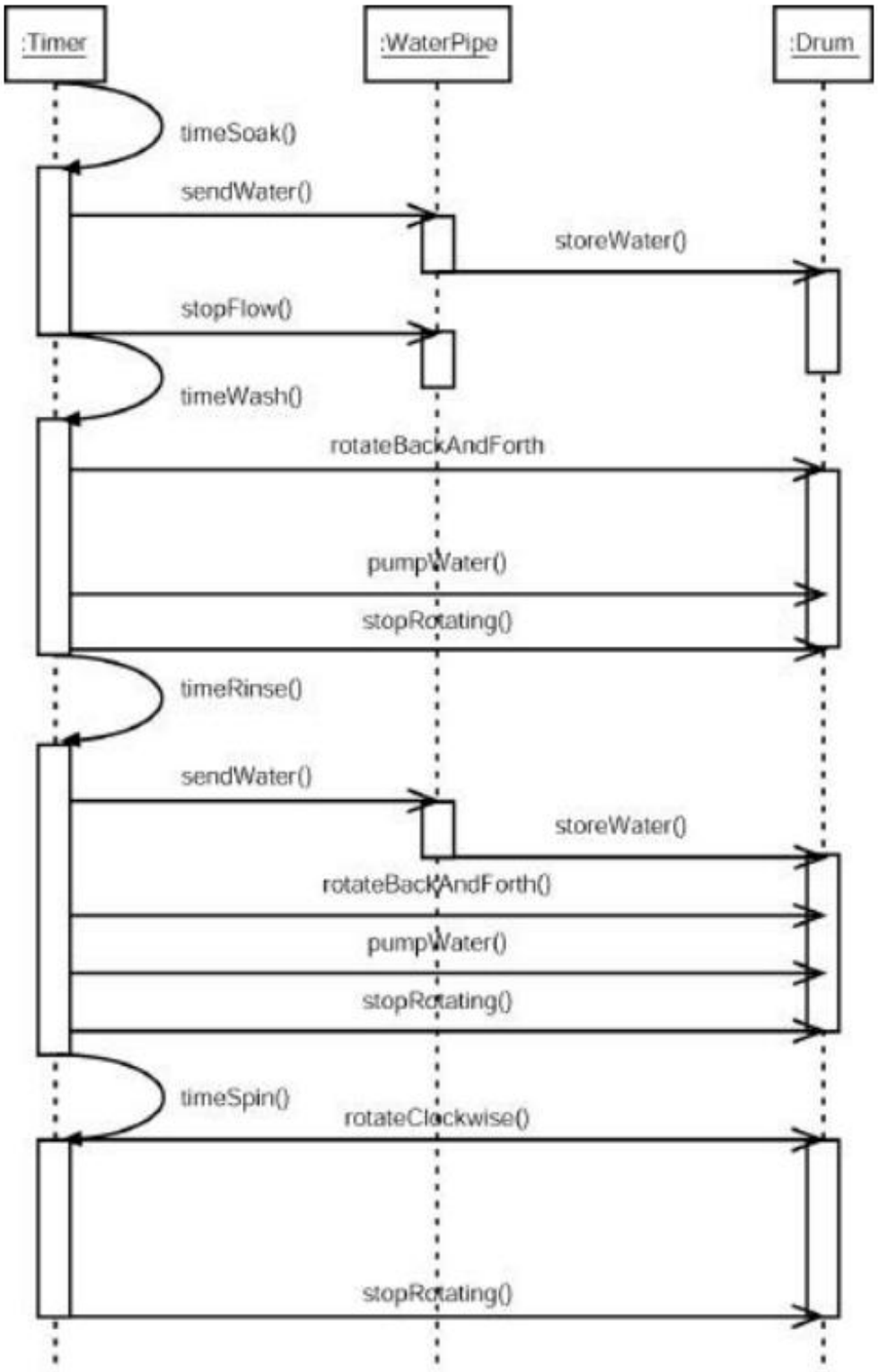
▣ 앞뒤로 회전한다.

▣ 시계방향으로 회전한다.

▣ 회전을 멈춘다.

▣ 물을 빼낸다.

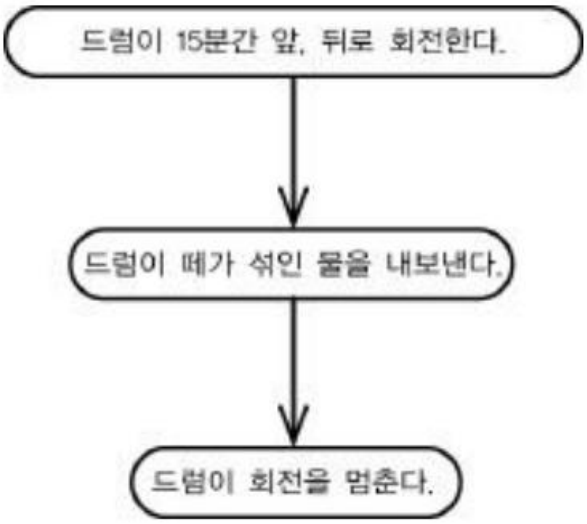
[그림 2.1.5] 는시간의흐름에따른입수관, 드럼, 배수관(그림상단에 익명의 객체로 되어있다) 사이의 상호 대화를 나타낸 시퀀스 다이어그램이다. 화살표는 하나의 객체에서 다른 객체로 전달되는 메시지를 의미한다. 이 다이어그램에서 시간은 위에서 아래로 흐른다. 첫 번째 메시지는 타이머가 자신에게 보내는 timeSoak() 이고, 두 번째는 타이머가 입수관(WaterPipe)으로 보내는 sendWater() 메시지이다. 마지막 메시지는 stopRotating( )으로써 타이 머에서 드럼으로 전달된다


:Timer
:WaterPipe
:Drum
timeSoak()
sendWater()
storeWater()
stopFlow()
timeWash()
rotateBackAndForth
pumpWater()
stopRotating()
timeRinse()
sendWater()
storeWater()
rotateBackAndForth()
pumpWater()
stopRotating()
timeSpin()
rotateClockwise()
stopRotating()

[그림 2.1.5]

2.1.6 Activity Diagram

유스 케이스 내부 혹은 객체의 동작 중에 발생 하는 활동(activity)은 대개 시퀀스 내에서 발견할 수 있다. 바로 앞에서 배운 전체 14단계가 활동에 해당되며, [그림 2.1.6]은 4단계부터 6단계까지를 활동 다이어그램으로 그려보았다



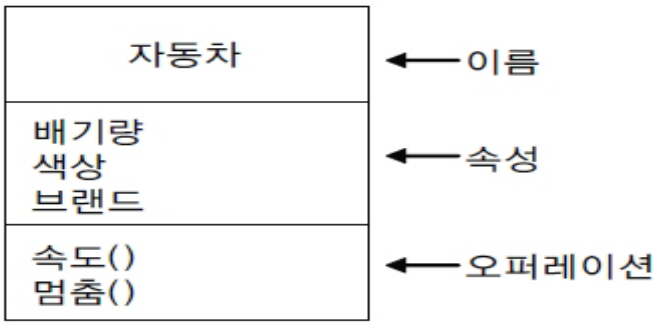
[그림 2.1.6]

2.2. 구조사물

UML모델의 정적인 부분이며, 개념적.물리적 요소를 표현한다.

