Report #1

UML 도구를 이용한 OOAD 개발 조사

TEAM 4

200710116 박석희

200710117 양다빈

200711475 허원선

200711476 홍창연

목 차

❑ OOAD(Object-Oriented Alalysis and Design)

1. 개요

소프트웨어 방법론의 많은 연구와 발전으로 발생하게 된 방법론은 현재 많이 이용되고 있는 구조적 개발 방법론의 소프트웨어의 재사용, 모듈화, 개발된 소프트웨어를 유지보수하는 데 효과적이지 못해 적은 규모의 소프트웨어를 개발하는 데는 적합하지만 대규모 소프트웨어를 개발하는데 는 적합하지 못하다는 결점을 채워줄 수 있는 것이다.

객체 지향 개발 방법론은 소프트웨어의 정적인 데이터 측면과 동적인 기능적 측면을 모두 중시한 시스템을 개발할 때 요구되는 여러 가지 강력한 **모델링 개념과 캡슐화 개념** 등을 잘 지원하고 있다. 즉, 이런 객체 지향 방법론은 구조적 개발 방법론에 비해 소프트웨어 **분석, 테스트, 유지보수, 확장 등을 쉽게 해주며** 이해가 잘되는 설계 결과를 만들어 낸다. 즉, 클래스는 자연스럽게 소프트웨어 모듈의 한 단위가 되고 객체 지향 설계 과정은 그러한 객체 지향 사이의 복잡도를 적절히 관리해 준다.

2. 객체 지향 모델링

가. 객체 모델링 방법론

소프트웨어 개발의 큰 문제점은 복잡성, 요구의 변화, 개발 기간의 시간적인 제약성이다. 이러한 문제 해결에 제안된 기본적인 개념으로 분해 기법과 추상화 기법을 들 수 있다.

분해 기법은 복잡한 문제를 작고 단순한 서브프로그램들로 나눈다. 나누어진 문제는 전체 문제보다 해결하기 쉽고, 서브프로그램들을 다시 결합하면 원래 의도한 문제를 해결할 수 있다.

추상화 기법은 다양하고 복잡한 개념을 단순 개념으로 사상하는 과정으로 서로 다른 많은 사항들을 무시한 채 같은 것으로 취급함으로써 복잡한 방법을 줄이는 것이다. 객체 지향 방법론은 두 가지 개념을 바탕으로 소프트웨어를 개발한다.

객체 지향 방법론은 시스템 정적구조 파악 및 추상화, 분류화, 일반화, 집단화 등을 하는 **정적 모델링 방법**과 처음부터 동적 모델을 고려한 객체 추출을 기반으로 하는 **동적 모델링 방법**으로 나눈다.

(1) 의미 정의에 대한 객체 모델링

정보처리 시스템은 문제 영역 P에 대한 해 영역 S를 양식으로 나타낸 것으로 P⊂S인 관계로 나타낸다. 실세계에서 문제의 의미를 분석하여 기술하고, 양식에 의한 설계를 하여 해를 기술한다. 또한 P는 의미 객체를 포함한 집합으로써 P의 객체들은 해보다는 문제를 기술하는 개념으로 사용한다.

시스템의 의미 객체는 고객, 주문, 항목, 피고용자들로서 주어진 시스템의 실체들이 된다. S는 문제 영역 P 이외에도 인터페이스, 응용, 기본 객체들을 포함한다.

인터페이스 객체는 사용자와 정보처리 시스템 간의 인터페이스를 설명하고 문제 영역 자체를 설명하지 않는다. 즉 의미 객체에 대한 사용자의 관점에서 기술한다.

응용 객체는 정보처리 시스템에 관한 제어 메커니즘이다. 유도기능을 가지고 개발된 시스템의 기능들을 순차적으로 제어한다. 즉 시스템의 주된 모듈로서 다른 객체들에 대한 메시지와 메뉴를 전달한다.

기본 객체는 문제 영역이나 응용되는 독립된 객체이다.

의미 객체 모델을 바탕으로 객체 지향 분석을 하는 과정은 의미 자료 모델을 갖추어야 할 자료의 추상화, 의미 객체의 식별, 집합관계, Classification과 Instantiation, 속성의 승계, 그리고 서브 클래스와 슈퍼 클래스를 정의하여 수행한다.

분석 단계는 문제점을 기술함으로 의미 객체별 요구와 양식 중심으로 기술하고 설계 단계는 그 해를 기술한다. 객체 지향 분석은 문제 영역을 모델링하기 위하여 의미 객체의 집합을 식별하여 그 양식을 정하고 시스템 요구에 의해서 관련성을 기술한다. 객체 지향 설계는 해 영역을 모델링하는데 의미 클래스, 응용 및 기본 클래스를 설계한다.

결과적으로 실체를 식별하여 의미 객체로 추상화하고 객체들의 속성과 관련성, 집합 관계를 기술하여 의미 자료 모델링을 하고, 여기에 객체 지향 프로그램 언어가 가진 특성, 즉 속성의 외부 서비스, 객체 간의 인터페이스 메시지, Classification 및 행위의 승계, 그리고 정보 은닉과 캡슐화를 추가하여 객체 지향 분석 및 설계를 수행한다.

(2) 객체 모델링

객체 모델링은 문제 분석과정부터 객체를 추상화시켜서 클래스로 정의하고, 관련성을 분석하여 상속성을 정의하는 분석 과정이다.

① 객체

객체는 사람, 장소, 물건 등과 같이 물리적이거나 온도, 액수, 구좌 등과 같이 개념적일 수 있다. 또한 실체 시스템의 구성요소와 일대일 대응관계를 갖는다. 그러나 객체는 개개의 요소를 의미하는 것이지 항목들의 일반적인 범주나 그룹을 의미하는 것은 아니다. 객체는 하나의 고유 식별자를 갖는 것을 의미한다.

행위는 객체가 제공하게 될 서비스와 수행하게 될 책임을 의미하는 것으로 외부에서 객체를 사용하는 방법과 시스템에서 주어진 일을 수행하는 방법이 된다. 임의의 조건 하에서 어떠한 사건이 발생할 때 취하는 객체의 행동 파악이 필요하다. 행위는 또 다른 사건을 유발할 수도 있고 객체를 생성 또는 파괴하기도 하며 메시지의 교환일 수도 있다. 또한 객체는 상태를 갖고 이 상태는 시간에 따라 변화를 일으킨다.

상태란 객체의 현재 상황이나 전체 흐름에서의 단계를 의미한다. 객체에 대한 상태를 정의하는 방법으로는 여러 가지가 있을 수 있으나 가장 가능한 방법으로는 현재 가동 중인 시스템인 경우 시스템 운영의 흐름 내에서 객체 관찰을 통해서 그리고 수행되고 있는 시스템이 없는 경우 전체의 흐름을 가정하여 파악하는 것이다.

② 객체 클래스

서로 비슷한 객체들끼리 하나의 추상화된 것으로 만들 필요가 있다. 또한 분류작업으로 간주할 수 있다. 각 객체 클래스는 이름을 갖게 되면 이들이 포함하게 될 객체의 일반적인 성질을 갖게 된다.

③ 관련성

객체는 서로 다른 객체와 어떠한 형태로든 관련성을 갖고 있다. 즉 객체들 간의 논리적관계를 가지고 있다.

일반화(Is-A 관련성 집합) 관계는 한 객체가 다른 객체의 상위 집합인지 하위 집합인지를 정의한다. 이때 상위 집합을 일반화라고 하고 하위 집합을 상세화라고 한다. 이는 집합의 개념으로 판단하는 것으로 집합에 속하는 원소들을 고려하여 하위 집합에 속하는 객체가 상위 집합에 속할 수 있어야 한다.

상속 관계에서 하위 집합의 원소는 상위 집합의 원소라고 정의한다. 이는 일반화가 지닌 모든 관련성 집합에 새로 정의된 상세화가 포함됨을 의미한다. 상세화는 일반화가 지닌 모든 관련성을 상속 받는다. 어떤 경우는 하나의 객체 클래스가 두 가지 이상의 일반화의 상세화일 수 있다. 이때 하위집합은 여러 개의 상위집합이 지닌 모든 관련성을 상속 받는다. 이를 다중 상속이라고 정의한다.

집성화 관계는 한 객체가 또 다른 객체를 자신의 서브 파트나 부품으로 구성한 경우 이들 상호간에 집성화 관계가 존재한다고 한다. 이 집성화 관계에는 참가 제한 사항이 추가된다.

Is-Member 관계에서는 하나의 객체로 취급하려는 객체와 그들 원소들간의 관계를 말한다. 즉 이진 관계이다.

나. 객체 모델링의 단계

객체 지향 분석은 시스템의 시나리오에 나타난 객체를 식별하고 그들 간의 상호 작용을 파악한 다음, 각 객체에 부과된 책임(행위)을 분석한다. 객체의 행위 분석은 인터뷰하기 전부터 시작할 수 있으므로 시스템의 시나리오를 문서로 작성한 이유도 객체 추출을 쉽게 하기 위한 것이다. 여기에서 중점을 두어야 할 내용은 시스템을 개발하게 된 동기가 무엇이며, 동기별로 어떤 사건이 일어날 것인가 하는 점이다. 즉 기술 문서와 함께 객체와 그들의 속성 및 행위를 식별하는 것이 객체 지향 분석의 중요한 과제이다

(1) 분석 단계

분석 단계는 기본 단계로서 분석을 위한 계획서 작성, 행위에 초점을 맞춘 행위 분석, 객체를 추출하고 그 행위의 기술, 객체들 간의 관련성 파악, 동적 모델링으로 구분할 수 있다. 기본 단계는 분석의 계획 단계이다. 시스템의 시나리오를 바탕으로 살펴보면 개념적 모델링, 외부 양식, 구현의 과정을 정하고 시스템의 시나리오에 나타난 목표를 설정하고, 시스템의 범위와 문제 공간의 기술, 분석을 위한 자료의 선정, 그리고 분석 단계의 계획을 세운다. 시스템의 범위와 문제 공간의 기술은 시스템의 환경과 조건을 파악하고 문제 분석의 공간 영역이 어디까지인가를 결정한다. 분석을 위한 자원 식별은 사용자의 파악과 영역을 설명할 수 있는 전문적인 지식 그리고 적당한 참고문서 등을 찾아낸다.

이상과 같이 분석과정의 상위 단계에서 나타난 행위를 설명하고 다음에 연결되는 1 단계부터 4 단계까지의 분석을 위한 계획을 세우고, 모델의 개념을 정립한다. 기본 단계에서 기술되는 문서는 분석계획, 현 시스템의 목표, 요구 정의, 품질 목표, 면담내용의 기술로서 시스템의 시나리오와 비교하여 모순점이 없는가를 평가할 수 있는 문서와 객체 모델링의 전 과정에 관한 계획문서이다. 요구정의와 품질목표, 면담 등은 인터뷰와 참조문서를 바탕으로 해서 이루어지는데 객체들의 활동을 설명하는 시나리오, 에러조건과 처리방법, 모델에 포함된 서비스, 서비스의 수행 책임자인 행위자의 식별의 내용을 포함해야 한다. 시스템의 목표와 계획이 세워지면 개념적인 모델링 단계에 들어가는데 이 과정으로

문제분석, 객체의 추출, 그리고 객체의 관련성 정의에 들어간다.

문제 분석 단계에서는 시스템의 기능을 정하고 그 기능을 누가 수행하며, 누구의 도움을 받아서 집행할 것인가를 결정한다. 시스템 기술에서 나타난 동기와 사건을 중심으로 야기되는 문제를 파악하고 그 해답 영역을 찾는다. 그리고 나서 해당 영역별로 어떤 활동이 일어날 것인가를 예측하고, 활동할 객체를 선정한다. 선정된 객체는 어떤 역할을 담당할 수 있는가를 분석하여 그 객체의 행위와 속성을 정의한다. 끝으로 행위자의 활동을 중심으로 하여 객체들의 행위를 결정하여 기술 문서를 작성한다. 기술 문서의 내용은 객체와 속성, 그리고 요구자의 행위와 행위자의 서비스를 기술한다. 활동 기술서는 요구자와 행위자의 객체가 수행하는 활동을 의미한다. 객체가 식별되면 요구자의 행위는 행동이라고 하고, 행위자의 행위는 서비스라고 정한다.

(2) 활동 기술 단계

객체 추출과 활동의 기술 단계는 문제 분석에서 추출된 모든 객체의 역할을 정하고, 객체의 속성과 행위를 비교 분석하여 미비점과 모순점을 찾아낸다. 그 다음에 객체들을 구분하여 요구자인지, 행위자인지를 식별하고 각각의 행위를 기술한다. 끝으로 트레이스의 방법을 기술하면 객체의 기초적인 모델링은 끝난다. 객체의 속성은 논리적인 특성이기 때문에 객체 활동에 관련된 성질을 정의하면 된다. 객체의 행위는 객체의 책임이 무엇인가를 파악하고, 그 객체가 제공할 서비스와 다른 객체에 요청할 서비스를 구분하여 정해 나간다.

모든 참여자가 객체로 나타나는데 참여자는 동기를 부여하고 서비스를 요청하는 요구자와 서비스를 수행할 행위자로 나눌 수 있다. 그러나 한 개의 객체가 요청과 수행의 두 가지 기능을 같이 가질 수 있다. 이럴 때는 객체 기능은 두 가지로 구분해서 설명할 수 있는 활동을 정의해야 된다. 객체는 요구자와 행위자가 되는데 요구자의 활동은 능동적인 행동을 의미하고, 행위자의 활동은 수동적인 서비스를 의미한다. 본 단계의 활동 기술서에서는 활동 기술서를 구별하여 설명하는 과정이 중요하다.

(3) 관련성 기술 단계

객체의 관련성 기술 단계에서는 객체의 관련성을 추상화에 의한 일반화, 특성화에 의한 집합관계, 그리고 링크와 결합에 의한 연결 관계를 찾아서 객체들을 그룹으로 나누는 단계이다. 추상화는 객체에 어떤 기능을 부여할 것인지를 기술함으로써 외부적 관점에서 설명하고자 하는 것으로 구현에 의해서도 그 기능이 변하지 않은 캡슐화를 시도할 수 있다. 또 공통된 특성과 행위를 특성화시켜서 다른 객체에 승계시켜 줄 수 있도록 일반화시키고 추상화된 객체를 새로운 객체(새로운 특성과 행위를 추가시켜서)로 재정의하는 특성화를 통해 보다 차원 높은 추상화를 유도해 낼 수 있다. 즉 추상화된 객체를 일반화와 특성화를 통해서 재차 추상화시킴으로써 승계를 위한 차원 높은 추상화 객체에 대해서 집합관계를 설계할 수 있다.

집성화 관계는 a-part-of 나 partwhole로 표현되는 객체의 그룹화를 위한 관계 정의를 하는 분석과정이다. 자원관리의 BM구조와 같이 제안된 방법으로 설계할 수 있는 DB에서 쉬운 예를 찾아볼 수 있다. 링크와 결합을 사용자가 정의하는 내부 레코드의 관계와 DB에서 parent-child 관계와 같이 정의한다. 관계의 비중을 나타내는 차수와 대응수, 그리고 종속성이 정의된다. 객체 모델에서 일반화, 집합관계 및 링크와 결합관

계는 매우 중요하다.

(4) 동적 모델링 단계

동적 모델링 단계에서는 앞 단계에서 설명한 문제분석, 객체 추출과 행위 기술, 그리고 객체들 간의 관련성을 중요하게 생각하는 분석이 정적 모델링이라고 한다면, 시스템 전체의 생명주기를 모델링하는 것을 동적 모델링이라고 할 수 있다. 동적 모델링에서는 객체의 상태가 변화되어 가는 과정을 중요시한다. 객체의 상태는 기술문서에 나타난 전후 조건들로부터 파악된다. 정적 모델링이 객체의 구조만 관심을 두고, 시간 조건을 배제한 환경에서 설계된다면, 동적 모델링은 시간에 따라서 변하고 시스템의 관점에서 객체를 관찰하게 된다. 따라서 두 가지 모델링의 관점은 서로 간에 균형을 유지할 수 있도록 설계되어야 한다. 동적 관점의 설계 내용은 각 객체에 대한 상태를 정의하고 동적 행위를 가진 객체의 상태가 객체의 생명에 어떤 영향을 줄 수 있는가를 정의한다.

다. 여러 가지 객체 모델링의 비교 분석

(1) Edwards

분석결과에서 C++로 코딩할 수 있고 정적 및 정적 차원에서 지원할 수 있으나 승계 정의를 고려하지 않았다. 정적 차원의 지원에서는 클래스, 인터페이스, 관계, 기능(관계)이 있고, 동적차원의 지원에서는 다중 객체로 일어난 이벤트를 중심으로 분석하고, 단순 이벤트는 STD에 의해 분석한다. 이벤트의 5개 영역은 객체 생성, 용어의 정의, 클래스 확장, 클래스 재구성, 객체의 재분류 등이 있다. 여기서 클래스의 확장과 재구성, 객체 분류는 객체를 클래스로 구분하는 과정이다.

(2) Page와 Jones의 이벤트 영역 분류

이벤트의 영역 분류를 통해서 객체를 모델링하는 과정은 다음과 같다.

① 객체 생성

② 용어 정의

③ 객체 정의와 분류에 의한 클래스 정의

④ ③ 단계와 반복 순서

⑤ 재분류

⑥ 관계의 재정의

⑦ 서로 다른 객체의 표준화와 동향

(3) Shlaer와 Mellor의 OOA

객체의 상호 관련은 이벤트를 통해서 정의되고 이때 활동을 통해서 이벤트를 생성한다. 전이는 이벤트가 발생할 때 일어난다.

활동을 확장할 때 DFD를 채용하며, DFD는 외부 속성을 표현한다. STD에 의해서 객체의 라이프 사이클을 결정한다.

(4) Rumbaugh의 OMT

Rumbaugh는 활동상태와 활동을 구분하여 설명하고 있다. 활동은 전이 가능한 이벤트와 동시에 일어나고, 활동상태는 상태가 추가된 연산과정으로 생각한다. 객체의 상태

는 객체가 가지고 있는 다른 객체들 간의 결합으로 모은 집합이다. 활동상태는 전이 가능한 이벤트에서 끝나고, 항상 활동으로 바꿀 수 있다. 그래서 자료 동적 및 정적 성질을 표현하는데 중점을 두고 있다.

객체 지향 분석은 다음과 같이 4가지 과정을 반복해서 이루어진다.

① 객체 모델링

② 동적 모델링

③ 기능 모델링

④ 모델 간의 관계정립

특히 Booch와 Gibson은 서브시스템을 사용하여 객체를 정의함으로써 설계과정의 단계를 상세하게 추가하고 있다. 즉 확장 가능한 객체를 서브시스템에 정의하고 그 자료구조를 설계하고 행위의 알고리즘을 설계하면서 새로운 객체를 식별하는 기회를 갖게 한다. 본 연구에서는 Rumbaugh, Booch와 Gibson의 방법을 참조하면서 객체 모델링 방법을 설명하였다.

3. 객체 지향 분석 설계에 대한 비교 분석

가. 객체 지향 분석 및 설계 프로세스

OOAD 프로세스에서는 객체를 의미, 인터페이스, 응용, 베이스/유틸리티 클래스들로 분류하여 나타낸다. 그리하여 OOAD프로세스에서는 클래스로 정제하는 활동을 한다. 이것은 공통된 행위에 대한 클래스의 추상화와 애트리뷰트에 대한 구조를 조사하는 것이다. 또한 설계자의 관점에서는 클래스가 완벽하고 올바르다는 것을 확인할 수 있으며 “메소드에 누락된 것이 있는가,” “메소드와 애트리뷰트가 올바른 위치에 있는가”에 대해서 클래스를 정제하는 작업이다. 많은 연구자들에 의해 OOAD 프로세스에 대한 연구가 이루어졌는데 그 중에서 Booch, Coad and Yourdom, Rumbaugh, Bulmn 등이 OOAD프로세스에 대한 많은 부분을 설명하였다. 특히, Bulman은 인터페이스, 응용, 베이스, 유틸리티 객체에 중점을 두고 설명하였다.

나. OOAD 방법

표현은 크게 정적인 관점과 동적인 관점 그리고 표기법에 관한 관점으로 구분할 수 있다. 객체 지향 정적모델은 객체 지향 동적 모델보다 많은 종류의 모델이 있으며, 동적 모델 중에서 구성요소 대부분이 객체 모델에서 행위로 매핑되거나 번역되는 상태 전이도로 구성된다. 표기법에 관한 관점은 관계의 표현을 포함하는 것으로써 대부분의 표현들은 일반화 관계로 지원된다. 집단화 관계도 지원되지만 아직까지 널리 이용되고 있지 않다.

다. OOAD 복잡도 관리

몇몇 연구자들이 방대한 OOAD 복잡도를 관리하기 위한 개념적인 메커니즘을 제공하고 있는데 이 메커니즘의 대부분은 구조적인 복잡도에 대하여 처리하고 있다. 예를 들어 Coad and Youndom’s의 서브젝트와 Wirfs-Brock et al의 서브시스템이 있는데 서브젝트는 어떤 루트 구조나 서브스트락처에서 객체를 캡슐화하는 매카니즘이며 루트로부터 구조를 상속받아 구성되므로 복잡도가 감소한다. 반면에 서브시스템은 루트구조에 기반을 두지 않고 상위 레벨 기능을 달성하기 위해 구성된 다른 서브시스템이나 클래스로 이루어져 있다.

라. OOAD 장단점

현재의 OOAD에 접근하는 일반적인 세 가지 방법은 다음과 같다.

(1) 결합에 의한 접근

객체 지향, 기능 지향 그리고 동적 지향 기술을 단독으로 모델 구조, 기능성, 동적행위로 이용하고 다른 모델을 통합하기 위한 방법론을 제공한다. 그러나 이 결합에 의한 접근은 다른 관점을 객체 지향 설계방법으로 통합할 때 내용이 변경될 수 있다.

(2) 적용될 수 있는 접근

기존의 기술을 객체 지향에 이용하거나 객체 지향 기술을 포함하여 확정한다.

(3) 순수한 객체 지향 접근

객체구조, 기능성, 동적행위를 모델화하기 위해 새로운 기술을 이용한다. OOAD를 지원하기 위한 정형화된 방법은 아직 표준화되어 있지 않으나 지금까지 기술한 OOAD에 대한 장단점은 다음과 같다.

OOAD의 장점	OOAD 단점
- 의미 클래스의 명시 - 애트리뷰트와 행위의 명시 - 메소드의 배치(law of Demeter) - 일반화와 집단화 구조의 표현과 명사 - 정적 뷰의 표현(구조)	- 인터페이스, 응용, 시스템, 클래스의 명시 - 애트리뷰트, 관련성, 행위 등을 결정 - 클래스 배치 - 다른 종류의 관계성의 표현과 명시 - 관계성을 위한 일치되고 올바른 표현(메시지, 페싱, 제어 등) - 정적, 동적 모델 통합 - 추상화/ Granularity의 일치 레벨의 유지

❑ UML

1. UML의 역사

가. Grady Booch, James Rumbaugh와 Ivar Jacobson, 세 사람은 80년대부터 각자의 객체지향 방법론을 연구하였다.

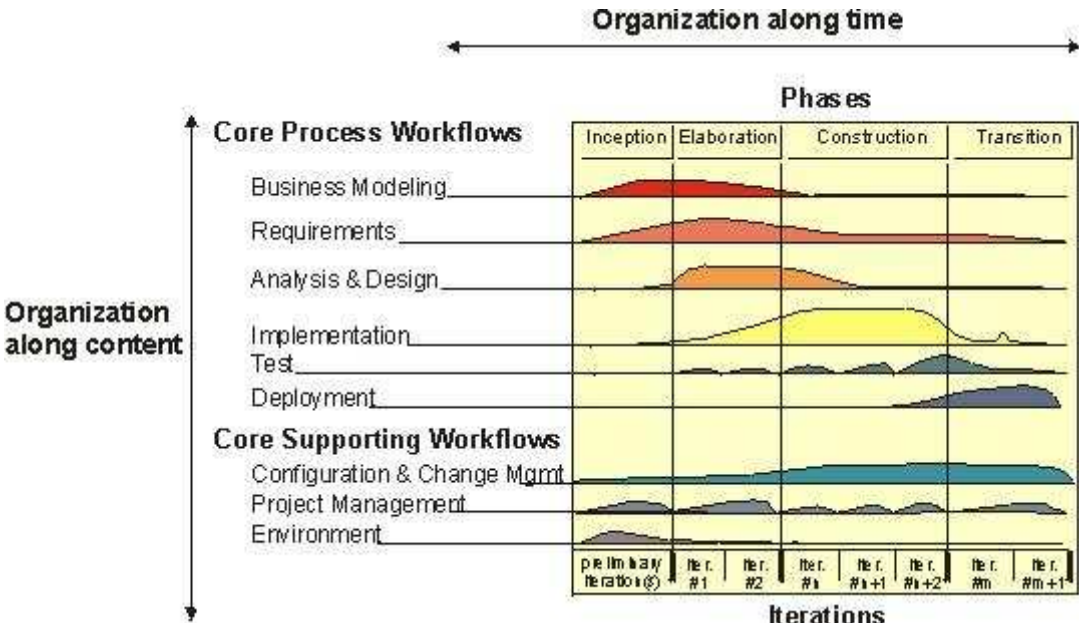
나. 1994년 Booch가 세운 Rational사에 Rumbaugh가 합류하고, 일년 후 Jacobson이 합류하면서 이들의 연구는 하나로 결집되어 UML 드레프트(draft) 버전을 만들어냈다.

다. 이것은 소프트웨어업계에 큰 반향을 일으키며 Microsoft, Oracle, HP, DEC, TI 등 유수의 멤버로 결성된 UML 컨소시엄을 발족하게 된다.

라. 1997년 UML 컨소시엄은 UML 버전 1.0을 만들어 내고 이를 OMG(Object Management Group)에 제출한다.

마. 그 해 말에 OMG는 이를 수정한 UML 1.1을 표준 모델링 언어로 채택되었다.

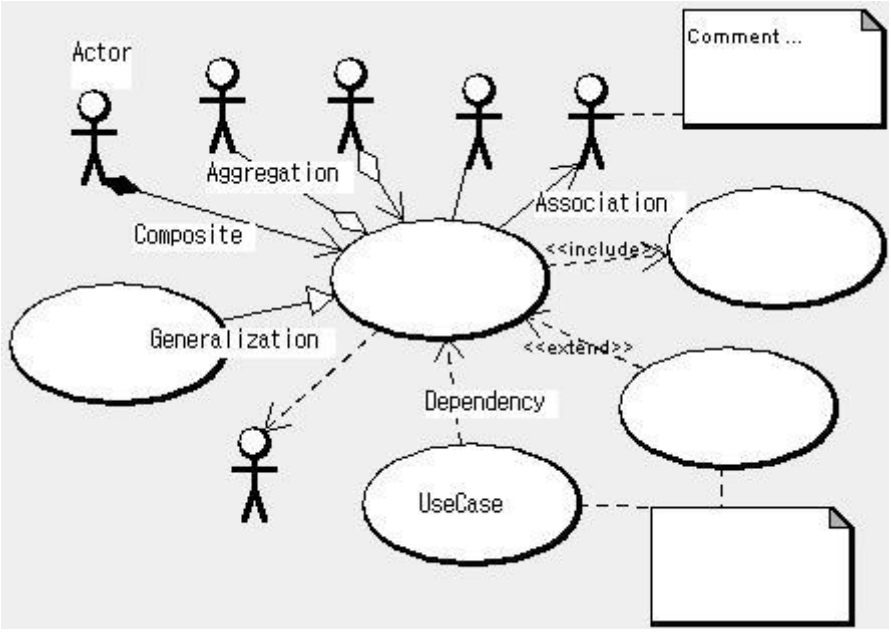
2. RUP (Rational Unified Process)



가. 시간의 흐름에 따른 가로축의 4단계(Phase)와 작업의 성격에 따른 세로축의 9개 워크플로우로 구분

나. 각 단계별로 각 워크플로우가 반복(Iteration)적으로 수행되는 것이 RUP의 방법론

3. Use Case Diagram



가. Actor

나. Use Case

다. Relation

(1) Actor와 UseCase간의 관계 : Association, Aggregation, Composition

=> 방향성에 따라 Bi-directional, Uni-directional

(2) UseCase간의 관계 : Generalization, Extend, Include

(3) 공용 : Dependency

4. Activity Diagram

가. Activity Diagram 개요

(1) 정의 : 처리 로직이나 조건에 따른 처리흐름을 순서에 따라 정의한 모델

(2) 작성목적

- 대상에 관계없이 처리순서 표현
- **비즈니스 프로세스 정의(이 용도로 가장 많이 사용됨)** : 업무의 As-is분석, To-be 분석 가능
- 프로그램 로직 정의 : 처리흐름의 도식화로 프로그램 로직 정의 가능
- 유즈케이스 실현

(3) 작성 시기 : 그 시점이 한정되어 있지 않고 다양하게 사용 가능

- 업무 프로세스 정의 시점.
- 유즈케이스 정의서 작성 시, 처리절차 기술할 때

- 오퍼레이션 사양 정의시

(4) 작성순서

- 작성대상 선정 : 업무프로세스 모델링, 오퍼레이션 사양 정의

↓

- Swim lane 정의 : 대상영역에 명확한 역할을 정의해야 할 때.

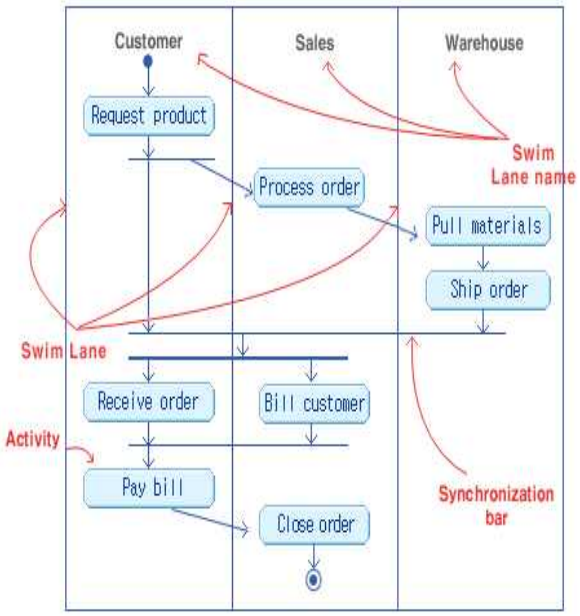
↓

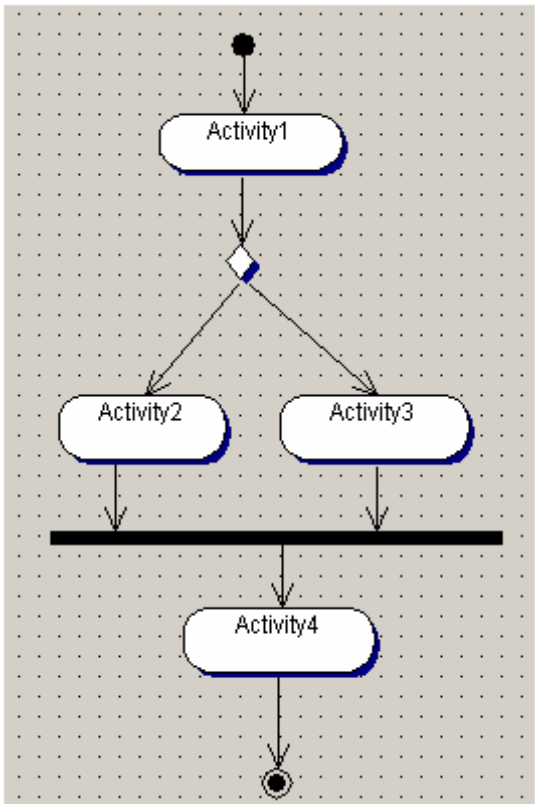
- 처리절차 모델링 : 시작점, 끝점 반드시 표현.

나. Activity Diagram 구성요소 및 표기법

이 다이어그램에서는 처리과정에 포함되어질 소시지 모양의 '**활동**'과 마름모 모양의 '**조건**' 둥근모양의 '**시작점**','**종료점**' 그리고 긴 얇은 직사각형 모양의 '**동시경로**' 로 구성된다.

아래로 향하는 화살표는 진행과정(**방향**)을 나타낸다.





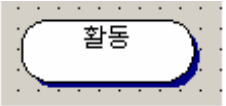
(1) 시작점, 종료점

- 당연한 말이지만 모든 과정에는 시작과 끝이 존재한다.
- Activity Diagram 에서의 시작과 종료는 아래와 같은 모양으로 표시한다.

● : 시작점

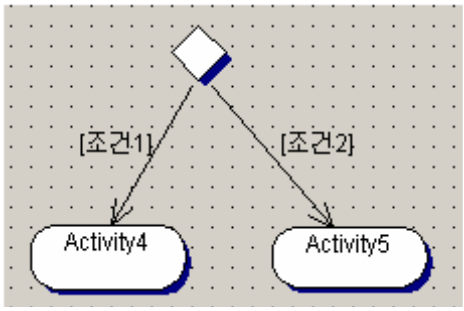
⊙ : 종료점

(2) 활동(동작)



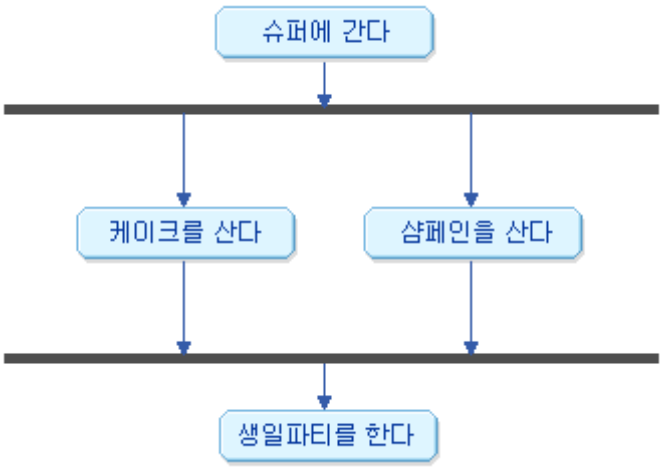
처리과정 중 각 단계별로 행해지는 동작(활동)을 나타낸다. 왼쪽 그림처럼 소시지 모양으로 표시한다.

(3) 조건



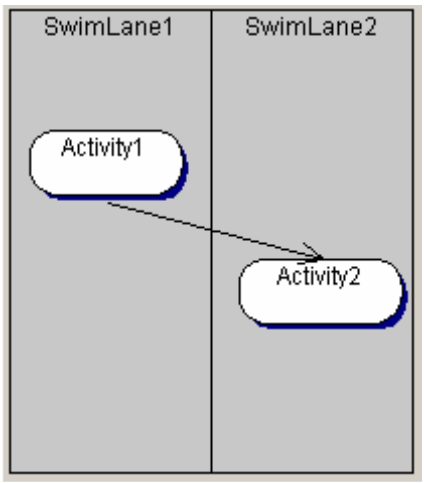
처리과정에는 특정 조건에 의한 분기를 포함할 수 있다. 조건에 의해 분기되는 지점을 '결정위치' 라고 하는데 아래처럼 마름모꼴로 표시한다. 이 마름모꼴의 결정위치에서 조건에 의해 처리경로가 분기되는데 이때 조건은 분기된 화살표에 대괄호로 감싸서 표시한다.

(4) 동시경로



처리과정 중에는 특정 활동(동작)이 동시에 같이 실행되다가 하나로 모이는 경우가 발생할 수 있다. 이 경우 두 개의 처리경로를 동시에 실행되는 부분을 속이 찬 긴 직사각형으로 표시한다. 또한 두 경로가 다시 하나로 합쳐질 경우에도 이 표식이 사용된다.

(5) 구획면



누가 그 활동(동작)을 하는가를 명확히 구분할 수 있는 방법을 제공해 준다. Swim Lane 이라고 하는 구획면을 이용하면 된다. 즉, 처리과정 중에 발생하는 각 동작(활동)의 책임이 누구에게 있는지를 나타낸다. 구획면은 사람이 될 수도 있고 시스템이 될 수도 있다. 각 동작을 수행하는 주체이기만 하면 된다.

(6) 신호

그리고 **'신호'**라는 것도 있는데 이것 역시 Activity Diagram 의 구성요소이다. (개인적으로 시그널 정보를 포함한 Activity Diagram 을 쉽게 볼 수는 없었다)

각 동작(활동)이 처리되는 과정 중에 신호를 보낼 수 있다. 신호가 보내어지면 그 신호를 받은 쪽은 활동을 개시해야 한다.

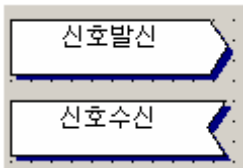
시그널은 다음과 같은 경우에 사용하면 다이어그램이 보다 상세해 질 수 있다.

① 비동기적인 흐름을 나타내고 싶을 경우

② 각 활동(단계) 간의 이동 중 발생하는 상황을 보다 명확히 하고 싶을 경우

③ 기타 개인적인 목적에 의해

Activity Diagram 에서 신호는 다음 그림처럼 오각형 모양으로 표시한다.



- 신호발신 : 출력 사건(output event)
- 신호수신 : 입력 사건(input event)

5. Class Diagram

가. Property

(1) class의 구조적인 특성을 나타낸다.

(2) 두 가지의 표기법으로 구성된다.

- 속성 (Attributes)
- 연관 (Associations)

나. Attributes (속성)

(1) Class box에 한 줄로 표시된다.

(2) 표현 방법

- visibility name : type multiplicity = default {property-string}
- Ex) - name : String [1] = "Untitled" {readOnly}
- Class Diagram 그림

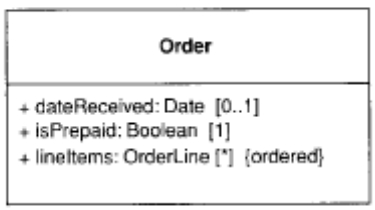


Figure 3.2 *Showing properties of an order as attributes*

다. Associations (연관)

(1) Property를 표시 하는 또 다른 방법

(2) 저자는 보통 Type이나 Boolean 같은 일반적인 것은 Attributes 방법으로 표시 하고, 중요한 class 는 Association 방법으로 표시 한다.

(3) Association 그림

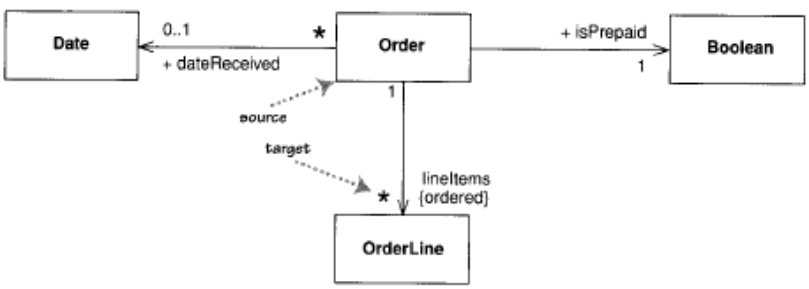


Figure 3.3 *Showing properties of an order as associations*

라. Multiplicity (다중성)

(1) Property내에 들어 갈 수 있는 객체의 수

(2) 속성 부분의 예제 중 String[1] 부분 이다. - name : *String [1]* = "Untitled" {readOnly}

- Mulitplicity 항목
  - ① Optional - 하한이 0
  - ② Mandatory - 하한이 1
  - ③ Single-valued - 상한이 1
  - ④ MultiValued - 상한이 1 이상(흔히 '*'로 표현)

(3) 속성에 순서가 의미가 있는 경우 **{ordered}**를 붙인다.

(4) 속성에 중복을 허용 한다면 **{nonunique}**를 붙인다.

(5) 기본값을 명확하게 표현하려면 {unordered}, {unique} 사용

(6) 중복을 허용하는 경우 {bag}

마. Bidirectional Associations

(1) class간에 서로 연관된 property

(2) 예제 diagram (일반적으로 Property Type이 복수인 경우 cars 라고 naming한다.)

Figure 3.4 *A bidirectional association*

(3) 속성을 명확하게 표현하기 위해 방향성 화살표를 추가 한다 (일반적으로 화살표 연관 표시는 동사형으로 표시).

Figure 3.5 *Using a verb phrase to name an association*

(4) 양방형 association의 경우 association의 한쪽이 둘 간의 관계를 통제해야 한다.

바. Operation

(1) Class가 수행 하는 action

(2) Operation의 표기

- visibility name (parameter-list) : return-type {property-string}
- Parameter 표기
  - + direction name : type = default value
- Ex) + balance0n (date : Date) : Money

(3) Visibility

- '-': Private, '#': Protected, '+': Public

(4) Operation의 Type

① 시스템의 상태를 변경하는 Operation - {query} 로 표시

② 시스템의 상태를 변경하지 않는 Operation - {modifier} 또는 {command}

- {query} 와 {command} 판별 기준은 시스템 변경 여부를 외부에서 관찰 가능 여부에 따른다.
- Cache 내용이 변경 되었어도 외부에서 변경 여부를 확인 할 수 없으므로 {query}에 해당한다.
- **일반적로 modifier 또는 command는 void 형을 리턴 하는 것으로 정의 한다.**

(5) Operation과 Method의 차이

① Operation - 객체에서 호출되는 Procedure

② Method - Procedure 자체

③ 3가지 각각이 supertype의 getPrice 함수를 override하는 subtype과 하나의 supertype을 가지고 있다면, 당신은 한 가지 명령에 네 가지의 그것을 구현하는 함수를 가지게 된다.

사. Generalization

(1) Ex) 기업 고객과 일반 고객

- 기업고객과 일반 고객 class가 customer class로 부터 상속을 받은 경우 기업 고객은 일반고객의 인스턴스가 가능하다.

(2) 소프트웨어 관점에서 일반화는 상속과 연관이 있다.

(3) 상속의 효과적인 사용은 substitutability 이다.

(4) 위의 예제에서 기업 고객은 고객으로 대치가 가능하며, 이상 없이 동작해야 한다.

(5) inheritance는 강력한 방법이지만 substitutability를 만족시키는데 overhead가 존재 한다.

(6) 디자인 패턴에는 subtyping without subclassing 방법들이 많이 존재 한다.

아. Note &Comment

- 단독으로 사용하거나 설명하고자 하는 요소에 점선으로 연결하여 사용한다.

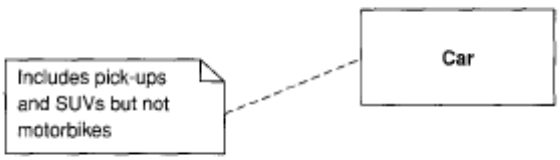


Figure 3.6 *A note is used as a comment on one or more diagram elements*

자. Dependency

(1) Dependency가 존재 하는 것은 한부분의 변화가 다른 한부분에 영향을 미치는 것이다.

(2) 그림으로 표현된 아래 예제는 multilayered app.에서 발생할 수 있는 dedpedency를 보여준다.

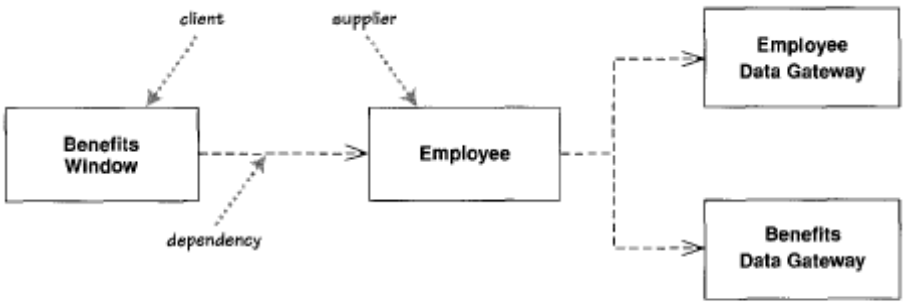


Figure 3.7 *Example dependencies*

(3) Dependency는 방향성을 가진다.
(4) 위의 예제에서 Benefits Window 는 Employee Data Gateway, Benefits Data Gateway와는 간접적인 dependency를 가진다.
(5) 프로젝트 수행시 가능하면 dependency를 줄이는 것을 base rule로 삼아야 한다.
(6) **Class Diagram에서 모든 dependency를 표현하는 것은 쓸모없는 일이다.**
(7) **전달하고자 하는 특별한 부분과 밀접한 연관이 있을 경우에만 선택적으로 dependency를 표현 하는 것을 추천한다.**
(8) 저자가 dependency를 사용하는 경우
- 어떤 객체가 다른 객체에 parameter로 전달되는 경우에 사용한다.
- 이 경우 <<parameter>>, <<local>>, <<global>> 과 함께 사용하게 된다.
(9) Dependency 분석은 툴을 이용하는 것이 이상적이다.
(10) Dependency에서 사용하는 Keyword 일부이다.

Table 3.1 *Selected Dependency Keywords*

Keyword	Meaning
«call»	The source calls an operation in the target.
«create»	The source creates instances of the target.
«derive»	The source is derived from the target.
«instantiate»	The source is an instance of the target. (Note that if the source is a class, the class itself is an instance of the class class; that is, the target class is a metaclass).
«permit»	The target allows the source to access the target's private features.
«realize»	The source is an implementation of a specification or interface defined by the target (page 69).
«refine»	Refinement indicates a relationship between different semantic levels; for example, the source might be a design class and the target the corresponding analysis class.
«substitute»	The source is substitutable for the target (page 45).
«trace»	Used to track such things as requirements to classes or how changes in one model link to changes elsewhere.
«use»	The source requires the target for its implementation.

차. Constraint Rules

(1) Association, Property, Generalization 은 제약을 명시 하는데 중요한 역할을 한다.
(2) Class Diagram은 제약을 표현 하는데 중요한 수단이다.
(3) Constraint 을 사용하는 규칙은 '{}' 안에 constraint를 명시 한다.

카. When to Use Class Diagrams

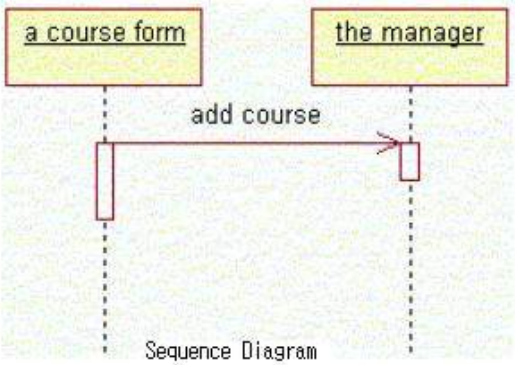
(1) Class Diagram은 UML의 중요한 요소(backbone) 이다.
(2) Class Diagram은 단순한 것부터 시작해야 한다.
(3) 모든 것에 모델을 그리지 말고, 중요한 부분에 집중해야 한다.
(4) 다수의 diagram 보다 적은 수의 diagram을 사용 하면서 갱신하는 것이 좋다.
(5) **Class diagram을 사용할 때는 구조에 너무 신경을 써서 action을 무시해서는 안 된다.**

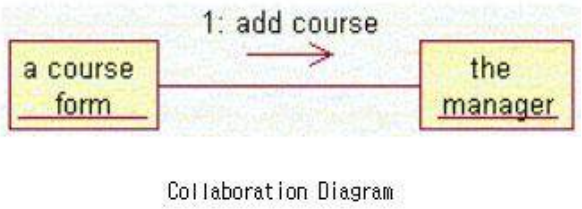
6. Sequence Diagram &Collaboration Diagram

가. 공통점 : 하나의 Use Case 내에서의 객체들 간의 상호작용 (Object Interaction)을 표현

나. 차이점 : Sequence Diagram은 시간의 흐름에 따른 상호작용을 표현

Collaboration Diagram은 상호작용의 내용과 순서만을 표현



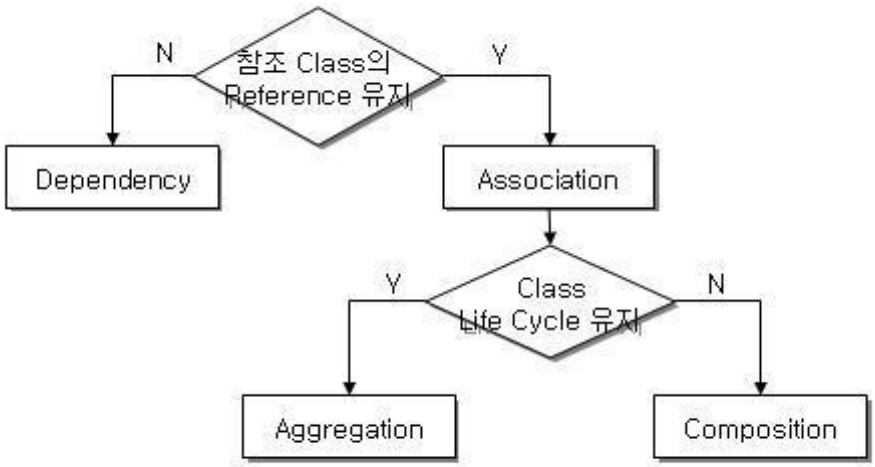


7. Class 간의 Relationship

가. Relationship 종류

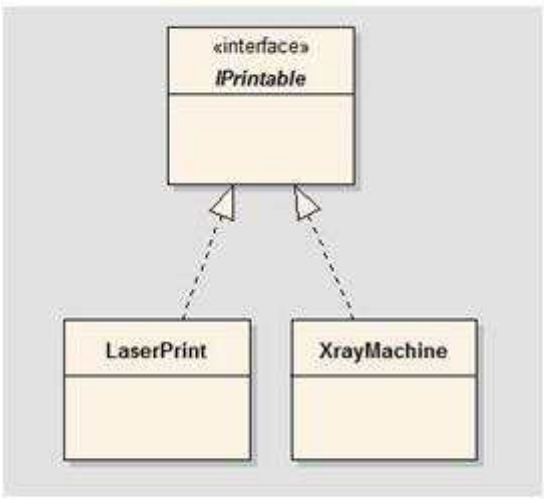
Class의 관계에는 크게 **인터페이스(Interface), 상속(Inheritance), 의존(Dependency), 연관(Association), 집합(Aggregation), 합성(Composition)**과 같이 6가지 종류가 있는데 **이중에서 비슷한 의미를 가지고 있는 Association, Aggregation, Dependency, Composition** 관계에 대해 좀 더 자세하게 다루도록 한다.

나. Relationship 구분도



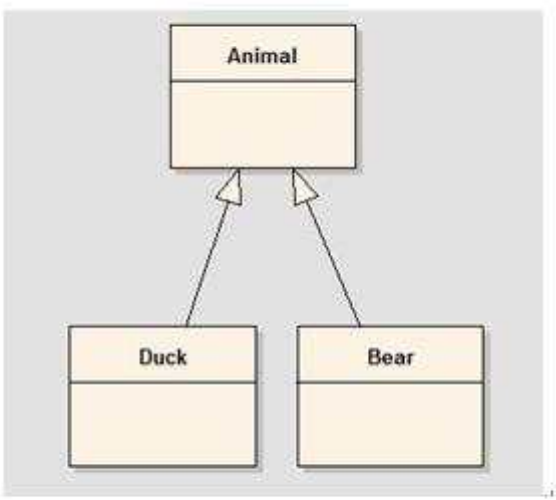
다. Interface와 Inheritance 표현

[Interface]



연산만 존재하고 Method가 존재하지 않는 Class의 참조.

[Inheritance]



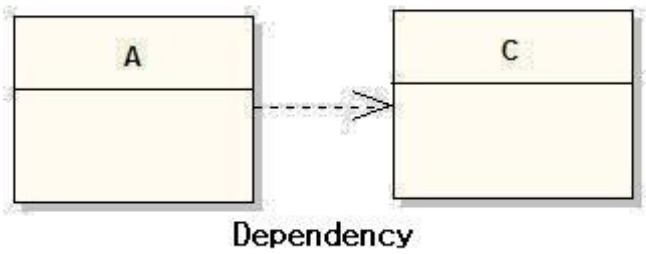
공통된 속성과 Method가 존재하는 Class의 참조.

(1) Dependency와 Association의 구분은 참조하는 Class의 Reference를 계속 유지하고 있을 경우 Association 이고 그렇지 않을 경우 Dependency이다. 또한 Association은 클래스의 속성으로 표현되고 양방향 가능하지만 Dependency는 연산의 인자, 지역 객체, 전역 객체로 표현되고 단방향으로만 참조 가능하다.

① [Dependency]

```
Class_A


Class_B
{
        Private
                Void ObjA_Call()
                {
                        // Method내부에서만 Class_A의 Reference를 유지함
                        // Method가 끝나면 Class_A의 Reference도 소멸됨
                    Class_A * ObjA;
                    ObjA = new Class_A();
                 }
}
```



Dependency

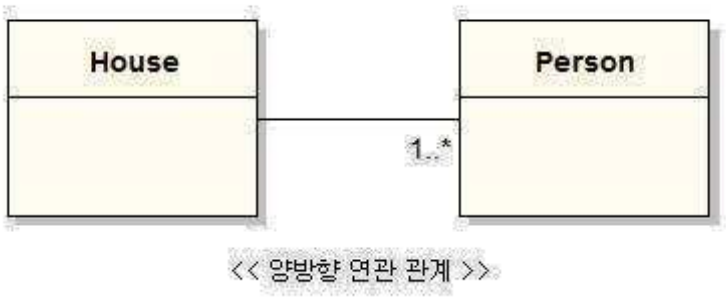
- Class에 변경이 생겼을 때 그 Class를 참조하는 다른 Class도 같이 영향을 받을 경우
- 한 Class의 Method 내에서 다른 Class의 객체를 받아 해당 객체의 Method를 호출 할 경우
- 한 Class Method 내에서 다른 Class의 객체를 반환 할 경우
- 한 Class의 전역 객체

② [Association]

```
Class_A
{
        Public
                Call();
}
Class_B
```

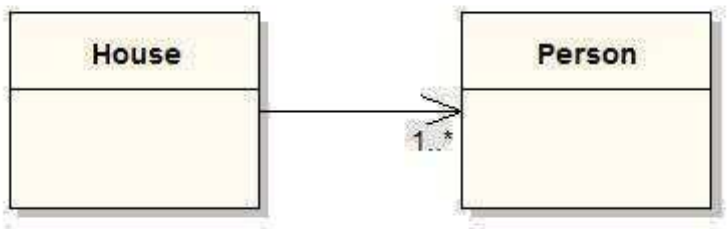
```
{
        Private
                Class_A                 * ObjA;
                Void Create_Obj()
                {
                        // Class_A의 Reference를 계속 유지함
                        ObjA = new Class_A();
                }
                Void ObjA_Call()
                {
                        ObjA->Call();
                }
}
```

- 한 객체가 다른 객체와 연결되어 있음을 나타날 때
- 연관 관계의 방향(Navigability)과 다양성 (Multiplicity)이 중요한 관점



<< 양방향 연관 관계 >>

- 양방향 연관관계 : 연결된 2 Class가 모두 서로를 참조 하고 있음(좋지 못한 관계)
  (한 사람 이상이 House에 속해야 함)



<< 단방향 연관 관계 >>

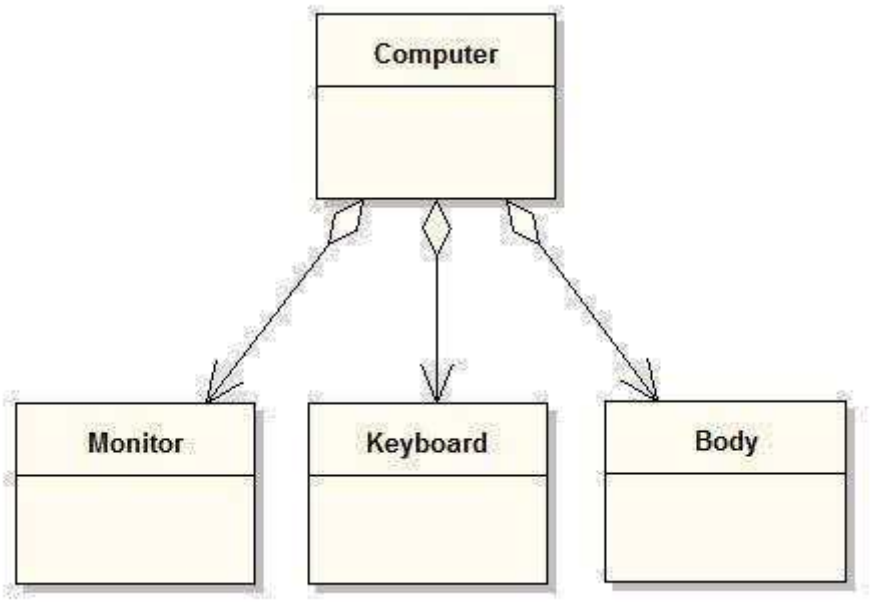
- 단방향 연관관계 : House Class는 Person Class를 참조하고 있지만 Person Class는 House Class를 참조하지 않음

(4) Aggregation과 Composition의 구분은 참조 Class의 Life Cycle이 호출 Class와 동일 할 경우 Composition 이고 다를 경우 Aggregation이다 즉, Composition관계는 참조 Class와 호출 Class의 생성과 소멸이 동일 하지만 Aggregation관계는 호출 Class가 소멸하더라도 참조 Class는 계속 유지 되는 관계이다.

① [Aggregation]

```
Class_A
Class_B
Class_C

Class_D
{
   Private
                Class_A * ObjA;
                Class_B * ObjB;
                Class_C * ObjC;
                Void Aggre( Class_A A, Class_B B, Class_C )
                {
                        // 부분이 되는 객체를 외부에서 생성하여 넘겨받음
                        // Class D가 소멸 되더라도 외부에서 생성된 Class_A,
                           Class_B, Class_C는 소멸 되지 않음
                       ObjA = A;
                       ObjB = B;
                       ObjC = C;
                }
}
```



- 부분 객체는 여러 전체 객체에 의해서 공유 될 수 있음

② [Compostion]

Class_A

Class_B

Class_C

Class_D

{

Private

Class_A * ObjA;

Class_B * ObjB;

Class_C * ObjC;

Void Composite()

{

// 강한 집합체의 의미

// 부분을 이루는 객체가 없이는 전체 객체의 의미가 없음

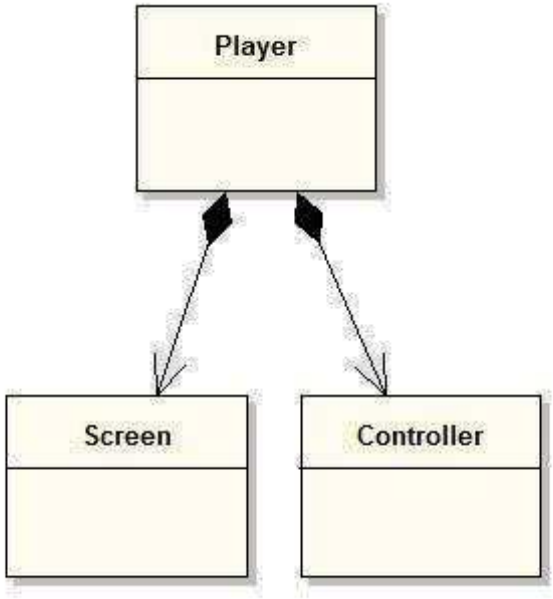
// Class_D가 소멸되면 Class_A, Class_B, Class_C도 함께 소멸됨

ObjA = new Class_A();

ObjB = new Class_B();

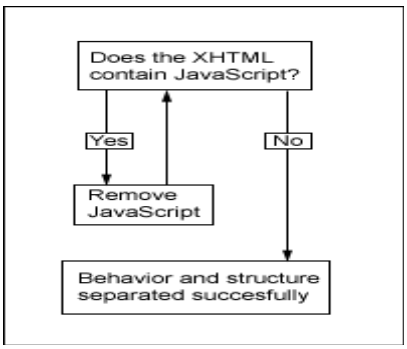
ObjC = new Class_C();

```
        }
    }
```



- 부분 객체는 전체 객체에 전속

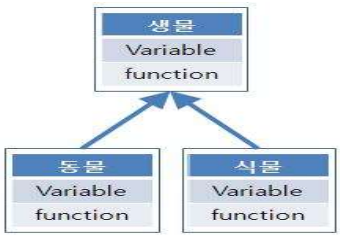
8. Behavior and Structure



가. 객체의 구조는 Attribute 로 표현하고, Behavior 는 Operation 을 생성한다. 객체간의 상호작용에서 전달되는 메시지는 메시지를 받는 객체에게는 Operation이 되는 경우가 많다.

9. Inheritance

가. 상속의 표기법



나. 상속관계를 끌어내는 법 : 일반화(Generalization)와 전문화(Specialization)

다. 기본적으로 정적인 바인딩(Static Binding)을 한다. 따라서 실행 중에 타입을 변환하는 일은 무척 어렵다.

라. 수강신청 예제에서 학교의 교수진은 전임교수와 시간 강사가 있다고 하자. 그런데 시간 강사이던 사람을 학교 측에서 전임교수로 임명했다고 한다.

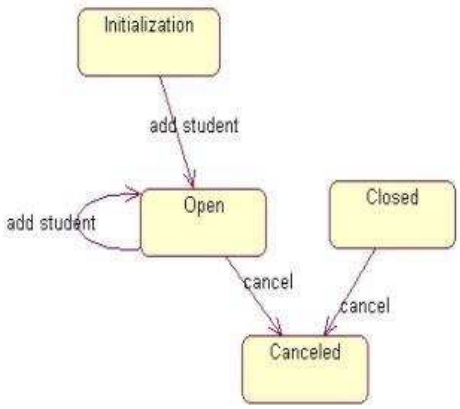
마. 이런 경우에 시간 강사인지 전임교수인지를 구분해주는 **역할(Role)을 다른 클래스에 위임(Delegation)**해서 문제를 해결할 수 있다. 이렇게 하면 실행 중에 해당 강사의 역할에 따라 동적으로 바인딩이 가능하게 되는 것이다.

10. Object Behavior

가. 유스케이스(Use case)는 시스템 레벨의 행위(Behavior)를 표현한 것이다. 유스케이스에서출발해서 이들 행위를 달성해야 하는 시스템의 책임(Responsibility)을 객체들에 나눠주고 이들 간의 협력(Collaboration)이 모여서 시스템 레벨의 행위를 달성한다.

나. 결국 객체지향 기술에서 시스템을 이루는 단위는 '객체'이다. 즉, 객체의 행위 수준까지 분석을 해야 한다.

다. 객체는 내부의 상태(state)와 내부의 행동에 의한 상태변화(state transition) 으로 분석한다.



11. Architecture

가. Architecture는 여러 가지 측면에서 시스템을 조망한 것이다. 다음은 Rational의 Philippe Krutchten 이 제안한 4+1 View 이다.

(1) 논리적 관점(Logical View) 혹은 디자인 관점(Design View)

- 시스템의 주요한 기능적인 요구사항에 대한 것

  기능적인 요구사항 하나하나를 다루는 것이 설계라면 전체적인 시스템의 기능에 영향을 미치는 설계 전략(design strategy)을 결정하는 것이 아키텍처라고 할 수 있다.

(2) 구현 관점(Implementation View)

- 구현 관점의 아키텍처는 실제 소프트웨어 모듈의 조직에 관한 것이다.
- Rose에서 구현관점은 'Component View' 패키지 내에서 컴포넌트 모델로 표현된다.

(3) 프로세스 관점(Process View)

- 프로세스 관점은 구현관점과 유사한다. 다만, 구현관점이 개발 과정에서의 소프트웨어

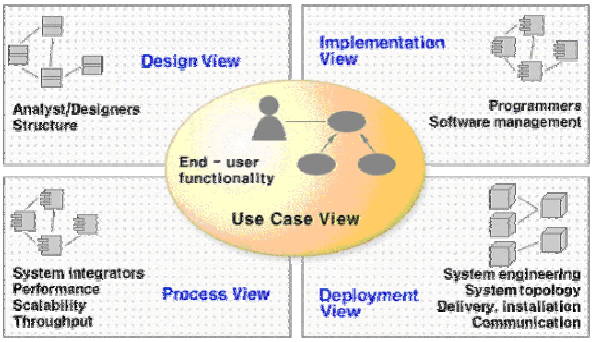
구조라면 프로세스관점은 실행 시(run-time)를 다룬다는 것이 차이점이다.

(4) 배포 관점(Deployment View)

- Rose에서는 'Deployment View'라는 다이어그램을 기본적으로 갖고 있어 배포도를 나타내도록 하고 있다.

(5) 유스케이스 관점(Use Case View)

- 유스케이스 관점의 키워드는 '통합'이다. 아키텍처에 대한 이들 각각의 관점은 결국 유스케이스 관점에서 평가하고, 검증할 수 있다는 것이다.



12. Iteration Planning

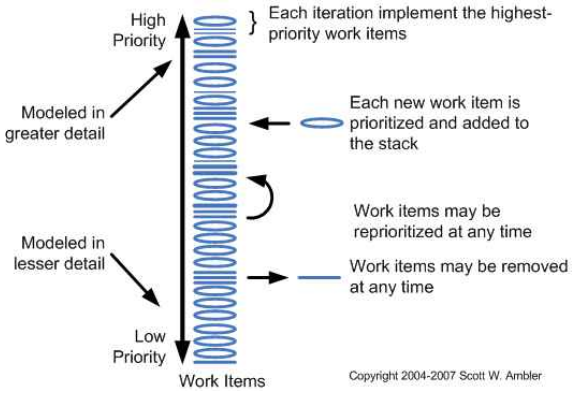
가. 객체지향 분석 및 설계의 대가인 Booch에 따르면 iteration 계획에는 기본적으로 다음의 세 가지 사항이 명시되어야 한다고 말한다.

(1) Capabilities being developed

(2) Risks being mitigated during this iteration

(3) Defects being fixed during this iteration

나. Iteration planning : 가장 위험한 요소부터 제거하는 방향으로 Iteration 수행



❑ StarUML

1. StarUML™ 개요

이 장에서는 StarUML™의 전반적이고 개략적인 내용들을 다룬다. StarUML™의 간략한 소개, StarUML™의 새로운 특징 및 전반적인 구성들을 설명한다.

가. StarUML 개요

StarUML™은 UML(Unified Modeling Language)을 지원하는 소프트웨어 모델링 플랫폼이다. UML 버전 1.4에 기반을 두고 있으며, UML 버전 2.0의 표기법을 적극적으로 지원하고 있다. 총 11가지의 다양한 종류의 다이어그램을 제공할 뿐만 아니라 UML 프로파일 개념과 템플릿 기반의 문서 및 코드 생성을 지원하여 MDA(Model Driven Architecture) 접근방법을 적극적으로 지원한다. 또한 고객의 환경에 대한 맞춤 능력이 우수하고 기능에 대한 확장성이 매우 뛰어난 것이 장점이다. 가장 선도적인 소프트웨어 모델링 도구 중의 하나인 StarUML™을 사용하면 소프트웨어 프로젝트의 생산성(Productivity), 품질(Quality)이 획기적으로 높아진다는 것을 실감하실 수 있다.

(1) 고객에 적응하는 UML 도구

StarUML™은 고객의 환경에 최대한 적응할 수 있도록 설계되어 있다. 따라서, 고객의 소프트웨어 개발 방법론, 프로젝트의 플랫폼, 언어 등에 모두 적응할 수 있는 커스터마이징 변수들을 제공한다.

(2) 진정한 MDA 지원 도구

소프트웨어 아키텍처는 향후 10년 이상 내다보는 매우 중요한 작업이다. OMG에서는 MDA 기술을 통해서 플랫폼에 독립적인 소프트웨어 모델을 구성하고 그것으로부터 플랫폼에 의존적인 모델이나 코드 등을 자동으로 얻을 수 있도록 하는 것을 지향하고 있다. StarUML™은 UML 1.4 표준 메타모델과 2.0 표기법을 최대로 준수하면서 UML Profile 개념을 제공하여 플랫폼에 독립적인 모델을 작성할 수 있도록 지원하며, 간단한 템플릿 문서 작성만으로 고객이 원하는 산출물을 쉽게 얻을 수 있다.

(3) 놀라운 확장성과 유연성

StarUML™은 놀라운 유연성과 확장성을 제공한다. 도구의 기능을 확장하기 위한 Add-In 프레임워크를 제공하고, COM Automation을 통한 모델/메타모델 및 도구의 모든 기능에 접근할 수 있으며, 메뉴 및 옵션 항목까지도 확장할 수 있도록 설계되어 있다. 또한 고객의 방법론에 맞도록 접근법(Approach) 및 프레임워크(Framework)를 직접 추가 작성할 수 있고 어떠한 외부 도구와도 통합이 가능하다.

나. 새로운 특징

StarUML™에는 다음과 같은 새로운 특징들을 제공한다.

특징	내용
정확한 UML 표준 모델	OMG에서 제정한 UML의 표준 명세에 따라 소프트웨어 모델을 작성할 수 있도록 한다. 외국산 제품과 같이 변칙적이며 벤더에 의존적인 UML 구문과 의미는 설계한 정보의 지속성을 향후 10년이상 내다 볼 때 매우 위험한 선택이 될 수 있다. StarUML™은 UML 1.4 표준 구문과 의미의 준수를 극대화하였으며, 견고한 메타모델의 기반에서 UML 2.0의 표기법을 적극적으로 수용하였다.
개방적 소프트웨어 모델 포맷	독자적인 포맷으로 작성된 모델의 활용성을 크게 떨어뜨리는 외국산 제품과는 달리 StarUML™에서의 모든 파일의 포맷은 XML로 구성된다. 또한 사람

	이 쉽게 식별할 수 있는 형태로 표현되어 있어서 누구든지 XML 파서를 이용해서 포맷을 원하는 형태로의 변환 및 사용이 가능한다. XML은 세계 표준인 만큼 장기적인 안목으로 본다면 10년 이상 지속될 수 있는 소프트웨어 모델이 될 수 있다.
진정한 MDA 지원 도구	UML 프로파일(UML Profile)을 완벽하게 지원하여 UML의 확장성을 극대화시킴으로써 금융, 국방, e-비즈니스, 보험, 항공우주 등 어떠한 영역의 애플리케이션도 모델링이 가능하게 된다. 진정한 의미의 플랫폼 독립적인 모델(PIM – Platform Independent Model)의 작성을 가능하게 하고 그로부터 플랫폼 의존적인 모델(PSM – Platform Specific Model)이나 각종 문서, 실제 실행 가능한 코드(Executable Code)를 얼마든지 자동으로 생성할 수 있다.
방법론 및 플랫폼의 적응성	StarUML™은 접근법(Approach)이라는 개념을 도입하여 어떠한 방법론/프로세스에도 적응할 수 있는 환경을 만들 수 있다. .NET, J2EE와 같은 플랫폼의 애플리케이션 프레임워크(Framework) 모델 뿐만 아니라 소프트웨어 모델의 기본 구조(e.g. 4+1뷰-모델 등)를 쉽게 정의할 수 있다.
뛰어난 확장성	StarUML™ 도구의 모든 기능이 Microsoft의 COM 자동화(Automation)가 되어 있어 어떠한 COM 지원 언어(Visual Basic Script, Java Script, VB, Delphi, C++, C#, VB.NET, Python, …)에서도 StarUML™을 제어하고 또한 통합된 추가 모듈을 개발할 수 있다.
소프트웨어 모델 검증 기능	사용자는 소프트웨어 모델링을 수행하는 동안 많은 실수를 범하게 된다. 이러한 실수가 최종 코딩단계로 그대로 전가될 때에는 더 큰 위험을 초래할 수 있다. 이를 방지하게 위하여 StarUML™은 사용자가 개발한 소프트웨어 모델을 자동으로 검증(Verification)하여 사전에 오류를 발견하게 함으로써 더욱 견고하고 완벽한 소프트웨어 설계를 수행할 수 있도록 도와준다.
유용한 Add-In들	StarUML™ 은 프로그래밍 언어의 소스코드를 생성하거나 소스코드를 모델로 변환하는 기능을 제공하는 다수의 Language Add-In들과 Rational Rose 파일 읽기, XMI를 통한 도구간 모델링 정보 교환, 그리고 디자인 패턴 지원 등의 각각의 기능을 제공하는 유용한 Add-In들을 빌트-인(Build-In)으로 제공한다. 이런 Add-In들을 활용하여 모델링한 정보의 재사용성, 생산성, 가용성, 상호 운용성을 높일 수 있다.

