Team Report

Introduction to OOAD using UML tools

과 목 명 : 소프트웨어공학개론

담 당 : 유 준 범 교수님

제 출 일 : 2010. 10. 27

8조

200310548 이정우

200911364 곽수미

200911372 김민하

200911373 김바울

목 차

1. OOAD (orient analysis and design)

1-1. OOAD란?

OOAD란 소프트웨어를 개발하는 하나의 방법론으로 모든 소프트웨어 시스템의 주요 기본 요소를 사물을 가리키는 객체와 그 객체들을 하나의 집합으로 묶은 클래스로 구성하는 객체 지향적인 분석과 설계 방법을 말합니다. 객체 지향적이란 것은 현실계에 실재하는 사물, 즉 객체들을 지향한다는 것 입니다. 예를 들어 쇼핑몰을 구축한다고 가정할 때 여기서의 객체는 "고객, 상품, 결재계좌 등"이라 할 수 있고 객체 지향적이라는 것은 바로 이러한 실재 생활들을 보다 편리하고 빠르게 이용할 수 있게 하는 프로그램을 현실 생활에 기초해서 설계하는 개념입니다. 객체 지향이란 개념은 이러한 실용성 때문에 최근에 급속히 확장해 가고 있습니다.

1-2. Object Attributes 와 Operation

Attributes는 Property 즉 "속성" 이라는 뜻이며, 객체나 데이터가 갖고 있는 특성을 추출해 사용합니다. 주로 명사형으로 표현되어지고 예를 들자면 “자동차라는 클래스에서 엔진, 휠, 핸들 등과 같은 객체”를 말합니다. Operation은 객체에 행동(Action)을 부여하여 객체로 하여금 어떠한 행동을 취하게 하는 역할을 합니다. 주로 Class File 에서는 "Method"라고 말합니다. 예를 들자면 자동차를 “운전하다, 세차하다, 팔다, 새로 사다”라는 동사형으로 표현되어집니다. Abstract는 추상화라고도 하며 "명사와 동사를 추출하는 것"을 말합니다. 실제 객체를 프로그램으로 응용해 내는 것 입니다. 이과정이 가장 중요한 과정인데, 추상화를 어떻게 하느냐에 따라 클래스 설계가 결정됩니다. 이것이 모델링의 핵심 과정이고 성패를 결정짓는 주요 요소입니다.

1-3. 모델과 모델링

모델이란 하나의 시스템을 개발하는데 있어 현실을 추상화 시켜 놓은 청사진을 말합니다. 그리고 모델링이란 정보시스템 설계 모델을 실제 현실 세계에 맞게 구현하는 작업을 말합니다. 이러한 모델이 필요한 이유는 미리 모델을 만들어 보면서 앞으로 개발하려는 시스템을 좀 더 구체적이고도 명확하게 인식할 수 있기 때문입니다. 따라서 모델링은 시스템을 현재 또는 원하는 모습으로 가시화하도록 도움을 주기 때문에 중요합니다.

1-4. 객체 지향 설계

1) 객체 모델링 단계

클래스를 만들기 위한 작업을 의미합니다. 추상화 단계라고도 합니다. 컴퓨터를 예를 들어 보면 컴퓨터란 사물(객체)를 처리하기 위해 각종 컴퓨터가 가지고 있는 특성을 분리하고 분

리된 특성의 공통점을 뽑아내는데 이때, CPU, 메모리, HDD, 모니터 등등이 아마도 컴퓨터 특성의 공통점이 될 것입니다.

2) 클래스 정의 단계

객체 모델링 단계에서 산출된 공통점을 가지고 실제로 컴퓨터가 처리할 수 있는 구조로 바꾸는 단계입니다. 메모리에 용량, 제조사, 속도 등의 욧가 있는데, 이러한 요소들을 어떤 방법을 사용하여 어떤 처리를 하고 어떤 결과를 산출할지를 설계하고 만든는 단계가 클래스 정의 단계입니다.

3) 객체 생성과 사용

정의된 클래스를 메모리에 적재해서 사용하는 단계입니다. 클래스를 정의했다고 메모리에 적재되는 것은 아닙니다. 클래스를 사용하기 위해서는 클래스를 인스턴스(Instance)화 하여야 하는데 이 인스턴스가 객체라고 할 수 있습니다.

1-5. 객체 지향 설계 원칙

1) SRP - 단일 책임 원칙(Single Responsibility Principle) : 클래스는 단 한가지의 변경이유(책임)만을 가져야 합니다.

2) OCP - 개방-폐쇄 원칙(Open-Closed Principle) : 소프트웨어 개체(클래스, 모듈, 함수 등등)는 확장에 대해 열려 있어야 하고, 수정에 대해서는 닫혀 있어야 합니다.

3) LSP - Liskov 교체 원칙(Liskov Substitution Principle) : 하위타입(Sub type)은 그것의 기반 타입(Base type)에 대해 치환 가능해야 합니다.

4) DIP - 의존 관계 역전 원칙(Dependency Inversion Principle) :

- 상위 수준의 모듈은 하위 수준의 모듈에 의존해서는 안 됩니다. 둘 모두 추상화에 의존해야 한다.

- 추상화는 구체적인 사항에 의존해서는 안 됩니다. 구체적인 사항은 추상화에 의존해야 합니다.

5) ISP - 인터페이스 격리 원칙(Interface Segregation Principle) : 클라이언트가 자신이 사용하지 않는 메소드에 의존하도록 강제 되어서는 안 됩니다.

2. UML

2-1. UML(Unified Modeling Language)이란?

UML이란 소프트웨어 개발 과정에서 산출되는 산출물들을 명시, 개발, 문서화하기 위한

모델링 언어입니다.

1) 가시화 언어 : 개념 모델 작성, 오류 없이 전달, 의사 소통의 용이, Graphic 언어

2) 명세화 언어 : 정확한 모델 제시, 완전한 모델 작성, 분석,설계의 결정 표현

3) 구축 언어 : 다양한 Prog. 언어와 연결, 왕복 공학 가능, (순 공학/역공 학)실행 시스템 예측 가능

4) 문서화 언어 : 시스템에 대한 통제, 평가, 의사소통의 문서(요구사항, Architecture, 설계, Source Code, Project 계획,Test, Prototype, Release)

2-2. UML을 이용한 모델링의 특성

사용사례 분석기법을 사용하여 사용자 관점에서의 업무과정을 포착할 수 있고 업무객체와 업무논리를 포착할 수 있도록 하는 커뮤니케이션 도구로써 업무객체와 업무논리로부터 응용시스템을 분석하고 설계할 수 있도록 합니다. 또 시스템의 복잡성을 관리할 수 있도록 하여 사람이 보통 한 번에 효과적으로 다룰 수 있는 한계를 넘을 수 있고 소프트웨어의 아키텍처를 정의할 수 있도록 하여 프로그래밍 언어, 시스템구현 환경과 독립적으로 시스템을 모델링 할 수 있습니다. (3계층 소프트웨어 아키텍처에 따라 사용자 인터페이스는 비쥬얼 베이직 혹은 자바, 업무논리는 C++ 혹은 자바, 데이터베이스 서버는 C++과 SQL로 구현) 재사용 가능한 시스템구성 컴포넌트를 모델링함으로써 재사용을 촉진시켜 이를 여러 시스템에서 사용할 수 있습니다.

2-3. UML의 구성요소

1) UML의 사물(thing)

UML의 사물(thing)에는 구조사물(structural things), 행동사물(behavioral things), 그룹사물(groupings things), 주해사물(annotational things) 이 4가지 종류가 있습니다.

(1) 구조사물(structural things) : 보통 명사형으로 표현되어 모델의 정적인 부분을 표현합니다. 구조사물의 용어로는 클래스(동일한 속성, 오퍼레이션, 관계, 그리고 의미를 공유하는 객체들을 기술), 인터페이스(클래스 또는 컴포넌트의 서비스를 명세화 하는 오퍼레이션들의 집합), 쓰임새(시스템이 수행하는 순차적 활동들을 기술하며, 행위자(actor)와 반응), 컴포넌트, 노드 등이 있습니다.

(2) 행동사물(behavioral things) : UML모델의 동적인 부분을 표현하며 시간과 공간에 따른 행동을 표현합니다. 용어로는 교류(interaction), 상태머신(상태의 순서를 지정하는 행동)이 있습니다.

(3) 그룹사물(groupings things) : UML모델을 조직하는 부분입니다.

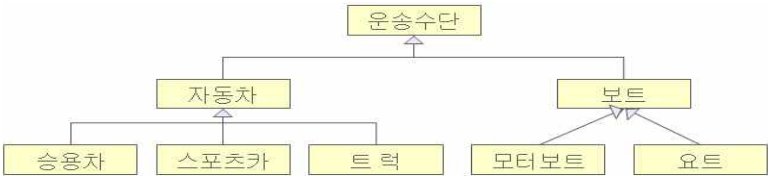
(4) 주해사물(annotational things)은 UML모델을 설명하는 부분으로 모델의 어떠한 요소들에 대하여 명확하게 설명하고 정하는 것 입니다.

2) UML의 관계

UML의 관계를 알기위해 먼저 클래스에 대해 보면 혼자서 존재하는 클래스는 거의 없으며, 다양한 방식으로 다른 사물과 교류하는 것을 알 수 있습니다. 클래스를 먼저 파악하고 나면, 클래스들 간의 관계에 주목해야 하는데 의존, 연관, 일반화, 실체화의 4가지 관계가 있습니다.

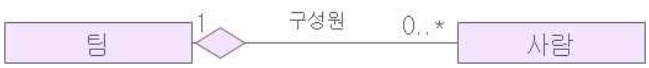
첫 째로, 의존(dependency)관계란 한쪽 사물의 변화가 다른 사물에 영향을 주는 관계로서 한 쪽의 변화가 다른 쪽에 영향을 미치는 경우입니다.

두 번째, 일반화(generalization)관계는 특수화(specialization)/일반화(generalization)관계라고도 하며 보다 일반적인 클래스와 이 클래스와 완전한 일관성을 가지며 보다 상세하고 추가적인 정보를 제공하는 클래스간의 분류학적 관계입니다.



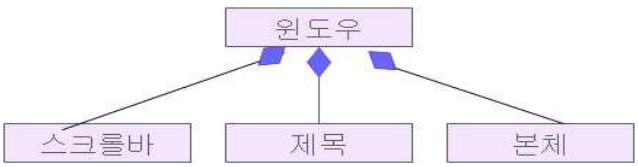
세 번째, 연관(association)관계는 객체들 간의 링크(연결)가 있음 을 표현하는 것으로 어느 한 사물 객체가 다른 사물 객체와 연결되어 있음을 나타냅니다. 연관관계에는 두 가지가 있는데 먼저 집합연관(aggregation)관계는 전체/부분 관계이며 전체와 부분은 서로 독립적으로 창조되고 파괴됩니다. 복합연관(composition)관계도 전체/부분 관계이며 부분의 창조와 파괴는 전체의 결과로서 일어남(동일한 수명을 가짐)을 뜻합니다.

(1) 집합 연관 (aggregation) 관계



(2) 복합 연관(composition) 관계

예 : 어느 특정 윈도우가 두 개의 스크롤바, 제목, 그리고 본체로 구성되어 있다면 윈도우가 없어지면 이들 구성요소들도 없어지게 됩니다.



마지막으로 실체화(realization)관계는 한쪽 분류자는 다른 쪽 분류자가 수행하기로 되어 있는 계약을 명세화하며, 일반화와 의존관계의 절충을 뜻합니다.

3) UML의 도해

(1) Use Case 다이어그램

먼저 Use Case는 사용자의 시스템에 대한 요건을 이해하기 위한 것으로, 가장 근본적인 요소입니다. Use case는 이후의 모든 시스템분석, 설계, 개발, 시험에 걸쳐 영향을 미치게 되며, 개발 진행기간 동안 시스템의 행위에 대한 테스트의 기본이 됩니다. Use Case는 시스템의 사용자 입장에서의 기능적인 요구사항과 시스템 내에서 혹은 외부 시스템과의 상호작용하는 행위들의 집합을 나타내야 하며, 이러한 모든 Use Case는 내부의 구현을 생각하지 않습니다. Actor는 Use Case와의 상호 작용 시에 Use Case를 사용하는 사용자의 역할들의 집합을 말하며, 사람뿐만 아니라 하드웨어 디바이스, 다른 외부 시스템 등이 될 수 있습니다. Use Case 다이어그램은 시스템의 Use Case View를 모델링하는 다이어그램입니다. Use Case View란 시스템의 구현을 배제하고 시스템 전체 혹은 시스템의 일부분과 외부와의 상호작용 및 시스템 자체의 행위(Behavior)등을 보는 관점을 나타냅니다. Use Case 다이어그램이 필요한 이유는, 다른 Use Case들 간 관계뿐만 아니라 사용자와 개발자, 시스템 설계자와의 의사소통 도구가 되고 개발자에게 시스템의 범위와 시스템의 기능적 요구 사항들을 시각화하여 쉬운 이해를 제공하기 때문입니다.

Use Case 다이어그램의 모델링 순서로는 모든 Actor들을 추출하여 Actor중 좀 더 일반적인 혹은 더 특화된 Actor들을 선별하여 구성합니다. 각 Actor에 관한 상호 작용에서 Use Case를 추출하여 이러한 상호 작용을 통한 상태의 변화를 Use Case로 추출합니다. 여기서 추출된 Use Case에서 좀 더 일반적인 혹은 특화된 Use Case로 구성합니다. Use Case는 다이어그램 중간에 타원을 그려서, 타원의 중앙 또는 아래에 Use Case의 이름을 적습니다. Actor를 그리려면 다이어그램의 왼쪽이나 오른쪽에 사람 모양을 그려 넣습니다.

다이어그램에 Use Case가 없다는 것은 시스템이 수행하지 않은 일을 보여주고 있는 것입니다. 따라서 다이어그램에 제공된 명확하고 간단한 Use Case 설명을 통해 시스템에 필요한 기능이 존재하는지의 여부를 쉽게 알 수 있습니다.

예 : 신용카드 인증 시스템

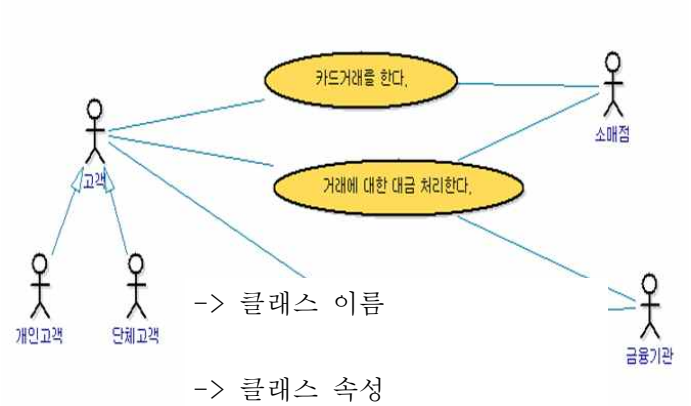
• Actor의 추출

- 신용카드를 사용하는 개인 고객, 단체 고객
- 신용카드로 거래를 하는 소매기관
- 고객의 신용 상태를 처리하는 기관

• Use Case의 추출

- 고객이 신용카드를 사용하여 카드 거래를 합니다.
- 거래에 대한 대금을 처리합니다.
- 고객개인의 계좌를 관리합니다.

이러한 Use Case를 가지고 UML을 표현하는 도해로는 클래스도(Class diagram), 객체 다이어그램(Object diagram), 쓰임새 다이어그램(Use case diagram), 순차 다이어그램(Sequence diagram), 협력 다이어그램(Collaboration diagram), 상태 다이어그램

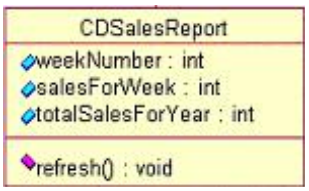


(Statechart diagram), am), 컴포넌트 다이어그램 (Component diagram), 배치 다이어그램(Deployment diagram)가 있다.

(2) 클래스 다이어그램(Class diagram)

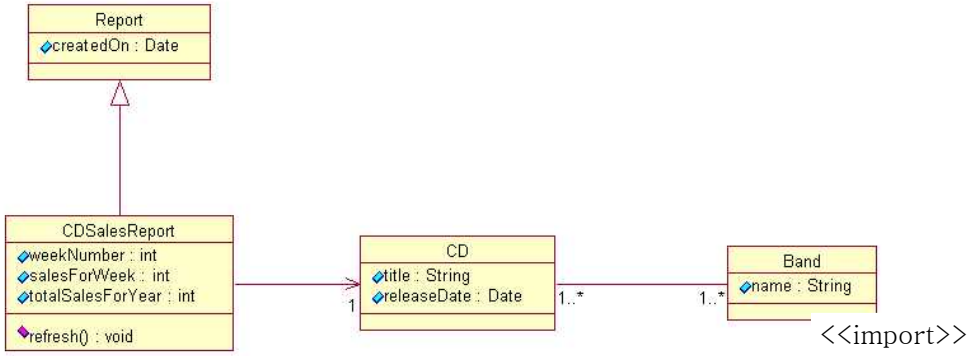
클래스 다이어그램은 다른 엔티티들(사람,제품,데이터)이 서로 어떻게 관계를 맺고 있는지를 보여줍니다. 개별적인 객체라기보다는 동일한 속성, 오퍼레이션, 관계, 그리고 의미를 공유하는 객체 집합으로 시스템의 어휘 모델링입니다. 다시 말해서, 이것은 시스템의 정적 구조라고 할 수 있습니다. 클래스에는 가시성(visibility), 다중성(multiplicity)이라는 두 가지 특성이 있는데, 먼저 가시성(visibility)이란 속성이나 오퍼레이션을 다른 분류자에 의해 사용될 수 있는지 여부를 지정하는 것으로서 특징으로서는 공용(public)특성(모든 외부 분류자들이 사용할 수 있음-"+"), 보호(protected)특성(이 분류자의 모든 자식들이 사용할 수 있음-"#"), 전용(private)특성(이 분류자 자신만이 사용할 수 있음-"-")이 있습니다. 여기서의 분류자란 구조적, 행동적 특성을 설명하기 위한 메커니즘으로 클래스, 컴포넌트, 노드, 쓰임새 등이 있습니다. 다중성(multiplicity)이란 하나의 클래스가 가질 수 있는 인스턴스 수와 특정 실체에 잠정적으로 허용 가능한 관계수의 범위를 뜻하는 특성이며, '1..*1..1 0..*' 등으로 표현합니다.

