1. StarUML 사용법

StarUML 소개

- 다양한 다이어그램을 간편한 사용자 인터페이스를 이용해 생성하는 UML 모델링 툴
- 클래스 다이어그램, 유스케이스 다이어그램, 순차 다이어그램, 활동 다이어그램, 컴포넌트 다이어그램, 배치 다이어그램 등을 지원
- StarUML 공식 사이트(http://staruml.io)에서 무료 다운로드

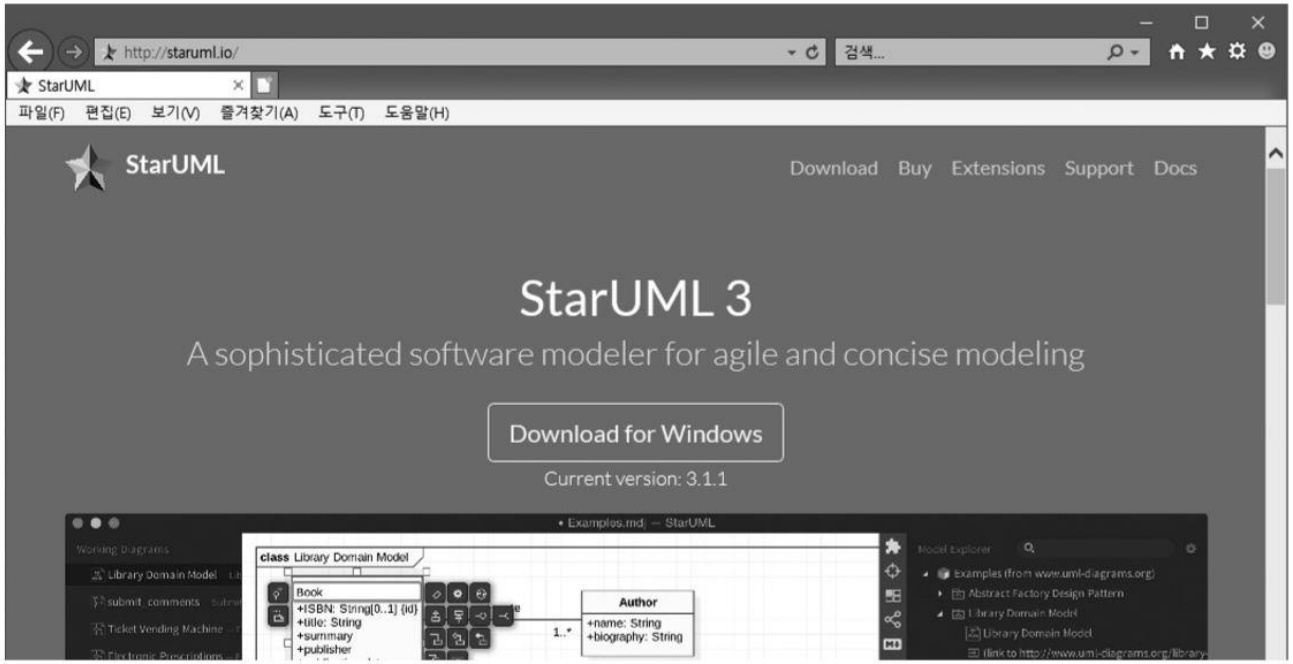


그림 12-1 StarUML 다운로드 사이트

1. StarUML 사용법

StarUML 설치와 화면 소개

- 메인 화면과 도구모음, 브라우저 영역, 다이어그램 영역, 편집기, 툴박스 영역으로 구분

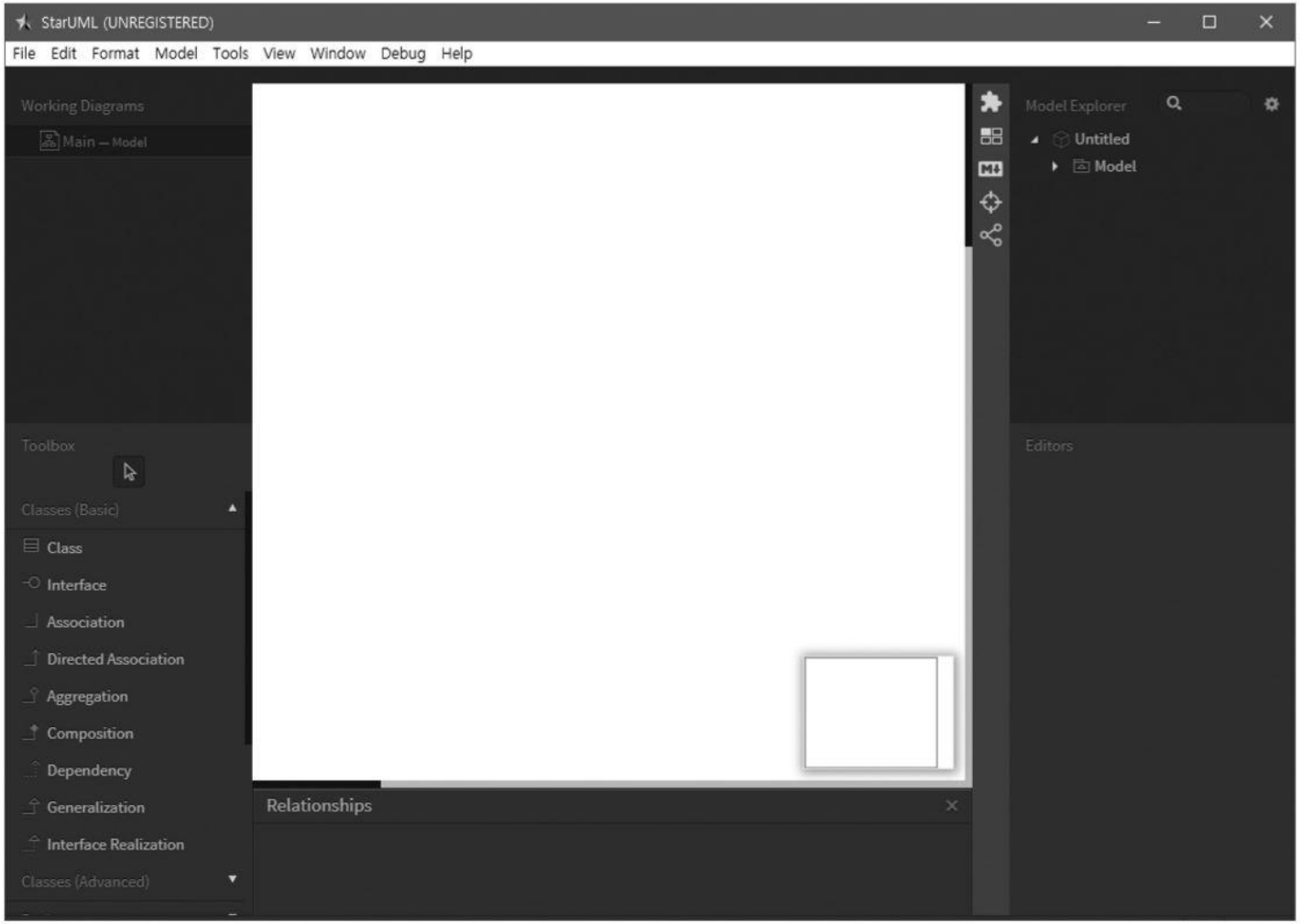


그림 12-2 StarUML 실행 화면

새 프로젝트 생성

- [New] 메뉴를 이용해 새 프로젝트를 생성

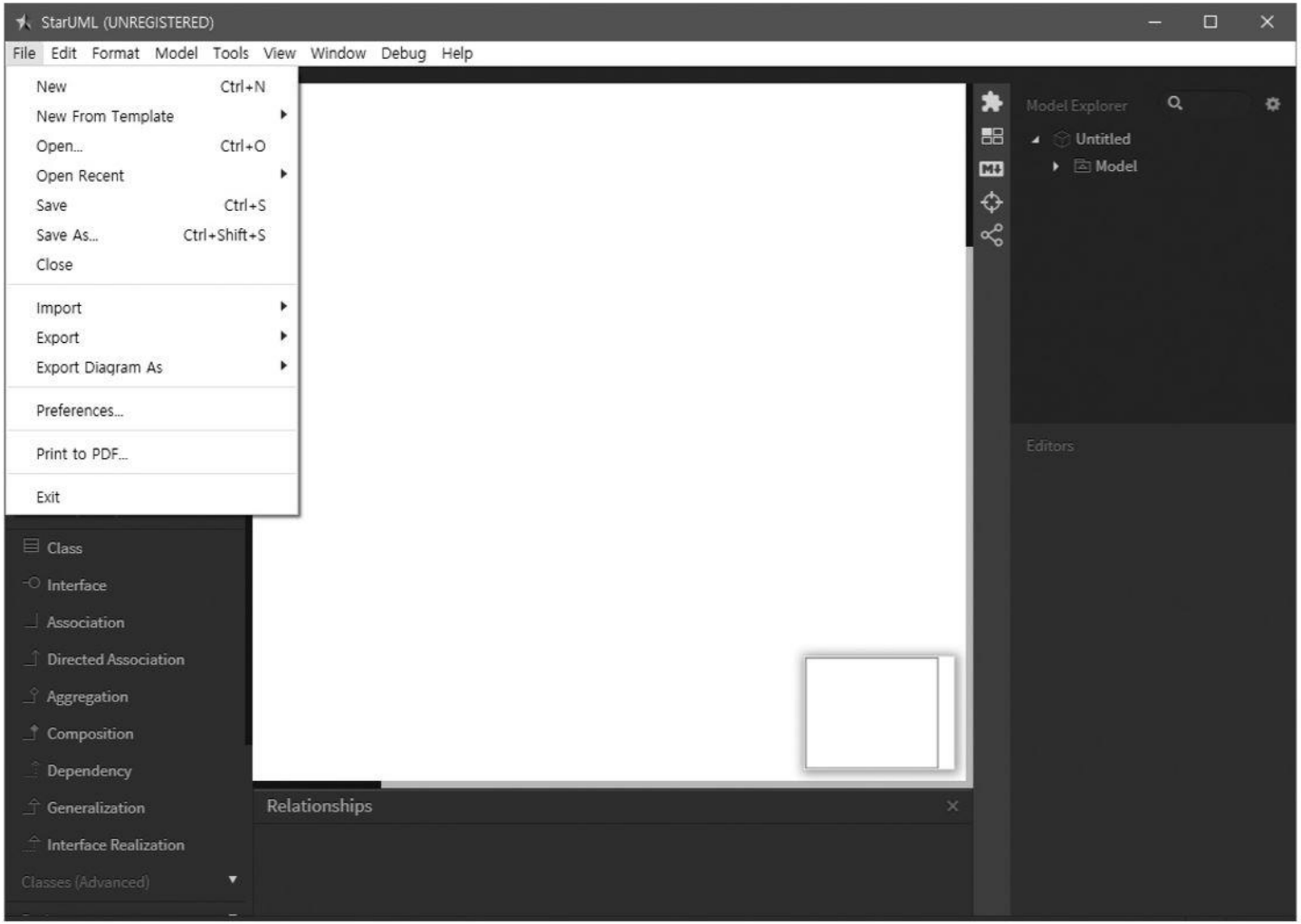


그림 12-3 새 프로젝트 생성

2. UML 기본 작성법

■ 새 다이어그램 작성

- Model Explorer에서 마우스 오른쪽 버튼을 누르고 [Add Diagram] 메뉴에서 원하는 다이어그램을 선택

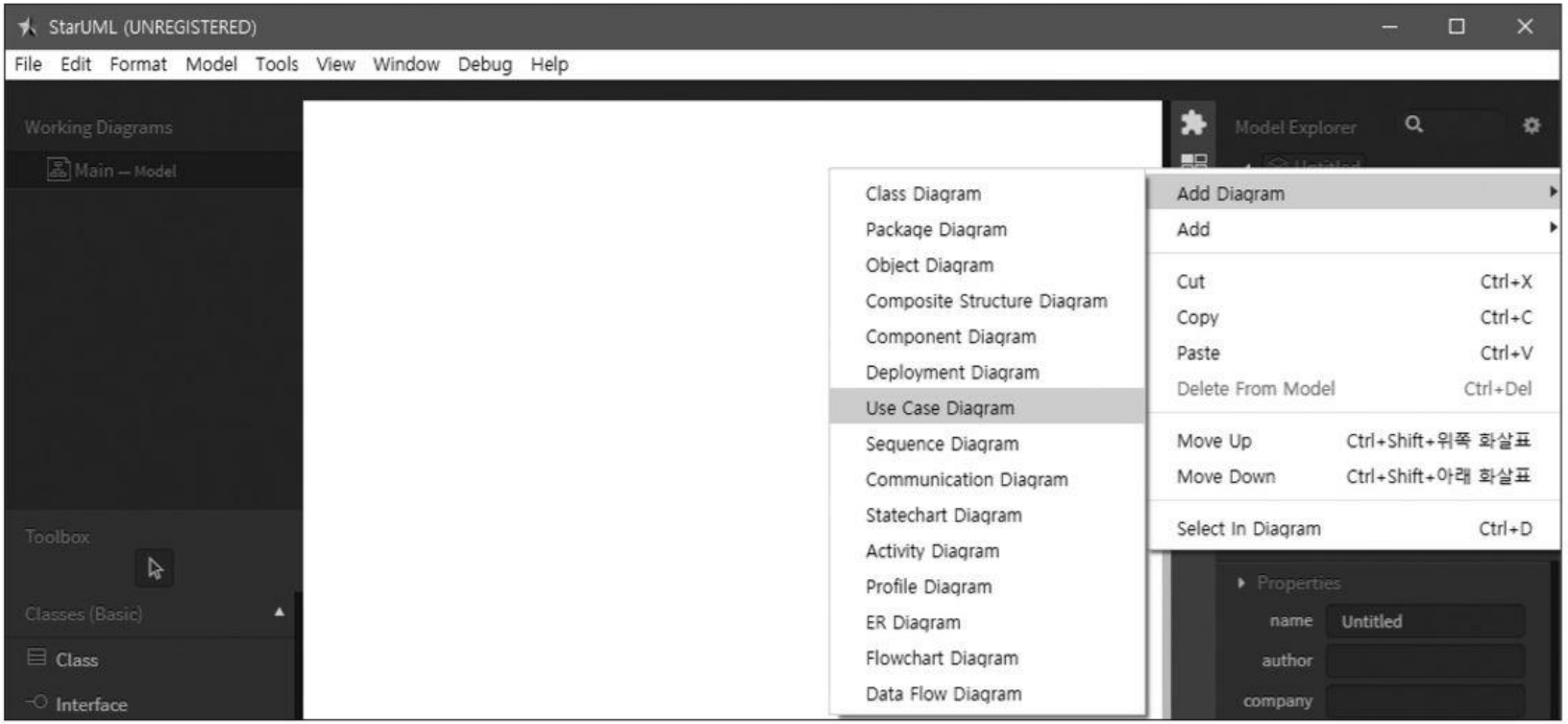