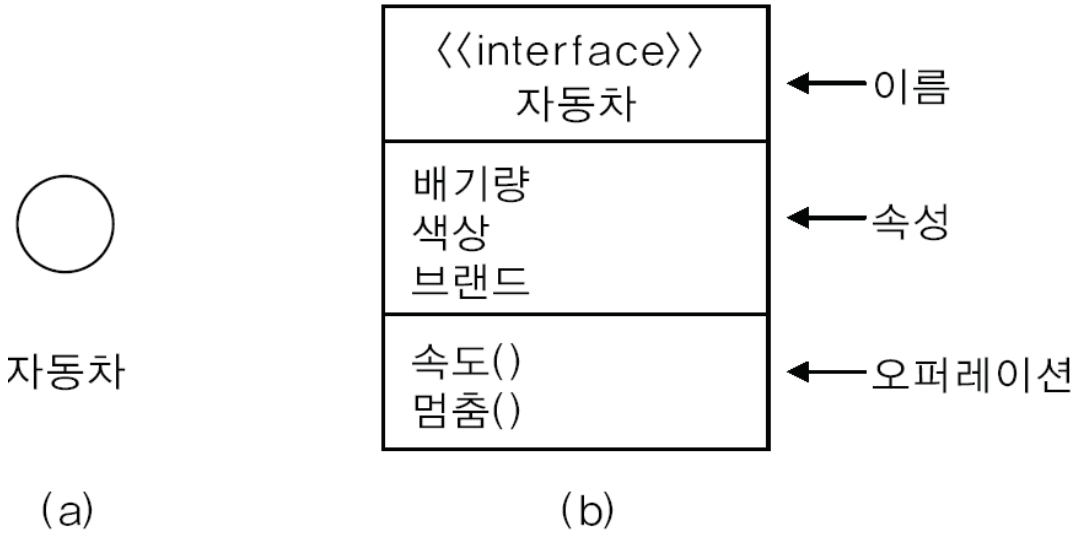
2.2.1 Class

동일한 속성, 오퍼레이션, 관계 그리고 의미를 공유하는 객체를 기술한 것으로 직사각형으로 표현한다. 사각형안에는 이름,속성,오퍼레이션 을 기재한다.



2.2.2 Interface

클래스 또는 컴포넌트의 서비스를 명세화하는 오퍼레이션을 모아놓은 것으로, 외부적으로 가시화되는 요소의 행동을 표현한다. 인터페이스는 특정 클래스나 컴포넌트 전체 또는 일부분의 행동을 나타낸다. 인터페이스는 원으로 표현하고 인터페이스명을 아래에 표시하거나 클래스 형식으로 표현하고 스테레오 타입으로 <<interface>>를 사용한다. 또 단독으로 나타나는 경우가 거의 없고, 인터페이스를 구현하는 클래스나 컴포넌트와 함께 나타난다.



2.2.3 통신(Communication)

교류를 정의하며, 서로 다른 요소와 역할들이 모여있는 것으로 행동적이고 구조적인 중

요성을 가지며, 하나의 클래스는 다수의 통신에 참여한다. 실선으로된 사각형으로 표현하고 보통 이름을 안에 넣는다.

2.2.4 Use Case

유스 케이스는 시스템이 수행하는 활동들을 순차적으로 기술한 것으로, 행위자에게 의미있는 결과를 제공한다. 유스 케이스는 모델에서 행동 사물을 구조화하기 위해 사용되고 통신으로 실현한다. 그리고 또 실선으로 된 타원으로 표현하고 보통 이름 안에 넣는다.

2.2.5 Active Class

활성의 2가지 형태가 있다.

1) 동작상태

객체에 있는 오퍼레이션을 호출하거나, 객체에 냥뮈을 전송하고 또 객체의 생성 소멸될 시에 순간적으로 작업을 실행한다.

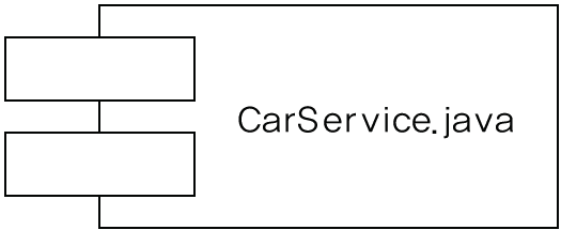
2) 활동상태

하나의 액티비티에 다른 제어 흐름을 가지는 활동 또는 동작상태로서 더작은 활동상태나 동작상태로 분해 가능하다.

2.2.6 Component

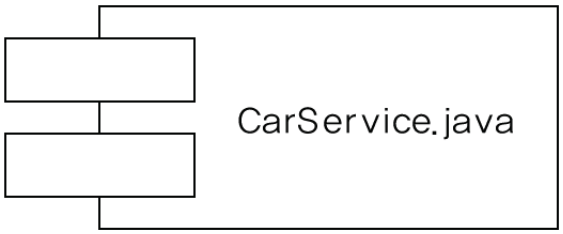
컴포턴트는 시스템의 물리적이고 대체 가능한 부분으로, 일반적으로 클래스,인터페이스, 그리고 통신과 같이 서로 다른 논리 요소를 물리적으로 패키지화한 것이다.

컴포넌트는 탭이 달린 직사각형으로 표시하며 이름을 안에 넣는다.



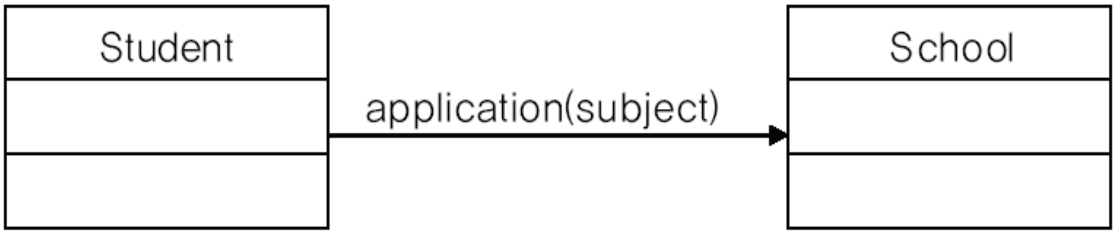
2.2.7 노드

노드는 실행할 때에 존재하는 물리적 요소로 컴포넌트가 노드에 존재할 수 있으며 노드에서 노드로 이동한다. 노드는 육면체로 표시하고 이름을 안에 넣는다.