클래스 다이어그램의 클래스 객체는 세 개의 섹션으로 설명합니다. 위 섹션은 클래스의 이름, 중간 섹션은 클래스의 속성, 가장 아래 섹션은 클래스의 메소드를 나타냅니다.



클래스 다이어그램의 경우 상속 관계를 그릴 때에는 화살표 방향을 위로 향하게 하여 수퍼클래스를 지시하게 하면서 화살표 모양은 완벽한 삼각형이 되도록 해야 합니다. 상관관계는 두 클래스들이 서로를 인식하고 있다면 일직선이 되어야 하고, 클래스의 한 편만 알고 있는 관계라면 화살표 표시가 되어있는 선을 그어야 합니다.

(3) 객체 다이어그램(Object diagram)

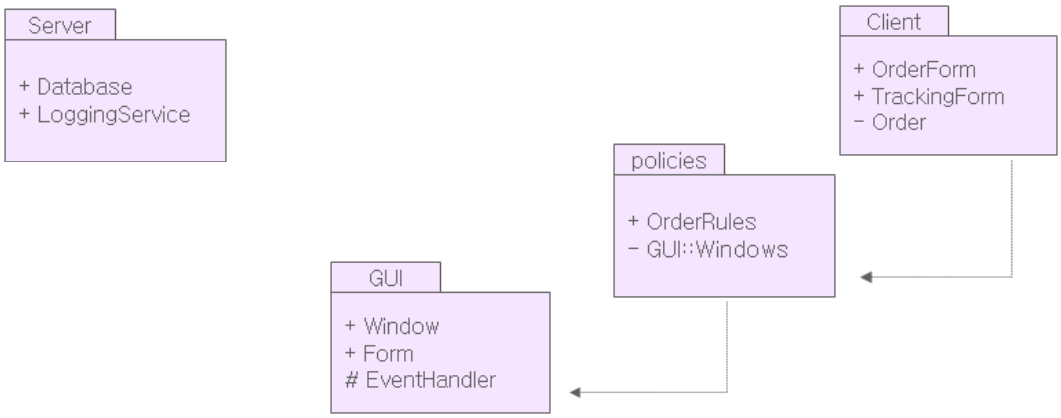


객체는 클래스로부터 생성되는 한 인스턴스라고도하며 클래스와 동일한 관계를 갖습니다. 다음 예와 같이 객체이름 : 클래스이름으로 표현되기도 하며 객체이름이 생략되기도 합니다.

(4) 패키지 다이어그램(packages diagram)

요소들을 그룹으로 조직하기 위한 범용 메커니즘으로 수입(import)과 수출(export)로 표현합니다. 한 패키지가 수출하는 부분은 그 패키지를 명시적으로 수입하는 패키지의 내용물들에만 보이며 그룹에 접근이 가능하더라도 그룹 내부에서 보호되는 것에는 접근이 불가능 합니다.

예 :



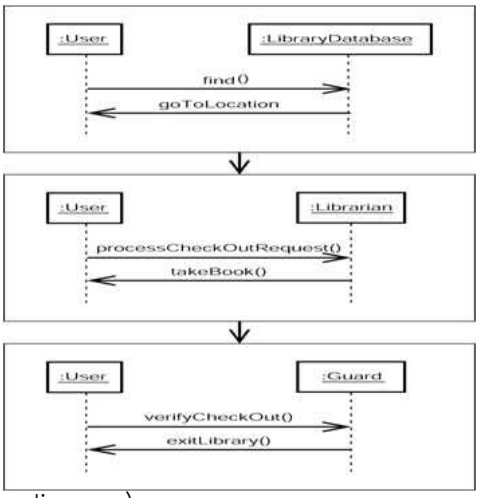
위 그림에서 policies는 명시적으로 GUI 패키지를 수입하고 있습니다. 따라서 policies 패키지의 내용물들은 GUI::window와 GUI::Form을 볼 수 있습니다. 그러나 GUI::EventHandler는 보호되어 있으므로 볼 수 없으며, Server패키지는 GUI를 수입하지 않으므로 어떤 내용물에도 접근할 수 없습니다.

(5) 교류 다이어그램

교류 다이어그램에는 순차 다이어그램(Sequence diagram)과 협력 다이어그램(Collaboration diagram)이 있습니다.

- 순차 다이어그램(Sequence diagram) : 객체들이 어떻게 상호작용 하는지를 메시지 순서에 초점을 맞추어 보여주는 다이어그램으로 2차원으로 그려지며 세로축 아래 방향으로는 발생 시간 순서로 메시지/호출의 흐름을 의미하고 가로축은 메시지가 전송되는 일련의 객체 인스턴스로 구성됩니다.

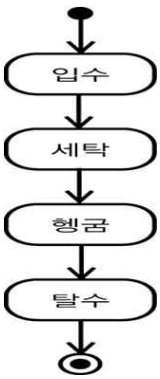
- 협력 다이어그램(Collaboration diagram) : 교류에 참여하는 객체의 구조를 강조하며 객체 역할간의 관계를 보여주고 상호작용이 일어나는 객체와 그들 간의 연결을 중심으로 상호작용을 표현합니다.



(6) 상태 다이어그램(state diagram)

객체는 시간에 따라 각기 다른 상태에 있을 수 있는데, 상태 다이어그램(state diagram)을 통하여 시간에 따른 상태의 변화를 나타낼 수 있습니다.

예 : 세탁기의 단계별 상태 변화



이처럼 여러 가지의 다이어그램을 사용하게 되면 설계된 시스템의 가능한 모든 시점의 다이어그램이 포함되어 있어서 각각의 UML 다이어그램은 자신이 나타내고 있는 시점을 하나로 합칠 수 있는 수단을 제공하여 모든 참여자를 만족시키게 됩니다.

3. UML 도구를 사용하여, OOAD를 수행

3-1UML 모델 및 다이어그램

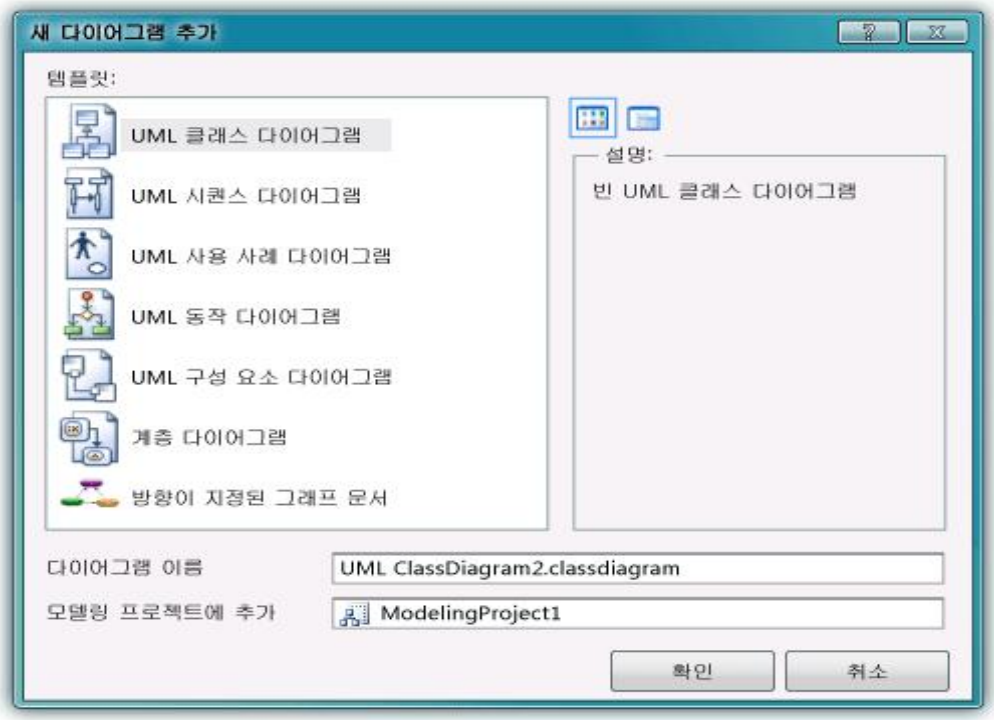
UML 모델 작업
모델을 만들고 다이어그램을 추가합니다.
모델을 편집하기 위해 다이어그램을 그립니다.
모델을 다른 팀 멤버가 작업할 수 있는 단위로 나누기 위해 패키지를 만듭니다.
특정한 목적으로 표준 UML 모델 요소를 확장하기 위해 스테레오타입을 사용하여 모델 요소를 사용자 지정합니다.
작업, 테스트 사례, 버그, 요구 사항, 문제점 또는 모델의 특정 파트와 연결된 다른 작업 유형 등을 추적할 수 있도록 작업 항목을 모델 요소에 연결합니다.
Visual Studio Ultimate을 사용하지 않는 사람을 포함하여 다른 사용자와 공유할 수 있도록 모델과 다이어그램을 저장합니다.

1) UML 모델링 다이어그램 만들기

-모델링 프로젝트에 다이어그램 만들기

다이어그램을 만들어 프로젝트에 추가하려면

① 메뉴에서 새 다이어그램을 클릭하면 새 다이어그램 추가 대화 상자가 나타납니다.



② 원하는 모델링 다이어그램의 형식을 클릭합니다.

③ 새 다이어그램의 이름을 입력합니다.

④ 모델링 프로젝트에 추가 상자에서 다음을 수행합니다.

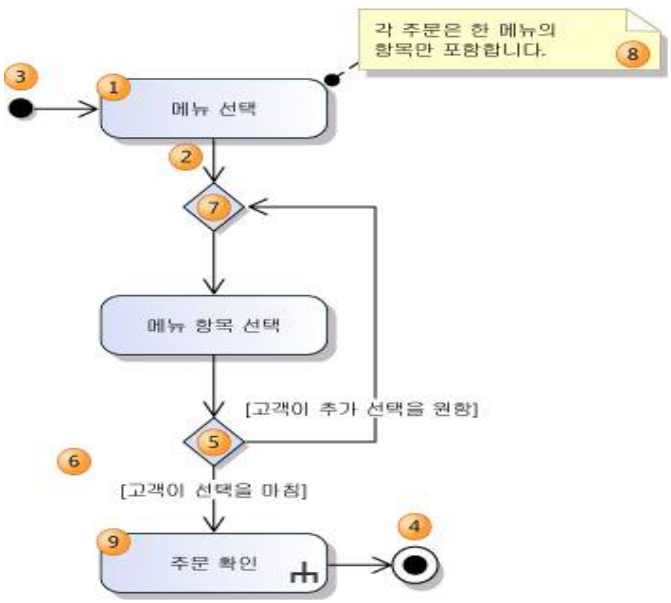
- 솔루션에 이미 있는 모델링 프로젝트를 선택하고 확인을 클릭합니다.

2) UML 모델링 다이어그램 그리기

모델링 다이어그램에는 관계로 연결된 모델 요소의 컬렉션이 표시됩니다. 각 요소는 모양으로 표시되고 각 관계는 두 모양 사이의 연결선으로 표시됩니다. 두 가지 종류의 도구가 있는데, 하나는 요소에 사용되고 다른 하나는 관계에 사용됩니다. 예를 들어 UML 클래스 다이어그램 도구 상자에서 클래스는 요소 도구이고 연결은 관계 도구입니다.

다이어그램	나타내는 요소
동작 다이어그램	비즈니스 프로세스에서 참가자와 동작 간의 워크플로
구성 요소 다이어그램	시스템 구성 요소, 구성 요소의 인터페이스, 포트 및 관계
클래스 다이어그램	시스템에서 데이터를 저장하고 교환하는 데 사용되는 형식 및 형식 관계
시퀀스 다이어그램	개체, 구성 요소, 시스템 또는 행위자 간의 상호 작용 시퀀스
사용 사례 다이어그램	시스템에서 지원하는 사용자 목표 및 작업
레이어 다이어그램	코드에서 서로 다른 파트 간의 종속성

-동작 다이어그램



간단한 제어 흐름

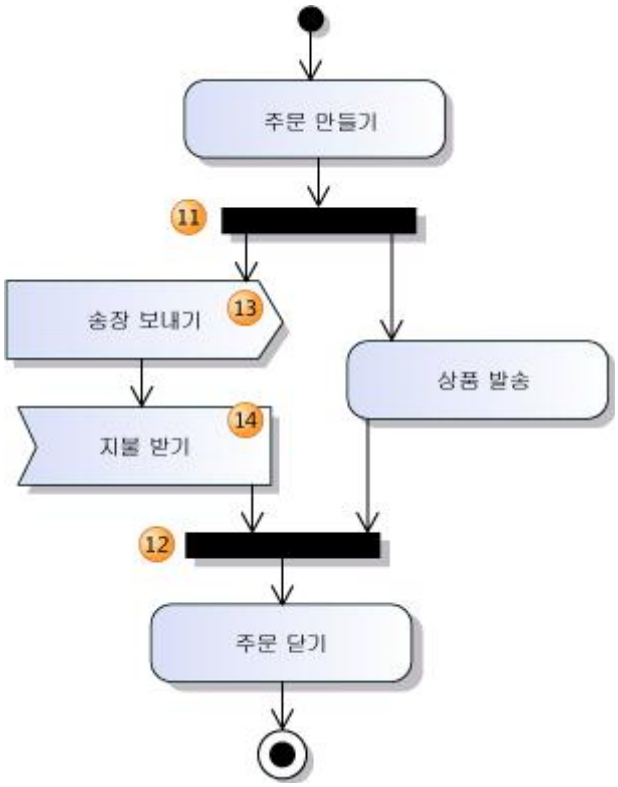
분기와 루프를 사용하여 동작 시퀀스를 나타낼 수 있습니다.

모양	요소	설명 및 주 속성
1	동작	사용자 또는 소프트웨어가 일부 작업을 수행하는 작업 내부 단계입니다. 들어오는 모든 흐름에 토큰이 도달하면 동작이 시작될 수 있습니다.동작이 끝나면 보내는 모든 흐름에 토큰이 전송됩니다. • 본문 - 동작을 자세하게 지정합니다. • 언어 - 본문의 식 언어입니다. • 로컬 사후 조건 - 실행이 끝날 때 충족되어야 하는 제약 조건이며 동작을 통해 달성하는 목표입니다. • 로컬 사전 조건 - 실행이 시작되기 전에 충족되어야 할 제약 조건입니다.
2	제어 흐름	동작 간의 제어 흐름을 보여 주는 연결선입니다. 다이어그램을 해석하려면 토큰이 한 동작에서 다음 동작으로 이동하는 경우를 가정해 봅니다. 제어 흐름을 만들려면 연결선 도구를 사용합니다.
3	초기 노드	작업에서 하나 이상의 첫 번째 동작을 나타냅니다. 작업이 시작되면 초기 노드에서 토큰이 이동합니다.
4	동작 최종 노드	작업의 끝입니다. 토큰이 도착하면 작업이 종료됩니다.
5	의사 결정 노드	흐름에서 조건부 분기입니다. 하나의 입력과 둘 이상의 출력을 포함합니다. 들어오는 토큰은 한 출력에서만 나타납니다.
6	가드	토큰이 연결선을 따라 흐를 수 있는지 여부를 지정하는 조건입니다. 의사 결정 노드의 보내는 흐름에서 주로 사용됩니다. 가드를 설정하려면 흐름을 마우스 오른쪽 단추로 클릭하고 속성을 클릭한 다음, 가드 속성을 설정합니다.
7	병합 노드	의사 결정 노드로 분할된 흐름을 병합하는 데 필요합니다. 둘 이상의 입력과 하나의 출력을 포함합니다. 어떤 입력의 토큰이라도 출력에서 나타납니다.
8	주석	연결된 요소에 대한 추가 정보를 제공합니다.
9	동작 호출 동작	다른 동작 다이어그램에서 보다 자세하게 정의되는 동작입니다. • 동기 - true이면 작업이 종료할 때까지 동작이 대기합니다. • 동작 - 호출된 동작입니다.
(표시되지 않음)	작업 호출 동작	클래스 인스턴스에서 작업을 호출하는 동작입니다.
	동작	동작 다이어그램에서 나타내는 워크플로입니다. 동작의 속성을 표시하려면 UML 모델 탐색기에서 해당 동작을 선택해야 합니다. • 읽기 전용 - true이면 개체의 상태를 변경하지 않아야 합니다. • 단일 실행 - true이면 한 번에 이 다이어그램을 실행할 수 있는 횟수가 최대 1회입니다.
	UML 동작 다이어그램	동작을 표시하는 다이어그램입니다. 속성을 표시하려면 다이어그램의 빈 부분을 클릭합니다.