그림 12-4 새 다이어그램 생성

■ 유스케이스 다이어그램 작성

- Model Explorer에서 마우스 오른쪽 버튼을 누른 다음 [Add Diagram]-[Use Case Diagram] 메뉴를 선택

표 12-1 유스케이스 다이어그램 툴박스

툴	기능
Package	모델 요소들을 논리적으로 그룹화할 때 사용함
UseCase Subject	시스템과 외부 시스템의 경계를 생성함
UseCase	시스템이 제공하는 기능을 의미함
Actor	시스템을 사용하는 사용자나 외부 시스템을 의미함
Association	유스케이스와 액터 간의 연관 관계를 나타냄
DirectedAssociation	유스케이스와 액터 간의 연관 관계가 있을 때 포함을 표시할 수 있음
Generalization	일반적인 요소와 더 구체적인 요소 간의 관계를 나타냄, 객체 지향의 상속과 같은 의미
Dependency	다른 요소가 요구되는 의존 관계를 나타냄
Include	유스케이스를 수행할 때 반드시 수행해야 하는 유스케이스를 나타낼 경우에 사용함
Extend	한 유스케이스가 특정 시점에 여러 형태로 분류될 경우에 사용함

유스케이스 다이어그램 작성

- 툴박스에서 [Actor]를 선택하고 메인 화면을 클릭해 액터를 생성
- 이름을 Messenger로 지정하고 드래그 앤 드롭으로 자유롭게 이동하거나 크기를 조정