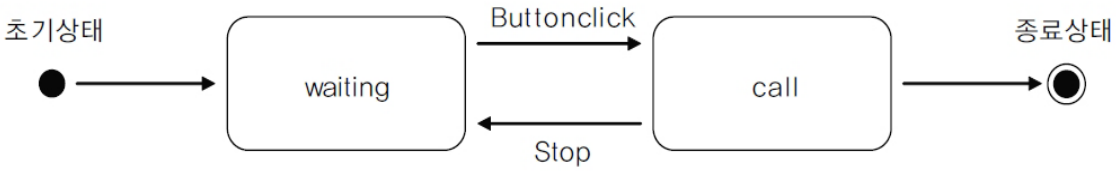
2.2.8 교류(Interaction)

교류는 행동이며, 목적을 위해 객체들간에 주고받은 메시지로 구성되어있다. 객체로 이루어진 공동체의 행동 또는 개별 오퍼레이션의 행동을 교류로 명세화 한다. 교류의 요소는 메시지, 활동순서, 링크 등이 있으며 메시지는 직선으로 나타내고, 항상 오퍼레이션을 포함 한다.



2.2.9 State Machine

State Machine은 상태의 순서를 지정하는 행동으로, 개별 클래스의 행동이나 여러 클래스들로 된 특정 행동을 하나의 상태로 지정한다. 또 서로 상태전이(상태에서 다른 상태로의 흐름), 사건(전이를 유발시키는 것), 활동(전이에 따른 응답)등으로 이루어져있다. 상태는 둥근 직사각형으로 표현하여 안에 이름을 넣고, 필요시 하위상태를 포함하는 형태로 구성된다.

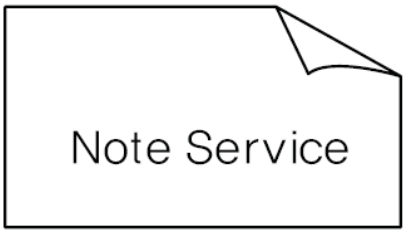


2.2.10 Package

패키지는 요소를 그룹을 묶은 것으로, 모든 사물은 패키지 내에 들어간다. 또 컴포넌트가 물리적인 것에 반해 패키지는 개념적(개발시에만 존재한다는 의미)로 탭이 달린 폴더로 표현하며, 보통 이름만 쓰는데, 때때로 그 안에 있는 내용을 표현하기도 한다.

2.2.11 Note

노트는 하나의 요소 또는 공동체에 첨부되는 제약과 주석을 함께 나타내기위해 사용되는 것으로, 접힌 직사각형으로 표현되며, 문자와 그래픽을 함께 나타낼 수 있다.



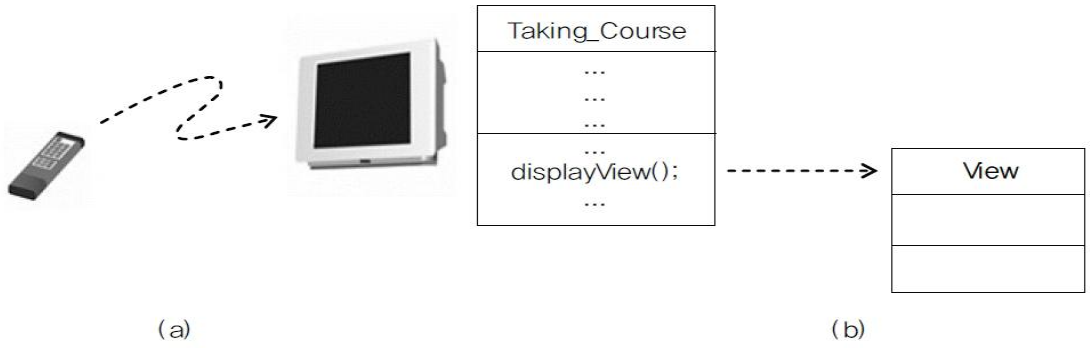
2.3 Relationship(관계)

UML에서 앞서 소개한 사물들간의 상관관계를 표시한 것으로서, 주로 '선'으로 표시한다 그 종류에는 대략 다음과 같은 관계들이 존재한다.

관계	UML 표기
Generalization (일반화)	
Realization (실체화)	
Dependency (의존)	
Association (연관)	
Directed Association (직접연관)	
Aggregation (집합, 집합연관)	
Composition (합성, 복합연관)	

2.3.1 의존

의존은 두 사물간의 의미적 관계로서, 한사물의 명세서(specification)이 바뀌면 그것을 사용하는 다른 사물에게 영향을 끼치는 것이나 그 반대의 반드시 성립하는 것은 아니다. 의존은 점선으로된 직선을 사용하며, 의존하고 있는 사물을 향하고 있는 형태로 되어있으며 주로 한클래스가 다른 클래스를 오퍼레이션의 매개변수로 사용하는 경우에 나타난다.



2.3.2 연관

연관은 구조적 관계로서 어느 한 사물 객체가 다른 사물 객체와의 연결을 의미한다. 두 클래스가 서로 연결되어 연관이 있다면, 한 쪽 객체에서 다른 객체로 옮겨갈 수 있으며 그 반대도 가능하다. 한 연관의 양쪽 끝이 같은 클래스를 향해 원을 그리며 순환하는 것 또한 가능하다.

(이는 한 클래스의 객체가 있을 때 다른 클래스의 다른 객체에게 연갈할 수 있다는 의미이다.)

그리고 연관에는

쌍방연관: 정확하게 두 클래스를 연결

다수연관: 2개 이상의 클래스를 연결하는 연관 두가지가 있다. 그리고 연관은 클래스들을 연결하는 실선으로 표현하며 기본형태에 연관의 이름, 역할, 다중성, 그리고 집합연관등을 추가해 표기한다. 다음은 연관의 특징들에 대해 서술했다.

1) 이름

연관은 이름을 가질 수 있다. 관계의 의미를 설명할 때 이름을 사용하며, 의미의 모호함을 없애기 위해 이름에 방향성을 추가할 수 있는데, 이름을 읽히기를 원하는 방향으로 방향 삼각형을 포기한다.