		동작 다이어그램의 이름, 다이어그램을 포함하는 파일 및 다이어그램에서 표시하는 동작이 모두 다를 수 있습니다.

-동시 흐름

동시에 실행되는 동작의 시퀀스를 기술할 수 있습니다.

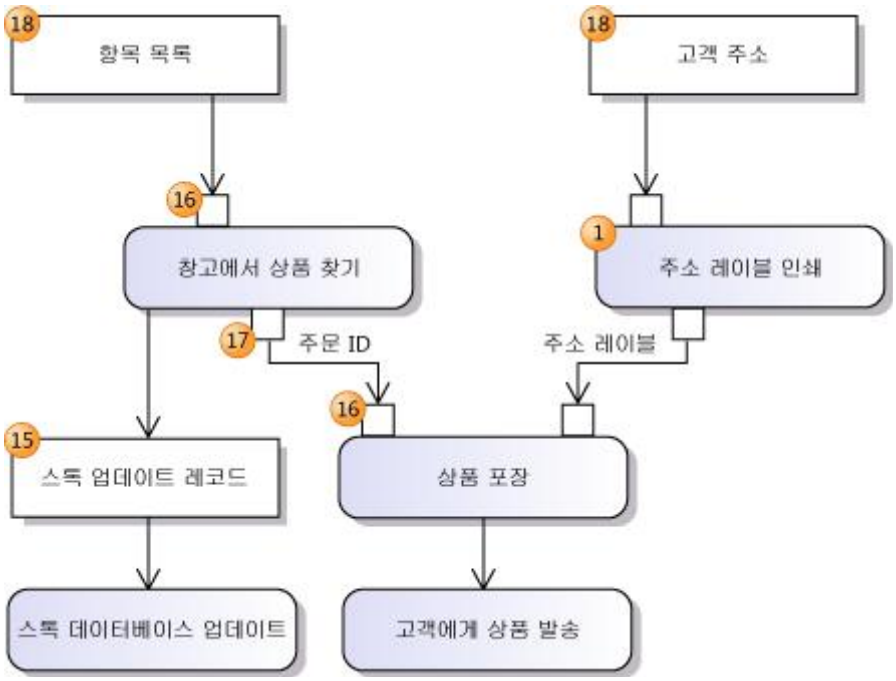


모양	요소	설명
11	분기 노드	단일 흐름을 동시 흐름으로 나눕니다.들어오는 각 토큰은 보내는 각 연결선에서 토큰을 생성합니다.
12	조인 노드	동시 흐름을 단일 흐름으로 결합합니다. 들어오는 모든 흐름에 대기 중인 토큰이 있는 경우 출력에서 토큰이 생성됩니다.
13	신호 보내기 동작	다른 동작 또는 같은 동작의 동시 스레드에 메시지나 신호를 보내는 동작입니다.메시지의 형식과 내용은 동작의 제목에 포함되거나 추가 주석에 지정됩니다. 이 동작은 데이터를 신호로 전송하며 이렇게 신호로 전송된 데이터는 입력 핀(16) 또는 개체 흐름의 동작에 전달될 수 있습니다.
14	이벤트 적용 동작	동작을 계속하기 전에 메시지 또는 신호를 기다리는 동작입니다. 이 동작에서 받을 수 있는 메시지의 형식은 제목에 포함되거나 추가 주석이 지정됩니다. 동작에 들어오는 제어 흐름이 없으면 메시지를 받을 때마다 토큰을 생성합니다. 이 동작은 데이터를 신호로 받으며 이렇게 신호로 받은 데이터는 출력 핀(17) 또는 개체 흐름에 전달될 수 있습니다. • 역 마샬링 - true이면 형식화된 출력 핀이 여러 개 있을 수 있

		으며 데이터가 이러한 출력 핀으로 역 마샬링됩니다. false이면 모든 데이터가 한 핀에 나타납니다.

-데이터 흐름

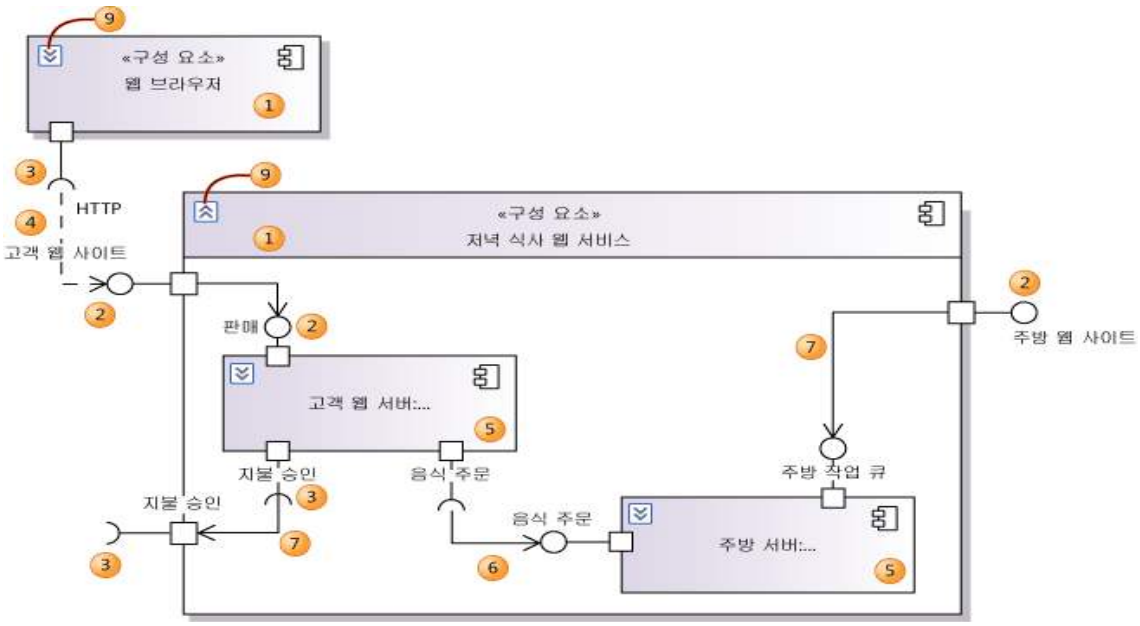
동작 간의 데이터 흐름을 기술할 수 있습니다.이 단원에 사용되는 요소에 대한 자세한 내용은 Guidelines for Drawing an Activity Diagram을 항목의 Drawing Data Flows 단원을 참조하십시오.



모양	요소	설명
15	개체 노드	흐름을 따라 지나가는 데이터를 나타냅니다. • 순서 지정 - 여러 토큰이 저장되는 방식입니다. • 선택 영역 - 다른 다이어그램에 정의할 수 있는 데이터 필터링 프로세스를 호출합니다. • 상한 - 0은 데이터가 흐름을 바로 지나가야 한다는 것을 나타내고 *는 데이터를 흐름에 저장할 수 있다는 것을 나타냅니다. • 형식 - 저장 및 전송된 개체의 형식입니다.
16	입력 핀	동작이 실행될 때 받을 수 있는 데이터를 나타냅니다. • 형식 - 전송된 개체의 형식입니다.
17	출력 핀	동작이 실행될 때 생성되는 데이터를 나타냅니다. • 형식 - 전송된 개체의 형식입니다.
18	동작 매개 변수 노드	동작에 의해 데이터를 생성하거나 받을 때 사용할 수 있는 개체 노드입니다. 다이어그램에서 나타내는 동작이 다른 동작에서 호출되거나 다이어그램에서 작업 또는 기능을 기술할 때 사용됩니다. • 형식 - 전송된 개체의 형식입니다.

(표시되지 않음)	개체 흐름	동작과 개체 노드 간의 데이터 흐름을 보여 주는 연결선입니다. 개체 흐름을 만들려면 연결선 도구를 사용하여 입력 핀이나 출력 핀 또는 개체 노드를 다른 요소에 연결합니다. • 선택 영역 – 다른 다이어그램에 정의할 수 있는 데이터 필터링 프로세스를 호출합니다. • 변환 – 다른 다이어그램에 정의할 수 있는 데이터 변환 프로세스를 호출합니다. • 멀티캐스트 – 수신자 개체 또는 구성 요소가 여러 개 있을 수 있음을 나타냅니다. • 다중 수신 – 여러 개체 또는 구성 요소로부터 입력을 받을 수 있음을 나타냅니다.

-구성 요소 다이어그램



모양	요소	설명 및 주 속성
1	구성 요소	재사용 가능한 시스템 기능 요소입니다. 구성 요소는 인터페이스를 통해 동작을 제공하고 사용하며 다른 구성 요소를 사용할 수 있습니다. 확장/축소 컨트롤(9)을 사용하여 구성 요소의 내부 파트를 숨기거나 표시할 수 있습니다. 구성 요소는 클래스의 한 종류입니다. • 간접 인스턴스화됨true(기본값)이면 구성 요소가 디자인 아티팩트로만 존재하고 런타임에 파트만 존재합니다.
2	제공된 인터페이스 포트	구성 요소에서 구현하고 다른 구성 요소나 외부 시스템에서 사용할 수 있는 그룹 메시지 또는 호출을 나타냅니다. 포트는 인터페이스를 형식으로 갖는 구성 요소의 속성입니다.
3	필요한 인터페이	구성 요소에서 다른 구성 요소나 외부 시스템에 보내는 그룹 메시지 또는 호출을 나타냅니다. 구성 요소는 최소한 이러한 작업을 제공하는 구성

	스 포트	요소와 결합하도록 디자인되었습니다. 포트는 인터페이스를 형식으로 갖습니다.
4	종속성	한 구성 요소의 필요한 인터페이스가 다른 구성 요소의 제공된 인터페이스에 의해 충족된다는 것을 나타내는 데 사용할 수 있습니다. 종속성은 모델 요소 간에 한 요소의 디자인이 다른 요소의 디자인에 의존한다는 것을 나타낼 때 좀 더 일반적으로 사용될 수 있습니다.
5	파트	일반적으로 다른 구성 요소를 형식으로 갖는 구성 요소의 특성입니다. 파트는 부모 구성 요소의 내부 디자인에 사용되며 부모 구성 요소 내에 중첩된 모양으로 표시됩니다. 기존 구성 요소 형식의 파트를 만들려면 UML 모델 탐색기의 구성 요소를 소유자 구성 요소로 끌어 옵니다. 새 형식의 파트를 만들려면 구성 요소 도구를 클릭하고 소유자 구성 요소를 클릭합니다. 예를 들어 Car 구성 요소에는 engine:CarEngine, backLeft:Wheel, frontRight:Wheel 등의 파트가 있습니다. 둘 이상의 파트에 같은 형식을 사용할 수 있으며 서로 다른 구성 요소에 같은 형식의 파트가 있을 수 있습니다. • 형식. 모델에서 다른 곳에 정의되는 파트의 형식입니다. 일반적으로 형식은 또 하나의 구성 요소입니다. • 복합성.기본값은 1입니다. 이 값을 0..1로 설정하여 파트가 null 값을 가질 수 있음을 나타내거나, *로 설정하여 파트가 특정 형식의 인스턴스 컬렉션임을 나타내거나, 숫자 범위로 확인되는 식으로 설정할 수 있습니다.
6	파트 어셈블리	한 파트의 필요한 인터페이스 포트와 다른 파트의 제공된 인터페이스 포트 간 연결입니다.파트 어셈블리는 구성 요소에 따라 다르게 구현될 수 있습니다.연결된 파트에는 동일한 부모 구성 요소가 있어야 합니다.
7	위임	구성 요소에 포함된 한 파트의 인터페이스에 포트를 연결합니다.구성 요소에 전송되는 메시지가 파트에서 처리되거나, 파트에서 보내는 메시지가 부모 구성 요소에서 전송된다는 것을 나타냅니다.
8	일반화	한 구성 요소가 다른 구성 요소에서 상속됨을 나타냅니다.파트와 인터페이스는 상속됩니다.
9	축소/확장 컨트롤	이 컨트롤을 사용하면 구성 요소의 내부 파트를 숨기거나 표시할 수 있습니다.
(표시되지 않음)	주석	추가 설명이 필요한 경우 사용합니다. 연결선 도구를 사용하여 다이어그램의 여러 요소에 주석을 연결할 수 있습니다.

-클래스 다이어그램

UML 클래스 다이어그램 형식의 속성

속성	기본값	속성이 나타나는 요소	설명
이름	기본 이름	모든 요소	요소를 식별합니다.
정규화된 이름	포함하는 패키지 :: 형식 이름	모든 요소	요소를 고유하게 식별합니다.요소를 포함하는 패키지의 정규화된 이름으로 접두사가 지정됩니다.
색	형식 종류의 기본값	모든 요소	이 도형의 색입니다.다른 속성과 달리 이 속성은 기본 모델 요소의 속성이 아닙니다.같은 형식의 여러 뷰에 서로 다른 색을 사용할 수 있습니다.

추상	False	클래스	true이면 클래스를 인스턴스화할 수 없으며 기본 클래스로 사용됩니다.
리프	False	클래스, 인터페이스	true이면 형식이 파생 형식을 가질 수 없습니다.
활성	False	클래스	true이면 이 형식의 각 인스턴스가 컨트롤 스레드와 연결됩니다.
표시 유형	공용	클래스, 인터페이스, 열거형	• 공용 - 전체에 표시됩니다. • 전용 - 이 형식은 자신을 소유하는 패키지 내에 표시됩니다. • 패키지 - 패키지 내에서만 표시됩니다.
작업 항목	0개 연결	모든 요소	이 요소와 연결된 작업 항목의 수입니다.
설명	(비어 있음)	모든 요소	항목에 대한 전반적인 설명을 여기에 입력할 수 있습니다.
템플릿 바인딩	(없음)	클래스, 인터페이스, 열거형	비어 있지 않으면 이 형식은 매개 변수 값을 이 템플릿 클래스에 바인딩하여 정의됩니다.템플릿 매개 변수의 바인딩을 표시하려면 이 속성을 확장합니다.
템플릿 매개 변수	(없음)	클래스, 인터페이스, 열거형	비어 있지 않으면 이 속성은 여기 나열된 매개 변수를 포함하는 템플릿 클래스입니다. 매개 변수를 추가하거나 각 매개 변수의 속성을 보려면 […] 모양을 클릭합니다.

-UML 클래스 다이어그램 특성의 속성

특성 시그니처는 UML 클래스 다이어그램의 클래스 또는 인터페이스에서 해당 특성을 나타내는 줄입니다.시그니처 형태는 다음과 같습니다.

+ AttributeName : TypeName [*]

+는 공용 표시 유형을 나타냅니다.-(전용), #(보호됨), ~(패키지) 등의 값도 허용됩니다.

특성이 정적이면 AttributeName에 밑줄이 표시됩니다.

특성에 형식이 없으면 : TypeName이 생략됩니다.

[*]는 복합성을 나타냅니다.따라서 복합성이 1이면 생략됩니다.

속성	기본값	설명
기본값	(비어 있음)	분류자가 인스턴스화되는 경우 특성의 값입니다.
읽기 전	False	true이면 특성의 값을 변경할 수 없습니다.
정적	False	true이면 이 특성의 단일 값이 이 형식의 모든 인스턴스에서 공유됩니다. true이면 다이어그램에 나타날 때 특성 이름에 밑줄이 표시됩니다.
이름	(새	소유하는 분류자 내에서 고유해야 합니다.

	이름)	
형식	(없음)	정수와 같은 기본 형식이거나 모델에 정의된 형식입니다.이 속성에 새 형식의 이름을 입력하면 UML 모델 탐색기의 지정되지 않은 형식에 형식이 추가됩니다.
표시 유형	공용	다음은 시그니처에 나타나는 문자 및 허용된 값입니다. + 공용 - 전체에 표시됩니다. - 전용 - 소유하는 형식 외부에 표시되지 않습니다. # 보호됨 - 소유자로부터 파생된 형식에 표시됩니다. ~ 패키지 - 같은 패키지에 있는 다른 형식에 표시됩니다.
작업 항목	0개 연결	연결된 작업 항목의 수입니다.읽기 전용입니다.
리프	False	true이면 파생 형식에서 이 특성을 재정의할 수 없습니다.
파생	False	true이면 이 특성이 다른 특성에서 계산됩니다.예를 들어 Diagonal은 Width와 Height에서 계산됩니다.세부 정보는 설명에 작성하거나 주석으로 연결되어야 합니다.
설명	(비어 있음)	전반적인 설명을 입력하거나, 특성의 값에 대한 제약 조건을 정의할 수 있습니다.
복합성	1	1 - 이 특성에는 지정된 형식의 단일 값이 있습니다. 0..1 - 이 특성에는 null 값이 있을 수 있습니다. * - 이 특성의 값은 값 컬렉션입니다. 1..* - 이 특성의 값은 적어도 하나의 값을 포함하는 컬렉션입니다. n..m - 이 특성의 값은 n개에서 m개 사이의 값을 포함하는 컬렉션입니다.
순서 지정됨	False	true이면 컬렉션이 순차 목록을 구성합니다.복합성이 1보다 큰 경우에 사용됩니다.
고유	False	true이면 컬렉션에 중복 값이 없습니다.복합성이 1보다 큰 경우에 사용됩니다.