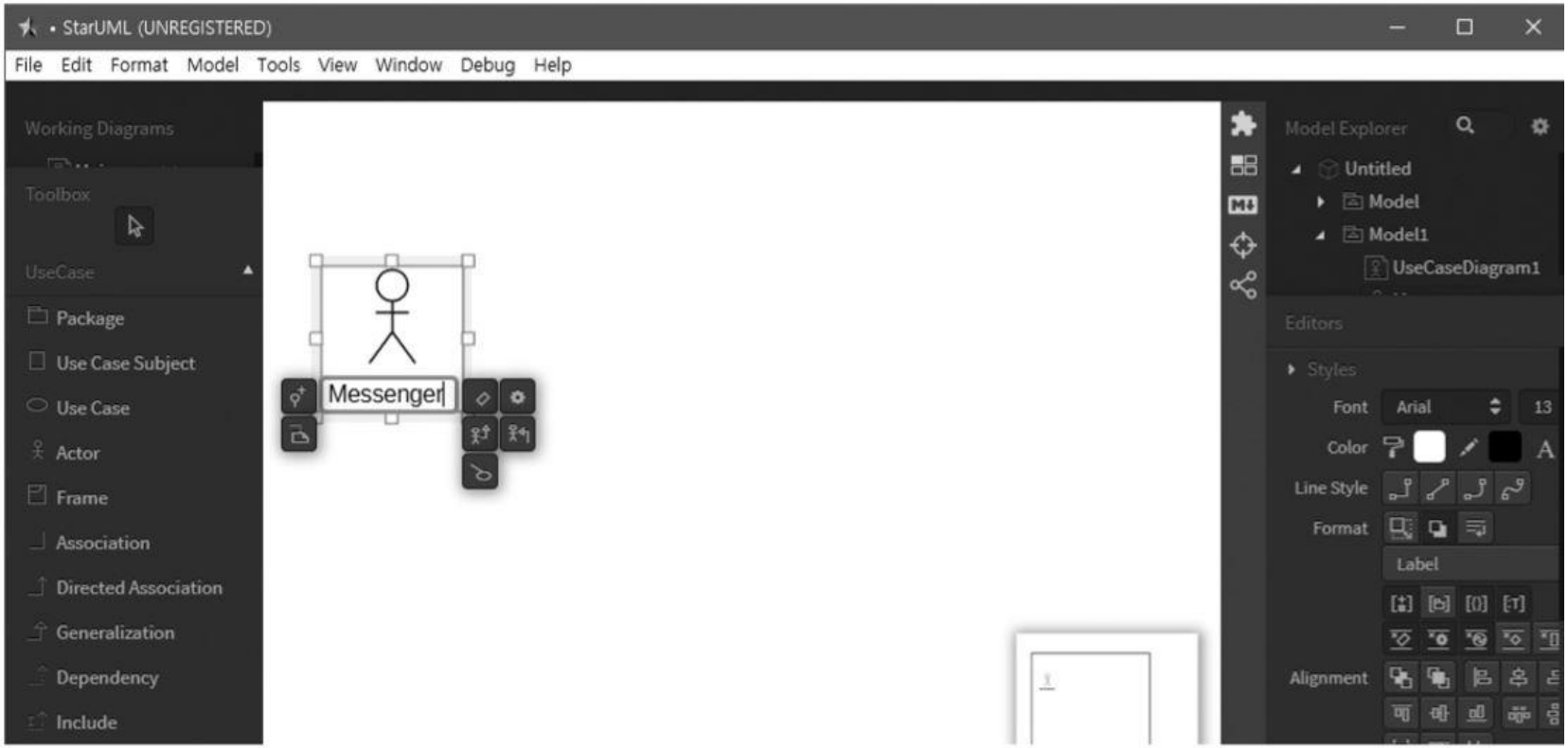


그림 12-5 액터 생성

유스케이스 다이어그램 작성

- 툴박스에서 [Use Case Subject]를 선택하고 윈도 영역을 클릭해 시스템 경계를 생성하고 이름을 Subject1로 지정

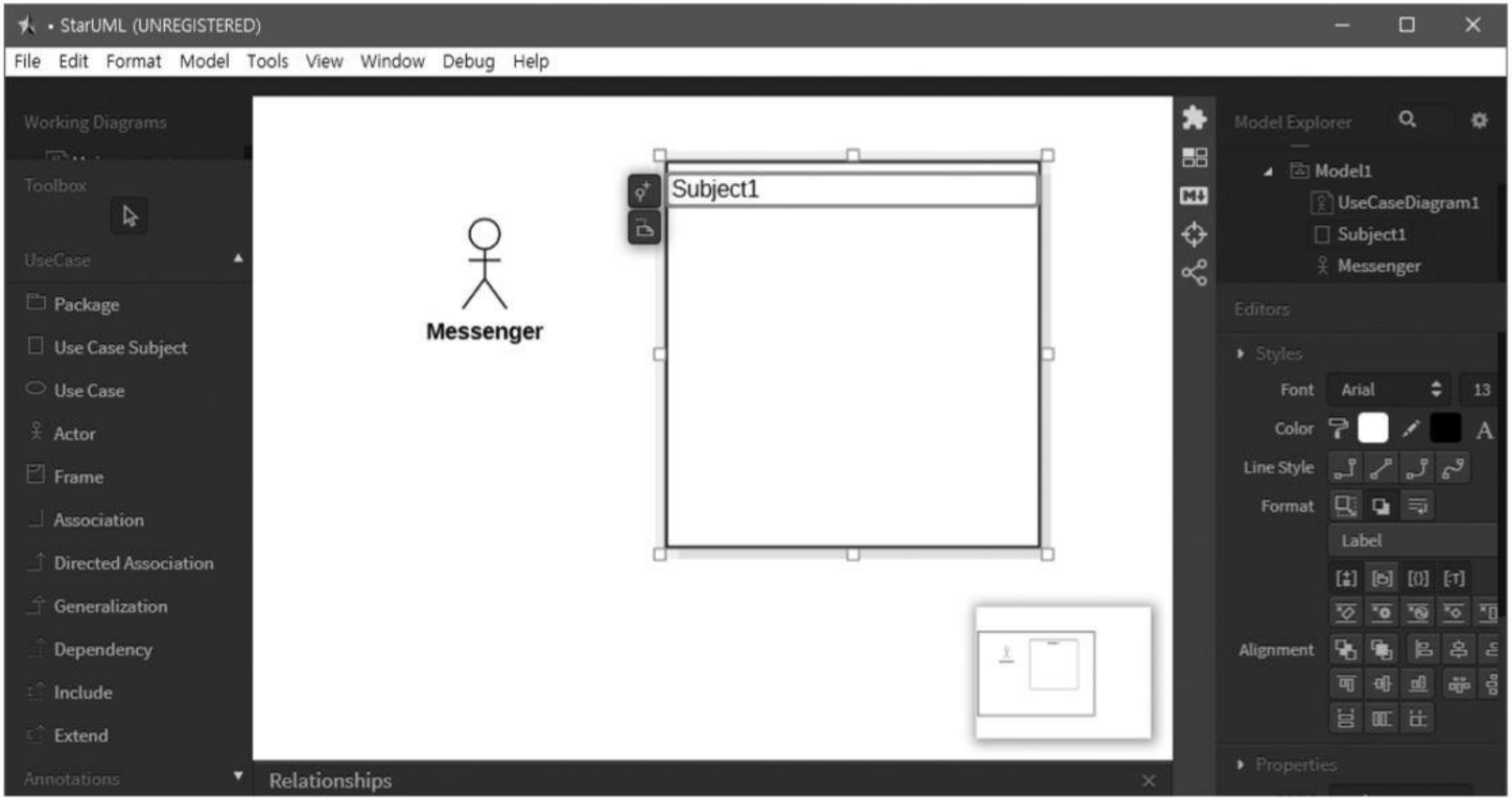


그림 12-6 시스템 경계 생성

■ 유스케이스 다이어그램 작성

- 툴박스에서 [Use Case]를 선택하고 윈도 영역을 클릭해 유스케이스 3개를 생성
- 각각 Input, Output, Description으로 이름을 지정