다. 시스템 요구사항

StarUML™을 구동하기 위해서 귀하의 시스템이 갖추어야 할 최소한의 요구사항을 나타낸다.

- Intel® Pentium® 233MHz or higher
- Windows® 2000, Windows XP™, or higher
- Microsoft® Internet Explorer 5.0 or higher
- 128 MB RAM (256MB recommended)
- 110 MB hard disc space (150MB space recommended)
- CD-ROM drive
- SVGA or higher resolution monitor (1024x768 recommended)
- Mouse or other pointing device

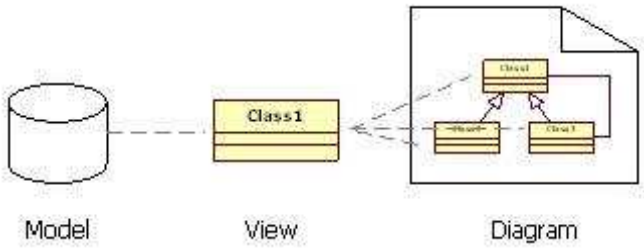
2. 기본 개념

이 장에서는 StarUML™을 효과적으로 사용하기 위해 필요한 필수적인 개념들을 소개한다. StarUML™에서 사용되는 기본 개념인 모델과 뷰 그리고 다이어그램에 대해 알아보고, 프로젝트와 유닛, 모델 조각의 운용방법을 기술한다. 또 확장 모듈과 거기에 포함되는 접근법, 프로파일, 프레임워크 등에 대한 개략적인 개념을 설명한다.

- 모델, 뷰, 그리고 다이어그램
- 프로젝트와 유닛
- 모듈

가. 모델, 뷰, 그리고 다이어그램

StarUML™에서는 모델과 뷰 그리고 다이어그램의 개념을 서로 분리해서 사용한다. 우선, 모델(Model)은 소프트웨어 모델에 관한 정보를 담고 있는 요소를 의미한다. 뷰(View)는 모델이 담고 있는 정보를 시각적으로 표현한 것을 의미하며, 다이어그램(Diagram)은 뷰 요소들의 집합으로써 사용자의 일정한 생각을 표현한 것을 의미한다.



나. 프로젝트와 유닛

(1) 프로젝트

프로젝트(Project)는 StarUML™에서 다루는 가장 기본이 되는 단위를 의미한다. 프로젝트는 하나 혹은 그 이상의 소프트웨어 모델들을 관리할 수 있으며 항상 존재하는 최상위 패키지(Top-level Package)로도 이해될 수 있다. 하나의 프로젝트는 일반적으로 하나의 파일에 저장된다.

(2) 프로젝트 구성

프로젝트 하위에는 다음의 요소들이 포함되며 관리될 수 있다.

프로젝트 하위요소	설명
모델(Model)	하나의 소프트웨어 모델을 관리하기 위한 요소이다.
서브시스템(Subsystem)	하나의 서브시스템을 표현한 모델들을 관리하기 위한 요소이다.
패키지(Package)	요소들을 관리하기 위한 가장 일반적인 요소이다.

(3) 프로젝트 파일

프로젝트 파일은 XML 형태로 저장되며 확장명은 ".UML" 이다. StarUML™에서 작성된 모든 모델, 뷰, 다이어그램들은 하나의 프로젝트 파일에 저장된다. 프로젝트를 여러 파일에 나누어 저장하려면 유닛을 사용할 수 있다. 프로젝트 파일에는 다음과 같은 정보들이 저장된다.

- 프로젝트가 사용하는 UML 프로파일들
- 프로젝트가 참조하는 유닛 파일들
- 프로젝트에 포함된 모든 모델 정보
- 프로젝트에 포함된 모든 다이어그램 및 뷰 정보

(4) 유닛

프로젝트는 기본적으로 하나의 파일에 저장되지만 프로젝트를 여러 명이 작업하거나 하는 등의 이유로 여러 개의 파일로 나누어서 다루어야 할 경우가 있다. 이러한 경우, 프로젝트를 여러 개의 유닛(Unit)으로 만들어서 다룰 수 있다. 유닛은 계층적으로 구성될 수 있어서 유닛의 하부에 여러 개의 서브 유닛을 가질 수도 있다. 유닛은 .UNT 파일에 저장되며, 프로젝트 파일(.UML) 혹은 다른 유닛 파일(.UNT)에서 참조된다.

(5) 유닛의 단위

패키지(Package), 서브시스템(Subsystem) 그리고 모델(Model) 요소만이 하나의 유닛(Unit)이 될 수 있다. 이러한 패키지류 요소들의 하위에 포함된 모든 요소들은 해당 유닛 파일(.PUX)내에 저장된다.

(6) 유닛 계층구조

프로젝트가 하위에 여러 개의 유닛을 관리할 수 있는 것 처럼, 유닛 자체도 하위에 여러 개의 유닛을 관리할 수 있다. 상위 유닛은 하위 유닛들에 대한 참조를 가지게 되고 이러한 관계에 의하여 유닛은 계층구조(Hierarchical Structure)를 이루게 된다.

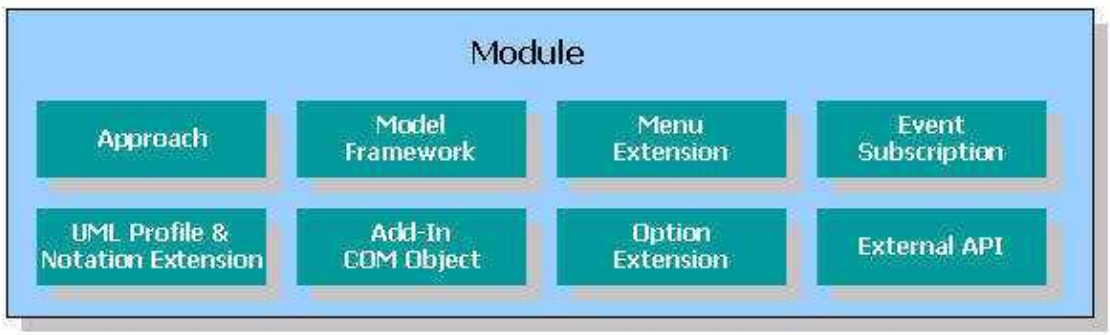
(7) 모델 조각

모델 조각은 프로젝트의 일부분을 별도의 파일로 저장한 것을 말한다. 모델 조각의 대상은 모델(Model), 서브시스템(Subsystem), 패키지(Package)에만 해당되며 이것은 ".MFG"라는 확장명의 파일로 저장된다. 이렇게 저장된 모델 조각 파일은 언제든지 어떤 프로젝트에서 쉽게 포함시킬 수 있으며 일단 포함된 모델 조각은 프로젝트의 일부로 완전히 포함되므로 유닛과는 다소 차이가 있다.

다. .모듈

(1) 모듈(Module)

모듈은 StarUML™을 확장하여 새로운 기능들과 특징들을 제공하기 위한 하나의 패키지이다. 모듈은 아래의 <그림>과 같은 여러 확장 요소들의 조합으로 만들 수 있다.
또한 목적에 따라 하나의 확장 요소만으로 독립적인 모듈을 구성하거나, 모듈 내에 같은 유형의 확장 요소들을 여러 개 만들 수도 있다.



StarUML™ 모듈을 사용하면 다음과 같은 것들이 가능해 진다.

- 메인 메뉴 혹은 팝업 메뉴의 확장
- 새로운 접근법의 추가
- 새로운 프레임워크의 추가
- 새로운 프로파일 추가
- 스테레오타입과 노테이션 확장을 통한 새로운 요소의 추가

- 새로운 기능에 대한 구현(COM Server를 통해 혹은 단순한 Script 파일을 통해서 가능)
- 다른 애플리케이션들과의 연동
- 그 외의 확장팩 지원 기능

(2) 접근법

소프트웨어를 개발하기 위한 방법론은 수도 없이 많으며, 각 회사나 조직마다 독자적인 방법론을 가지고 있거나 이미 존재하는 것을 조금씩 다른 방법으로 사용하기도 한다. 또한 개발할 소프트웨어에 대한 애플리케이션 영역(Application Domain)과 사용될 프로그래밍 언어와 플랫폼도 모두 다르다. 이러한 특성 때문에 소프트웨어 모델링 도구는 초기에 설정해야 할 사항들이 많아진다. StarUML™에서는 이러한 내용들을 한번에 설정할 수 있는 접근법(Approach)이라는 개념을 제공한다.

(3) 접근법의 구성

하나의 접근법은 다음의 항목들로 구성된다.

접근법 구성 항목	설명
프로젝트 구조(Project Structure)	시작할 프로젝트의 기본 구조를 설정한다. 패키지, 서브시스템, 모델 요소들로 기본적인 구조를 만들고 적절한 위치에 다이어그램을 기본으로 배치시킬 수 있다.
사용할 프로파일들(Import Profiles)	시작할 프로젝트에 기본으로 사용할 UML프로파일들을 자동으로 포함시킨다.
사용할 프레임워크들(Import Frameworks)	시작할 프로젝트에 기본으로 사용할 프레임워크들을 자동으로 로드하여 포함시킨다.
읽어들일 모델 조각들(Import Model fragments)	시작할 프로젝트에 기본으로 삽입할 모델 조각들을 자동으로 읽어들여 특정 요소 하위에 포함시킨다.

(4) 프레임워크

StarUML™에서의 프레임워크(Framework)는 클래스 라이브러리(Class Library)나 MFC, VCL, JFC 등과 같은 애플리케이션 프레임워크 등을 표현하는 소프트웨어 모델을 의미한다. 프레임워크가 프로젝트에 포함되어 사용되어지면, 사용자는 특정 클래스 라이브러리나 애플리케이션 프레임워크에 의존하는 소프트웨어를 모델링하는데 있어서 매우 편리한다.

(5) 프레임워크의 구성

프레임워크는 하나의 프레임워크 파일(.FRW)과 하나 이상의 유닛(.UNT)으로 구성된다.

구성요소	설명
프레임워크 파일(.FRW)	프레임워크 파일에는 어떤 유닛이 포함될지와 사용될 UML 프로파일에 대한 정보를 담고 있다.
유닛 파일(.UNT)	유닛 파일은 프레임워크에 대한 실질적인 모델 정보들을 담고 있다.

(6) UML 프로파일

UML(Unified Modeling Language)은 어떠한 생각이나 개념도 표현할 수 있을 만큼 일반적이다. 그러나, 그것은 또 다른 측면에서의 약점이 되기도 하는데, 특정 영역(Domain)의 개념들에 대해서는 정교하게 표현할 수 없다. 이러한 약점을 보완하기 위해서 UML 을 확장할 수 있는 메커니즘을 제공하고 있는데, 그것이 바로 UML 프로파일(Profile)이다. StarUML™은 UML 프로파일의 개념을 직접적으로 수용하여 UML을 쉽게 확장할 수 있도록 지원한다.

(7) UML 프로파일의 구성요소

UML 프로파일은 다음의 구성요소들로 이루어진다.

구성요소	설명
스테레오타입 (Stereotype)	스테레오타입은 특정 UML 요소에 붙여짐으로써 의미를 더 명확하게 하고 더불어 부차적으로 확장속성을 부여하여 더 정확한 모델링이 가능하도록 한다. 스테레오타입은 그래픽 노테이션을 표현하기 위하여 아이콘 파일을 지정할 수도 있지만, 확장 노테이션 정의 파일(.PNX)를 사용하여 노테이션 도식 방법을 직접 정의할 수도 있다. 확장 노테이션에 관한 자세한 내용은 개발자 매뉴얼을 참고하십시오.
확장속성 (TagDefinition)	확장속성은 UML 요소가 가지고 있는 프로퍼티만으로는 정확한 모델링이 불가능할 경우, 새로운 속성을 부여하여 부가적인 정보를 요소에 기록할 수 있도록 한다. StarUML™에서 확장속성은 특정 스테레오타입에 포함되거나 아니면 독립적으로 존재할 수도 있다.
데이터타입 (DataType)	프로파일이 기본적으로 포함하는 데이터타입들을 나타낸다.
다이어그램타입 (DiagramType)	다이어그램타입은 StarUML™에서 제안하고 있는 확장 요소로 다이어그램에 적용하여 새로운 다이어그램을 정의할 수 있도록 해 준다.
요소 프로토타입 (ElementPrototype)	요소 프로토타입은 StarUML™에서 제안하고 있는 확장 요소로 기존에 정의된 요소에 속성들을 미리 설정하여 요소 생성을 위한 하나의 견본을 정의할 수 있도록 한다. 이렇게 정의한 요소 프로토타입은 팔레트에 연결하여 요소를 직접 생성하거나 외부 API를 통해 요소를 생성할 수 있다.
모델 프로토타입 (ModelPrototype)	모델 프로토타입은 StarUML™에서 제안하고 있는 확장 요소로 요소 프로토타입과 유사하나 모델에 국한되어 적용한 것이다. 모델 프로토타입으로 정의한 요소는 모델 추가 메뉴에 표시된다.
팔레트(Palette)	팔레트는 StarUML™에서 제안하고 있는 확장 요소로 StarUML™에 기본적으로 포함되어 있는 팔레트 외에 추가적인 팔레트를 구성할 수 있도록 해 준다.

(8) UML 프로파일의 응용

UML 프로파일은 다음과 같은 용도 등으로 응용될 수 있다. 그리고 OMG(Object Management Group)에서는 특정 용도를 위한 UML 프로파일의 표준을 제정하고 있으므로 그것들을 참고하실 수 있다.

- 특정 프로그래밍 언어(C/C++, Java, C#, Python)용 프로파일
- 특정 개발 방법론(RUP, Catalysis, UML Components, ...)을 위한 프로파일
- 특정 도메인(EAI, CRM, SCM, ERP, ...)을 위한 프로파일

(9) 모듈의 추가

사용자가 개발하거나 써드 파티 벤더에 의해 배포되는 모듈을 추가적으로 설치하면 확장된 기능을 StarUML™ 내에서 사용할 수 있다. 시스템에 새로 추가되는 모듈을 설치하는데 별도의 복잡한 등록과정이 필요치는 않다. 모듈을 설치하려면 해당 모듈을 <install-dir>₩modules₩ 하위 디렉토리에 서브 디렉토리를 만든 다음 모듈을 구성하는 파일들을 복사하면 된다.

(10) StarUML™ 추가 제공 모듈

StarUML™은 StarUML™ 플랫폼 위에 동작하는 몇 가지의 모듈을 추가적으로 제공하고 있다.

- StarUML™은 UML 표준 프로파일과 몇 가지의 접근법, 시퀀스 및 협동 다이어그램간의 변환 등의 기능을 제공하는 기본 모듈(Standard Module)을 기본으로 제공한다.
- 템플릿 기반으로 각종 문서와 코드를 생성할 수 있게 해주는 Generator 모듈을 제공한다.
- Java 언어의 프로파일과 J2SE/J2EE 프레임워크, 코드 생성 및 역공학을 지원하는 Java 언어 모듈을 제공한다.

- C++ 언어의 프로파일과 MFC 프레임워크, 코드 생성 및 역공학을 지원하는 C++ 언어 모듈을 제공한다.
- C# 언어의 프로파일 .NET BCL 프레임워크, 코드 생성 및 역공학을 지원하는 C# 언어 모듈을 제공한다.
- 모델 상호 교환을 위한 XMI 가져오기와 내보내기를 지원하는 XMI 모듈을 제공한다.
- Rational Rose 파일을 읽어 오기 위한 Rose 모듈을 제공한다.
- 디자인 패턴을 지원하는 패턴 모듈을 제공한다.

3. 프로젝트 관리하기

이 장에서는 프로젝트를 관리하는 방법에 대해서 상세하게 설명한다. 새로운 프로젝트를 만드는 것에서부터 프로젝트의 일부를 유닛들로 구성하는 방법, 모델 조각을 만들고 불러오는 방법, 프레임워크를 불러오는 방법과 UML프로파일을 포함하거나 제거하는 방법등에 대해서 알아본다.

- 프로젝트 관리하기
- 유닛 관리하기
- 모델 조각으로 작업하기
- 프레임워크 불러오기
- UML 프로파일 사용하기

가. 프로젝트 관리하기

(1) 새 프로젝트 만들기

새로운 소프트웨어 프로젝트를 시작하려면 새 프로젝트를 만들어야 한다. 새 프로젝트는 아무런 초기화도 되어있지 않은 비어있는 프로젝트를 만들거나 접근법에 따라 서로 다르게 초기화된 프로젝트를 만들 수 있다.

- 새 프로젝트를 만드는 방법 1 - 새 프로젝트:

① [File]->[New Project] 메뉴를 선택한다.

② 사용자가 기본 접근법으로 선택한 접근법으로 초기화된 프로젝트가 바로 만들어 진다. 접근법의 종류에 따라 프로파일이 포함되고, 프레임워크가 로딩될 수 있다.

- 새 프로젝트를 만드는 방법 2 - 새 프로젝트 선택 대화상자

① [File]->[New Project By Approach] 메뉴를 선택한다.

② 새 프로젝트 선택 대화상자의 접근법 선택 페이지에 사용 가능한 접근법들이 목록에 나타난다. 이 중에서 한가지를 선택하고 [OK] 버튼을 누른다.

③ 선택한 접근법에 따라 프로젝트가 생성되고 초기화된다. 접근법의 종류에 따라 프로파일이 포함되고, 프레임워크가 로딩될 수 있다.

※ 참고

- 접근법의 목록은 사용자의 설치환경에 따라 다를 수 있다.
- 기본 접근법으로 사용할 접근법을 변경하려면 새 프로젝트 선택 대화상자를 열고 사용하려는 접근법을 선택한 후 “Set As Default Approach"을 체크하면 된다.

(2) 프로젝트 열기

파일에 저장된 프로젝트를 불러와서 작업을 하기 위해서는 프로젝트 파일을 열어야 한다. 프로젝트가 하나 이상의 유닛들을 포함하고 있다면 그것들도 프로젝트을 열 때 함께 로드된다.

- 프로젝트를 여는 방법:

① [File]->[Open] 메뉴를 선택한다.

② 프로젝트 열기 대화상자가 나오면 프로젝트 파일(.UML)을 하나 선택하고 [Open] 버튼을 누른다.

③ 선택된 프로젝트 파일이 열린다.

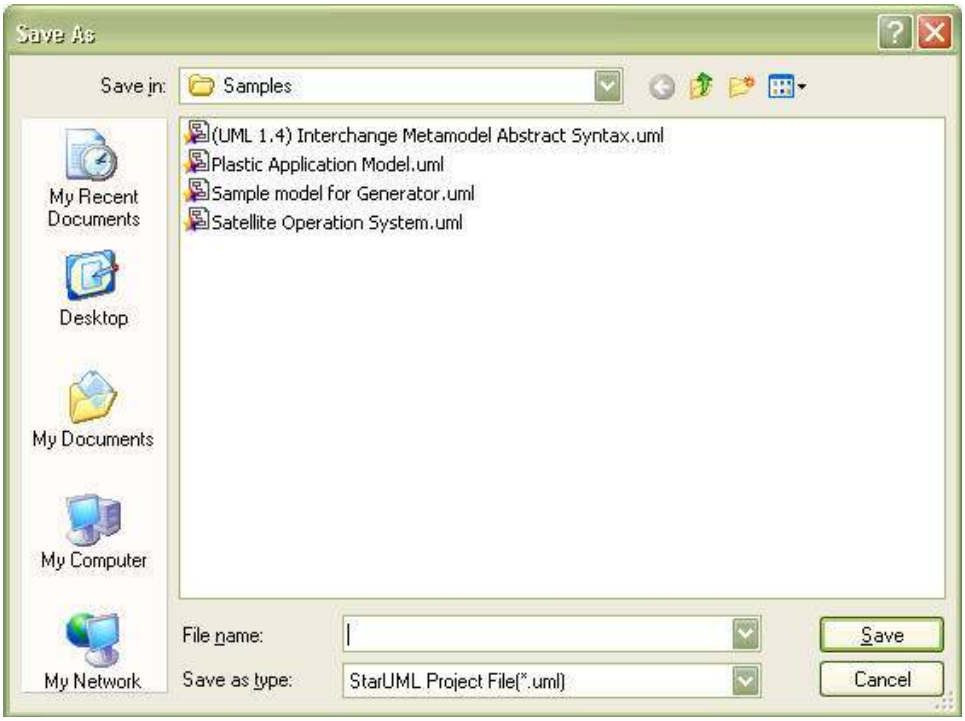
(3) 프로젝트 저장하기

프로젝트에서 사용자가 수행했던 작업의 내용들을 파일에 저장하기 위해서는 프로젝트를 파일에 안전하게 저장해야 한다. 프로젝트 파일에 바로 저장하거나 다른 이름으로 저장할 수 있으며, 포함된 들의 정보도 한번에 저장할 수 있다.

- 프로젝트를 저장하는 방법:

① [File]->[Save] 메뉴를 선택한다.

② 프로젝트에 파일 이름이 지정되어 있지 않는 경우에는 프로젝트 저장 대화상자가 나타난다. 여기서 파일 이름을 입력하고 [Save] 버튼을 누른다.



③ 프로젝트가 파일에 저장된다.

- 프로젝트를 저장하는 방법:

① [File]->[Save As] 메뉴를 선택한다.

② 프로젝트 저장 대화상자가 나타나면 새로운 파일 이름을 입력하고 [Save] 버튼을 누른다.

③ 프로젝트가 다른 이름의 파일에 저장된다.

(4) 프로젝트 닫기

프로젝트를 더 이상 편집하지 않을 때에는 프로젝트를 닫을 수 있다.

- 프로젝트를 닫는 방법:

① [File]->[Close] 메뉴를 선택한다.

② 프로젝트의 변경 내용이 저장되지 않았다면 사용자에게 저장할 것인지를 물어본다. 사용자는 의도에 따라 적절히 예, 아니오를 선택한다.

③ 프로젝트가 닫히고 더 이상 편집할 수 없는 상태가 된다.

(5) 모델, 서브시스템, 패키지로 요소 관리하기

소프트웨어 모델은 수 많은 요소들과 다이어그램들로 구성된다. 이러한 요소 및 다이어그램들을 효과적으로 그룹화하여 관리하는 것은 매우 중요한다. StarUML™에서는 총 3가지의 다양한 그룹화 요소들(모델, 서브시스템, 패키지)을 지원하여 사용자의 목적에 따라 사용하실 수 있다.

- StarUML™에서 제공하는 그룹화 요소

그룹화 요소	설명
모델(Model)	모델(Model)은 물리적인 시스템을 특정 목적(관점)에 의해 기술하기 위한 그룹화 요소이다. 예를 들어 시스템의 특정 관점(e.g. 분석관점, 설계관점, 사용자관점 등)에 따라 시스템을 표현할 수 있다.
서브시스템(Subsystem)	서브시스템(Subsystem)은 물리적인 시스템의 부분 혹은 전

	체를 명세화하기 위해 요소들을 그룹화하는 요소이다.
패키지(Package)	패키지(Package)는 모델 요소들을 논리적으로 그룹화하여 관리하기 위한 요소이다. 패키지는 요소들을 조직화하기 위한 어떠한 용도로 사용되어도 무방한 매우 일반적인 요소이다.

나. 유닛 관리하기

프로젝트의 내용을 하나의 파일에서 관리할 수도 있지만 여러 사람의 팀 작업을 위해서는 여러 개의 유닛으로 나누어서 작업하는 것이 편리한다. 이 장에서는 유닛을 만들고, 제거하는 등의 관리 방법을 설명한다.

- 유닛 만들기
- 유닛 병합하기
- 유닛 저장하기
- 유닛 제거하기

(1) 유닛 만들기

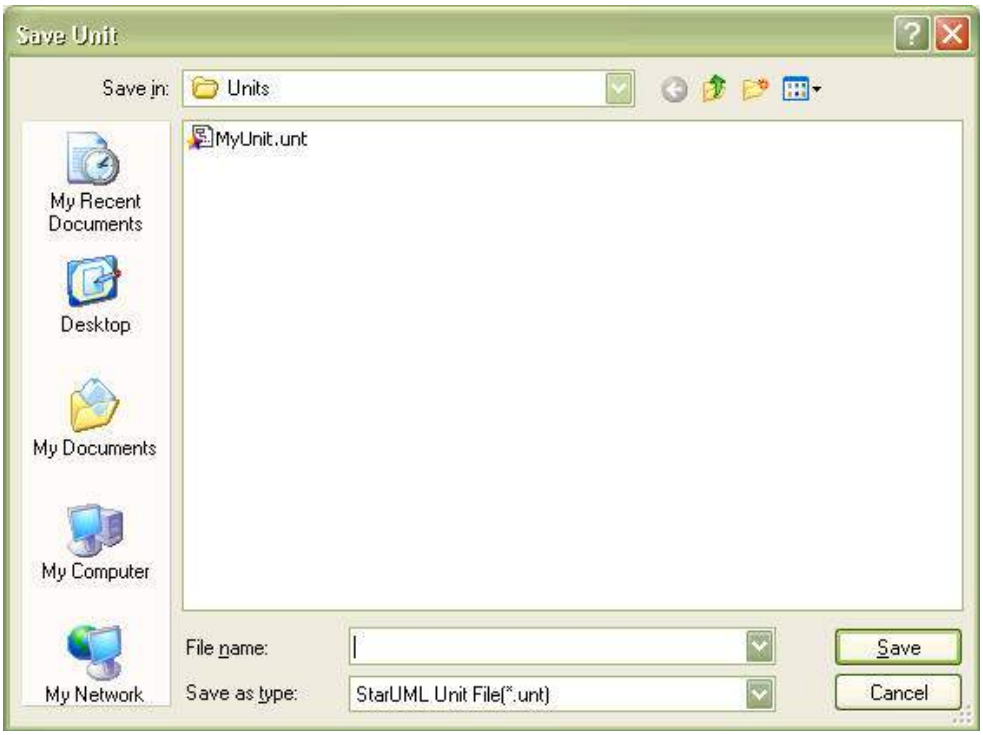
프로젝트 혹은 다른 유닛의 일부분을 별도의 파일로 저장하여 관리해야 할 경우가 있다. 예를 들어 하나의 프로젝트에 여러 명이 팀 작업을 해야 하는 경우, 프로젝트를 여러 개의 유닛으로 분할하여 CVS, Microsoft Visual SourceSafe와 같은 도구로 팀작업을 할 수 있다. 패키지(Package), 모델(Model) 또는 서브시스템(Subsystem) 요소만이 별도의 유닛으로 만들어질 수 있다.

- 새 유닛을 만드는 방법:

① 별도의 유닛으로 만들고자 하는 요소(패키지, 모델 또는 서브시스템)를 선택한다.

② 마우스 오른쪽 버튼을 눌러 [Unit]->[Control Unit] 메뉴를 선택한다.

③ 저장 대화상자가 나오면 유닛의 파일 이름을 입력하고 [Save] 버튼을 누른다.



④ 선택한 요소가 유닛으로 만들어진다.

(2) 유닛 병합하기

유닛에 포함된 요소들을 별도의 유닛 파일로 다루지 않고 상위 유닛이나 프로젝트에

통합하여 더 이상 유닛 파일을 관리하고 싶지 않은 경우 유닛을 병합할 수 있다.

- 유닛 병합하는 방법:

① 병합할 유닛에 해당하는 요소(패키지, 모델 또는 서브시스템)를 모델 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 누른 후 [Unit]->[Uncontrol Unit] 메뉴를 선택한다.

③ 유닛이 프로젝트 혹은 상위 유닛으로 병합된다.

※ 참고

- 유닛을 병합하더라도 유닛 파일(.UNT)은 삭제되지 않는다.
  유닛 파일의 삭제를 원하시면 직접 탐색기 등에서 삭제하셔야 한다.

(3) 유닛 저장하기

유닛의 정보들을 변경하였다면 유닛을 파일에 안전하게 저장하여야 한다.

유닛에 지정된 파일에 바로 저장하거나 다른 이름으로 저장할 수 있다.

- 유닛 저장하는 방법:

① 저장할 유닛을 모델 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 누른 후 [Unit]->[Save Unit] 메뉴를 선택한다.

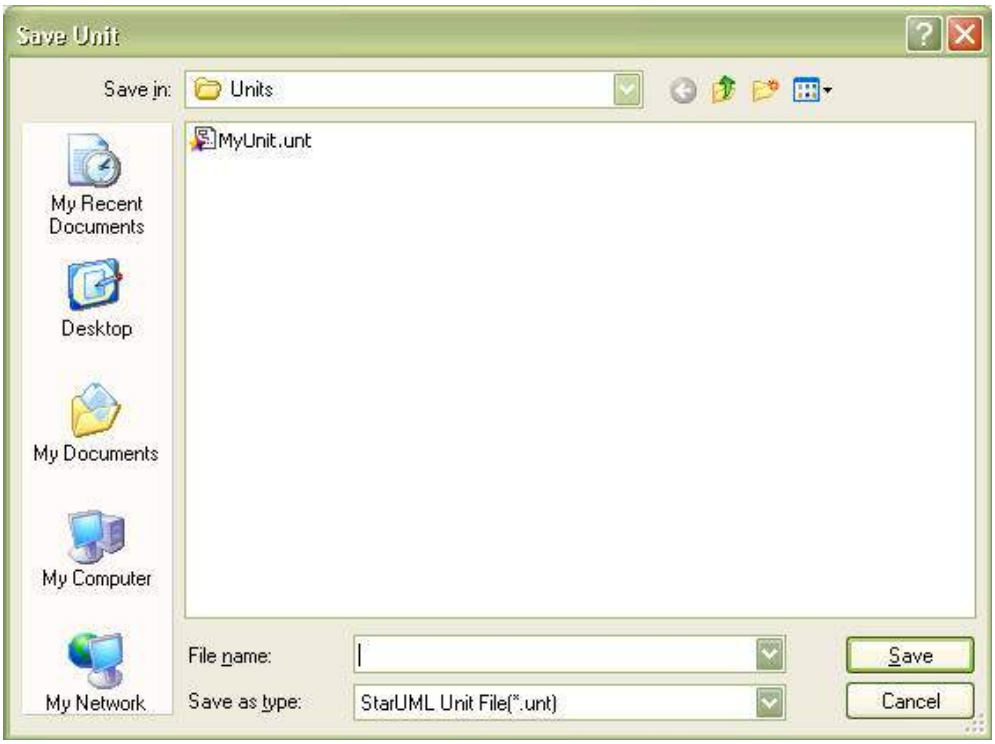
③ 유닛이 파일에 저장된다.

- 유닛 다른이름으로 저장하는 방법:

① 다른 이름으로 저장할 유닛을 모델 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 누른 후 [Unit]->[Save Unit As] 메뉴를 선택한다.

③ 유닛 저장 대화상자가 나오면 새로운 유닛 파일 이름을 입력하고 [Save] 버튼을 누른다.



④ 유닛이 새로운 파일에 저장된다.

(4) 유닛 제거하기

프로젝트에 포함된 유닛이 더 이상 필요하지 않은 경우에는 유닛을 제거할 수 있다.

유닛을 제거하면 유닛에 포함되는 모든 요소가 삭제되며 그 이후로 프로젝트를 불러올 때 유닛이 더 이상 로드되지 않다. 단지, 유닛의 내용을 프로젝트 혹은 상위 유닛으로 통합하고 해당 유닛을 더 이상 관리하고 싶지 않는 경우에는 유닛 삭제가 아닌 유닛 병합하기를 선택하십시오.

- 유닛을 제거하는 방법:

① 유닛을 제거하려면 우선 모델 탐색기에서 유닛에 해당하는 요소(패키지, 모델 또는 서브시스템)를 선택한다.

② 마우스 오른쪽 버튼을 누른 후 [Unit]->[Delete Unit] 메뉴를 선택한다.

③ 유닛을 제거할 것인지를 한번 더 확인하는 대화상자가 나온다. 여기서 [Yes]를 선택한다.

④ 유닛이 프로젝트로부터 완전히 제거된다.

※ 참고

- 유닛에 해당하는 요소를 선택한 뒤 [Edit]->[Delete From Model] 메뉴를 선택하여도 유닛을 제거한 것과 동일한 효과를 나타낼 수 있다.
- 유닛을 프로젝트로 부터 완전히 제거할 것인지 아니면 유닛을 프로젝트에 병합할 것인지를 확실하게 구분하여 결정하여야 한다.
- 유닛 파일(.UNT)은 유닛을 제거하더라도 삭제되지 않는다. 파일 제거를 원하시면 직접 파일 삭제를 하도록 한다.

다. 모델 조각으로 작업하기

프로젝트의 일부 요소들을 독립적인 파일로 저장하여 다루고 싶을 때 모델 조각(Model Fragment)을 사용할 수 있다.

- 모델 조각 만들기
- 모델 조각 불러오기

(1) 모델 조각 만들기

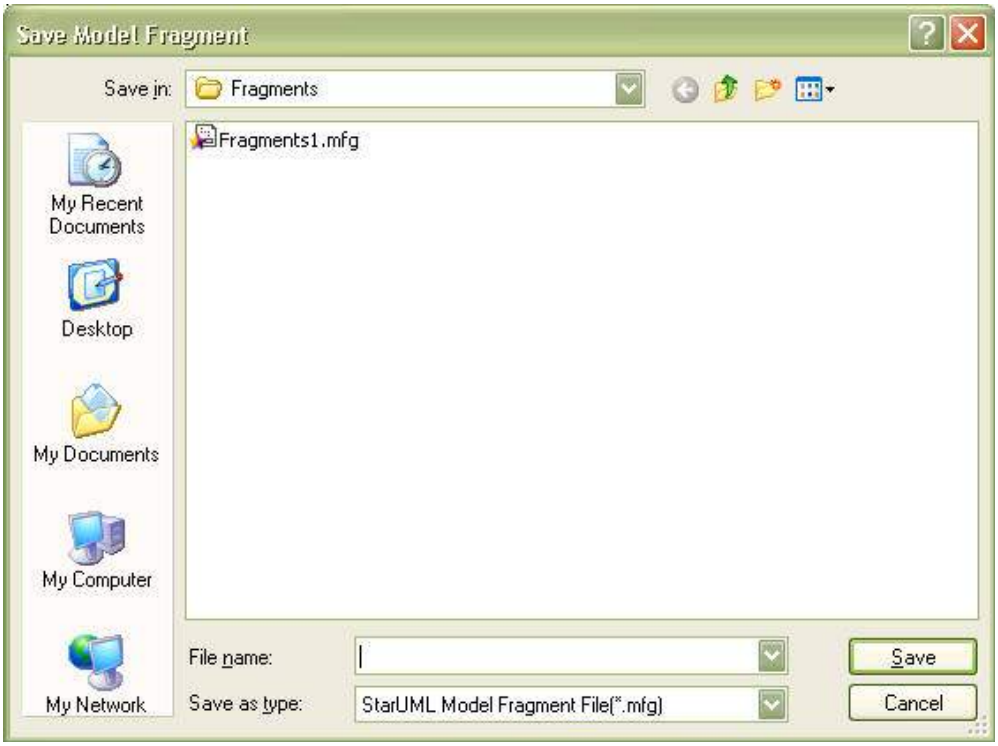
프로젝트의 일부 내용을 별도의 파일로 저장하여 다른 사람에 의해 사용되거나 차후에 다시 사용될 것을 고려한다면 모델 조각(Model Fragment)을 만들 수 있다. 모델 조각은 유닛과 달리 다른 파일로부터 참조되거나 다른 파일을 참조하지 않으로 그 자체로써 독립적인 단위이다. 모델 조각은 언제든지 프로젝트에 불러와서 포함될 수 있다.

- 모델 조각을 만드는 방법:

① 모델 조각(Model Fragment)으로 만들 패키지(Package), 서브시스템(Subsystem) 혹은 모델(Model)을 모델 탐색기에서 선택한다.

② [File]->[Export]->[Model Fragment] 메뉴를 선택한다.

③ 모델조각 저장 대화상자가 나오면 모델 조각 파일의 이름을 입력하고 [Save] 버튼을 누른다.



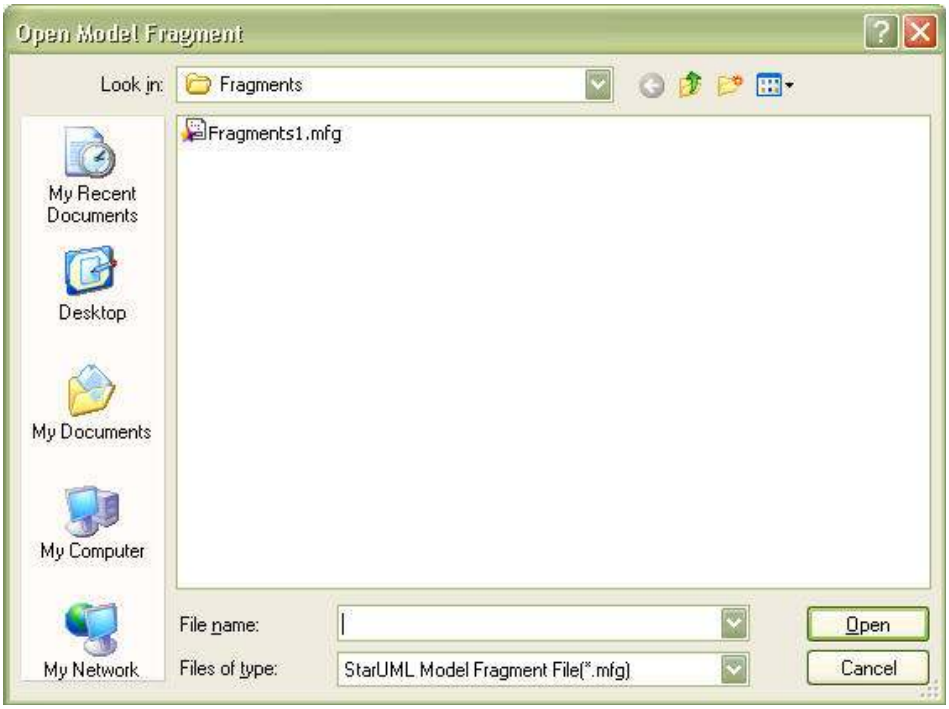
(2) 모델 조각 불러오기

별도의 모델 조각 파일(.MFG)에 저장되어 있는 요소들을 프로젝트 내부로 불러올 수 있다. 모델 조각을 프로젝트로 불러오면 모델 조각 파일에 저장된 요소들이 복제되어 포함되므로 모델 조각 파일을 직접 참조하지는 않다.

- 모델 조각을 불러오는 방법:

① [File]->[Import]->[Model Fragment] 메뉴를 선택한다.

② 모델 조각 열기 대화상자가 나오면 읽어올 모델 조각 파일(.MFG)을 선택하고 [Open] 버튼을 누른다.



③ 읽어올 모델 조각을 어떤 요소 하부에 둘 것인지를 결정하기 위해 요소 선택 대화상자가 나온다. 여기서 모델 조각을 포함할 요소(패키지, 모델, 서브시스템 또는 프로젝트)를 선택하고 [OK] 버튼을 누른다.

④ 모델 조각이 선택한 요소 하위에 추가된다.

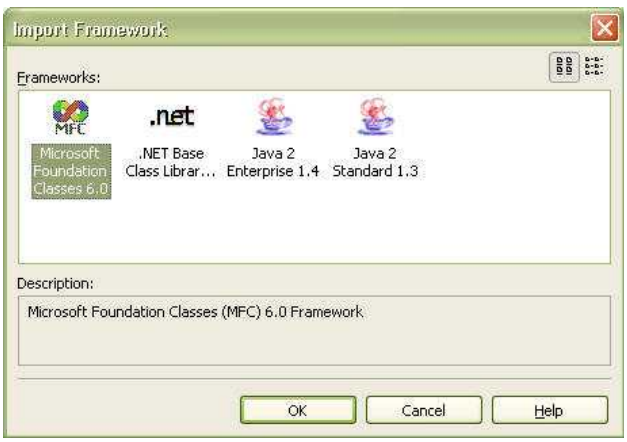
라. 프레임워크 불러오기

프로젝트에서 특정 프레임워크를 사용하기 위해서는 먼저 프레임워크를 불러와야 한다. 프레임워크를 불러오면 프레임워크에 포함되어 있는 모든 요소들을 사용할 수 있는데, 프레임워크를 구성하는 들은 일반적으로 읽기전용(readonly) 파일이므로 프레임워크 자체 요소들을 변경을 가할 수 없다.

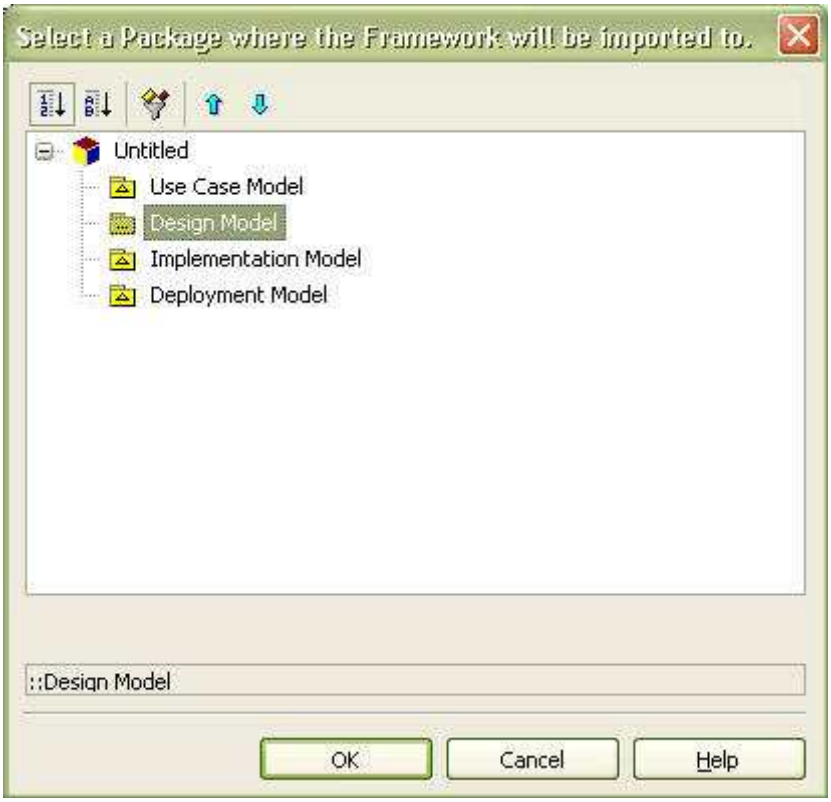
- 프레임워크를 불러오는 방법:

① [File]->[Import]->[Framework] 메뉴를 선택한다.

② 프레임워크 가져오기 대화상자가 나타나면 가져올 프레임워크를 목록에서 선택하고 [OK] 버튼을 누른다.



③ 읽어올 프레임워크를 어떤 요소 하부에 둘 것인지를 결정하기 위해 요소 선택 대화 상자가 나온다. 여기서 프레임워크를 포함할 요소(패키지, 모델, 서브시스템 또는 프로젝트)를 선택하고 [OK] 버튼을 누른다.



④ 프레임워크가 선택한 요소 하위에 추가된다.

※ 참고

- 프레임워크를 불러오더라도 프레임워크의 요소들은 프로젝트에 저장되지 않다. 프레임워크를 구성하는 유닛들이 프로젝트에서 참조되므로 프로젝트를 읽어올 때에는 프레임워크에 해당하는 유닛 파일들이 존재해야 한다.
- 불러온 프레임워크를 제거하려면 프레임워크에 해당되는 유닛들을 직접 제거하여야 한다.

마. UML 프로파일 사용하기

(1) UML 프로파일 포함하기

정의되어 있는 UML 프로파일들을 현재 프로젝트 내부로 포함시켜 사용할 수 있다. UML 프로파일이 프로젝트에 포함되면 프로파일에서 정의된 스테레오타입(Stereotype), 확장속성(TagDefinition) 및 데이터타입(DataType) 등을 프로젝트에서 사용할 수 있다.

- UML 프로파일을 포함하는 방법:

① [Model]->[Profiles] 메뉴를 선택한다.

② 프로파일 관리자 윈도우가 화면에 나타나면, 왼쪽의 사용 가능한 프로파일의 목록에서 포함시킬 프로파일을 선택하고 [Include] 버튼을 누른 뒤 [Close] 버튼을 누른다.

③ 선택한 프로파일이 현재 프로젝트에 포함된다.

(2) UML 프로파일 제거하기

현재 프로젝트에 포함되어 있는 UML 프로파일들을 제외시킬 수 있다. UML 프로파일이 프로젝트에서 제외되면 프로파일에 정의된 스테레오타입(Stereotype), 확장속성(TagDefinition) 및 데이터타입(DataType)을 프로젝트에서 사용할 수 없게 된다.

- UML 프로파일을 제거하는 방법:

① [Model]->[Profiles] 메뉴를 선택한다.

② 프로파일 관리자 윈도우가 화면에 나타나면, 오른쪽의 포함된 프로파일의 목록에서 제외시킬 프로파일을 선택하고 [Exclude] 버튼을 누른 뒤 [Close] 버튼을 누른다.

③ 선택한 프로파일이 현재 프로젝트에서 제외된다.

※ 참고

- 이미 UML 프로파일에 정의된 스테레오타입, 확장속성 등을 사용하고 있다면, 프로파일이 제거된 후 일부 프로파일에 의존적이던 내용들이 다르게 나타날 수 있으므로, 프로

파일의 제외는 주의를 기울여야 한다.

- 프로파일 관리자에 나타나는 프로파일 목록은 사용자의 설치환경에 따라 다를 수 있다.

4. StarUML 모델링하기

이 장에서는 본격적으로 StarUML™에서 다이어그램과 요소를 생성하고 편집하는 방법에 대해 설명한다. 또한 모델 탐색기를 사용하여 프로젝트 구조를 구성하는 방법을 알아본다.

- 다이어그램 및 요소 편집하기
- 모델 구조 구성하기

가. 다이어그램 및 요소 편집하기

(1) 새 다이어그램 생성하기

StarUML™은 모두 11가지의 UML 다이어그램을 지원한다. 사용자는 필요에 따라 원하는 다이어그램을 생성하여 작성할 수 있다.

- 새로운 다이어그램을 생성하는 방법:

① 모델 탐색기 혹은 다이어그램 영역에서 새로 생성된 다이어그램이 포함될 요소를 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Add Diagram] 메뉴를 선택한다. 그런 다음 생성하고자 하는 다이어그램의 종류를 선택하면 새 다이어그램이 생성된다.

- 생성할 수 있는 다이어그램 종류:

다이어그램 종류	설명
 Class Diagram	클래스 다이어그램(Class Diagram)은 클래스관련 요소들의 여러 가지 정적인 관계를 시각적으로 표현한 것이다. 클래스 다이어그램은 클래스(Class) 뿐만 아니라 인터페이스(Interface), 열거형(Enumeration), 패키지(Package) 및 여러 가지 관계들 뿐만 아니라 인스턴스(Instance)와 그것들의 연결(Link) 등도 포함할 수 있다.
 Use Case Diagram	유스케이스 다이어그램(Use Case Diagram)은 특정 시스템 혹은 개체내의 유스케이스(Use Case)들과 그 외부의 액터(Actor)들 간의 관계를 표현한 것이다. 유스케이스는 해당 시스템의 기능을 표현하며 그것들이 어떤 외부 액터들과 상호작용하는지를 나타낸다.
 Sequence Diagram	시퀀스 다이어그램(Sequence Diagram)은 인스턴스들이 어떻게 상호작용을 하는지를 묘사한다. 하나의 협동-인스턴스집합(CollaborationInstanceSet)에 포함된 인스턴스(Instance)들 상호간에 주고받는 자극(Stimulus)들의 집합인 상호작용-인스턴스집합(InteractionInstanceSet)을 직접적으로 표현한다. 시퀀스 역할 다이어그램(Sequence Role Diagram)은 역할(ClassifierRole) 중심의 관점을 반영한 반면, 시퀀스 다이어그램(Sequence Diagram)은 인스턴스(Instance) 중심의 관점을 반영한 것이다.
 Sequence Diagram (Role)	시퀀스 역할 다이어그램(Sequence Role Diagram)은 역할 개념들이 어떻게 상호작용을 하는지를 묘사한다. 하나의 협동(Collaboration)에 포함된 역할(ClassifierRole)들 상호간에 주고받는 메시지(Message)들의 집합인 상호작용(Interaction)을 직접적으로 표현한다. 시퀀스 다이어그램(Sequence Diagram)은 인스턴스(Instance) 중심의 관점을 반영한 반면, 시퀀스 역할 다이어그램(Sequence Role Diagram)은 역할(ClassifierRole) 중심의 관점을 반영한 것이다.
 Collaboration Diagram	협동 다이어그램(Collaboration Diagram)은 인스턴스들이 어떻게 협동하는지를 묘사한다. 하나의 협동-인스턴스집합(CollaborationInstanceSet)에 포함된 인스턴스(Instance)들의 협동 모델을 직접적으로 표현한다. 협동 역할 다이어그램(Collaboration Role Diagram)은 역할(ClassifierRole) 중심의 관점을 반영한 반면, 협동 다이어그램(Collaboration

	Diagram)은 인스턴스(Instance) 중심의 관점을 반영한 것이다.
Collaboration Diagram (Role)	협동 역할 다이어그램(Collaboration Role Diagram)은 역할 개념들이 어떻게 협동하는지를 묘사한다. 하나의 협동(Collaboration)에 포함된 역할(ClassifierRole)들의 협동 모델을 직접적으로 표현한다. 협동 다이어그램(Collaboration Diagram)은 인스턴스(Instance) 중심의 관점을 반영한 반면, 협동 역할 다이어그램(Collaboration Role Diagram)은 역할(ClassifierRole) 중심의 관점을 반영한 것이다.
Statechart Diagram	상태 다이어그램(Statechart Diagram)은 특정 개체의 동적인 행위를 상태(State)와 그것들간의 전이(Transition)를 통해 묘사한다. 일반적으로 클래스의 인스턴스에 대한 행위를 묘사하는데 사용되지만 그 밖의 요소들에 대해서도 얼마든지 사용될 수 있다.
Activity Diagram	액티비티 다이어그램(Activity Diagram)은 상태 다이어그램의 특별한 형태로써, 활동들의 수행 흐름을 묘사하는데 적합한다. 일반적으로 작업흐름(Workflow)을 표현하기 위해 많이 사용되며, 클래스, 패키지 혹은 연산 등의 개체에 대해 주로 사용된다.
Component Diagram	컴포넌트 다이어그램(Component Diagram)은 소프트웨어 컴포넌트 사이의 의존관계를 묘사한다. 소프트웨어 컴포넌트를 구성하는 요소들과 그것들을 구현하는 요소들도 모두 표현될 수 있다.
Deployment Diagram	디플로이먼트 다이어그램(Deployment Diagram)은 물리적인 컴퓨터 및 장비 등의 하드웨어 요소들과 그것에 들이 배치되는 소프트웨어 컴포넌트, 프로세스 및 객체들의 형상을 묘사한다.
Composite Structure Diagram	복합구조 다이어그램(Composite Structure Diagram)은 분류자(Classifier)의 내부 구조를 표현하는 다이어그램이다. 여기에는 Classifier가 시스템의 다른 부분들과의 상호작용하는 지점 등을 포함한다.

(2) 다이어그램에 요소 생성하기

다이어그램에 새로운 요소를 생성하고자 한다면 먼저 다이어그램이 열려 있어야 한다. 팔레트에는 다이어그램의 종류에 따라 생성될 수 있는 요소들이 열거되어 있으며, 열거된 요소들의 목록은 다이어그램의 종류에 따라 달라진다.

- 팔레트로부터 요소를 생성하는 방법:

① 팔레트에서 생성할 요소 선택한다.

② 다이어그램 영역에서 생성할 위치에 마우스를 클릭한다.(드래그하여 영역을 그리면 해당 크기로 요소가 생성되고, 특별히 두 요소를 연결하는 요소를 생성하고자 하는 경우에는 두 요소를 정확하게 연결하여야 한다.)

- 요소를 연속적으로 여러 개 생성하는 방법:

① 팔레트에서 생성할 요소를 선택한다.

② 팔레트의 팝업 메뉴에서 [Lock] 메뉴를 선택하거나, 생성할 요소를 한번 더 클릭한다.

③ 요소를 연속적으로 여러 개를 생성한다.

④ 더 이상 생성하지 않는 경우에는 팔레트에서 [그림-Select] 항목을 클릭한다.

※ 참고

- 팔레트에서 요소를 다이어그램에 생성하는 것은 하나의 모델 요소(Model element)와 그것을 표현한 하나의 뷰 요소(View element)를 생성하는 것이다.

(3) 다이어그램에 뷰 요소 생성하기

다이어그램에서 팔레트로부터 새로운 요소를 생성하는 것 외에, 이미 존재하는 모델

요소(Model element)에 대한 뷰 요소(View element)만을 생성할 수 있다.

- 새로운 뷰 요소를 생성하는 방법(드래그-앤드-드롭 방식):

① 모델 탐색기에서 생성할 뷰에 해당하는 모델을 먼저 선택한다.

② 해당 모델 요소를 끌어다가 다이어그램 영역에 놓으면 뷰 요소가 생성된다. (이 경우에는 연관된 모든 요소들과의 연결들도 자동으로 나타난다.)

(4) 다이어그램에서 요소 편집하기

다이어그램 영역에서 요소를 직접 편집할 수 있다.

- 요소를 편집하는 방법:

① 다이어그램에서 편집하고자 하는 뷰 요소를 마우스로 더블클릭한다.

② 퀵 다이얼로그가 화면에 나타나면 요소의 이름(Name), 가시성(Visibility) 등을 편집하거나 버튼을 눌러 하위 요소를 생성할 수도 있다.

③ [Enter] 키를 치거나 다이어그램의 다른 영역을 클릭하면 편집 내용이 적용된다.

(5) 뷰 크기 조정 및 이동

다이어그램 영역에 있는 뷰들의 크기와 위치를 조정할 수 있다. 또한 키보드의 특수키와 커서키의 조합으로 뷰들의 크기와 위치를 미세하게 조정할 수도 있다.

- 뷰의 크기를 조정하는 방법:

① 다이어그램에서 크기를 조정하고자 하는 뷰를 마우스로 클릭하여 선택한다.

② 뷰 선택 후 나타나는 선택 표시 위의 포인트 중 원하는 방향의 포인트를 드래그하여 크기를 조정한다.

- 키보드를 이용하여 뷰의 크기를 조정하는 방법:

① 다이어그램에서 크기를 조정하고자 하는 뷰를 마우스로 클릭하여 선택한다.

② Shift+ 커서키를 사용하여 원하는 방향으로 뷰의 크기를 조정한다. Shit+ 커서키는 현재 설정된 그리드의 단위만큼 크기가 변경되며, Shift+ Alt+ 커서키로는 미세하게 뷰의 크기를 조정할 수 있다.

- 뷰를 이동시키는 방법:

① 다이어그램에서 이동시키고자 하는 뷰를 마우스로 클릭하여 선택한다. 이동시킬 뷰가 여러 개 이면 Ctrl+ 클릭 으로 여러 뷰를 선택하거나 뷰들을 포함하도록 영역을 드래그하여 선택한다.

② 선택된 영역 내에 마우스를 클릭한 후 원하는 방향으로 드래그하여 뷰들을 이동시킨다.

- 키보드를 이용하여 뷰를 이동시키는 방법:

① 다이어그램에서 이동시키고자 하는 뷰를 마우스로 클릭하여 선택한다. 이동시킬 뷰가 여러 개 이면 Ctrl+ 클릭 으로 여러 뷰를 선택하거나 뷰들을 포함하도록 영역을 드래그하여 선택한다.

② Ctrl+ 커서키를 사용하여 원하는 방향으로 뷰들을 이동시킨다. Ctrl+ 커서키는 현재 설정된 그리드의 단위만큼 위치가 이동되며, Ctrl+ Alt+ 커서키로는 미세하게 뷰의 위치를 변경할 수 있다.

(6) 단축생성 구문을 사용하여 요소 생성하기

퀵 다이얼로그의 단축생성 구문기능을 이용하면 별도의 마우스 조작 없이 키보드만으로 요소를 생성할 수 있다.

- 단축생성 구문을 사용하여 요소를 생성하는 방법:

① 다이어그램에서 뷰를 선택한다.

② [Enter] 키를 눌러 퀵다이얼로그를 구동한다.
③ 퀵다이얼로그 입력란에서 생성을 원하는 요소에 맞게 구문을 입력한다.
- 단축생성 구문
단축 생성 구문은 간단한 텍스트 입력만으로 타겟 모델을 생성하거나 타겟 모델과 관계를 생성할 수 있다. 단축 생성 구문의 일반적인 규칙은 다음과 같다. 생성하려는 관계의 표기법과 관계를 맺게 될 타겟 모델의 이름들을 기술한다. 만약 타겟 모델의 이름이 없으면 관계와 적합한 적절한 모델 요소를 새로 생성하고 관계를 생성한다. 각 다이어그램에서 사용할 수 있는 단축 생성 구문의 relationship-notation은 다음과 같다.

다이어그램 종류	표기법	현재 요소	설명
Class Diagram Component Diagram Deployment Diagram Composite Structure Diagram	<=	Classifier	현재 요소로부터 타겟 요소로 specialization 관계를 맺는다.
	=>	Classifier	현재 요소로부터 타겟 요소로 generalization 관계를 맺는다.
	--	Classifier	현재 요소로부터 타겟 요소로 association 관계를 맺는다.
	<-	Classifier	타겟 요소로부터 현재 요소로 navigable association 관계를 맺는다.
	->	Classifier	현재 요소로부터 타겟 요소로 navigable association 관계를 맺는다.
	<>-	Classifier	현재 요소는 타겟 요소를 aggregate하는 관계를 맺는다.
	-<>	Classifier	타겟 요소가 현재 요소를 aggregate하는 관계를 맺는다.
	<*>-	Classifier	현재 요소는 타겟 요소를 compose하는 관계를 맺는다.
	-<*>	Classifier	타겟 요소가 현재 요소를 compose하는 관계를 맺는다.
	<--	Classifier	타겟 요소가 현재 요소에 대해서 dependency 관계를 맺는다.
	-->	Classifier	현재 요소가 타겟 요소에 대해서 dependency 관계를 맺는다.
	)-	Classifier	타겟 요소가 현재 요소에 대해서 requirement 관계를 맺는다.
	-(	Classifier	현재 요소가 타겟 요소에 대해서 requirement 관계를 맺는다.
	@-	Classifier	타겟 요소가 현재 요소를 realization하는 관계를 맺는다.
	-@	Classifier	현재 요소가 타겟 요소를 realization하는 관계를 맺는다.
Usecase Diagram	()-	UseCase	현재 요소와 연결된 타겟 모델(Actor)과 communication 연결을 한다.
	-()	Actor	현재 요소와 연결된 타겟 모델(UseCase)과 communication 관계를 맺는다.
	<i-	UseCase	타겟 요소로부터 현재 요소로 include 관계를 맺는다.
	-i>	UseCase	현재 요소로부터 타겟 요소로 include 관계를 맺는다.
	<e-	UseCase	타겟 요소로부터 현재 요소로 extend 관계를 맺는다.
	-e>	UseCase	현재 요소로부터 타겟 요소로 extend 관계를 맺는다.
Sequence Diagram Seqeunce	<-	Object, ClassifierRole	현재 요소로부터 타겟 요소로 stimulus를 생성한다.

Diagram(Role)	->	Object, ClassifierRole	타겟 요소로부터 현재 요소로 stimulus를 생성한다.
	<->	Object, ClassifierRole	타겟 요소로부터 현재 요소로 return을 가지는 stimulus를 생성한다.
	<-	Stimulus, Message	현재 Stimulus의 Sub-Stimulus(타겟 요소로부터 들어오는)를 생성한다.
	->	Stimulus, Message	현재 Stimulus의 Sub-Stimulus(타겟 요소로 나가는)를 생성한다.
	<->	Stimulus, Message	현재 Stimulus의 Sub-Stimulus(타겟 요소로부터 나가면서 리턴을 가지는)를 생성한다.
	<~	Stimulus, Message	현재 Stimulus 앞에 존재하는 Stimulus(타겟 요소로부터 들어오는)를 생성한다.
	~>	Stimulus, Message	현재 Stimulus 앞에 존재하는 Stimulus(타겟 요소로 나가는)를 생성한다.
	<_	Stimulus, Message	현재 Stimulus 뒤에 존재하는 Stimulus(타겟 요소로부터 들어오는)를 생성한다.
	_>	Stimulus, Message	현재 Stimulus 뒤에 존재하는 Stimulus(타겟 요소로 나가는)를 생성한다.
Collaboration Diagram Collaboration Diagram(Role)	<-	Object, ClassifierRole	현재 요소로부터 타겟 요소로 stimulus를 생성한다.
	->	Object, ClassifierRole	타겟 요소로부터 현재 요소로 stimulus를 생성한다.
	<->	Object, ClassifierRole	타겟 요소로부터 현재 요소로 return을 가지는 stimulus를 생성한다.
Statechart Diagram/ Activity Diagram	<-	State, ActionState	타겟 요소에서 현재 요소로 나가는 transition을 생성한다.
	->	State, ActionState	현재 요소에서 타겟 요소로 나가는 transition을 생성한다.
	-*	State, ActionState	타겟 요소(Initial State)로부터 현재 요소로 들어오는 transition을 생성한다.
	-@	State, ActionState	현재 요소에서 타겟 요소(Final State)로 나가는 transition을 생성한다.
	<-<>	State, ActionState	타겟 요소(Decision)에서 현재 요소로 들어오는 transition을 생성한다.
	-><>	State, ActionState	현재 요소에서 타겟요소(Decision)로 나가는 transition을 생성한다.
	-(H) -(h)	State, ActionState	현재 요소에서 타겟요소(History)로 나가는 transition을 생성한다.
	-(H*) -(h*)	State, ActionState	현재 요소에서 타겟요소(Deep History)로 나가는 transition을 생성한다.
	<-|	State, ActionState	타겟 요소에서 현재 요소로 join해서 들어오는 transition을 생성한다.
	|->	State, ActionState	현재 요소에서 타겟 요소로 fork해서 나가는 transition을 생성한다.

(7) 복사 및 붙여넣기

요소들을 복사 혹은 잘라내어 다른 곳으로 붙여넣기를 할 때에는 모델 요소와 뷰 요 소를 구분하여야 한다. 모델 요소를 복사하면 모델 요소 하위에 붙여넣기를 해야 하며 이 경우에는 복사된 모델 요소의 모든 하위 요소들까지 복사된다. 뷰 요소를 복사하여 동일한 다이어그램 혹은 다른 다이어그램으로 복사할 수 있으며 이 경우에는 다이어그램에만 붙여넣기를 할 수 있고 모델 요소 하위로는 뷰 요소들을 붙여넣기 할 수 없다.

이 경우에도 뷰 요소의 종류 및 다이어그램의 종류에 따라 복사/붙여넣기가 제한될 수 있다.

- 모델 요소를 복사하여 붙여넣는 방법:

① 복사하고자 하는 모델 요소를 모델 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 누른 후 [Copy] 메뉴를 선택하면 모델 요소가 클립보드에 복사된다.

③ 복사된 모델 요소를 붙여넣을 요소를 모델 탐색기에서 다시 선택한다.

④ 마우스 오른쪽 버튼을 누른 후 [Paste] 메뉴를 선택하면 복사된 모델 요소가 클립보드로부터 생성되어 선택된 요소 하위에 포함된다.

복사된 모델 요소는 그것을 실제 포함할 수 있는 요소 하위에만 붙여넣기를 할 수 있다.

- 다이어그램의 뷰 요소들을 복사하여 붙여넣는 방법:

① 다이어그램 영역에서 복사하고자 하는 뷰 요소들을 선택한다. (마우스로 클릭하거나 영역을 드래그 하여 여러개를 선택할 수 있다. [Shift] 키를 누르고 뷰 요소를 클릭하면 추가적으로 선택된다.)

② 마우스 오른쪽 버튼을 누른 후 [Copy] 메뉴를 선택하면 뷰 요소들이 클립보드에 복사된다.

③ 복사된 뷰 요소들을 붙여 넣을 다이어그램을 열어 활성화시킨다. (모델 탐색기 혹은 다이어그램 탐색기에서 더블 클릭을 하거나 다이어그램 탭 중에서 선택한다.)

④ 마우스 오른쪽 버튼을 누른 후 [Paste] 메뉴를 선택하면 복사된 뷰 요소들이 활성화된 다이어그램으로 복사되어 포함된다.

- 다이어그램별 복사/붙여넣기 특징

다이어그램 종류	복사/붙여넣기 특징
Class Diagram	Class, UseCase, Component, Deployment, CompositeStructure 다이어그램들 간에는 복사/붙여넣기가 자유로움
UseCase Diagram	Class, UseCase, Component, Deployment, CompositeStructure 다이어그램들 간에는 복사/붙여넣기가 자유로움
Sequence Diagrams	복사/붙여넣기를 허용하지 않음
Collaboration Diagrams	복사/붙여넣기를 허용하지 않음
Statechart Diagram	동일한 StateMachine 내의 다이어그램 사이에서만 복사/붙여넣기 허용
Activity Diagram	동일한 ActivityGraph 내의 다이어그램 사이에서만 복사/붙여넣기 허용
Component Diagram	Class, UseCase, Component, Deployment, CompositeStructure 다이어그램들 간에는 복사/붙여넣기가 자유로움
Deployment Diagram	Class, UseCase, Component, Deployment, CompositeStructure 다이어그램들 간에는 복사/붙여넣기가 자유로움
CompositeStructure Diagram	Class, UseCase, Component, Deployment, CompositeStructure 다이어그램들 간에는 복사/붙여넣기가 자유로움

(8) 프로퍼티 설정하기

모델 요소는 여러 가지의 프로퍼티(Property) 정보를 가지고 있다. 사용자는 이러한 프로퍼티의 값을 설정함으로써 모델을 원하는데로 수정한다. 이러한 프로퍼티에는 다음과 같은 종류가 있다.

- 프로퍼티 종류:

프로퍼티 종류	설명
이름(Name)	모델 요소의 이름을 나타냄
스테레오타입(Stereotype)	모델 요소에 대한 스테레오타입을 나타냄
타입식(TypeExpression)	특정 타입을 가리키기 위한 식(Expression)을 나타냄
문자열(String)	문자열을 나타냄
논리형(Boolean)	True 혹은 False를 나타냄

열거형(Enumeration)	몇개의 리터럴 중 한가지를 선택
레퍼런스(Reference)	특정 요소를 가리킴
컬렉션(Collection)	여러 개의 요소를 나타냄 (컬렉션편집기를 통해 편집)

- 이름(Name) 프로퍼티 설정하기

이름은 프로퍼티 편집기의 "Name"항목에 기록한다. 단, 이름에는 모델의 경로를 구분하기 위하여 사용되는 :: 특수문자는 사용될 수 없다. 그리고 이름은 자신이 속한 이름 공간에서 유일해야 한다. 예를 들어 패키지(Package) 내에서 클래스(Class)들의 이름은 유일해야 한다. 이름이 다른 요소와 충돌하는 경우에는 사용자에게 경고 메시지를 보여준다.

- 스테레오타입(Stereotype) 프로퍼티 설정하기

스테레오타입은 프로퍼티 편집기에서 "Stereotype"항목에 기록한다. 이 때 입력하는 내용은 스테레오타입의 이름이며 이것은 UML 프로파일 내에 정의된 스테레오타입 이거나 정의되지 않은 단순한 이름일 수 있다. 스테레오타입 프로퍼티를 설정하기 위한 방법은 다음과 같은 것들이 있다.

① 정의된 스테레오타입 입력 : 현재 프로젝트에 포함되어 있는 프로파일들에 의해 정의된 스테레오타입의 이름을 입력한다. 해당 스테레오타입을 직접 가리키게 된다.

② 정의되지 않은 스테레오타입 입력 : 현재 프로젝트에 포함되어 있는 프로파일들에서 정의되지 않은 스테레오타입의 이름을 입력한다. 단순한 문자열 정보에 지나지 않다.

③ 스테레오타입 대화상자에서 선택 : 스테레오타입 대화상자를 열면 정의된 스테레오타입들을 바로 선택할 수 있다.

- 타입식 프로퍼티는 속성(Attribute), 파라미터(Parameter) 등에 포함되어 있다. 타입식을 편집하기 위해서는 프로퍼티 편집기의 "Type"부분에 타입식을 기록한다. 타입식 프로퍼티를 설정하는 방법은 다음과 같은 것들이 있다.

① 정의된 타입 이름 입력 : 현재 프로젝트에 포함되어 있는 분류자 요소들(클래스, 인터페이스, 시그널, 예외, 컴포넌트, 노드, 서브시스템 등)의 이름을 입력한다. 직접 해당 요소를 가리키게 된다.

② 정의된 타입 경로명 입력 : 현재 프로젝트에 포함되어 있는 분류자 요소들의 경로명을 직접 입력한다. (예: "::Logical View::Package1::Class1")

③ 정의되지 않은 타입 이름입력 : 현재 프로젝트에 포함된 분류자와 관계 없는 이름을 입력한다. 단순한 문자열 정보에 지나지 않다.

④ 요소 선택 대화상자에서 선택 : 요소 선택 대화상자를 열면 정의된 타입 혹은 프로파일에 정의된 데이터타입을 바로 선택할 수 있다.

(9) 모델 요소 문서화하기

모델 요소에 대한 상세한 설명을 기록할 수 있다.

- 모델 요소를 문서화하는 방법:

① 상세 설명을 기록하고자 하는 요소를 모델 탐색기에서 선택하거나 다이어그램 영역에서 선택한다.

② 메인 윈도우의 인스펙터 영역에서 [Documentation] 탭을 선택한다.

③ 상세한 내용을 편집 영역에 기록한다.

파일 혹은 URL 첨부하기 요소에 관련된 파일이나 웹페이지의 주소(URL) 등을 여러 개 첨부해 둘 수 있다. 첨부된 파일이나 웹페이지는 간편하게 연관된 애플리케이션으로 열거나 웹브라우저를 통해 URL로 이동할 수 있다.

- 파일 혹은 URL을 첨부하는 방법:

① 요소를 모델 탐색기 혹은 다이어그램 영역에서 선택한다.

② 메인 윈도우의 인스펙터 영역에서 [Attachments] 탭을 선택한다.

③ 마우스 오른쪽 버튼을 누른 뒤 [Add] 메뉴를 선택하거나 도구모음에서 [Add] 버튼을 누른다.

④ 첨부 대화상자가 나타나면 첨부할 파일의 경로명을 포함한 이름이나 웹페이지의 주소를 입력(오른쪽에 있는 탐색 버튼을 눌러 직접 선택할 수도 있음)하고 [OK] 버튼을 누른다.

- 첨부된 항목을 제거하는 방법:

① 요소를 모델 탐색기 혹은 다이어그램 영역에서 선택한다.

② 메인 윈도우의 인스펙터 영역에서 [Attachments] 탭을 선택한다.

③ 목록에서 삭제할 첨부 항목을 선택한 다음, 마우스 오른쪽 버튼을 누르고 [Delete] 메뉴를 선택하거나 도구모음에서 버튼을 누른다.