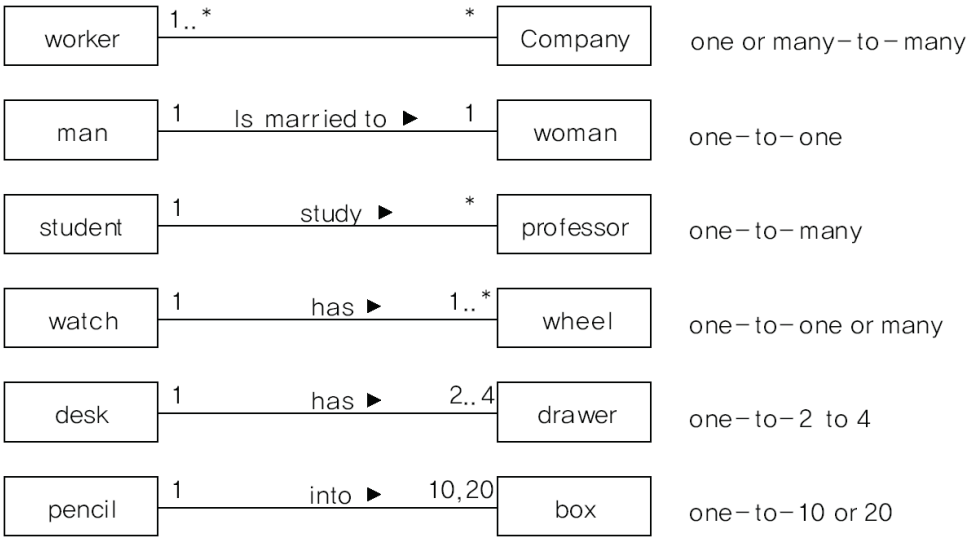
2) 역할

클래스가 연관에 참여하면 그것이 수행해야 하는 특별한 역할을 가지며, 수행하는 역할을 명시적으로 이름지을 수 있다. 또 클래스 옆에 원하는 역할을 써줌으로서 연관 관계 내에서의 역할을 표시한다

3) 다중성

연관은 객체간 구조적 관계를 표현하는데, 한 연관에 참여하는 하나의 객체에 대해 상대 쪽에는 몇 개의 객체가 연결되어 있는지를 밝히는 것이 중요한 때가 있다. '몇 개'를 일컫어 연관 역할에 대한 다중성이라 한다.

다중성은 하나(1), 제로 혹은 하나(0...1), 다수(0...n), 하나이상(1...n)등으로 표현한다.

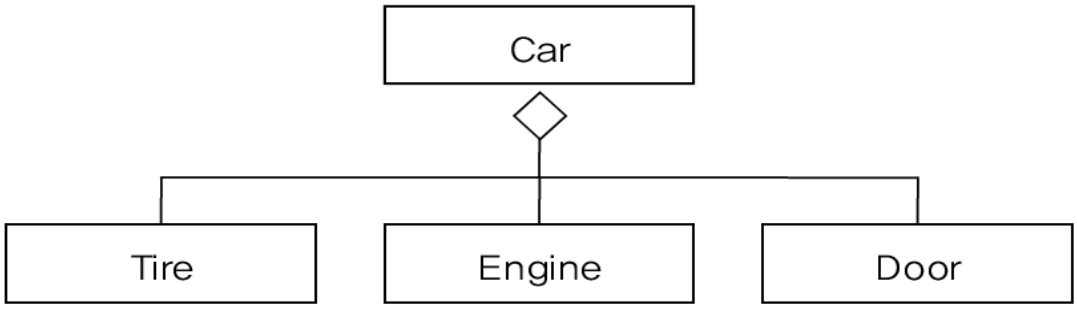


4) 집합연관

순수한 연관은 두 동료 클래스 사이의 구조적 관계를 표현하는데, 두 클래스는 개념적으로 같은 수준에 위치한다. 여기서 '전체/부분'관계에 대하여 모델링 하려고하면

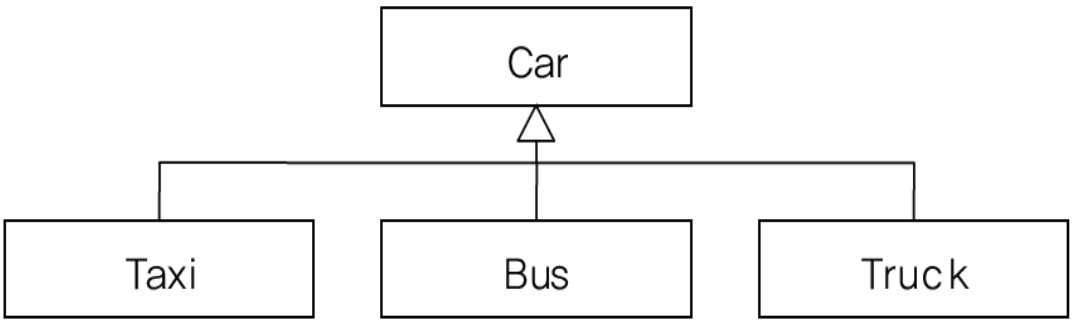
- 한 클래스는 더 큰 것('전체')을 대표하고 그것은 더 작은 것들('부분')으로 구성되어 있으며 이러한 관계를 집합연관 또는 'has-a'관계 라고표현한다.

5) 또 전체 쪽 한 객체가 부분쪽 객체들을 소유하며, 연관에 속이 빈 다이아몬드를 전체 쪽에 추가하여 표현한다.



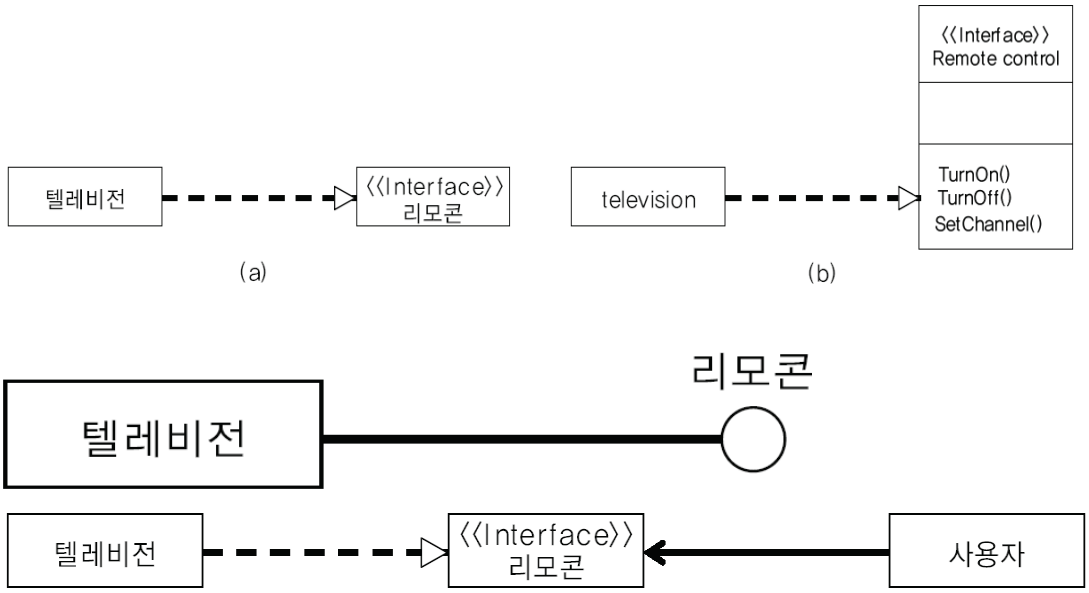
6) 일반화(Generalization)

일반화는 일반화된 사물과 좀 더 특수화된 사물 사이의 관계로서 'is-a' 라고 한다. 일반화는 자식 객체가 부모 객체가 사용되는 어느 곳에서나 사용될 수 있다는 것을 의미하며 그 반대는 성립하지 않는다.(자식 객체는 부모 객체를 대신할 수 있다는 것) 또 실선으로 속이 빈 큰 화살표를 부모를향해 그리며, 부모/자식 관계를 표현할 때 주로 사용한다. 주로 클래스와 인터페이스 사이에서 상속관계를 보여주기 위해사용하며, 다양한 사물들 사이에서도 일반화를 사용한다.