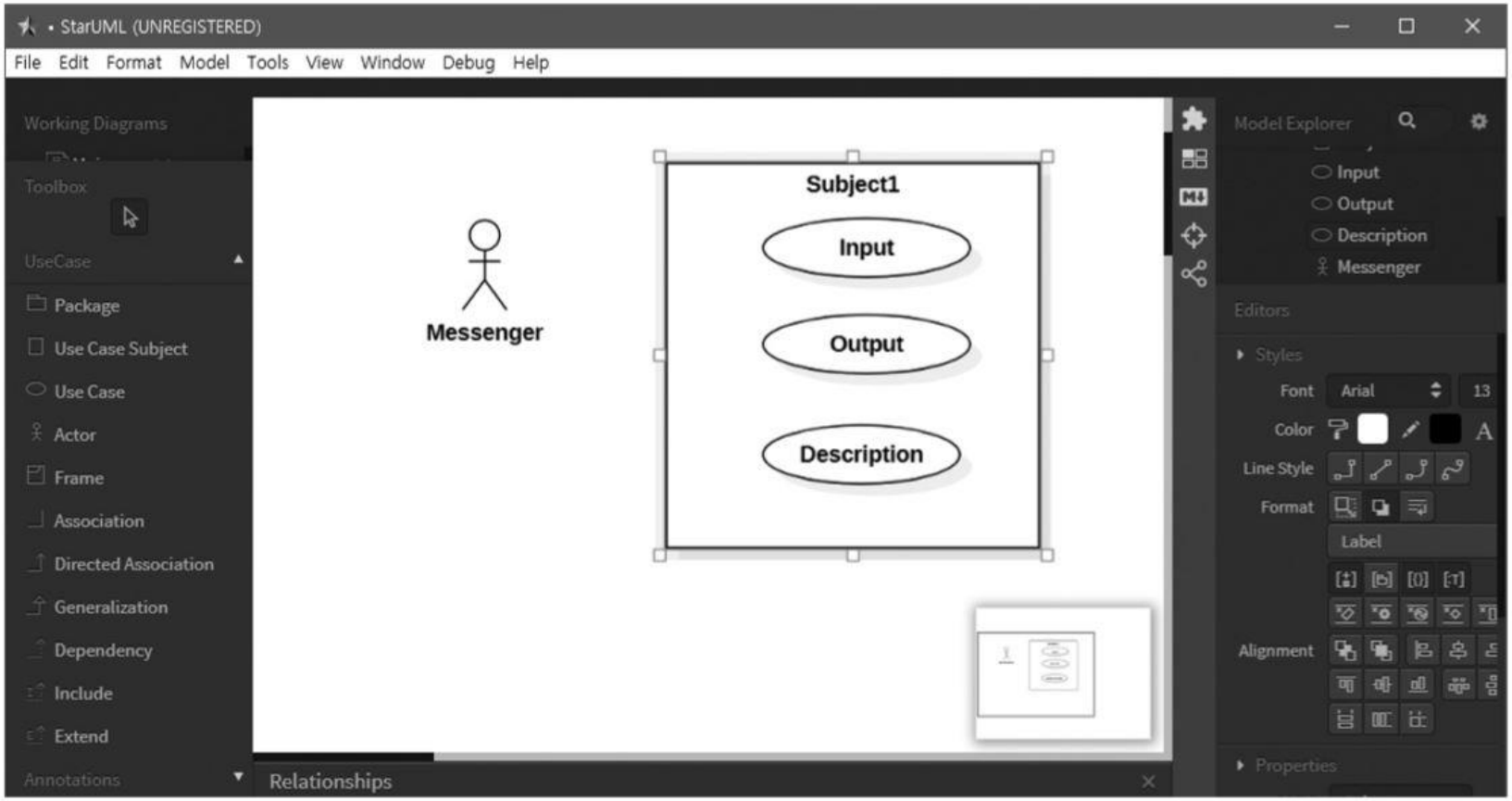


그림 12-7 유스케이스 생성

■ 유스케이스 다이어그램 작성

- 툴박스의 [Association]을 사용해 그림과 같이 액터와 유스케이스의 연관 관계를 지정
- [Properties] 창을 이용해 다중성을 표현