(10) 제약사항 기록하기

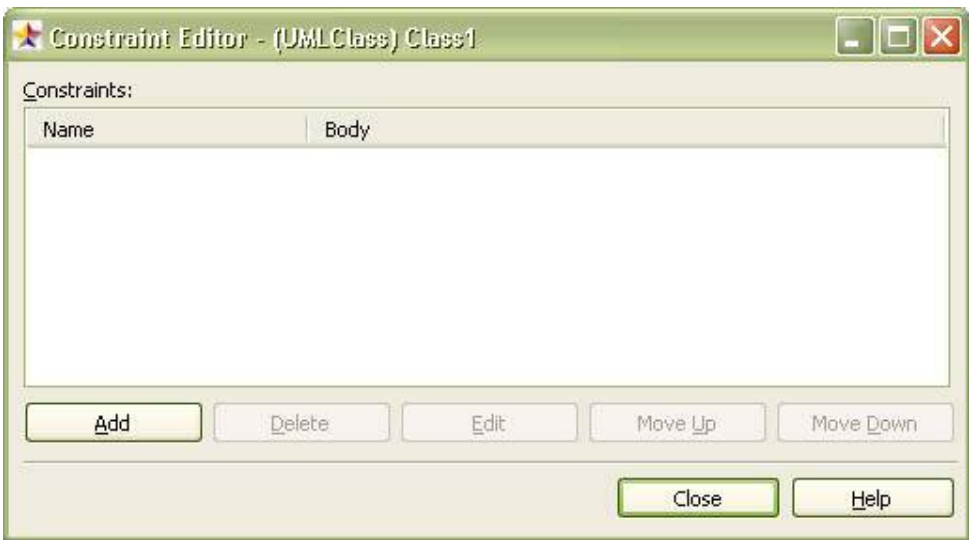
요소에 여러 개의 제약사항(Constraint)를 기록할 수 있다. 제약사항이란 요소에 적용되는 규칙으로써 인간의 언어로 쉽게 설명하거나, UML에 정의된 OCL(Object Constraint Language)의 구문에 맞추어 작성할 수도 있다.

- 제약사항을 추가하는 방법:

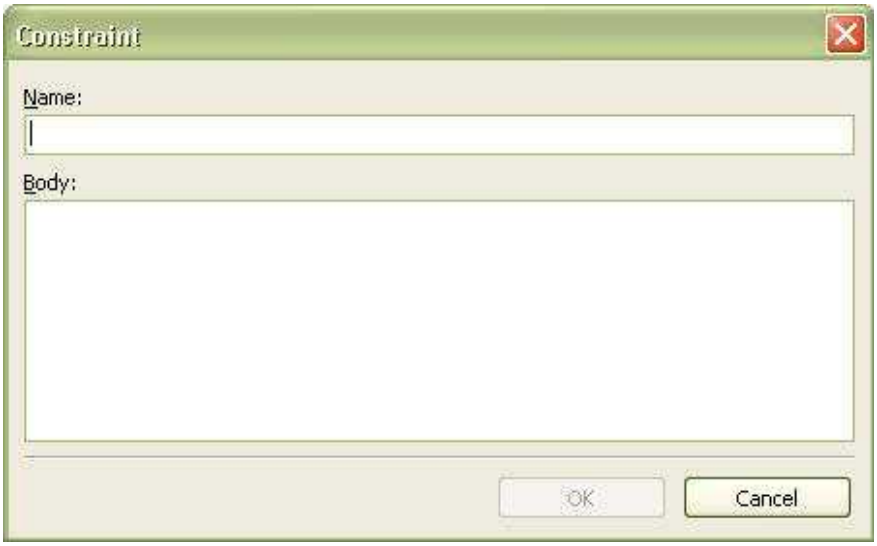
① 제약사항을 추가하고자 하는 요소를 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Constraints] 메뉴를 선택한다.

③ 제약사항 편집기가 나타나면 [Add] 버튼을 누른다.



④ 제약사항 대화상자가 나타나면 이름과 내용을 입력한 다음 [OK] 버튼을 누른다.



- 제약사항을 삭제하는 방법:

① 제약사항을 삭제하고자 하는 요소를 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Constraints] 메뉴를 선택한다.

③ 제약사항 편집기가 나타나면 삭제할 제약사항을 목록에서 선택한 다음 [Delete] 버튼을 누른다.

- 제약사항의 내용을 변경하는 방법:

① 제약사항의 내용을 변경하고자 하는 요소를 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Constraints] 메뉴를 선택한다.

③ 제약사항 편집기가 나타나면 변경할 제약사항을 목록에서 선택한 다음 [Edit] 버튼을 누른다.

④ 제약사항 대화상자가 나타나면 이름과 내용을 수정한다. 그런 다음 [OK] 버튼을 누른다.

(11) 확장속성 편집하기

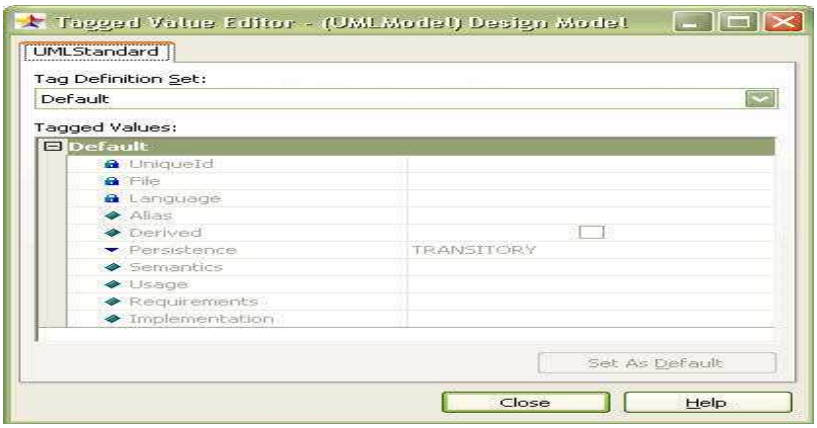
요소의 기본적인 프로퍼티 외에 UML 프로파일에 의해서 부가적으로 확장된 속성(TaggedValue)들을 편집할 수 있다.

- 확장속성을 편집하는 방법:

① 확장속성을 편집할 요소를 모델 탐색기 혹은 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Tagged Values] 메뉴를 선택한다.

③ 확장속성 편집기가 나타나면 편집할 확장속성이 포함된 프로파일에 해당하는 탭을 선택한다.



④ 그리고 확장속성을 포함하는 집합을 [Tag Definition Set] 콤보상자에서 선택하고 [Tagged Values] 목록에서 속성을 선택하여 값을 편집한다.

- 편집한 확장속성값을 기본값으로 되돌리는 방법:

① 확장속성을 가진 요소를 모델 탐색기 혹은 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Tagged Values] 메뉴를 선택한다.

③ 확장속성 편집기가 나타나면 확장속성이 포함된 프로파일에 해당하는 탭을 선택한다.

④ 그리고 확장속성을 포함하는 집합을 [Tag Definition Set] 콤보상자에서 선택하고 [Tagged Values] 목록에서 속성을 선택한 다음 [Set As Default] 버튼을 누른다.

(12) 뷰 요소 삭제하기

뷰 요소를 삭제한다는 것은 모델 요소는 삭제하지 않고 순수하게 그것을 화면상으로 나타내고 있는 뷰 요소만을 삭제한다는 것을 의미한다.

- 뷰 요소를 삭제하는 방법:

① 뷰 요소를 삭제하기 위해서는 다이어그램 상에 나타나 있는 뷰 요소를 먼저 선택한다.

② [Del]키를 누르거나 메뉴에서 [Edit]->[Delete]를 선택하면 된다.

※ 참고

- 뷰 요소를 삭제하더라도 관련된 모델 요소는 삭제되지 않다.

(13) 선 색상 적용하기

뷰 요소들의 테두리 선이나 연결 선 등의 색상을 변경 적용할 수 있다.

- 선 색상 적용하는 방법:

① 다이어그램 영역에서 선 색상을 변경할 요소들을 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Format]->[Line Color] 메뉴를 선택한다.

③ 색상 대화상자가 나타나면 적용하고자 하는 색상을 선택한 다음 [OK] 버튼을 누른다.

(14) 채움 색상 적용하기

뷰 요소들의 내부 영역에 채워질 색상을 변경 적용할 수 있다.

- 채움 색상을 적용하는 방법:

① 다이어그램 영역에서 채움 색상을 변경할 요소들을 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Format]->[Fill Color] 메뉴를 선택한다.

③ 색상 대화상자가 나타나면 적용하고자 하는 색상을 선택한 다음 [OK] 버튼을 누른다.

(14) 글꼴 적용하기

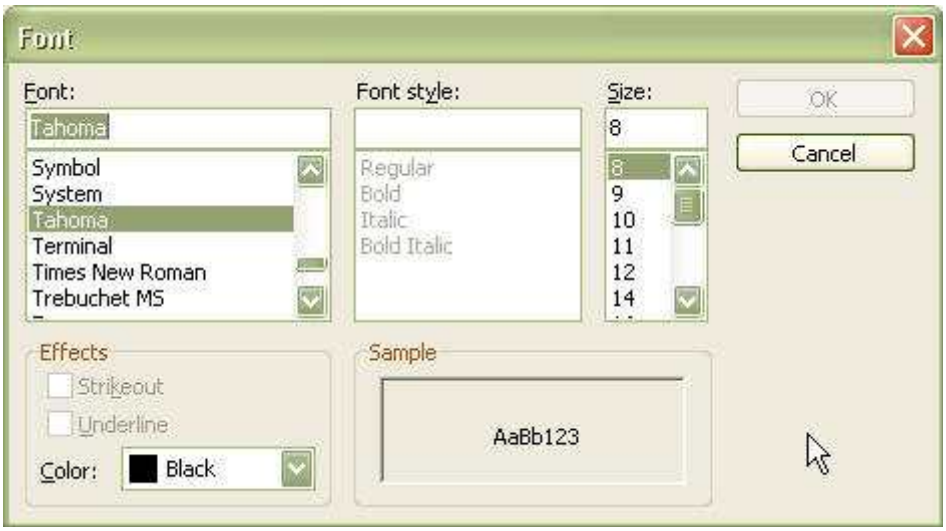
뷰 요소들의 나타나는 텍스트들의 글꼴 모양, 색상 및 크기 등을 변경 적용할 수 있다.

- 글꼴을 적용하는 방법:

① 다이어그램 영역에서 글꼴을 변경할 요소들을 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤, [Format]->[Font] 메뉴를 선택한다.

③ 글꼴 대화상자가 나타나면 원하는 글꼴 모양, 크기 및 색상 등을 선택한 다음 [OK] 버튼을 누른다.

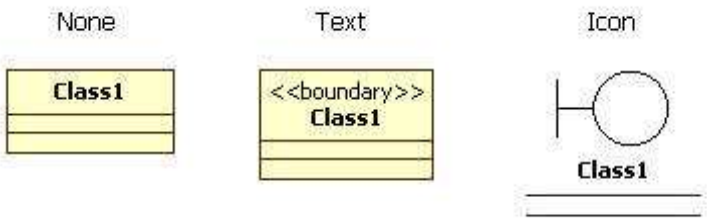


※ 참고

- UML 관련된 뷰 요소들은 [Style]을 지정할 수 없는 경우가 있다. 그것은 글꼴 스타일은 UML 표기법상에 정의되어 있어서 변경할 수 없는 경우이기 때문이다.

(15) 스테레오타입 표시하기

스테레오타입에 따라 뷰 요소는 서로 다른 형태로 표현될 수 있다. 표현될 수 있는 형태는 다음과 같은 것들이 있다.



① None[Shift+Ctrl+N] : 스테레오타입을 표시하지 않다.

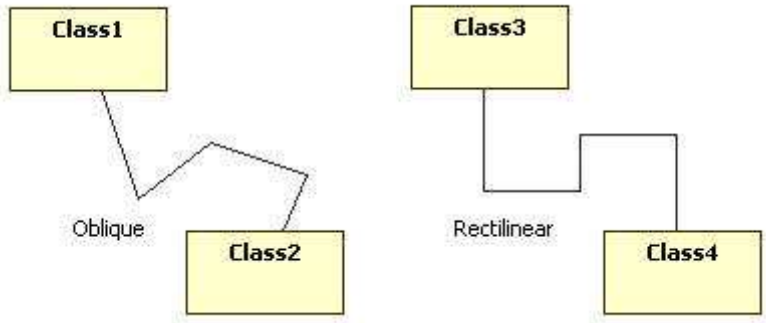
② Textual[Shift+Ctrl+T] : "<<", ">>"로 스테레오타입의 이름을 묶어서 표시한다.

③ Icon[Shift+Ctrl+I] : 뷰 요소를 스테레오타입의 아이콘 형태로 표시한다. 이 경우에는 반드시 해당 스테레오타입에 아이콘이 등록되어 있어야 하며, 그렇지 않은 경우에는 텍스트로 표시한다.

④ Decoration[Shift+Ctrl+E] : 뷰 요소를 텍스트와 작은 크기의 스테레오타입의 아이콘으로 표시한다. 이 경우에는 반드시 해당 스테레오타입에 아이콘이 등록되어 있어야 하며, 그렇지 않은 경우에는 텍스트로 표시한다. Actor, Interface, Component, Node, Artifact 등 일부 요소는 스테레오타입에 아이콘이 등록되어 있지 않아도 기본 아이콘으로 데코레이션 형태로 표시된다.

(16) 선 모양 설정하기

연관(Association), 의존관계(Dependency), 일반화(Generalization) 등과 같은 선 형태의 뷰 요소들은 다음과 같은 두 가지 형태의 선 모양을 지원한다.



① Rectilinear : 선이 항상 90도 각도로 꺾인 형태의 모양을 유지한다.

② Oblique : 선이 자유로운 각도로 꺾여서 표현될 수 있다.

- 선 모양을 변경하는 방법:

① 선 형태를 가지는 뷰 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 눌러 [Format]->[Line Style] 메뉴를 선택한 뒤, 직교선 혹은 꺾은선 중 원하는 선 모양에 해당하는 메뉴를 선택하여 체크한다.

(17) 자동크기 설정하기

뷰 요소들의 크기는 사용자가 유동적으로 변경할 수 있지만, 매번 크기를 직접 변경하지 않고 자동으로 그 크기를 결정하도록 설정해 둘 수 있다.

- 뷰 요소에 자동크기를 설정하는 방법:

① 자동으로 크기를 결정하고자 하는 뷰 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Auto Resize] 메뉴를 체크한다.

③ 자동크기 설정을 해제하려면 체크된 메뉴를 한번 더 선택하여 체크를 해제하면 된다.

(18) 속성 감추기

클래스(Class), 예외(Exception), 유스케이스(UseCase) 등과 같이 속성(Attribute)을 가지는 요소들은 다이어그램에서 자신의 속성 구획(Attribute compartment) 영역에 속성들을 나타낸다. 사용자는 필요에 따라서 이 속성들을 나타나게 할 수도 있고 또는 감출 수도 있다.

- 속성을 감추는 방법:

① 속성을 감추고자 하는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Suppress Attributes] 메뉴를 선택한다.

③ 속성을 다시 나타나게 하려면, 위 과정을 한번 더 수행하면 된다.

(19) 연산 감추기

클래스(Class), 예외(Exception), 유스케이스(UseCase), 서브시스템(Subsystem) 등과 같이 연산(Operation)을 가지는 요소들은 다이어그램에서 자신의 연산 구획(Operation compartment) 영역에 연산들을 나타낸다. 사용자는 필요에 따라서 이 연산들을 나타나게 할 수도 있고 또는 감출 수도 있다.

- 연산을 감추는 방법:

① 연산을 감추고자 하는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Suppress Operations] 메뉴를 선택한다.

③ 연산을 다시 나타나게 하려면, 위 과정을 한번 더 수행하면 된다.

(20) 리터럴 감추기

열거형(Enumeration)은 리터럴(Literal)들을 가지는데 이것은 다이어그램에서 열거형의 리터럴 구획(Literal compartment) 영역에 나타난다. 사용자는 필요에 따라서 이 리터럴들을 나타나게 할 수도 있고 또는 감출 수도 있다.

- 리터럴을 감추는 방법:

① 리터럴을 감추고자 하는 열거형 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Suppress Literals] 메뉴를 선택한다.

③ 리터럴들을 다시 나타나게 하려면, 위 과정을 한번 더 수행하면 된다.

(21) 워드랩 적용하기

요소의 이름이 한 단어 이상으로 길게 정의되어 있을 경우 뷰의 크기가 지나치게 늘어나 다이어그램의 전체적인 가시성을 떨어뜨리게 된다. 워드랩을 이용하면 요소의 긴 이름을 여러 줄로 나누어 표시하여 뷰의 크기를 적당하게 조정할 수 있다.

- 워드랩 적용하는 방법

① 워드랩을 적용하고자 하는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[WordWrapName] 메뉴를 선택한다.

③ 워드랩을 해제하려면, 위 과정을 한번 더 수행하면 된다.

(22) 상위 이름 표시하기

뷰 요소에는 일반적으로 자신의 이름만을 표시한다. 그러나, 여러 개의 패키지들로 구성된 프로젝트에서는 동일한 이름을 가진 요소가 서로 다른 패키지에 존재할 수 있고, 그것들이 하나의 다이어그램에 나타나야 하는 경우가 있다. 이러한 경우, 자신이 어디에 속해 있는지를 표현하여야 두 요소를 구분할 수 있으로 상위 요소의 이름도 함께 표현해야 하며 이때는 "상위요소이름::자신의이름"과 같은 형식으로 표시된다.

- 상위 이름을 표시하는 방법:

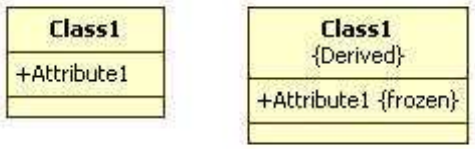
① 상위의 이름을 표시하고자 하는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Show Parent Name] 메뉴를 선택한다.

③ 이미 상위 이름이 표시된 요소들에서 상위 이름을 표시하지 않고 싶을 때에는 위 과정을 한번 더 반복하면 된다.

(23) 프로퍼티 표시

요소의 확장 속성 중에서 태그 값(Tagged Value)이나 요소의 Changeability 속성은 다이어그램에서 뷰 요소의 프로퍼티 부분으로 표시된다. 사용자는 필요에 따라서 이 프로퍼티 부분을 나타나게 할 수도 있고 감출 수도 있다.



- 프로퍼티를 표시하는 방법:

① 프로퍼티를 표시하고자 하는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Show Properties] 메뉴를 선택한다.

③ 프로퍼티를 감추려면, 위 과정을 한번 더 수행하면 된다.

※ 참고

- AssociationEnd 요소의 Changeability 프로퍼티 값이 CHANGEABLE 이거나 Ordering 프로퍼티 값이 UNORDERED 인 경우에는 해당 프로퍼티 값이 다이어그램의 뷰 요소의 프로퍼티 부분에 표시되지 않다.

(24) 연산 시그너쳐 표시하기

클래스(Class), 서브시스템(Subsystem) 등과 같이 연산을 포함하는 요소가 다이어그램에 나타날 때 연산들의 파라미터 이름과 타입을 모두 표시하게 할 수도 있고 이것들을 감출 수도 있다.

- 연산 시그너쳐 표시는 방법

① 연산의 시그너쳐를 표시하고자 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Show Operation Signature] 메뉴를 선택한다.

③ 나타난 연산의 시그너쳐를 감추려면 위 과정을 한번 더 수행하면 된다.

(25) 구획의 가시성 표시하기

속성(Attribute), 연산(Operation) 혹은 리터럴(Literal) 등을 가지는 클래스(Class), 유스케이스(UseCase), 서브시스템(Subsystem) 등과 같은 요소들이 다이어그램에 표현될 때에는 각 속성이나 연산들을 나타내는 구획(Compartment)을 가지고 있다. 클래스는 속성과 연산의 구획을, 서브시스템은 연산의 구획을, 그리고 열거형은 리터럴과 연산의 구획을 가진다. 이러한 구획 부분에 나타나는 요소들(속성, 연산 등)의 가시성을 표시하거나 표시하지 않을 수 있다.

- 구획의 가시성 표시하는 방법

① 구획의 가시성을 표시하려는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Show Compartment Visibility] 메뉴를 선택한다.

③ 구획의 가시성이 표시된 요소들에서 구획의 가시성을 표시하지 않으려면 위 과정을 한번 더 반복하면 된다.

(26) 구획의 스테레오타입 표시하기

속성(Attribute), 연산(Operation) 혹은 리터럴(Literal) 등을 가지는 클래스(Class), 유스케이스(UseCase), 서브시스템(Subsystem) 등과 같은 요소들이 다이어그램에 표현될

때에는 각 속성이나 연산들을 나타내는 구획(Compartment)을 가지고 있다. 클래스는 속성과 연산의 구획을, 서브시스템은 연산의 구획을, 그리고 열거형은 리터럴과 연산의 구획을 가진다. 이러한 구획 부분에 나타나는 요소들(속성, 연산 등)의 스테레오타입을 표시하거나 표시하지 않을 수 있다.

- 구획의 스테레오타입을 표시하는 방법

① 구획의 스테레오타입을 표시하려는 요소들을 다이어그램 영역에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Show Compartment Stereotype] 메뉴를 선택한다.

③ 구획의 스테레오타입이 표시된 요소들에서 구획의 스테레오타입을 표시하지 않으려면 위 과정을 한번 더 반복하면 된다.

(27) 다이어그램 열기

다이어그램을 편집하기 위해서는 우선 다이어그램을 열어야 한다. 다이어그램을 열면 다이어그램에 대한 탭이 생기게 되고, 탭을 선택하면 편집할 수 있는 활성 다이어그램(Active Diagram)이 된다.

- 다이어그램 여는 방법:

① 열고자 하는 다이어그램을 모델 탐색기 혹은 다이어그램 탐색기에서 찾는다.

② 해당 다이어그램을 항목을 마우스로 더블 클릭하면 다이어그램이 열리고 자동으로 활성 다이어그램이 된다.

(28) 다이어그램 활성화 시키기

여러 다이어그램을 열었을 때 특정 다이어그램을 편집하려면 해당 다이어그램을 활성화시켜야 한다. 이미 열려있는 다이어그램을 활성화하려면, 해당 다이어그램을 탭에서 클릭하면 된다. 열려져 있는 다이어그램이 많을 경우에는 팝업 메뉴에 나타나는 다이어그램 목록을 보고 선택하여 활성화시킬 수 있다.

- 메뉴에서 선택하여 다이어그램을 활성화 시키는 방법:

① 다이어그램 탭에서 마우스 오른쪽 버튼을 누른 뒤 [Pages] 메뉴를 선택한다.

② 서브 메뉴로 나타나는 다이어그램 목록 중 활성화시킬 다이어그램 이름을 선택한다.

(29) 다이어그램 닫기

다이어그램을 더 이상 편집하지 않는다면 다이어그램을 닫아 둘 수 있다. 다이어그램을 닫는 것은 다이어그램이 삭제되는 것과는 분명히 다르며, 언제든지 다시 열 수 있다.

- 다이어그램을 닫는 방법

① 닫고자 하는 다이어그램의 탭을 선택하여 활성 다이어그램으로 둔다.

② 탭 위에서 마우스 오른쪽 버튼을 누른 다음 [Close Diagram] 메뉴를 선택한다.

- 열려진 모든 다이어그램을 닫는 방법:

① [View]->[Close All Diagrams] 메뉴를 선택한다.

(30) 다이어그램 삭제하기

다이어그램이 더 이상 필요하지 않는 경우에는 다이어그램을 삭제할 수 있다. 다이어그램을 삭제하면 다이어그램에 관련된 모든 내용이 사라지므로 주의해야 한다.

- 다이어그램을 삭제하는 방법:

① 삭제하고자 하는 다이어그램을 모델 탐색기 혹은 다이어그램 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 누른 뒤 [Delete From Model]를 선택한다.

(31) 요소 찾기

보통 소프트웨어 모델 내에는 매우 많은 요소들이 만들어진다. 이렇게 많은 요소들

사이에서 원하는 요소를 찾아가기란 쉽지 않은 일이다. 이런 경우 요소들을 검색하여 원하는 요소로 빠르게 찾아갈 수 있다.

- 요소를 찾는 방법:

① [Edit]->[Find] 메뉴를 선택한다.

② 찾기 대화상자가 나타나면 [Text To Find] 부분에 찾으려는 요소의 이름 혹은 그 일부분을 입력하고, 찾으려는 요소의 타입에 제한을 두려면 [Option-Element Type] 부분에 요소의 타입을 지정한다. 검색에서 대소문자를 구분하고 싶다면 [Option-Case Sensitive] 부분을 체크한다. 그런 다음 [OK] 버튼을 누른다.

③ 검색된 결과들은 정보 영역의 [Message] 부분에 추가되며, 각 메시지를 더블 클릭하면 해당 요소로 찾아간다.

(32) 요소 정렬하기

다이어그램에 나열되어 있는 요소들을 일정한 방향 혹은 간격 등 다양한 방법으로 정렬할 수 있다.

- 요소 정렬 기능

- 요소들을 정렬하는 방법:

① 다이어그램 영역에서 정렬하고자 하는 요소들을 선택한다. (맨 앞으로 가져오기와 맨 뒤로 보내기를 제외한 정렬은 최소한 두 개 이상의 요소들을 선택해야 한다.)

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Alignment] 메뉴를 선택한 다음 원하는 정렬 기능에 해당하는 메뉴를 선택한다.

(33) 다이어그램 자동배치

다이어그램의 요소들이 무질서하게 배치되어 사용자가 알아보기 힘들 경우에 요소들을 보기 좋게 자동으로 배치시킬 수 있다.

- 다이어그램의 요소들을 자동으로 배치하는 방법:

① 자동으로 배치하고자 하는 다이어그램을 활성 다이어그램(현재 편집중인 다이어그램)으로 둔다.

② 마우스 오른쪽 버튼을 누른 뒤 [Format]->[Layout Diagram] 메뉴를 선택한다.

※ 참고

- 시퀀스 다이어그램에서는 자동배치 기능을 사용할 수 없다.

(34) 확대/축소 설정하기

다이어그램의 영역에 요소가 너무 많아 한눈에 보기 힘들거나 요소의 글자가 너무 작아서 잘 보이지 않을 때, 다이어그램을 확대하거나 축소할 수 있다.

① [View]->[Zoom] 메뉴를 선택한다.

② 그런 다음 다이어그램을 한 단계(5%) 확대하려면 [Zoom In] 메뉴를 선택, 한 단계

축소하려면 [Zoom Out] 메뉴를 선택하고, 다이어그램 전체를 화면에 모두 나타나게 하고 싶으면 [Fit To Window] 메뉴를 선택한다. 아니면 다이어그램의 원하는 크기에 해당하는 비율(50%, 75%, 100%, 125%, 150%, 175%, 200%)을 바로 선택하셔도 된다.

(35) 다이어그램을 이미지파일로 저장하기

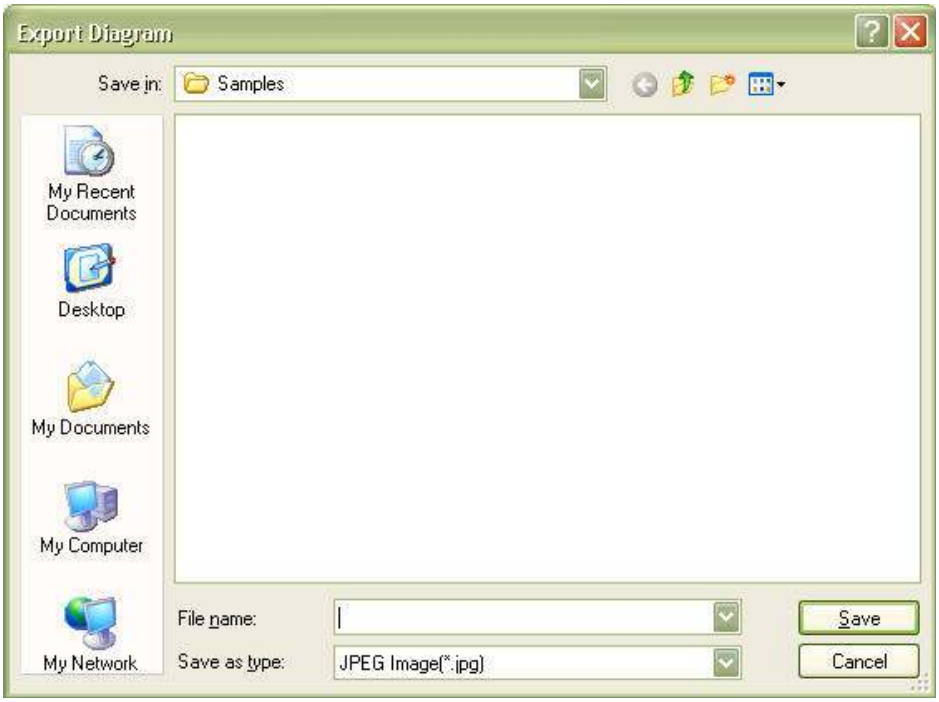
다이어그램을 별도의 이미지 파일로 저장할 수 있다. StarUML™에서는 JPEG(.jpg, .jpeg), 비트맵(.bmp), 메타파일(.wmf)과 확장메타파일(.emf)의 이미지 포맷을 지원한다.

- 다이어그램을 이미지로 저장하는 방법:

① 먼저 이미지로 저장하려는 다이어그램을 열어 활성 다이어그램(Active Diagram-현재 편집중인 다이어그램)으로 만든다.

② 메인 메뉴에서 [File]->[Export Diagram] 메뉴를 선택한다.

③ 저장 대화상자가 나타나면 파일 이름을 입력하고 원하는 파일 형식을 선택하고 [Save] 버튼을 누른다.



※ 참고

- 메타파일(.wmf) 이미지 포맷의 경우 특정 뷰어에 따라 이미지가 제대로 보이지 않을수 있다. 메타파일 이미지 포맷 저장시 확장메타파일(.emf) 이미지 포맷을 사용할 것을 권장한다.

(36) 다이어그램을 비트맵으로 복사하기

편집하고 있는 다이어그램을 다른 문서 등에 삽입하기 위해 다이어그램 이미지를 비트맵으로 복사할 수 있다. 비트맵으로 복사하면 편집 시에 보이는 그대로 문서에 삽입할 수 있지만, 확대 축소를 할 경우 이미지가 거칠게 보이거나 찌그러져 보일 수 있다.

- 다이어그램을 비트맵으로 복사하는 방법:

① 비트맵으로 복사하려는 다이어그램을 열어 활성 다이어그램으로 만든다.

② 메인 메뉴에서 [Edit]->[CopyDiagramAsBitmap] 메뉴를 선택한다.

(37) 다이어그램 탐색하기

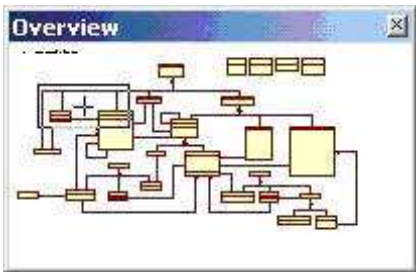
다이어그램이 많은 내용을 포함하고 있는 경우에는 그 크기가 매우 커질 수 있으므로, 화면 크기의 한계로 인하여 다이어그램의 많은 영역들이 가려지게 된다. StarUML™은 다이어그램의 영역을 효과적으로 탐색할 수 있는 방법들을 제공하여 사용자가 원하는 다이어그램의 영역으로 빠르게 이동할 수 있도록 도와준다. 다이어그램을 탐색하는 방법에는 다음과 같은 것들이 있다.

- 스크롤바(ScrollBar)와 휠(Wheel)로 탐색하기

스크롤 바를 이용하여 원하는 다이어그램 영역으로 이동한다. 휠-마우스(Wheel Mouse)를 사용하는 경우에는 마우스의 휠을 사용하여 상하 이동을 할 수 있다.

- 버드-뷰(Bird View)로 탐색하기

다이어그램 영역의 오른쪽 맨 아래에 작은 아이콘  이 있다. 이것을 클릭하면 다이어그램 전체를 작은 영역에 보여주는데, 마우스를 누른 채로 원하는 영역으로 이동한 뒤 마우스 버튼을 떼면 된다. 먼 영역으로 이동하고자 할 때에는 이 기능이 편리한다.



- Ctrl + 마우스로 이동하기

Ctrl키를 누른 채로 화면에 마우스 버튼을 누르고 마우스를 이동하면 다이어그램이 이동한다. 가까운 영역으로 이동하고자 할 때에는 이 기능이 편리한다.

(38) 기본 다이어그램으로 설정하기

프로젝트에는 많은 다이어그램들이 포함된다. 이 중에서도 가장 기본적인 다이어그램(Default Diagram)이 하나 혹은 그 이상 있을 수 있다. 예를 들어, 프로젝트의 전체적인 구조를 표현한 다이어그램을 기본 다이어그램으로 설정할 수 있다. 기본 다이어그램은 클래스 다이어그램(Class Diagram), 유스케이스 다이어그램(UseCase Diagram), 컴포넌트 다이어그램(Component Diagram) 그리고 디플로이먼트 다이어그램(Deployment Diagram)만이 될 수 있다. 기본 다이어그램을 프로젝트를 읽어온 직후 자동으로 열리게 된다.

- 기본 다이어그램으로 설정하는 방법:

① 기본(Default)으로 설정하고자 하는 다이어그램을 모델 탐색기 혹은 다이어그램 탐색기에서 선택한다.

② 인스펙터 영역에서 [Properties] 탭을 선택한다.

③ 프로퍼티 편집기에서 "DefaultDiagram"프로퍼티를 체크한다.

나. 모델 구조 구성하기

(1) 모델 요소 생성하기

다이어그램에는 나타나지 않게 모델 요소만을 생성할 수 있다. 이러한 모델은 어떠한 다이어그램에도 나타나지 않으며, 차후에 언제든지 다이어그램에 하나 이상의 뷰를 나타나게 할 수 있다.

- 모델 요소를 생상하는 방법:

① 생성할 요소가 포함될 요소를 모델 탐색기에서 선택한다.

② 마우스 오른쪽 버튼을 눌러 [Add] 메뉴를 선택한 뒤, 생성할 요소에 해당하는 메뉴를 선택하거나, 메인 메뉴에서 [Model]->[Add] 메뉴를 선택하여도 동일한 작업이 된다.

③ 요소가 생성되어 선택된 요소에 하위에 추가된다.

(2) 모델 요소 삭제하기

모델 요소를 삭제하는 것은 여러 가지 부수적인 요소들도 삭제된다. 따라서, 모델 요소를 삭제하는 것은 신중해야 한다. 모델 요소가 삭제될 경우 다음 요소들도 삭제된다.

① 포함하는 모델 요소 : 삭제될 모델이 직접 포함하고 있는 모든 모델 요소들도 함께 삭제된다.

② 관계 모델 요소 : 일반화(Generalization), 연관(Association), 의존(Dependency)등과 같이 삭제될 모델 요소에 연결된 관계류 들도 모두 삭제된다.

③ 뷰 요소 : 삭제될 모델들에 해당하는 모든 뷰 요소들이 삭제된다.

- 모델 요소를 삭제하는 방법:

① 삭제하고자 하는 모델 요소를 모델 탐색기에서 선택한다. 혹은 그에 해당하는 뷰 요소를 다이어그램 영역에서 선택한다.

② [Ctrl+ Del] 키를 누르거나 메뉴에서 [Edit]->[Delete From Model]를 선택한다.

③ 선택된 모델 요소가 삭제된다.

(3) 모델 요소 이동하기

모델 요소를 다른 요소의 하위로 이동할 수 있다. 예를 들어 클래스를 다른 패키지의 하위로 이동한다든지, 속성을 다른 클래스의 하위로 이동한다든지 하는 등의 행위를 수행할 수 있다. 모델 요소의 이동은 그것을 포함할 수 있는 요소로만 이동할 수 있고, 그 외에는 이동할 수 없다.

- 모델 요소를 다른 위치로 이동하는 방법:

① 모델탐색기에서 이동하고자 하는 요소를 선택한다.

② 해당 요소를 끌어서 그것을 포함할 요소에 놓는다.

(4) 모델 요소 순서 변경 하기

소프트웨어 모델의 구성을 직관적으로 볼 수 있게 하기 위하여 모델 요소들 간의 순서를 변경할 수 있다. 모델 요소들 간의 순서 변경은 동일한 종료의 요소들간에만 할 수 있다. 또, 모델 탐색기의 정렬 방식이 "Storage Order" 일 때만 유효한다.

- 모델 요소의 순서를 변경하는 방법:

① 모델탐색기에서 순서를 변경하고자 하는 요소를 선택한다.

② [Move Up] 버튼 또는 [Move Down] 버튼을 눌러 모델 요소를 한 칸씩 이동시킨다.

Attribute, Operation, Enumeration Literal 등 컬렉션 편집에 표시되는 요소들은 컬렉션 편집기에서 순서를 변경할 수 있다.

- 컬렉션 편집기에서 모델 요소의 순서를 변경하는 방법:

① 순서를 변경할 요소의 상위 요소를 선택한다.

② [Model]->[Collection Editor...] 메뉴를 선택하여 컬렉션 편집기를 실행한다.

③ 요소가 포함된 컬렉션에 해당하는 탭을 선택한다.

④ 순서를 변경할 요소를 선택한다.

⑤ [Move Up] 버튼 또는 [Move Down] 버튼을 눌러 모델 요소의 순서를 변경한다. Ctrl+ 커서키를 사용하여 순서를 변경할 수도 있다.

(5) 모델 정렬

모델 탐색기에 보이는 모델들의 구조를 저장 순서 또는 알파벳순으로 정렬할 수 있다. 정렬된 모델은 모델 탐색기가 정렬하여 보여주는 것일 뿐이며 실제 모델간의 순서가 변경되는 것은 아니다. 모델을 정렬하려면 모델 탐색기의 [저장 순서로 정렬] 버튼 또는 [알파벳 순서로 정렬] 버튼을 클릭하면 된다. 모델 정렬방식을 변경하면 모델 탐색기의 각 노드의 펼침 상태가 취소되고, 가장 상위 레벨의 노드만 펼쳐진다.

5. 다이어그램 모델링하기

가. UseCase 다이어그램 모델링하기

유스케이스 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Actor
- UseCase
- Association
- Derected Association
- Generalization
- Dependency
- Include
- Extend
- System Boundary
- Package

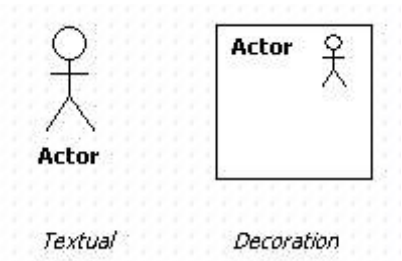
(1) Actor

- 의미:

액터(Actor)는 일반적으로 시스템 외부에 존재하면서 시스템과 상호작용하는 개체이다. 액터는 사람이거나 기계 혹은 소프트웨어 등이 될 수 있다.

- Actor 생성 방법:

Actor를 생성하려면, Toolbox>UseCase의 Actor 버튼을 클릭하고 Main 윈도우창에서 Actor가 위치할 곳을 클릭한다. Actor는 Stick Man 형태로 표현되지만, 사각형 모양에 오른쪽 상단에 아이콘이 포함된 Decoration View 형태로 사용되기도 한다. Actor를 Decoration View 형태로 보이도록 하기 위해서는 [Format]->[StereotypeDisplay]->[Decoration] 메뉴 아이템을 선택하거나 툴바의 버튼에서 [ ] 콤보 버튼의 [Decoration] 항목을 선택한다.



- Actor가 사용하는 UseCase를 한 번에 여러 개 생성하는 방법:

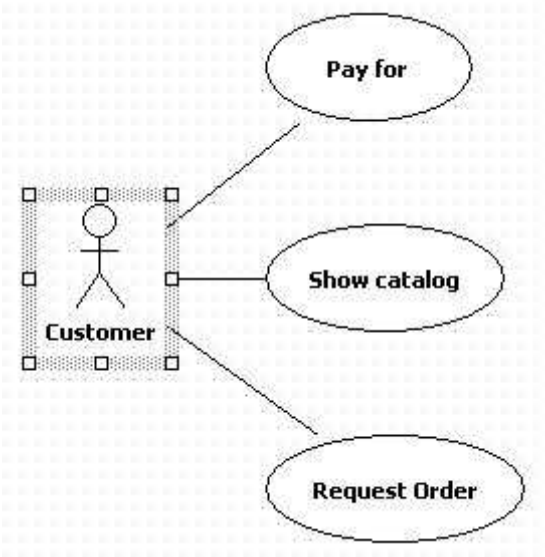
Actor가 사용하는 여러 개의 UseCase를 한꺼번에 만들려면 Actor의 단축 생성 구문을 사용한다.

① Actor를 더블 클릭해서 퀵다이얼로그가 나타나면, 퀵다이얼로그에서 "-()" 문자열

다음에 생성하려는 UseCase의 이름을 입력한다. 각 UseCase 이름은 "," 문자로 구분해서 입력한다.

② 그리고 [Enter]키를 누르면 Actor와 연관 관계를 가지는 여러 개의 UseCase가 수직으로

자동 배열되어 생성된다.



(2) UseCase

- 의미:

유스케이스(UseCase)는 시스템의 행위(behavior)를 정의하기 위해 사용하는 요소이다. 일반적으로 유스케이스는 액터와 상호작용한다.

- UseCase를 생성하는 방법:

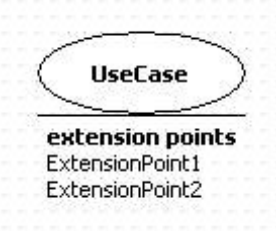
UseCase를 생성하려면, [Toolbox]->[UseCase]->[UseCase] 버튼을 클릭하고 Main 윈도우창에서 UseCase가 위치할 곳을 클릭한다.

UseCase는 Textual, Decoration, Iconic의 3가지 형태로 표현 가능하다.

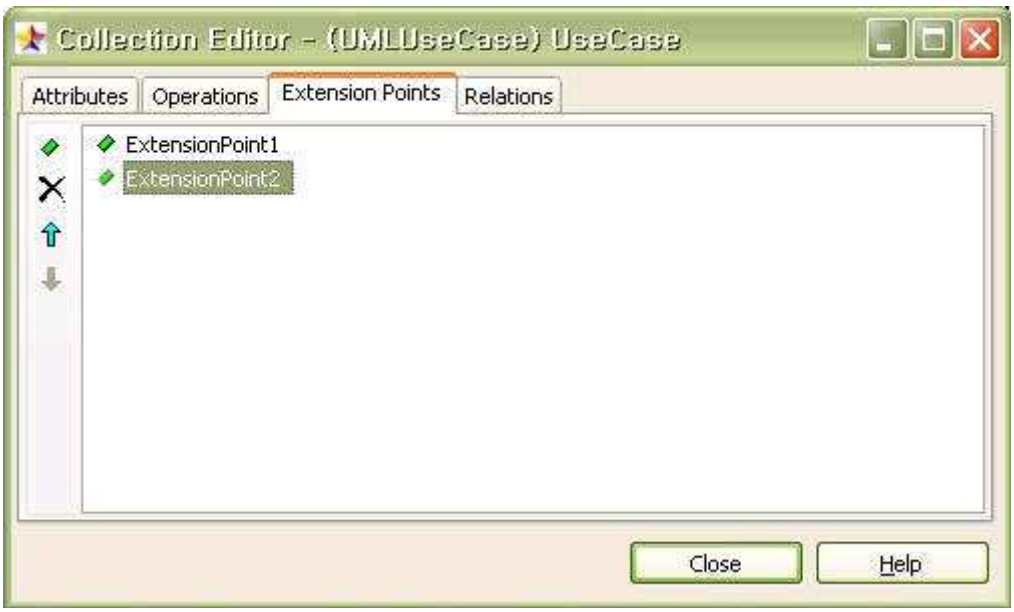
[Format]->[StereotypeDisplay]의 하부 메뉴 아이템을 선택하거나 [ ] 버튼의 아이템을 선택하면, UseCase의 스타일을 변경할 수 있다.

- Extension 추가하는 방법:

확장점은 유스케이스에서 확장되어지는 하나 또는 여러 개의 위치를 참조한다.



UseCase에 ExtensionPoints를 입력하려면 UseCase의 [CollectionEditor...] 팝업 메뉴를 클릭하거나 UseCase의 ExtensionPoints 컬랙션 속성의 ... 버튼을 클릭해서 [CollectionEditor]에서 값을 수정한다.



- UseCase Specification 속성 입력 방법:

UseCase 작성 시 많이 사용되는 속성들인 BasicFlow, AlternativeFlow등을 입력하기 위해서는 [TaggedValues...] 팝업 메뉴를 선택하거나 Ctrl+F7 버튼을 클릭하여 Tagged Value Editor의 UseCaseSpecification을 선택하여 필요한 속성의 값을 입력한다.

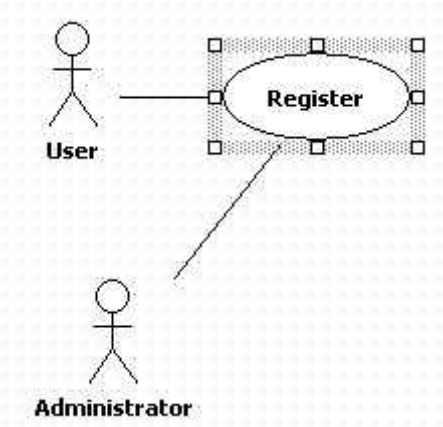
- UseCase로부터 Actor 생성하는 방법:

현재 선택된 UseCase와 연관 관계를 가지는 Actor 여러 개를 한꺼번에 만들려면

UseCase의 단축 생성 구문을 사용한다.

① UseCase를 더블 클릭하거나 UseCase를 선택하고 [Enter]키를 누른다. Quick Dialog가 나타나면, Quick Dialog에서 "()-" 문자열 다음에 연관된 Actor의 이름을 입력한다. 각 Actor 이름은 "," 문자로 구분해서 입력한다.

② 그리고 [Enter]키를 누르면 UseCase와 연관 관계를 가지는 Actor들이 생성된다.



(3) Association / Derected Association

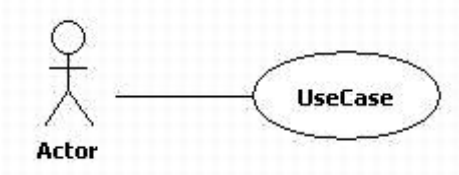
- 의미:

연관(Association)은 클래스류(Class, Interface, Enumeration, Signal, Exception, Component, Node, UseCase, Actor) 사이의 의미적 관계를 정의한다.

- Association 생성하는 방법:

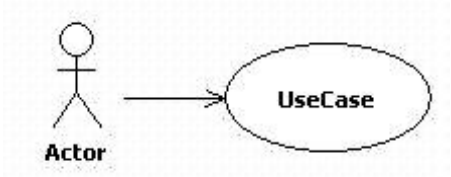
Association을 생성하려면, [Toolbox]->[UseCase]->[Association] 버튼을 클릭하고 Main 윈도우창에서 연결하려는 첫 번째 요소에서 두 번째 요소로 마우스를 누르고 드래그하면

된다.



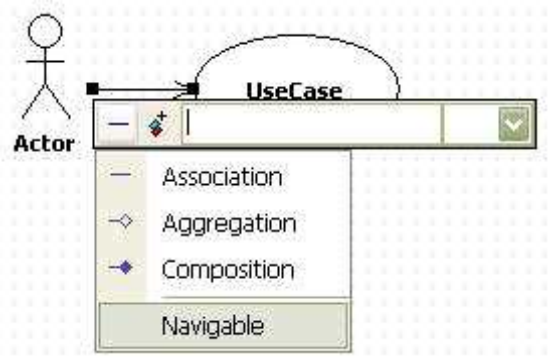
- DirectedAssocitaion 생성하는 방법:

Association 생성방법과 동일하며, 두 요소간 마우스 드래그를 화살표 방향으로 한다.



또는 Association을 생성하고 Actor쪽 association의 끝을 클릭하고 Quick Dialog의

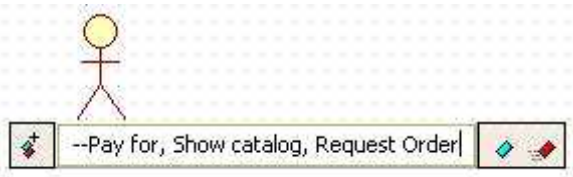
Navigable의 체크를 취소하면 DirectedAssociation으로 변한다.



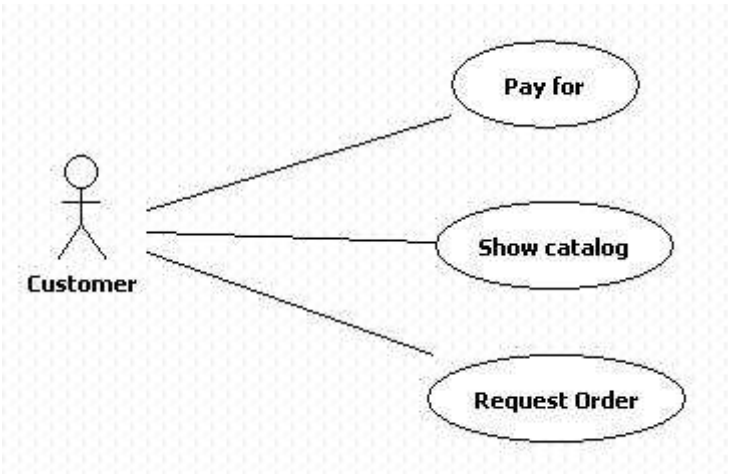
- 요소로부터 Assocation/Directed Association 관계의 요소 생성하는 방법:

현재 선택된 요소로부터 Assocation/DirectedAssociation 관계를 갖는 요소를 만들려면 요소의 단축 생성 구문을 사용한다.

① 요소를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "--" 또는 "->" 문자열 다음에 Association/DirectedAssociation 관계를 갖는 다른 요소의 이름을 입력한다. 여러 개의 요소와 관계를 맺기 위해서는 각 요소 이름은 "," 문자로 구분해서 입력한다.



② 그리고 [Enter]키를 누르면 선택된 요소와 Association/DirectedAssociation 연관 관계를 가지는 여러 요소들이 생성되고 자동 배열되어 생성된다.



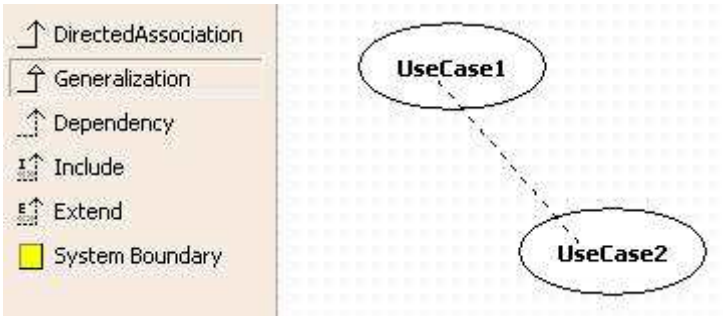
(4) Generalization

- 의미:

일반화(Generalization)">일반화(Generalization)는 더 일반적인 요소와 더 구체적인 요소를 연결하는 관계이다.

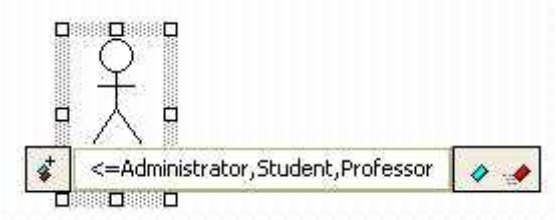
- Generalization 생성하는 방법:

Generalization을 생성하려면, [Toolbox]->[UseCase]->[Generalization] 버튼을 클릭하고 Main 윈도우창에서 연결하려는 자식 요소에서 부모 요소로 마우스를 누르고 드래그하면 된다.

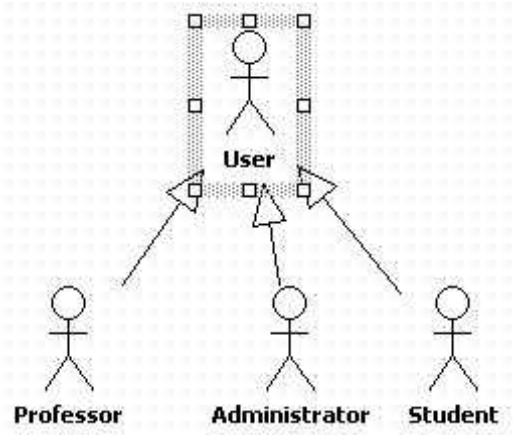


- Actor를 상속하는 여러 개의 자식 Actor 생성하는 방법:

① 특정 요소를 상속하는 하위 요소가 여러 개일 경우에 Quick Dialog의 단축 생성 구문에서 다음과 같이 입력하면 현재 요소를 상속하는 여러 개의 하위 요소를 한꺼번에 생성한다.



② 하위 요소들은 선택된 요소의 아래에 생성되면서 정렬된다.



만약 상속할 상위 요소가 여러 개인 경우에는 Quick Dialog의 단축 생성 구문에서 "<=" 대신에 "=>" 문자열을 사용한다.

(5) Dependency

- 의미:

의존관계(Dependency)">의존관계(Dependency)는 어떤 요소의 구현이나 기능을 위해 다른 요소의 존재가 요구 되어지는 의존적인 관계를 의미한다.

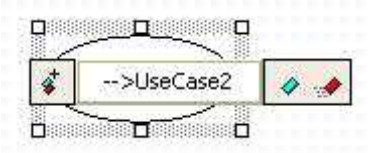
- Dependency 생성 방법:

Dependency를 생성하려면, [Toolbox]->[UseCase]->[Dependency] 버튼을 클릭

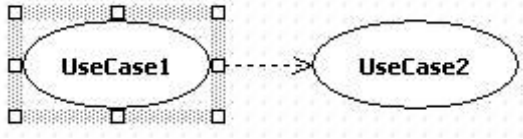
하고 Main 윈도우창에서 요소에서 의존 하는 요소로 마우스를 누르고 드래그하면 된다.

- UseCase로부터 의존하는 다른 UseCase 생성하는 방법:

퀵다이얼로그의 단축 생성구문을 다음과 같이 입력하면 된다.



그러면 다음과 같이 두 요소간의 Dependency가 생성된다.



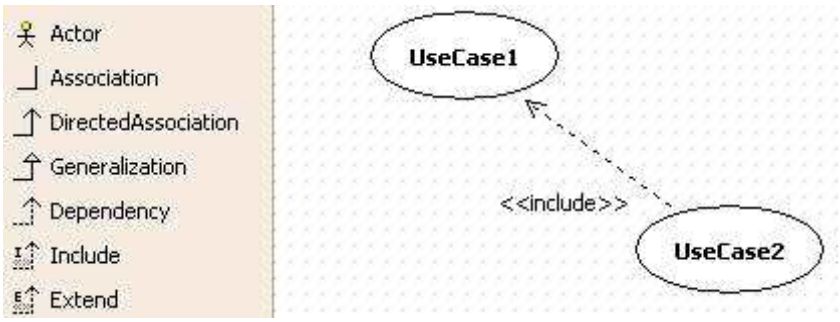
(6) Include

- 의미:

포함관계(Include)는 어떤 유스케이스가 특정 유스케이스의 행위를 포함한다는 것을 정의한다.

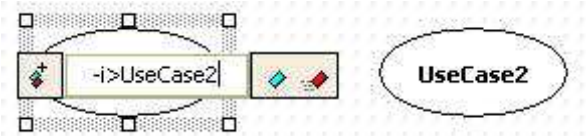
- Include 생성 방법:

Include를 생성하려면, [Toolbox]->[UseCase]->[Include] 버튼을 클릭하고 Main 윈도우창에서 요소에서 포함할 요소로 마우스를 누르고 드래그하면 된다.

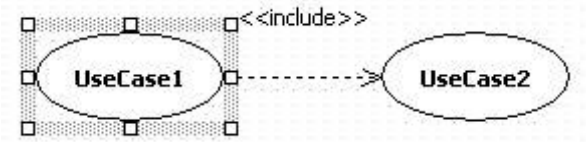


- UseCase로부터 Include 관계의 다른 UseCase 생성하는 방법:

퀵다이얼로그의 단축 생성구문을 다음과 같이 입력하면 된다.



그러면 다음과 같이 두 요소간의 Include가 생성된다.



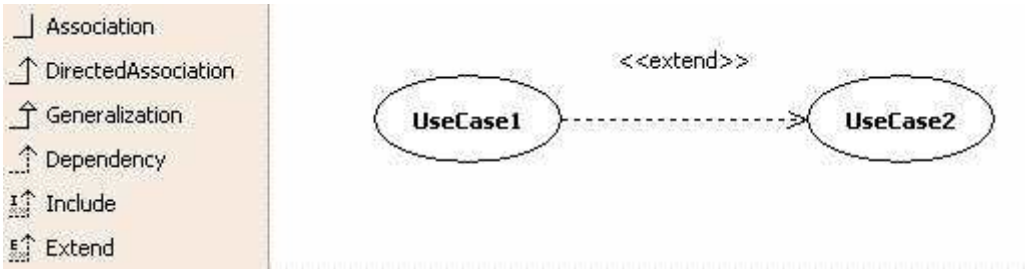
(7) Extend

- 의미:

확장관계(Extend)">확장관계(Extend)는 어떤 유스케이스가 특정 유스케이스에 정의된 행위로 추가 확장될 수 있다는 것을 나타낸다.

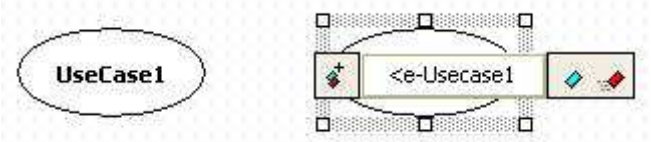
- Extend 생성 방법:

Extend를 생성하려면, [Toolbox]->[UseCase]->[Extend] 버튼을 클릭하고 Main 윈도우창에서 요소에서 확장할 요소로 마우스를 누르고 드래그하면 된다.

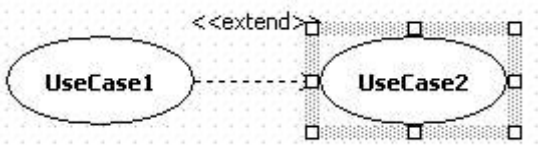


- UseCase로부터 Extend 관계의 다른 UseCase 생성하는 방법:

퀵다이얼로그의 단축 생성구문을 다음과 같이 입력하면 된다.



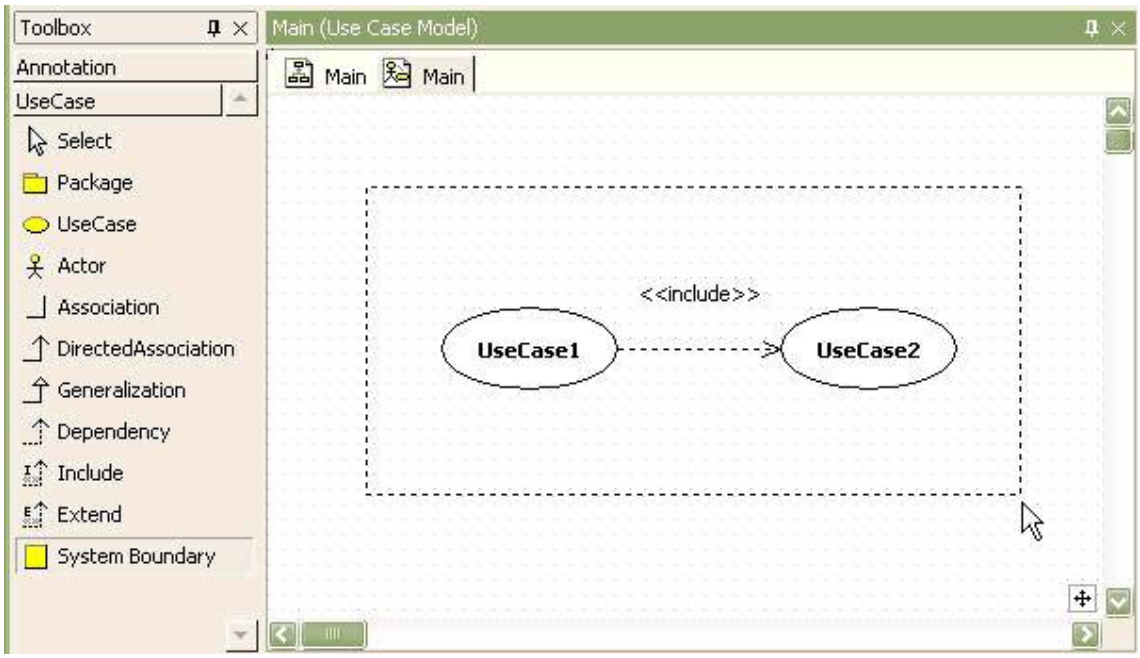
그러면 다음과 같이 두 요소간의 Extend가 생성된다.



(8) System Boundary

- System Boudary를 생성하는 방법:

System Boundary를 생성하려면, [Toolbox]->[UseCase]->[SystemBoundary] 의 System Boundary 버튼을 클릭하고 Main 윈도우창에서 System Boundary가 삽입될 위치에 마우스를 클릭하고 생성될 크기만큼을 드래그 한다.



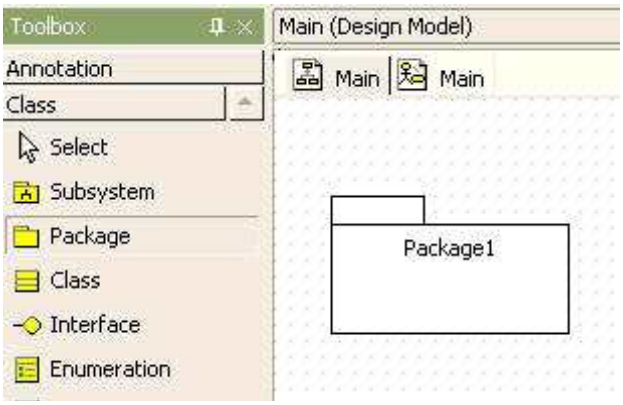
(9) Package

- 의미:

패키지(Package)는 모델 요소들을 논리적으로 그룹화 하여 관리하기 위한 요소이다. 패키지는 요소들을 조직화하기 위한 어떠한 용도로 사용되어도 무방한 매우 일반적인 요소이다. 패키지 대신 모델(Model), 서브시스템(Subsystem)의 더욱 특수화된 요소를 사용할 수도 있다.

- Package 생성하는 방법:

Package를 생성하려면, [Toolbox]->[UseCase]->[Package] 버튼을 클릭하고 Main 윈도우창에서 Package가 위치할 곳을 클릭한다.



나. Class 다이어그램 모델링하기

클래스 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Subsytem
- Package
- Class
- Interface
- Enumeration
- Signal
- Exception
- Port
- Part
- Association
- DirectedAssociation
- Aggregation
- Composition
- Generalization
- Dependency
- Realization
- AssociationClass

- Connector
- Object
- Link

(1) Subsystem

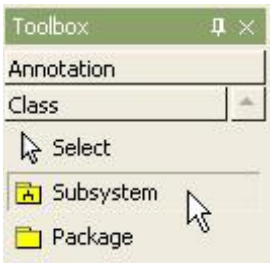
- 의미:

서브시스템(Subsystem)">서브시스템(Subsystem)은 물리적인 시스템의 부분 혹은 전체를 명세화하기 위해 요소들을 그룹화하는 요소이다.

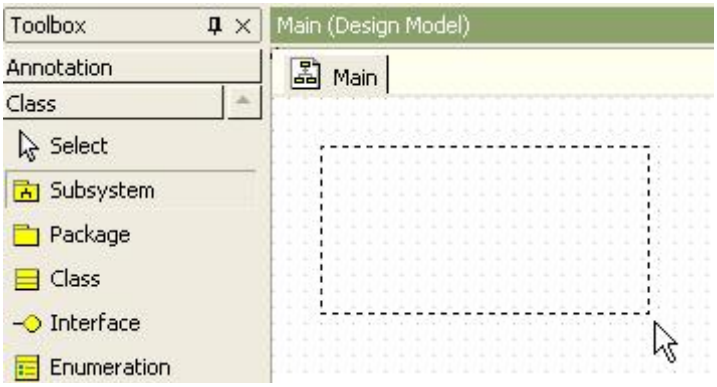
- Subsystem 생성하는 방법:

Subsystem을 생성하려면,

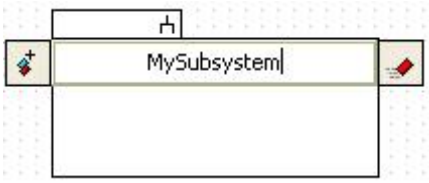
① [Toolbox] -> [Class] -> [Subsystem] 버튼을 클릭하고



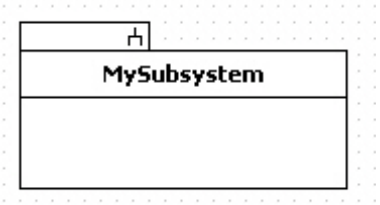
② Main 윈도우창에서 Subsystem가 위치할 곳을 클릭하고 원하는 크기만큼 드래그 한다.



③ 그러면 subsystem이 class diagram상에 생성되고 퀵다이얼로그가 나타난다.



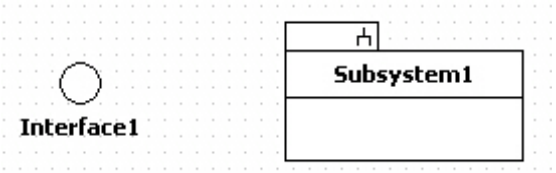
④ 퀵다이얼로그에서 subsystem의 이름을 입력하고 [Enter] 키를 누른다.



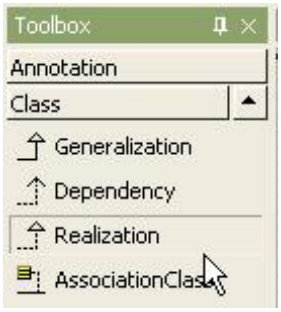
- Subsystem이 제공하는 Interface 생성하는 방법:

Subsystem은 Interface를 제공한다. Subsystem가 제공하는 Interface를 표현하려면,

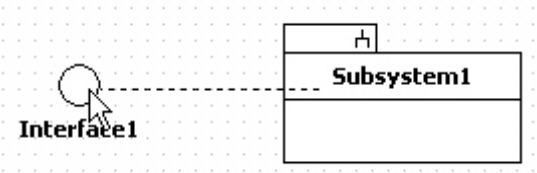
① interface와 susbystem을 생성하고



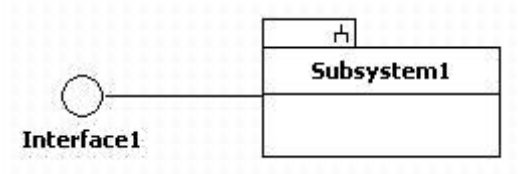
② [Toolbox] -> [Realization] 버튼을 클릭하고



③ Subsystem에서 Interface까지 마우스를 드래그 한다.

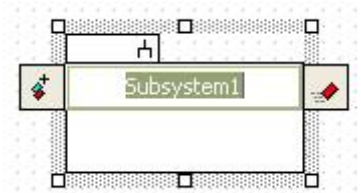


④ 그러면 interface와 subsystem 사이에 providing interface 관계가 생성된다.

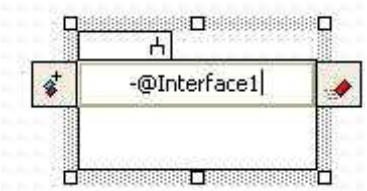


만약 Interface와 Realization을 동시에 생성하려면,

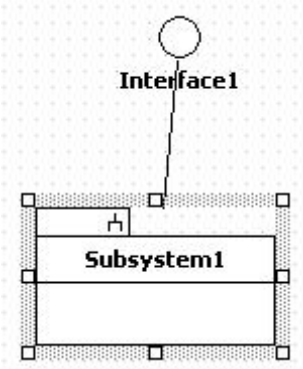
① Subsystem을 더블클릭하여 Quick Dialog가 나타나면



① 다음과 같이 단축 생성 구문을 입력한다.



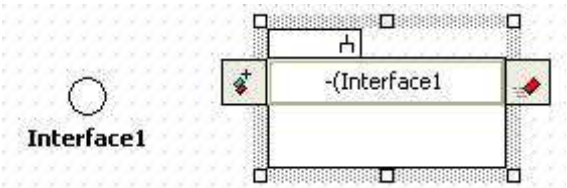
② [Enter]키를 누르면 Subsystem과 연결된 인터페이스를 생성된다.



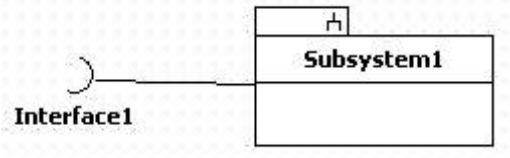
- Subsystem이 요구하는 Interface 생성하는 방법:

Subsystem이 Requiring Interface를 표현하려면 다음과 같이 단축 생성 구문을 입력한다.

① subsystem을 더블클릭하면 퀵다이얼로그가 나타난다. 그러면 다음과 같이 "-(" 문자열 다음에 interface 이름을 입력하고 [Enter] 키를 누른다.



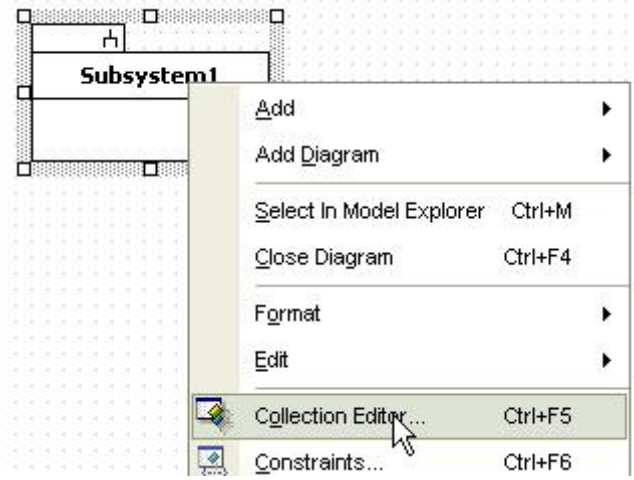
② 그러면 Subsystem이 Interface에 대해서 Requirement를 갖는 관계를 생성한다.



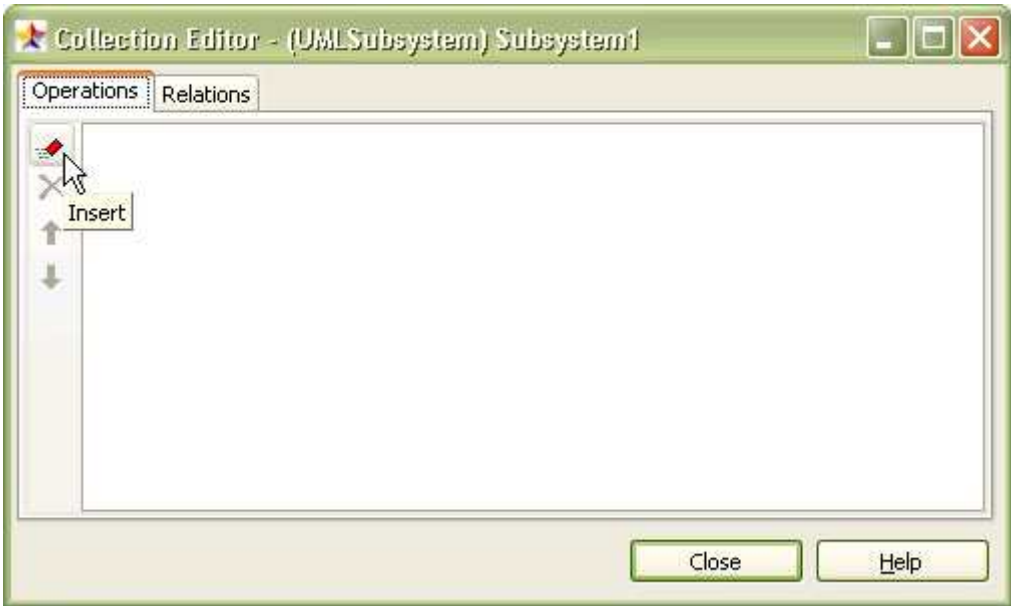
- Subsystem이 Providing Operation 추가하는 방법:

Subsystem은 Operation을 가질 수 있다. Subsystem이 갖는 Operation을 추가하려면,

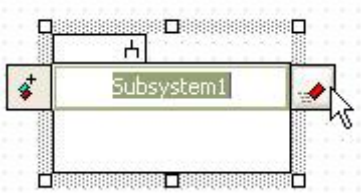
① Subsystem의 [CollectionEditor...] 팝업 메뉴를 클릭해서



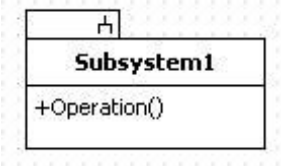
② Collection Editor의 Operations 탭에서 Operation을 추가한다.