-UML 클래스 다이어그램 작업의 속성

작업 시그니처는 UML 클래스 다이어그램의 클래스 또는 인터페이스에서 해당 작업을 나타내는 텍스트 줄입니다. 시그니처 형태는 다음과 같습니다.

+ OperationName (parameter1 : Type1 [*], ...) : ReturnType [*]

+는 공용 표시 유형을 나타냅니다.-(전용), #(보호됨), ~(패키지) 등의 값도 허용됩니다.

OperationName은 정적 속성이 true이면 밑줄이 표시되고 추상 속성이 true이면 기울임꼴로 표시됩니다.

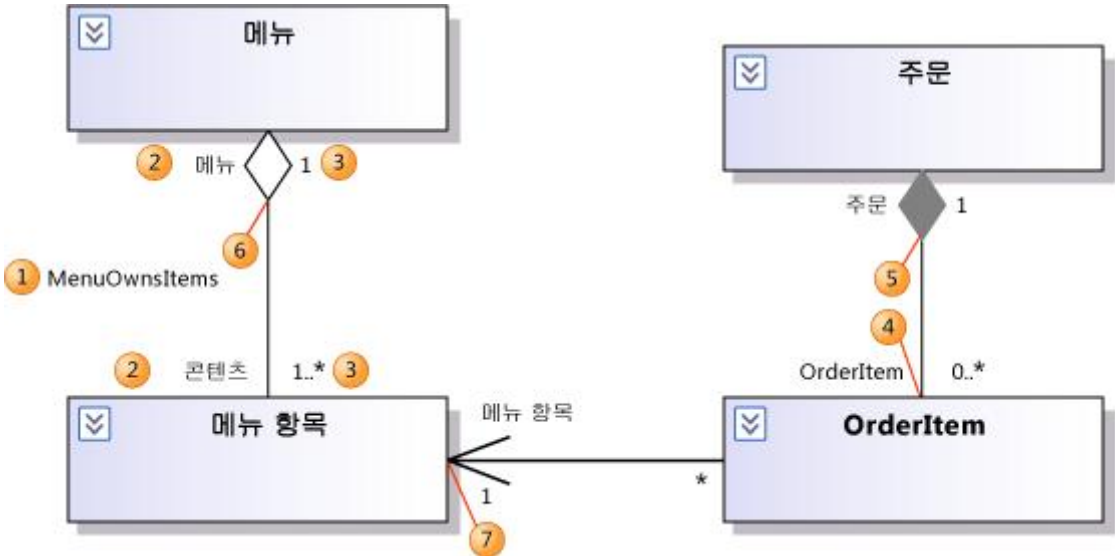
반환 형식이 정의되지 않으면 : ReturnType이 생략됩니다.

[*]는 매개 변수 또는 반환 형식이 다양함을 나타냅니다.따라서 복합성이 1이면 생략됩니다.

속성	기본값	설명
이름	(새 이름)	포함하는 형식 내에서 고유해야 합니다.
매개	(없음)	name:Type, name:Type, … 형식의 목록입니다. 목록을 편집하려면

변수		[…] 기호를 클릭합니다. 형식은 기본 형식이거나 모델에 정의된 형식일 수 있습니다.이 속성에 새 형식의 이름을 입력하면 UML 모델 탐색기의 지정되지 않은 형식에 형식이 추가됩니다.
반환 형식	(없음)	(없음), 기본 형식 또는 모델에 정의된 형식입니다.이 속성에 새 형식의 이름을 입력하면 UML 모델 탐색기의 지정되지 않은 형식에 형식이 추가됩니다.
사후 조건	(없음)	작업 실행 전후에 시스템 상태의 관계를 지정하는 선택적 조건입니다.
사전 조건	(없음)	작업이 실행되기 전에 시스템 상태에 대한 가정을 지정하는 선택적 조건입니다.
본문 조건	(없음)	작업에서 반환하는 값에 대한 선택적 제약 조건입니다.
표시 유형	공용	다음은 시그니처에 나타나는 문자 및 허용된 값입니다. + 공용 - 전체에 표시됩니다. - 전용 - 소유하는 형식 외부에 표시되지 않습니다. # 보호됨 - 소유자로부터 파생된 형식에 표시됩니다. ~ 패키지 - 같은 패키지에 있는 다른 형식에 표시됩니다.
시그니처	+Name()	이 작업의 표시 유형, 이름, 매개 변수 및 반환 형식을 요약합니다.다이어그램에서 시그니처를 편집하거나 개별 속성을 편집하여 이러한 속성을 변경할 수 있습니다.
작업 항목	0개 연결	연결된 작업 항목의 수입니다.읽기 전용입니다.
동시성	순차	순차 - 동시성 제어 없이 작업이 디자인됩니다.이 작업을 동시에 호출하면 오류가 발생할 수 있습니다. 가드됨 - 다른 인스턴스가 완료될 때까지 자동으로 작업이 차단됩니다. 동시 - 여러 호출을 동시에 실행할 수 있도록 작업이 디자인됩니다.
정적	False	true이면 이 작업은 이 형식의 모든 인스턴스에서 공유됩니다. true이면 다이어그램에 나타날 때 작업 이름에 밑줄이 표시됩니다.
추상	False	true이면 이 작업에 코드가 연결되지 않습니다.따라서 소유 클래스는 추상적입니다.
리프	False	디자이너는 파생 클래스에서 이 작업을 재정의할 수 없도록 합니다.
쿼리	False	true이면 이 작업에 의해 시스템 상태가 크게 변경되지 않습니다.따라서 시스템 상태를 확인하기 위한 테스트에 이 속성을 사용할 수 있습니다.
복합성	1	1 - 지정된 형식의 단일 값입니다. 0..1 - null일 수 있습니다. * - 지정된 형식의 값 컬렉션입니다. 1..* - 최소한 하나 이상의 값을 포함하는 컬렉션입니다. n..m - n개에서 m개 사이의 값을 포함하는 컬렉션입니다.
순서 지정됨	False	true이면 컬렉션이 순차 목록을 구성합니다.복합성이 1보다 큰 경우에 사용됩니다.
고유	False	true이면 컬렉션에 중복 값이 없습니다.복합성이 1보다 큰 경우에 사용됩니다.

-UML 클래스 다이어그램 연결의 속성



속성	설명
이름(1)	연결을 식별합니다. 다이어그램에서 연결의 중간점 근처에도 나타납니다.
정규화된 이름	연결을 고유하게 식별합니다. 연결의 첫 번째 역할을 포함하는 패키지의 정규화된 이름으로 접두사가 지정됩니다.
작업 항목	이 연결에 연결된 작업 항목의 수입니다.
색	연결선의 색입니다.다른 속성과 달리 이 속성은 모델에서 기본 관계의 속성이 아니라 이 연결 뷰의 속성입니다.
첫 번째 역할 두 번째 역할	연결의 각 끝을 역할이라고 합니다. 각 역할은 연결의 반대쪽 끝에 있는 클래스에서 이와 동등한 특성의 속성을 기술합니다. 예제 다이어그램에서 Menu와 Menu Item 사이의 연결에는 Menu와 Contents라는 역할이 있습니다. Contents는 Menu 클래스의 특성 이름입니다.

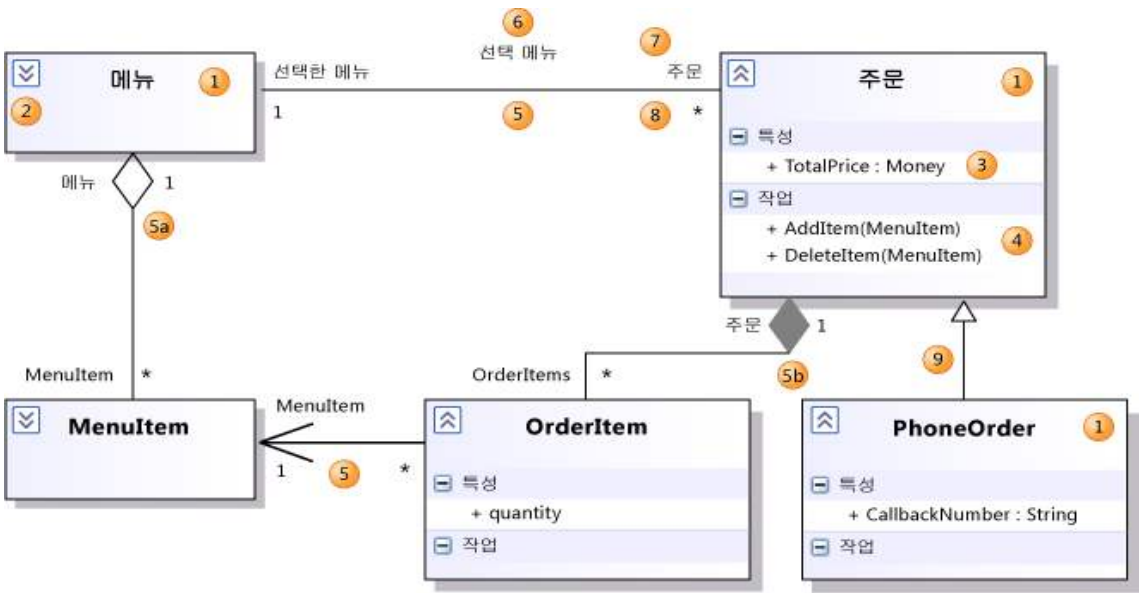
-각 역할의 속성

각 역할의 속성을 보려면 첫 번째 역할 또는 두 번째 역할 속성을 확장합니다.

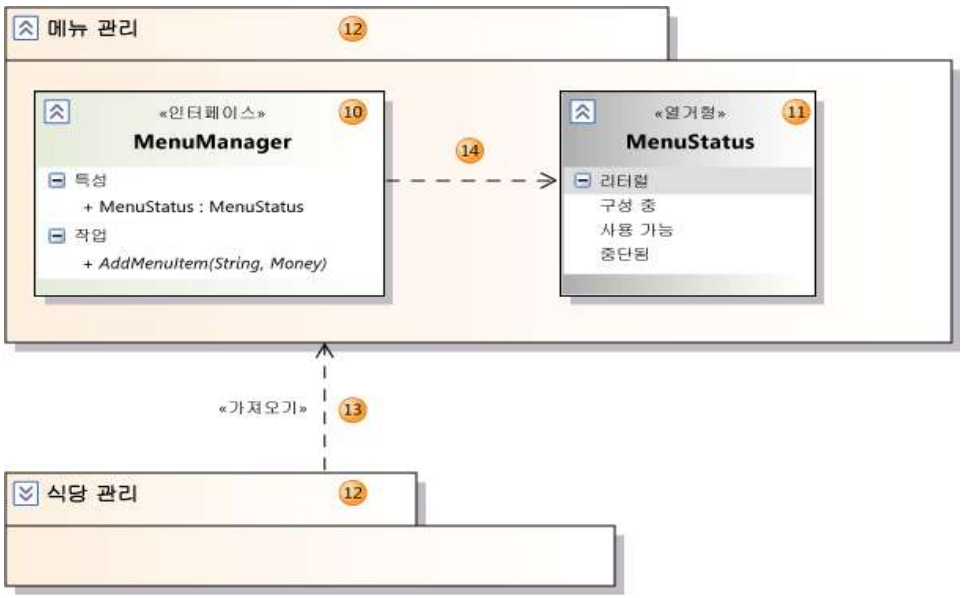
속성	기본값	설명
역할 이름(2)	이 역할의 형식 이름	역할의 이름이며,다이어그램에서 연결의 끝 근처에 나타납니다.
집계	없음	없음(4) - 클래스 인스턴스 간의 일반적인 관계를 나타냅니다. 복합(5) - 이 역할의 개체는 반대 역할의 개체를 포함합니다.복합 도구를 사용하여 복합 집합체와의 연결을 만들 수 있습니다. 공유(6) - 이 역할의 개체는 다른 역할의 개체에 대한 참조를 포함합니다.집합체 도구를 사용하여 공유 집합체와의 연결을 만들 수 있습니다. 정확한 해석이 지역 규칙에 공개되어 있습니다.
파생	False	true이면 링크의 이 끝에 있는 개체가 다른 특성 및 연결에서 계산됩니다.예를 들어 MyWorkPlace는 MyEmployer.WorkPlace에서 계산됩니다.세부 정보는 설명에 입력하거나 주석으로 연결되어야 합니다.
파생 Union	False	true이면 이 역할은 파생 형식의 역할 집합을 결합한 것입니다.

탐색 가능	True	이 방향으로 연결을 읽을 수 있습니다.반대 역할의 인스턴스가 있으면 기술하는 소프트웨어에서 이 역할의 연결된 인스턴스를 효율적으로 결정할 수 있습니다. 한 역할만 탐색 가능한 경우에는 연결에서 탐색 가능 방향으로 화살표(7)가 나타납니다. 기본적으로 연결 도구는 한 방향으로 탐색 가능한 연결을 만듭니다. 이를 양방향 연결로 변환하려면 연결을 선택하고 스마트 태그를 클릭한 다음, 양방향 만들기를 클릭합니다.
읽기 전용	False	true이면 연결의 인스턴스를 만든 후에는 변경할 수 없습니다.링크가 항상 같은 개체를 대상으로 합니다.
복합성(3)	1	1 - 연결의 이 끝이 항상 하나의 개체에 연결됩니다.그림에서 모든 Menu Item에는 하나의 Menu가 있습니다. 0..1 - 연결의 이 끝이 하나의 개체에 연결되거나 링크가 없습니다. * - 연결의 다른 끝에 있는 모든 개체가 이 끝에 있는 개체 컬렉션에 연결되고 컬렉션은 비어 있을 수 있습니다. 1..* - 연결의 다른 끝에 있는 모든 개체가 이 끝에서 최소한 하나 이상의 개체에 연결됩니다.그림에서 모든 Menu에는 Menu Item이 최소한 하나 이상 있습니다. n..m - 다른 끝에 있는 각 개체는 이 끝에 있는 개체에 대한 n ~ m개의 링크 컬렉션을 포함합니다.
순서 지정됨	False	true이면 반환된 컬렉션이 순차 목록을 구성합니다.복합성이 1보다 큰 경우에 사용됩니다.
고유	False	true이면 반환된 컬렉션에 중복 값이 없습니다.복합성이 1보다 큰 경우에 사용됩니다.
표시 유형	공용	공용 - 전체에 표시됩니다. 전용 - 소유하는 형식 외부에 표시되지 않습니다. 보호됨 - 소유자로부터 파생된 형식에 표시됩니다. 패키지 - 같은 패키지에 있는 다른 형식에 표시됩니다.

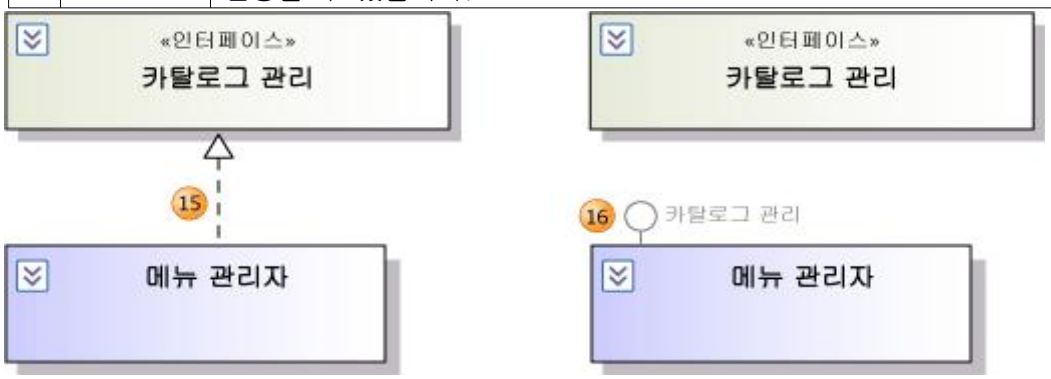
-UML 클래스 다이어그램



모양	요소	설명
1	클래스	지정된 구조 또는 동작 특징을 공유하는 개체의 정의입니다.
1	분류자	클래스, 인터페이스 또는 열거형의 일반 이름입니다. 구성 요소, 사용 사례 및 행위자도 분류자입니다.
2	축소/확장 컨트롤	분류자의 세부 정보를 볼 수 없으면 해당 분류자의 왼쪽 위에 있는 확장기를 클릭합니다. 각 세그먼트에서 [+]를 클릭해야 할 수도 있습니다.
3	특성	분류자의 각 인스턴스에 연결된 형식화된 값입니다. 특성을 추가하려면 특성 섹션을 클릭하고 Enter 키를 누릅니다. 그런 다음 특성의 시그니처를 입력합니다.
4	작업	분류자 인스턴스에서 수행할 수 있는 메서드 또는 함수입니다. 작업을 추가하려면 작업 섹션을 클릭하고 Enter 키를 누릅니다. 그런 다음 작업의 시그니처를 입력합니다.
5	연결	두 분류자의 멤버 간 관계입니다.
5a	집계	공유 소유권 관계를 나타내는 연결입니다. 소유자 역할의 집합체 속성은 공유로 설정됩니다.
5b	컴퍼지션	전체와 부분 관계를 나타내는 연결입니다. 소유자 역할의 집합체 속성은 복합으로 설정됩니다.
6	연결 이름	연결 이름입니다. 이 이름은 비워 둘 수 있습니다.
7	역할 이름	연결의 한 쪽 끝에 있는 역할의 이름입니다. 이 이름은 연결된 개체를 참조하는 데 사용할 수 있습니다.이전 그림에서 임의의 Order O에 대해 O.ChosenMenu는 연결된 Menu입니다. 각 역할에는 고유한 속성이 있으며, 이러한 속성은 연결 속성 아래에 나열됩니다.
8	복합성	이 쪽 끝의 개체 중에서 다른 쪽의 각 개체에 연결할 수 있는 개체의 수를 나타냅니다. 이 예제에서 각 Order는 정확히 하나의 Menu에만 연결되어야 합니다. *는 연결할 수 있는 링크 수에 상한이 없음을 의미합니다.
9	일반화	특정 분류자는 일반 분류자에서 정의의 일부를 상속합니다. 일반 분류자는 연결선의 화살표 끝에 있습니다. 특성, 연결 및 작업은 특정 분류자에서 상속합니다. 두 분류자 간의 일반화 관계를 만들려면 상속 도구를 사용합니다.