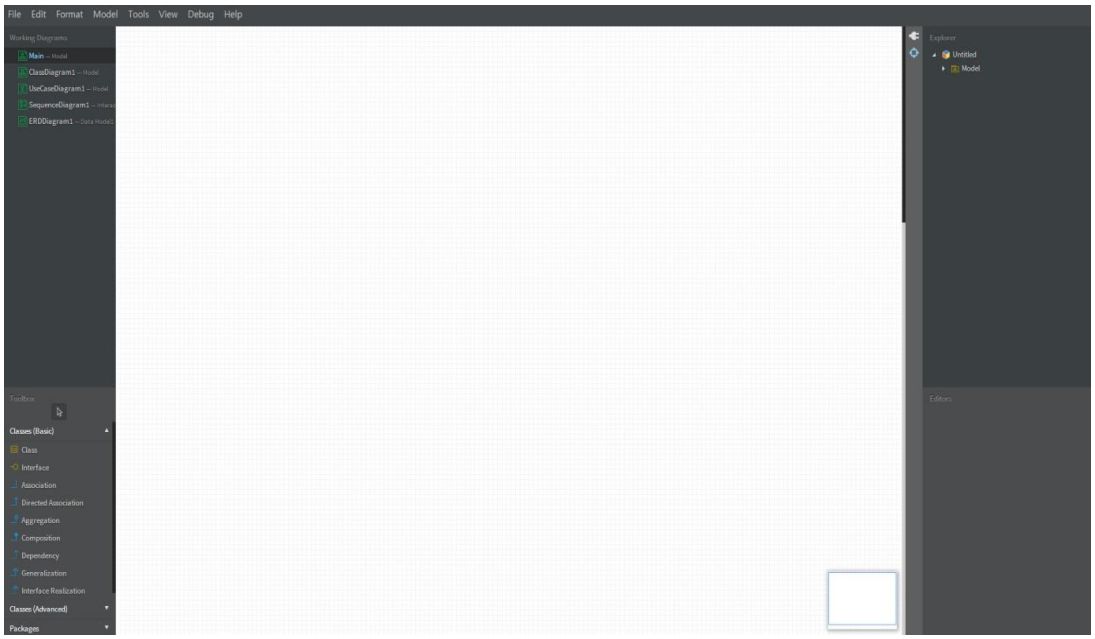
7) 실체화

실체화란 객체들 사이의 의미적 관계로서 한 객체가 다른 객체의 계약을 지정하며, 의존과 일반화의 혼압이기도 하고, 그 표기법도 의존과 일반화에서 가져왔다. 또 인터페이스와 인터페이스에 오퍼레이션 서비스를 제공하는 클래스나 컴포넌트 사이의 관계를 지정하기 위해 사용한다. 여기서 인터페이스란 오퍼레이션의 모음으로서 클래스나 컴포넌트의 서비스를 명세화하기 위해 사용하는 것으로, 속성을 갖지 않으며, 클래스와 비슷한 방법으로 모델링한다.



3. UML Tools 소개

3.1 StarUML



프로그램이 가볍고 처음 쓰는 사용자들도 사용하기 쉬워서 가장 대중적으로 쓰인다

3.1.1 지원하는 기능

UML 2.0 다이어그램

다양한 언어(Java, C++, C# 모듈)

커스터마이징 템플릿에 기반한 마이크로소프트 오피스 문서(워드, 엑셀, 파워포인트) 생성

커스터 마이징을 할 수 있는 코드생성

MDA 기술(UML 프로파일 및 커스터 마이징을 할 수 있는 다이어그램) 지원

다이어그램 확장성(UML이외의 영역에서 자신만의 다이어그램 정의)

버전 컨트롤 및 팀 모델링(분산된 파일 관리)

뛰어난 호환성(로즈, XMI 모듈)

3.1.2 특징

UML 2.0

UML은 OMG(Object Management Group)가 지속적으로 관리하는 통합 표준이다. 최근에 UML 2.0이 릴리즈 되었으며 StarUML은 UML 2.0 을 지원하며 최신 UML 표준을 지원하고 있다.

MDA (Model Driven Architecture)

MDA는 OMG가 도입한 새로운 기술이다. MDA의 장점을 얻기 위해서는 소프트웨어 모델링 툴은 많은 커스터마이징 요소들을 지원해야만 하는데 StarUML은 MDA를 지원할 수 있도록 설계되었고 UML 프로파일, 접근법, 모델 프레임워크, 표기법 확장, MDA 코드 및 문서 템플릿 등 수많은 커스터마이징 요소들을 제공한다. 이러한 것들은 사용자의 조직문화, 프로세스 및 프로젝트에 툴을 맞출 수 있도록 도움을 준다.