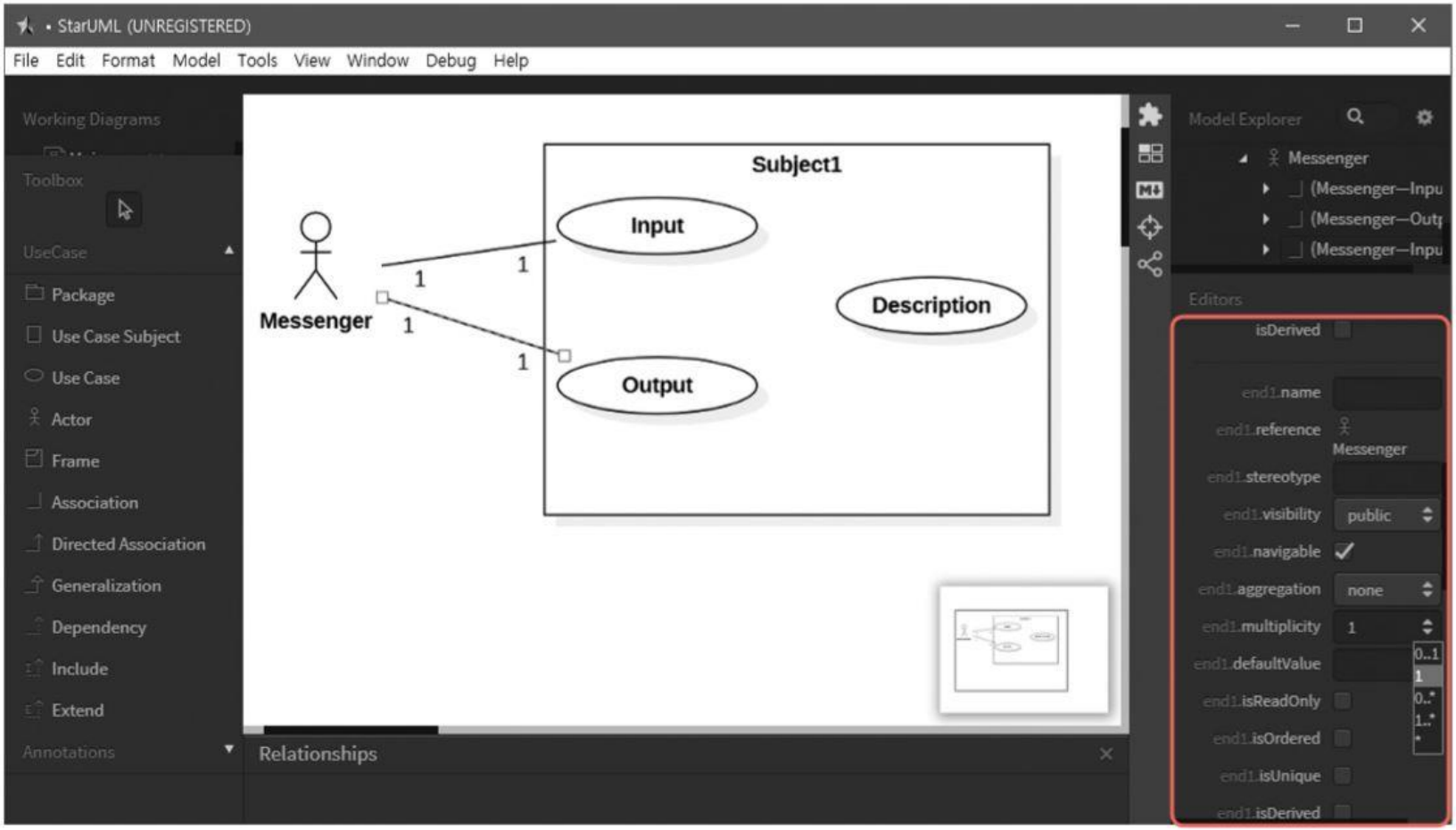


그림 12-8 액터와 유스케이스의 연관 관계

2. UML 기본 작성법

■ 유스케이스 다이어그램 작성

- 툴박스의 [Include]를 사용해 그림과 같이 유스케이스 간 포함 관계를 지정

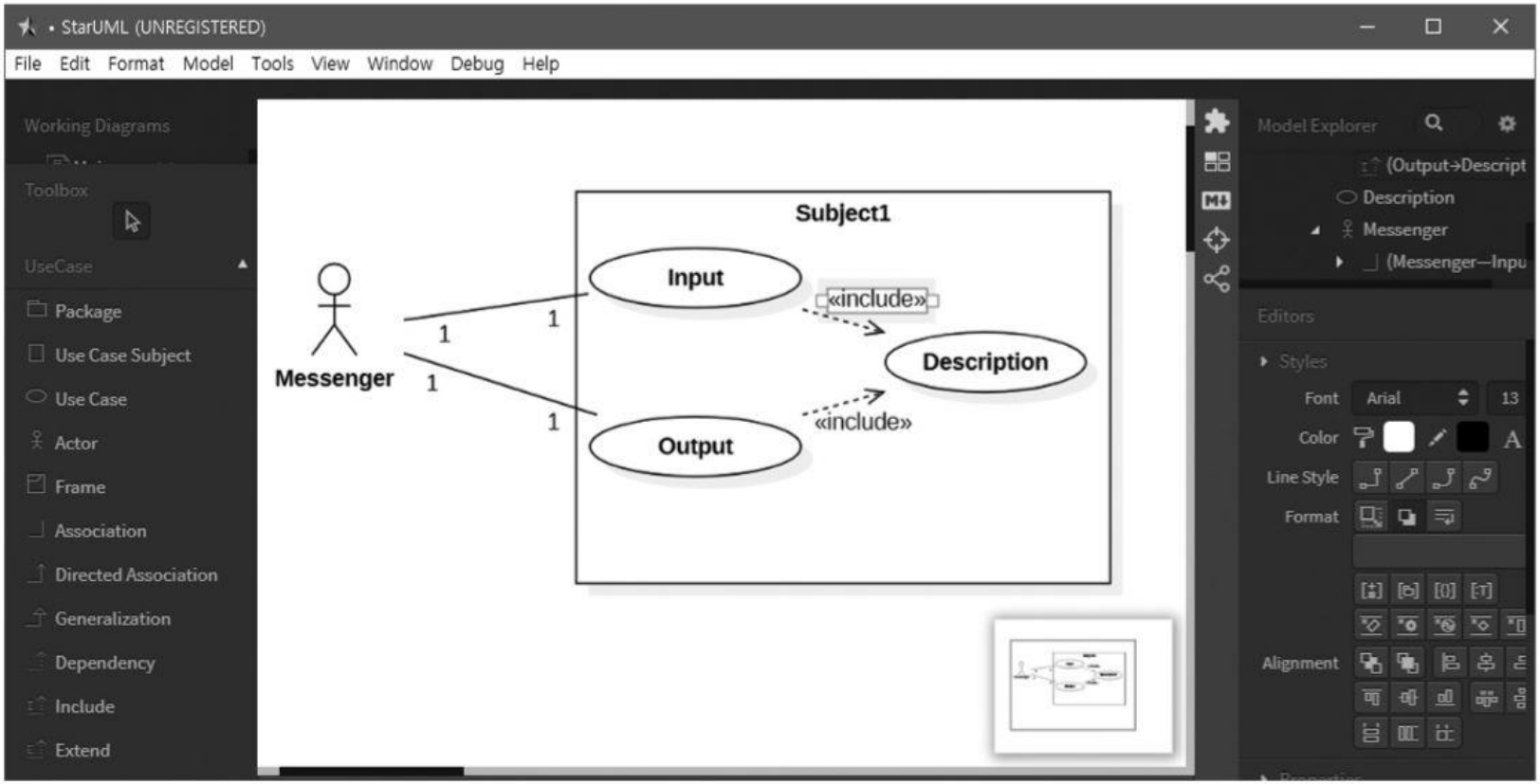


그림 12-9 유스케이스 간 포함 관계

■ 클래스 다이어그램 작성

- Model Explorer에서 마우스 오른쪽 버튼을 누른 다음 [Add Diagram]-[Class Diagram] 메뉴를 선택

표 12-2 클래스 다이어그램 툴박스

툴	기능
Class	객체의 속성과 메서드를 모델링한 것
Interface	클래스에서 메서드 선언 부분만 모델링한 것
Association	클래스 사이의 연관 관계를 나타냄(Qualifier는 사용 가능)
DirectedAssociation	한 클래스와 다른 클래스가 연관 관계가 있을 때 사용함(Qualifier는 사용할 수 없음)
Aggregation	클래스 사이의 포함 관계를 나타냄
Composition	클래스 사이의 종속 관계를 나타냄
Dependency	한 클래스의 변화가 다른 클래스의 변화에 영향을 주는 관계를 나타냄