모양	요소	설명
10	인터페이스	외부에서 볼 수 있는 개체 동작의 일부에 대한 정의입니다.
11	열거형	리터럴 값 집합으로 구성되는 분류자입니다.
12	패키지	분류자, 연결, 동작, 수명선, 구성 요소 및 패키지 그룹입니다. 논리 클래스 다이어그램은 멤버 분류자와 패키지가 패키지 내에 포함되어 있음을 나타냅니다. Package1 내의 Class1이 패키지 외부의 Class1과 구별되도록 이름은 패키지 내에서 범위가 지정됩니다. 패키지 이름은 콘텐츠의 정규화된 이름 속성의 일부로 나타납니다. UML 다이어그램의 연결된 패키지 속성을 설정하여 패키지를 참조할 수 있습니다.그러면 해당 다이어그램에서 만드는 모든 요소가 패키지의 일부가 됩니다. 이러한 요소는 UML 모델 탐색기에서 패키지 아래에 나타납니다.
13	namespace	한 패키지가 다른 패키지의 모든 정의를 포함한다는 것을 나타내는 패키지 간 관계입니다.
14	종속성	화살촉 끝에 있는 분류자가 변경되면 종속 분류자의 정의 또는 구현이 변경될 수 있습니다.



모양	요소	설명
15	인식	클래스는 인터페이스에서 정의하는 작업 및 특성을 구현합니다. 클래스와 인터페이스 간의 인식 관계를 만들려면 상속 도구를 사용합니다.
16	인식	같은 관계의 대체 표현입니다. 롤리팝 기호의 레이블은 인터페이스를 식별합니다. 이 표현을 만들려면 기존 인식 관계를 선택합니다. 그러면 연결 근처에 스마트 태그가 나타납니다.이 스마트 태그를 클릭하고 롤리팝으로 표시를 클릭합니다.

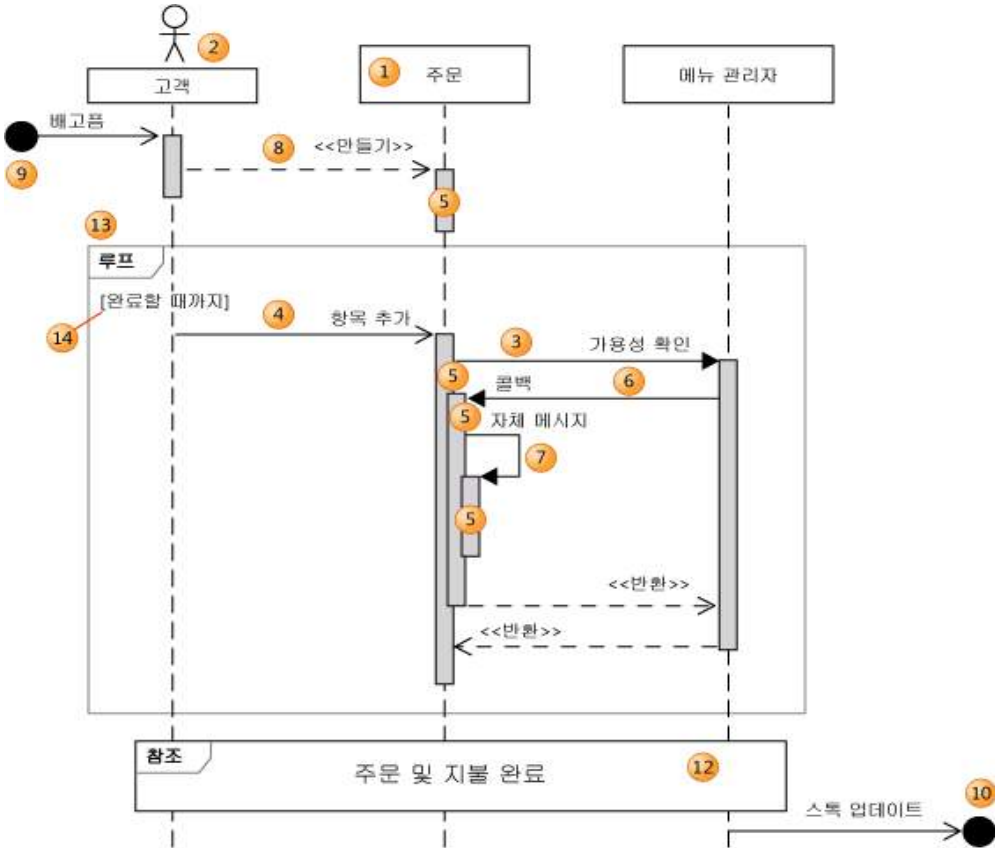
-시퀀스 다이어그램

시퀀스 다이어그램에는 다음과 같은 두 가지 종류가 있습니다.

① .NET 시퀀스 다이어그램은 프로그램 코드에서 생성할 수 있으며 프로젝트 내에 배치할 수 있습니다.

② UML 시퀀스 다이어그램은 UML 모델의 일부이며 UML 모델링 프로젝트 내에서만 존재합니다.

요소의 일부 속성이 다르지만 두 종류의 시퀀스 다이어그램은 비슷합니다.

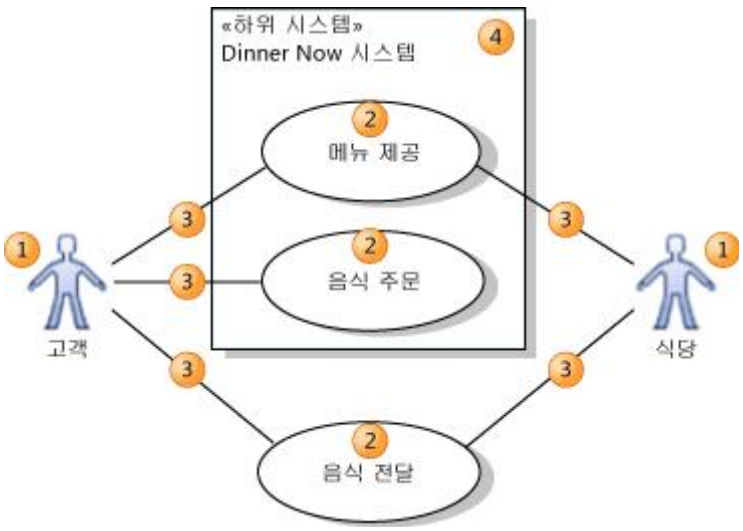


모양	요소	Description
1	수명선	줄 아래로 시간이 진행하는 동안 상호 작용 중에 참가자에 발생하는 이벤트 시퀀스를 나타내는 세로 선입니다. 이 참가자는 클래스, 구성 요소 또는 행위자 인스턴스가 될 수 있습니다.
2	행위자	개발 중인 시스템 외부의 참가자입니다. 행위자 속성을 설정하여 수명선 맨 위에 행위자 기호가 나타나도록 만들 수 있습니다.
3	동기 메시지	보낸 사람은 계속하기 전에 동기 메시지에 대한 응답을 기다립니다. 다이어그램은 호출 및 반환을 모두 보여줍니다. 동기 메시지는 동일한 방식으로 동작하는 다른 종류의 메시지 뿐만 아니라 프로그램 내에서 일반 함수 호출을 나타내는 데 사용됩니다.
4	비동기 메시지	보낸 사람이 계속하기 전에 응답을 보낼 필요가 없는 메시지입니다. 비동기 메시지는 보낸 사람의 호출만 보여줍니다. 별도의 스레드 또는 새 스레드의 컬렉션 사이의 통신을 나타냅니다.
5	실행 발생	참가자의 수명선에 나타나고 참가자가 작업을 실행하는 기간을 나타내는 세로 음영의 사각형입니다. 실행은 참가자가 메시지를 받는 곳에서 시작됩니다. 시작 메시지가 동기 메시지인 경우 실행은 보낸 사람에게 다시 «반환» 화살표로 종료됩니다.
6	콜백 메시지	이전 호출에서 반환을 기다리는 참가자에게 다시 반환하는 메시지입니다. 결과 실행 발생은 기존 발생 위에 나타납니다.
7	자체 메시지	참가자가 자신에게 보내는 메시지입니다. 결과 실행 발생은 보내는 실행 위에 나타납니다.

8	메시지 만들기	참가자를 만드는 메시지입니다. 참가자가 메시지 만들기를 받으면 수신자가 받는 첫 번째 메시지여야 합니다.
9	찾기 메시지	알 수 없거나 지정되지 않은 참가자의 비동기 메시지입니다.
10	손실된 메시지	알 수 없거나 지정되지 않은 참가자에게 보내는 비동기 메시지입니다.
11	주석	메모는 수명선의 모든 점에 첨부할 수 있습니다.
12	상호 작용 사용	다른 다이어그램에 정의되어 있는 메시지의 시퀀스를 묶습니다. 상호 작용 사용을 만들려면 도구를 클릭한 다음 포함하려는 수명선으로 끕니다.
13	결합 조각	조각의 컬렉션입니다. 각 조각은 메시지를 하나 이상 포함할 수 있습니다.결합 조각에는 여러 가지 종류가 있습니다. 조각을 만들려면 메시지를 마우스 오른쪽 단추로 클릭하고 코드 감싸기를 가리킨 다음, 조각의 형식을 클릭합니다.
14	조각 가드	조각이 발생하는지 여부와 관련한 조건을 지정하는 데 사용할 수 있습니다. 가드를 설정하려면 단편을 선택한 다음 가드를 선택한 값을 입력합니다.
	상호 작용	메시지의 컬렉션은 시퀀스 다이어그램에서 표시되는 수명선입니다. 상호 작용의 속성을 보려면 UML 모델 탐색기에서 해당 동작을 선택해야 합니다.
	시퀀스 다이어그램	상호 작용을 표시하는 다이어그램입니다. 속성을 보려면 다이어그램의 빈 부분을 클릭합니다. **참고** 시퀀스 다이어그램의 이름, 다이어그램에서 표시하는 상호 작용 및 다이어그램을 포함하는 파일이 모두 다를 수 있습니다.

-사용 사례 다이어그램

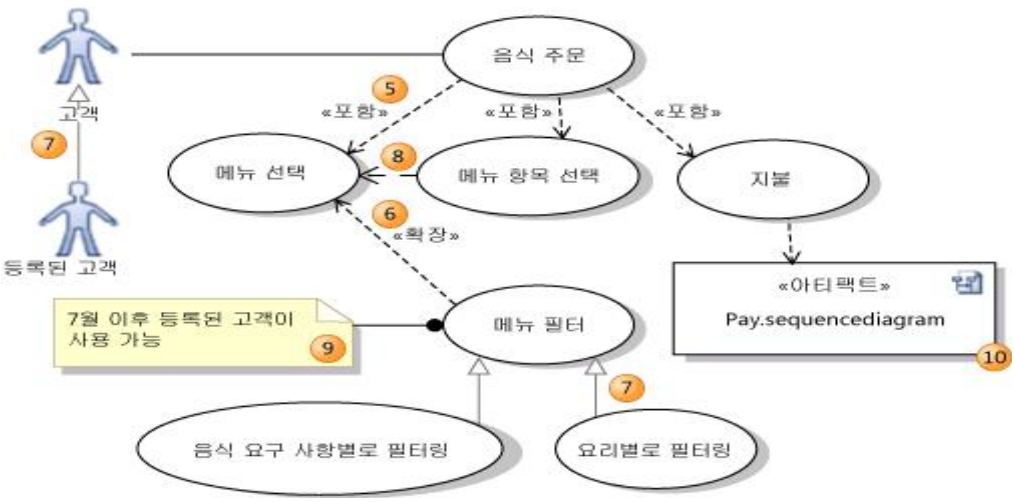
행위자, 사용 사례, 하위 시스템



모양	요소	설명 및 주 속성
1	행위자	응용 프로그램이나 시스템과 상호 작용하는 사용자, 조직 또는 외부 시스템을 나타냅니다.행위자는 형식의 한 유형입니다.

		• 이미지 경로 - 기본 행위자 아이콘 대신 사용해야 할 이미지의 파일 경로입니다.아이콘은 Visual Studio 프로젝트 내의 리소스 파일이어야 합니다.
2	사용 사례	특정 목표를 위해 하나 이상의 행위자가 수행하는 동작을 나타냅니다.사용 사례는 형식의 한 유형입니다. • 제목 - 사용 사례가 나타나는 하위 시스템입니다.
3	연결	행위자가 사용 사례에 참여한다는 것을 나타냅니다.
4	하위 시스템 또는 구성 요소	작업 중인 시스템이나 응용 프로그램 또는 하위 파트입니다. 큰 네트워크에서 응용 프로그램의 단일 클래스까지 무엇이든 가능합니다. 시스템 또는 구성 요소에서 지원하는 사용 사례는 사각형 영역 안에 나타납니다.일부 사용 사례를 사각형 외부에 표시하면 시스템 범위를 명확하게 할 수 있으므로 유용합니다. 기본적으로 사용 사례 다이어그램의 하위 시스템은 구성 요소 다이어그램의 구성 요소와 형식이 같습니다. • 간접 인스턴스화됨 - false이면 실행 시스템에 이 하위 시스템과 직접 대응하는 개체가 하나 이상 있고,true이면 하위 시스템은 구성 파트의 인스턴스화를 통해서만 실행 시스템에 나타나는 디자인 생성자입니다.

-사용 사례 구성



모양	요소	설명
5	포함	포함하는 사용 사례는 포함된 사용 사례를 호출합니다.포함은 사용 사례가 세부 단계로 구분되는 방식을 나타내는 데 사용됩니다.포함된 사용 사례는 화살촉 끝에 있습니다. 사용 사례 다이어그램에는 단계의 순서가 표시되지 않습니다.이러한 세부적인 내용은 동작 다이어그램, 시퀀스 다이어그램 또는 다른 문서를 사용하여 나타낼 수 있습니다.
6	확장	확장하는 사용 사례는 확장된 사용 사례에 목표와 단계를 추가합니다.확장은 특정 조건에서만 동작합니다.확장된 사용 사례는 화살촉 끝에 있습니다. 사용 사례 다이어그램에는 어떤 상황에서 확장이 사용되는지 정확하게 나타나지 않습니다. 이러한 내용은 주석 또는 다른 문서에 기록할 수 있

		습니다.
7	상속	특수화된 요소와 일반화된 요소를 연결합니다.일반화된 요소는 화살촉 끝에 있습니다. 특수화된 사용 사례는 일반화 사례의 목표 및 행위자를 상속하고 좀 더 구체적인 목표와 단계를 추가할 수 있습니다. 특수화된 행위자는 일반화 행위자의 사용 사례, 특성 및 연결을 상속하고 더 추가할 수 있습니다.
8	종속성	소스 디자인이 대상 디자인에 의존한다는 것을 나타냅니다.
9	주석	전반적인 설명을 다이어그램에 추가하는 데 사용됩니다.
10	아티팩트	아티팩트는 다른 다이어그램이나 문서에 대한 링크를 제공합니다.솔루션 탐색기에서 파일을 끌어 아티팩트를 만들 수 있습니다.이 파일은 종속성을 사용하여 다이어그램의 다른 요소에 연결될 수 있습니다.아티팩트는 일반적으로 사용 사례에 대해 자세히 설명하는 시퀀스 다이어그램, OneNote 페이지, Word 문서 또는 PowerPoint 프레젠테이션에 사용 사례를 연결하는 데 사용됩니다.해당 문서는 Visual Studio 솔루션의 항목이거나 SharePoint 사이트 같은 공유 위치의 문서일 수 있습니다. • 하이퍼링크:다이어그램 또는 문서의 URL이나 파일 경로입니다. 아티팩트를 두 번 클릭하면 해당 아티팩트가 연결된 파일 또는 웹 페이지가 열립니다.
11(표시되지 않음)	패키지	사용 사례, 행위자 및 하위 시스템을 패키지 안에 포함할 수 있습니다.패키지 모양은 다이어그램에 나타나지 않지만 다이어그램의 LinkedPackage 속성을 설정할 수 있습니다.이후에 다이어그램에서 만드는 요소는 패키지 안에 배치됩니다.