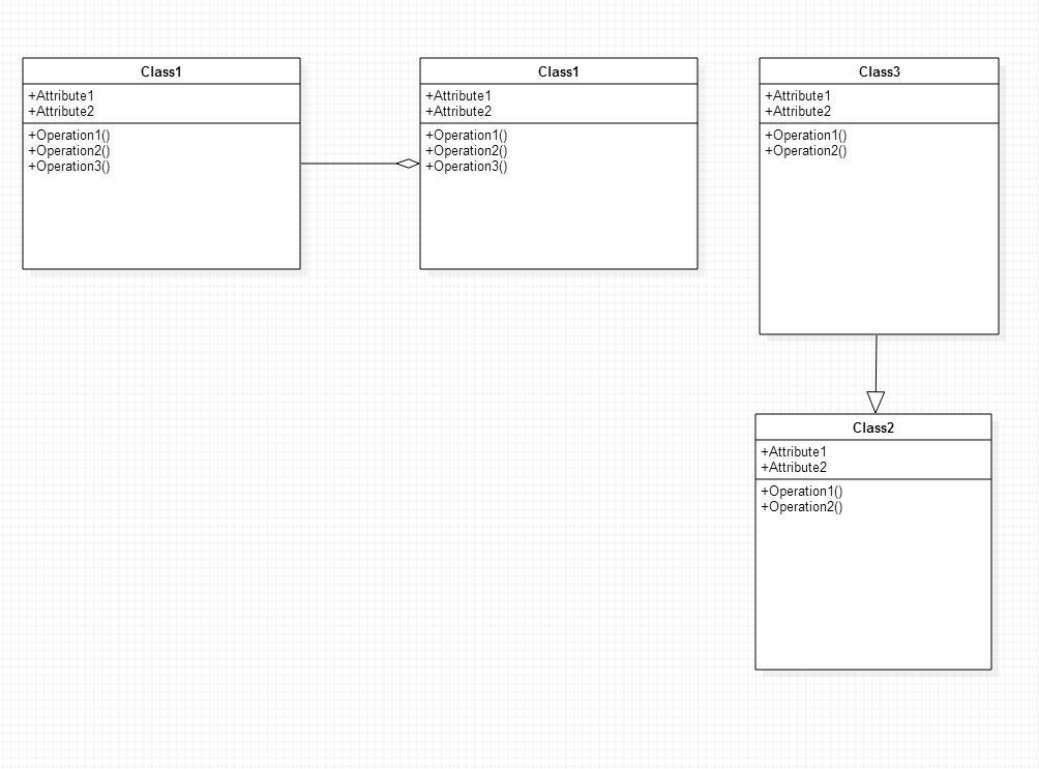
플러그인 아키텍처

많은 사용자들은 소프트웨어 모델링 툴에 보다 많은 기능을 요구한다. 이러한 사항에 부합하기 위해, 툴은 플랫폼에 매우 잘 정의된 플러그를 가져야만 하는데 StarUML 은 누구든지 COM과 호환 가능한 언어(C++, Delphi, C#, VB 등)에서 플러그인 모듈을 개발할 수 있게 단순하며 강력한 플러그인 아키텍쳐를 제공한다.

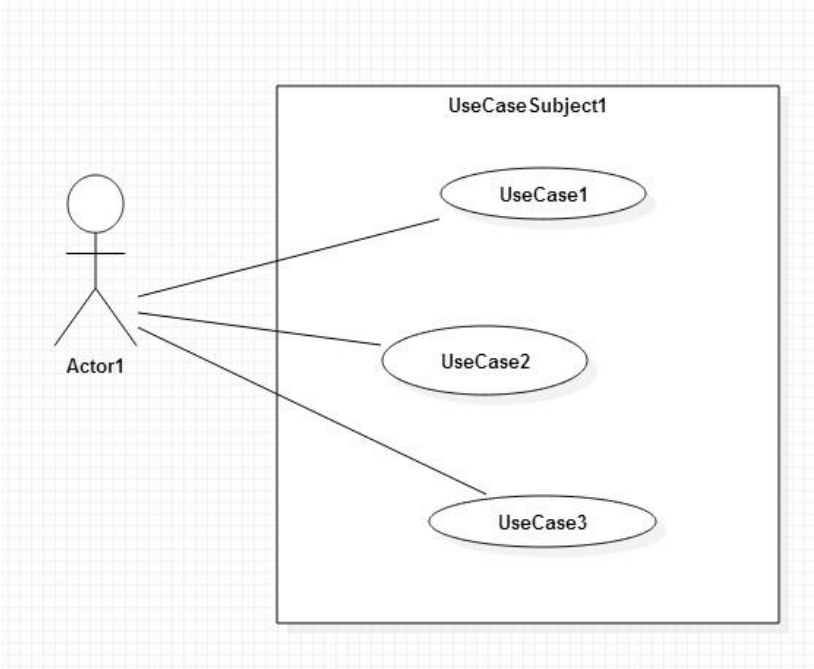
사용성

사용성은 소프트웨어 개발의 가장 중요한 사항입니다. StarUML은 퀵 다이얼로그, 키보드 조작, 다이어그램 오버뷰 등과 같이 많은 사용자들에게 친숙한 특징을 제공할 수 있도록 적용되었습니다.