③ 또는 Subsystem의 Quick Dialog에서  버튼을 클릭하여 Subsystem에 Operation을 추가한다.



④ 그러면 새로운 operation이 추가된다.



(2) Class

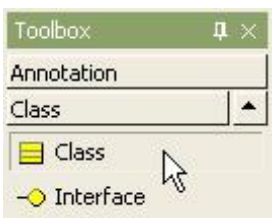
- 의미:

클래스(Class)는 객체의 구조와 행위를 묘사하는 속성(Attribute)과 연산(Operation)의 집합을 선언하는 요소이다. 그리고 클래스는 템플릿 파라미터 (Template Parameter)를 가질 수 있다.

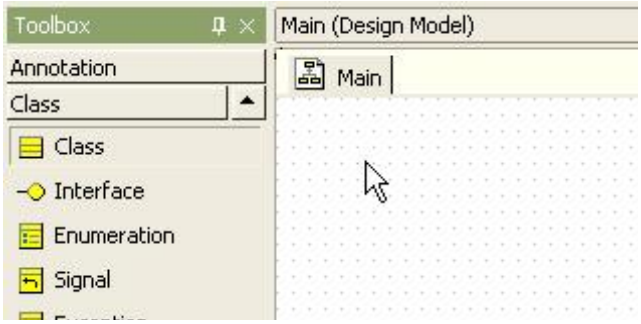
- Class 생성하는 방법:

Class를 생성하려면,

① [Toolbox] -> [Class] -> [Class] 버튼을 클릭하고

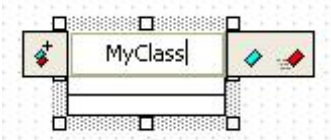


② Main 윈도우창에서 Class가 위치할 곳을 클릭한다.



③ 새로운 class가 다이어그램상에 생성되고, 퀵다이얼로그가 나타난다.

④ 퀵다이얼로그에서 class의 이름을 입력하고 [Enter] 키를 누른다.



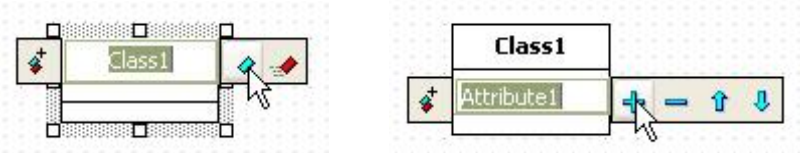
- Attribute 추가하는 방법:

Class에 attribute를 추가하는 방법은 다음과 같이 3가지 방법이 있다.

Quick Dialog를 이용하는 경우

① class를 더블클릭한다.

② [Add Attribute] 버튼을 누른다.



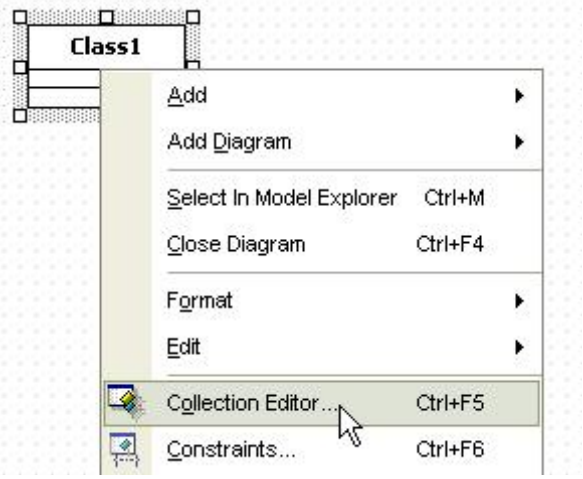
- Class 또는 Model Explorer의 팝업 메뉴 이용하는 경우

① main window 또는 model explorer에서 class를 선택한다.

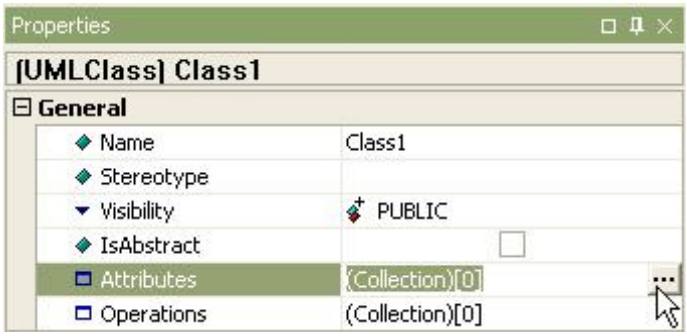
② 그리고 마우스 오른쪽 클릭을 통해서 [Add] -> [Attribute] 팝업 메뉴를 선택하여 attribute를 추가한다.

- Collection Editor 이용하는 경우에는

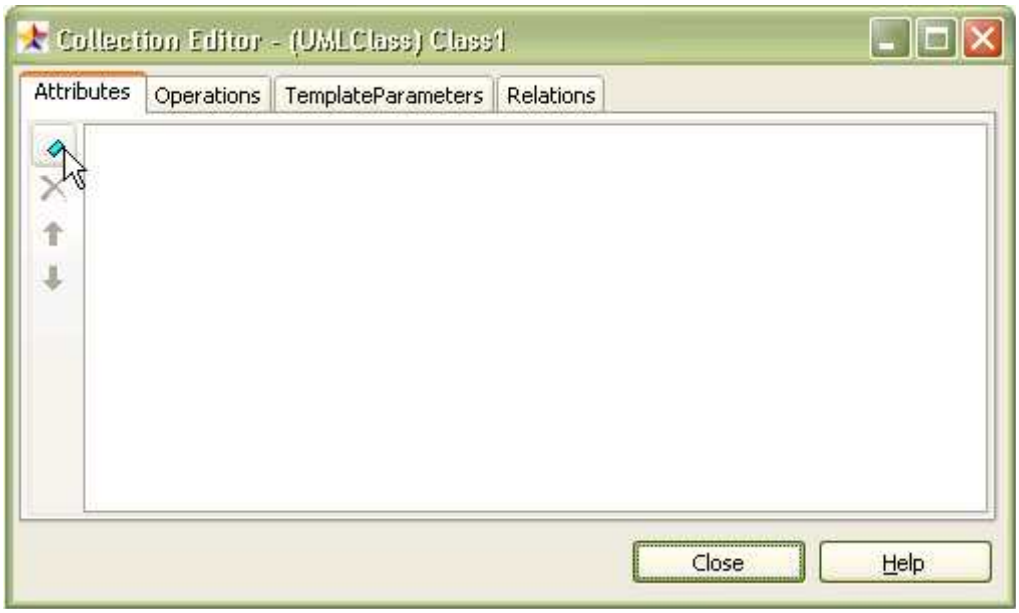
① Class의 [CollectionEditor...] 팝업 메뉴를 선택한다.



② 또는 Properties 윈도우의 Attributes의 ... 편집 버튼을 통해서 Collection Editor를 열어서



③ Attributes 탭에서 ◆ 버튼을 이용하여 Attribute를 입력할 수 있다.



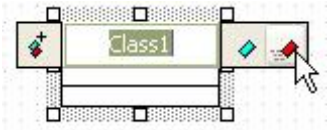
- Operation 추가하는 방법:

Class에 Operation을 추가하는 방법은 다음과 같이 3가지 방법이 있다.

- Quick Dialog를 이용하는 경우

① class를 더블클릭하면 퀵다이얼로그가 나타난다.

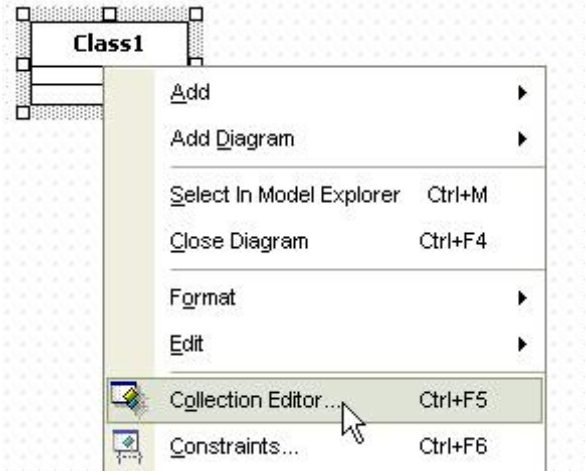
② 퀵다이얼로그에서 [Add Operation] 버튼을 누르면 operation이 추가된다.



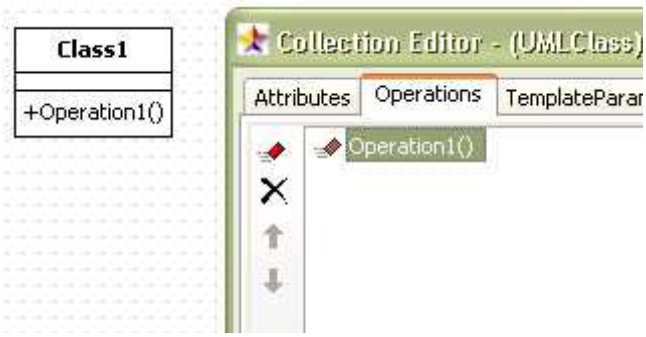
Class 또는 Model Explorer의 팝업 메뉴 이용하는 경우에는 Main 윈도우 또는 Model Explorer에서 Class를 선택하고 오른쪽 마우스 버튼을 눌러서 [Add] -> [Operation] 팝업 메뉴를 선택하여 Operation을 입력할 수 있다.

- Collection Editor를 이용하는 경우에는,

① Class의 [CollectionEditor...] 팝업 메뉴 또는 Properties 윈도우의 Operations의 콜랙션 편집 버튼을 통해서 Collection Editor를 열어서



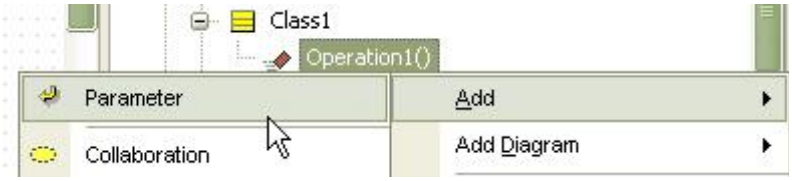
② Operations탭에서  버튼을 이용하여 Operation을 입력할 수 있다.



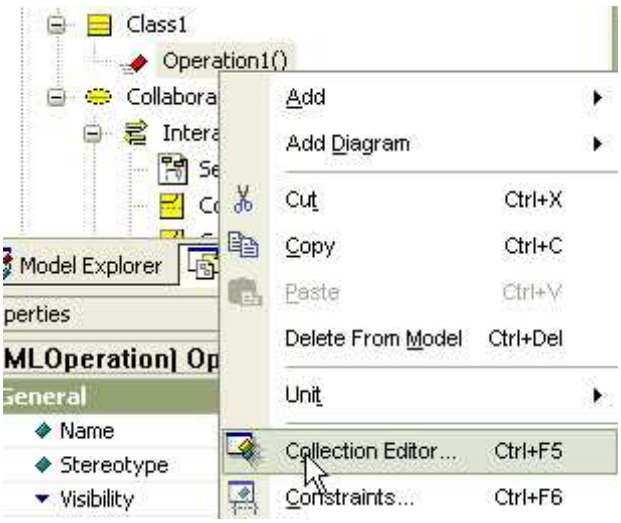
- Operation Parameter 추가하는 방법:

- operation에 parameter를 추가하기 위해서는

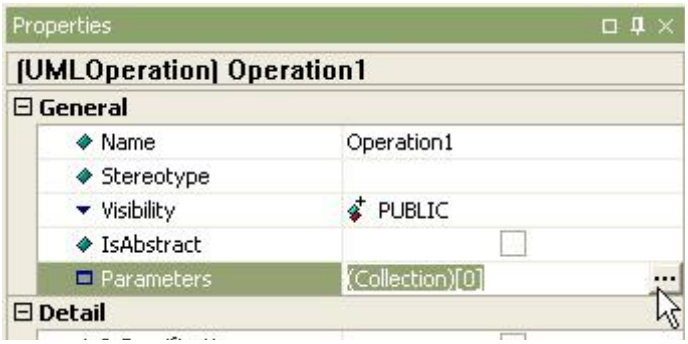
① model explorer에서 operation을 선택하고 [Add] -> [Parameter] 팝업 메뉴를 선택한다. 그러면 parameter가 추가된다.



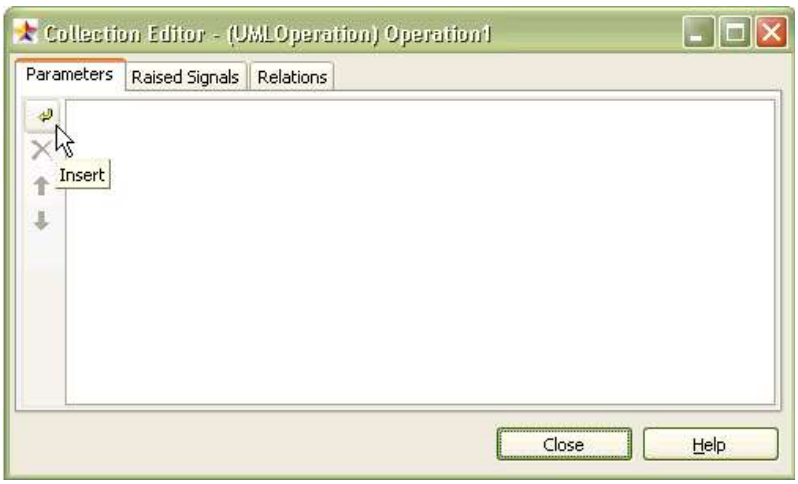
② 또는 model explorer에서 [Collection Editor...] 팝업 메뉴를 선택하거나



③ properties window에서 parameters의 ... 버튼을 클릭한다.



④ 그리고 collection editor의 parameters 탭에서 버튼을 이용해서 operation parameter를 추가할 수 있다.

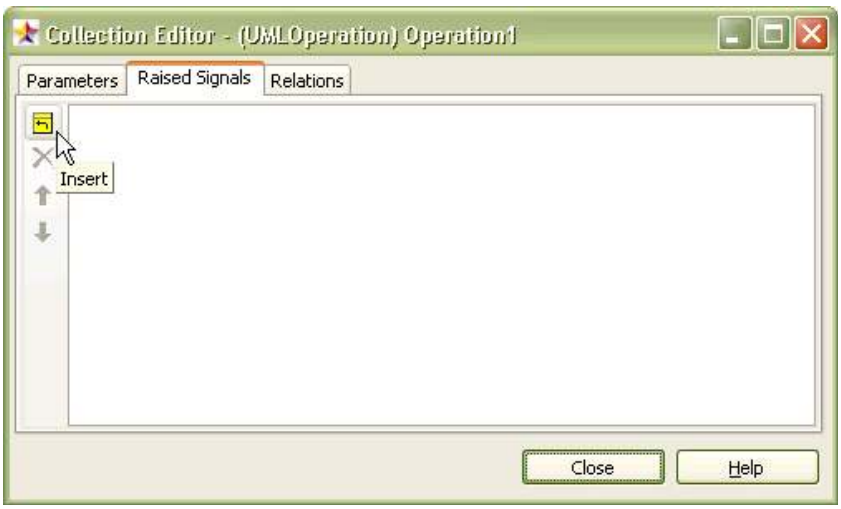


-Operation에 Exception 추가하는 방법:

Class에 Operation Exception을 추가하려면 (이 과정을 수행하기 전에 signal이 반드시 존재해야한다)

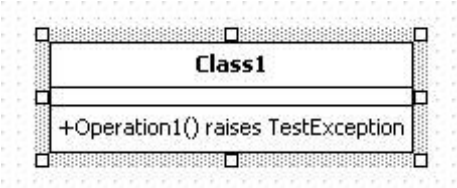
① properties window에서 raised signals 속성의 ... 버튼을 클릭한다.

② collection editor의 raised signal 탭에서  버튼을 클릭한다.



③ [Select a Signal] 다이얼로그가 나타나면 할당할 signal 또는 exception을 선택하고 [OK] 버튼을 누른다.

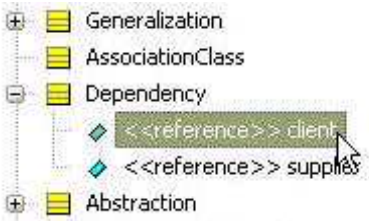
④ 그러면 다음과 같이 operation에 exception이 추가된다.



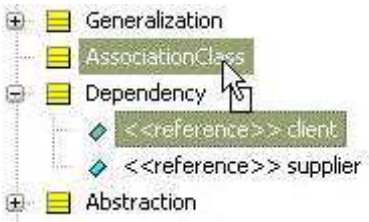
- Class간의 Attribute/Operation 이동하는 방법:

- class의 attribute와 operation을 다른 class로 이동하려면,

① model explorer에서 attribute(또는 operation)를 클릭한다.



② 선택된 요소를 다른 class로 드래그/드롭 한다.

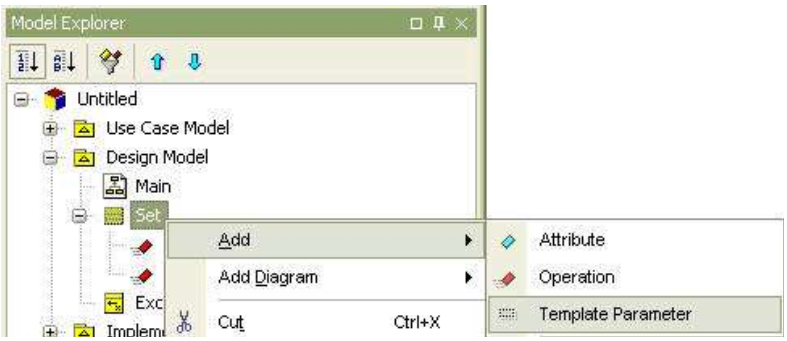


③ 그러면 다음과 같이 선택된 요소가 이동된다.

- Class에 TemplateParameter 추가하는 방법:

Class에 TemplateParameter를 추가하는 방법은 다음과 같이 2가지 방법이 있다.

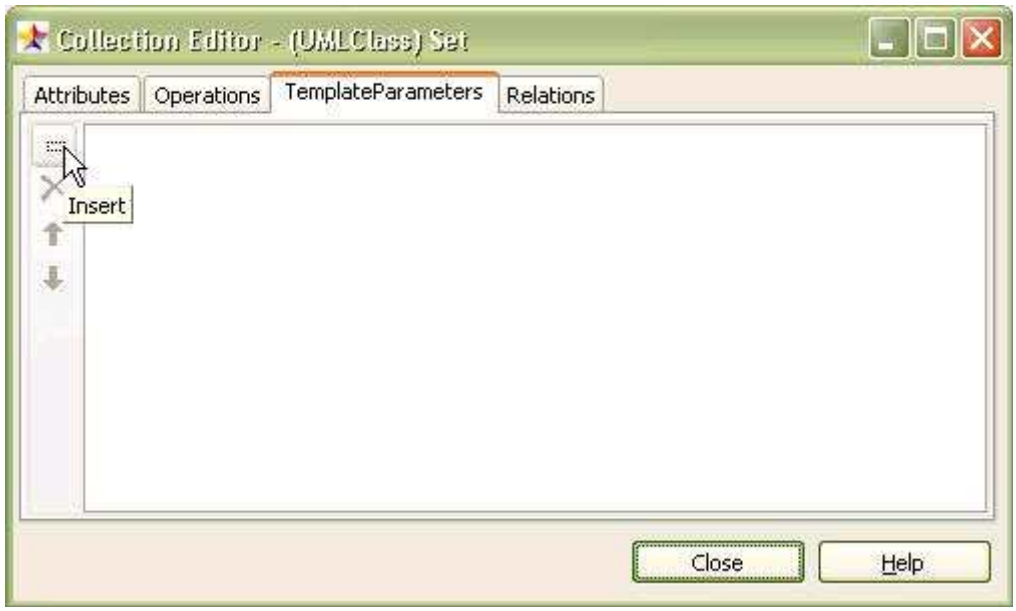
Main 윈도우 또는 Model Explorer에서 Class를 선택하고 오른쪽 마우스 버튼을 눌러서 [Add] -> [TemplateParameter] 팝업 메뉴를 선택하여 TemplateParameter를 입력할 수 있다.



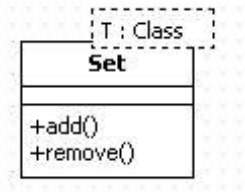
Class의 [Collection Editor...] 팝업 메뉴 또는 Properties 윈도우의 TemplateParameters의

 편집 버튼을 통해서 Collection Editor를 열어서 TemplateParameters 탭에서

 버튼을 이용하여 TemplateParameter를 입력할 수 있다.

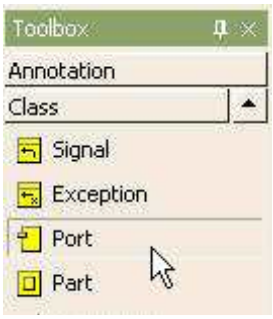


그러면 class에 새로운 template parameter가 추가되고 다음과 같이 class가 보인다.

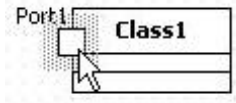


- Port 생성 방법:
- Class에 Port를 생성하려면,

① [Toolbox] -> [Class] -> [Port] 버튼을 클릭하고 .



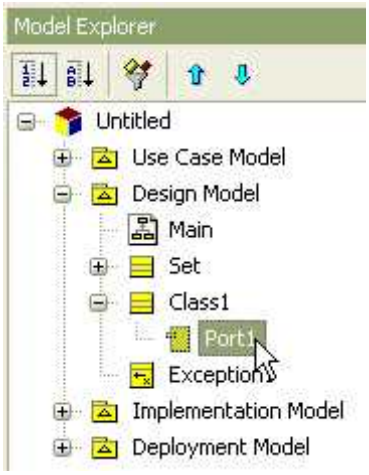
② Main 윈도우에서 Port가 위치할 Class를 클릭한다.



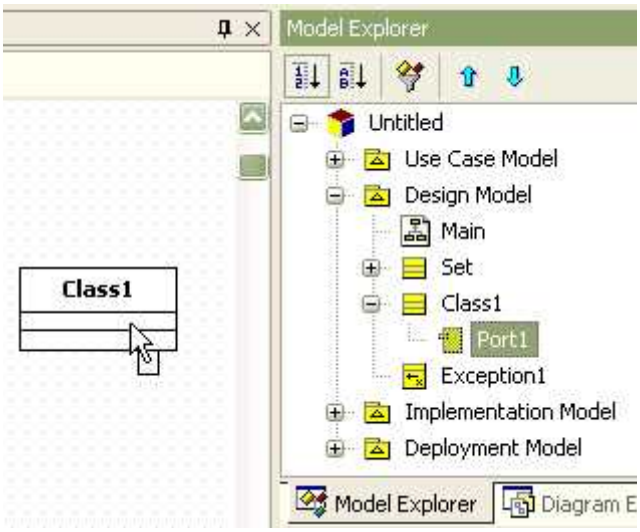
- Port 드래그를 통한 뷰 생성 방법:

Model Explorer로부터 드래그를 통하여 Port를 생성할 수 있다.

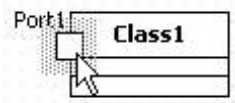
① model explorer에서 port를 드래그 한다.



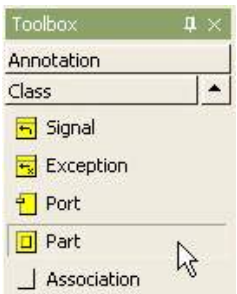
② 그리고 diagram상의 class로 드롭 한다. 만약 Class가 아닌 Diagram 위로 드롭 하면 Port와 함께 Class의 뷰가 생성된다.



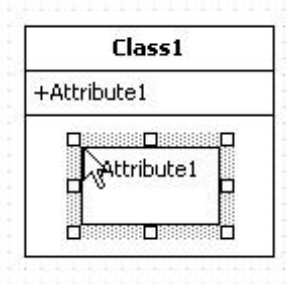
③ 그러면 다음과 같이 class가 port를 포함하게 된다.



- Part 생성 방법:
- Class에 Part를 생성하려면,
① [Toolbox] -> [Class] -> [Part] 버튼을 클릭하고

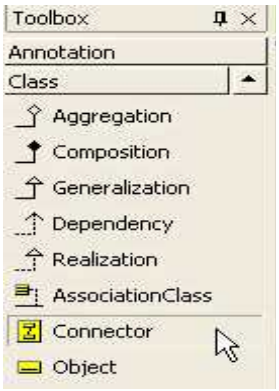


② Main 윈도우에서 Part가 위치할 Class를 클릭한다.

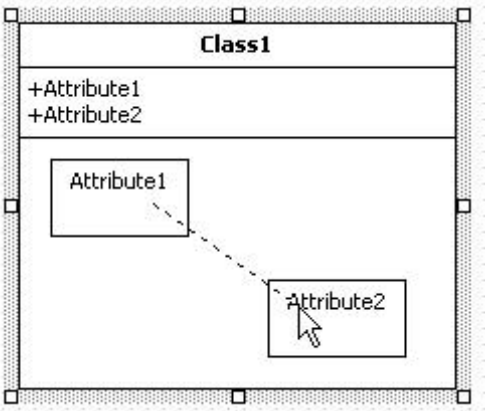


- Connector 생성 방법:
- Connector를 생성하려면,

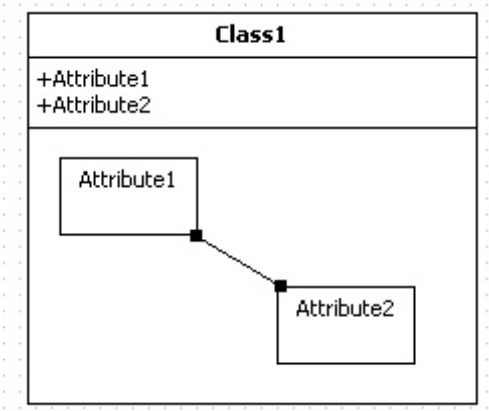
① [Toolbox] -> [Class] -> [Connector] 버튼을 클릭하고



② Main 윈도우에서 Part에서 연결할 Part로 마우스를 누르고 드래그하면 된다.

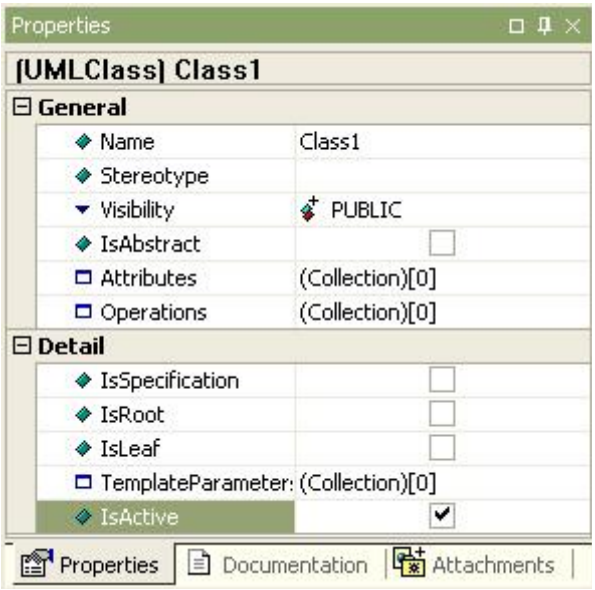


③ 그러면 두개의 part 사이에 다음과 같이 connector가 생성된다.

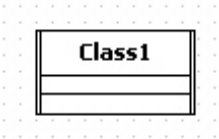


- Active Class로 설정하는 방법:
- Active Class로 설정하려면,

① Properties의 IsActive 속성을 true로 설정한다.



② active class로 변경되면 다음과 같이 class가 보인다.



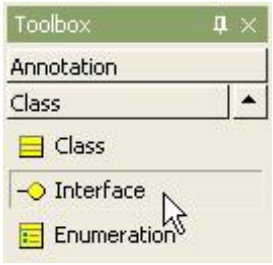
(3) Interface

- 의미:

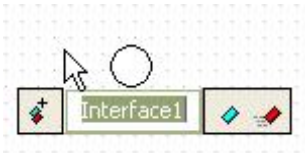
인터페이스(Interface)">인터페이스(Interface)는 클래스에 의해 제공되는 서비스를 구성하는 연산들을 포함하는 요소이다. 또한 연산들을 효과적인 그룹으로 나누고 그것들을 특징지을 수 있는 방법을 제공한다. 인터페이스에서부터 객체가 생성될 수 없다.

- Interface을 생성하는 방법:
- Interface를 생성하려면,

① [Toolbox] -> [Class] -> [Interface] 버튼을 클릭하고



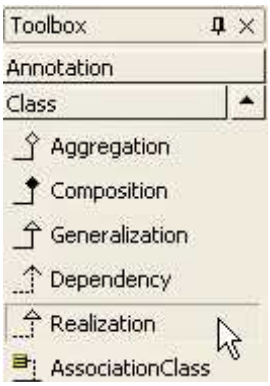
② Main 윈도우창에서 Interface가 위치할 곳을 클릭하면 퀵다이얼로그가 나타난다.



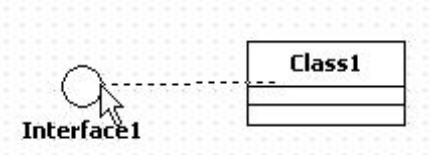
③ 퀵다이얼로그에서 interface의 이름을 입력하고 [Enter] 키를 누르면 다음과 같이 interface가 생성된다.

- Interface Providing 관계 생성 방법:
- Interface Providing 관계를 설정하려면,

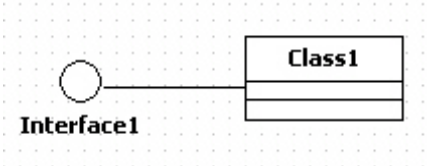
① [Toolbox] -> [Class] -> [Realization] 버튼을 클릭하고



② Main 윈도우창에서 요소(Class, Port, Part, Package, Subsystem)를 선택하고 Interface로 마우스를 누르고 드래그하면 된다.



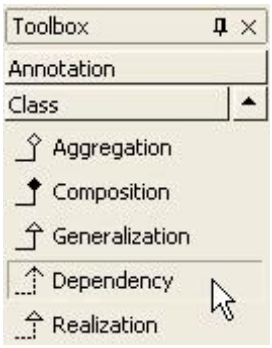
③ 그러면 providing interface 관계가 다음과 같이 생성된다.



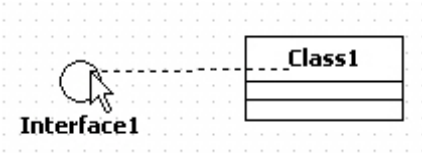
- Interface Requiring 관계 생성 방법:

- Interface Requiring 관계를 설정하려면,

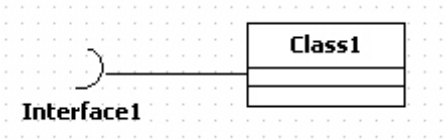
① [Toolbox] -> [Class] -> [Dependency] 버튼을 클릭하고



① Main 윈도우창에서 요소(Class, Port, Part, Package, Subsystem)를 선택하고 Interface로 마우스를 누르고 드래그하면 된다.



② 그러면 requiring interface 관계가 다음과 같이 생성된다.



(4) Enumeration

- 의미:

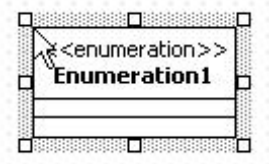
열거형(Enumeration)">열거형(Enumeration)은 미리 정의된 값들을 리스트로 가지는 데이터타입의 일종이다. 열거형이 가지는 값들을 열거형 리터럴(Enumeration Literal)이라 부른다.

- Enumeration 생성 방법:

- Enumeration을 생성하려면,

① [Toolbox] -> [Class] -> [Enumeration] 버튼을 클릭하고

② Main 윈도우창에서 Enumeration이 위치할 곳을 클릭한다.



(5) Signal

- 의미:

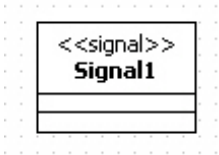
시그널(Signal)은 객체간의 비동기적(asynchronous) 통신 신호에 대한 명세(specification) 이다.

- Signal 생성 방법:

- Signal을 생성하려면,

① [Toolbox] -> [Class] -> [Signal] 버튼을 클릭하고

② Main 윈도우창에서 Signal이 위치할 곳을 클릭한다.



(6) Exception

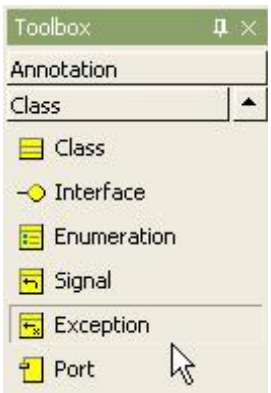
- 의미:

예외(Exception)는 실행 오류 시에 연산(Operation)에 의해 발생하는 시그널(Signal)이다.

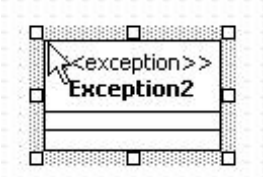
- Exception 생성 방법:

- Exception을 생성하려면,

① [Toolbox] -> [Class] -> [Exception] 버튼을 클릭하고



② Main 윈도우창에서 Exception이 위치할 곳을 클릭한다.



(7) Association

- 의미:

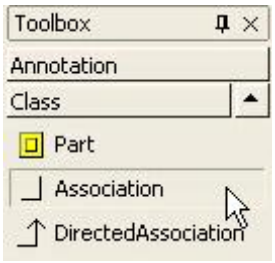
연관(Association)은 클래스류(Class, Interface, Enumeration, Signal, Exception, Component, Node, UseCase, Actor) 사이의 의미적 관계를 정의한다.

현재의 연관-끝(AssociationEnd) 방향의 포함관계를 의미한다. (- NONE: 집합이 아님을 나타냄, - AGGREGATE: 집합을 나타냄, - COMPOSITE: 합성을 나타냄)

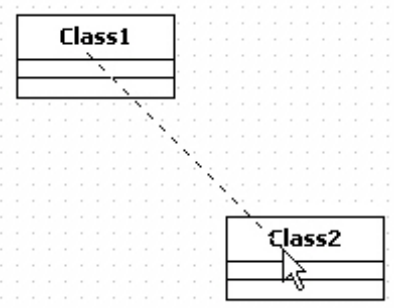
- Association 생성 방법:

- Association을 생성하려면,

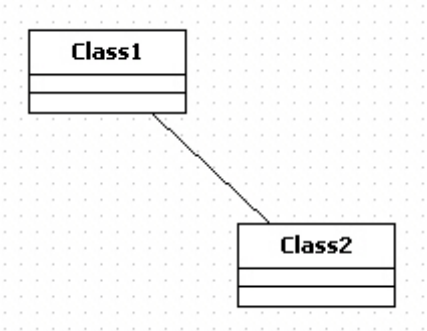
① [Toolbox] -> [Class] -> [Association] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 연관할 요소로 마우스를 누르고 드래그하면 된다.



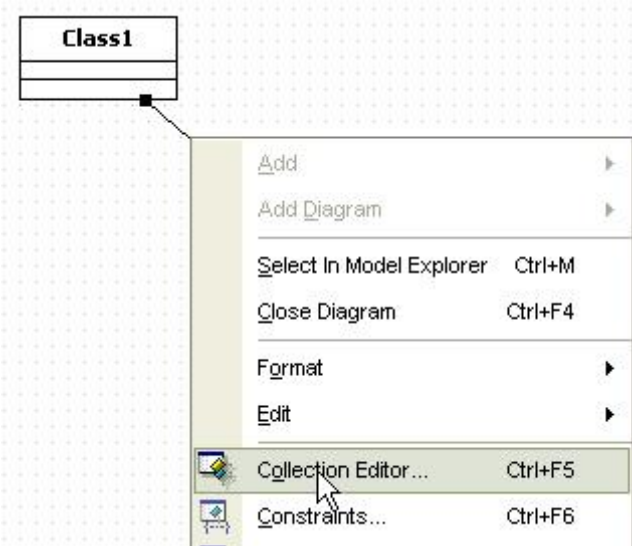
③ 두개의 class사이에 새로운 association이 다음과 같이 생성된다.



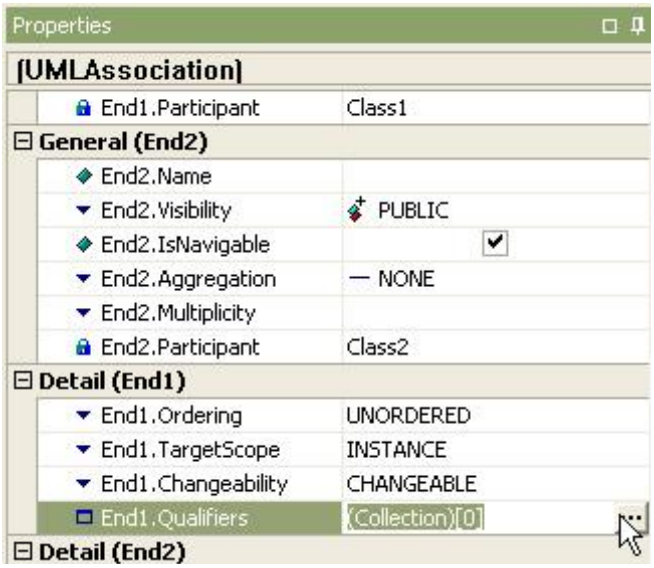
- Qualifier 추가하는 방법:

- Association에 Qualifier를 추가하려면,

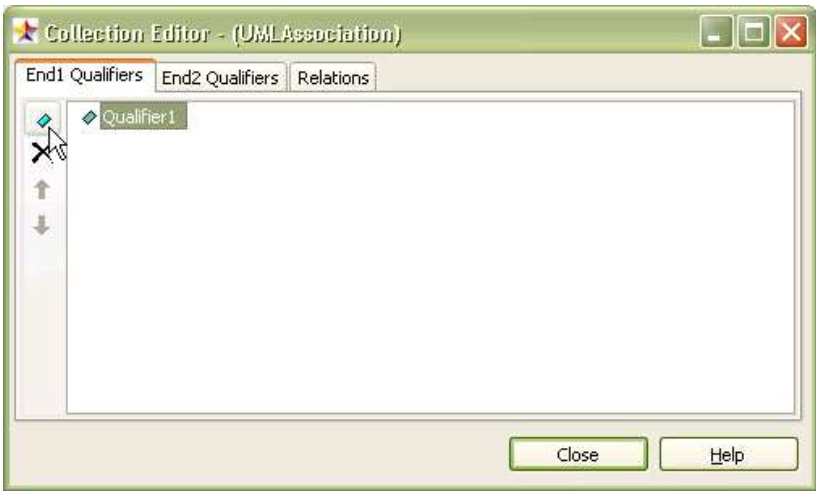
① association의 [Collection Editor...] 팝업 메뉴를 선택한다.



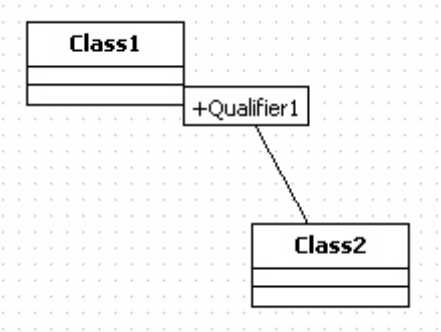
② properties window에서 End.Qualifiers 속성의 ··· 버튼을 클릭한다.



③ collection editor의 qualifiers 탭에서  버튼을 클릭하여 qualifier를 추가한다.

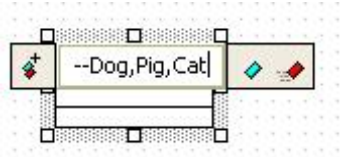


④ 결과는 다음과 같다.

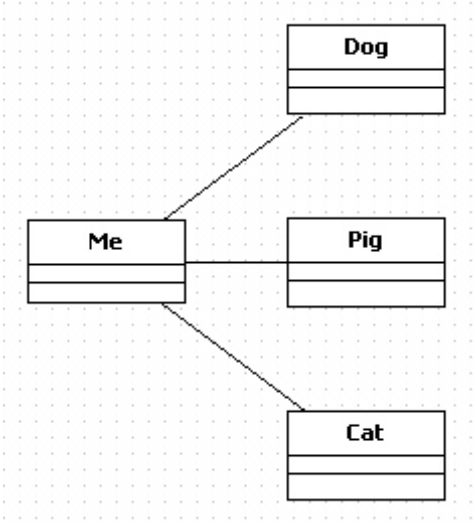


- Class로부터 연관하는 여러 개의 Class를 한꺼번에 생성하는 방법:
- 현재 class와 연관된 여러 개의 class를 한꺼번에 생성하려면,

① class를 더블클릭하고, 다음과 같이 "--" 문자열 다음에 연관할 class의 이름을 입력하고 [Enter] 키를 누른다.



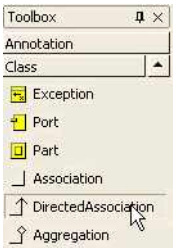
② 그러면 다음과 같이 연관된 class들이 생성되고 자동으로 정렬된다.



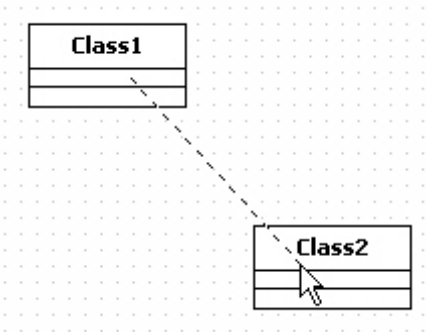
(8) DirectedAssociation

- DirectedAssociation 생성 방법:
- Association 생성방법과 동일하다.

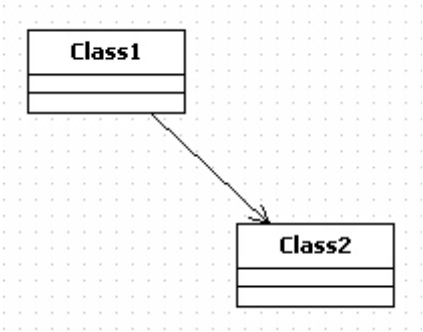
① [Toolbox] -> [Class] -> [DirectedAssociation] 버튼을 클릭한다.



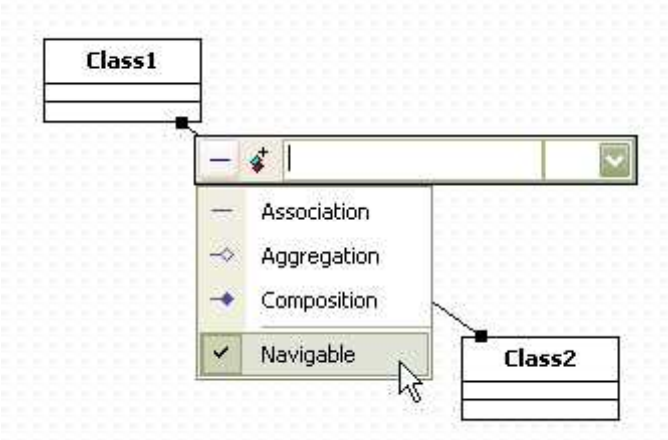
② 두개의 요소사이에 화살표 방향으로 드래그 한다.



③ 결과는 다음과 같다.

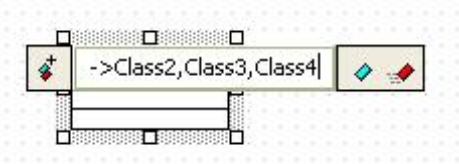


- Association으로부터 DirectedAssocation으로 변경 방법:
Association을 생성하고 화살표 반대편쪽 assocation의 끝을 클릭하고
Quick Dialog의 Navigable의 체크를 취소하면 DirectedAssociation으로 변한다.

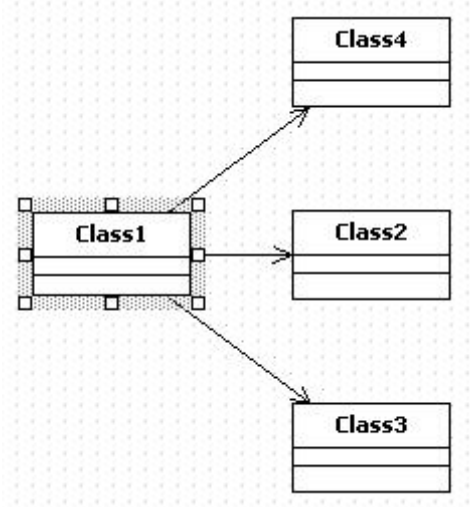


- Class로부터 DirectedAssociation 관계를 갖는 또 다른 클래스 생성 방법:
현재 선택된 Class로부터 DirectedAssociation 관계를 갖는 요소를 만들려면 요소의 단축 생성 구문을 사용한다.

① 요소를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "->" 문자열 다음에 DirectedAssociation 관계를 갖는 다른 요소의 이름을 입력한다. 여러 개의 요소와 관계를 맺기 위해서는 각 요소 이름은 "," 문자로 구분해서 입력한다.



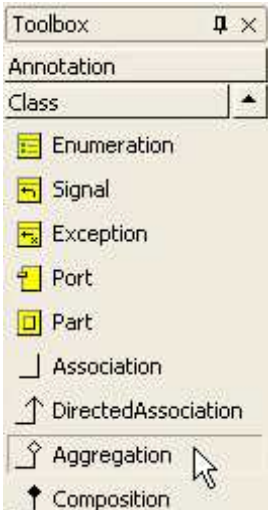
② 그리고 [Enter]키를 누르면 선택된 클래스와 DirectedAssociation 연관 관계를 가지는 여러 요소들이 생성되고 자동 배열되어 생성된다.



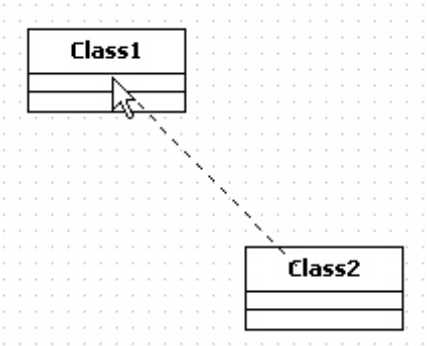
(9) Aggregation

- Aggregation 생성 방법:
- Aggregation을 생성하려면,

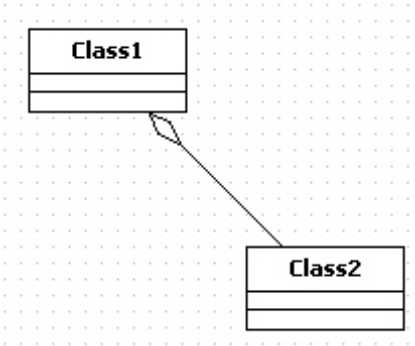
① [Toolbox] -> [Class] -> [Aggregation] 버튼을 클릭하고



② Main 윈도우창에서 포함되어지는 요소에서 포함하는 요소로 마우스를 누르고 드래그하면 된다.



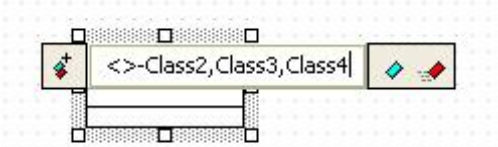
③ 결과는 다음과 같다.



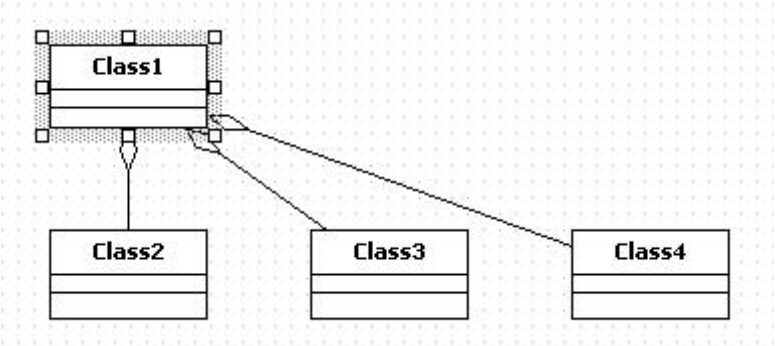
- 선택된 Class에 Aggregate되어지는 Class들을 생성하는 방법:

현재 선택된 Class로부터 Aggregated 관계를 갖는 Class를 만들려면 단축 생성 구문을 사용한다.

① Class를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "<>-" 문자열 다음에 Aggregated 관계를 갖는 다른 Class의 이름을 입력한다. 여러 개의 요소와 관계를 맺기 위해서는 각 Class 이름은 "," 문자로 구분해서 입력한다.



② 그리고 [Enter]키를 누르면 선택된 Class와 Aggregation 관계를 가지는 여러 Class들이 생성되고 자동 배열되어 생성된다.



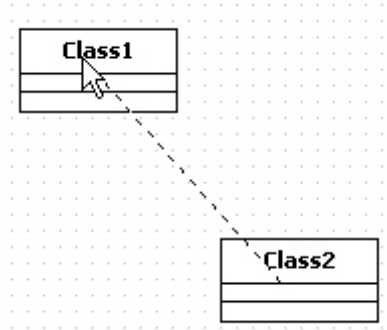
(10) Composition

- Composition 생성 방법:

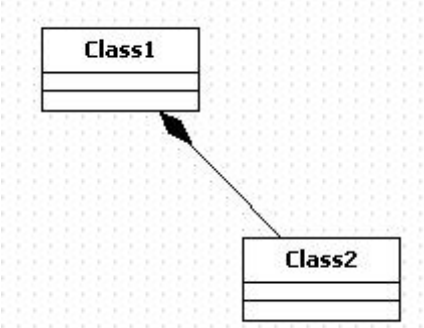
- Composition을 생성하려면,

① [Toolbox] -> [Class] -> [Composition] 버튼을 클릭하고

② Main 윈도우창에서 포함되어지는 요소에서 포함하는 요소로 마우스를 누르고 드래그하면 된다.



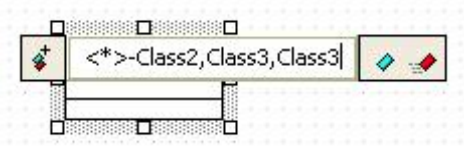
③ 그러면 두개의 class 사이에 새로운 composition 관계가 다음과 같이 생성된다.



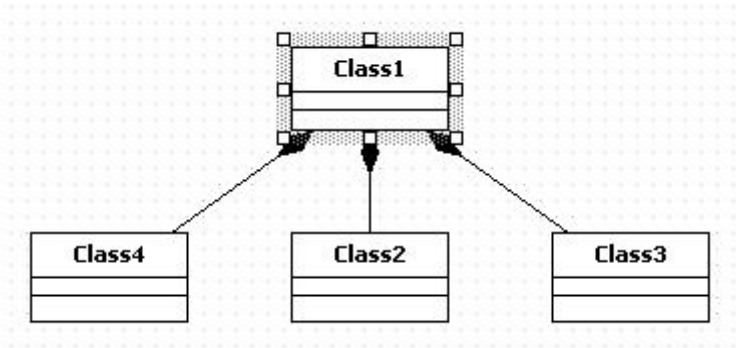
- 선택된 Class에 Composed Class들을 생성하는 방법:

현재 선택된 Class로부터 Composed 관계를 갖는 Class를 만들려면 단축 생성 구문을 사용한다.

① Class를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "<*>-" 문자열 다음에 포함되는 다른 Class의 이름을 입력한다. 여러 개의 Class를 포함하기 위해서 각 Class 이름은 "," 문자로 구분해서 입력한다.



② 그리고 [Enter]키를 누르면 선택된 Class와 Composition 관계를 가지는 여러 Class들이 생성되고 자동 배열되어 생성된다.



(11) Generalization

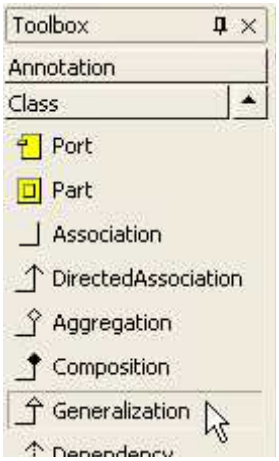
- 의미:

일반화(Generalization)는 더 일반적인 요소와 더 구체적인 요소를 연결하는 분류학적 관계이다.

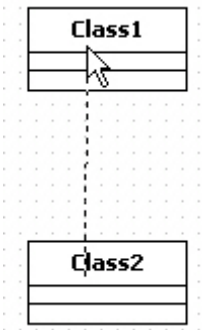
- Generalization 생성 방법:

- Generalization을 생성하려면,

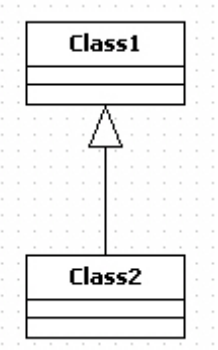
① [Toolbox] -> [Class] -> [Generalization] 버튼을 클릭하고



② Main 윈도우창에서 자식 요소에서 부모 요소로 마우스를 누르고 드래그하면 된다.

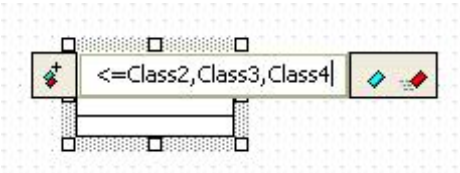


③ 그러면 새로운 generalization이 다음과 같이 생성되어진다.

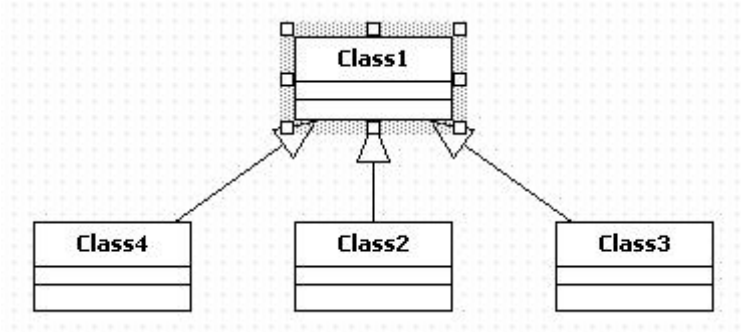


- Class를 상속하는 여러 개의 자식 Class 생성 방법:
- 현재 선택된 class로부터 한꺼번에 여러 개의 자식 Class를 생성하려면,

① class를 더블클릭해서 퀵다이얼로그가 나타나면 다음과 같이 "<=" 문자열 다음에 하위 class 이름을 입력한다. 각 class들은 ","로 구분한다.



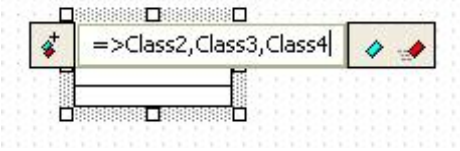
② 자식 class들이 선택된 class 아래에 생성되고 정렬된다.



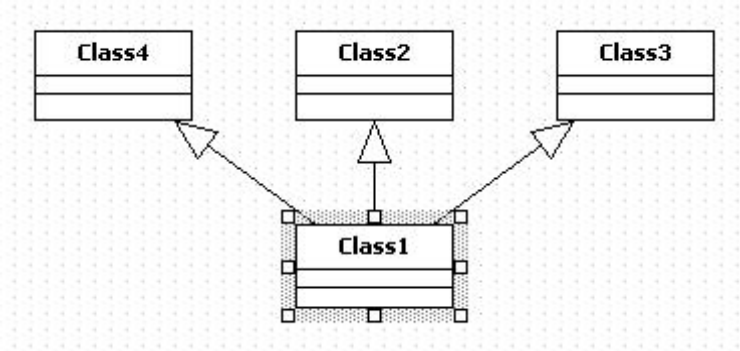
- 여러 개의 부모 Class를 한꺼번에 생성하는 방법:

한꺼번에 여러 개의 부모 Class를 생성하기 위해서는 단축생성 구문을 사용한다.

① 퀵다이얼로그에서 다음과 같이 입력한다.



② 그러면 선택된 class 위에 부모 class들이 생성되고 자동으로 정렬된다.



(12) Dependency

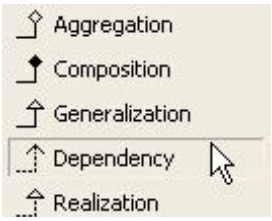
- 의미:

의존관계(Dependency)는 어떤 요소의 구현이나 기능을 위해 다른 요소의 존재가 요구 되어지는 의존적인 관계를 의미한다.

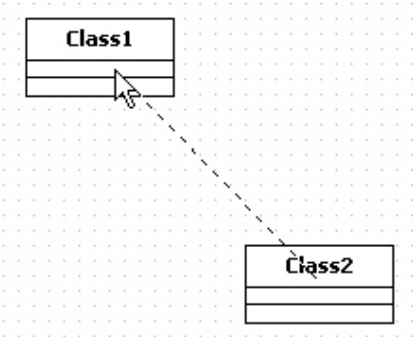
- Dependency 생성 방법:

- Dependency를 생성하려면,

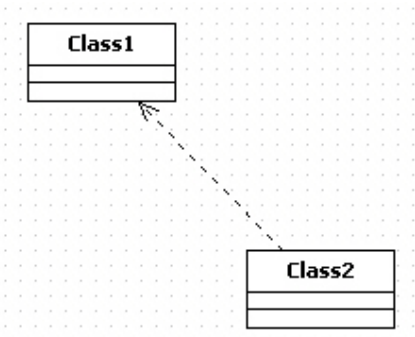
① [Toolbox] -> [Class] -> [Dependency] 버튼을 클릭하고



② Main 윈도우창에서 요소가 의존하는 다른 요소 방향으로 마우스를 누르고 드래그 하면 된다.



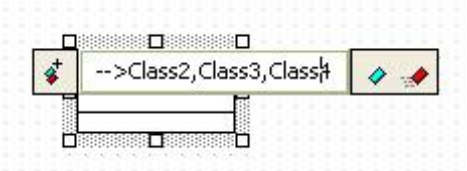
③ 그러면 두개의 요소사이에 dependency가 생성된다.



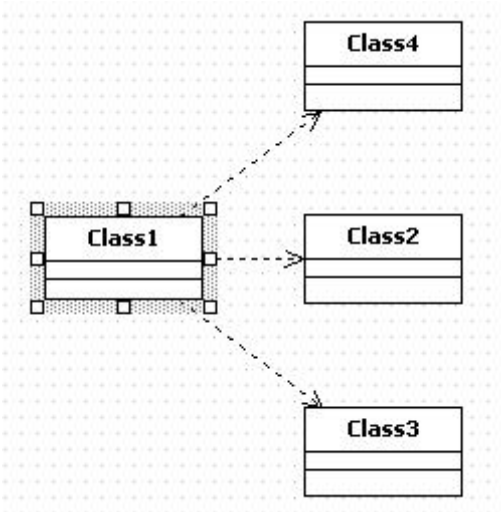
- 요소로부터 의존하는 다른 요소 생성 방법:

그리고 [Enter]키를 누르면 선택된 요소가 의존 관계를 가지는 여러 요소들이 생성되고 자동 배열되어 생성된다.

① 요소를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-->" 문자열 다음에 의존하는 다른 요소의 이름을 입력한다. 여러 개의 요소에 대해서 의존한다면 각 요소 이름은 "," 문자로 구분해서 입력한다.



② 현재 선택된 요소로부터 의존 관계를 갖는 다른 요소를 만들려면 단축 생성 구문을 사용한다.



(13) Realization

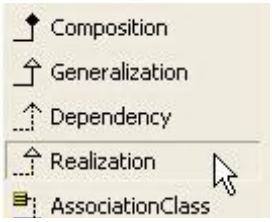
- 의미:

실체화(Realization)는 명세(specification) 요소와 그것을 구현하는(implementation) 요소의 실체화 관계를 정의한다. 주로 인터페이스(Interface)와 그것을 구현하는 요소(클래스, 컴포넌트 등)를 연결하는데 사용한다.

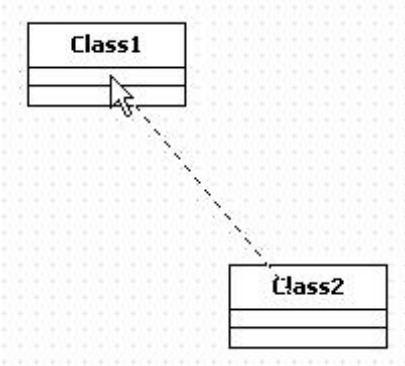
- Realization 생성 방법:

- Realization을 생성하려면,

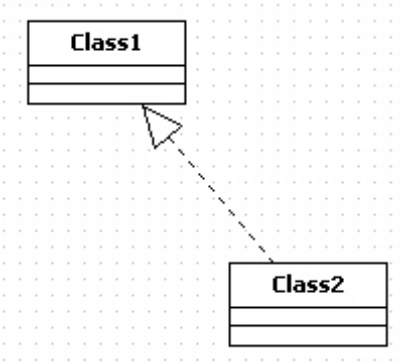
① [Toolbox] -> [Class] -> [Realization] 버튼을 클릭하고



② Main 윈도우창에서 요소가 Realize의 부모 요소 방향으로 마우스를 누르고 드래그하면 된다.



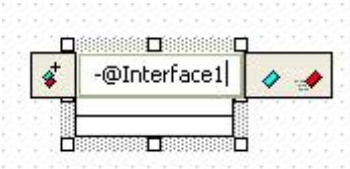
③ 결과는 다음과 같다.



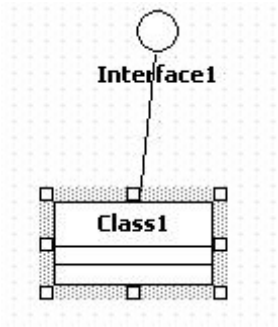
- 요소로부터 실체화할 다른 요소 생성 방법:

현재 선택된 요소로부터 Realization 관계를 갖는 다른 요소를 만들려면 단축 생성 구문을 사용한다.

① 요소를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-@" 문자열 다음에 Realization 대상이 되는 요소의 이름을 입력한다. 여러 개의 요소에 대해서 의존한다면 각 요소 이름은 "," 문자로 구분해서 입력한다.



② 그리고 [Enter]키를 누르면 선택된 요소가 Realization 관계를 가지는 여러 요소들이 생성되고 자동 배열되어 생성된다.



(14) AssociationClass

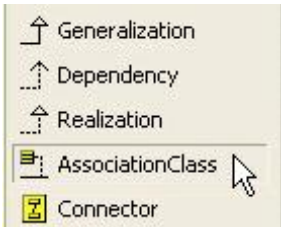
- 의미:

연관클래스(AssociationClass)는 클래스(Class)와 연관(Association)을 연결하여 연관 자체가 클래스의 의미도 가질 수 있도록 하는 연결 고리 역할을 한다.

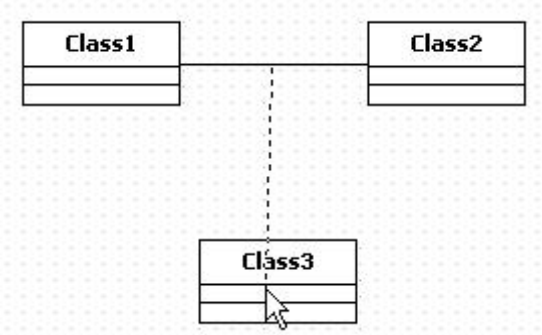
- AssociationClass 생성 방법:

- AssociationClass를 생성하려면,

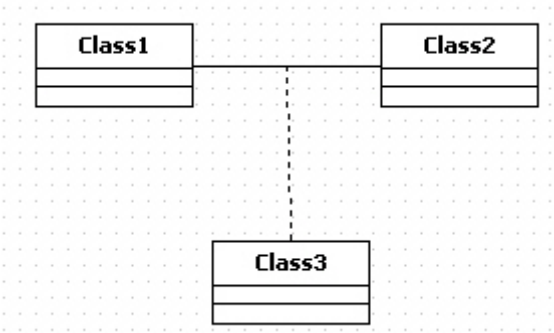
① [Toolbox] -> [Class] -> [AssociationClass] 버튼을 클릭하고



② Main 윈도우에서 Association에서 AssociationClass가 되는 Class 방향으로 마우스를 누르고 드래그하면 된다.



③ 결과는 다음과 같다.



(15) Object

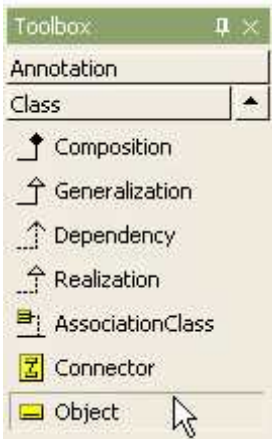
- 의미:

객체(Object)는 특정 클래스의 인스턴스(instance) 이다.

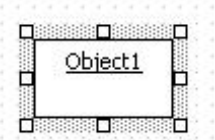
- Object 생성 방법:

- Object를 생성하려면,

① [Toolbox] -> [Class] -> [Object] 버튼을 클릭하고



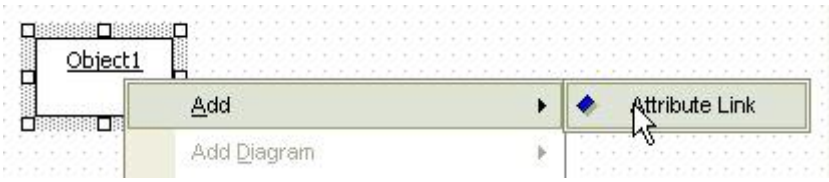
② Main 윈도우창에서 Object가 위치할 곳을 클릭한다.



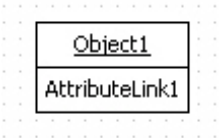
- Object에 AttributeLink 추가 방법:

Object에 AttributeLink를 추가하는 방법은 다음과 같이 2가지 방법이 있다.

Object 또는 Model Explorer의 팝업 메뉴 이용하는 경우에는 Main 윈도우 또는 Model Explorer에서 Object를 선택하고 오른쪽 마우스 버튼을 눌러서 [Add] -> [Attribute Link] 팝업 메뉴를 선택하여 Attribute Link를 추가할 수 있다.



Collection Editor 이용하는 경우에는 Object의 [Collection Editor...] 팝업 메뉴 또는 Properties 윈도우의 Slots의 편집 버튼을 통해서 Collection Editor를 열어서 Slots 탭에서  버튼을 이용하여 Attribute Link를 입력할 수 있다.



(16) Link

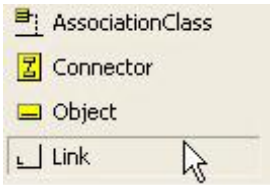
- 의미:

링크(Link)는 객체사이의 연결(connection)이다.

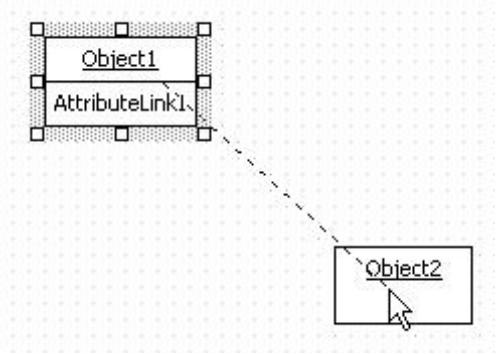
- Link 생성 방법:

- Link를 생성하려면,

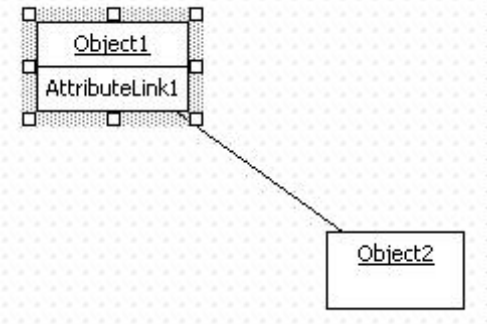
① [Toolbox] -> [Class] -> [Link] 버튼을 클릭하고



② Main 윈도우창에서 Link할 Object에서 다른 Object 방향으로 마우스를 누르고 드래그하면 된다.



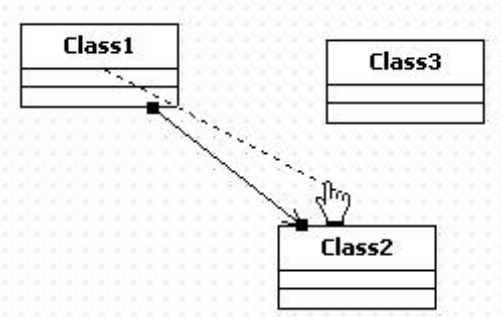
③ 결과는 다음과 같다.



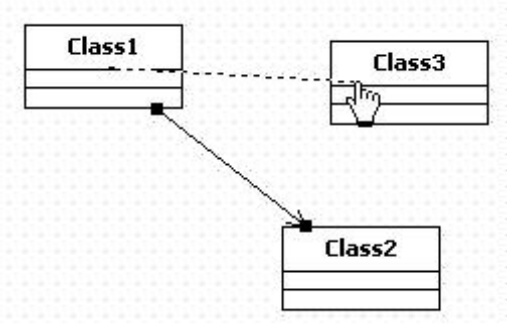
(17) Relationship

- 연결 끝을 변경하는 방법:
- 관계들이 연결된 요소를 다른 요소로 연결하려면

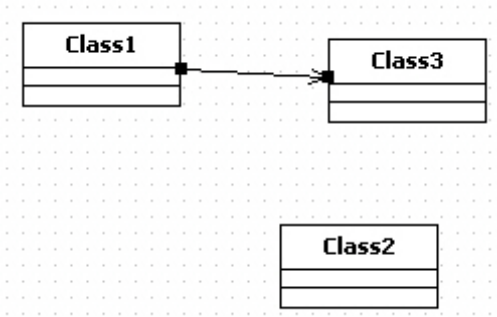
① 관계의 끝점을 드래그 한다.



② 그리고 연결하려는 다른 요소로 드롭 한다.



③ 그러면 연결의 끝이 변경된다.



다. Sequence 다이어그램 모델링하기

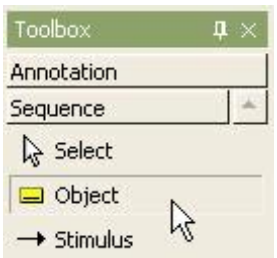
시퀀스 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Object
- Stimulus
- SelfStimulus
- Combined Fragment
- Interaction Operand
- FrameSubsytem

(1) Object

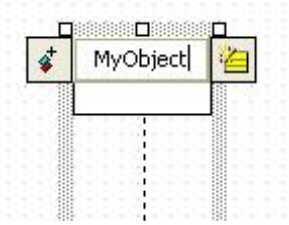
- Object 생성 방법:
- Object를 생성하려면,

① [Toolbox] -> [Sequence] -> [Object] 버튼을 클릭하고

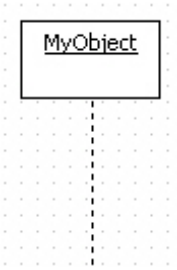


② Main 윈도우창에서 Object가 위치할 곳을 클릭한다.

③ 퀵다이얼로그에 object의 이름을 입력한다.



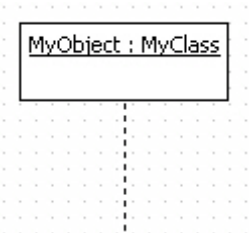
④ 그리고 [Enter] 키를 누른다.