-레이어 다이어그램

모양	요소	설명
1	레이어	시스템에 있는 물리적 아티팩트의 논리 그룹입니다. 아티팩트가 레이어에 연결되어 있으면 레이어의 왼쪽 위 모퉁이에 숫자가 나타납니다. 레이어에 연결된 아티팩트의 목록을 보려면 레이어를 마우스 오른쪽 단추로 클릭하고 링크 보기를 클릭합니다. • 사용할 수 없는 네임스페이스 종속성 - 이 레이어와 연결된 아티팩트가 지정된 네임스페이스에 종속될 수 없도록 지정합니다. • 사용할 수 없는 네임스페이스 - 이 레이어와 연결된 아티팩트가 지정된 네임스페이스에 속하면 안 되도록 지정합니다. • 필요한 네임스페이스 - 이 레이어와 연결된 아티팩트가 지정된 네임스페이스 중 하나여야 하도록 지정합니다.
2	종속성	한 레이어가 다른 레이어의 기능을 사용할 수 있지만 반대의 경우는 가능하지 않음을 나타냅니다. • 방향 - 종속성의 방향을 지정합니다.
3	양방향 종속성	한 레이어가 다른 레이어의 기능을 사용할 수 있으며 반대의 경우도 가능함을 나타냅니다. • 방향 - 종속성의 방향을 지정합니다.
4	주석	다이어그램 또는 다이어그램의 요소에 전반적인 설명을 추가하는 데 사용합니다.
5	주석 링크	다이어그램의 요소에 주석을 연결하는 데 사용합니다.

-레이어 탐색기

솔루션에 있는 하나 이상의 아티팩트, 예를 들면 프로젝트, 클래스, 네임스페이스, 프로젝트 파일 및 소프트웨어의 기타 파트에 각 레이어를 연결할 수 있습니다.

연결된 아티팩트를 검사하려면

① 레이어 다이어그램에서 하나 이상의 레이어를 마우스 오른쪽 단추로 클릭하고 링크 보기를 클릭합니다.

② 레이어 탐색기가 열리고 선택한 레이어에 연결된 아티팩트가 표시됩니다.레이어 탐색기에는 아티팩트 링크의 각 속성을 보여 주는 열이 있습니다.

레이어 탐색기의 열	설명
범주	클래스, 네임스페이스, 소스 파일 등과 같은 아티팩트의 종류입니다.
레이어	아티팩트에 연결되는 레이어입니다.
유효성 검사 지원	True이면 레이어 유효성 검사를 통해 프로젝트가 이 요소에 대한 종속성을 따르는지 확인할 수 있습니다. False이면 링크가 레이어 유효성 검사에 참여하지 않습니다.
식별자	연결된 아티팩트에 대한 참조입니다.

3) 패키지 및 네임스페이스 정의

-네임스페이스

패키지는 작업을 여러 영역으로 나눌 때 유용합니다. 각 패키지는 여러 패키지에 정의된 이름이 서로 충돌하지 않도록 네임스페이스를 정의합니다. 예를 들어 패키지가 MyPackage인 경우 패키지 내의 클래스는 MyPackage::MyClass와 같은 이름을 갖게 됩니다. 모델도 네임스페이스를 정의하므로 모델에 포함된 모든 요소의 정규화된 이름은 모델 이름으로 시작합니다.

-패키지 만들기

UML 클래스 다이어그램에서 패키지를 만들려면

① UML 클래스 다이어그램을 열거나 새로 만듭니다.

② 패키지 도구를 클릭합니다.

③ 다이어그램에서 아무 곳이나 클릭합니다. 그러면 새 패키지 모양이 나타납니다.

④ 기존 패키지 내부를 클릭하면 한 패키지를 다른 패키지 안에 중첩시킬 수 있습니다.

⑤ 패키지의 새 이름을 입력합니다.

-패키지 내에 모델 요소 만들기

다음과 같은 네 가지 방법을 사용하여 패키지 내에 모델 요소를 배치할 수 있습니다.

① UML 모델 탐색기에서 패키지에 새 요소를 추가합니다.

② UML 클래스 다이어그램에서 패키지에 클래스 및 기타 형식을 추가합니다.

③ 다이어그램에서 만든 새 요소가 특정 패키지 내에 배치되도록 다이어그램의 LinkedPackage 속성을 설정합니다.클래스 다이어그램, 구성 요소 다이어그램 및 사용 사례 다이어그램을 이런 방식으로 패키지에 연결할 수 있습니다.

④ UML 모델 탐색기에서 패키지 내부 또는 외부로 요소를 이동합니다.

-패키지 간의 가져오기 관계

가져오기는 가져온 패키지에 정의된 요소, 즉 관계의 화살표 끝에 있는 요소가 가져오는 패키지에서도 효과적으로 정의된다는 의미입니다. 표시 유형이 패키지로 정의된 요소는 가져오는 패키지에서도 볼 수 있습니다. 가져오기 관계에 루프를 만들지 마십시오.

다른 패키지에서 한 패키지의 요소를 참조하려면 요소의 정규화된 이름을 사용해야 합니다.

예를 들어 SalesCommon 패키지에서 CustomerAddress 형식을 정의한다고 가정해 봅니다. 또한 RestaurantSales라는 다른 패키지에서 Customer Address 형식의 특성을 포함하는 MealOrder 형식을 정의하려고 합니다. 이때 다음과 같은 두 옵션을 사용할 수 있습니다.

①정규화된 이름 SalesCommon::CustomerAddress를 사용하여 특성의 형식을 지정합니다.CustomerAddress의 표시 유형 속성이 공용으로 설정된 경우에만 이렇게 해야 합니다.

②RestaurantSales 패키지에서 SalesCommon 패키지로 가져오기 관계를 만듭니다. 그러면 정규화된 이름을 사용하지 않고 CustomerAddress를 사용할 수 있습니다.

-패키지 속성

속성	기본값	설명
이름	(새 이름)	패키지 이름입니다. 다이어그램 또는 속성 창에서 변경할 수 있습니다.
정규화된 이름	Container :: package name	이 패키지를 포함하는 모델 또는 패키지의 이름이 앞에 나오는 전체 이름입니다.
프로필	(비어 있음)	이 패키지에 연결된 프로필 목록입니다. 이러한 프로필은 패키지 내의 요소에 적용할 수 있는 스테레오타입을 제공합니다.스테레오타입을 사용하여 모델 사용자 지정을 참조하십시오.
표시 유형	공용	부모 패키지 외부에서 패키지의 표시 유형입니다.
작업 항목	(비어 있음)	연결된 작업 항목의 목록입니다.
정의 위치	(이름)	패키지의 세부 정보가 저장되는 파일 이름입니다. 파일은 ModelDefinition 프로젝트 폴더에 있습니다. 이 정보는 소스 제어 용도로 사용될 경우 도움이 됩니다.
설명	(비어 있음)	패키지에 대한 설명입니다.
스테레오타입	(비어 있음)	이 패키지에 적용되는 스테레오타입입니다. 사용할 수 있는 스테레오타입 목록은 이 패키지 및 이 패키지를 포함하는 패키지에 대해 선택한 프로필에 따라 결정됩니다.

-패키지 저장 방식

새 패키지를 만들면 *ModelDefinition* 프로젝트 폴더에 새 *.uml* 파일이 만들어집니다. 마찬가지로 패키지인 루트 모델도 *.uml* 파일에 저장됩니다. 또한 각 다이어그램은 두 개의 파일, 즉 다이어그램의 모양을 나타내는 파일과 모양의 위치를 기록하는 *.layout* 파일에 저장됩니다.

4) 프로필 및 스테레오타입을 사용하여 모델 사용자 지정

요소의 속성 목록을 변경할 수 있는 스테레오타입을 모델 요소에 적용할 수 있습니다. 스테레오타입은 프로필이라는 컬렉션 내에 정의됩니다. 스테레오타입을 사용하려면 프로필에 패키지를 연결합니다. 이렇게 하면 프로필에 정의된 스테레오타입을 패키지의 요소에 적용할 수 있습니다. 일부 프로필은 사용자 지정 프로필을 정의할 수 있습니다. 스테레오타입은 요소의 속성 목록에 설정할 수 있습니다. 다음 예제와 같이 다이어그램에 나타나는 주요 모양의 경우 적용된 스테레오타입도 모양에 함께 나타납니다.

5) 모델 요소에 작업 항목 연결

작업 항목에 모델 요소를 연결할 수 있습니다. 이렇게 하면 작업, 테스트 사례, 버그, 요구사항, 문제점 또는 모델의 특정 파트와 연결된 다른 작업 유형 등을 추적할 수 있습니다. 또한 연결된 작업 항목에 문서를 첨부하여 모델 요소에 문서를 연결할 수도 있습니다. 예를 들어 다음과 같은 링크를 만들 수 있습니다.

- 스토리가 작업 시퀀스로 구현되는 방식을 설명하는 동작 다이어그램에 사용자 스토리 작업 항목을 연결합니다.

- 올바로 구현되었는지 확인하는 테스트 사례 작업 항목에 사용 사례 다이어그램의 사용 사례를 연결합니다.

- 특성 구현의 오류에 대한 버그 작업 항목에 UML 클래스 다이어그램에 있는 클래스의 특성을 연결합니다.

- 개발을 추적하는 작업(Task) 작업 항목에 구성 요소 다이어그램의 구성 요소를 연결합니다. 이러한 작업은 대개 더 작은 여러 작업의 부모입니다.

다이어그램 또는 UML 모델 탐색기에서 선택할 수 있는 요소에 작업 항목을 연결할 수 있습니다. 예를 들면 다음과 같은 요소입니다.

- UML 클래스, 수명선, 사용 사례, 하위 시스템, 동작, 개체 노드, 구성 요소, 인터페이스 등 UML 모델의 모든 요소

- 연결, 일반화, 종속성, 흐름, 메시지 등 UML 모델의 모든 관계

- 클래스의 특성과 작업, 수명선의 실행 발생, 동작의 입력/출력 핀, 구성 요소의 파트와 포트 등 요소의 파트

- 레이어 및 레이어 종속성

- 주석 및 주석 링크
- 다이어그램. 다이어그램을 선택하려면 다이어그램의 빈 부분을 클릭합니다.

(1)Team Foundation Server에 연결하려면

① 보기 메뉴에서 팀 탐색기를 클릭합니다.

② 올바른 서버 및 프로젝트가 표시되지 않으면 기존 팀 프로젝트 추가를 클릭하고 올바른 서버와 프로젝트를 선택합니다.

③ 팀 탐색기에서 작업 항목을 만들고 연결하거나 볼 프로젝트를 클릭합니다.

④ 프로젝트가 팀 탐색기에 강조 표시된 상태로 나타납니다.

(2)모델 요소에 연결된 작업 항목을 열려면

① 팀 탐색기에서 모델 요소가 연결된 Team Foundation Server에 연결되어 있는지 확인합니다.

② 다이어그램 또는 UML 모델 탐색기에서 모델 요소를 마우스 오른쪽 단추로 클릭하고 작업 항목 보기를 클릭합니다.

③ 작업 항목 목록 창이 열립니다. 여기에서는 연결된 작업 항목의 목록을 보여 줍니다.

6) 모델과 다이어그램을 저장

Visual Studio Ultimate을 사용하지 않는 사람을 포함하여 다른 사용자와 공유할 수 있도록 모델과 다이어그램을 저장합니다.

3-2. 관련 작업

관련 작업
현재 시스템에서 파트 간의 동작, 관계 및 조직을 시각화하고 분석합니다.
모델을 사용하여 사용자 요구를 명확하게 하고 전달합니다.
모델을 사용하여 시스템의 전체 구조 및 동작을 기술하고 사용자 요구를 충족하도록 합니다.
소프트웨어가 사용자 요구 및 시스템의 전반적인 아키텍처와 일관성 있게 유지되도록 합니다.
모델을 사용하여 개발 중에 시스템을 쉽게 이해하고 변경합니다.

1) 기존 코드 탐색

코드에서 기존 관계와 패턴을 검사하기 위해 종속성 그래프를 생성할 수 있습니다.

작업
코드 관계를 탐색합니다. 종속성 그래프나 다른 그래프를 생성하여 코드의 관계를 확인합니다.

기존 코드를 찾습니다. 아키텍처 탐색기를 사용하여 Visual Studio 솔루션의 소스 코드나 컴파일된 코드를 찾고 탐색할 수 있습니다.
코드 상호 작용을 탐색합니다. 코드에서 시퀀스 다이어그램을 생성하여 코드의 상호 작용을 파악합니다.
코드 구조를 탐색합니다. 클래스 다이어그램을 만들어 프로젝트의 클래스 구조를 검사합니다.
시스템의 상위 수준 디자인을 나타내고 이 디자인을 기준으로 코드의 유효성을 검사합니다. 레이어 다이어그램을 만들어 시스템의 상위 수준 디자인과 해당 종속성을 나타냅니다.이 디자인을 기준으로 코드의 유효성을 검사하여 코드가 디자인과 일관성을 유지하는지 확인합니다.
사용자 요구 사항 및 시스템 디자인을 전달합니다. 동작, 구성 요소, 클래스, 시퀀스 및 사용 사례 다이어그램과 같은 UML 다이어그램을 그려 소프트웨어 시스템의 아키텍처와 사용자 요구 사항을 모델링합니다.

2) 사용자 요구 사항 모델링

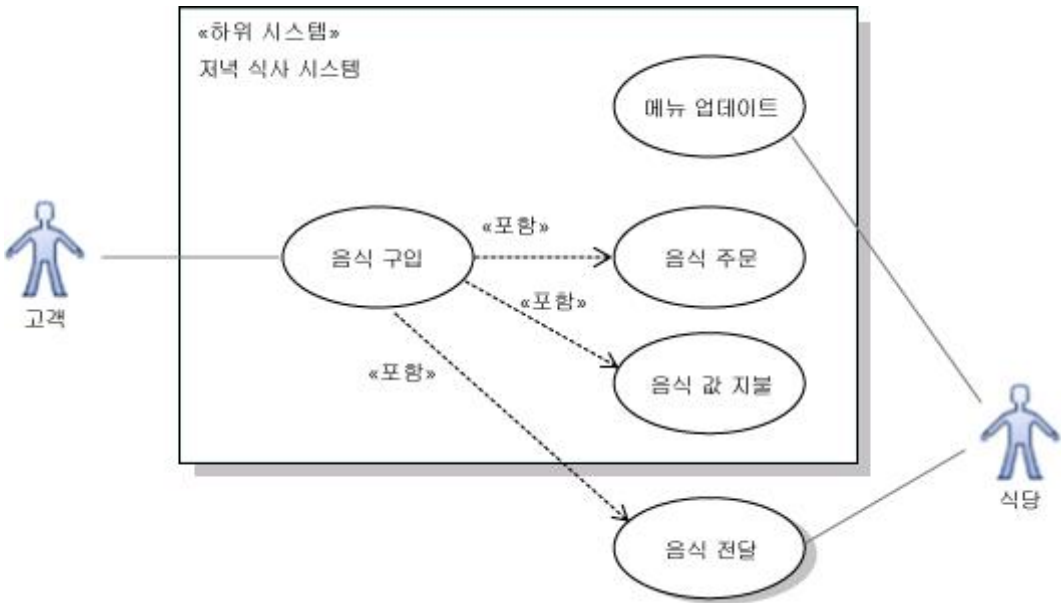
사용자 요구 사항을 나타내는 다양한 종류의 뷰를 만들 수 있습니다.

다이어그램 또는 문서	요구 사항 모델에서 기술하는 내용
사용 사례 다이어그램	시스템 사용자 및 시스템 사용자가 시스템에서 수행하는 작업
개념 클래스 다이어그램	요구 사항을 기술하는 데 사용되는 형식의 용어 및 시스템 인터페이스에서 볼 수 있는 형식
동작 다이어그램	사용자가 수행하는 동작과 시스템 또는 시스템 요소 사이의 정보 및 워크플로
시퀀스 다이어그램	사용자와 시스템 또는 시스템 요소 사이의 상호 작용 시퀀스. 동작 다이어그램을 대체할 수 있는 뷰입니다.
추가 문서 또는 작업 항목	성능, 보안, 유용성 및 안정성 기준
추가 문서 또는 작업 항목	특정 사용 사례에 제한되지 않는 제약 조건 및 규칙

-시스템이 사용되는 방식 기술

시스템 사용자 및 시스템 용도를 기술하려면 사용 사례 다이어그램을 만듭니다. 예를 들어 온라인 식사 판매 시스템은 고객이 메뉴에서 품목을 선택할 수 있고 공급 식당에서 메뉴를 업데이트할 수 있도록 해야 합니다.

사용 사례가 어떻게 구성되는지 하위 사례를 표시할 수도 있습니다. 예를 들어 식사를 주문하는 것은 구매의 한 부분이며 지불과 배달을 포함합니다.



또한 개발 중인 시스템의 범위에 포함되는 사용 사례를 표시할 수 있습니다. 사용 사례 다이어그램은 해당 사례에 대한 요약만 제공합니다. 자세한 설명을 제공하려면 다이어그램의 사용 사례를 별도의 문서 및 다른 다이어그램으로 연결하는 링크를 만들어야 합니다. 사용 사례 다이어그램을 그리면 팀에서 다음과 같이 할 수 있습니다.