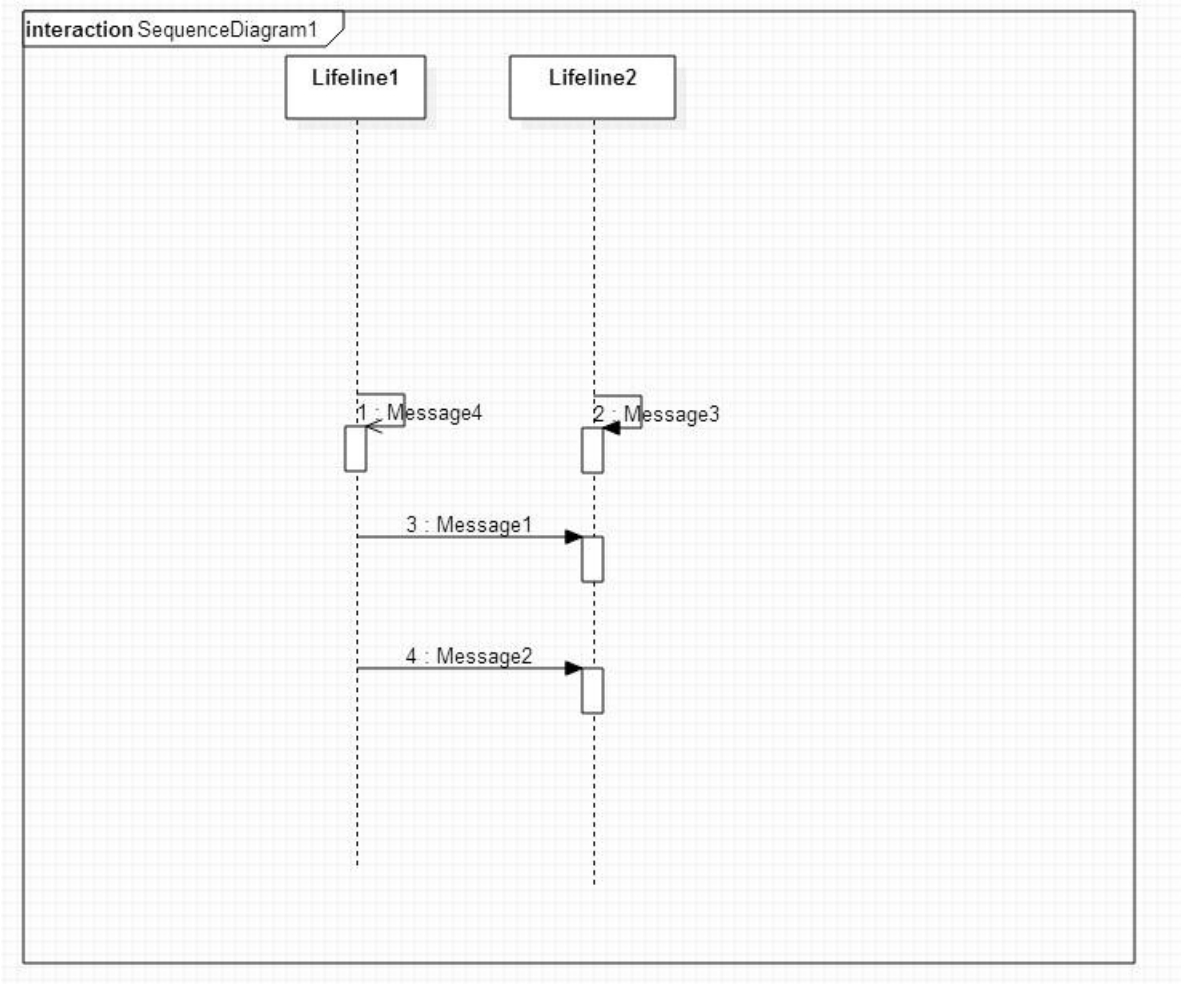
3.1.3 실 사용 화면



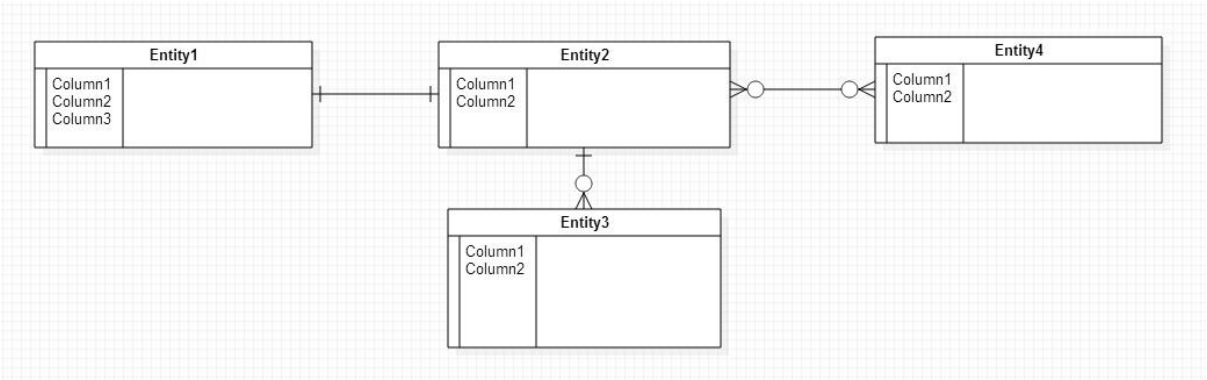
1 Class Digram



2 Use Case Digram



3 Sequence Digram

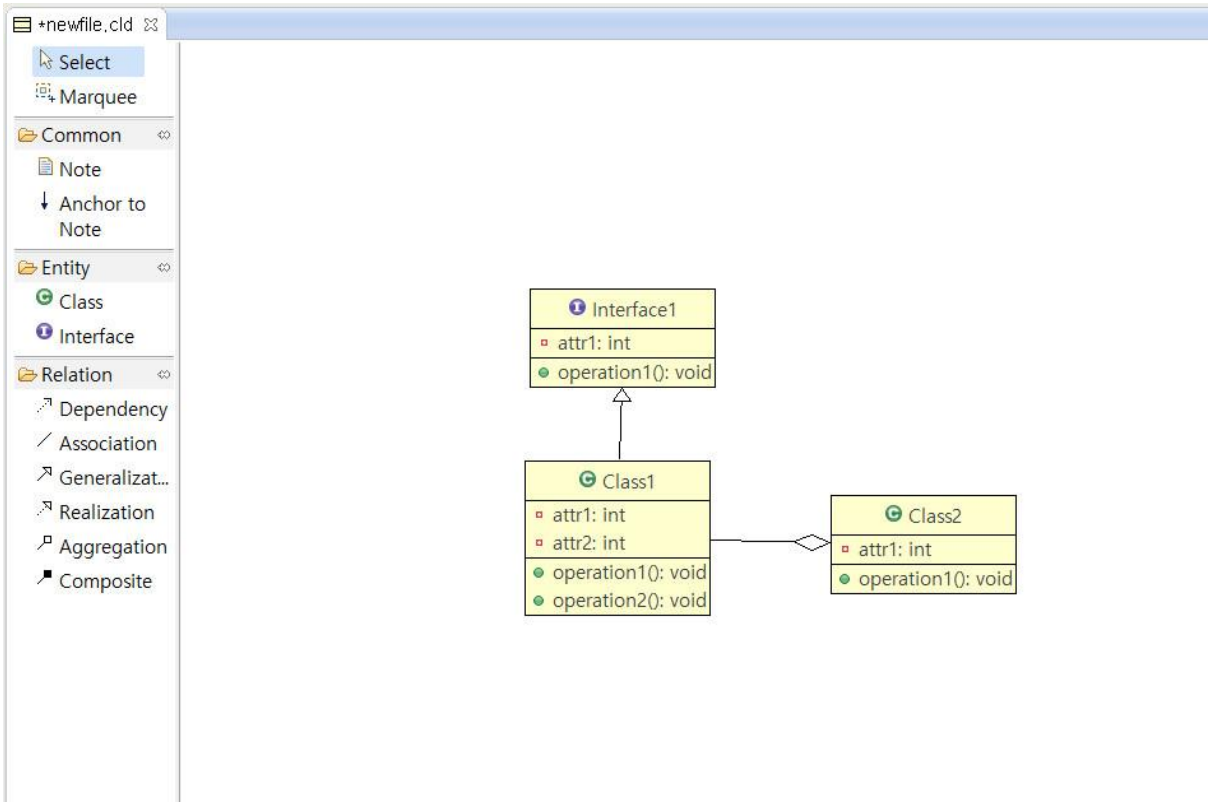


4 ER Digram

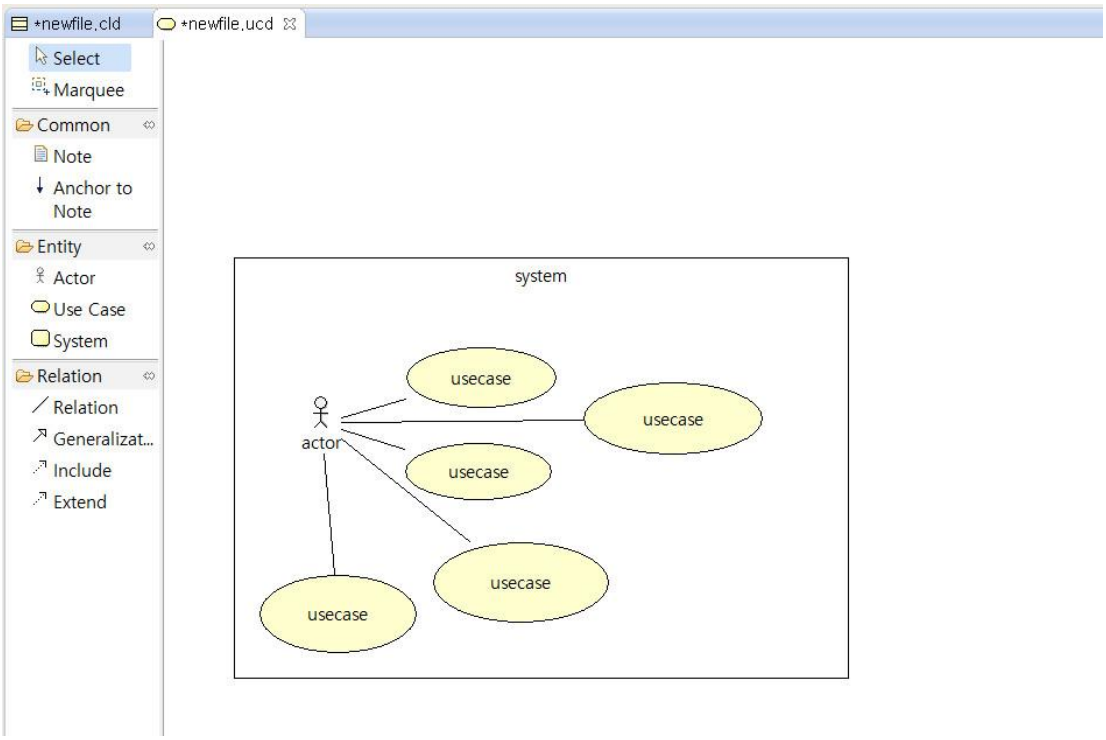