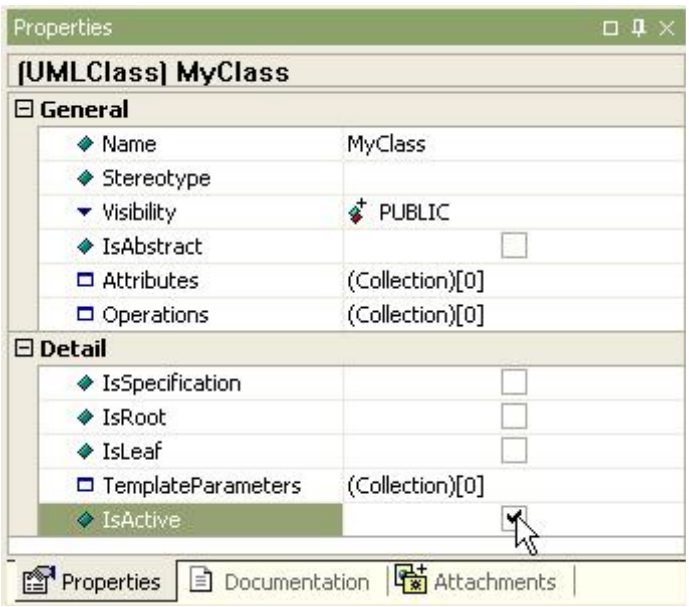
- Active Object 설정 방법:

- Object를 Active Object로 변경하려면,

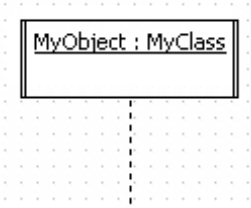
① 할당된 Class의 IsActivie 속성을 true로 변경하면 된다.



② 위의 예에서는 MyClass의 IsActive 속성을 변경한다.



③ 결과는 다음과 같다.

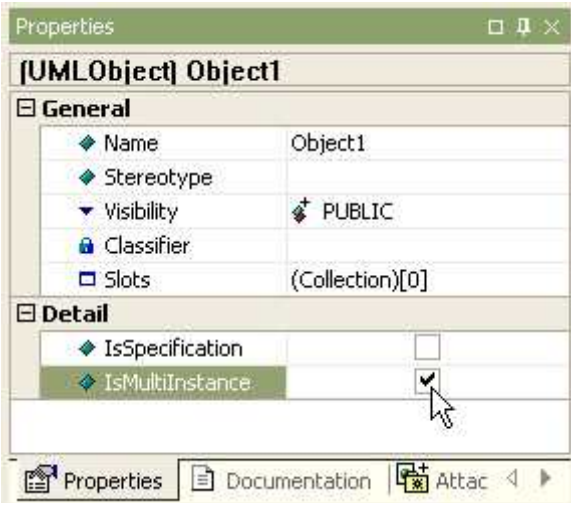


④ 만약 Class가 할당되어 있지 않다면 Active Object로 변경할 수 없다.

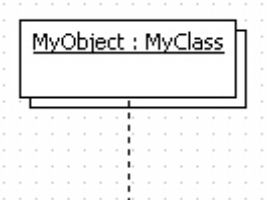
- Multi Object 설정 방법:

Object를 Multi Object로 설정하려면,

① Properties창에서 IsMultiInstance 속성을 true로 설정한다.



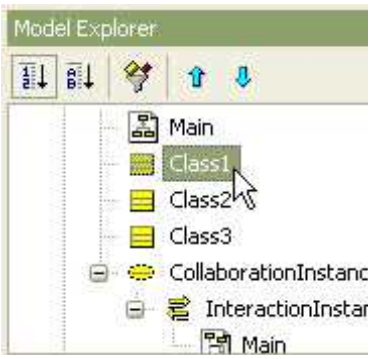
② 그러면 object는 multi object로 보인다.



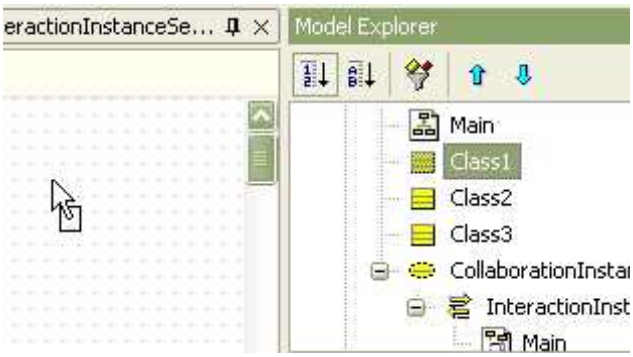
- Class로부터 Object 생성 방법:

- class로부터 object를 생성하기 위해서는

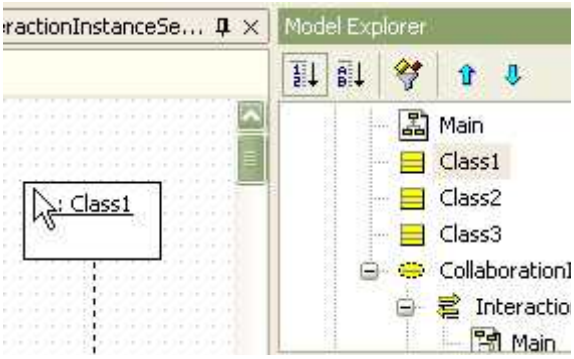
① model explorer에서 class를 선택한다.



② 선택된 class를 main window로 드래그/드롭 한다.

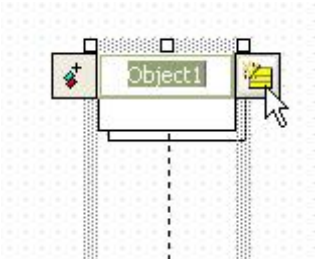


③ 그러면 object가 다이어그램 상에 생성된다.



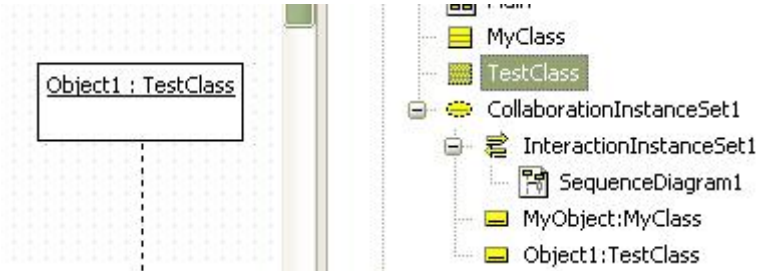
- Object로부터 Class 생성 방법:
- Object에 대해서 Class가 할당되어 있지 않다면

① Object를 더블 클릭해서 Quick Dialog의 Add Class 버튼을 이용하여 새로운 Class를 생성하고 Object에 할당할 수 있다.

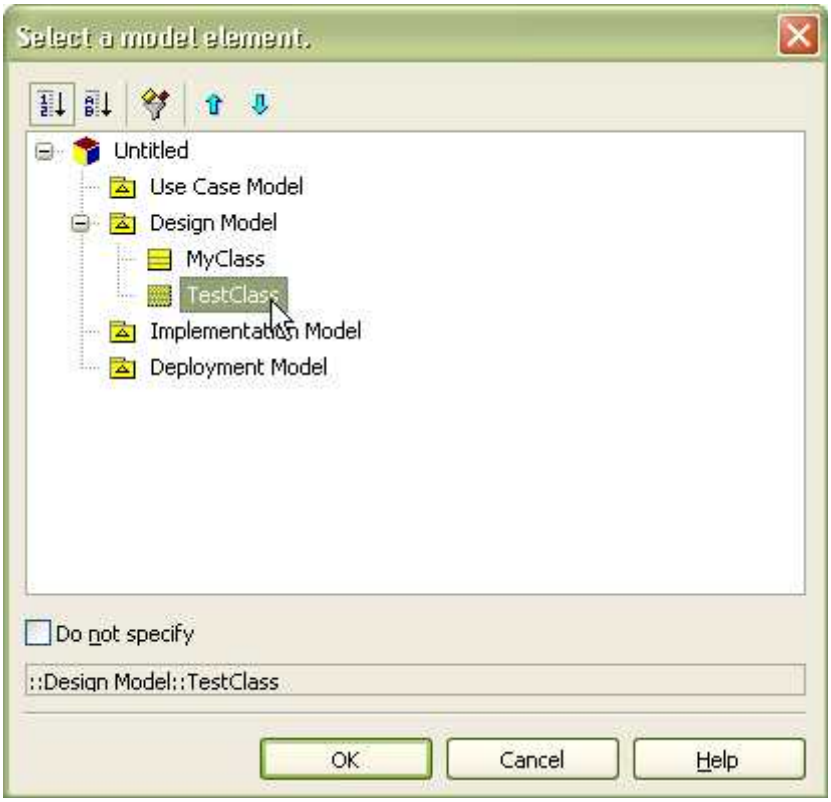


② [Enter element name] 다이얼로그에서 새로운 class의 이름을 입력한다.

③ 그러면 새로운 class가 생성되고 object에 할당된다.



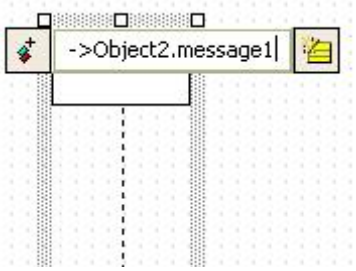
④ 만약 존재하는 class를 object에 할당하려면, object의 classifier 속성에서 ... 버튼을 클릭한다. 그리고 [Select a model element] 다이얼로그에서 할당할 class를 선택한다.



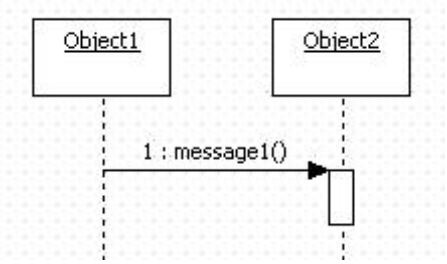
- Shortcut Create Syntax로 객체에서 나가는 Stimulus 생성 방법:
- 현재 선택된 객체에서 다른 객체로 나가는 stimulus를 생성하려면

① 객체를 더블 클릭하고나 객체가 선택된 상태에서 [Enter]키를 누른다.

② 그리고 Quick Dialog가 나타나면, "->" 문자열(들어오는 Stimulus는 "<-" 문자열, 나가고 리턴을 가지는 Stimulus는 "<->") 다음에 대상 객체의 이름과 Stimulus 이름을 입력한다.



③ 그리고 [Enter]키를 누르면 선택된 객체에서 지정한 객체로 나가는 Stimulus를 생성하고 현재 다이어그램에서 가장 마지막 순서로 배치한다.



(2) Stimulus

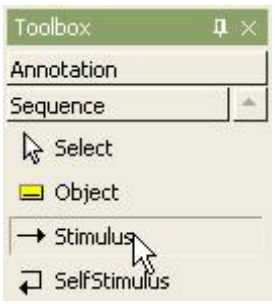
- 의미:

자극(Stimulus)은 두 인스턴스(Instance)간의 커뮤니케이션을 나타낸다.

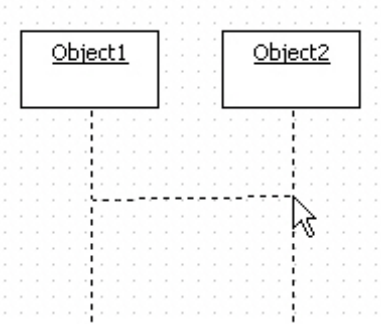
- Stimulus 생성 방법:

- Stimulus를 생성하려면,

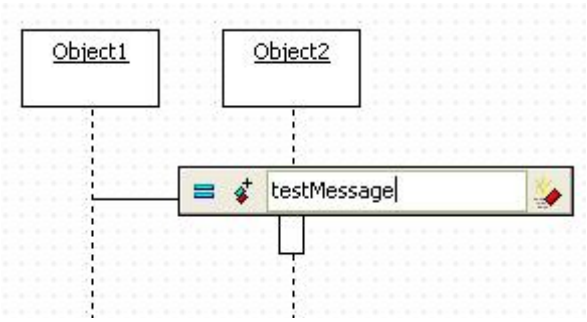
① [Toolbox] -> [Sequence] -> [Stimulus] 버튼을 클릭하고



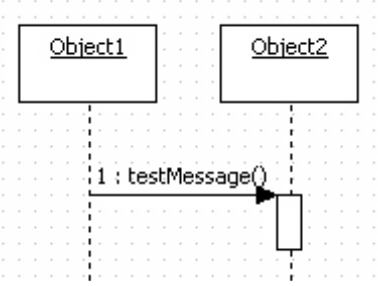
② Main 윈도우창에서 자극을 주는 방향으로 Object(또는 Lifeline)에서 다른 Object(또는 Lifeline)로 마우스를 누르고 드래그하면 된다.



③ Stimulus의 퀵다이얼로그가 나타나면 stimulus의 이름을 입력하고 [Enter] 키를 누른다.



④ 그러면 다음과 같이 stimulus가 생성된다.

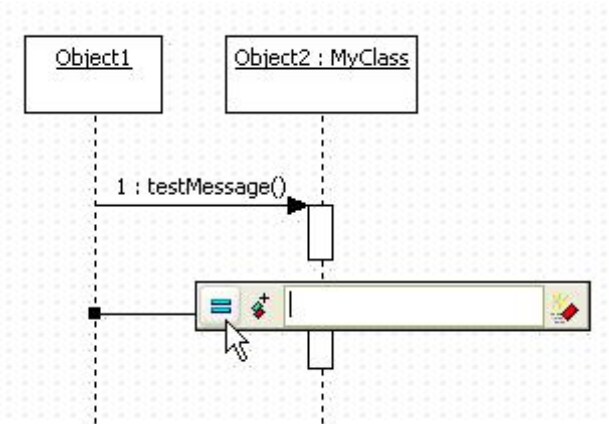


- Class에서 정의된 Operation을 stimulus로 사용 방법:

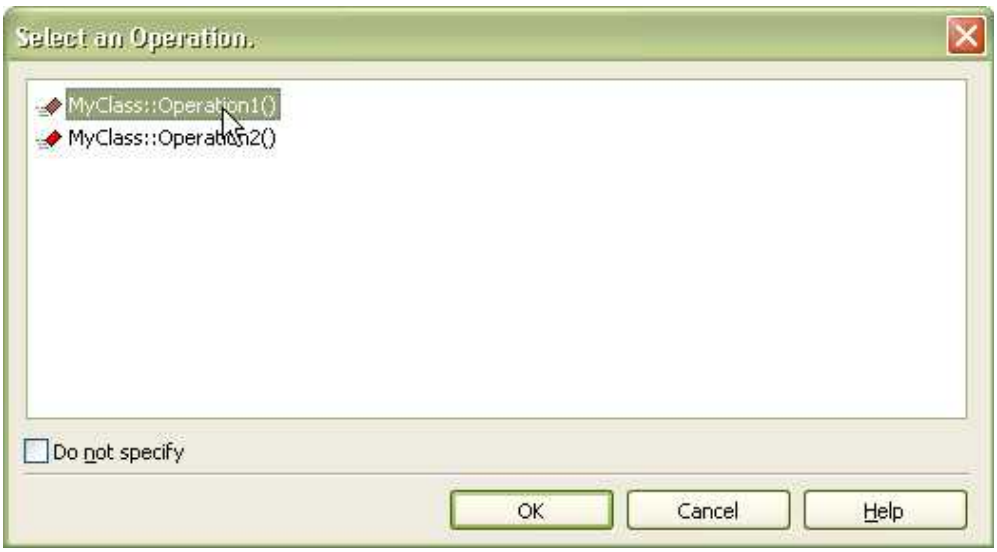
Stimulus의 Receiver측의 Class가 할당된 경우에 Receiver의 Operation을 Stimulus로 할당하려면,

① stimulus를 더블 클릭한다.

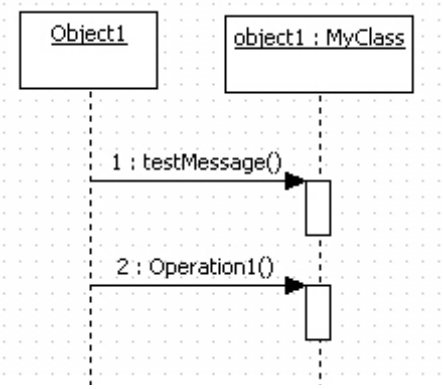
② 퀵다이얼로그에서  버튼을 클릭한다.



③ [Select an operation] 다이얼로그가 나타나면 할당할 operation을 선택하고 [OK] 버튼을 누른다.



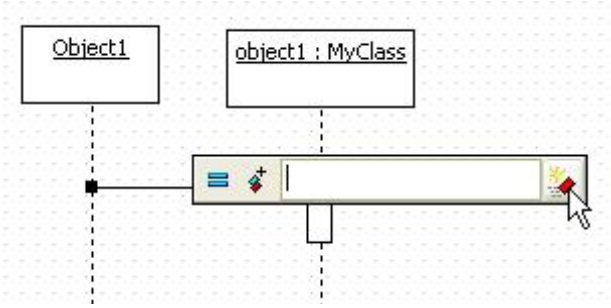
④ 선택된 operation에 매핑된 새로운 stimulus가 다음과 같이 추가된다.



- Object에서 Class의 Opeation 생성 방법:

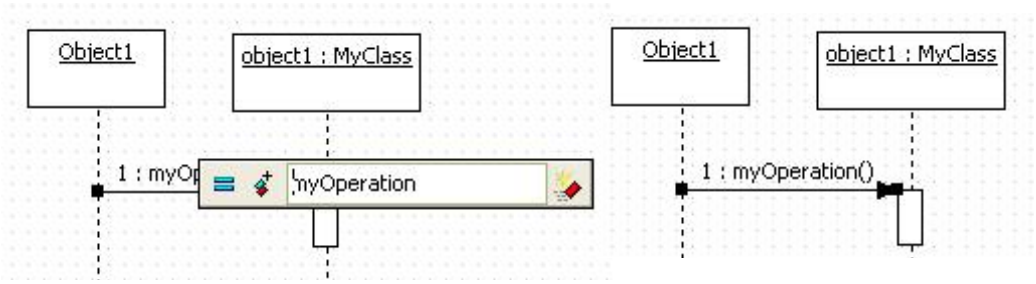
Stimulus의 Receiver측 Class에 포함될 Operation을 생성해서 Stimulus로 할당하려면,

① stimulus를 더블클릭한다. 그리고 퀵다이얼로그에서  버튼을 클릭한다.

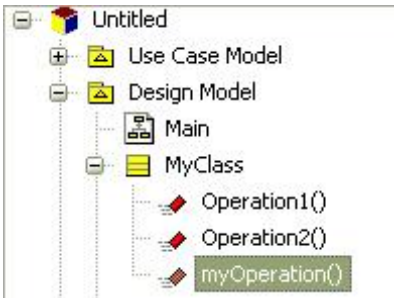


② 그리고 새로운 operation의 이름을 입력하고 [OK] 버튼을 클릭한다.

③ 그러면 새로운 operation이 추가된다.



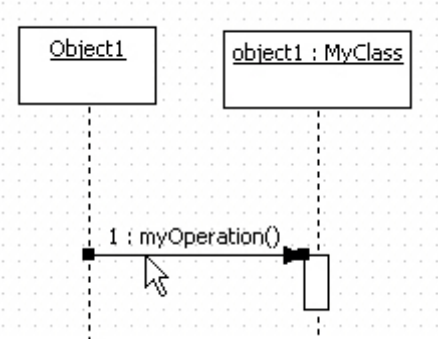
④ 새로운 operation이 추가되었는지 model explorer에서 확인한다.



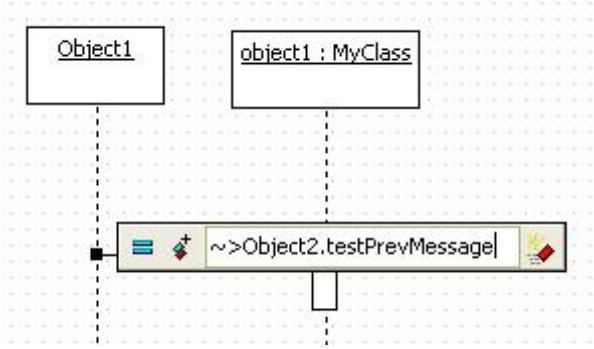
- Shortcut Create Syntax로 앞의 순서를 갖는 Stimulus 생성 방법:

- 현재 선택된 Stimulus 앞의 순서를 가지는 다른 Stimulus를 생성하려면

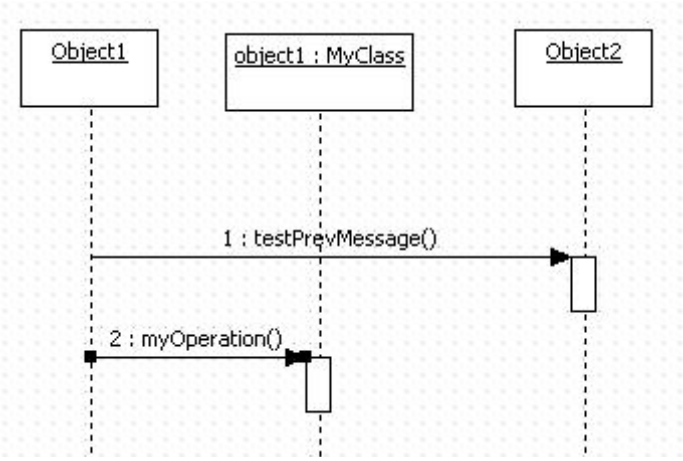
① Stimulus를 더블 클릭하거나 Stimulus가 선택된 상태에서 [Enter]키를 누른다.



② 그리고 Quick Dialog가 나타나면, "~>" 문자열(들어오는 Stimulu는 "<~" 문자열) 다음에 대상 객체의 이름과 Stimulus 이름을 입력한다.

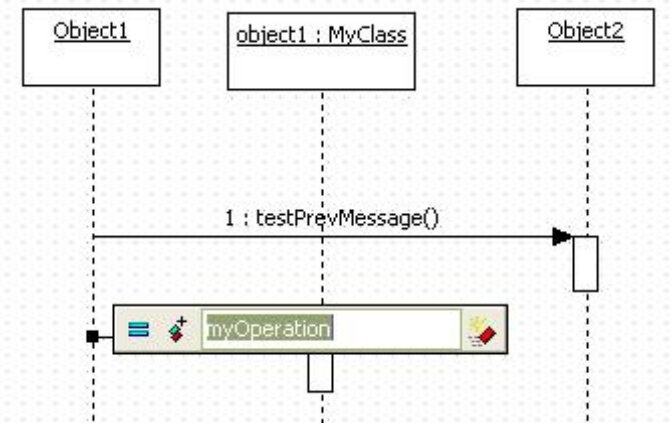


③ 그리고 [Enter]키를 누르면 지정한 객체로 나가는 새로운 Stimulus가 생성되고, 선택된 Stimulus 위에 배치된다.

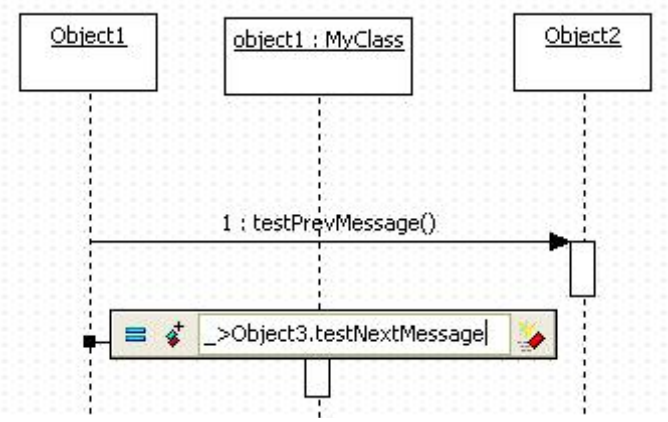


- Shortcut Create Syntax로 뒤의 순서를 갖는 Stimulus 생성 방법:
- 현재 선택된 Stimulus 뒤의 순서를 가지는 다른 Stimulus를 생성하려면

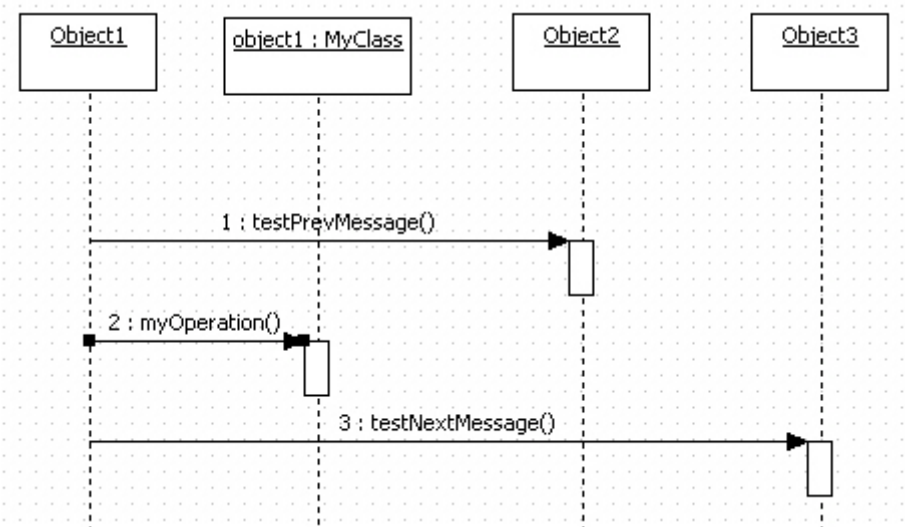
① Stimulus를 더블 클릭하거나 Stimulus가 선택된 상태에서 [Enter]키를 누른다.



② 그리고 Quick Dialog가 나타나면, "_>" 문자열(들어오는 Stimulu는 "<_" 문자열) 다음에 대상 객체의 이름과 Stimulus 이름을 입력한다.

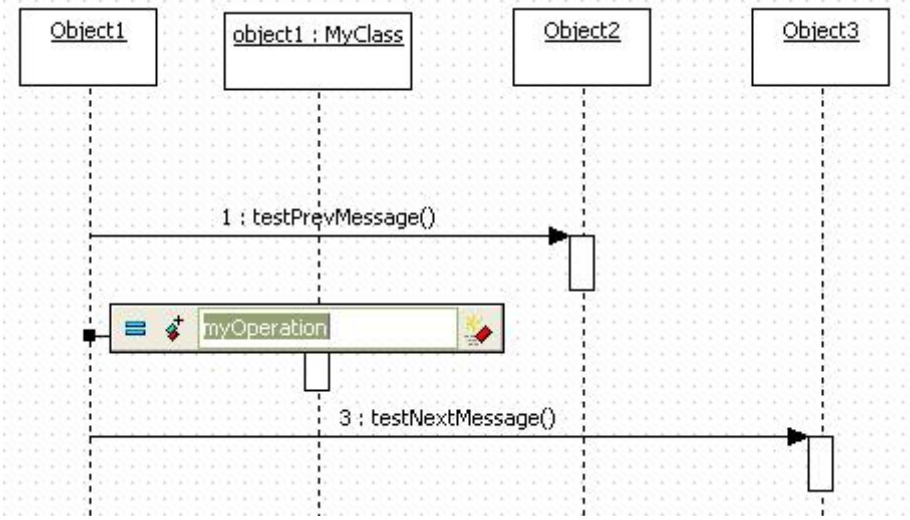


③ 그리고 [Enter]키를 누르면 지정한 객체로 나가는 새로운 Stimulus가 생성되고, 선택된 Stimulus 밑에 배치된다.

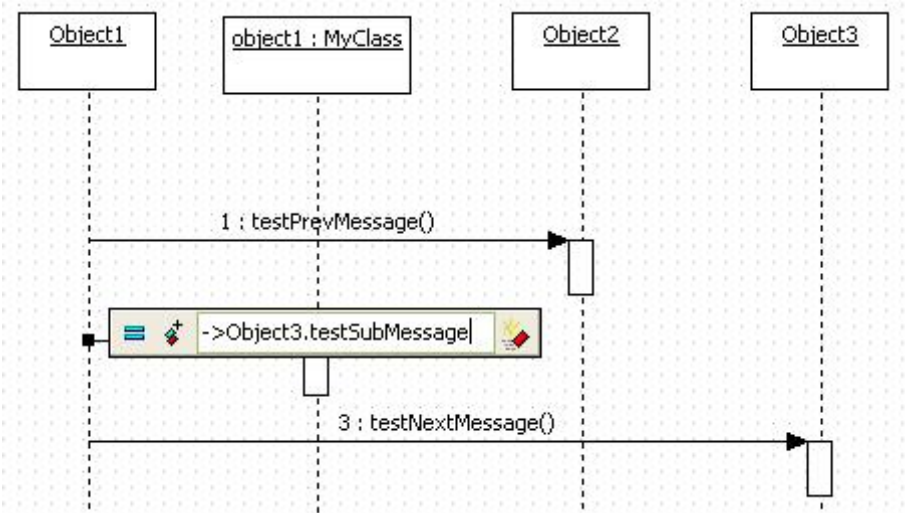


- Shortcut Create Syntax로 Sub Stimulus 생성 방법:
- 현재 선택된 Stimulus의 Activation에 포함되는 다른 Stimulus를 생성하려면

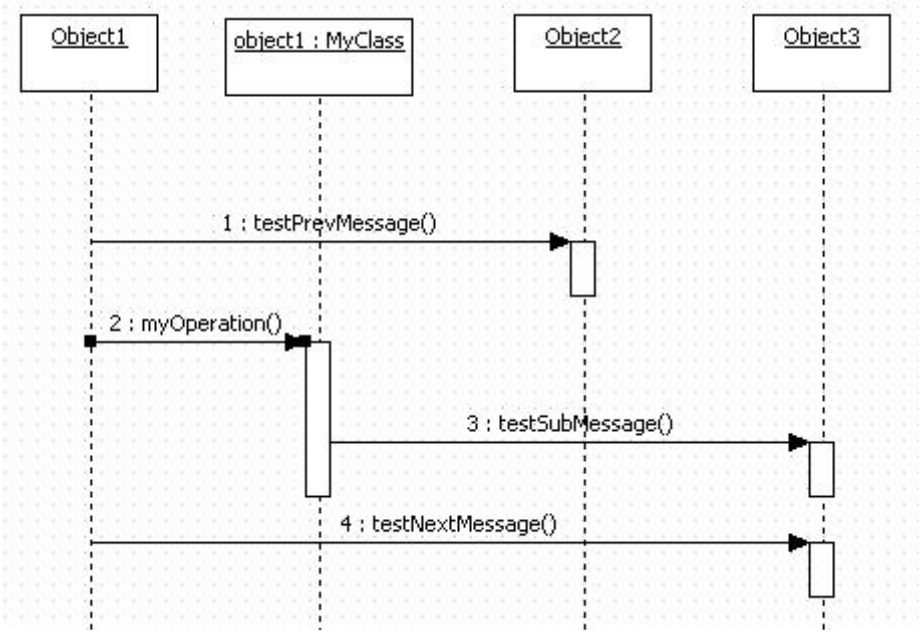
① Stimulus를 더블 클릭하거나 Stimulus가 선택된 상태에서 [Enter]키를 누른다.



② 그리고 Quick Dialog가 나타나면, "->" 문자열(들어오는 Stimulu는 "<-" 문자열) 다음에 대상 객체의 이름과 Stimulus 이름을 입력한다.

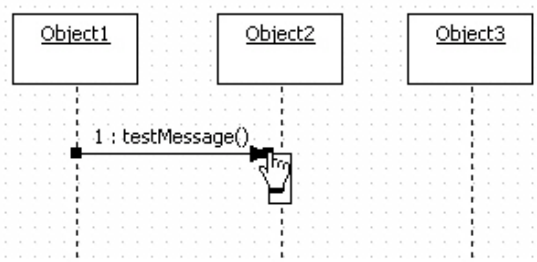


③ 그리고 [Enter]키를 누르면 지정한 객체로 나가는 새로운 Stimulus가 생성되고, 선택된 Stimulus의 Activation의 맨 마지막 Stimulus 위치로 배치된다.

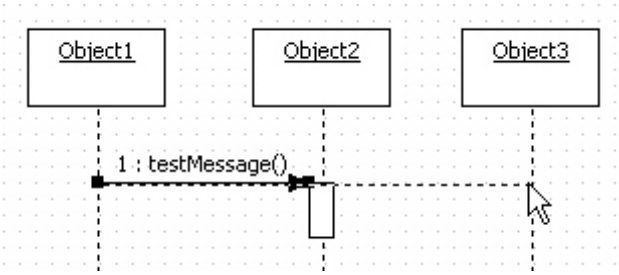


- Stimulus 연결 재설정 방법:
- Stimulus의 양쪽 끝을 다른 요소로 연결하려면,

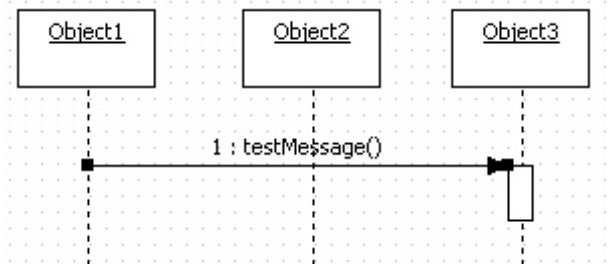
① stimulus의 끝을 클릭한다.



② 그리고 연결할 또 다른 object로 드래그/드롭 한다.

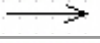
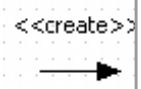


③ 그러면 다음과 같이 stimulus가 다른 object에 연결된다.



- Stimulus의 ActionKind 변경 방법:

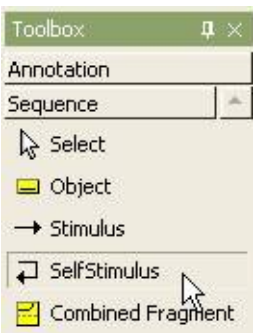
Stimulus의 ActionKind 속성 값은 5가지 종류 중에 하나로 설정되어야 하며, ActionKind의 종류에 따라서 다음과 같이 다르게 보이다. ActionKind의 종류는 Properties에서 변경할 수 있다.

ActionKind	Shape
CALL	
SEND	
RETURN	
CREATE	
DESTROY	

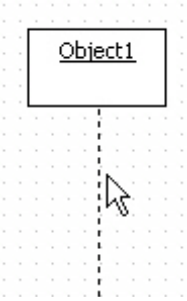
(3) SelfStimulus

- SlefStimulus 생성 방법:
- SelfStimulus를 생성하려면,

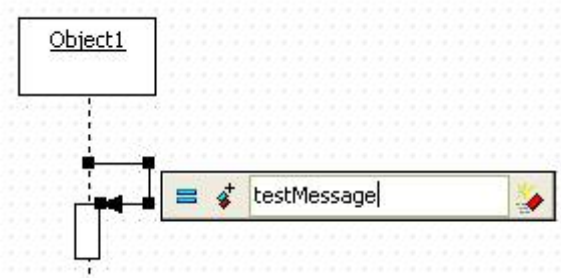
① [Toolbox] -> [Sequence] -> [SelfStimulus] 버튼을 클릭하고



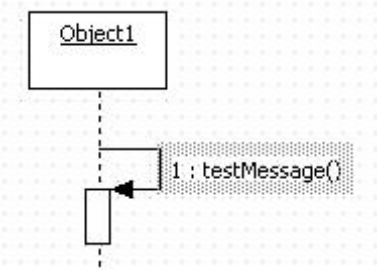
② Main 윈도우창에서 SelfStimulus를 삽입할 Object(또는 Lifeline)에서 마우스를 클릭한다.



③ Object의 퀵다이얼로그가 나타나면 stimulus의 이름을 입력하고 [Enter] 키를 누른다.



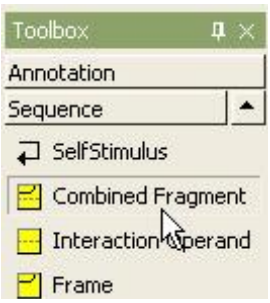
④ 다음과 같이 self-stimulus가 보인다.



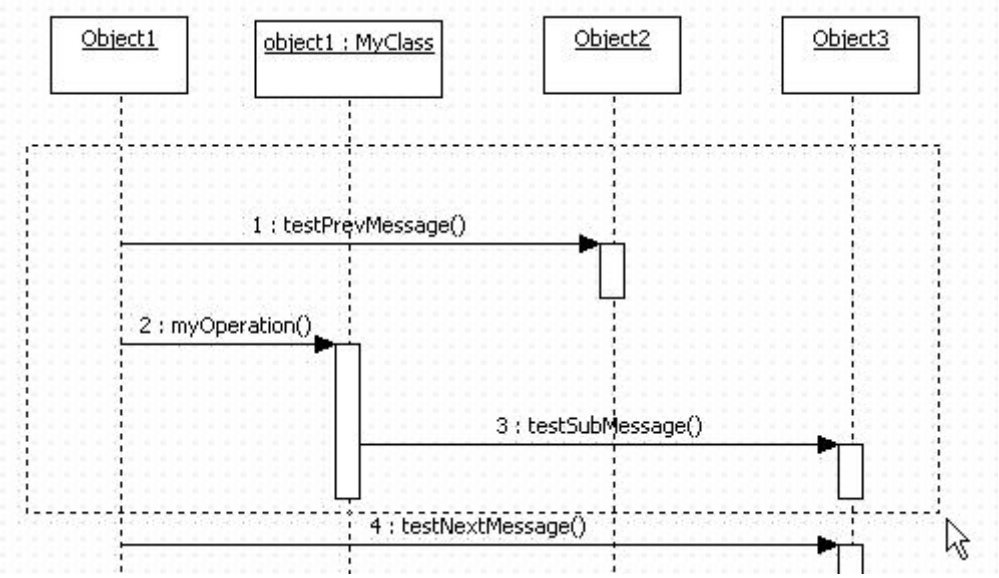
(4) Combined Fragment

- Combined Fragment 생성 방법:
- Combined Fragment를 생성하려면,

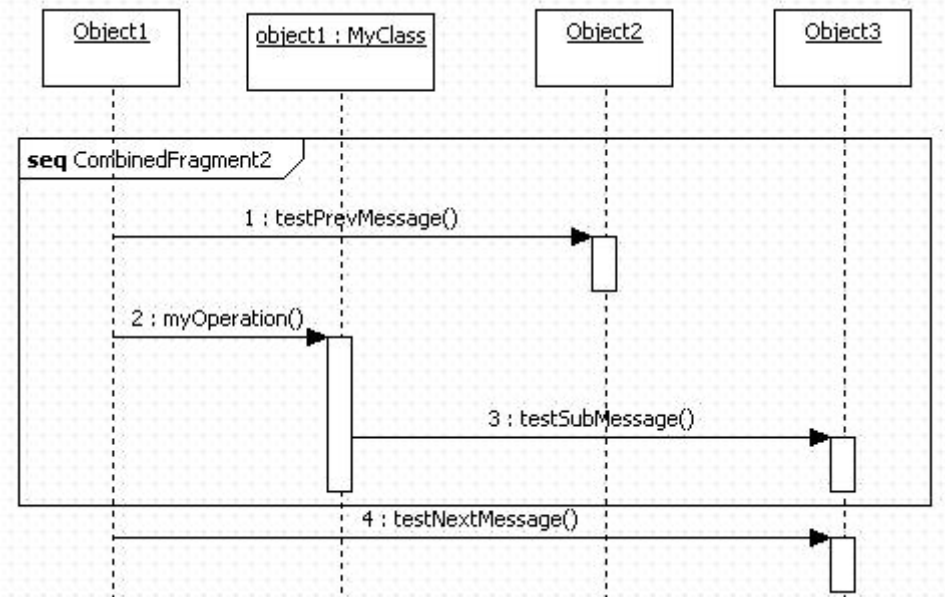
① [Toolbox] -> [Sequence] -> [Combined Fragment] 버튼을 클릭하고



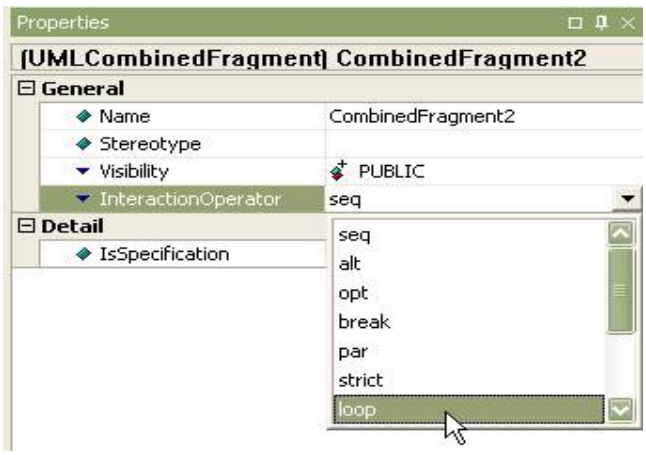
② Main 윈도우창에서 Combined Fragment가 위치할 곳을 클릭하고 원하는 범위만큼 드래그 한다.



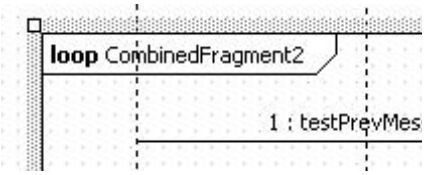
③ 그러면 combined fragment가 생성된다.



④ 그리고 interaction operator를 다음과 같이 변경한다.



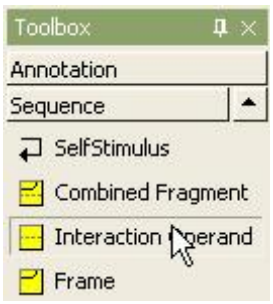
⑤ interaction operator 변경 후에 다음과 같이 combined fragment가 보인다.



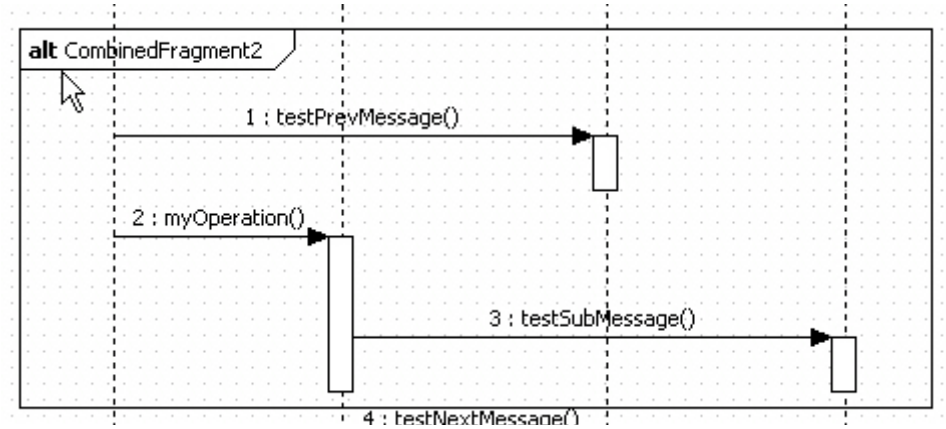
(5) Interaction Operand

- Interaction Operand 생성 방법:
- Interaction Operand를 생성하려면,

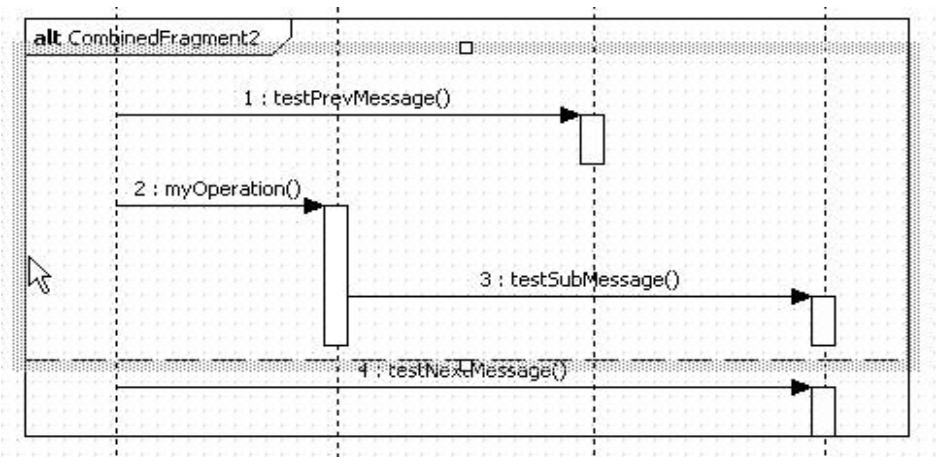
① [Toolbox] -> [Sequence] -> [Interaction Operand] 버튼을 클릭하고



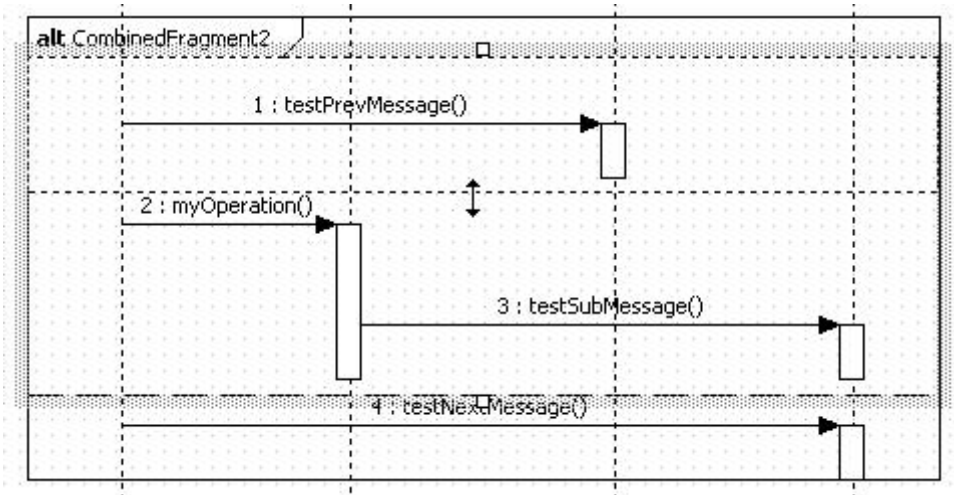
② Interaction Operand가 삽입될 Combined Fragment에 클릭한다.



③ 새로운 interaction operand가 combined fragment에 추가되면 interaction operand를 클릭한다.



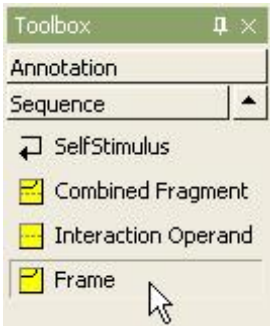
④ interaction operand의 크기 조절점이 보이면 드래그해서 범위를 정렬한다.



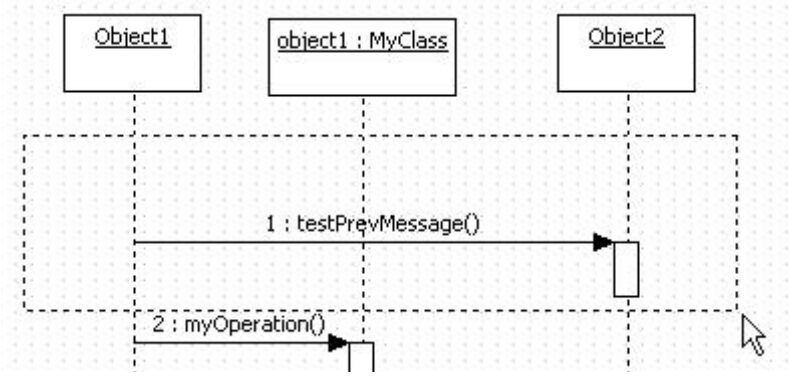
(6) FrameSubsystem

- Frame
- Frame 생성 방법:
- Frame을 생성하려면,

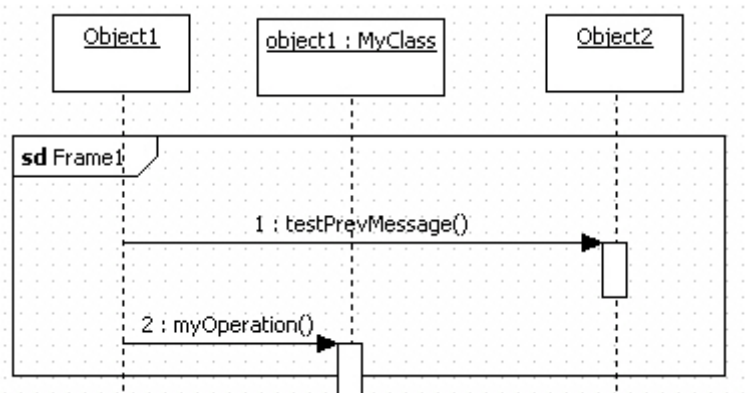
① [Toolbox] -> [Sequence] -> [Frame] 버튼을 클릭하고



② Main 윈도우창에서 Frame이 위치할 곳을 클릭하고 범위를 드래그 한다.

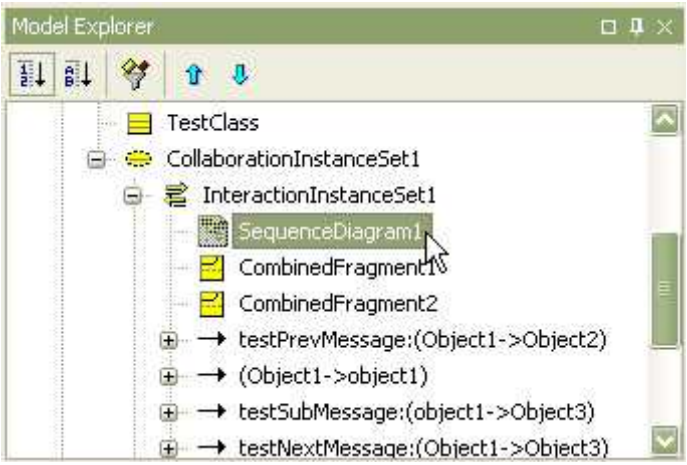


③ 그러면 새로운 frame이 다음과 같이 생성된다.

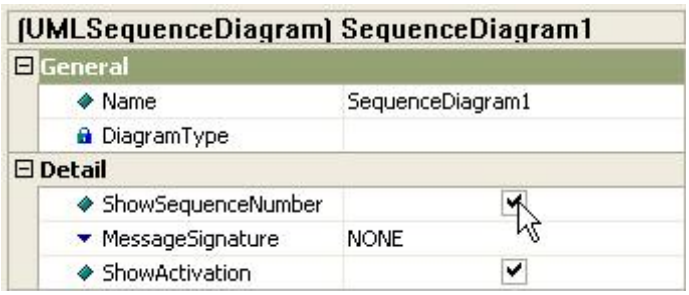


- Diagram
- 다이어그램에서 시퀀스 넘버 보이기:
- 다이어그램에서 Stimulus의 시퀀스 넘버를 나타내거나 감추려면

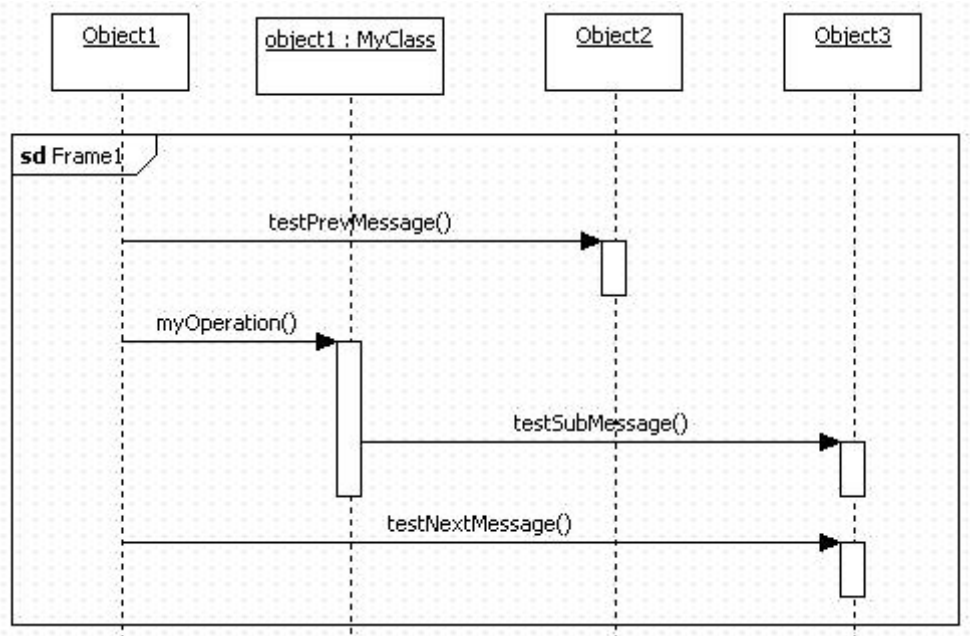
① model explorer 또는 main window에서 sequence diagram을 선택한다.



② diagram의 ShowSequenceNumber 속성 값을 true 또는 false로 변경한다.

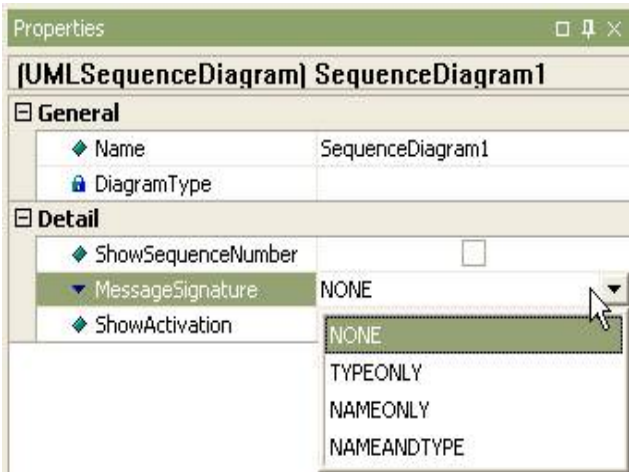


③ 다음은 ShowSequenceNumber 속성 값이 false 일 때의 sequence diagram이다.

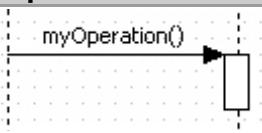
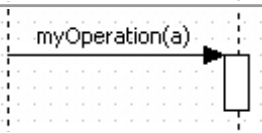


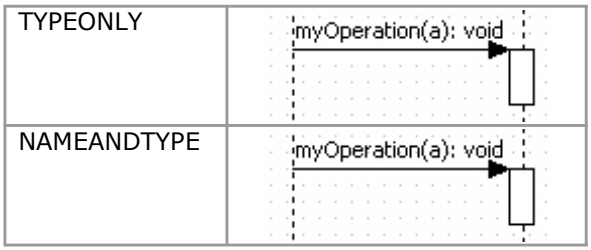
- 다이어그램에서 메시지의 시그너처 스타일 변경 방법:

다이어그램에서 Stimulus의 signature는 다음과 같이 4가지가 존재한다.



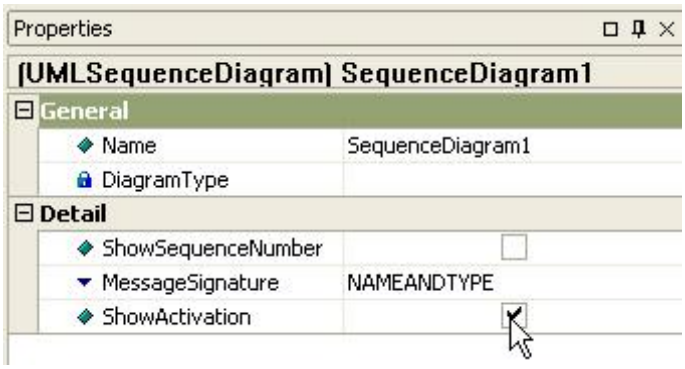
stimulus의 signature를 변경하려면 시Model Expolorer의 다이어그램을 선택하거나, Main 윈도우의 다이어그램을 선택하고 Message Signature 속성 값을 변경한다.

Style	Example
NONE	
NAMEONLY	



- 다이어그램의 모든 Activation 스타일 변경 방법:

다이어그램에서 Stimulus의 Activation을 나타내거나 감추려면 Model Expolorer의 다이어그램을 선택하거나, Main 윈도우의 다이어그램을 선택하고 ShowActivation 속성을 true 또는 false로 설정한다.



라. Activity 다이어그램 모델링

Activity 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- ActionState
- SubactivityState
- InitialState
- FinalState
- Synchronization
- Decision
- Flow Final
- Object Flow
- Signal Accept State
- Signal Send State
- Transition
- SelfTransition
- Swimlane

(1) ActionState

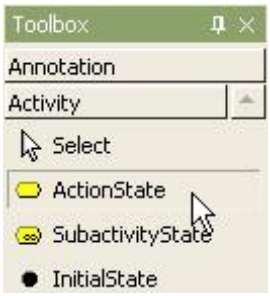
- 의미:

활동상태(ActionState)는 하나의 진입-액션(Entry Action)을 가지고 있는 상태(State)이다. 활동은 개념적으로 하나의 활동, 작업 등을 표현하며 활동 다이어그램에서 표현되어지는 요소이다.

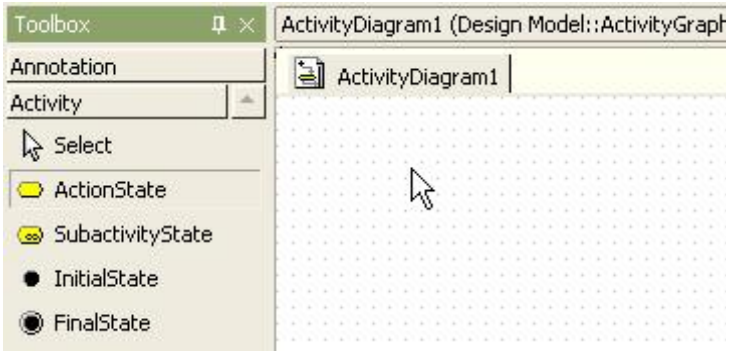
- 생성 방법:

- ActionState를 생성하려면,

① [Toolbox] -> [Activity] -> [ActionState] 버튼을 클릭하고



② Main 윈도우창에서 ActionState가 위치할 곳을 클릭한다.



③ action state가 생성되고 퀵다이얼로그가 나타난다.

④ action state의 이름을 퀵다이얼로그에서 입력하고 [Enter] 키를 누르면 다음과 같이 action state가 보인다.

(2) SubactivityState

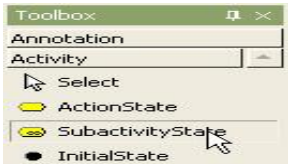
- 의미:

하위-활동상태(SubactivityState)는 하나의 액티비티그래프(ActivityGraph)를 호출한다. 하위-활동상태로 수행이 진입하면 해당 활동그래프(ActivityGraph)가 수행을 시작하고 수행이 종료되면 하위-활동상태의 수행이 종료된 것으로 간주된다.

- 생성 방법:

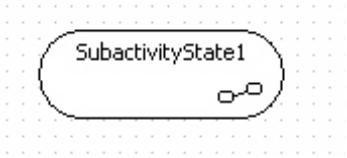
- SubactivityState를 생성하려면,

① [Toolbox] -> [Activity] -> [SubactivityState] 버튼을 클릭하고



② Main 윈도우창에서 SubactivityState가 위치할 곳을 클릭한다. subactivity state가 다이어그램 상에 생성되어지고 퀵다이얼로그가 나타나면 subactivity state

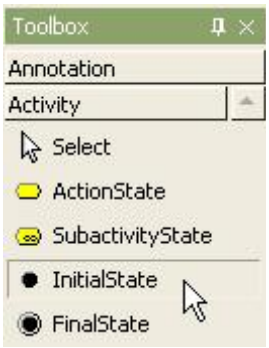
의 이름을 입력하고 [Enter] 키를 누른다.



(3) InitialState

- 생성 방법:
- InitialState를 생성하려면,

① [Toolbox] -> [Activity] -> [InitialState] 버튼을 클릭하고

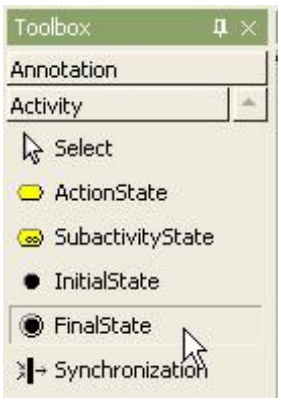


② Main 윈도우창에서 InitialState가 위치할 곳을 클릭한다.

(4) FinalState

- 생성 방법:
- FinalState를 생성하려면,

① [Toolbox] -> [Activity] -> [FinalState] 버튼을 클릭하고

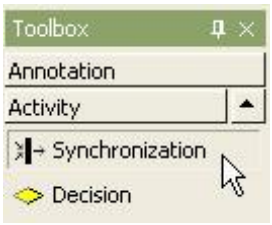


② Main 윈도우창에서 FinalState가 위치할 곳을 클릭한다.

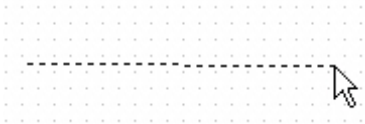
(5) Synchronization

- 생성 방법:
- Synchronization을 생성하려면,

① [Toolbox] -> [Activity] -> [Synchronization] 버튼을 클릭하고



② Main 윈도우창에서 Synchronization이 위치할 곳을 클릭하고 원하는 크기만큼 드래그 한다.



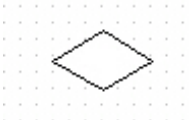
③ 그러면 다음과 같이 synchronization이 다이어그램에 생성된다.

(6) Decision

- 생성 방법:
- Decision을 생성하려면,

① [Toolbox] -> [Activity] -> [Decision] 버튼을 클릭하고

② Main 윈도우창에서 Decision이 위치할 곳을 클릭한다. 그러면 다음과 같이 decision이 다이어그램에 생성된다.

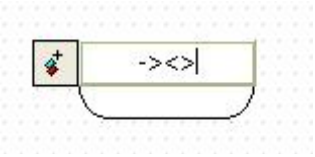


- State로부터 Decision 생성하기:

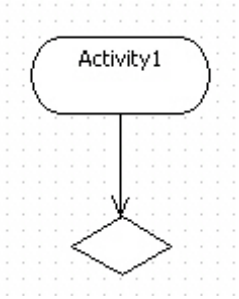
현재 State에서 나가는 transition과 Decision을 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-><>" 문자열

(Decision에서 State로 들어오는 경우는, "<-<>" 문자열)을 입력한다.



② 그리고 [Enter] 키를 누르면 선택된 State에서 나가는(들어오는) transition과 Decision이 생성된다.



(7) Flow Final

- 생성 방법:
- Flow Final을 생성하려면,

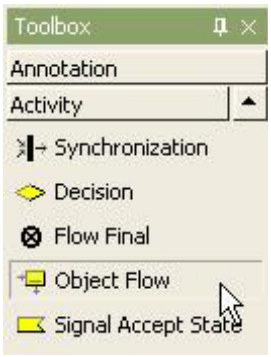
① [Toolbox] -> [Activity] -> [Flow Final] 버튼을 클릭하고

② Main 윈도우창에서 Flow Final이 위치할 곳을 클릭한다.

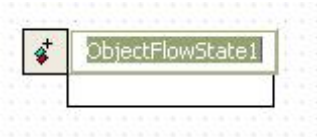
(8) Object Flow

- 생성 방법:
- Object Flow를 생성하려면,

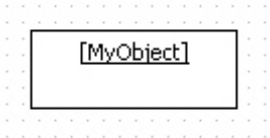
① [Toolbox] -> [Activity] -> [Object Flow] 버튼을 클릭하고



② Main 윈도우창에서 Object Flow가 위치할 곳을 클릭하면 다음과 같이 퀵다이얼로그가 나타난다.



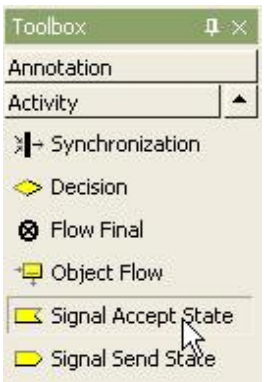
③ 퀵다이얼로그에서 object flow state의 이름을 입력하고 [Enter] 키를 누른다.



(9) Signal Accept State

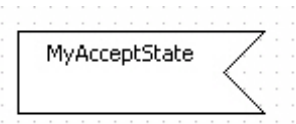
- 생성 방법:
- Signal Accept State를 생성하려면,

① [Toolbox] -> [Activity] -> [Signal Accept State] 버튼을 클릭하고



② Main 윈도우창에서 Signal Accept State가 위치할 곳을 클릭한다.

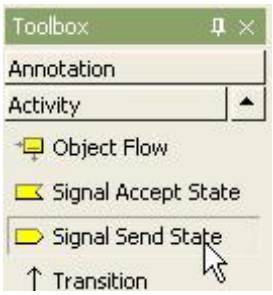
③ 퀵다이얼로그에서 signal accept state의 이름을 입력하고 [Enter] 키를 누른다.



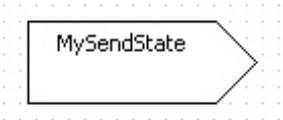
(10) Signal Send State

- 생성 방법:
- Signal Send State를 생성하려면,

① [Toolbox] -> [Activity] -> [Signal Send State] 버튼을 클릭하고



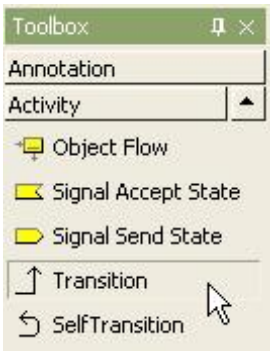
② Main 윈도우창에서 Signal Send State가 위치할 곳을 클릭하면 signal send state가 생기고 퀵다이얼로그가 나타난다. 퀵다이얼로그에서 signal send state의 이름을 입력하고 [Enter] 키를 누른다.



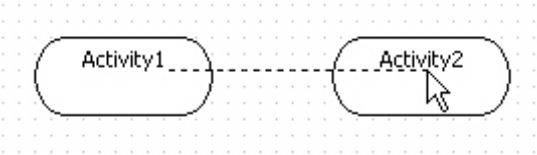
(11) Transition

- Transition 생성 방법:
- Transition을 생성하려면,

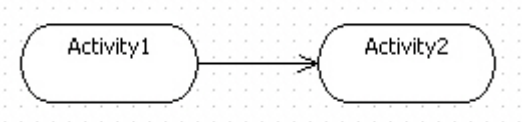
① [Toolbox] -> [Activity] -> [Transition] 버튼을 클릭하고



② Main 윈도우창에서 State에서 전이하는 방향의 다른 State로 마우스를 누르고 드래그하면 된다.



③ 그러면 transition이 생성된다.

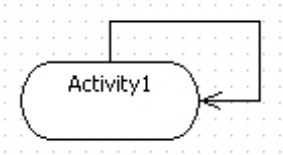


(12) SelfTransition

- SelfTransition 생성 방법:
- SelfTransition을 생성하려면,

① [Toolbox] -> [Activity] -> [SelfTransition] 버튼을 클릭하고

② Main 윈도우창에서 SelfTransition하는 State를 클릭하면 self-transition이 생성된다.

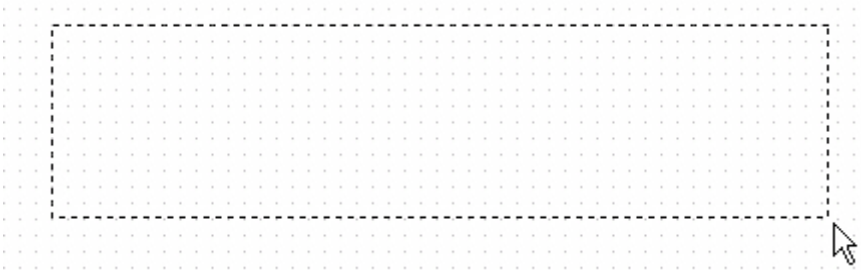


(13) Swimlane

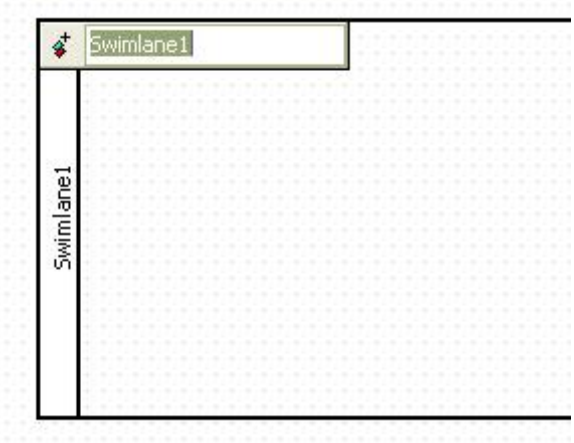
- Horizontal Swimlane 생성 방법:
- Horizontal Swimlane를 생성하려면,

① [Toolbox] -> [Activity] -> [Horizontal Swimlane] 버튼을 클릭하고

② Main 윈도우창에서 Horizontal Swimlane가 위치할 곳을 클릭하고 원하는 크기만큼 드래그 한다.



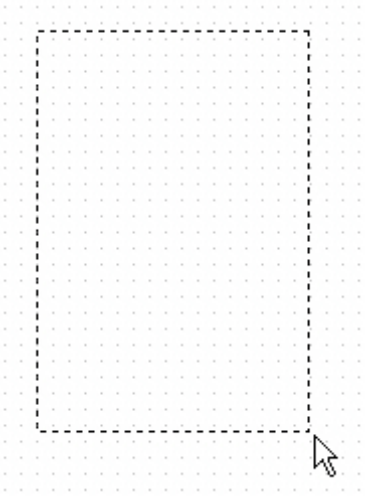
③ 그러면 horizontal swimlane이 생성되고 퀵다이얼로그가 나타난다. 퀵다이얼로그에서 swinlane의 이름을 입력하고 [Enter]키를 누른다.



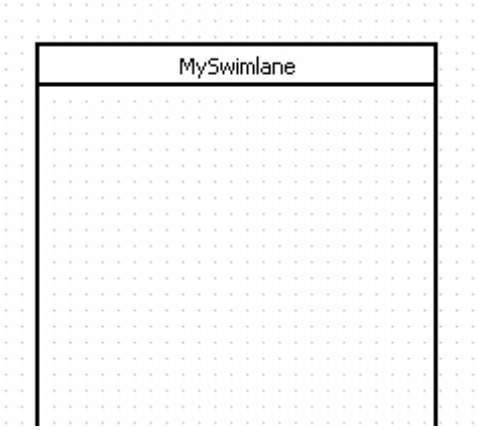
- Vertical Swimlane 생성 방법:
- Vertical Swimlane를 생성하려면,

① [Toolbox] -> [Activity] -> [Vertical Swimlane] 버튼을 클릭하고

② Main 윈도우창에서 Vertical Swimlane가 위치할 곳을 클릭하고 원하는 크기만큼 드래그 한다.



③ 그러면 vertical swimlane이 생성되고 퀵다이얼로그가 나타난다. 퀵다이얼로그에서 swinlane의 이름을 입력하고 [Enter]키를 누른다.



라. Collaboration 다이어그램 모델링하기

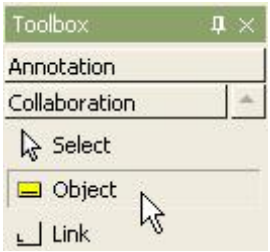
Collaboration 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Object
- Link
- SelfLink
- Stimulus
- Frame

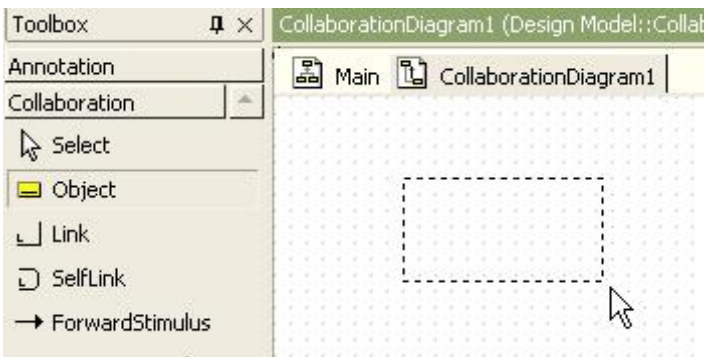
(1) Object

- 생성 방법:
- Object를 생성하려면,

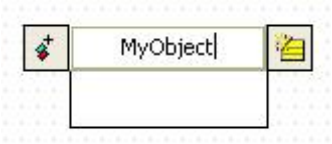
① [Toolbox] -> [Collaboration] -> [Object] 버튼을 클릭하고



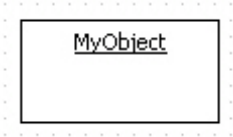
② Main 윈도우창에서 Object가 위치할 곳을 클릭한다.