① 상세한 구현 내용에 신경 쓰지 않고 사용자가 시스템으로 수행할 작업에만 집중할 수 있습니다.

② 현재 시스템 또는 특정 릴리스의 시스템에 포함할 범위를 논의할 수 있습니다.

-요구 사항을 기술하는 데 사용되는 용어 정의

UML 클래스 다이어그램을 사용하면 다음과 같은 용도로 사용되는 비즈니스 개념의 일관성 있는 용어 모음을 개발할 수 있습니다.

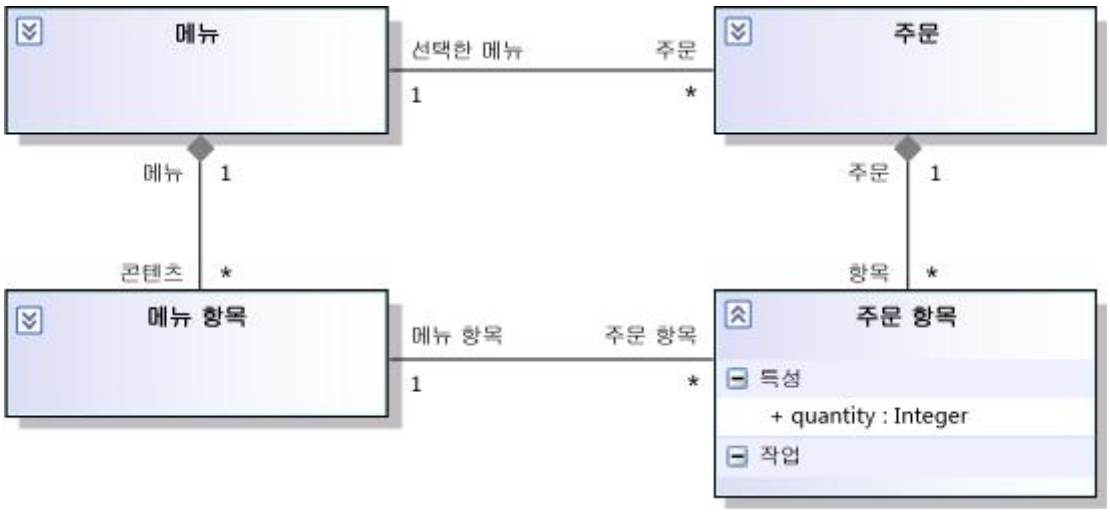
① 시스템이 동작하는 비즈니스를 논의하는 사용자에게 유용합니다.

② 예를 들어 사용 사례, 비즈니스 규칙 및 사용자 스토리 설명에서 사용자의 요구를 기술하는 데 유용합니다.

③ 시스템 API에서 또는 사용자 인터페이스를 통해 교환되는 정보의 유형을 나타내는 데 유용합니다.

④ 시스템 또는 수용 테스트 설명을 기술하는 데 유용합니다.

이 용도로 사용되는 UML 클래스 다이어그램의 콘텐츠를 개념 클래스 다이어그램이라고 합니다. 도메인 모델 또는 분석 클래스 모델이라고도 합니다. 개념 클래스 다이어그램에서는 시스템의 내부 디자인 정보는 전혀 표시하지 않고 요구 사항 설명에 필요한 클래스만 나타냅니다. 즉, 이 다이어그램에는 시스템의 내부 디자인에 대한 정보가 나타나지 않습니다. 일반적으로 개념 클래스에는 작업 또는 인터페이스를 나타내지 않습니다. 예를 들어 Dinner Now 시스템에 대해 다음과 같은 개념 클래스를 그릴 수 있습니다.



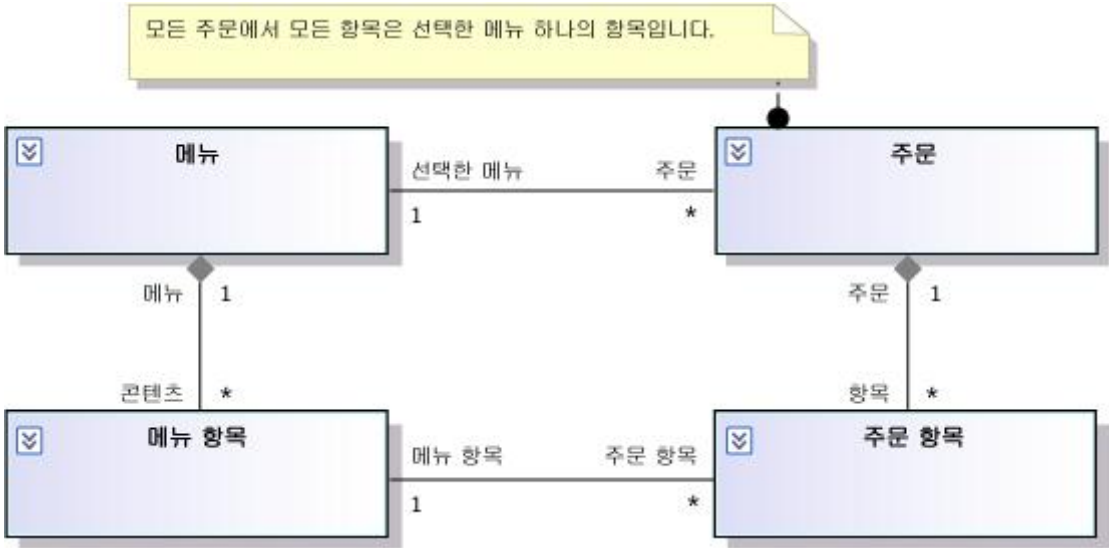
개념 클래스 다이어그램에서는 요구 사항 모델 전체에서 사용하는 용어 모음을 제공합니다. 예를 들어 Order a Meal 사용 사례에 대한 자세한 설명에서 다음과 같이 작성할 수 있습니다. 고객은 Order를 만들기 위해 Menu를 선택한 다음, Menu에서 Menu Items를 선택하여 Order에서 Order Items를 만듭니다. 설명에 사용된 용어가 모델에서 클래스 이름이 되는 것에 주목하십시오. 이 다이어그램은 클래스 관계에서 모호성을 제거합니다.

① 사용자의 요구를 논의할 때 사용되는 기본 용어를 정의하고 표준화할 수 있습니다.

② 이러한 용어의 관계를 명확하게 나타낼 수 있습니다.

-비즈니스 규칙 표시

비즈니스 규칙은 특정 사용 사례와 연결되지 않고 시스템 전체에서 관찰되어야 하는 요구 사항입니다. 대부분의 비즈니스 규칙은 개념 클래스 간 관계에 대한 제약 조건입니다. 이러한 정적비즈니스 규칙을 개념 클래스 다이어그램의 관련 클래스와 연결된 주석으로 작성할 수 있습니다. 예를 들면 다음과 같습니다.



-서비스 품질 요구 사항 기술

서비스 품질 요구 사항은 몇 가지 범주로 구성됩니다.

① 성능

② 보안

③ 유용성

④ 안정성

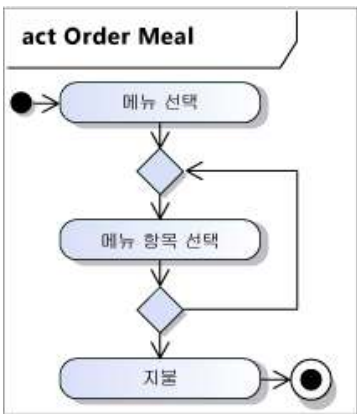
⑤ 견고성

특정 사용 사례를 기술할 때 이러한 요구 사항 중 일부를 포함할 수 있습니다. 다른 요구 사항은 사용 사례에 제한되지 않으며 별도의 문서에서 가장 효과적으로 작성됩니다. 가능한 경우 요구 사항 모델에 정의된 용어를 따르면 유용합니다.

-사용자와 시스템 사이의 워크플로 표시

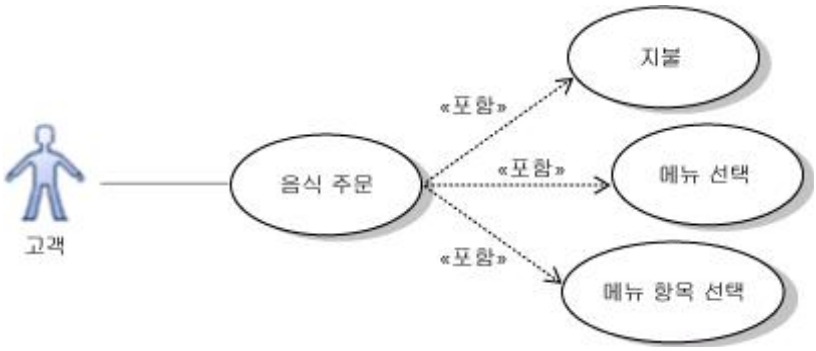
동작 다이어그램을 사용하면 각 사용 사례 사이의 워크플로를 나타낼 수 있습니다.대부분의 경우 시스템 내부와 외부에서 사용자가 수행할 주요 작업을 보여 주는 동작 다이어그램을 그려 요구 사항 모델을 시작하면 유용합니다.

예를 들면 다음과 같습니다.



같은 정보를 서로 다른 뷰로 나타내려는 경우 사용 사례 다이어그램과 동작 다이어그램을 그릴 수 있습니다.

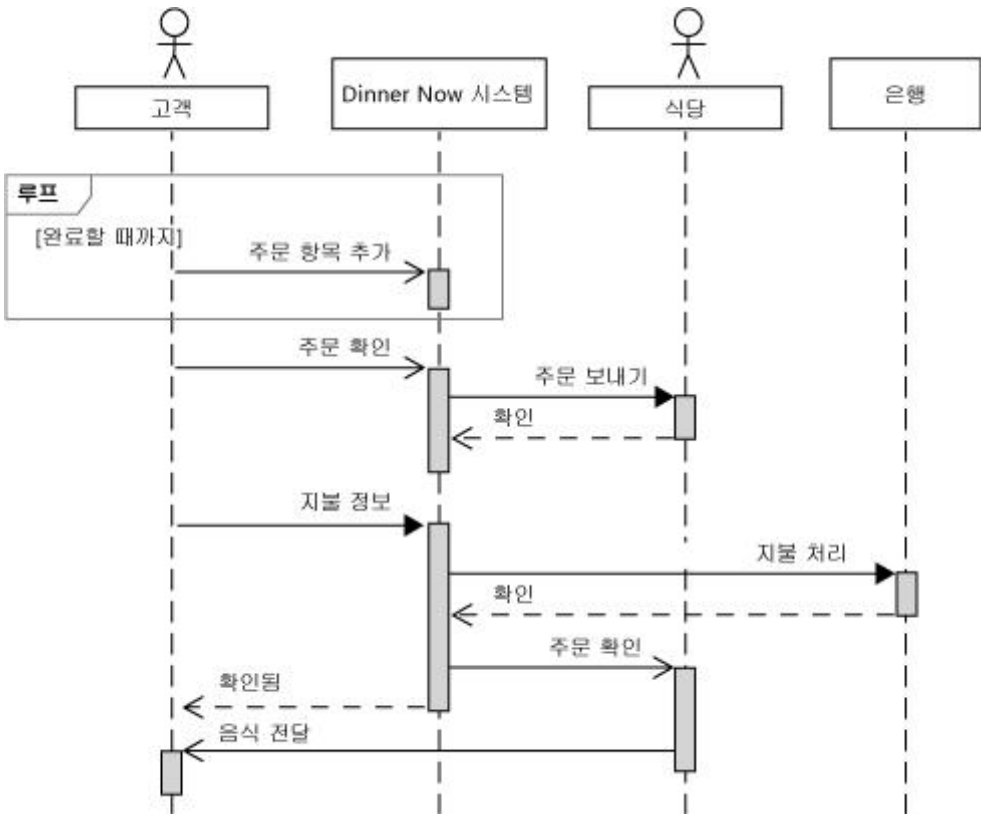
예를 들면 다음과 같습니다.



-사용자와 시스템 사이의 상호 작용 표시

시퀀스 다이어그램을 사용하면 시스템과 외부 행위자 간 또는 각 시스템 구성 요소 간의 메시지 교환을 나타낼 수 있습니다. 이렇게 하면 사용 사례에서 상호 작용 시퀀스를 명확히 나타내는 단계 뷰를 제공할 수 있습니다. 시퀀스 다이어그램은 사용 사례에 상호 작용 대상이 여러 개 있는 경우와 시스템에 API가 있는 경우 특히 유용합니다.

예를 들면 다음과 같습니다.



시퀀스 다이어그램은 생성 중인 시스템으로 들어오는 메시지를 쉽게 볼 수 있다는 장점이

있습니다.

-모델을 사용하여 불일치 줄이기

일반적으로 모델을 만들면 사용자의 요구 사항에서 불일치 및 모호성 문제가 상당히 줄어듭니다. 각 다이어그램에서 제공하는 뷰의 관계를 조사하면 사용자가 다루는 주요 개념을 쉽게 이해할 수 있으며 사용자가 시스템에서 무엇을 필요로 하는지 스스로 이해하도록 도움을 줄 수 있습니다.

3) 소프트웨어 시스템의 아키텍처 모델링

모델을 사용하면 디자인 전체에서 사용되는 패턴을 기술할 수도 있습니다. 모델의 목적은 자연어를 사용하여 기술할 때 발생하는 모호성을 줄이고, 동료와 함께 디자인을 시각화하고 대체 디자인을 논의하는 데 도움을 주는 것입니다. 모델은 다른 문서 또는 토론과 함께 사용되어야 합니다. 모델 자체는 아키텍처의 완전한 사양을 나타내지 않습니다.

시스템 아키텍처는 다음과 같은 두 영역으로 나눌 수 있습니다.

① 고급 디자인. 여기에서는 주요 구성 요소를 나타내고 이러한 구성 요소가 각 요구 사항을 수행하기 위해 다른 요소와 상호 작용하는 방식을 기술합니다.

② 구성 요소의 디자인 전체에 사용되는 디자인 패턴 및 규칙. 패턴은 프로그래밍 목표를 달성하기 위한 특정한 방법을 기술합니다.

-고급 디자인

고급 디자인은 시스템의 주요 구성 요소를 나타내고 이러한 구성 요소가 디자인 목표를 달성하기 위해 다른 요소와 상호 작용하는 방식을 기술합니다.

① 요구 사항 이해 : 디자인은 사용자의 요구를 명확하게 이해하는 것에서 시작됩니다.

② 아키텍처 패턴 : 시스템의 아키텍처 요소 및 핵심 기술에 대한 선택 항목입니다.

③ 구성 요소 및 구성 요소 인터페이스 : 시스템의 주요 파트를 보여 주는 구성 요소 다이어그램을 그리고, 시스템의 파트가 서로 상호 작용할 때 사용하는 인터페이스를 나타낼 수 있습니다.각 구성 요소의 인터페이스에는 시퀀스 다이어그램에서 식별한 모든 메시지가 포함됩니다.

④ 구성 요소 간 상호 작용 : 각 사용 사례, 이벤트 또는 들어오는 메시지에 대해 필요한 응답을 얻기 위해 시스템의 주요 구성 요소가 상호 작용하는 방식을 보여 주는 시퀀스 다이어그램을 그릴 수 있습니다.

⑤ 구성 요소 및 인터페이스의 데이터 모델 : 구성 요소 간에 전달되고 구성 요소 내에 저장되는 정보를 기술하기 위해 클래스 다이어그램을 그릴 수 있습니다.

-요구 사항 이해

완전한 응용 프로그램의 고급 디자인은 요구 사항 모델 또는 기타 사용자 요구에 대한 기

술과 함께 가장 효과적으로 개발됩니다. 요구 사항 모델은 다음과 같은 중요한 정보를 제공합니다.

① 제공된 인터페이스 : 제공된 인터페이스는 시스템이나 구성 요소에서 사용자에게 제공해야 하는 서비스 또는 작업을 나열합니다. 이때 사용자는 사람일 수도 있고 다른 소프트웨어 구성 요소일 수도 있습니다.

② 필요한 인터페이스 : 필요한 인터페이스는 시스템이나 구성 요소에서 사용할 수 있는 서비스 또는 작업을 나열합니다. 어떤 경우에는 이러한 모든 서비스를 시스템의 일부로 디자인할 수 있습니다.다른 경우, 특히 대부분의 구성에서 다른 구성 요소와 결합할 수 있는 구성 요소를 디자인하는 경우에는 외부적 문제에 의해 필요한 인터페이스가 설정됩니다.

③ 서비스 품질 요구 사항 : 성능, 보안, 안정성 및 시스템에서 충족해야 할 기타 목표와 제약 조건입니다. 요구 사항 모델은 시스템 사용자의 관점에서 작성됩니다. 이때 사용자는 사람일 수도 있고 다른 소프트웨어 구성 요소일 수도 있습니다. 사용자는 시스템의 내부 작업에 대해 아무 것도 모릅니다. 이와 달리 아키텍처 모델의 목표는 내부 작업을 기술하고 이러한 작업이 사용자의 요구를 충족하는 방식을 보여 주는 데 있습니다.