3.2 Amateras UML

이클립스에서 사용되는 플러그인으로 설치가 쉬우며 누구나 무료로 사용 가능하다.

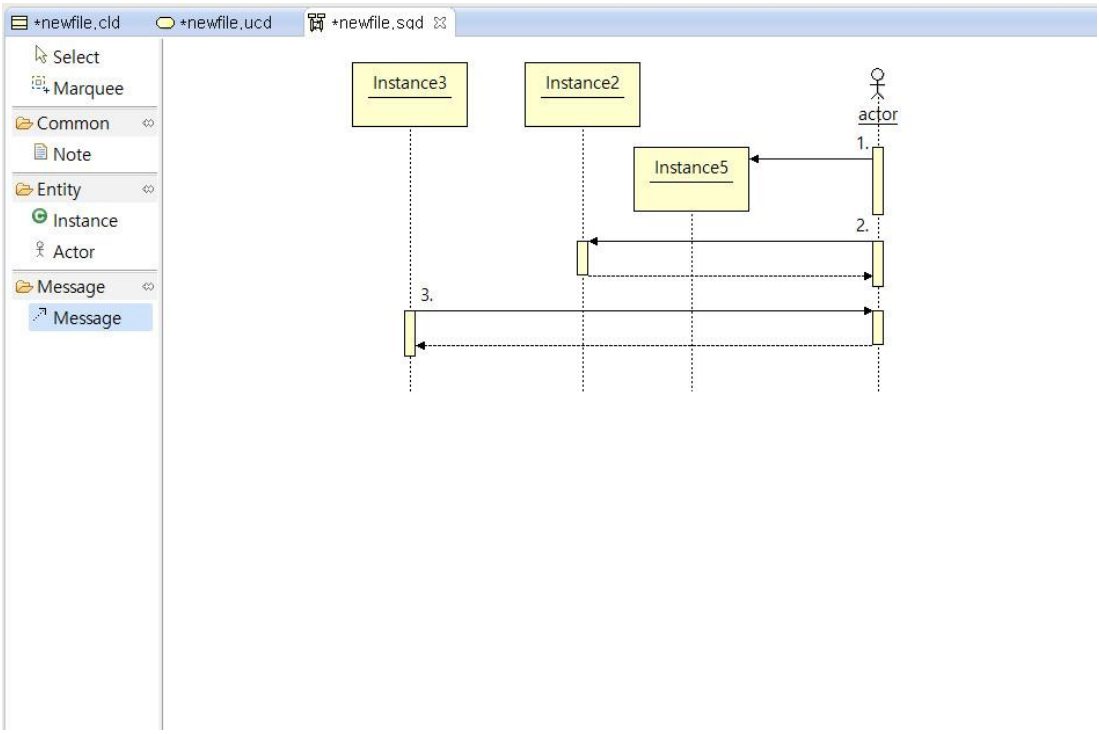
3.2.1 실 사용 화면



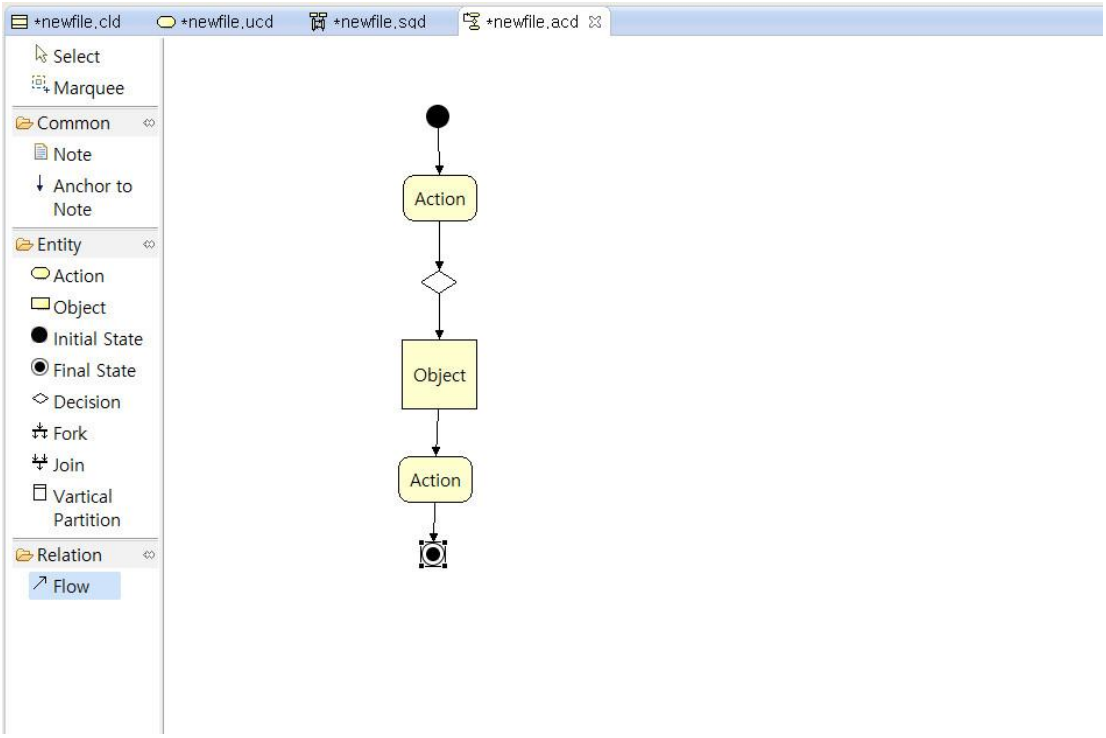
5 Class Digram



6 Use Case Digram



7 Sequence Digram



8 Activity Digram