③ 그러면 퀵다이얼로그가 보이고, 거기에 object의 이름을 입력한다.



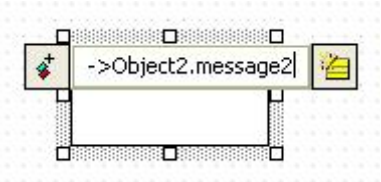
④ 그리고 [Enter] 키를 누른다.



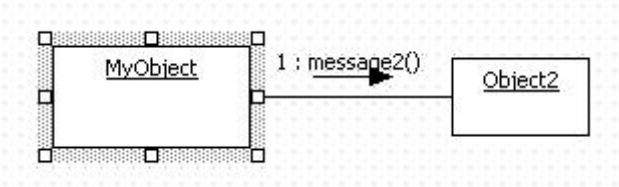
- Shortcut Create Syntax로 객체에서 나가는 Stimulus 생성 방법:
- 현재 선택된 객체에서 다른 객체로 나가는 stimulus를 생성하려면

① 객체를 더블 클릭하고나 객체가 선택된 상태에서 [Enter]키를 누른다.

② 그리고 Quick Dialog가 나타나면, "->" 문자열(들어오는 Stimulus는 "<-" 문자열) 다음에 대상 객체의 이름과 Stimulus 이름을 입력한다.

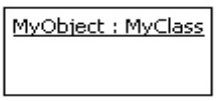


③ 그리고 [Enter]키를 누르면 선택된 객체에서 지정한 객체로 나가는 Stimulus를 생성하고 현재 다이어그램에서 가장 마지막 순서로 설정한다.

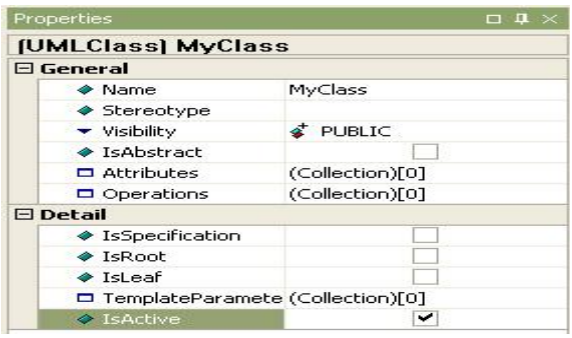


- Active Object 설정 방법:
- Object를 Active Object로 변경하려면,

① 할당된 Class의 IsActivie 속성을 true로 변경하면 된다.



② 위의 경우는, MyClass의 IsActive 속성을 변경한다.

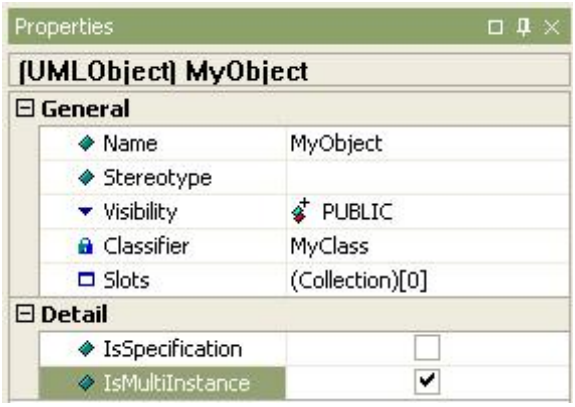


③ 만약 Class가 할당되어 있지 않다면 Active Object로 변경할 수 없다.

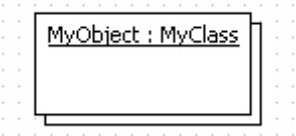
- Multi Object 설정 방법:

- Object를 Multi Object로 설정하려면,

① Properties창에서 IsMultiInstance 속성을 true로 설정한다.



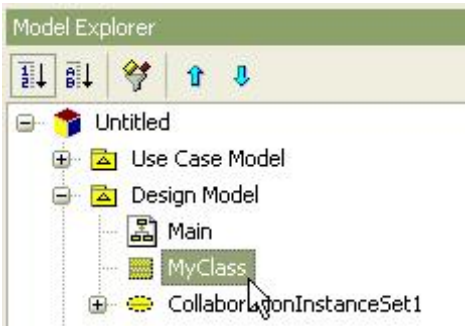
② 그러면 object는 다음과 같이 multi object 형식으로 보인다.



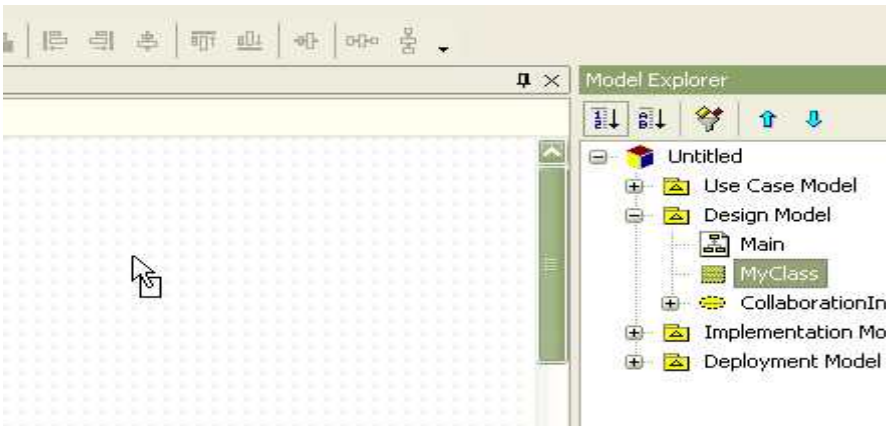
- Class로부터 Object 생성 방법:

- 클래스로부터 object를 생성하기 위해서,

① model explorer에서 class를 선택한다.

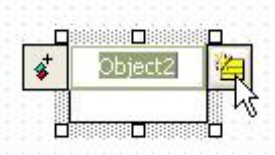


② 선택된 class를 collaboration diagram으로 드래그 한다.

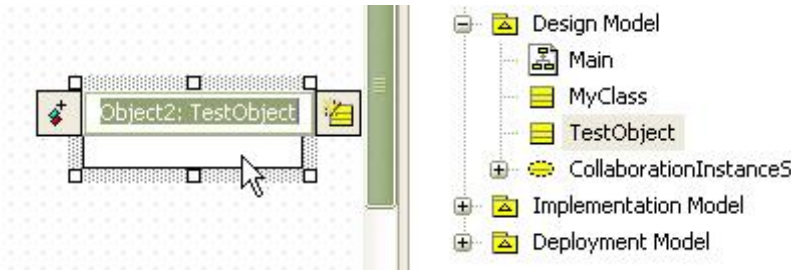


③ 그러면 object(선택된 class의 instance)가 생성된다.

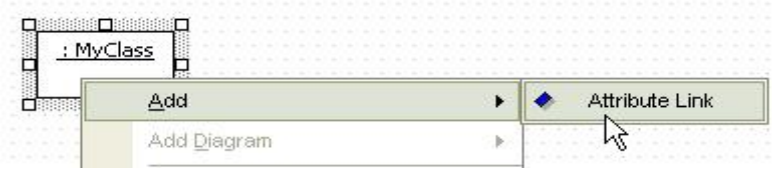
- Object로부터 Class 생성 방법:
- Object에 대해서 Class가 할당되어 있지 않다면
① Object를 더블 클릭해서 Quick Dialog의 Add Class 버튼을 이용하여 새로운 Class를 생성하고 Object에 할당할 수 있다.



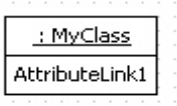
② [Enter element name] 다이얼로그에서 생성할 class의 이름을 입력한다.
③ 그러면 새로운 class가 생성되고 object에 class로 할당된다.



만약 존재하는 Class를 Object에 할당하려면 Properties의 Classifier 속성의 [그림] 버튼을 클릭해서 [Select a model element] Dialog에서 Object에 할당할 Class를 선택한다.
- Object에 AttributeLink 추가 방법:
Object에 AttributeLink를 추가하는 방법은 다음과 같이 2가지 방법이 있다.
① Object 또는 Model Explorer의 팝업 메뉴 이용
② Collection Editor 이용
- Object 또는 Model Explorer의 팝업 메뉴 이용하는 경우
① main window 또는 model explorer에서 object를 선택한다.
② 선택된 object의 오른쪽 마우스를 클릭한다. 그리고 [Add] -> [Attribute Link] 팝업 메뉴를 선택하면 attribute link를 삽입한다.

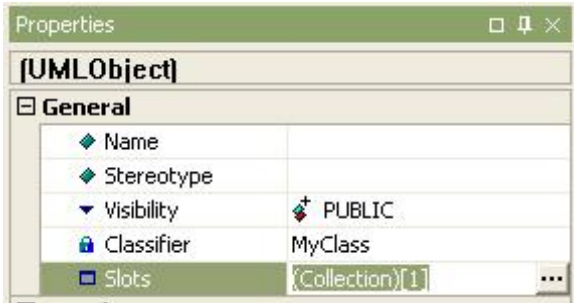


③ 그러면 새로운 attribute link가 생성된다.

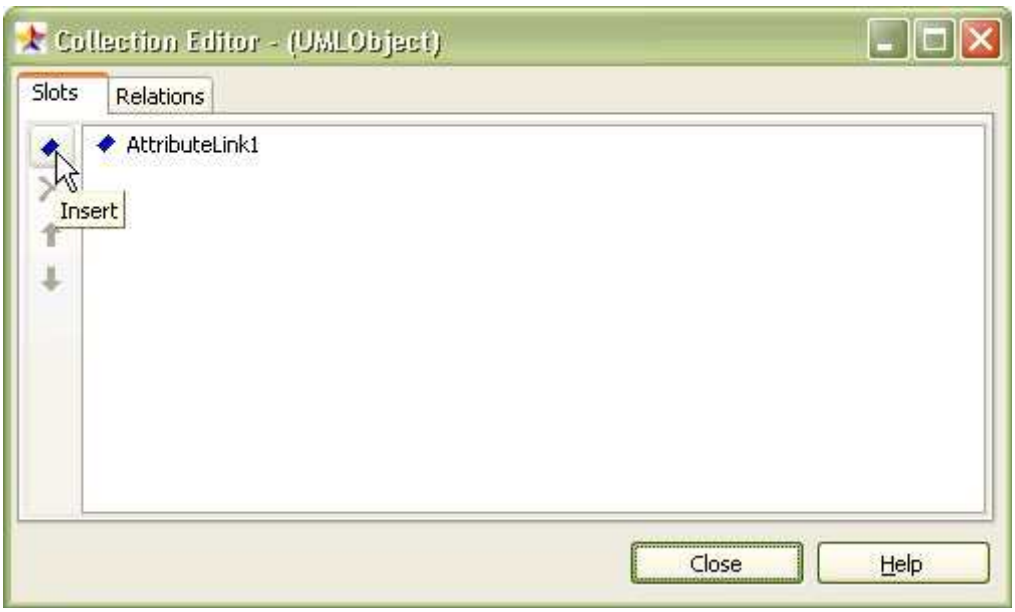


- Collection Editor 이용하는 경우

① object의 [CollectionEditor...] 팝업 메뉴를 선택한다. properties window에서 slots 속성의 ... 버튼을 클릭한다.



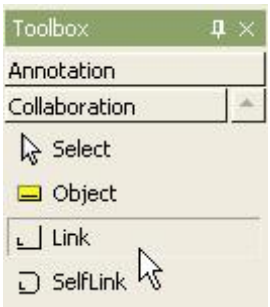
② collection editor의 slots 탭에서 ◆ 버튼을 이용하여 attribute link를 추가한다.



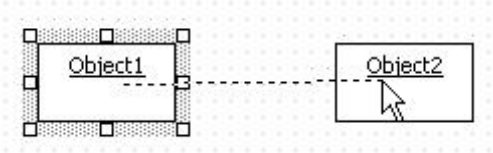
(2) Link

- Link 생성 방법:
- Link를 생성하려면,

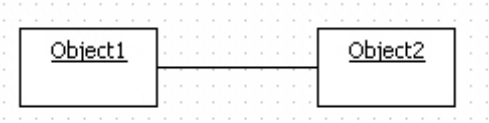
① [Toolbox] -> [Collaboration] -> [Link] 버튼을 클릭하고



② Main 윈도우창에서 Link할 Object에서 다른 Object 방향으로 마우스를 누르고 드래그하면 된다.



③ 그러면 두개의 object 사이에 link가 생성된다.

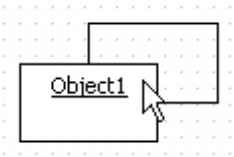


(3) SelfLink

- SelfLink 생성 방법:
- SelfLink를 생성하려면,

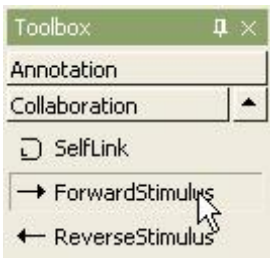
① [Toolbox] -> [Collaboration] -> [SelfLink] 버튼을 클릭하고

② Main 윈도우창에서 SelfLink을 연결할 Object 클릭한다.

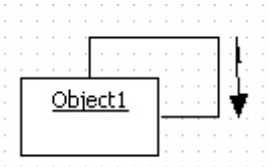


- SlefStimulus 생성 방법:
- SelfStimulus를 생성하려면,

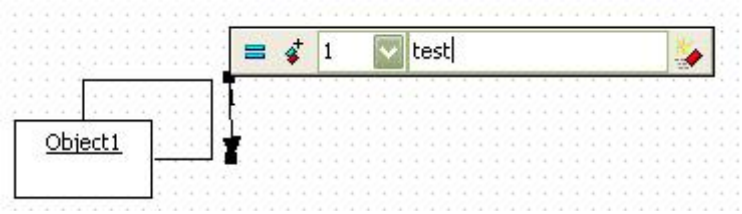
① [Toolbox] -> [Collaboration] -> [ForwardStimulus/ReverseStimulus] 버튼을 클릭하고



② SelfStimulus를 연결하고자 하는 SelfLink를 클릭한다.



③ 그리고 stimulus를 더블 클릭하고 stimulus의 이름을 퀵다이얼로그에 입력하고 [Enter] 키를 누른다.

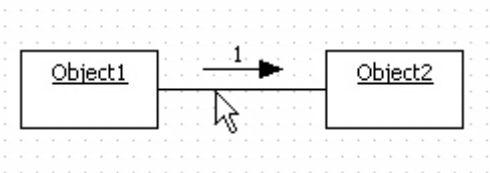


(4) Stimulus

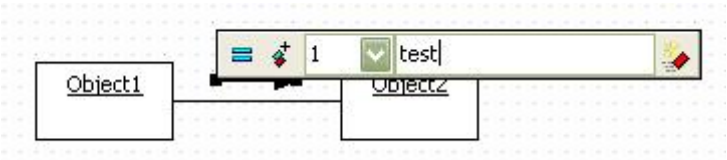
- 생성 방법:
- Stimulus를 생성하려면,

① [Toolbox] -> [Collaboration] -> [ForwardStimulus/ReverseStimulus] 버튼을 클릭하고

② Stimulus를 연결하고자 하는 Link를 클릭한다.

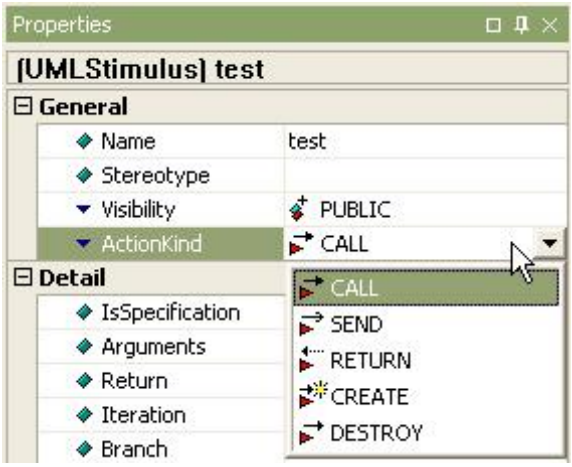


③ 그리고 stimulus를 더블 클릭하고 stimulus의 이름을 퀵다이얼로그에 입력하고 [Enter] 키를 누른다.



④ 그러면 다음과 같이 stimulus가 생성된다.

- Stimulus의 ActionKind 변경 방법:
- Stimulus의 ActionKind 속성 값은 4가지 종류 중에 하나로 설정되어야 하며,



ActionKind의 종류에 따라서 다음과 같이 다르게 보이다. ActionKind의 종류는 Properties에서 변경할 수 있다.

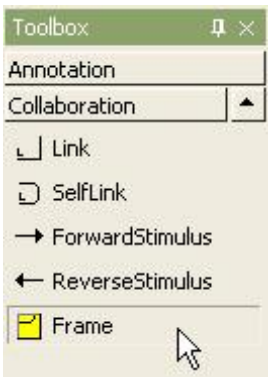
ActionKind	Shape
CALL	
SEND	
RETURN	
CREATE	<<create>>
DESTROY	<<destroy>

(5) Frame

- 생성 방법:

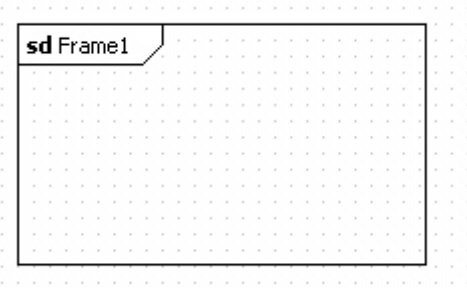
- Frame을 생성하려면,

① [Toolbox] -> [Collaboration] -> [Frame] 버튼을 클릭하고



② Main 윈도우창에서 Frame이 위치할 곳을 클릭한다.

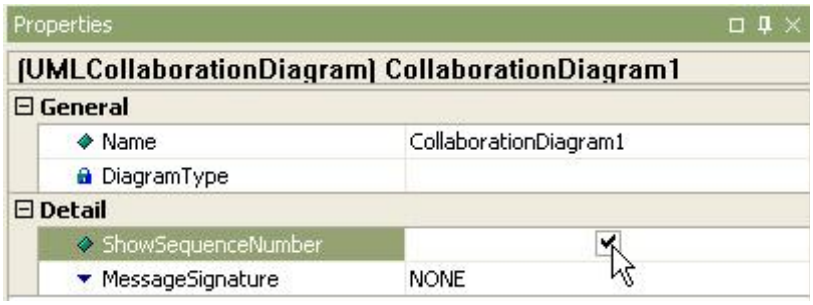
③ 결과는 다음과 같다.



(6) Diagram

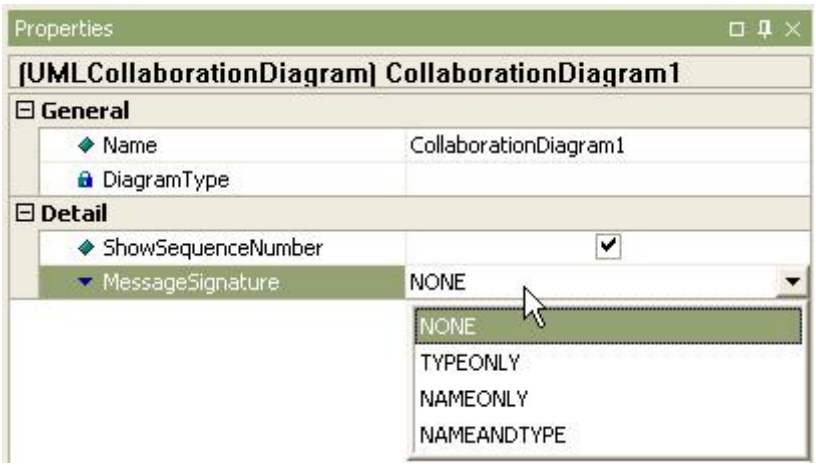
- 다이어그램에서 시퀀스 넘버 보이기:

다이어그램에서 Stimulus의 시퀀스 넘버를 나타내거나 감추려면 Model Expolorer의 다이어그램을 선택하거나, Main 윈도우의 다이어그램을 선택하고 ShowSequence 속성을 true 또는 false로 설정한다.



- 다이어그램에서 메시지의 시그너처 스타일 변경 방법:

- 다이어그램에서 Stimulus의 signature는 다음과 같이 4가지가 존재한다.



stimulus의 signature를 변경하려면 Model Expolorer의 다이어그램을 선택하거나, Main 윈도우의 다이어그램을 선택하고 Message Signature 속성 값을 변경한다.

Style	Description
NONE	shows only message name
NAMEONLY	shows message name and arguement name
TYPEONLY	shows message name, arguement type, and return type
NAMEANDTYPE	shows message name, arguement name, arguement type, and return type

마. Statechart 다이어그램 모델링하기

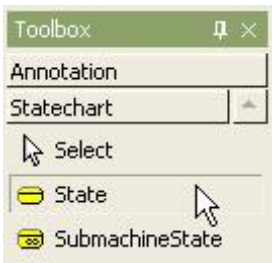
Statechart 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- State
- SubmachineState
- InitialState
- FinalState
- JunctionPoint
- ChoicePoint
- ShallowHistory
- DeepHistory
- Synchronization
- Flow Final
- Transition
- SelfTransition

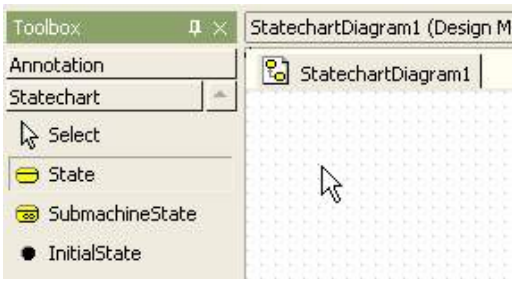
(1) State

- 생성 방법:
- State를 생성하려면,

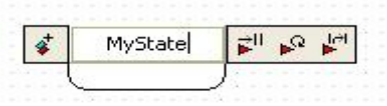
① [Toolbox] -> [Statechart] -> [State] 버튼을 클릭하고



② Main 윈도우창에서 State가 위치할 곳을 클릭한다.

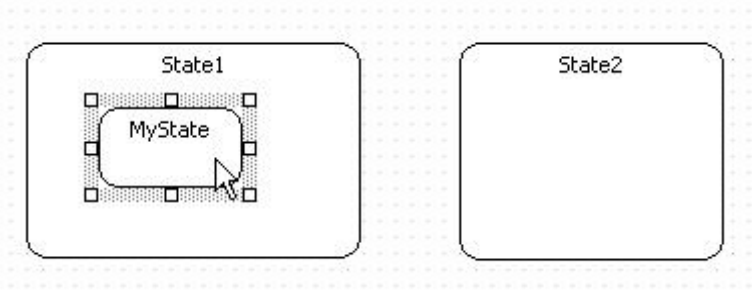


③ state가 생성되고 퀵다이얼로그가 나타나면 state의 이름을 입력한다.

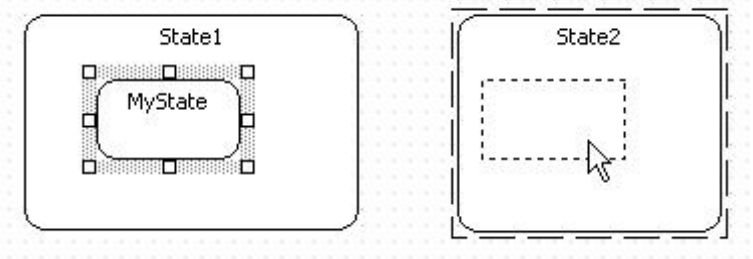


④ 그리고 [Enter] 키를 누르면 생성이 완료된다.

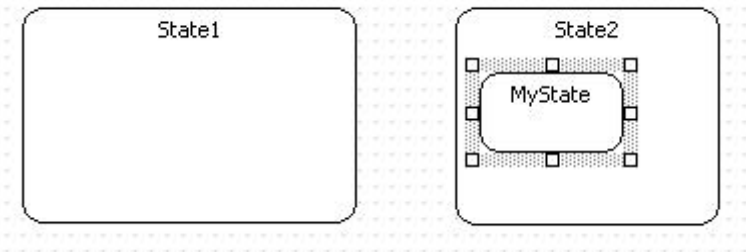
- State를 State안으로 이동하기:
- state를 또 다른 state 안으로 이동하기 위해서는,
① 이동할 state를 클릭한다.



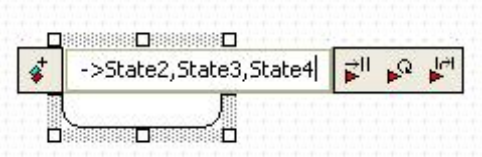
② 선택된 state를 또 다른 state로 드래그해서 드롭 한다.



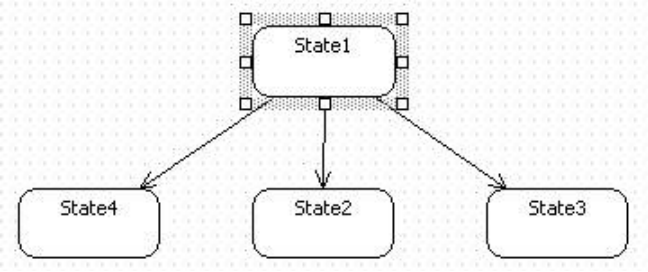
③ 선택된 state가 또 다른 state로 옮겨진다.



- 한꺼번에 여러 개의 State로 Transition하기:
현재State로부터 나가거나 들어오는 transition을 가지는 State를 만들려면 단축 생성 구문을 사용한다.
① State를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "->" 문자열 (또는 "<-" 문자열) 다음에 포함되는 다른 State의 이름을 입력한다. 여러 개의 State를 포함하기 위해서 각 State 이름은 "," 문자로 구분해서 입력한다.



② 그리고 [Enter]키를 누르면 선택된 State에서 나가는(들어오는) 여러 State들이 생성되고 자동 배열되어 생성된다.



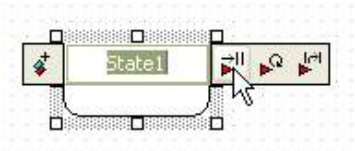
- Entry/Do/Exit Action 추가하는 방법:

State에 Entry/Do/Exit Action을 추가하는 방법은 다음과 같이 3가지 방법이 있다.

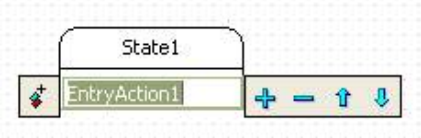
- Quick Dialog를 이용하는 경우,

① state를 더블 클릭한다.

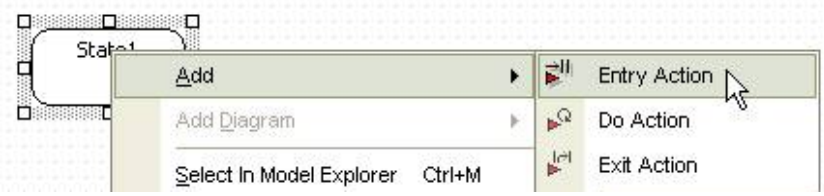
② 퀵다이얼로그에서 [Add Entry/Add DoAction/Add ExitAction] 버튼을 클릭한다.



③ 그러면 state에 원하는 action이 삽입된다.

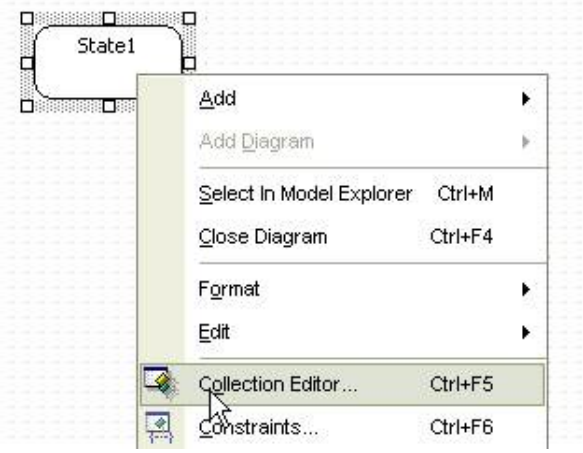


State 또는 Model Explorer의 팝업 메뉴 이용하는 경우, main window 또는 model explorer에서 state를 선택하고 마우스 오른쪽 클릭하고 [Add] -> [Entry/Do/Exit] 팝업메뉴를 선택한다.

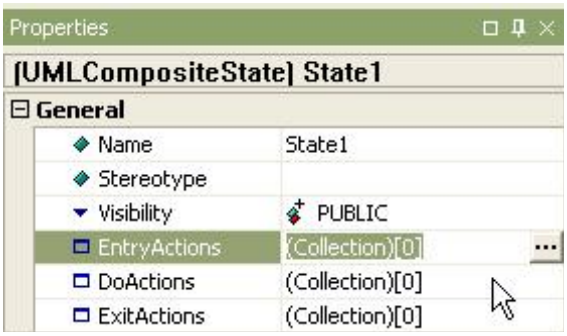


- Collection Editor 이용하는 경우,

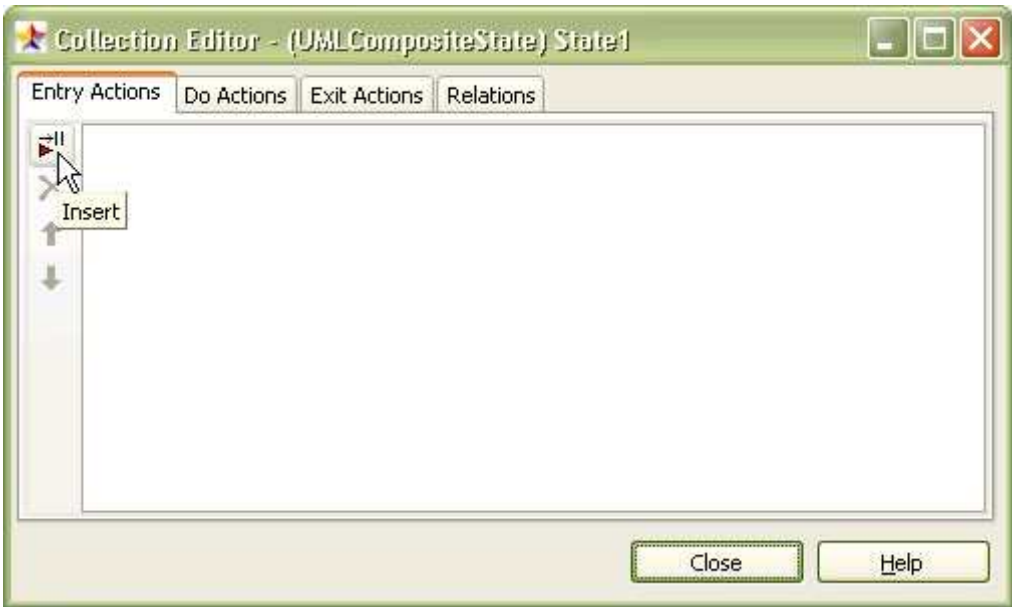
① state의 [CollectionEditor...] 팝업 메뉴를 선택하고



② properties window에서 EntryActions/DoActions/ExitActions 속성의 ... 버튼을 클릭한다.



③ 그리고 collection editor의 Entry Actions/Do Actions/Exit Actions 탭에서  버튼을 이용하여 action을 추가한다.



(2) SubmachaineState

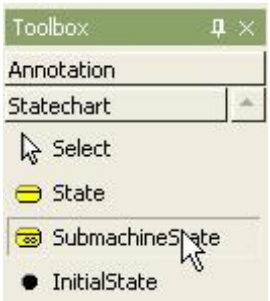
- 의미:

하위-머신상태(SubmachineState)는 하나의 상태머신(StateMachine)을 호출한다. 하위-머신으로 수행이 진입하면 해당 상태머신이 수행을 시작하고 수행이 종료되면 하위-머신상태의 수행이 종료된 것으로 간주한다.

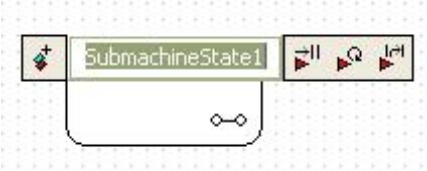
- 생성 방법:

- SubmachineState를 생성하려면,

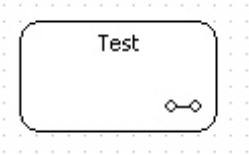
① [Toolbox] -> [Statechart] -> [SubmachineState] 버튼을 클릭하고



② Main 윈도우창에서 SubmachineState가 위치할 곳을 클릭하면, submachine state가 생성되고 퀵다이얼로그가 나타난다.



③ 퀵다이얼로그에서 이름을 입력하고 [Enter] 키를 누르면 다음과 같이 submachine state가 생성된다.



(3) InitialState

- 생성 방법:

- InitialState를 생성하려면,

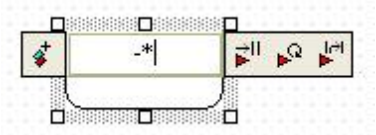
① [Toolbox] -> [Statechart] -> [InitialState] 버튼을 클릭하고,

② InitialState를 위치할 곳을 main window에서 클릭한다.

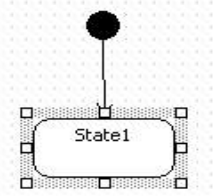
- State에서 initialstate 생성하기:

현재 State로 나가는 transition을 가지는 InitialState를 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-*" 문자열과 InitialState의 이름을 입력한다.



② 그리고 [Enter] 키를 누르면 선택된 State로 들어오는 transition과 InitialState가 생성된다.

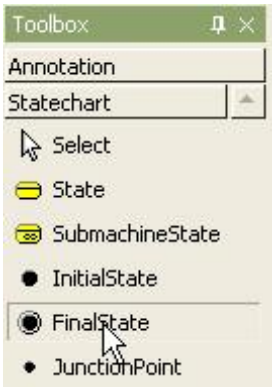


(4) FinalState

- 생성 방법:

- FinalState를 생성하려면,

① [Toolbox] -> [Statechart] -> [FinalState] 버튼을 클릭하고

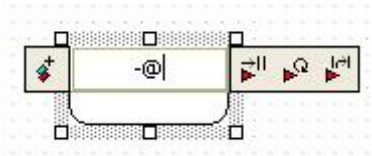


② Main 윈도우창에서 FinalState가 위치할 곳을 클릭한다.

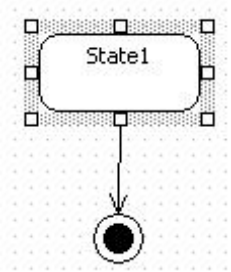
- State에서 FinalState 생성하기:

현재 State로 나가는 transition과 FinalState를 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-@" 문자열과 FinalState의 이름을 입력한다.



② 그리고 [Enter]키를 누르면 선택된 State에서 나가는 transition과 FinalState가 생성된다.

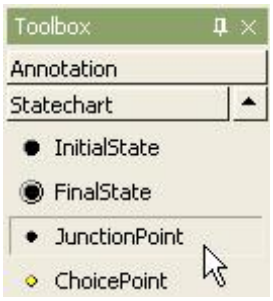


(5) JunctionPoint

- 생성 방법:

- JunctionPoint를 생성하려면,

① [Toolbox] -> [Statechart] -> [JunctionPoint] 버튼을 클릭하고

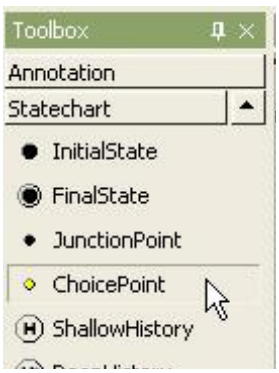


② Main 윈도우창에서 JunctionPoint가 위치할 곳을 클릭한다.

(6) ChoicePoint

- 생성 방법:
- ChoicePoint를 생성하려면,

① [Toolbox] -> [Statechart] -> [ChoicePoint] 버튼을 클릭하고

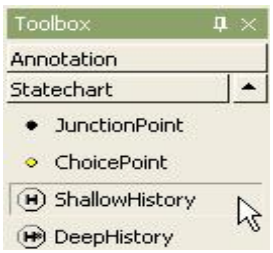


② Main 윈도우창에서 ChoicePoint가 위치할 곳을 클릭한다.

(7) ShallowHistory

- 생성 방법:
- ShallowHistory를 생성하려면,

① [Toolbox] -> [Statechart] -> [ShallowHistory] 버튼을 클릭하고

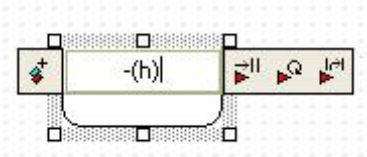


② Main 윈도우창에서 ShallowHistory가 위치할 곳을 클릭한다.

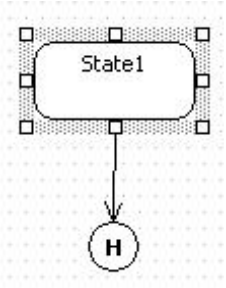
- State에서 History State 생성:

현재 State에서 나가는 transition과 History를 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-(h) -(H) -(h*) -(H*)"을 입력한다.



② 그리고 [Enter]키를 누르면 선택된 State에서 나가는 transition과 History가 생성된다.

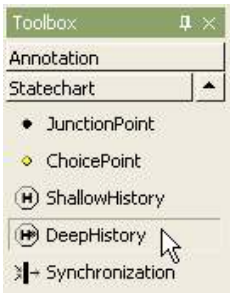


(8) DeepHistory

- 생성 방법:

- DeepState를 생성하려면,

① [Toolbox] -> [Statechart] -> [DeepState] 버튼을 클릭하고



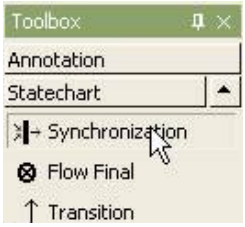
② Main 윈도우창에서 DeepHistory가 위치할 곳을 클릭한다.

(9) Synchronization

- 생성 방법:

- Synchronization을 생성하려면,

① [Toolbox] -> [Statechart] -> [Synchronization] 버튼을 클릭하고

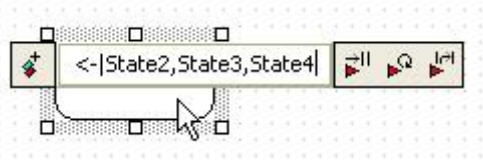


② Main 윈도우창에서 Synchronization이 위치할 곳을 클릭한다.

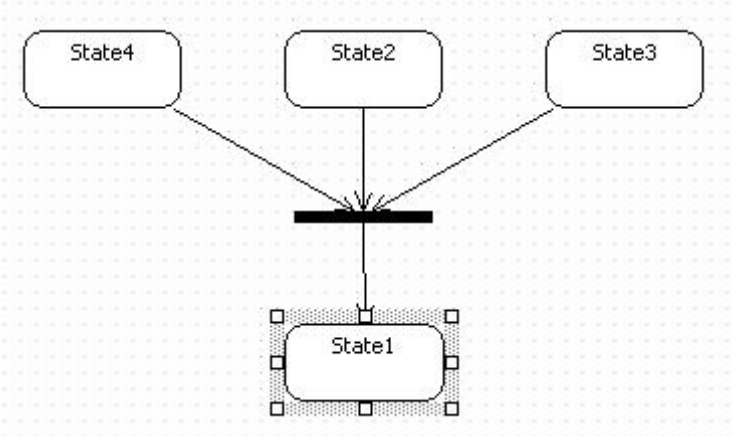
- Join 만들기:

현재 State로 들어오는 Join transition을 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면 "<-|" 문자열과 Join 되어질 State의 이름을 입력한다. 여러 개의 State를 포함하기 위해서 각 State 이름은 "," 문자로 구분해서 입력한다.



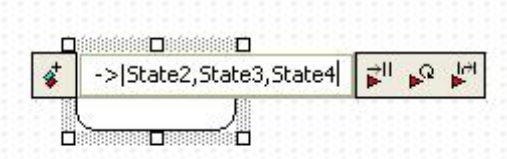
② 그리고 [Enter]키를 누르면 선택된 State로 Join되는 여러 State들이 생성되고 자동 배열되어 생성된다.



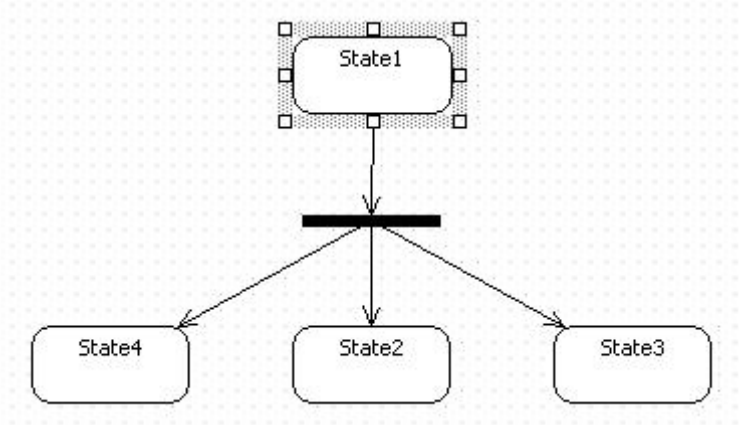
- Fork 만들기:

현재 State에서 나가는 Fork transition을 만들려면 단축 생성 구문을 사용한다.

① State를 더블 클릭해서 Quick Dialog가 나타나면 "->|" 문자열과 Fork 되어질 State의 이름을 입력한다. 여러 개의 State를 포함하기 위해서 각 State 이름은 "," 문자로 구분해서 입력한다.



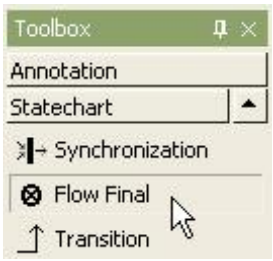
② 그리고 [Enter]키를 누르면 선택된 State로 Fork되는 여러 State들이 생성되고 자동 배열되어 생성된다.



(10) Flow Final

- 생성 방법:
- Flow Final을 생성하려면,

① [Toolbox] -> [Statechart] -> [Flow Final] 버튼을 클릭하고

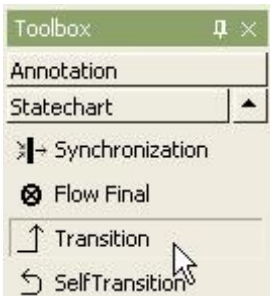


② Main 윈도우창에서 Flow Final이 위치할 곳을 클릭한다.

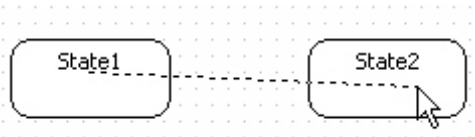
(11) Transition

- Transition 생성 방법:
- Transition을 생성하려면,

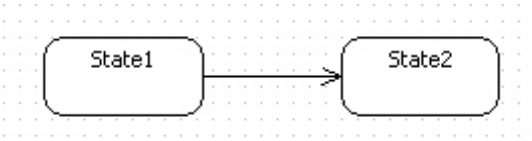
① [Toolbox] -> [Statechart] -> [Transition] 버튼을 클릭하고



② Main 윈도우창에서 State에서 전이하는 방향의 다른 State로 마우스를 누르고 드래그하면 된다.

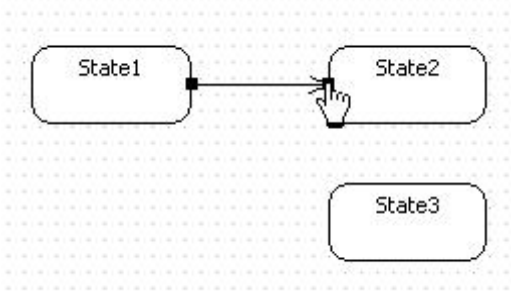


③ 그러면 두개의 state 사이의 transition이 생성된다.

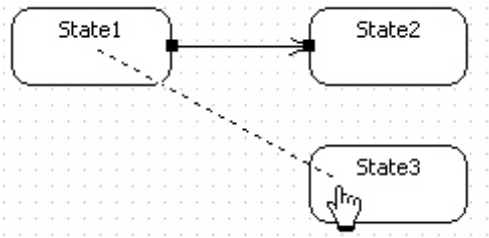


- Transition 끝 변경하기:
- Transition 끝을 다른 State에 연결하려면

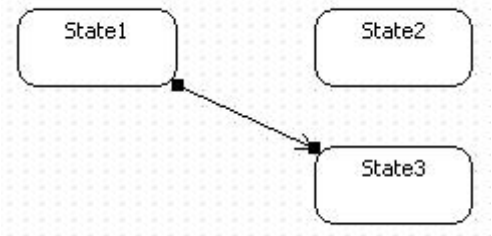
① transition의 끝을 선택한다.



② Transition의 끝점을 드래그해서 연결하려는 다른 State로 드래그/드롭 한다.



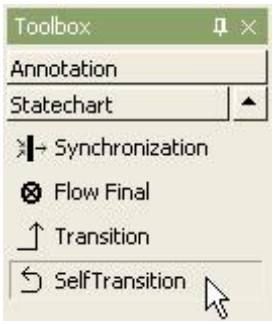
③ 그러면 transition 연결이 다음과 같이 변경된다.



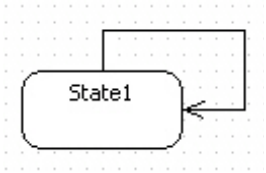
(12) SelfTransition

- SelfTransition 생성 방법:
- SelfTransition을 생성하려면,

① [Toolbox] -> [Statechart] -> [SelfTransition] 버튼을 클릭하고



② Main 윈도우창에서 SelfTransition하는 State를 클릭한다.



바. Component 다이어그램 모델링하기

Component 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

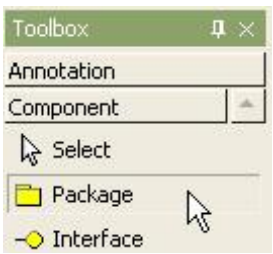
- Package
- Interface
- Component
- ComponentInstance
- Artifact
- Port
- Part
- Association
- Dependency
- Realization
- Link
- Connector

(1) Package

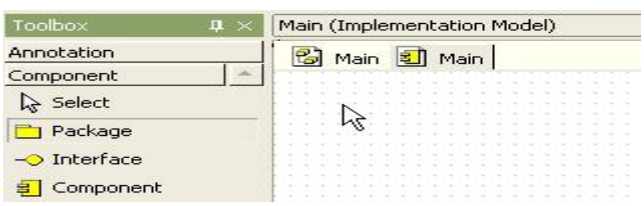
패키지(Package)는 모델 요소들을 논리적으로 그룹화하여 관리하기 위한 요소이다. 패키지는 요소들을 조직화하기 위한 어떠한 용도로 사용되어도 무방한 매우 일반적인 요소이다. 패키지 대신 모델(Model), 서브시스템(Subsystem)의 더욱 특수화된 요소를 사용할 수도 있다.

- Package 생성하는 방법:
- Package를 생성하려면,

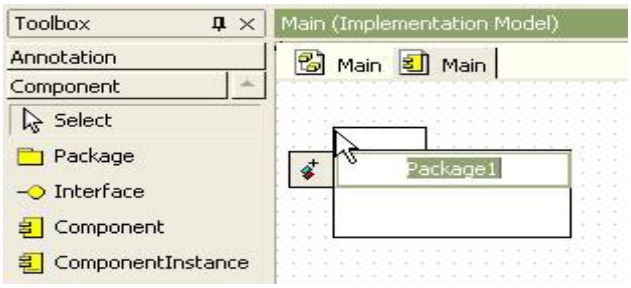
① [Toolbox] -> [Component] -> [Package] 버튼을 클릭하고



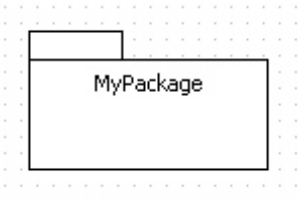
② Main 윈도우창에서 Package가 위치할 곳을 클릭한다.



③ 패키지가 생성되고 퀵다이얼로그가 나타난다.



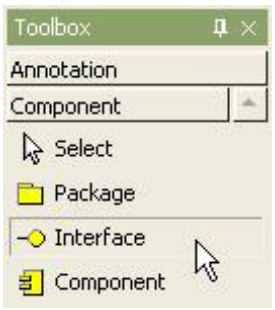
④ 퀵다이얼로그에서 package 이름을 입력하고 [Enter] 키를 입력하면 다음과 같이 패키지가 보이다.



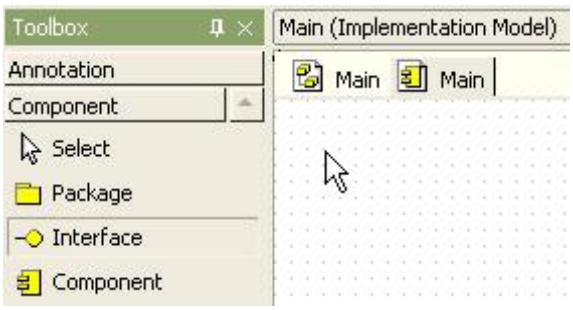
(2) Interface

- Interface을 생성하는 방법:
- Interface를 생성하려면,

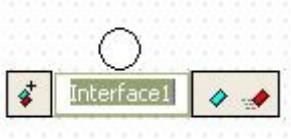
① [Toolbox] -> [Component] -> [Interface] 버튼을 클릭하고



② Main 윈도우창에서 Interface가 위치할 곳을 클릭한다.



③ 퀵다이얼로그에서 interface의 이름을 입력한다.



④ [Enter] 키를 누르면 인터페이스는 다음과 같이 보인다.

(3) Component

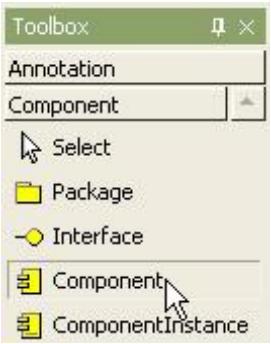
- 의미:

컴포넌트(Component)">컴포넌트(Component)는 독립적이며, 배치가 자유롭고, 교체가 가능한 시스템의 한 부분을 나타낸다. 컴포넌트는 구현을 캡슐화하고 인터페이스(Interface)를 외부에 공개한다.

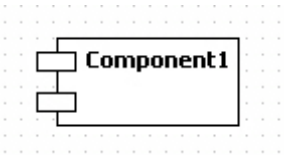
- 생성 방법:

- Component를 생성하려면,

- [Toolbox] -> [Component] -> [Component] 버튼을 클릭하고



② Main 윈도우창에서 Component가 위치할 곳을 클릭한다. 그리고 퀵다이얼로그에서 component의 이름을 입력하고 [Enter] 키를 누르면 다음과 같이 결과가 보인다.

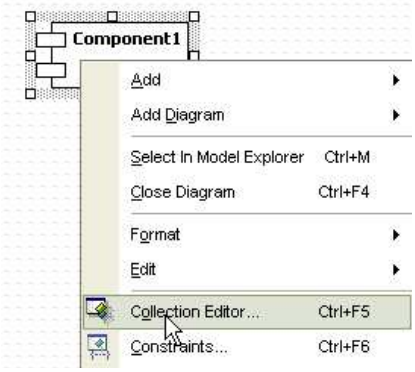


- Resident 추가하는 방법:

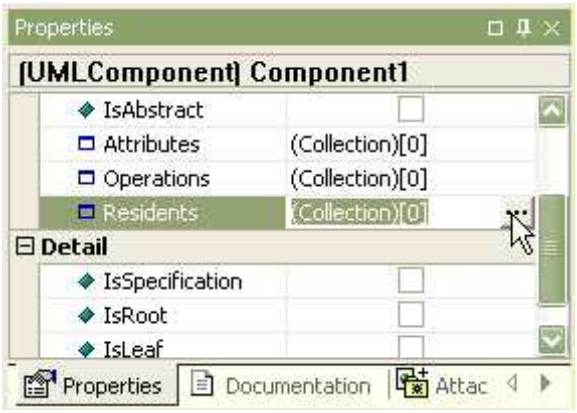
Model Explorer에서 Component를 선택하고 [CollectionEditor...] 팝업 메뉴 또는 Properties 윈도우의 Residents의 콜랙션 편집 버튼을 통해서 Collection Editor를 열어서 Residents 탭에서 resident 추가 버튼을 이용하여 Resident를 입력할 수 있다.

- Component에 Resident 요소를 추가하려면

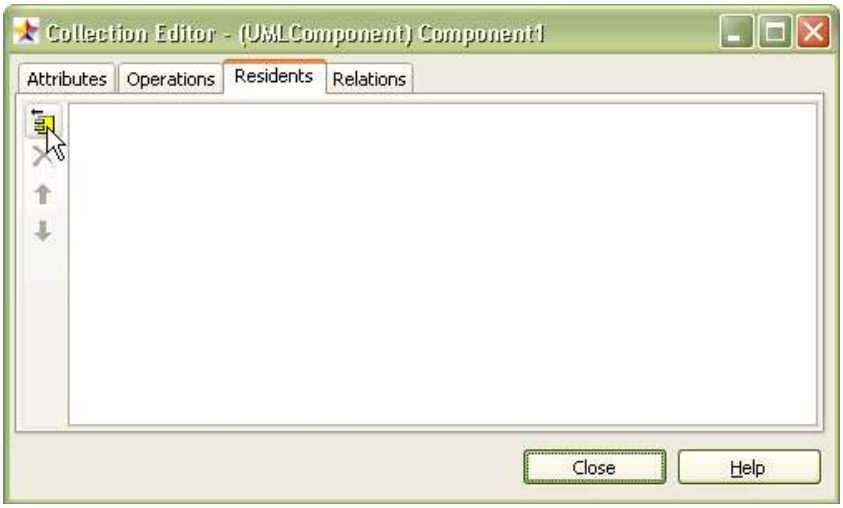
① Component의 [CollectionEditor...] 팝업 메뉴를 선택한다.



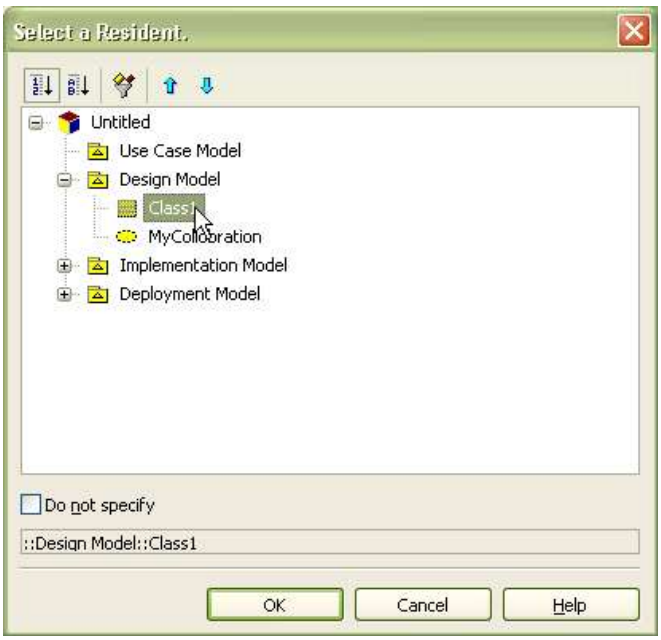
② 또는 properties window에서 residents의 ... 버튼을 클릭한다.



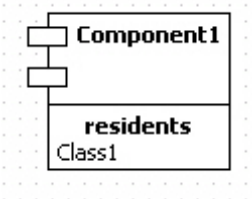
③ collection editor의 residents 탭에서  버튼을 이용하여 resident 요소를 추가한다.



④ [Select a Resident] 다이얼로그에서 resident 요소를 선택한다.

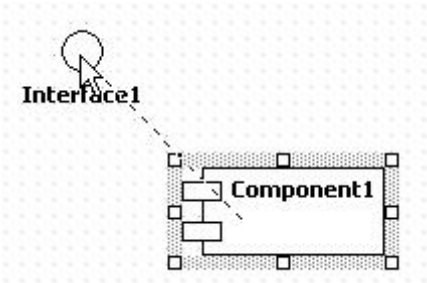


⑤ resident 요소가 컴포넌트에 할당되어지며 다음과 같이 보인다.

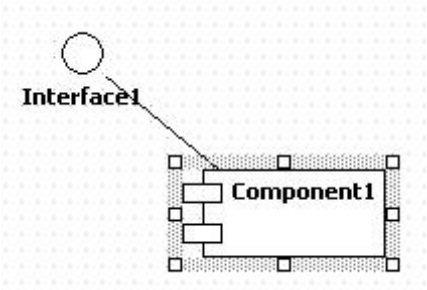


- Interface Providing 관계 설정 방법:
- Interface Providing 을 생성하려면,
① [Toolbox] -> [Component] -> [Realization] 버튼을 클릭하고

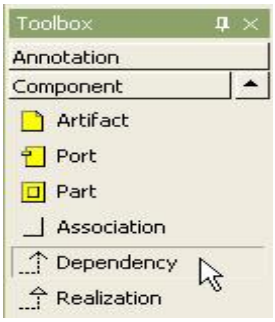
② Main 윈도우창에서 Component에서 Interface로 마우스를 누르고 드래그하면 된다.



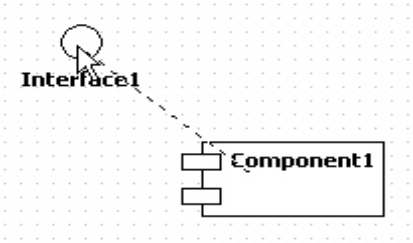
③ 결과는 다음과 같다.



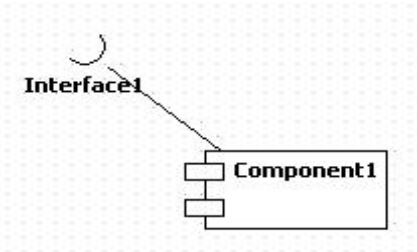
- Interface Requiring 관계 설정 방법:
- Interface Requiring 관계를 생성하려면,
① [Toolbox] -> [Component] -> [Dependency] 버튼을 클릭하고,



② Main 윈도우창에서 Component에서 Interface로 마우스를 누르고 드래그하면 된다.



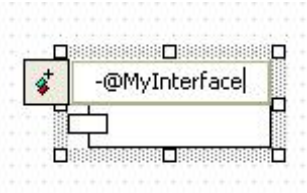
③ 마지막으로 interface requiring relationship is created.



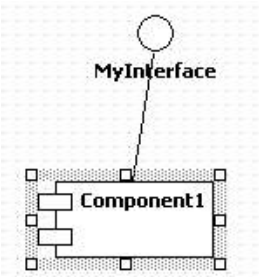
- Interface Providing 관계 생성 방법:

현재 선택된 Class로부터 Interface Providing 관계를 만들려면 단축 생성 구문을 사용한다.

① Class를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-@" 문자열 다음에 Interface의 이름을 입력한다. 여러 개의 Interface를 제공한다면 각 Interface 이름은 "," 문자로 구분해서 입력한다.



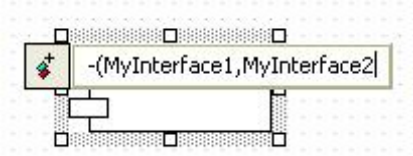
② 그리고 [Enter] 키를 누르면 선택된 Class와 Interface Provinding 관계를 가지는 Interface들이 생성되거나 연결되고 자동 배열되어 생성된다.



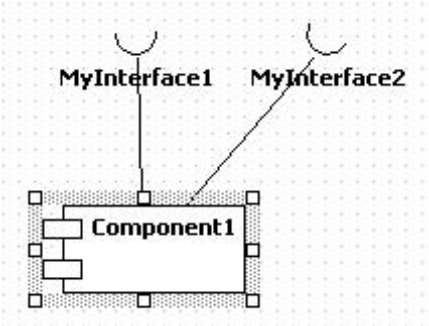
- Interface Requiring 관계 생성 방법:

현재 선택된 Class로부터 Interface Requiring 관계를 만들려면 단축 생성 구문을 사용한다.

① Class를 더블 클릭해서 Quick Dialog가 나타나면, Quick Dialog에서 "-(" 문자열 또는 "-->" 문자열 다음에 Interface의 이름을 입력한다. 여러 개의 Interface를 요구한다면 각 Interface 이름은 "," 문자로 구분해서 입력한다.



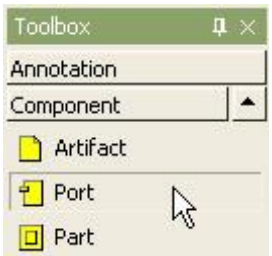
② 그리고 [Enter]키를 누르면 선택된 Class와 Interface Requiring 관계를 가지는 Interface들이 생성되거나 연결되고 자동 배열되어 생성된다.



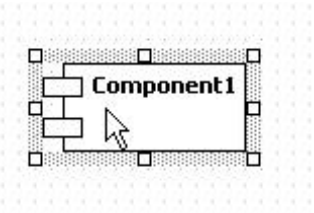
- Port 생성 방법:

- Component에 Port를 생성하려면,

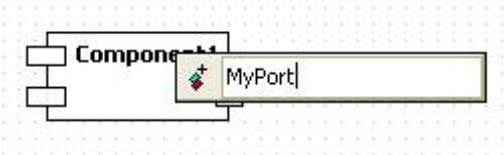
① [Toolbox] -> [Component] -> [Port] 버튼을 클릭하고



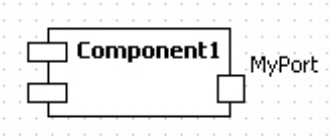
② Main 윈도우에서 Port가 위치할 Component를 클릭한다.



③ component상에 port가 생성되면 퀵다이얼로그에 port의 이름을 입력하고 [Enter] 키를 누른다.



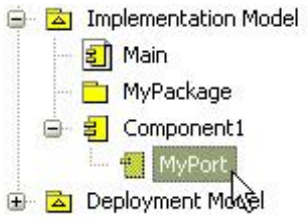
④ 그러면 컴포넌트가 다음과 같이 보인다.



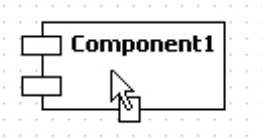
- Port 드래그를 통한 뷰 생성 방법:

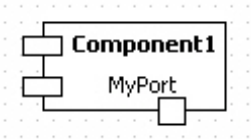
Model Explorer로부터 드래그를 통하여 Port를 생성할 수 있다.

① model explorer속의 port를 드래그해서 main diagram상의 component 위에 드롭 한다.



② port가 component상에 나타난다. 만약 Component가 아닌 Diagram 위로 드롭 하면 Port와 함께 Component의 뷰가 생성된다.

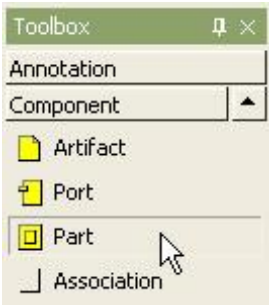




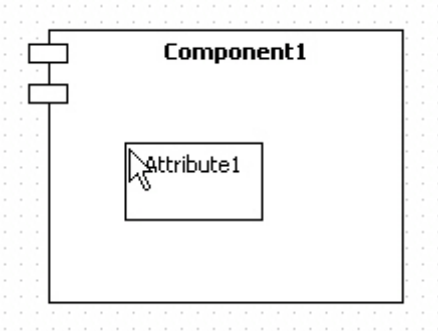
- Part 생성 방법:

- Component에 Part를 생성하려면,

① [Toolbox] -> [Component] -> [Part] 버튼을 클릭하고



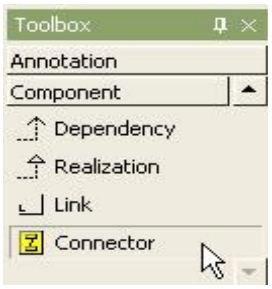
② Main 윈도우에서 Part가 위치할 Component를 클릭한다.



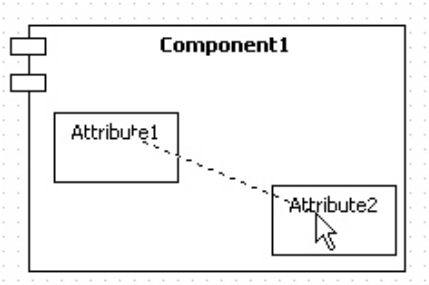
- Connector 생성 방법:

- Connector를 생성하려면,

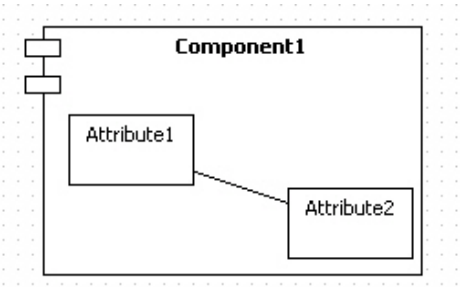
① [Toolbox] -> [Component] -> [Connector] 버튼을 클릭하고



② Main 윈도우창에서 연결할 Part(또는 Port)에서 다른 Part(또는 Port)로 마우스 버튼을 누르고 드래그하면 된다.



③ 그러면 두개의 part 사이에 connector가 다음과 같이 생성된다.



(4) ComponentInstance

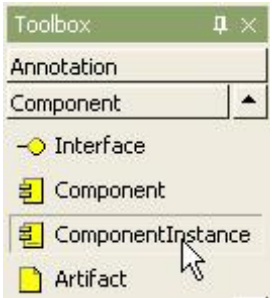
- 의미:

컴포넌트 인스턴스(ComponentInstance)">컴포넌트 인스턴스(Component Instance)는 컴포넌트의 한 사례(Instance)이다. "Classifier" 프로퍼티에 어떤 컴포넌트(Component)의 사례 인지를 지정할 수 있다.

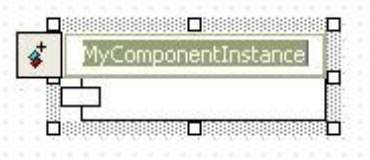
- 생성 방법:

- ComponentInstance를 생성하려면,

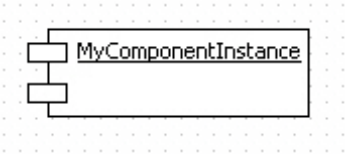
① [Toolbox] -> [Component] -> [ComponentInstance] 버튼을 클릭하고



② Main 윈도우창에서 ComponentInstance가 위치할 곳을 클릭한다.



③ 퀵다이얼로그에서 component instance의 이름을 입력하고 [Enter] 키를 누르면 다음과 같이 결과가 나타난다.

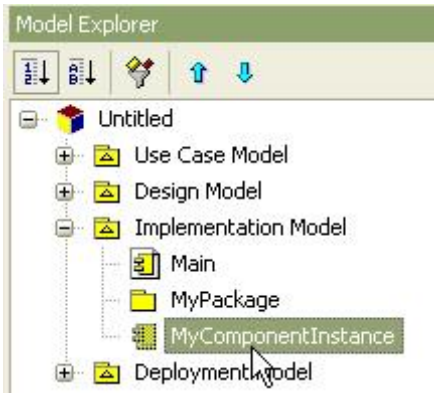


- Attribute Link 추가하기:

ComponentInstance에 AttributeLink를 추가하는 방법은 다음과 같이 2가지 방법이 있다. Main 윈도우 또는 Model Explorer에서 ComponentInstance를 선택하고 오른쪽 마우스 버튼을 눌러서 [Add] -> [Attribute Link] 팝업 메뉴를 선택하여 Attribute Link를 추가할 수 있다.

- ComponentInstance 또는 Model Explorer의 팝업 메뉴 이용하는 경우,

① main window 또는 model explorer에서 ComponentInstance를 선택한다.



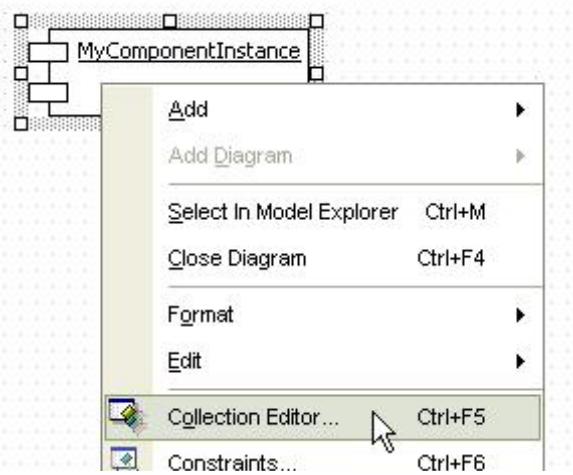
② 오른쪽 마우스 버튼을 눌러서 [Add] -> [Attribute Link] 팝업 메뉴를 선택한다.

③ 그러면 다음과 같이 attribute link가 추가된다.

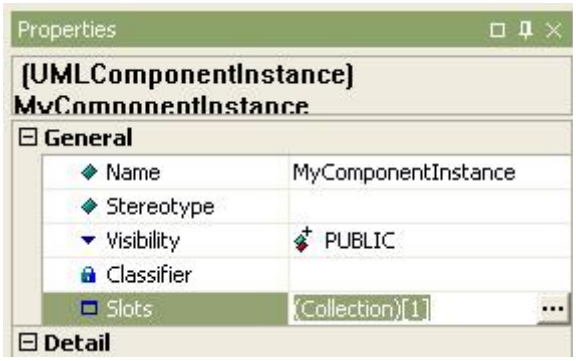
ComponentInstance의 [CollectionEditor...] 팝업 메뉴 또는 Properties 윈도우의 Slots의 편집 버튼을 통해서 Collection Editor를 열어서 Slots 탭에서 attribute link 추가 버튼을 이용하여 Attribute Link를 입력할 수 있다.

- Collection Editor 이용하는 경우,

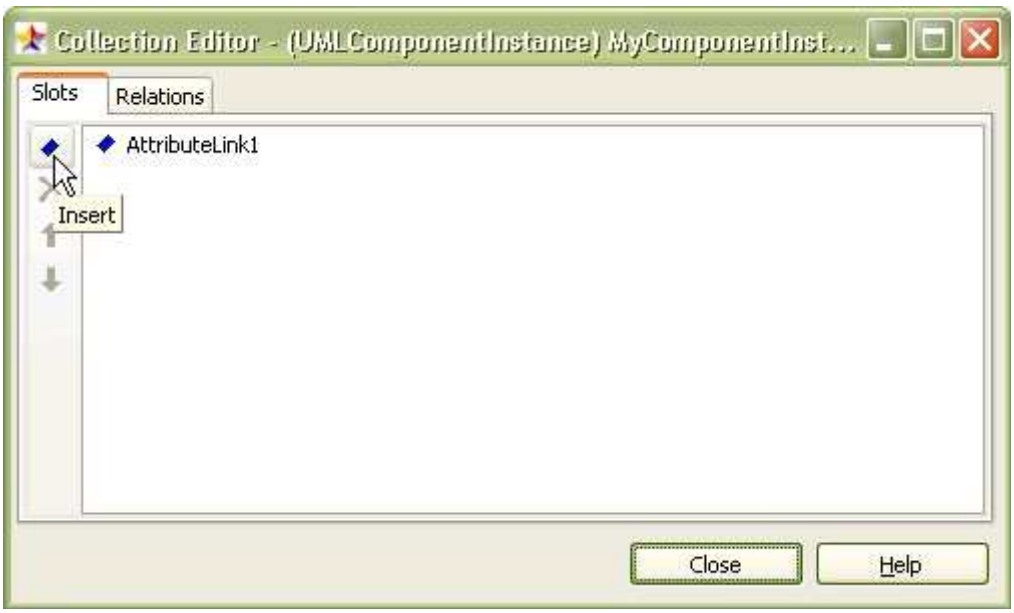
① ComponentInstance의 [CollectionEditor...] 팝업 메뉴를 선택한다.



② properties window에서 slots 속성의 ... 버튼을 클릭한다.



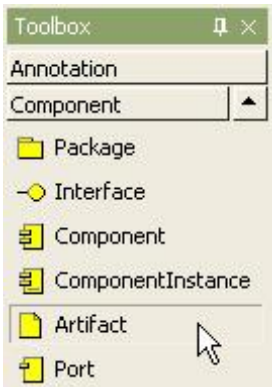
③ collection editor에서 slots tab 탭의 버튼을 이용하여 attribute link를 추가한다.