Generalization	클래스 사이의 일반화(상속) 관계를 나타냄
Interface Realization	인터페이스와 클래스를 연결할 때 사용함

■ 클래스 다이어그램 작성

- 툴박스의 [Class]를 이용해 Item, Input, Description, Output 네 가지의 클래스 다이어그램을 생성
- 클래스에서 마우스 오른쪽 버튼을 누른 다음 [Add]-[Attribute] 메뉴로 속성을, [Add]-[Operation] 메뉴로 오퍼레이션을 생성

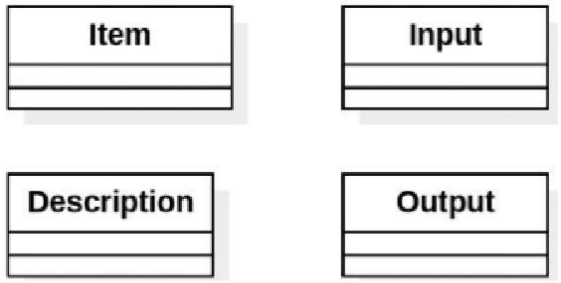


그림 12-10 클래스 생성

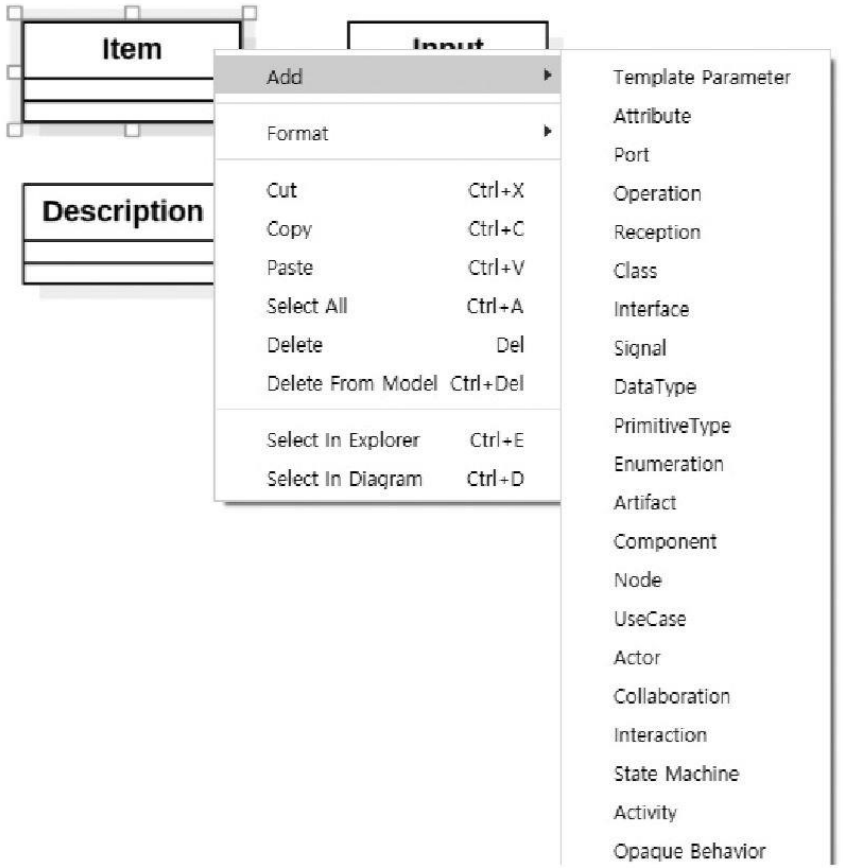


그림 12-11 클래스의 속성과 오퍼레이션 생성 메뉴

클래스 다이어그램 작성

- 클래스에 속성과 오퍼레이션을 생성
- 툴박스의 [Association]을 사용해 Input과 Description, Output과 Description 사이의 연관 관계를 지정
- 툴박스의 [Generalization]을 사용해 Item과 Description, Input과 Output 사이에 일반화 관계를 지정