요구 사항 모델과 아키텍처 모델을 별개로 유지하면 사용자와 요구 사항에대해 더 쉽게 논의할 수 있으므로 유용합니다. 또한 요구 사항을 변경하지 않으면서 디자인을 리팩터링하고 대체 아키텍처를 고려할 수 있습니다. 다음과 같은 두 가지 방법으로 요구 사항 모델과 아키텍처 모델을 분리할 수 있습니다.

④ 두 모델을 각각 같은 솔루션의 다른 프로젝트에 둡니다. 이렇게 하면 UML 모델 탐색기에 별도의 모델로 나타나고, 여러 명의 팀 멤버가 동시에 모델에서 작업할 수 있습니다.두 모델 간에 제한된 종류의 추적을 만들 수 있습니다.

⑤ 두 모델을 각각 같은 UML 모델의 다른 패키지에 둡니다. 이렇게 하면 모델 간의 종속성을 쉽게 추적할 수 있지만 두 명 이상의 사용자가 동시에 모델에서 작업할 수 없습니다. 또한 매우 큰 모델을 Visual Studio로 로드할 때 시간이 오래 걸립니다. 따라서 이 방법은 큰 프로젝트에 적합하지 않습니다.

요구 사항 모델 또는 아키텍처 모델에 나타내야 할 세부 정보의 양은 프로젝트의 규모와 팀의 크기 및 배포에 따라 다릅니다. 짧은 프로젝트의 소규모 팀은 비즈니스 개념 및 일부 디자인 패턴의 클래스 다이어그램을 스케치하는 정도로 충분하지만 여러 나라에 배포되는 큰 프로젝트는 매우 상세하게 기술해야 합니다.

-아키텍처 패턴

개발 초기 단계에서는 디자인이 의존하는 주요 기술 및 요소를 선택해야 합니다.

① 데이터베이스와 파일 시스템 사이의 선택, 네트워크 응용 프로그램과 웹 클라이언트 사이의 선택 등과 같은 기본 기술 선택

② Windows Workflow Foundation 또는 ADO.NET Entity Framework 사이의 선택과 같은 프레임워크 선택

③ 엔터프라이즈 서비스 버스 또는 지점 간 채널 사이의 선택과 같은 통합 방식 선택

이러한 선택은 대개 규모 및 유연성과 같은 서비스 품질 요구 사항에 의해 결정되며 상세 요구 사항이 알려지기 전에 지정할 수 있습니다.큰 시스템의 경우 하드웨어와 소프트웨어의 구성은 서로 긴밀하게 관련되어 있습니다. 이러한 선택은 아키텍처 모델을 사용하고 해석하는 방식에 영향을 줍니다.

-구성 요소 및 구성 요소 인터페이스

① 구성 요소 다이어그램을 만들어 시스템의 주요 파트를 나타냅니다.

② 구성 요소와 해당 인터페이스 간의 종속성을 그려 시스템의 구조를 나타냅니다.

③ 구성 요소의 인터페이스를 사용하여 각 구성 요소에서 제공하거나 필요로 하는 서비스를 나타냅니다.

④ 규모가 큰 디자인의 경우 별도의 다이어그램을 그려 각 구성 요소를 더 작은 파트로 분해할 수 있습니다.

-구성 요소

아키텍처 모델의 중심이 되는 뷰는 시스템의 주요 파트 및 각 파트가 서로 의존하는 방식을 보여 주는 구성 요소 다이어그램입니다. 큰 시스템의 구성 요소 다이어그램에는 대개 다음과 같은 구성 요소가 포함됩니다.

① 프레젠테이션 : 사용자에 대한 액세스를 제공하는 구성 요소이며 일반적으로 웹 브라우저에서 실행됩니다.

② 웹 서비스 구성 요소 : 클라이언트와 서버 간의 연결을 제공합니다.

③ 사용 사례 컨트롤러 : 각 시나리오의 단계를 통해 사용자를 안내합니다.

④ 비즈니스 핵심 : 요구 사항 모델의 클래스를 기반으로 하는 클래스를 포함하고, 핵심 작업을 구현하며, 비즈니스 제약 조건을 적용합니다.

⑤ 데이터베이스 : 비즈니스 개체를 저장합니다.

⑥ 로깅 및 오류 처리 구성 요소

-구성 요소 간의 종속성

구성 요소 자체뿐만 아니라 구성 요소 간의 종속성도 나타낼 수 있습니다. 두 구성 요소 간의 종속성 화살표는 한 구성 요소의 디자인 변경이 다른 구성 요소의 디자인에 영향을 줄 수 있음을 나타냅니다. 이는 대개 한 구성 요소가 다른 구성 요소에서 제공하는 서비스나 기능을 직접 또는 간접적으로 사용하기 때문입니다. 올바른 구조의 아키텍처에는 종속성이 명확하게 배열되어 다음과 같은 조건을 만족합니다.

① 종속성 그래프에 루프가 없습니다.

② 모든 종속성이 한 레이어의 구성 요소에서 다른 레이어의 구성 요소로 움직이도록 구성 요소를 레이어로 정렬할 수 있습니다.두 레이어 간의 모든 종속성은 같은 방향으로 이동합니다.

종속성 관리는 유지 관리가 가능한 소프트웨어를 생성하는 데 있어서 가장 중요합니다.

-인터페이스

구성 요소에 인터페이스를 배치하면 각 구성 요소에서 제공하는 주요 작업 그룹을 구분하고 이름을 지정할 수 있습니다. 예를 들어 웹 기반 판매 시스템의 구성 요소에는 고객이 상품을 구매하는 인터페이스, 판매업체에서 카탈로그를 업데이트하는 인터페이스, 시스템을 관리하는 세 번째 인터페이스가 있을 수 있습니다. 구성 요소에 포함할 수 있는 제공된 인터페이스와 필요한 인터페이스의 수에는 제한이 없습니다. 제공된 인터페이스와 필요한 인터페이스를 둘 다 정의하면 디자인의 나머지 부분에서 구성 요소가 완벽하게 분리되므로 다음과 같은 기법을 사용할 수 있습니다.

① 관련 구성 요소가 테스트 도구에 의해 시뮬레이션되도록 구성 요소를 테스트 도구에 배치합니다.

② 구성 요소를 다른 구성 요소와 독립적으로 개발합니다.

③ 인터페이스를 다른 구성 요소에 결합하여 다른 컨텍스트에서 구성 요소를 다시 사용합니다.

-구성 요소를 파트로 분해

각 구성 요소 내에서 하위 구성 요소를 파트로 나타낼 수 있습니다. 다음과 같은 경우에는 파트를 사용합니다.

① 부모 구성 요소의 디자인에서 항상 파트의 구성 요소 형식을 사용해야 하는 경우. 따라서 파트의 디자인이 부모 구성 요소의 디자인에 반드시 필요합니다.

② 부모 구성 요소에 구체적인 요소가 없는 경우. 예를 들어 뷰와 사용자 상호 작용을 처리하는 실제 구성 요소의 컬렉션을 나타내는 개념적 구성 요소, 즉 프레젠테이션 레이어가 있을 수 있습니다.

다음과 같은 경우에는 필요한 인터페이스를 통해 액세스되는 별도의 구성 요소를 사용합니다.

③ 요청하는 구성 요소가 인터페이스를 통해 런타임에 다른 제공하는 구성 요소에 결합될 수 있는 경우

④ 한 공급자를 다른 공급자로 쉽게 바꿀 수 있는 디자인인 경우

대개 필요한 인터페이스를 사용하는 것이 파트를 사용하는 것보다 좋습니다. 이 방법은 디자인하는 데 시간이 더 걸리지만 결과 시스템의 유연성이 뛰어납니다. 또한 구성 요소를 별도로 테스트하기도 쉽습니다. 이를 통해 개발 계획에서 결합을 줄일 수 있습니다.

-구성 요소 간 상호 작용

① 시스템의 사용 사례를 식별합니다.

② 각 사용 사례에 대해 시스템의 구성 요소가 다른 구성 요소 및 사용자와 공동으로 작

업하여 필요한 결과를 달성하는 방식을 보여 주는 다이어그램을 하나 이상 그립니다. 일반적으로 시퀀스 다이어그램 또는 동작 다이어그램을 그릴 수 있습니다.

③ 인터페이스를 사용하여 각 구성 요소가 수신하는 메시지를 지정합니다.

④ 인터페이스에서 작업의 효과를 기술합니다.

⑤ 각 구성 요소에 대해 절차를 반복하여 파트의 상호 작용 방식을 보여 줍니다.

예를 들어 웹 기반 판매 시스템의 요구 사항 모델에서 고객의 구매를 사용 사례로 정의할 수 있습니다.이 경우 프레젠테이션 레이어의 구성 요소와 고객의 상호 작용을 나타내고, 웨어하우스 및 회계 구성 요소와 고객의 상호 작용을 나타내는 시퀀스 다이어그램을 만들 수 있습니다.

-시작 이벤트 식별

대부분의 소프트웨어 시스템에서 수행되는 작업은 다양한 입력 또는 이벤트에 제공하는 응답에 의해 편리하게 나눌 수 있습니다.시작 이벤트는 다음 이벤트 중 하나일 수 있습니다.

① 사용 사례의 첫 번째 동작 : 이 동작은 요구 사항 모델에서 사용 사례의 단계 또는 동작 다이어그램의 동작으로 나타날 수 있습니다.

② 프로그래밍 인터페이스의 메시지 : 개발하는 시스템이 규모가 큰 시스템의 구성 요소인 경우에는 구성 요소 인터페이스 중 하나에서 작업으로 기술해야 합니다.

③ 시스템에서 모니터링되는 특정 조건 또는 정기적인 이벤트(예: 시간).

-디자인 패턴

디자인 패턴은 소프트웨어의 특정 요소, 특히 시스템의 여러 파트에서 되풀이되는 요소를 디자인하는 방법을 요약한 것입니다. 프로젝트 전체에 일정한 방법을 사용하면 디자인 비용을 줄이고 사용자 인터페이스의 일관성을 유지할 수 있으며 코드를 이해하고 변경하는 데 필요한 노력을 줄일 수 있습니다.

소프트웨어 아키텍처 작업의 일부는 디자인 전체에 사용해야 할 패턴을 결정하는 것입니다. 프로젝트가 진행됨에 따라 새 패턴 및 기존 패턴의 업데이트가 검색되므로 대개 이 작업은 계속됩니다. 초기 단계에서 주요 디자인 패턴을 각각 시험하도록 개발 계획을 구성하는 것이 좋습니다.

디자인 패턴은 문서에 기술되며 일반적으로 다음과 같은 요소를 포함합니다.

① 이름

② 패턴을 적용할 수 있는 컨텍스트에 대한 설명. 예를 들면 개발자가 패턴을 적용할 때 고려해야 할 기준을 지정합니다.

③ 패턴으로 해결하는 문제에 대한 간단한 설명

④ 주요 파트 및 파트 관계 모델. 클래스 또는 구성 요소와 인터페이스 및 이들 간의 종속성과 연결 등이 될 수 있습니다. 요소는 대개 다음과 같은 두 범주에 포함됩니다.

⑤ 개발자가 패턴이 사용되는 코드의 모든 부분에서 복제해야 할 요소. 템플릿 형식을

사용하여 이러한 요소를 기술할 수 있습니다.

⑥ 개발자가 사용해야 할 프레임워크 클래스를 기술하는 요소

⑦ 시퀀스 다이어그램 또는 동작 다이어그램을 사용하는 파트 간 상호 작용 모델

⑧ 명명 규칙

⑨ 패턴으로 문제를 해결하는 방식에 대한 설명

⑩ 개발자가 채택할 수 있는 패턴 변형에 대한 설명

4) 개발하는 동안 시스템 유효성 검사

Visual Studio Ultimate를 사용하면 소프트웨어를 사용자 요구 사항 및 시스템 아키텍처에 맞게 일관성을 유지할 수 있습니다.

작업
모델의 일관성을 확인합니다. 프로젝트에서 모델을 사용하고 해석하는 방식에 따라 일부 요소 조합을 허용하지 않는 것이 유용할 수 있습니다.예를 들어 UML 클래스가 항상 .NET 규격 이름을 갖도록 제한할 수 있습니다.이와 같은 제약 조건은 Visual Studio 확장에서 정의할 수 있습니다.
소프트웨어가 사용자의 요구 사항에 맞는지 확인합니다. 요구 사항 및 아키텍처 모델을 사용하여 시스템과 시스템 구성 요소의 테스트를 구성할 수 있습니다.이렇게 하면 사용자 및 다른 관련자에게 중요한 요구 사항을 테스트할 수 있으며 요구 사항이 변경될 경우 테스트를 신속하게 업데이트할 수 있습니다.
소프트웨어가 시스템의 계획된 디자인과 일관성을 유지하는지 확인합니다. 레이어 다이어그램은 응용 프로그램 구성 요소 간의 계획된 종속성을 기술합니다. 개발하는 동안 코드의 실제 종속성이 계획된 디자인을 따르는지 확인할 수 있습니다.

5) 개발 프로세스에서 모델 사용

모델을 사용하면 시스템 동작 환경 시각화, 사용자 요구 명시화, 시스템 아키텍처 정의, 코드 분석, 요구 사항을 충족하는 코드 작성 등의 작업을 수행하는 데 도움이 됩니다.

-모델을 사용하는 방법

① 모델링 다이어그램을 그리면 요구 사항, 아키텍처 및 고급 디자인과 관련된 개념을 명확하게 할 수 있습니다.

② 모델을 사용하면 요구 사항의 불일치를 나타낼 수 있습니다.

③ 모델을 사용하면 중요한 개념을 전달할 때 자연어를 사용하는 경우보다 모호성을 줄일 수 있습니다.

④ 경우에 따라 모델을 사용하여 코드 또는 기타 아티팩트(예: 데이터베이스 스키마 또는 문서)를 생성할 수 있습니다.예를 들어 Visual Studio Ultimate의 모델링 구성 요소는 모델에서 생성됩니다.

Extreme Agile에서 High Ceremony까지 다양한 프로세스에서 모델을 사용할 수 있습니다.

-남은 작업 예상

요구 사항 모델을 사용하면 각 반복의 크기를 기준으로 프로젝트의 전체 크기를 예상할 수 있습니다. 또한 사용 사례와 클래스의 수 및 복잡성을 평가하면 필요한 개발 작업을 예상할 수 있습니다. 처음 몇 개의 반복을 완료한 경우 적용된 요구 사항과 적용할 요구 사항을 비교하면 남은 프로젝트의 비용과 범위를 대략적으로 계산할 수 있습니다.

-추상화 수준

모델은 소프트웨어에 따라 추상화 범위를 가집니다. 가장 구체적인 모델은 프로그램 코드를 직접 나타내고, 가장 추상적인 모델은 코드에 나타내거나 나타내지 않는 비즈니스 개념을 나타냅니다. 모델은 여러 종류의 다이어그램을 통해 볼 수 있습니다. 다양한 종류의 다이어그램은 서로 다른 추상화 수준에서 디자인을 기술하는 데 도움이 됩니다.

디자인 수준	다이어그램 형식
비즈니스 프로세스 시스템이 사용될 컨텍스트를 이해하면 사용자가 거기에서 무엇을 필요로 하는지 이해하는 데 도움이 됩니다.	• 동작 다이어그램은 비즈니스 목표를 달성하기 위해 사용자와 시스템 간의 워크플로를 기술합니다. • 개념 클래스 다이어그램은 비즈니스 프로세스 내에서 사용되는 비즈니스 개념을 기술합니다.
사용자 요구 사항 사용자가 시스템에서 필요로 하는 것에 대한 정의입니다.	• 사용 사례 다이어그램은 사용자 및 기타 외부 시스템과 현재 개발 중인 시스템과의 상호 작용을 요약합니다.다른 문서를 각 사용 사례에 첨부하여 해당 사례에 대해 상세히 기술할 수 있습니다. • UML 클래스 다이어그램은 사용자와 시스템이 의사소통하는 정보의 형식을 기술합니다. • 비즈니스 규칙 및 서비스 품질 요구 사항을 별도의 문서에 기술할 수 있습니다.
고급 디자인 시스템의 전체 구조, 즉 주요 구성 요소 목록 및 이러한 구성 요소가 함께 연결되는 방식을 보여 줍니다.	• 구성 요소 다이어그램은 시스템이 여러 파트로 구성되는 방식을 기술합니다. • 시퀀스 다이어그램은 각 사용 사례를 구현하기 위해 구성 요소가 통신하는 방식을 보여 줍니다. • UML 클래스 다이어그램은 구성 요소의 인터페이스 및 구성 요소 간에 전달되는 데이터의 형식을 기술합니다.
디자인 패턴 디자인의 모든 파트에 사용되는 디자인 문제 해결 방법 및 규칙입니다.	• UML 클래스 다이어그램은 패턴의 구조를 기술합니다. • 시퀀스 다이어그램 또는 동작 다이어그램은 상호 작용과 알고리즘을 보여 줍니다.
코드 분석 몇 가지 형식의 다이어그램을 코드에서 생성할 수 있습니다.	• 시퀀스 다이어그램은 코드에서 개체 간의 상호 작용을 보여 줍니다. • 레이어 다이어그램은 클래스 간의 종속성을 보여 줍니다.업데이트된 코드는 레이어 다이어그램과 비교하여 유효성을 검사할 수 있습니다. • .NET 클래스 다이어그램은 코드의 클래스를 보여 줍니다.