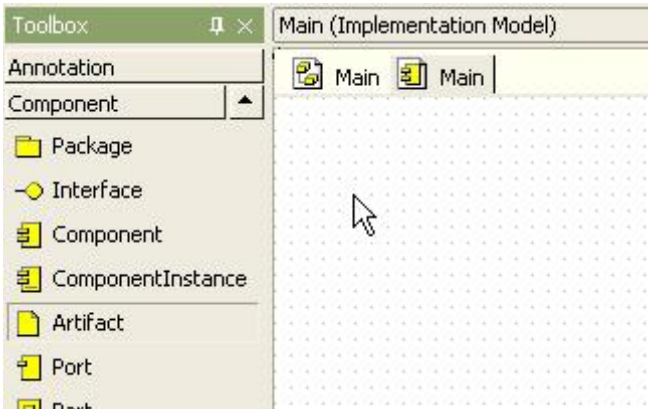
(5) Artifact

- 생성 방법:
- Artifact를 생성하려면,

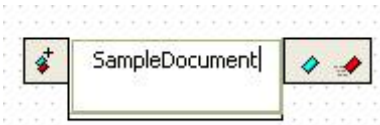
① [Toolbox] -> [Component] -> [Artifact] 버튼을 클릭하고



② Main 윈도우창에서 Artifact가 위치할 곳을 클릭한다.



③ 다이어그램상에 artifact가 생성되고 퀵다이얼로그가 나타나면, artifact의 이름을 퀵다이얼로그에 입력한다.

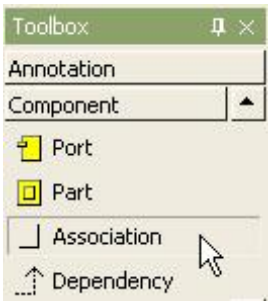


④ 그리고 [Enter] 키를 입력하면 생성이 완료된다.

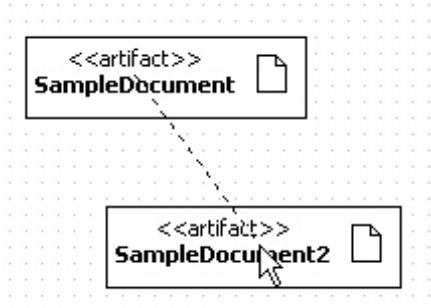
(6) Association

- Association 생성 방법:
- Association을 생성하려면,

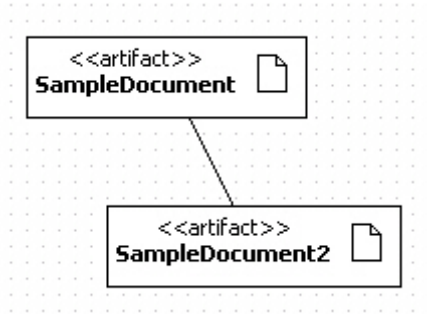
① [Toolbox] -> [Component] -> [Association] 버튼을 클릭하고



② Main 윈도우창에서 연관할 요소에서 다른 요소로 마우스 버튼을 누르고 드래그하면 된다.



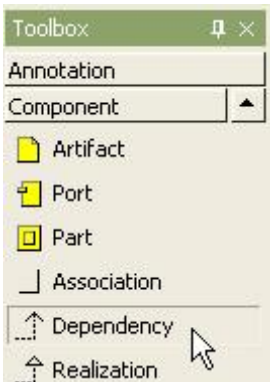
③ 두개의 요소사이에 association이 다음과 같이 생성된다.



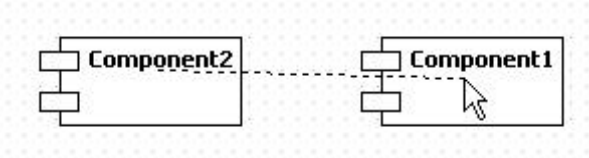
(7) Dependency

- Dependency 생성 방법:
- Dependency를 생성하려면,

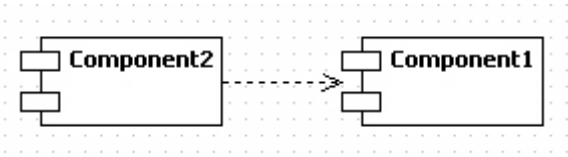
① [Toolbox] -> [Component] -> [Dependency] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 의존하는 방향의 요소로 마우스 버튼을 누르고 드래그하면 된다.



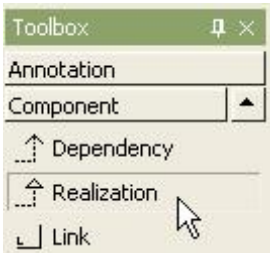
③ 두개의 요소사이에 dependency가 생성된다.



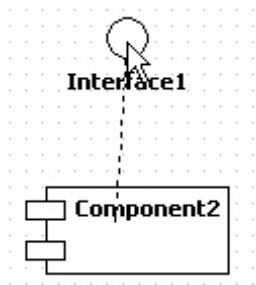
(8) Realization

- Realization 생성 방법:
- Realization을 생성하려면,

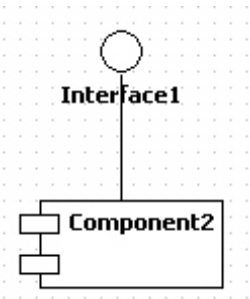
① [Toolbox] -> [Component] -> [Realization] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 Realization할 방향의 요소로 마우스 버튼을 누르고 드래그하면 된다.



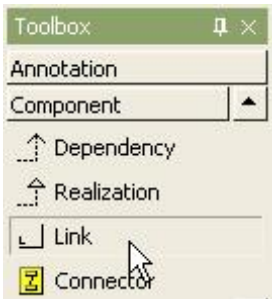
③ 그러면 realization이 다음과 같이 생성된다.



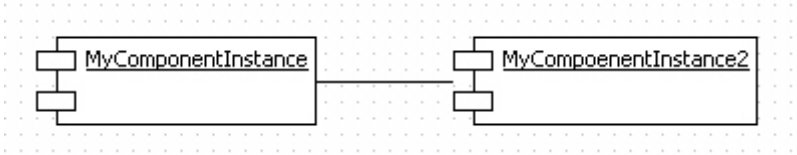
(9) Link

- Link 생성 방법:
- Link를 생성하려면,

① [Toolbox] -> [Component] -> [Link] 버튼을 클릭하고



② Main 윈도우창에서 연결할 ComponentInstance에서 다른 ComponentInstance로 마우스 버튼을 누르고 드래그하면 다음과 같이 link가 생성된다.



사. Deployment 다이어그램 모델링하기

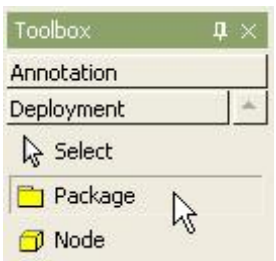
Deployment 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Package
- Node
- NodeInstance
- Artifact
- Port
- Part
- Association
- DirectedAssociation
- Dependency
- Link
- Connector

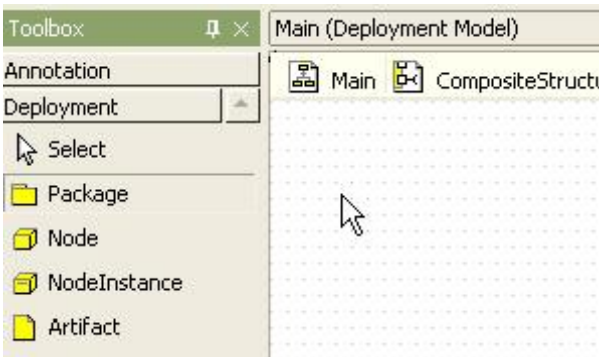
(1) Package

- Package 생성 방법:
- Package를 생성하려면,

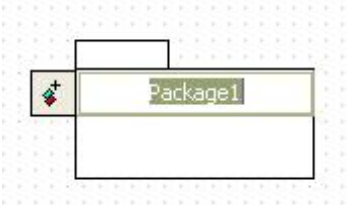
① [Toolbox] -> [Deployment] -> [Package] 버튼을 클릭하고



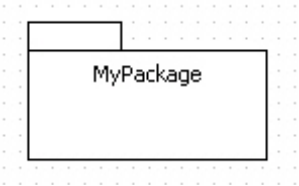
② Main 윈도우창에서 Package가 위치할 곳을 클릭한다.



③ 그러면 package가 생성되고 퀵다이얼로그가 나타난다.



④ 퀵다이얼로그에서 package 이름을 입력하고 [Enter] 키를 누르면 작업이 완료된다.



(2) Node

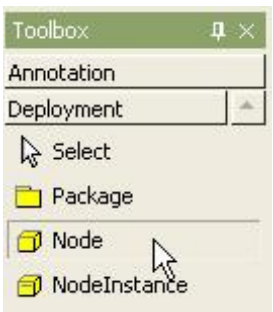
- 의미:

노드(Node)는 일반적으로 메모리와 계산처리 능력을 지닌 물리적 객체를 의미한다.
노드(Node)에 여러 개의 컴포넌트(Component)를 배치(deploy)시킬 수 있다.

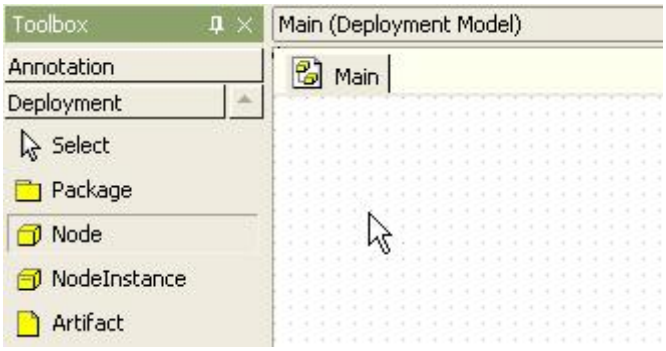
- Node 생성 방법:

- Node를 생성하려면,

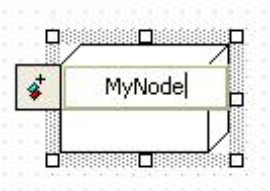
① [Toolbox] -> [Deployment] -> [Node] 버튼을 클릭하고



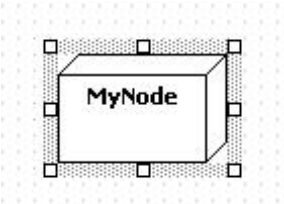
② Main 윈도우창에서 Node가 위치할 곳을 클릭한다.



③ node가 생성되고 퀵다이얼로그가 화면에 나타난다. 그러면 퀵다이얼로그에서 node의 이름을 입력한다.

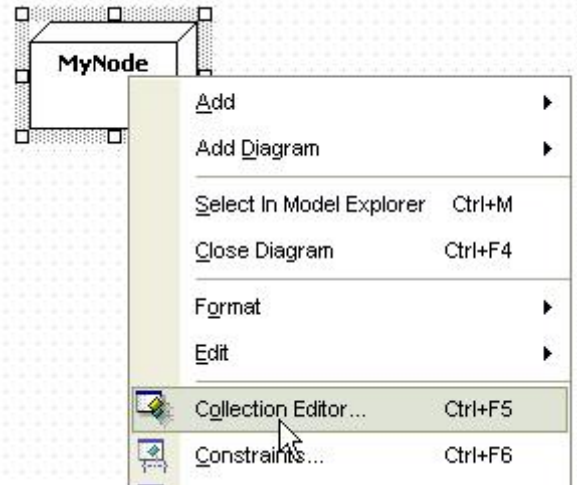


④ [Enter] 키를 누르면 다음과 같이 노드 생성이 완료된다.

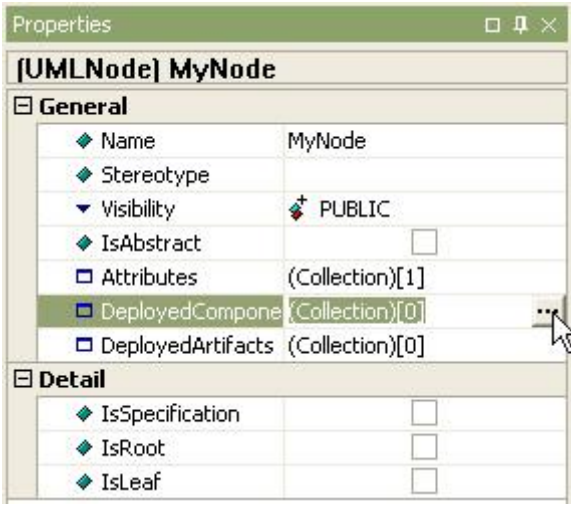


- DeployedComponent 추가하는 방법:
- deployed component 요소를 노드에 추가하려면

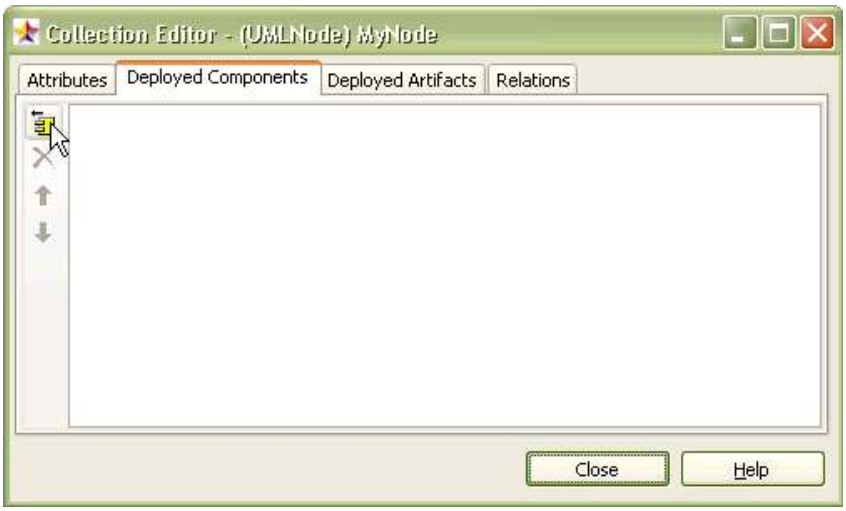
① 노드의 [CollectionEditor...] 팝업 메뉴를 선택한다.



② 또는 properties window에서 DeployedComponents 속성의 ... 버튼을 클릭한다.

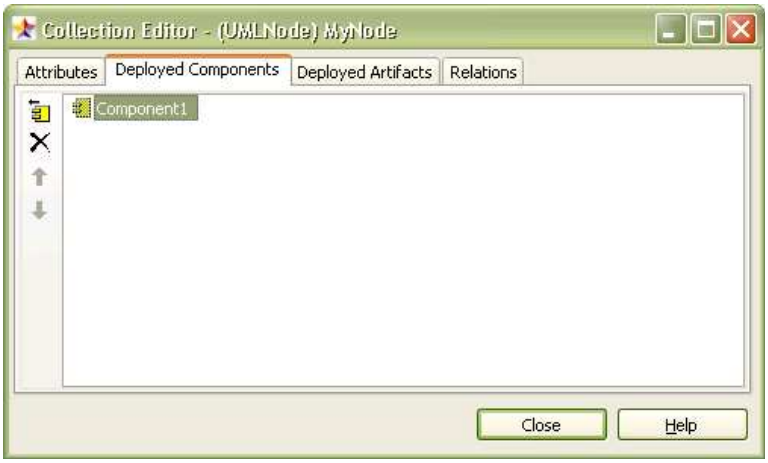


③ collectio editor의 deployed components 팁에서  버튼을 이용하여 deployed component 요소를 추가한다.

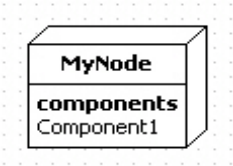


④ [select the component to deploy] 다이얼로그에서 deployed component 요소를 선택한다.

⑤ 그리고 OK 버튼을 클릭하면 deployed component 요소가 노드에 추가된다.



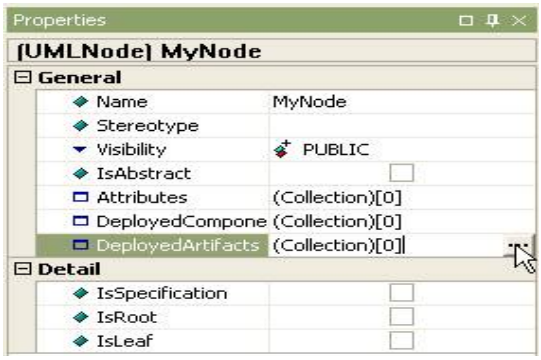
⑥ 그러면 노드는 다음과 같이 보인다.



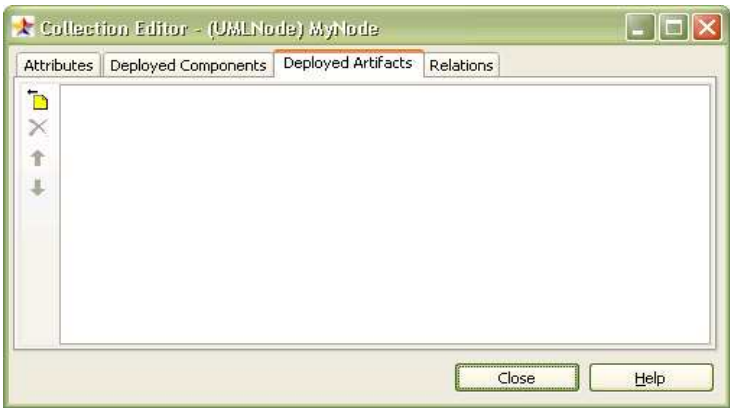
- DeployedArtifact 추가하는 방법:
- Node에 DeployedArtifact 요소를 추가하려면

① 노드의 [CollectionEditor...] 팝업 메뉴를 선택한다.

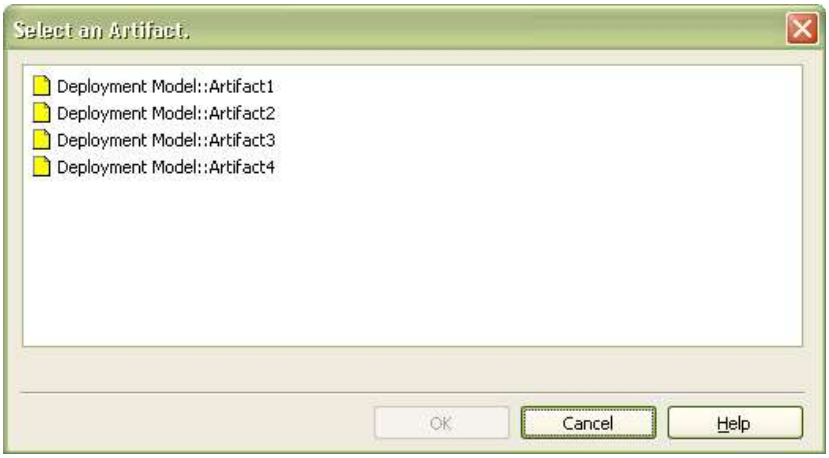
② 또는 properties window에서 DeployedArtifacts 속성의 ... 버튼을 클릭한다.



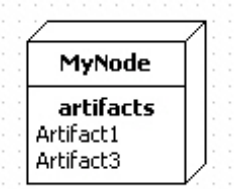
③ collection editor의 deployed artifacts 탭에서  버튼을 이용하여 deployed artifact를 추가한다.



④ [select a artifact] 다이얼로그에서 deployed artifact를 선택하고 OK 버튼을 클릭한다.

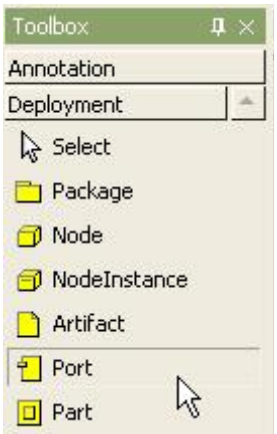


⑤ 그러면 다음과 같이 artifact가 노드에 추가된다.

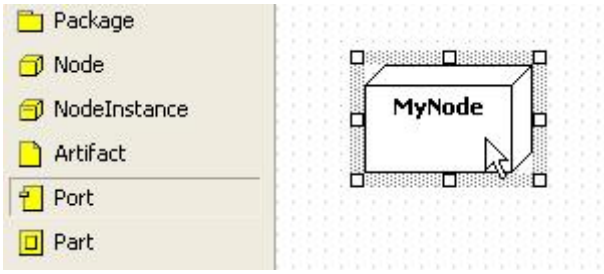


- Port 생성 방법:
- Node에 Port를 생성하려면 ,

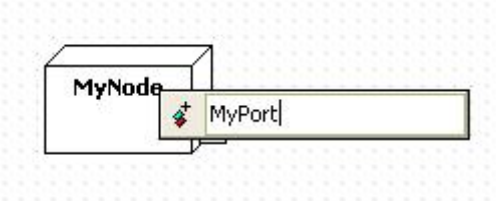
① [Toolbox] -> [Deployment] -> [Port] 버튼을 클릭하고,



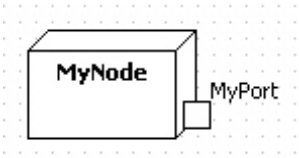
② Main 윈도우에서 Port가 위치할 Node를 클릭한다.



③ 노드 상에 포트가 생성되고 퀵다이얼로그가 나타난다.

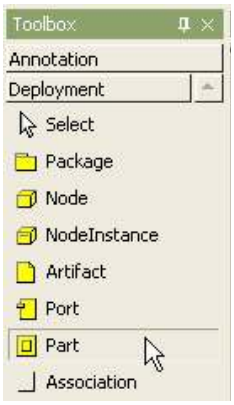


④ 퀵다이얼로그에서 node 이름을 입력하고 [Enter] 키를 누른다. 결과는 다음과 같이 나타난다.

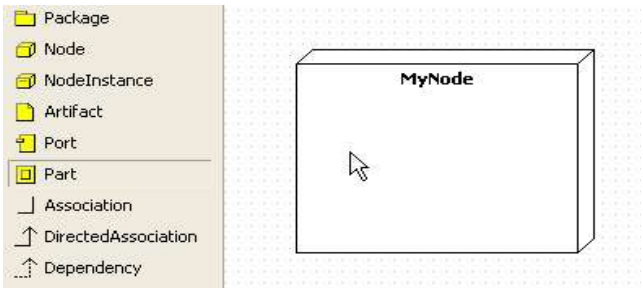


- Part 생성 방법:
- Node에 Part를 생성하려면 ,

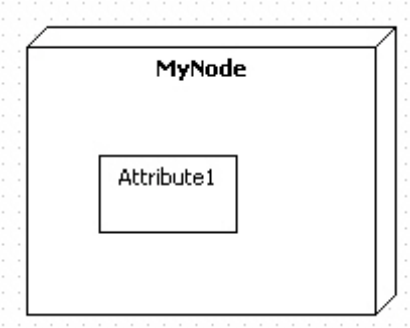
① [Toolbox] -> [Deployment] -> [Part] 버튼을 클릭하고



② Main 윈도우에서 Part가 위치할 Node를 클릭한다.

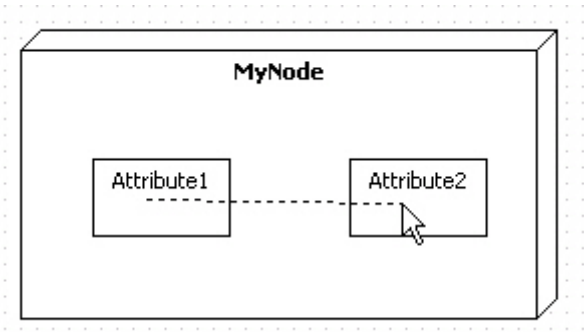


③ Part는 다음과 같이 노드에 생성된다.

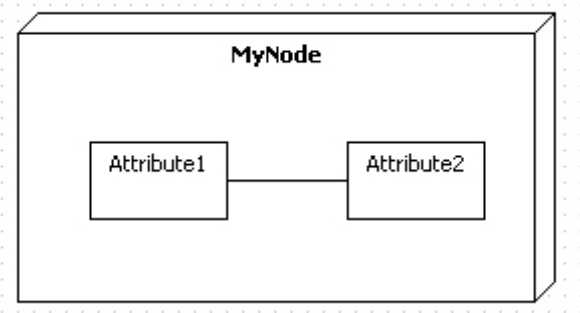


- Connector 생성 방법:
- Connector를 생성하려면,
① [Toolbox] -> [Deployment] -> [Connector] 버튼을 클릭하고

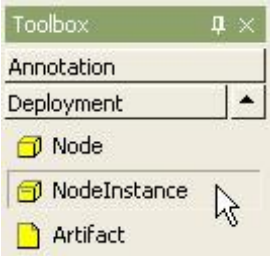
② Main 윈도우에서 연결할 Part(또는 Port)에서 다른 Part(또는 Port)로 마우스 버튼을 누르고 드래그하면 된다.



③ 결과는 다음과 같다.

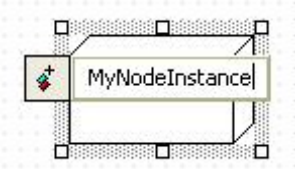


(3) NodeInstance
- 의미:
- 노드 인스턴스(NodeInstance)">노드 인스턴스(Node Instance)는 노드(Node)의 한 사례(Instance)이다. "Classifier" 프로퍼티에 어떤 노드(Node)의 사례 인지를 지정할 수 있다.
- NodeInstance 생성 방법:
- NodeInstance를 생성하려면,
① [Toolbox] -> [Deployment] -> [NodeInstance] 버튼을 클릭하고

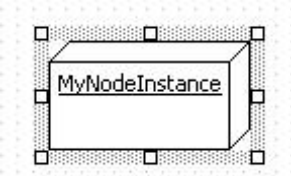


② Main 윈도우에서 NodeInstance가 위치할 곳을 클릭한다.

③ 퀵다이얼로그가 나타나면 node instance의 이름을 입력하고 [Enter] 키를 누른다.



④ 결과는 다음과 같다.



- Attribute Link 추가하기:

- NodeInstance에 AttributeLink를 추가하는 방법은 다음과 같이 2가지 방법이 있다.

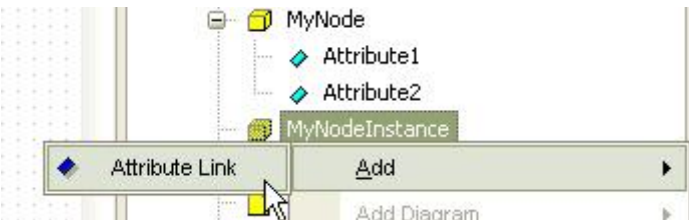
① NodeInstance 또는 Model Explorer의 팝업 메뉴 이용

② Collection Editor 이용

- NodeInstance 또는 Model Explorer의 팝업 메뉴 이용 경우,

① main window 또는 model explorer에서 NodeInstance를 선택한다.

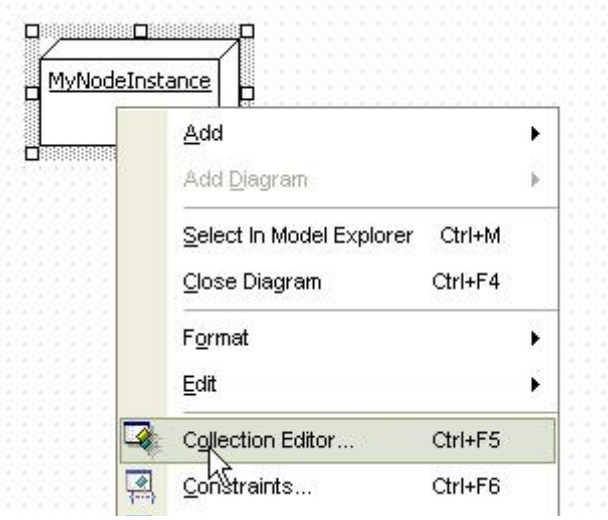
② 오른쪽 마우스 버튼을 눌러서 [Add] -> [Attribute Link] 팝업 메뉴를 선택하여 Attribute Link를 추가할 수 있다.



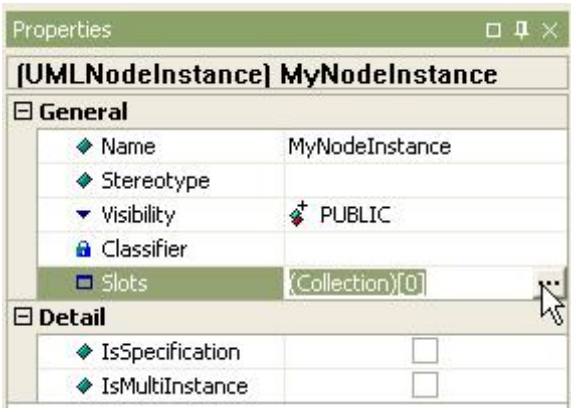
③ The node doesn't show attribute link on the view.

- Collection Editor 이용하는 경우,

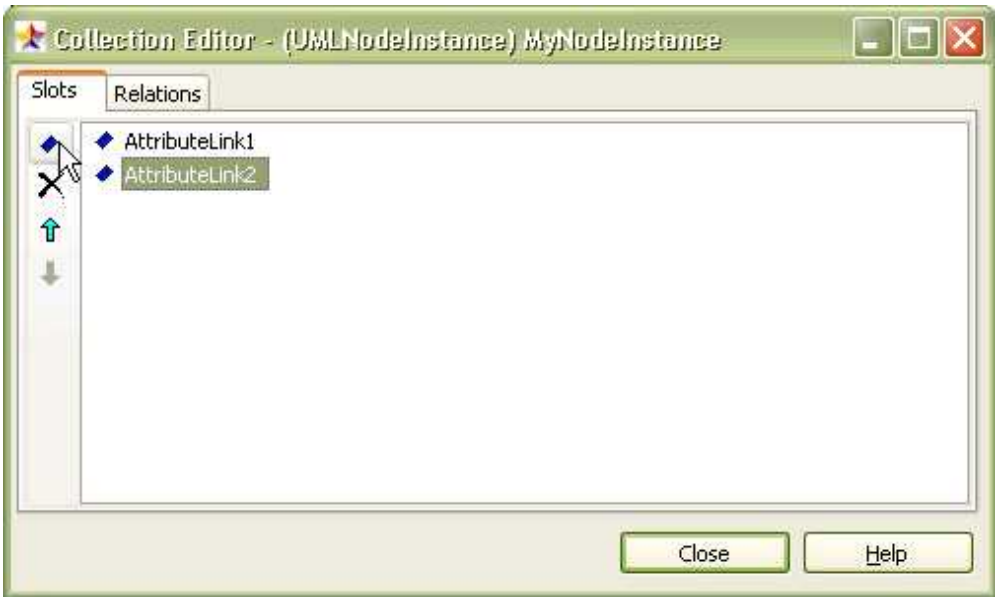
① NodeInstance의 [CollectionEditor...] 팝업 메뉴를 선택하고,



② 또는 properties window에서 slots property의 ... 버튼을 클릭한다.



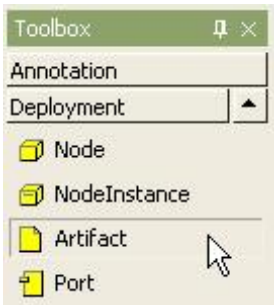
③ collection editor의 slots 탭에서 버튼을 이용하여 attribute link를 추가한다.



(4) Artifact

- Artifact 생성 방법:
- Artifact를 생성하려면,

① [Toolbox] -> [Deployment] -> [Artifact] 버튼을 클릭하고



② Main 윈도우창에서 Artifact가 위치할 곳을 클릭한다.

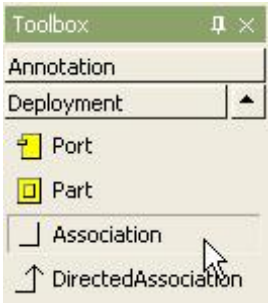
③ 퀵다이얼로그에서 artifact 이름을 입력하고 [Enter] 키를 누른다.

④ 결과는 다음과 같다.

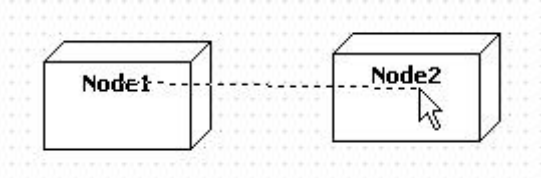
(5) Association

- Association 생성 방법:
- Association를 생성하려면,

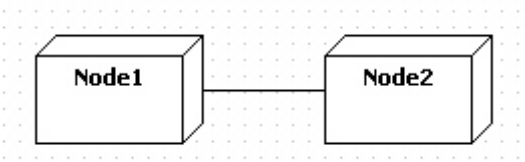
① [Toolbox] -> [Deployment] -> [Association] 버튼을 클릭한다.



② main window에서 연관될 요소사이를 드래그/드롭 한다.



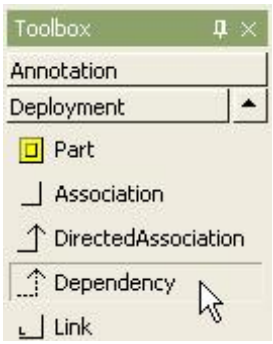
③ 결과는 다음과 같다.



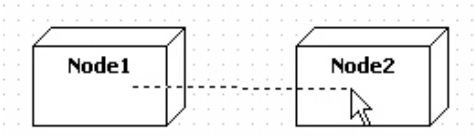
(6) Dependency

- Dependency 생성 방법:
- Dependency를 생성하려면,

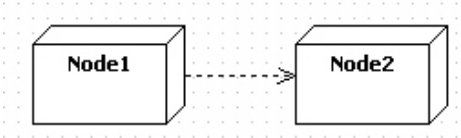
① [Toolbox] -> [Deployment] -> [Dependency] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 의존하는 방향의 다른 요소로 마우스 버튼을 누르고 드래그하면 된다.



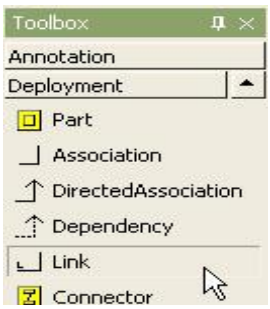
③ 그러면 두개의 요소사이에 dependency가 다음과 같이 생성된다.



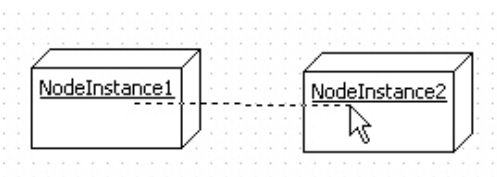
(7) Link

- Link 생성 방법:
- Link를 생성하려면,

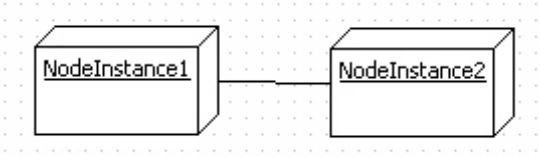
① [Toolbox] -> [Deployment] -> [Link] 버튼을 클릭하고.



② Main 윈도우창에서 연결할 NodeInstance에서 다른 NodeInstance로 마우스 버튼을 누르고 드래그하면 된다.



③ 그러면 두개의 node instances 사이에 link가 생성된다.



아. Composite Structure 다이어그램 모델링하기

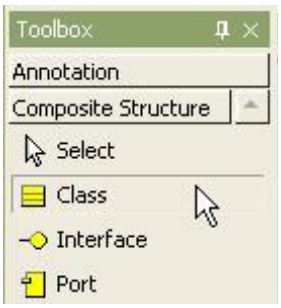
Composite Structure 다이어그램에서 편집할 수 있는 요소들은 다음과 같다.

- Class
- Interface
- Port
- Part
- Dependency
- Connector

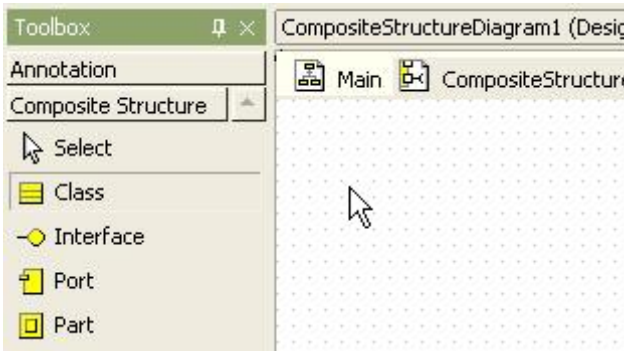
(1) Class

- Class 생성 방법:
- Composite Structure 다이어그램에서 Class를 생성하려면,

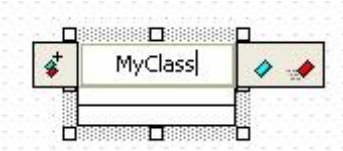
① [Toolbox] -> [Composite Structure] -> [Class] 버튼을 클릭하고,



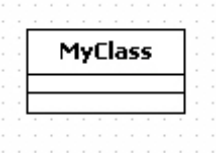
② Main 윈도우창에서 Class가 위치할 곳을 클릭한다.



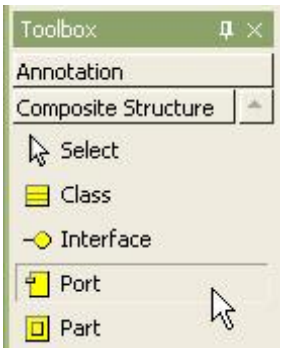
③ 퀵다이얼로그에서 클래스 이름을 입력하고,



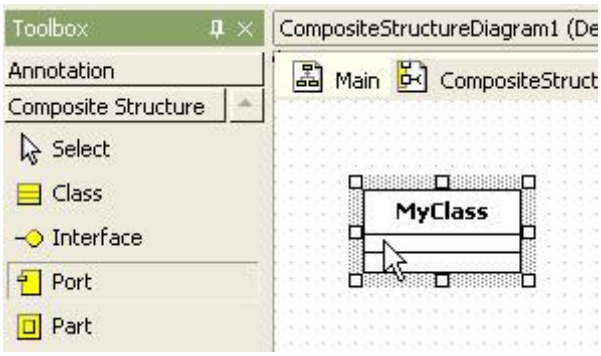
④ 그리고 [Enter] 키를 누른다. 그러면 클래스가 생성이 완료된다.



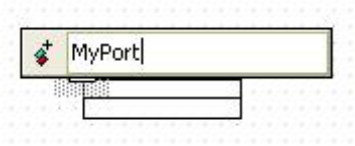
- Port 생성 방법:
- Component에 Port를 생성하려면,
① [Toolbox] -> [Composite Structure] -> [Port] 버튼을 클릭하고,



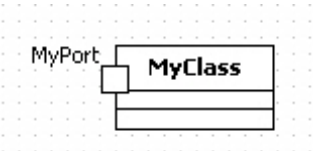
② Main 윈도우에서 Port가 위치할 Component를 클릭한다.



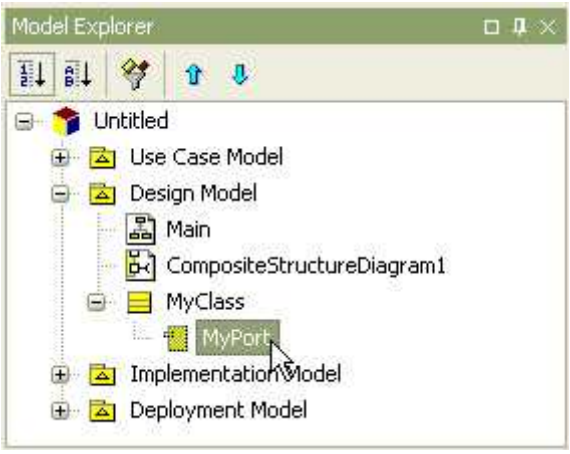
③ 그리고 퀵다이얼로그에서 포트의 이름을 입력한다.



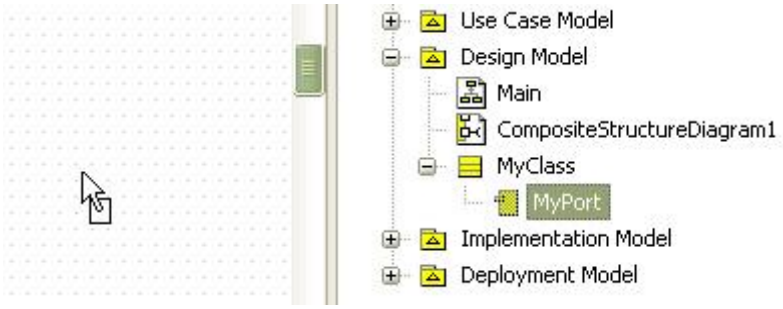
④ [Enter] 키를 누르면, 포트가 생성이 완료된다.



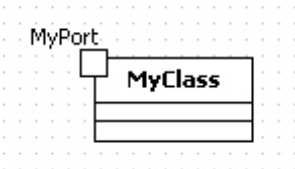
- Port 드래그를 통한 뷰 생성 방법:
Model Explorer로부터 드래그를 통하여 Port를 생성할 수 있다. 이때에는 드래그한 Port를 Component위로 드롭 한다.
① Model explorer에서 포트를 선택한다.



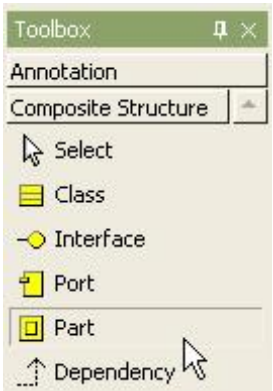
② 선택된 포트를 composite structure 다이어그램상의 클래스에 드래그 드롭 한다.



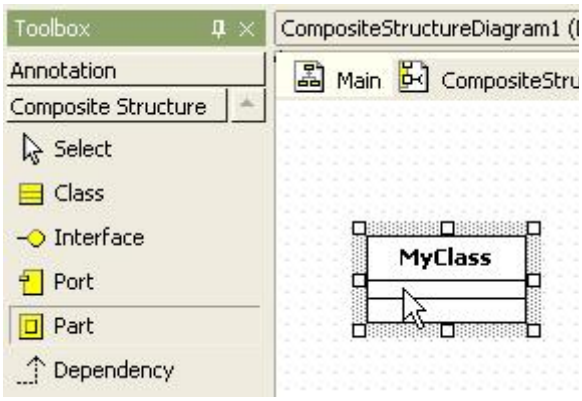
③ 만약 Component가 아닌 Diagram 위로 드롭하면 Port와 함께 Component의 뷰가 생성된다.



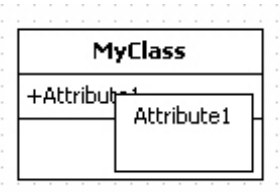
- Part 생성 방법:
- Component에 Part를 생성하려면,
① [Toolbox] -> [Composite Structure] -> [Part] 버튼을 클릭한다.



② Main 윈도우에서 Part가 위치할 Component를 클릭한다.



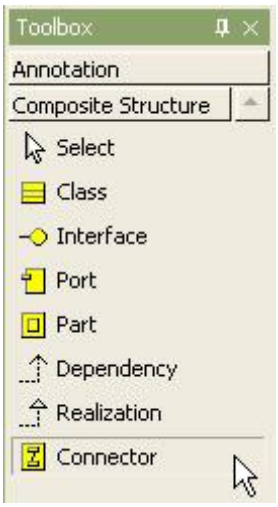
③ 그러면 클래스 속에 파트가 생성된다.



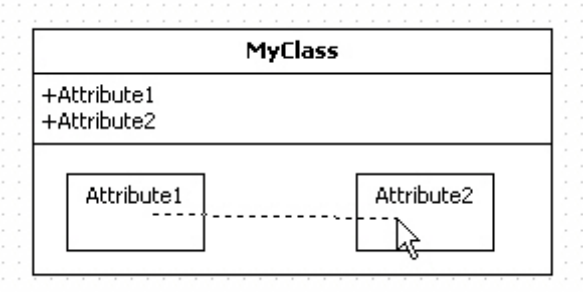
- Connector 생성 방법:

- Connector를 생성하려면, Toolbox>Component의 Connector 버튼을 클릭하고 하면 된다.

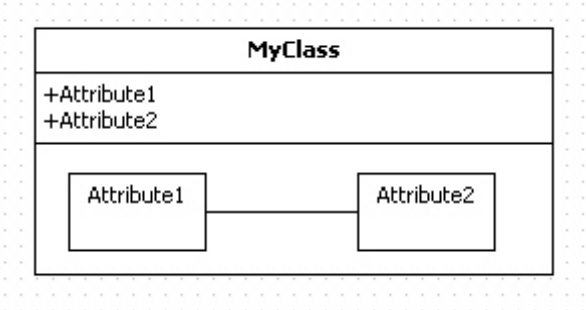
① [Toolbox] -> [Composite Structure] -> [Connector] 버튼을 클릭하고,



② Main 윈도우창에서 연결할 Part(또는 Port)에서 다른 Part(또는 Port)로 마우스 버튼을 누르고 드래그 한다.



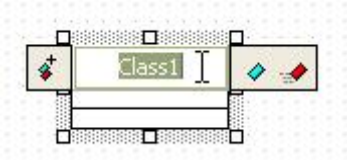
③ 그러면 두개의 파트사이에 커넥터가 생성된다.



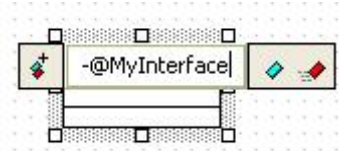
- Interface Providing 관계 생성 방법:

현재 선택된 Class로부터 Interface Providing 관계를 만들려면 단축 생성 구문을 사용한다.

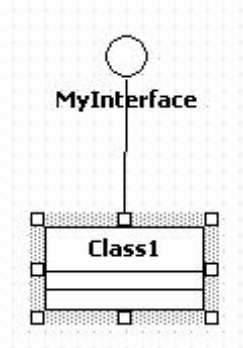
① Class를 더블 클릭해서 Quick Dialog가 나타나면,



② Quick Dialog에서 "-@" 문자열 다음에 Interface의 이름을 입력한다. 여러 개의 Interface를 제공한다면 각 Interface 이름은 "," 문자로 구분해서 입력한다.



③ 그리고 [Enter] 키를 입력하면 선택된 클래스에서 제공되는 인터페이스가 생성되고 자동으로 배치된다.

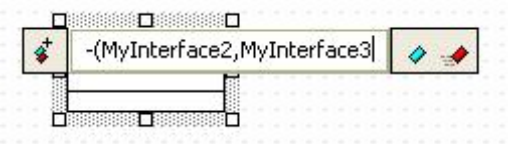


- Interface Requiring 관계 생성 방법:

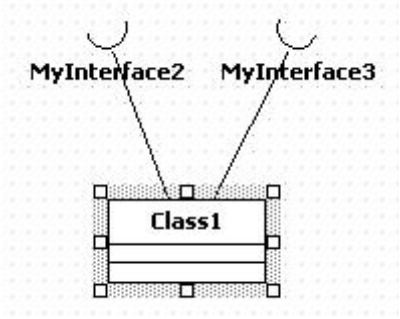
현재 선택된 Class로부터 Interface Requiring 관계를 만들려면 단축 생성 구문을 사용한다.

① Class를 더블 클릭해서 Quick Dialog가 나타나면,

② Quick Dialog에서 "-(" 문자열 또는 "-->" 문자열 다음에 Interface의 이름을 입력한다. 여러 개의 Interface를 요구한다면 각 Interface 이름은 "," 문자로 구분해서 입력한다.



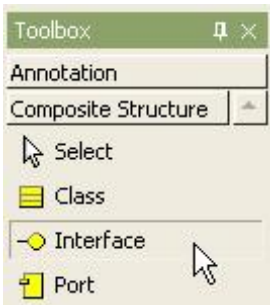
③ 그리고 [Enter]키를 누르면 선택된 Class와 Interface Requiring 관계를 가지는 Interface들이 생성되거나 연결되고 자동 배열되어 생성된다.



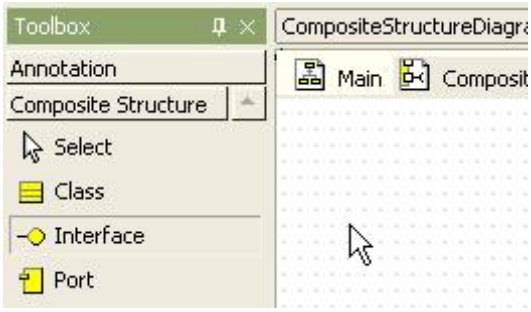
(2) Interface

- Interface 생성 방법:
- Interface를 생성하려면,

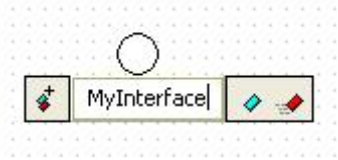
① [Toolbox] -> [Composite Structure] -> [Interface] 버튼을 클릭하고



② Main 윈도우창에서 Interface가 위치할 곳을 클릭한다.

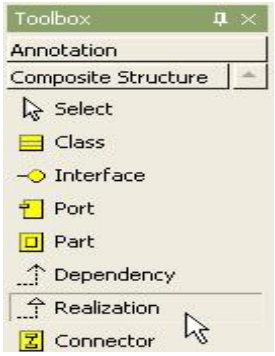


③ 퀵다이얼로그에서 인터페이스의 이름을 입력한다.

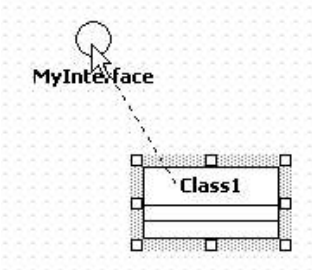


④ 그리고 [Enter] 키를 누르면 인터페이스 생성이 완료된다.

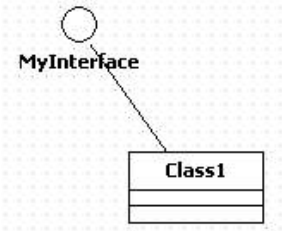
- Interface Providing 관계 설정 방법:
- Interface Providing 관계를 설정하려면,
① [Toolbox] -> [Composite Structure] -> [Realization] 버튼을 클릭하고,



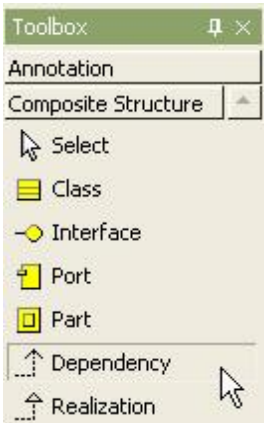
② Main 윈도우창에서 요소(Class, Port, Part, Package, Subsystem)를 선택하고 Interface로 마우스를 누르고 드래그하면 된다.



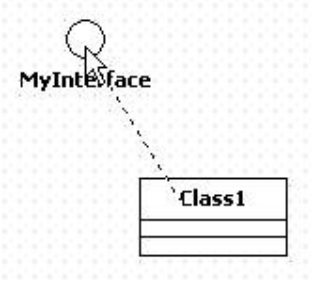
③ 그러면 providing interface 관계가 생성된다.



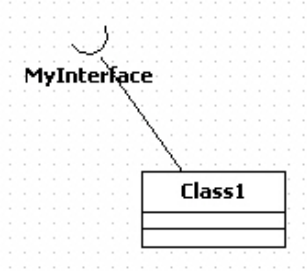
- Interface Requiring 관계 설정 방법:
- Interface Requiring 관계를 설정하려면,
① [Toolbox] -> [Composite Structure] -> [Dependency] 버튼을 클릭하고



② Main 윈도우창에서 요소(Class, Port, Part, Package, Subsystem)를 선택하고 Interface로 마우스를 누르고 드래그하면 된다.



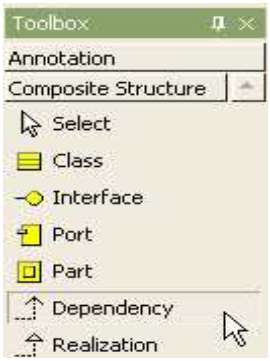
③ 그러면 interface requiring 관계가 생성된다.



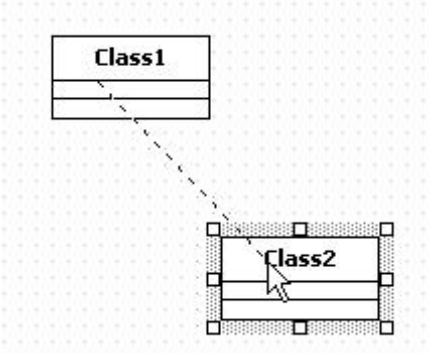
(3) Dependency

- Dependency 생성 방법:
- Dependency를 생성하려면,

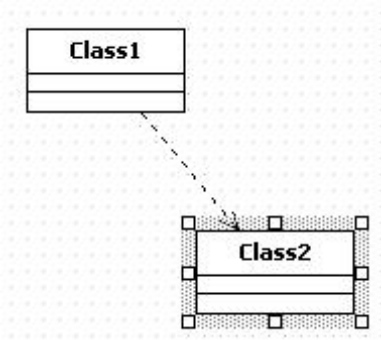
① [Toolbox] -> [Composite Structure] -> [Dependency] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 의존하는 다른 요소로 마우스 버튼을 누르고 드래그하면 된다.



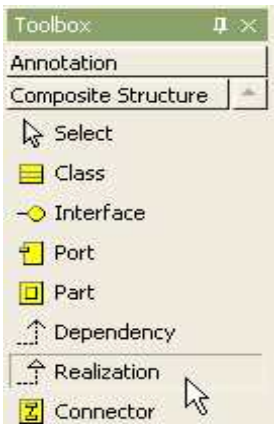
③ 그러면 의존 관계가 생성된다.



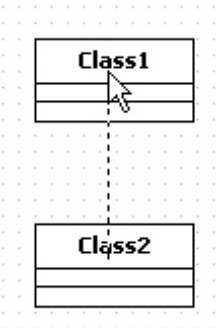
(4) Realization

- Realization생성 방법:
- Realization을 생성하려면,

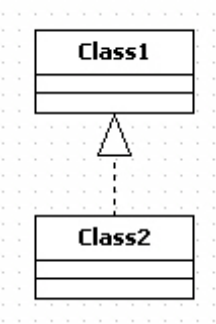
① [Toolbox] -> [Composite Structure] -> [Realization] 버튼을 클릭하고



② Main 윈도우창에서 요소에서 Realization할 다른 요소로 마우스 버튼을 누르고 드래그하면 된다.



③ 그러면 다음과 같이 두 요소 사이에 realization 관계가 생성된다.



(5) Collaboration

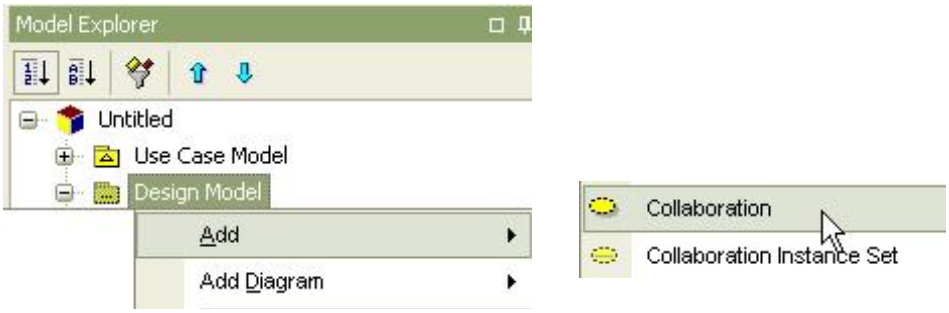
- 의미:

협동(Collaboration)은 연산(Operation) 혹은 클래스류들이 어떻게 실체화(realize)되는지를 표현한다. 협동은 객체와 링크들이 수행할 역할(role)들의 집합으로써 구성되어 진다.

- Collaboration 생성 방법:

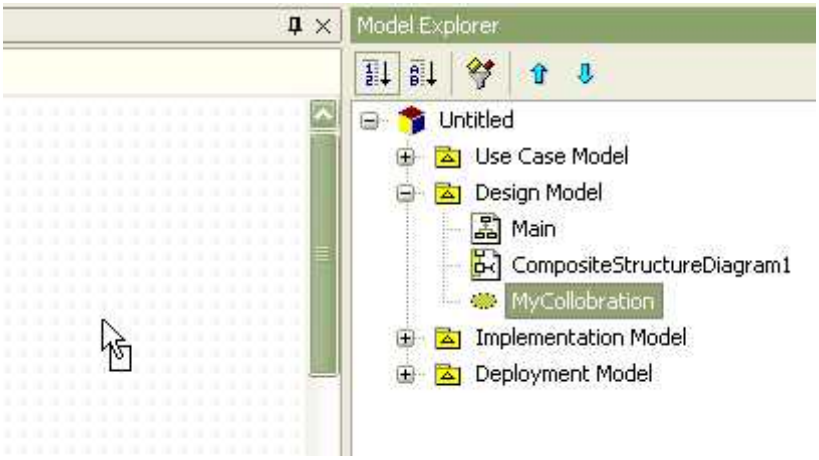
- Collaboration을 생성하려면

① Model Explorer에서 패키지를 선택한다. 그리고 [Add] -> [Collaboration] 팝업 메뉴를 선택한다.

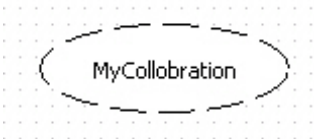


② 그러면 collaboration이 생성되어진다. 이때 collaboration의 이름을 입력한다.

③ 그리고 collaboration을 main window로 드래그/드롭 한다.



④ 그러면 collaboration이 다이어그램에 위치하게 된다.



5. 코드와 문서 생성하기

이 장에서는 Generator를 효과적으로 사용하기 위해 필요한 Generator의 기본 개념과 Generator를 이용한 문서 생성 방법, 배치 처리 방법에 대해서 알아본다.

가. 기본 개념

(1) Generator란 무엇인가?

StarUML Generator는 StarUML의 모델요소들(UML Model Elements)과 템플릿 문서(Tempate document)를 이용하여 각기 다른 형태의 산출물(MS Word, MS Excel, MS PowerPoint...)로 생성해주는 에드인이다. 사용자 스스로가 얼마든지 새로운 템플릿을 정의하고 기존 템플릿을 변형하여 사용할 수 있게 구성되어 있으므로, 하나의 모델로부터 여러 가지 템플릿을 이용하여 다양한 형태의 산출물을 쉽고 빠르게 생성할 수 있다.

(2) Key Features

StarUML Generator는 다음과 같은 기능들을 제공한다.

- 사용자 정의 템플릿

사용자가 직접 해당 포맷의 템플릿을 작성할 수 있다. 별도의 복잡한 포맷 디자이너를 사용하지 않고도, doc(워드파일), xls(엑셀파일), ppt(파워포인트파일)파일이 곧 바로 템플릿이 되므로 쉽게 작성할 수 있다.

- 템플릿 파라미터 지원

정의된 템플릿이 사용자의 환경이나 의도에 따라 변수를 적용할 수 있도록 파라미터 지정을 제공하고 있다. 이를 통해 조그만 변화 때문에 일일이 템플릿을 수정해야 하는 불편함을 제거하고 단순히 파라미터의 값만 지정함으로써 템플릿의 유연성을 확보할 수 있도록 제공한다.

- 산출물을 일괄 생성하는 배치처리

정의된 템플릿이 사용자의 환경이나 의도에 따라 변수를 적용할 수 있도록 파라미터 지정을 제공하고 있다. 이를 통해 조그만 변화 때문에 일일이 템플릿을 수정해야 하는 불편함을 제거하고 단순히 파라미터의 값만 지정함으로써 템플릿의 유연성을 확보할 수 있도록 제공한다.

- 머리말/바닥글 및 다양한 스타일 적용 가능

머리말과 바닥글 부분에 정보를 입력할 수 있으며, 워드 고유의 스타일 적용이 모두 가능해지므로 사용자가 원하는 모든 포맷의 템플릿을 작성할 수 있어서 곧 바로 출력만 하면 되는 양질의 문서 생성이 가능해진다.

- 엑셀의 셀 반복 및 데이터 입력

엑셀의 셀을 반복하여 설계의 각종 데이터를 수집하여 입력할 수 있다. 이를 통해 측정 정보를 다양한 그래프로 표현할 수 있다. 또한 여러 가지 기준에 따른 필터링이나 정렬을 적용할 수 있어서 분석과 보고서에 활용할 수 있다.

- 파워포인트 슬라이드 작성

파워포인트는 슬라이드를 다 계층으로 반복하여 생성할 수 있으며, 슬라이드에는 다이어그램 및 각종 정보들을 자유자재로 삽입 할 수 있다. 이를 통해 검토회의 및 각종 프레젠테이션용 산출물들을 자동으로 생성할 수 있어서, 발표를 위한 별도의 작업 노력을 크게 줄일 수 있다.

- 텍스트-기반의 다양한 산출물 생성

텍스트 기반의 프로그램 코드, XML, HTML 등의 어떠한 산출물이라도 템플릿 기반으로 생성할 수 있다.

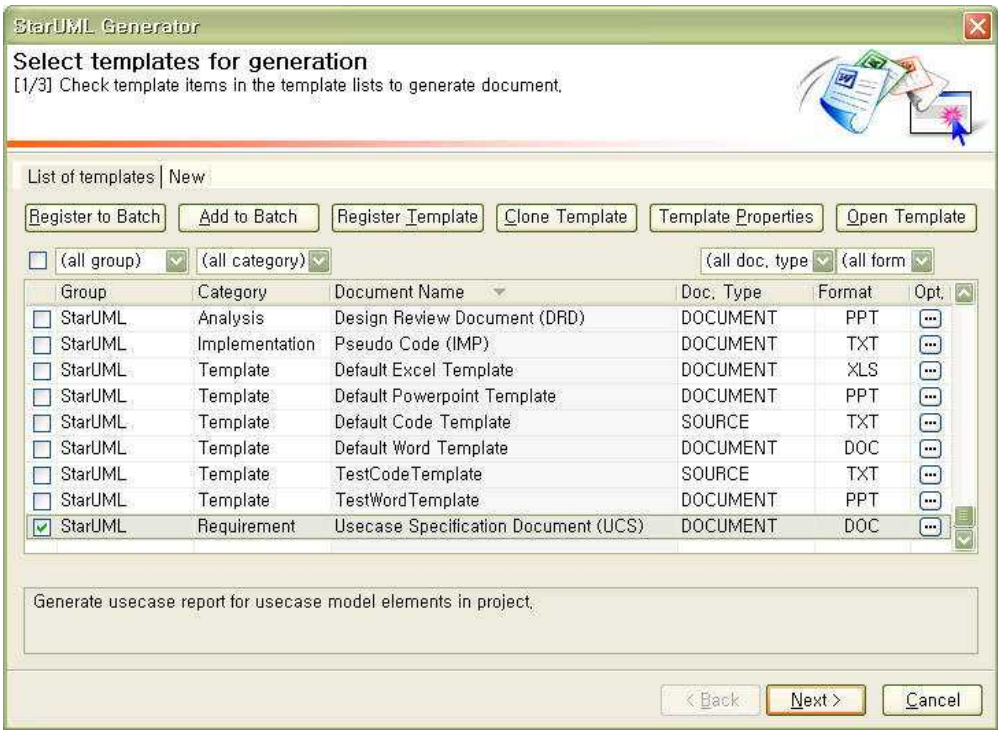
나. 템플릿으로부터 생성

(1) 문서 생성하기

템플릿으로부터 문서를 생성하기 위해서는 현재 작업 중인 프로젝트가 템플릿과 대응되는 UML 모델이어야 한다.

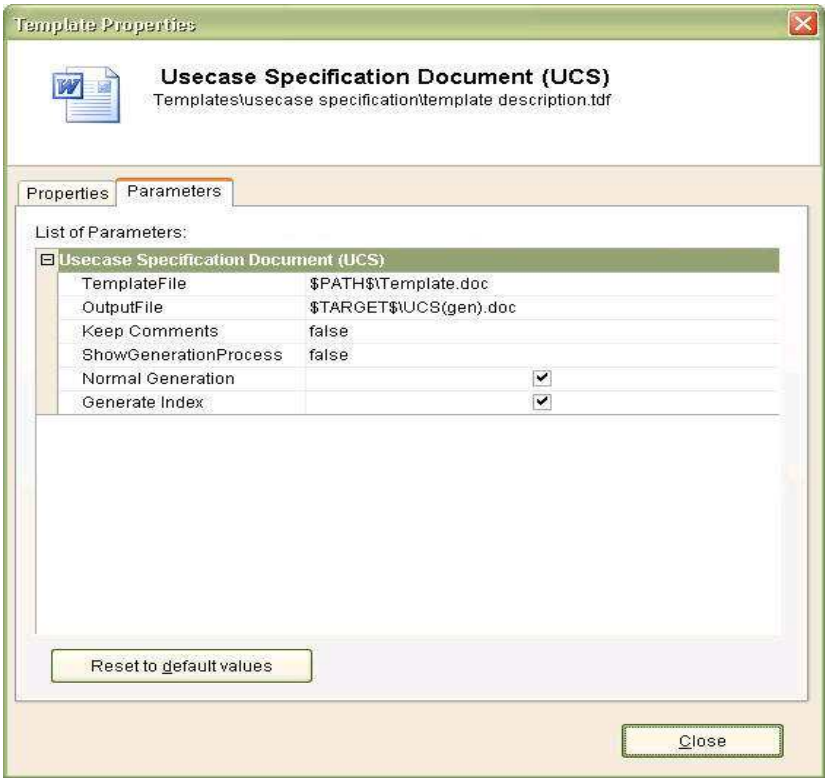
① [Tools]->[StarUML Generator...] 메뉴를 선택한다.

② [Select templates for generation] 페이지가 화면에 나타나면 List of templates에서 생성할 템플릿의 체크박스를 선택하고 [Next] 버튼을 클릭한다.

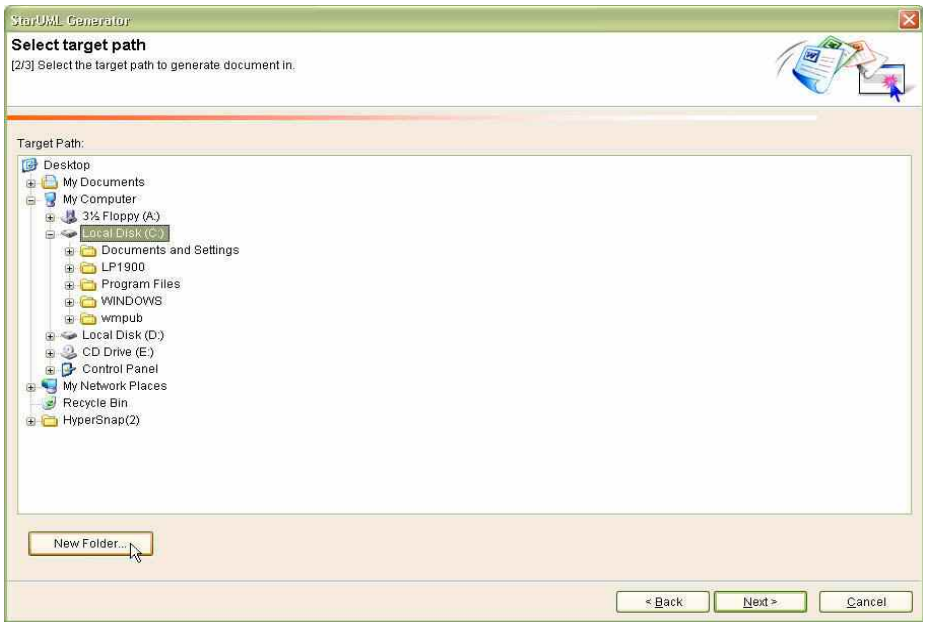


③ 템플릿 자체의 정보를 변경하지 않고 현재의 문서 생성에만 사용자가 원하는 인자가 적용되게 설정할 수 있다. 만약 생성 인자 설정을 변경하려면, 선택한 템플릿의

마지막 그리드에 있는 [...] 버튼을 클릭한다. 그리고 템플릿 인자의 값을 원하는 설정으로 변경한다. (생성 인자에 대한 자세한 내용은 템플릿 등록 > 파라미터 정보를 참조한다)



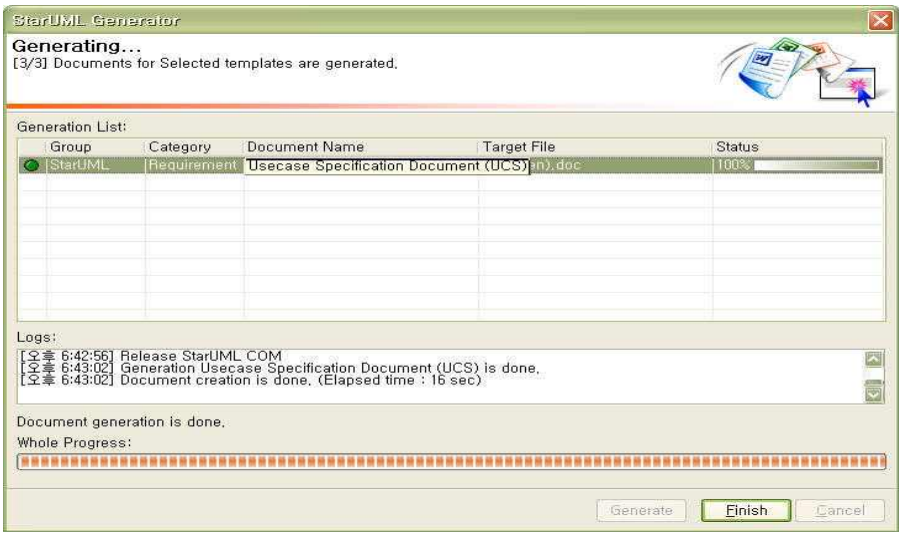
④ [Select target path] 페이지가 화면에 나타나면 생성될 문서가 저장될 폴더를 선택하고 [Next]를 클릭한다.



만약 현재 선택된 폴더의 하위에 폴더를 추가하려면, 왼쪽 하단의 [New Folder...] 버튼을 클릭하고, 이름 설정 다이얼로그에서 추가될 폴더의 이름을 입력한다.

⑤ [Generating...] 페이지가 화면에 나타나면 [Generate] 버튼을 클릭한다.
템플릿으로부터 문서가 생성되면서, 각 템플릿의 생성 진행 정도를 진행 바를 통해서

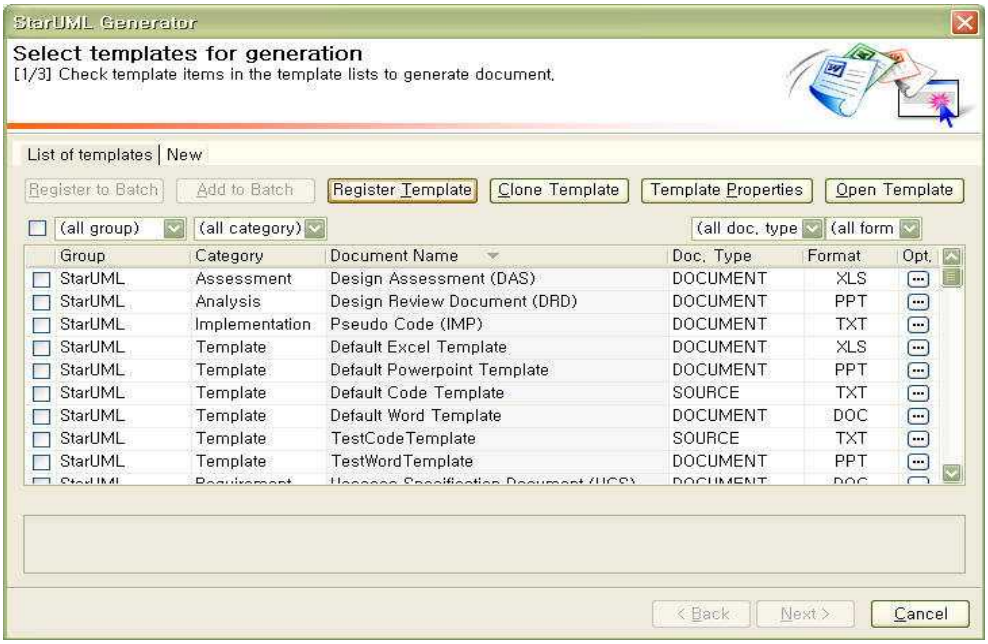
확인할 수 있다. 그리고 생성 과정의 로그는 [Logs:] 창에 기록된다. 만약 현재 생성중인 문서를 취소하려면 [Cancel] 버튼을 클릭한다. 그리고 취소 확인 다이얼로그에서 확인 버튼을 클릭한다.
문서 생성이 완료되면 로그 창에 생성 완료에 대한 로그(Document Creation is done)가 기록되고 [Finish] 버튼이 활성화된다. 문서 생성을 완료하려면 [Finish] 버튼을 클릭하여 문서 생성 과정을 종료한다. 생성된 문서를 확인하기 위해서는 [Generation List]에서 문서 목록을 더블 클릭하여 생성되어진 문서를 바로 열어 확인할 수 있다.