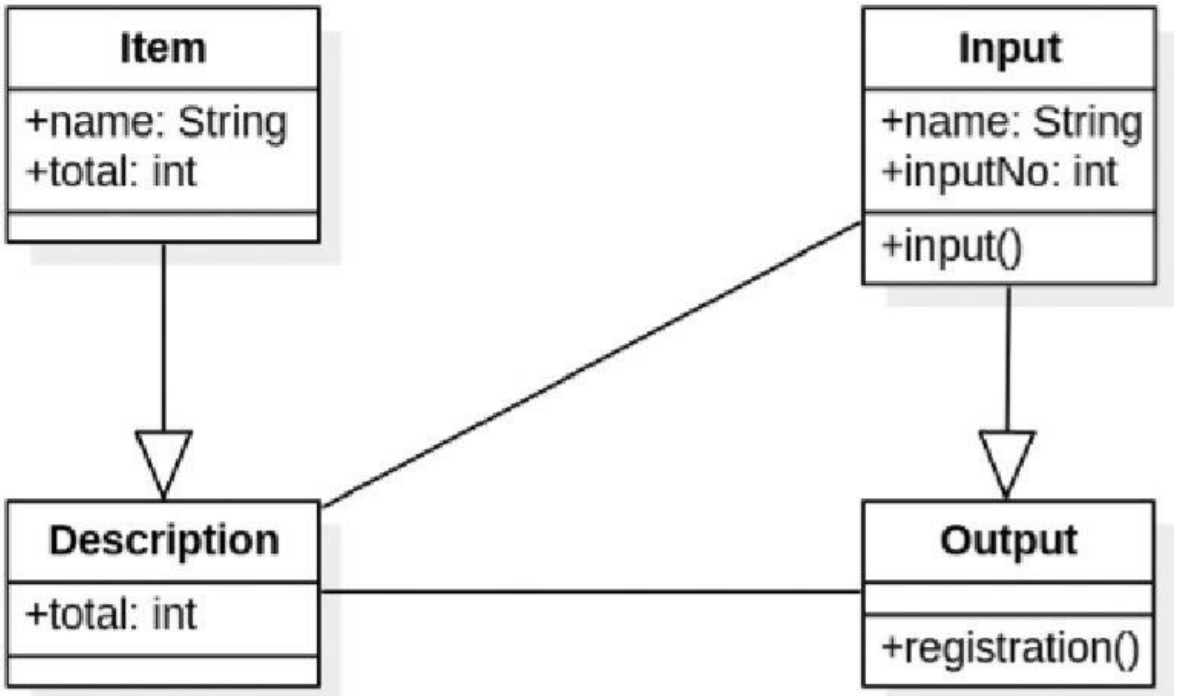


그림 12-12 클래스의 속성, 오퍼레이션, 관계 생성

■ 순차 다이어그램 작성

- Model Explorer에서 마우스 오른쪽 버튼을 누른 다음 [Add Diagram]-[Sequence Diagram] 메뉴를 선택

표 12-3 순차 다이어그램 툴박스

툴	기능
Lifeline	클래스로부터 생성된 객체
Message	두 객체 간에 메시지를 전송함
SelfMessage	자기 자신에게 메시지를 전송함

- 툴박스의 [Lifeline]를 이용해 Messenger, Input, Output, Description 객체를 생성

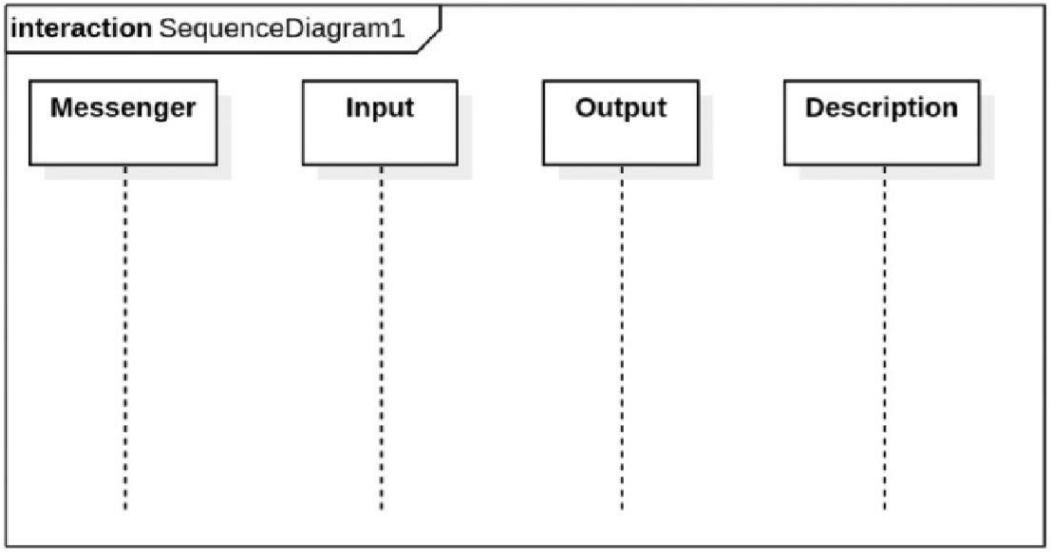


그림 12-13 객체 생성

순차 다이어그램 작성

- 툴박스의 [Message]를 이용해 Messenger에서 Input으로, Input에서 Description으로 Messenger에서 Output으로, Output에서 Description으로 메시지 순서를 지정