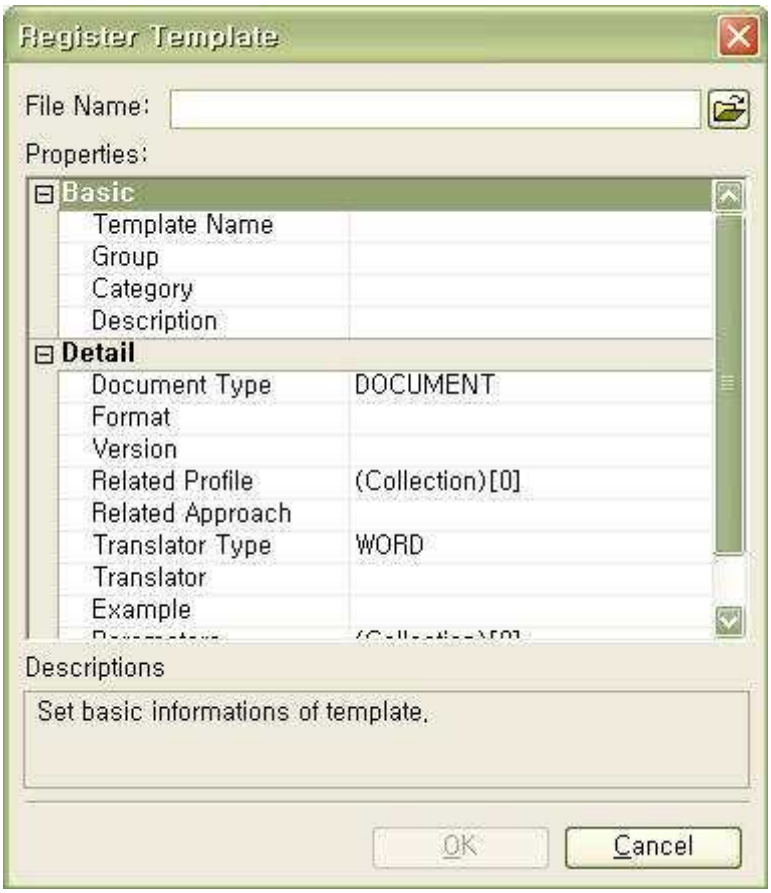
(2) 템플릿 등록하기

사용자가 직접 작성한 자신만의 템플릿 문서를 Generator에 등록할 수 있다.

① [Select templates for generation] 페이지에서 [Register Template] 버튼을 클릭한다.



② [Register Template] 다이얼로그가 나타나면,  버튼을 클릭하여 템플릿 정보가 저장되어질 파일의 위치를 선택한다.



③ [Properties:] 입력창에서 등록할 템플릿의 정보들을 입력하고 [OK] 버튼을 클릭하여 템플릿을 등록한다.

- 기본 정보

템플릿 등록 시에 템플릿명과 그룹, 분류, 설명에 대한 기본정보를 설정한다.

항목	설명
Template Name	등록될 템플릿 명을 지정한다.
Group	템플릿이 속하는 그룹을 지정한다. 기존에 있는 그룹 외에 새로운 그룹을 정의할 수 있다.
Category	프로젝트 관리, 요구사항 등의 분류를 지정한다.
Description	템플릿의 개략적인 설명을 기술한다.

- 상세정보

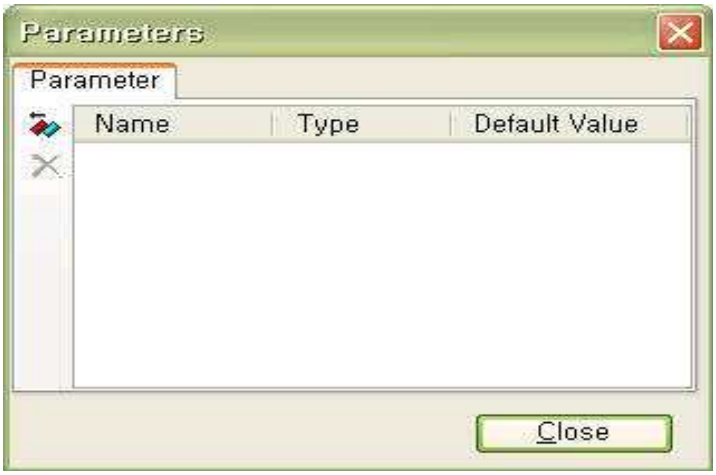
템플릿 등록 시에 필요한 상세 정보들을 설정한다.

항목	설명
Document Type	템플릿의 문서 형태를 지정한다. DOCUMENT(문서), CODE(소스코드) 중 하나를 선택한다.
Format	문서 생성의 결과로 생성될 문서의 포맷을 지정한다.
Version	템플릿의 버전 정보를 "1.0"과 같은 형식을 기록한다.
Related Profile	템플릿을 적용할 모델에 포함할 프로파일들을 선택한다.
Related Approach	템플릿을 적용할 모델의 기본 접근법을 선택한다.
Translator Type	문서 생성기의 종류를 명시한다. WORD(워드 문서 생성기), EXCEL(엑셀 문서 생성기), POWERPOINT(파워포인트 문서 생성기), TEXT(텍스트 문서 생성기), COM(COM 컴포넌트 기반의 사용자 정의 생성기), SCRIPT(스크립트 기반의 사용자 정의 생성기), EXE(실행파일 기반의 사용자 정의 생성기) 중 하나를 선택한다.
Translator	문서 생성기의 이름을 명시한다. 빌트인 문서 생성기(WORD, EXCEL, POWERPOINT, TEXT)는 지정하지 않다. 사용자 정의 생성기는 생성기의 파일명을 적는다.
Example	문서 생성에 맞는 예제 모델 파일을 지정한다.
Parameters	문서 생성에 필요한 인자를 정의한다.
Related files	문서 생성에 필요한 부가적인 파일들을 지정한다.

- 파라미터 정보

① Parameters 항목에서 ··· 버튼을 클릭한다.

② [Parameters] 다이얼로그가 나타나면, 새로운 파라미터를 삽입하기 위해서  버튼을 클릭하며, 파라미터를 삭제하기 위해서는  버튼을 클릭한다.



③ [New Parameter] 다이얼로그가 나타나면 파라미터의 이름과 타입, 그리고 디폴트값을 입력하고 [OK] 버튼을 클릭한다.

- Translator Type에 따른 기본 파라미터 목록을 다음과 같이 설정한다.

항목	타입	Translator Type	설명
TemplateFile	FILENAME or STRING	WORD,EXCEL, POWERPOINT	문서 생성 시에 사용될 템플릿 문서 파일명을 지정한다.
OutputFile	FILENAME or STRING	WORD,EXCEL, POWERPOINT, TEXT	문서 생성의 결과로 생성될 파일명을 지정한다.
Keep Comment	BOOLEAN	WORD,EXCEL, POWERPOINT	생성된 문서 내에 커맨드 정보를 남길지 여부를 지정한다.
ShowGenerationProcess	BOOLEAN	WORD,EXCEL, POWERPOINT	MS Office에 일어나는 생성 과정을 보여줄지 여부를 지정한다. 이 항목이 true값으로 설정되면 생성시 성능이 떨어질 수 있다.
Normal Generation	BOOLEAN	WORD	커맨드의 상대 경로를 최상위 프로젝트로 지정한다. 만약 false로 설정되면 현재 모델에서 선택된 요소에서부터 상대 경로가 지정된다.
Generate Index	BOOLEAN	WORD	생성된 문서 내에서 색인 요소를 검색하여, 문서 색인을 생성한다.
intermediate	STRING	TEXT	코드 생성에 필요한 중간 코드를 저장할 파일명을 지정한다.
target	STRING	TEXT	생성되는 코드 파일이 여러 개일 경우에 코드 파일들이 존재할 최상위 경로명을 지정한다.

※ 참고

- 파일명과 관련된 파라미터 값을 설정하는 경우에는 StarUML Generator에 명시된

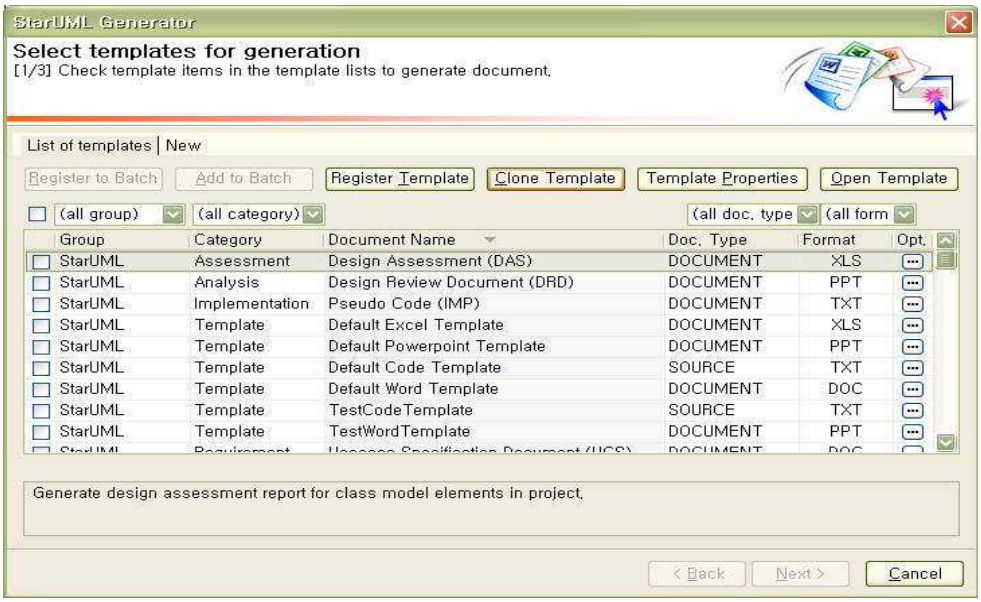
상수를 사용할 수 있다. 상수에는 다음과 같은 것들이 있다.

이름	설명
$PATH$	현재 템플릿 정보와 템플릿이 존재하는 디렉터리 경로를 의미한다. (예) $PATH$\BusinessActorReport.doc
$GROUP$	그룹명을 의미한다.
$CATEGORY$	분류 명을 의미한다.
$NAME$	템플릿 명을 의미한다.
$TARGET$	**[Generator]** 다이얼로그에서 사용자가 지정한 문서 생성 경로를 의미한다.

(3) 템플릿 복제하기

- 기존에 등록되어진 템플릿을 복제하여 새로운 템플릿을 생성할 수 있다.

① [Select templates for generation] 다이얼로그에서 복제를 원하는 템플릿을 선택하고 [Clone Template] 버튼을 클릭한다. 또는 템플릿을 선택하고 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Clone Template] 메뉴 아이템을 선택한다.



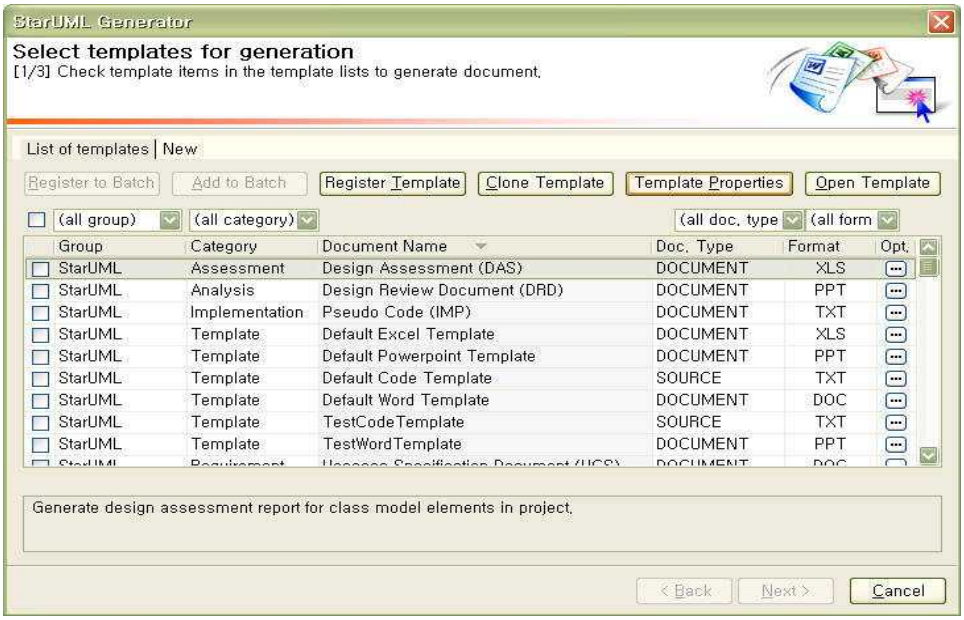
② [Clone Template] 다이얼로그가 나타나면 [New Template Name] 입력창에서 템플릿의 이름과, [New Template Folder] 입력창에서 템플릿 리소스들이 저장될 상위 폴더를 입력하고, [OK] 버튼을 클릭하여 템플릿을 복제한다.

③ [List of templates] 목록에 복제된 템플릿이 추가된다. 추가된 템플릿을 선택하여 템플릿의 정보를 수정한다. (템플릿 정보 수정에 대한 자세한 내용은 템플릿 정보 보기 및 수정을 참조한다)

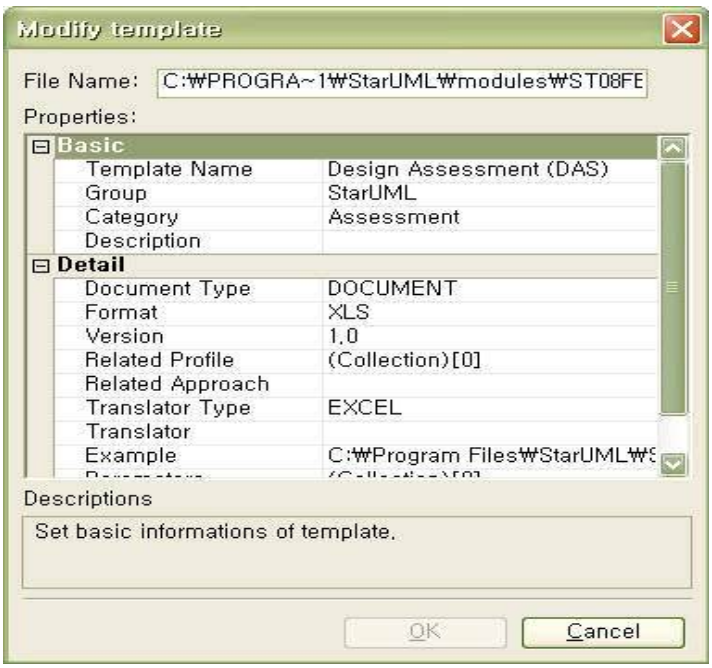
(4) 템플릿 정보 보기 및 수정

- 등록된 템플릿의 정보들을 보거나 수정할 수 있다.

① [Select templates for generation] 다이얼로그에서 수정을 원하는 템플릿을 선택하고 [Template properties] 버튼을 클릭한다. 또는 템플릿을 선택하고 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Show Template Properties] 메뉴 아이템을 선택한다.



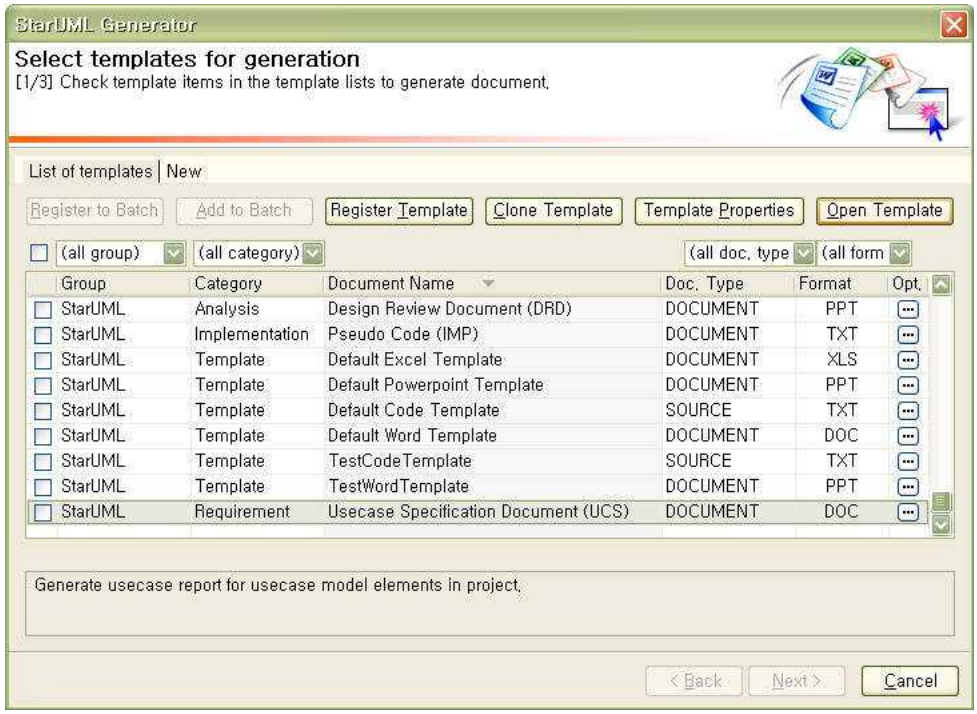
② [Modify Template] 다이얼로그가 나타나면 [Properties:] 입력창에서 템플릿의 정보들을 수정하고 [OK] 버튼을 클릭하여 템플릿을 수정을 종료한다. (정보 수정에 대한 자세한 내용은 템플릿 등록 > 기본/상세/파라미터 정보를 참조한다)



(5) 템플릿 문서 열기 및 편집하기

- 등록된 템플릿 문서를 열어서 사용자가 원하는 형태로 편집할 수 있다.

① [Select templates for generation] 다이얼로그에서 편집을 원하는 템플릿 문서를 선택하고 [Open Template] 버튼을 클릭한다. 또는 템플릿을 선택하고 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Open Template] 메뉴 아이템을 선택한다.



② 선택된 템플릿 문서가 열리면 사용자가 원하는 형태로 템플릿 문서를 편집한다. 생성에 사용되는 커맨드는 MS Office의 Comment에 기록되어 있으며, 기본적으로 REPEAT, DISPLAY, IF, SCRIPT이라는 스트링으로 시작하며, 그 다음에 인자가 연속적으로 오는 형태를 취한다. (템플릿 문서 수정을 위한 자세한 내용은 StarUML 5.0 개발자 가이드 > 11장. Writing Templates를 참조한다)

(6) 템플릿 삭제하기

- [Select templates for generation] 다이얼로그에서 등록된 템플릿 중에서 삭제를 원하는 템플릿을 선택하고, 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Delete Template] 메뉴 아이템을 클릭한다.
- 삭제된 템플릿과 관련된 리소스 파일과 폴더가 삭제되므로, 삭제된 템플릿을 다시 복원할 수 없으므로 주의해서 삭제한다.

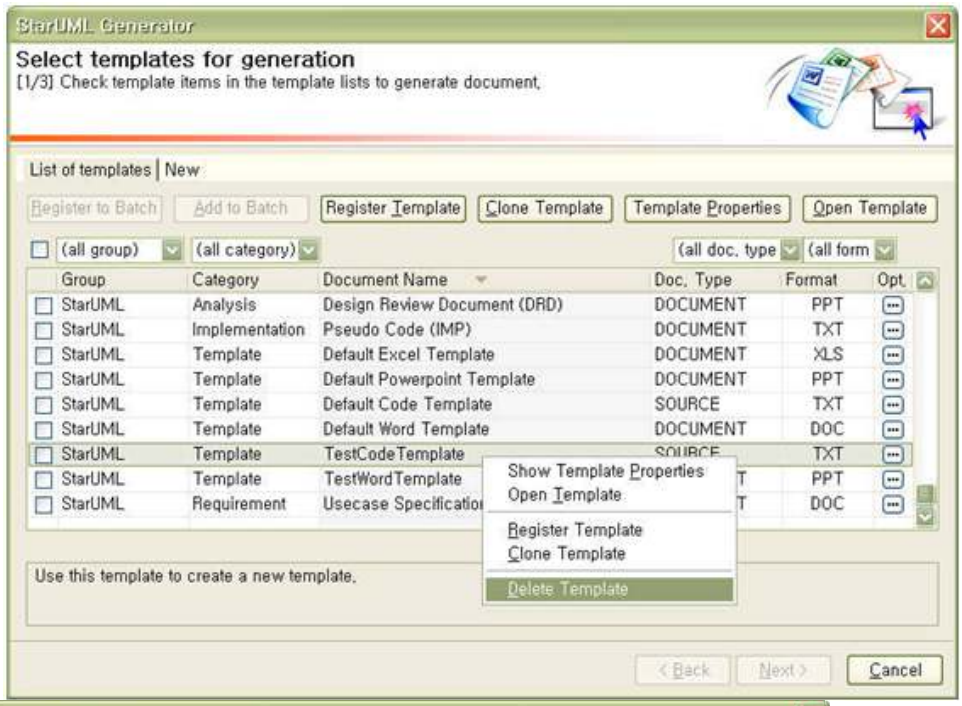
다. 배치 사용하기

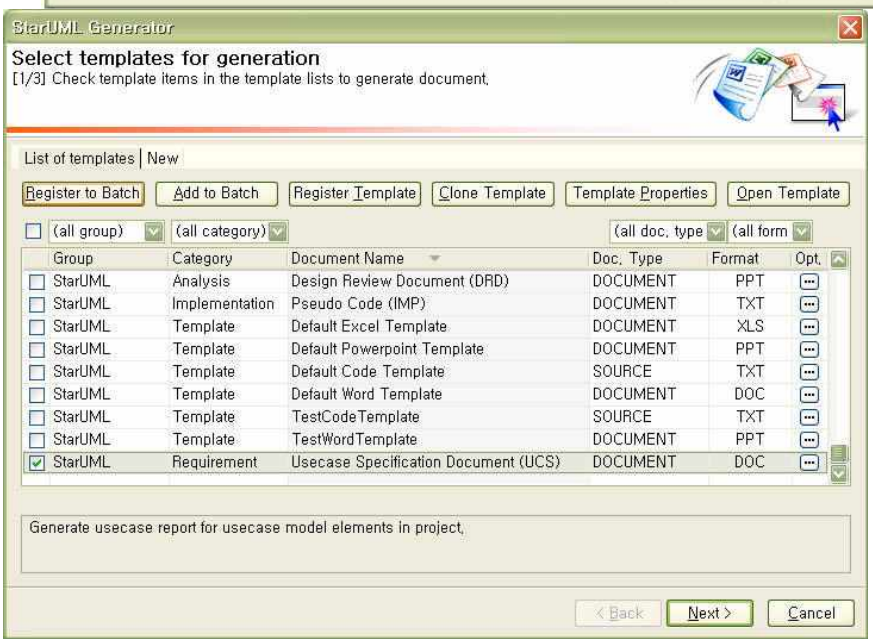
List of templates에는 Generator에 등록된 모든 템플릿들이 나열된다. 만약 사용자가 특정 템플릿에 대해서 빈번하게 문서 생성을 수행해야 한다면, 이러한 템플릿들을 하나의 배치 작업으로 등록하여 다음 문서 생성시에 별도의 템플릿 선택 과정 없이 배치만을 선택하여 문서를 생성할 수 있게 해준다.

(1) 템플릿을 새로운 배치에 추가

- 새로운 배치 작업을 만들고, 현재 선택된 템플릿을 새로 만들어진 배치 작업에 등록한다.

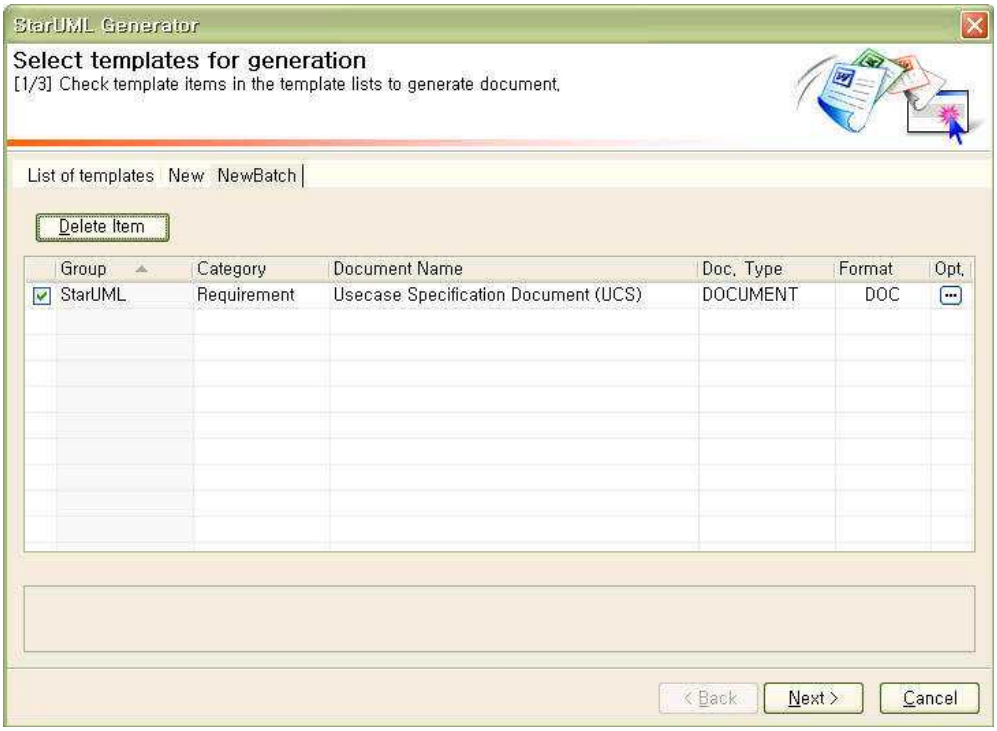
① [Select templates for generation] 페이지의 [List of templates] 탭의 템플릿 목록에서 새로운 배치 작업에 추가할 템플릿의 체크박스를 선택하고, [Register to Batch]를 클릭한다.





② [Register Batch] 다이얼로그가 나타나면 [Batch Name], [Description]을 입력하고 [OK]버튼을 클릭한다.

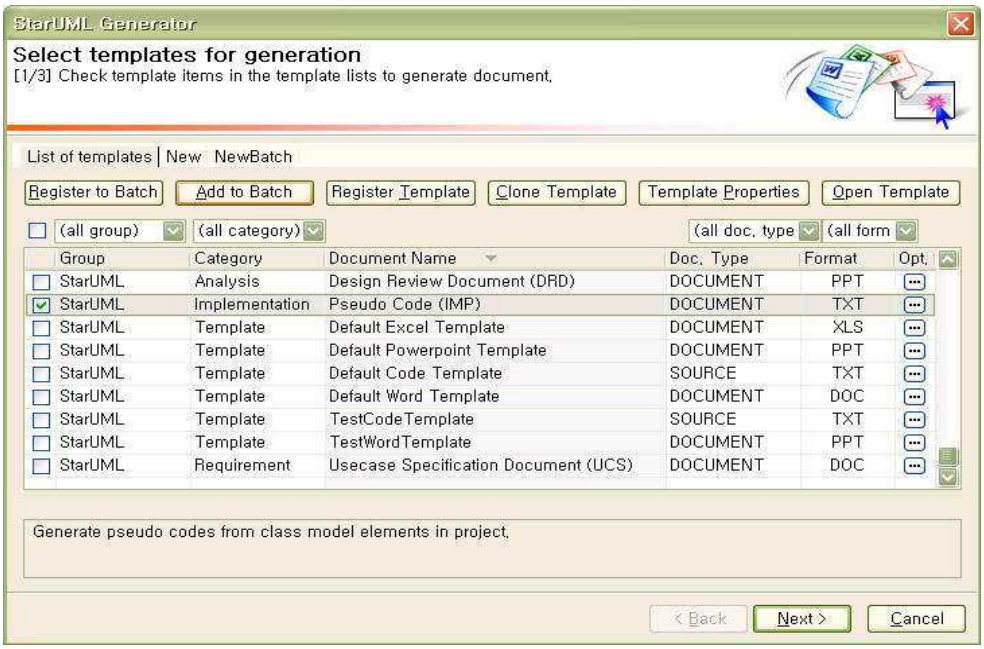
③ 새로운 배치 작업 이름으로 탭이 생성되고, 새로운 배치 작업 탭이 보이고 선택한 템플릿이 배치 작업에 포함되어진 것을 볼 수 있다.



(2) 템플릿을 기존 배치에 추가

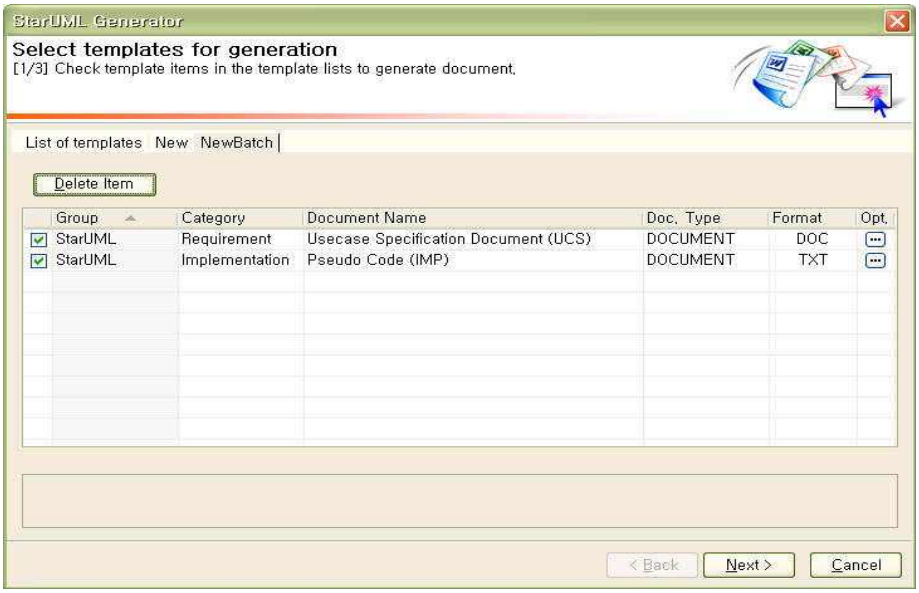
- 선택된 템플릿을 기존에 존재하는 배치 작업에 추가한다.

① [Select templates for generation] 페이지의 [List of templates] 탭의 템플릿 목록에서 추가할 템플릿의 체크박스를 선택하고, [Add to Batch]를 클릭한다.



② [Select Batch] 다이얼로그에서 선택한 템플릿이 포함될 배치 작업 이름을 선택하고 [OK]버튼을 클릭한다.

③ 배치 작업 탭이 보이고, 선택한 템플릿이 배치 작업에 포함되어진 것을 볼 수 있다.

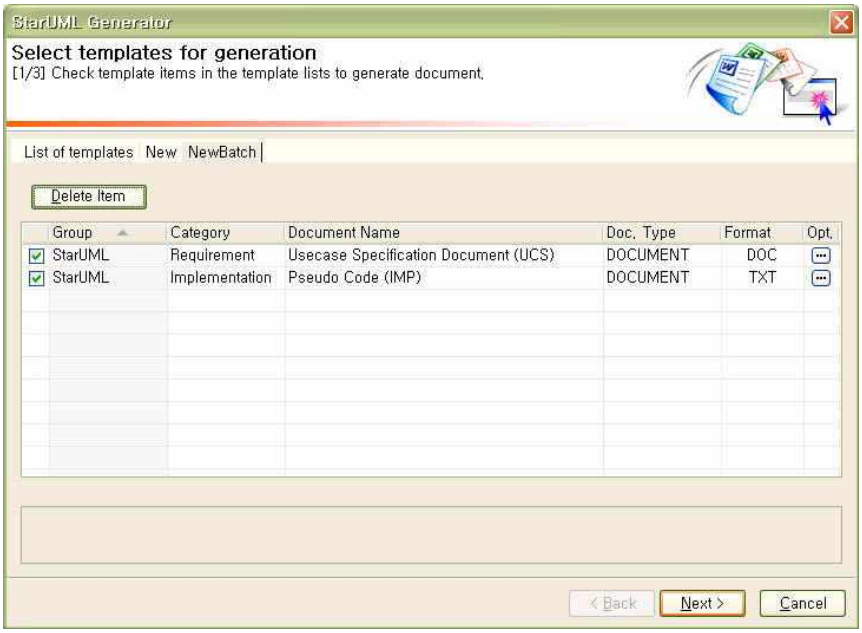


(3) 배치 작업 수행

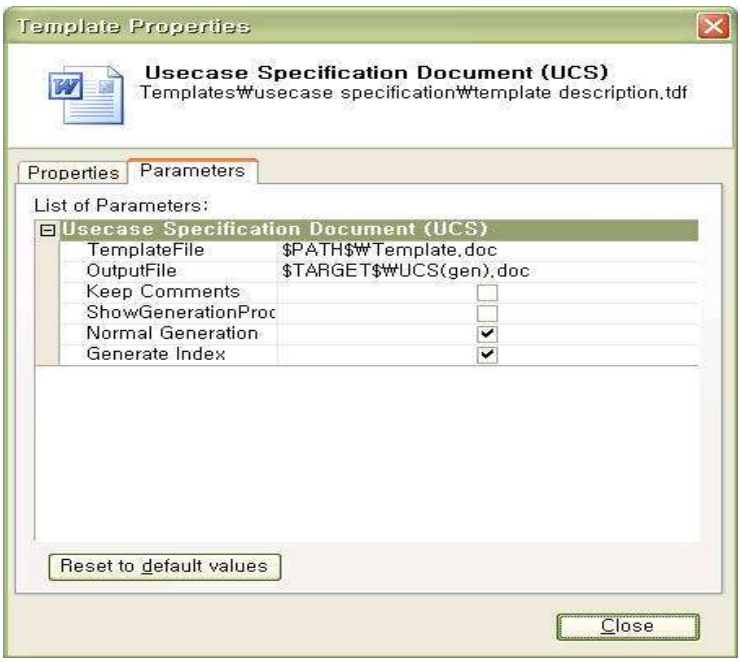
- 배치를 이용하여 여러 문서의 생성을 일괄적으로 처리한다.

① [Select templates for generation] 페이지에서 수행하려는 배치 작업 탭을 클릭한다.

② 생성할 템플릿의 체크박스를 선택하고 [Next] 버튼을 클릭한다. 기본적으로 배치 작업내의 모든 템플릿들이 선택되어 있다.



③ 템플릿 자체의 정보를 변경하지 않고 현재의 문서 생성에만 사용자가 원하는 인자가 적용되게 설정할 수 있다. 만약 생성 인자 설정을 변경하려면, 선택한 템플릿의 마지막 그리드 있는 [...] 버튼을 클릭한다. 그리고 템플릿 인자의 값을 원하는 설정으로 변경한다. (생성 인자에 대한 자세한 내용은 템플릿 등록 > 파라미터 정보를 참조한다)



④ [Select target path] 페이지가 화면에 나타나면 생성될 문서가 저장될 폴더를 선택하고 [Next]를 클릭한다. 만약 현재 선택된 폴더의 하위에 폴더를 추가하려면, 왼쪽 하단의 [New Folder...] 버튼을 클릭하고, 이름 설정 다이얼로그에서 추가될 폴더의 이름을 입력한다.

⑤ [Generating...] 페이지가 화면에 나타나면 [Generate] 버튼을 클릭한다. 템플릿으로부터

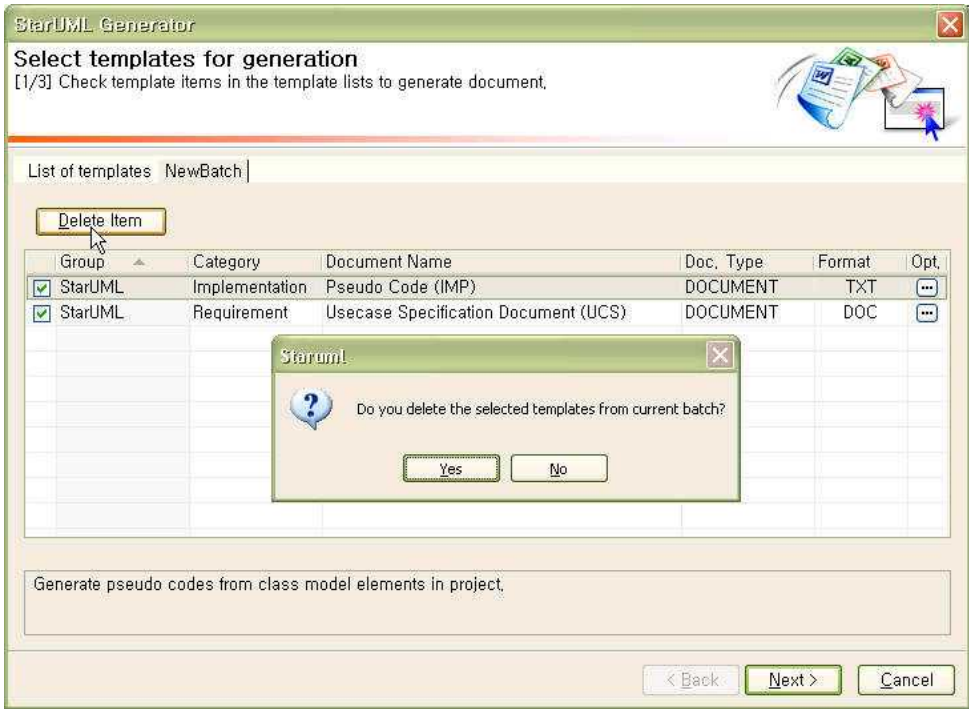
문서가 생성되면서, 각 템플릿의 생성 진행 정도를 진행 바를 통해서 확인할 수 있다. 그리고 생성 과정의 로그는 [Logs:] 창에 기록된다. 만약 현재 생성중인 문서를 취소하려면 [Cancel] 버튼을 클릭한다. 그리고 취소 확인 다이얼로그에서 확인 버튼을 클릭한다.

⑥ 문서 생성이 완료되면 로그 창에 생성 완료에 대한 로그(Document Creation is done)가 기록되고 [Finish] 버튼이 활성화된다. 문서 생성을 완료하려면 [Finish] 버튼을 클릭하여 문서 생성 과정을 종료한다. 또는[Generation List]에서 문서 목록을 더블 클릭하여 생성되어진 문서를 바로 열어 확인할 수 있다.

(4) 배치 작업 목록에서 템플릿 제거

- 배치 작업 목록에 추가한 템플릿을 배치 작업 목록에서 다시 제거할 수 있다.

① [Select templates for generation] 페이지의 해당 배치 탭에서 제거할 템플릿의 체크박스를 선택하고, [Delete Item]을 클릭한다.

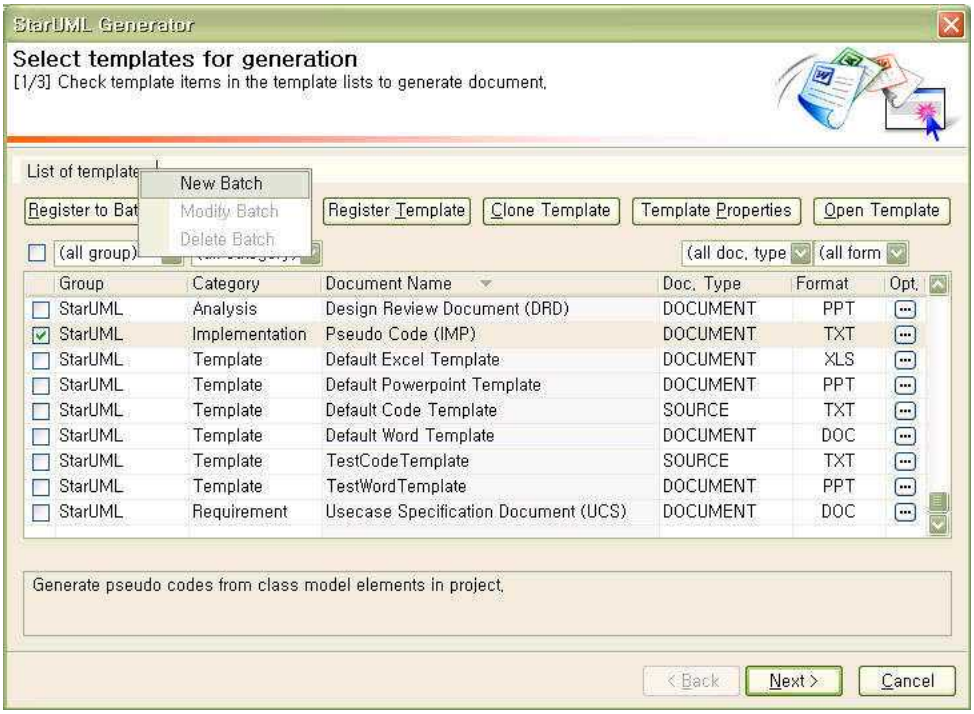


② 선택한 템플릿이 배치 작업 목록에서 제거되어진 것을 볼 수 있다.

(5) 배치 만들기

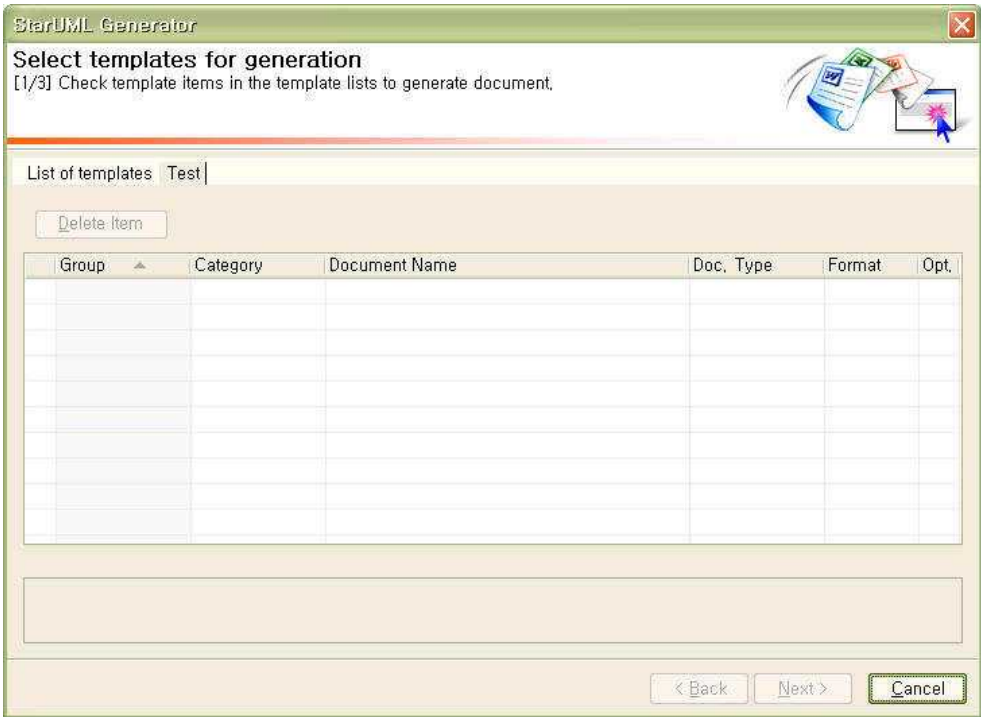
- 배치 작업을 새로 만들 수 있다.

① [Select templates for generation] 페이지의 탭에서 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [New Batch] 메뉴 아이템을 클릭한다.



② [Register Batch] 다이얼로그가 나타나면 [Batch Name], [Description]을 입력하고 [OK]버튼을 클릭한다.

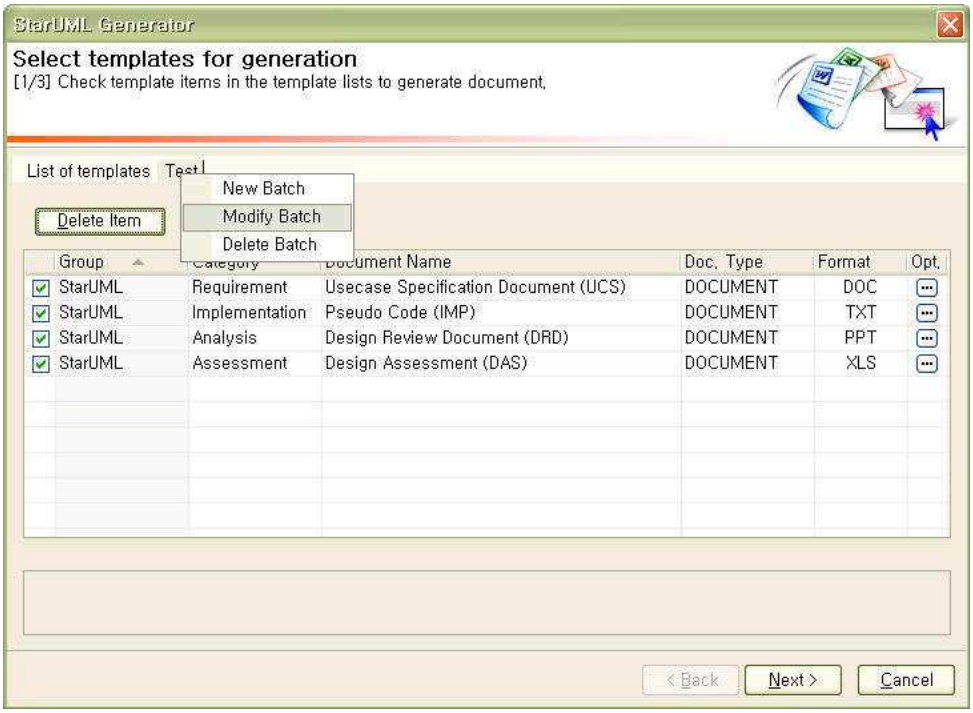
③ [Select templates for generation] 페이지의 탭에 새로운 배치 작업 탭이 생성되어진다.



(6) 배치 수정

- 기존 배치 작업의 이름과 설명 정보를 수정할 수 있다.

① [Select templates for generation] 페이지에서 수정할 배치 탭을 선택하고, 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Modify Batch] 메뉴 아이템을 클릭한다.

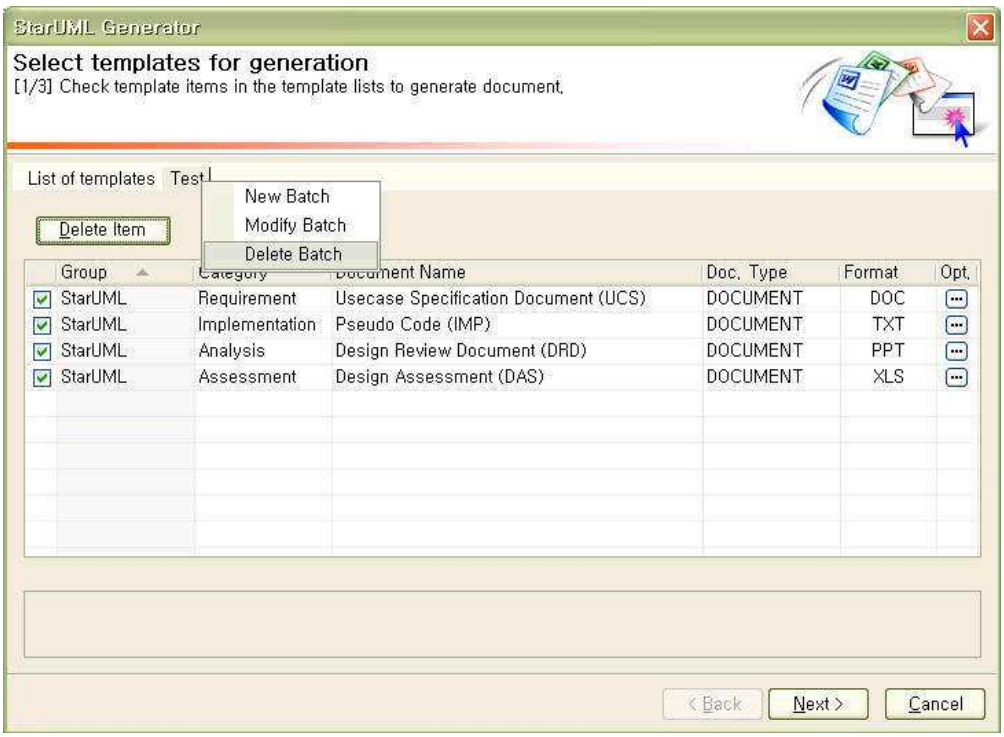


② [Register Batch] 다이얼로그가 나타나면 [Batch Name], [Description]을 수정하고 변경사항을 반영하기 위해서 [OK]버튼을 클릭한다.

(7) 배치 삭제

- 선택된 배치 작업을 삭제한다.

① [Select templates for generation] 페이지에서 삭제할 배치 탭을 선택하고, 오른쪽 마우스 버튼을 클릭하고 팝업 메뉴에서 [Delete Batch] 메뉴 아이템을 클릭한다.



② 선택한 배치 작업 탭이 삭제되어진 것을 볼 수 있다. (배치는 기존 템플릿에 대한 참조 정보들로만 구성되므로, 배치 작업이 삭제되어도 포함된 템플릿은 삭제되지 않다)

라. 템플릿 설치와 제거

(1) 템플릿 구성 요소

템플릿은 staruml-generator 하단에 설치되어진다. staruml-generator 하단에 templates 폴더에 StarUML Generator에서 사용하는 모든 템플릿들이 존재한다. 또한 배치 작업 목록들도 존재한다. 일반적으로 한 개의 템플릿은 한 개의 폴더 안에 관련된 모든 리소스 파일을 포함하고 있으며, 모든 템플릿 폴더들은 templates 폴더의 바로 아래에 위치한다. 하나의 템플릿은 두 가지 종류의 파일로 구성된다. 첫 번째는 템플릿 정의 기술 파일(.tdf)과 템플릿 문서(.doc, .ppt, .xls, .cot)이다. 템플릿 정의 기술 파일에서는 템플릿 등록하기 > 기본/상세/파라미터 정보에서 설정한 내용들이 저장되어 있으며, 템플릿 문서는 모델을 가져와서 보여주기 위한 커맨드와 스타일을 포함하고 있다. 배치 목록은 staruml-generator 폴더 아래의 batches 폴더에 배치 작업 파일(.btf) 형태로 존재한다.

- staruml-generator 모듈의 폴더 구성은 다음과 같다.

staruml-generator₩

templates₩

template1₩

template1.tdf

template1.doc

template2₩

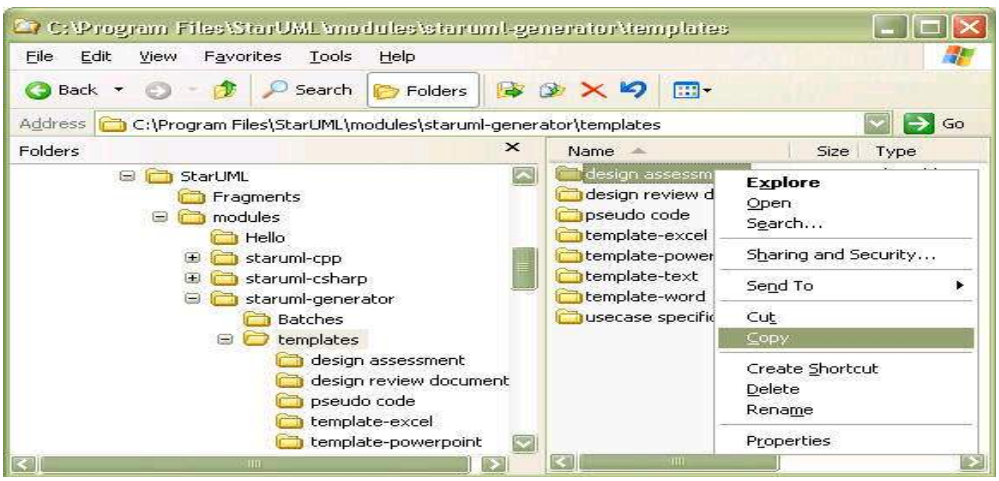
...

batches₩

batch1.btf

(2) 템플릿 설치와 제거

템플릿을 설치하는 방법은 매우 간단한다. 만약 다른 사용자에게 자신이 작성한 템플릿 배포하여 설치하고 싶다면 staruml-generator₩templates 폴더 아래에서 배포할 템플릿을 포함하는 폴더를 복사하고, 설치될 컴퓨터의 staruml-generator₩templates 폴더 아래의 경로에 붙여 넣으면 설치가 종료된다.



템플릿을 제거하는 방법 또한 매우 간단한다. staruml-generator₩templates 폴더 아래에서 삭제하고자 하는 해당 템플릿 폴더를 삭제하면 설치가 제거된다.

(3) 배치 작업의 설치와 제거

배치 작업을 설치하고 제거하는 방법도 매우 간단한다. 우선 배치 작업 목록에서 사용되는 템플릿 폴더를 템플릿 설치와 제거 순서에 따라서 설치하고, staruml-generator₩batches 폴더 아래에서 원하는 배치작업(.btf) 파일을 복사하여, 설치될 컴퓨터의 staruml-generator₩batches 폴더 아래로 복사하면 배치의 설치가 완료 된다.

배치 작업의 설치 제거는 staruml-generator₩batches 폴더 아래의 .btf 파일을 삭제하는 것으로써 완료된다.

6. 모델 검사하기

이 장에서는 소프트웨어 모델의 검사에 관해서 설명한다. 검사 방법 및 검사에 사용되는 규칙들을 설명한다.

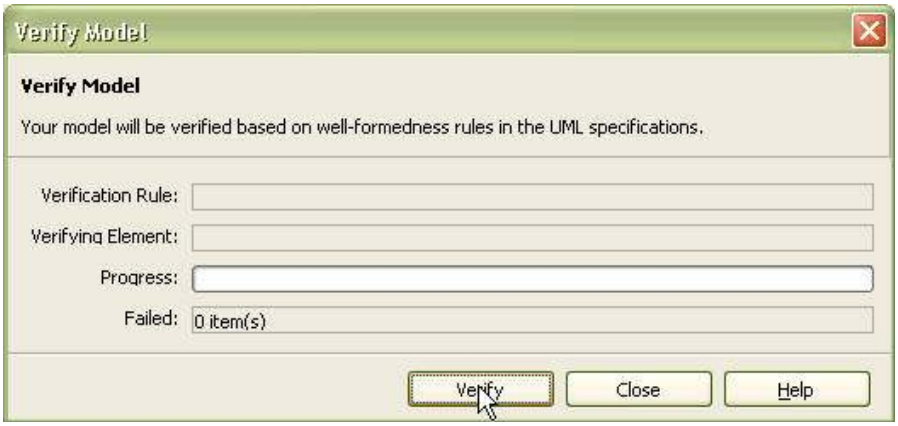
가. 모델 검사하기

UML로 소프트웨어를 모델링하다 보면 수많은 실수를 하게 된다. 그러나 그런 사소한 실수들이 시간이 지난 후 치명적인 오류를 유발할 수도 있으므로 사전에 그런 가능성을 미리 찾아내는 것이 중요한다. StarUML™에서는 UML의 기본적인 규칙들을 적용하여 소프트웨어 모델을 검사할 수 있다.

(1) 모델을 검사하는 방법:

① [Model]->[Verify Model] 메뉴를 선택한다.

② 모델 검사 대화상자가 나타나면 [Verify] 버튼을 누른다.



③ 검사가 끝나면 정보 영역의 [Message] 부분에 검사를 통과하지 못한 요소들과 오류 내용이 나타나고, 그것을 더블 클릭하면 해당 요소를 찾아간다.

나. 기본적인 검사규칙들

모델을 검사하기 위한 38가지의 규칙들이 정의되어 있다. 이 정의들은 주로 UML 명세의 Well-formedness Rule들에 따라 정의되었다.

- 모델 검사 규칙 목록

번호	규칙내용	Elements Applied
1	연관(Association)내의 연관끝(AssociationEnd)들은 서로 유일한 이름을 가져야 한다.	Association
2	집합(Aggregation) 혹은 합성(Composition)인 연관끝(AssociationEnd)은 하나의 연관(Association) 내에서 두 개 이상 존재할 수 없다.	Association
3	파라미터(Parameter)는 서로 유일한 이름을 가져야 한다.	BehavioralFeature
4	분류자(Classifier) 내에서는 동일한 이름의 속성(Attribute)이 존재할 수 없다.	Classifier
5	반대측 연관끝(AssociationEnd)들의 이름들은 서로 유일해야 한다.	Classifier
6	속성(Attribute)의 이름은 반대측의 연관끝(AssociationEnd) 혹은 분류자(Classifier)에 포함된 요소의 이름과 동일할 수 없다.	Classifier
7	반대측 연관끝(AssociationEnd)의 이름은 분류자(Classifier)에 포함된 요소나 그것의 속성(Attribute)의 이름과 동일할 수 없다.	Classifier
8	루트(Root) 요소는 더 일반화된 요소를 가질 수 없다.	GeneralizableElement
9	말단(Leaf) 요소는 더 특수화된 요소를 가질 수 없다.	GeneralizableElement
10	순환적인 상속(Inheritance) 구조는 허용되지 않다.	GeneralizableElement
11	인터페이스(Interface)의 모든 특징(Feature)들은 공개(Public) 이어야 한다.	Interface
12	컴포넌트 인스턴스(ComponentInstance)는 자신의 원시(origin)로써 정확히 하나의 컴포넌트(Component)를 지정해야 한다.	ComponentInstance
13	노드 인스턴스(NodeInstance)는 자신의 원시(origin)로써 정확히 하나의 노드(Node)를 지정해야 한다.	NodeInstance
14	연관끝-역할(AssociationEndRole)은 역할(ClassifierRole)로 연결되어야 한다.	AssociationEndRole
15	역할(ClassifierRole)은 자신만의 특징(Feature)을 가질 수 없다.	ClassifierRole
16	역할(ClassifierRole)은 다른 어떤 역할(ClassifierRole)의 역할이 될 수 없다.	ClassifierRole
17	메시지(Message)의 송신자(sender)와 수신자(receiver)는 반드시 해	Message

	당 인터랙션(Interaction)의 문맥인 협동(Collaboration)에 참여하는 것이어야 한다.	
18	액터(Actor)는 유스케이스(UseCase), 클래스(Class) 혹은 서브시스템(Subsystem)으로 연결되는 연관(Association)만을 가질 수 있다.	Actor
19	복합 상태(CompositeState)는 최대 1개의 초기 상태(Initial state)만을 가질 수 있다.	Composite State
20	복합 상태(CompositeState)는 최대 1개의 깊은 이력(Deep history)만을 가질 수 있다.	Composite State
21	복합 상태(CompositeState)는 최대 1개의 얕은 이력(Shallow history) 만을 가질 수 있다.	Composite State
22	동시성 복합 상태(concurrent composite state)는 최소한 2개 이상의 복합 상태를 포함해야 한다.	Composite State
23	동시성(concurrent) 상태는 하위상태(sub state)로써 복합 상태(composite state)만을 가질 수 있다.	Composite State
24	최종상태(Final state)는 나가는 전이(transition)를 가질 수 없다.	FinalState
25	초기상태(Initial state)는 나가는 전이(transition)를 최대 1개를 가질 수 있고, 들어오는 전이(transition)는 가질 수 없다.	Pseudostate
26	이력(History) 상태들은 최대 1개의 나가는 전이(transition)를 가질 수 있다	Pseudostate
27	점합점(junction vertex)은 최소한 들어오는 전이(transition) 1개, 그리고 나가는 전이(transition) 1개는 가져야 한다.	Pseudostate
28	선택점(choice vertex)은 최소한 들어오는 전이(transition) 1개, 그리고 나가는 전이(transition) 1개는 가져야 한다.	Pseudostate
29	상태머신(StateMachine)은 분류자(Classifier) 혹은 행위적 특징(BehavioralFeature) 둘 중 하나에 결합될 수 있다.	StateMachine
30	최상위 상태(top state)는 항상 복합 상태(composite)이어야 한다.	StateMachine
31	최상위 상태(top state)를 포함하는 상태는 존재할 수 없다.	StateMachine
32	최상위 상태(top state)로부터 나가는 전이(transition)는 존재할 수 없다.	StateMachine
33	하위-머신상태(SubmachineState)는 동시성을 가질 수 없다.	SubmachineState
34	의사상태(pseudostate)로 향하는 전이(transition)는 트리거(Trigger)를 가질 수 없다.	Transition
35	활동그래프(ActivityGraph)는 패키지(Package), 분류자(Classifier) 혹은 행위적 특징(BehavioralFeature) 중 하나의 동적행위를 묘사할 수 있다.	ActivityGraph
36	활동상태(ActionState)는 내부전이(internal transition), 퇴거-행위(exit action), 주-행위(do activity)를 가질 수 없다.	ActionState
37	활동상태(ActionState)으로부터 나오는 전이(transition)는 트리거 이벤트(trigger event)를 가질 수 없다.	ActionState
38	하위-활동상태(SubactivityState)는 활동그래프(ActivityGraph)로의 연결을 가져야 한다.	Subactivity State

7. 인쇄하기

이 장에서는 다이어그램의 인쇄에 관한 내용들을 설명한다. 페이지 설정, 다이어그램을 다양한 방법으로 인쇄하는 방법 및 인쇄결과 미리 보기에 대해서 설명한다.

가. 페이지 설정하기

인쇄할 페이지에 관한 다양한 속성(다이어그램의 정보 출력, 용지의 여백 그리고 테두리 표시 등)들을 설정할 수 있다.

(1) 다이어그램의 정보를 표시하는 방법:

① [File]->[Page Setup] 메뉴를 선택하면 페이지 설정 대화상자가 나타난다.

② 먼저, 다이어그램의 정보를 어느 위치에 표시할 것인지를 결정한다. 나타내지 않으려면 [Option] 그룹에서 [None]을 선택하고 위쪽에 나타내려면 [On Header]를, 아랫부분에 나타내려면 [On Footer]를 선택한다.

③ 그런 다음, 어떤 정보들을 표시할 지를 선택한다. 표시할 수 있는 정보에는 [Diagram Name], [ProjectTitle], [PageNumber], [DiagramKind], [Date/Time]이 있다.

(2) 용지의 방향을 설정하는 방법:

① [File]->[Page Setup] 메뉴를 선택하면 페이지 설정 대화상자가 나타난다.

② 용지를 세로로 출력하려면 [Page Orientation] 그룹에서 [Portrait]를 선택하고,

③ 용지를 가로 방향으로 출력하려면 [Landscape]를 선택한다.

(3) 용지의 여백을 조절하는 방법:

① [File]->[Page Setup] 메뉴를 선택하면 페이지 설정 대화상자가 나타난다.

② 용지의 여백을 [Margin] 그룹에서 [Top], [Bottom], [Left], [Right] 부분에 밀리미터 단위로 정수를 입력한다.

(4) 용지에 테두리를 나타나게 하는 방법:

① [File]->[Page Setup] 메뉴를 선택하면 페이지 설정 대화상자가 나타난다.

② 테두리가 나타나게 하고 싶은 위치를 [Border] 그룹에서 [Top], [Bottom], [Left], [Right] 부분을 체크한다.

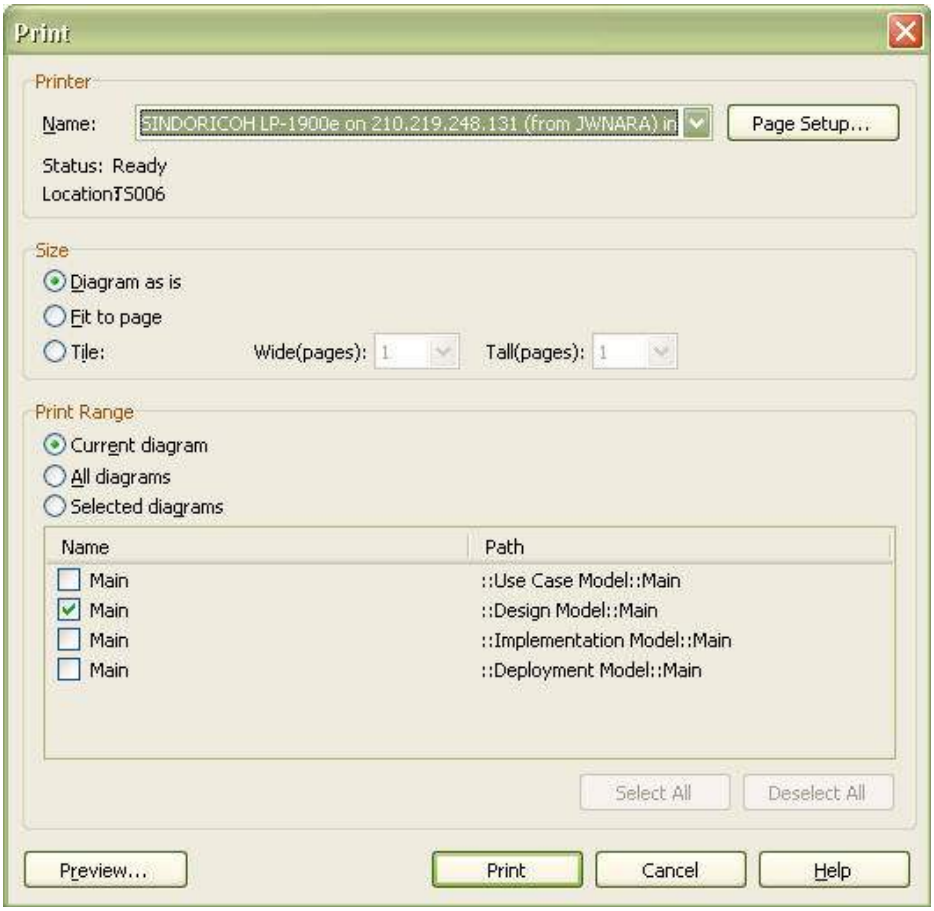
③ 그리고 테두리의 두께를 [Thickness] 부분에 입력한다.

나. 다이어그램 인쇄하기

다이어그램을 다양한 방법으로 인쇄할 수 있는 기능들을 제공한다. 출력하고자 하는 다이어그램들 선택, 다이어그램의 출력 크기, 다이어그램을 나누어 인쇄하기 등의 기능을 설명한다.

(1) 현재 다이어그램을 인쇄하는 방법:

① [File]->[Print] 메뉴를 선택하면 인쇄 대화상자가 나타난다.



② [Printer] 그룹의 [Name] 부분에서 출력할 프린터를 선택한다.

③ [Print Range] 그룹에서 [Current diagram]을 선택하고 [Print] 버튼을 누른다.

(2) 원하는 다이어그램들만 인쇄하는 방법:

① [File]->[Print] 메뉴를 선택하면 인쇄 대화상자가 나타난다.

② [Printer] 그룹의 [Name] 부분에서 출력할 프린터를 선택한다.[PrintRange] 그룹에서 [Selected diagrams]를 선택하고, 인쇄하기를 원하는 다이어그램들을 그 아래의 목록에서 체크 표시를 한다.

③ [Print] 버튼을 누른다.

(3) 다이어그램을 용지 크기에 맞추어 인쇄하는 방법:

① 인쇄 대화상자에서 인쇄할 다이어그램(들)을 선택한다.

② [Size] 그룹에서 [Fit to page]를 선택하고 [Print] 버튼을 누른다.

(4) 다이어그램을 여러 장에 나누어 인쇄하는 방법:

① 인쇄 대화상자에서 인쇄할 다이어그램(들)을 선택한다. [Size] 그룹에서 [Tile]을 선택하고, 몇 장에 나누어서 인쇄할 것인지를 [Wide(pages)] 부분과 [Tall(pages)] 부분에 정수로 입력한다. (예: 가로 3, 세로 2를 선택했으면 가로 3장, 세로 2장의 크기 이므로 총 3*2=6장에 걸쳐 출력된다.)

② [Print] 버튼을 누른다.

다. 인쇄결과 미리보기

다이어그램을 용지에 인쇄하기 전에 그 결과를 미리 예측할 수 있다.

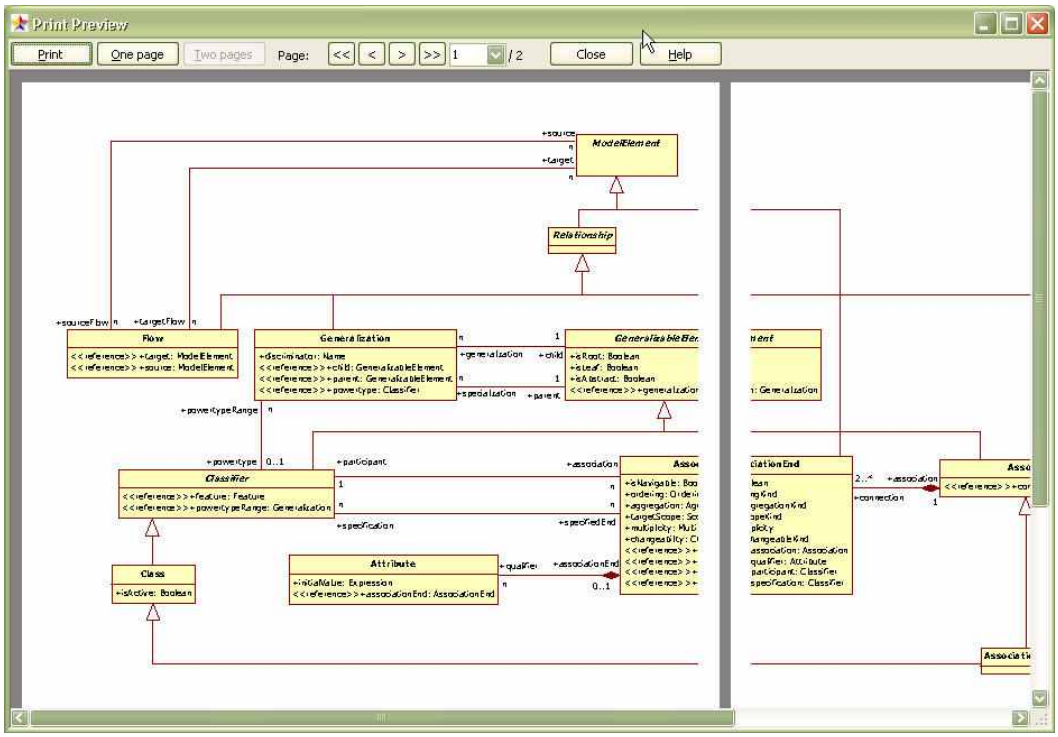
(1) 인쇄 결과를 미리 보는 방법:

① [File]->[Print] 메뉴를 선택해서 인쇄 대화상자를 나타나게 한 다음, 인쇄할 다이어그램에 관한 정보를 입력한다. (다이어그램 인쇄하기 참고)

② 인쇄 대화상자 아래쪽에 있는 [Preview] 버튼을 누른다.

③ 인쇄 미리보기 대화상자가 나타나면 1장씩 볼 것인지 2장씩 볼 것인지, 그리고 보고

싶은 페이지 번호를 조절해가면서 인쇄 결과를 미리 볼 수 있다.



④ 여기서 바로 인쇄하려면 [Print] 버튼을 그만 보려면 [Close] 버튼을 누른다.