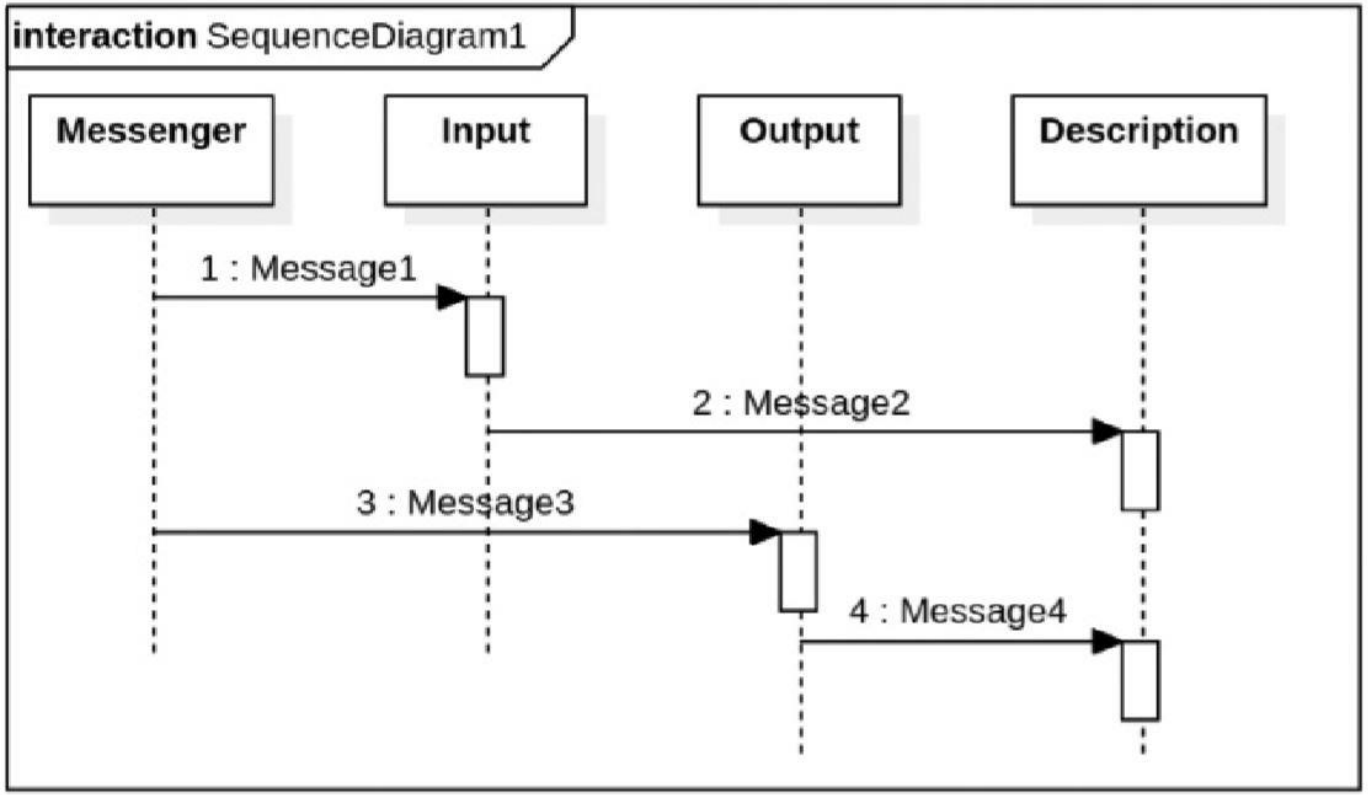


그림 12-14 객체 간의 메시지 전송

순차 다이어그램 작성

- 툴박스의 [Message]를 이용해 Messenger에서 Input으로, Input에서 Description으로 Messenger에서 Output으로, Output에서 Description으로 메시지 순서를 지정

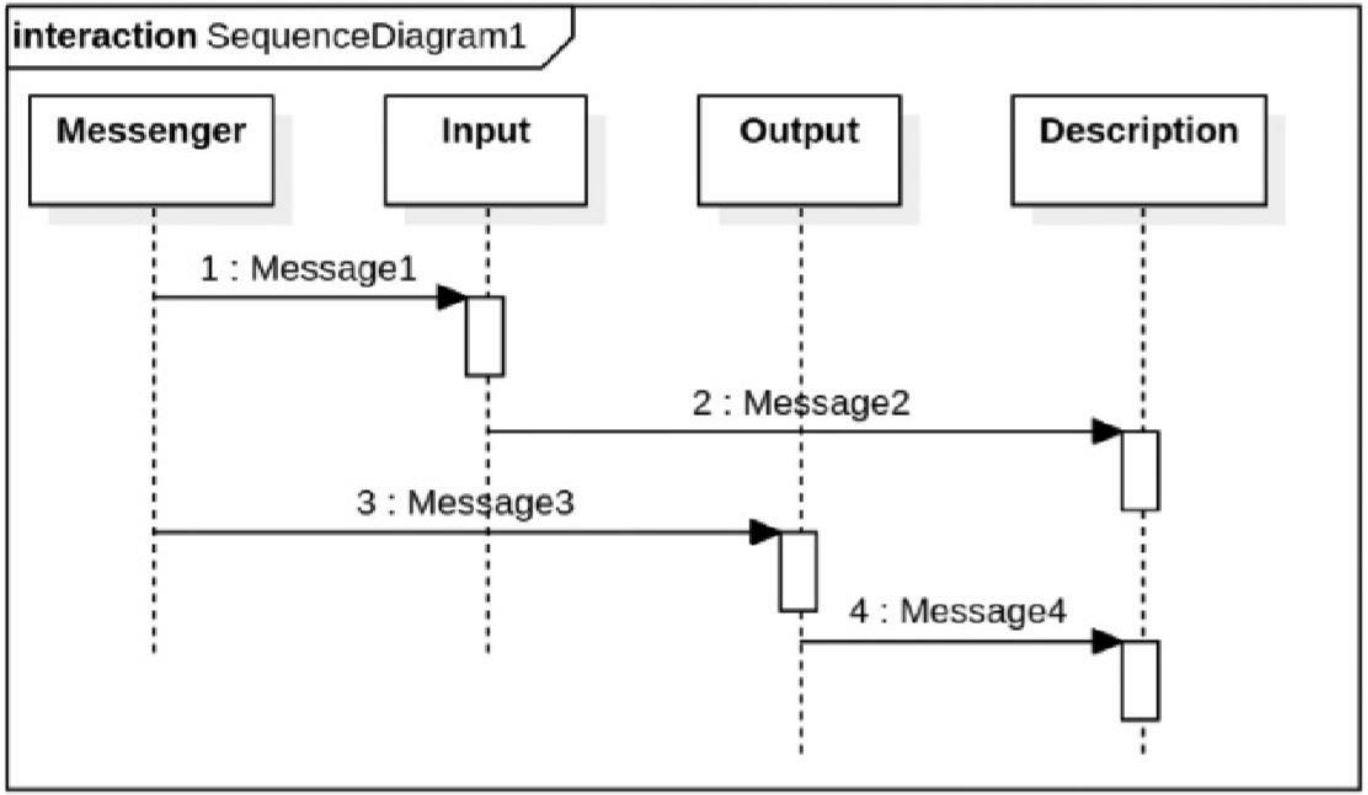


그림 12-14 객체 간의 메시지 전송

순차 다이어그램 작성

- 툴박스의 [Decision]를 이용해 Description과 ItemAdd, ItemRegistration 사이에 분기점을 넣음
- 툴박스의 [Control Folow]를 이용해 연결

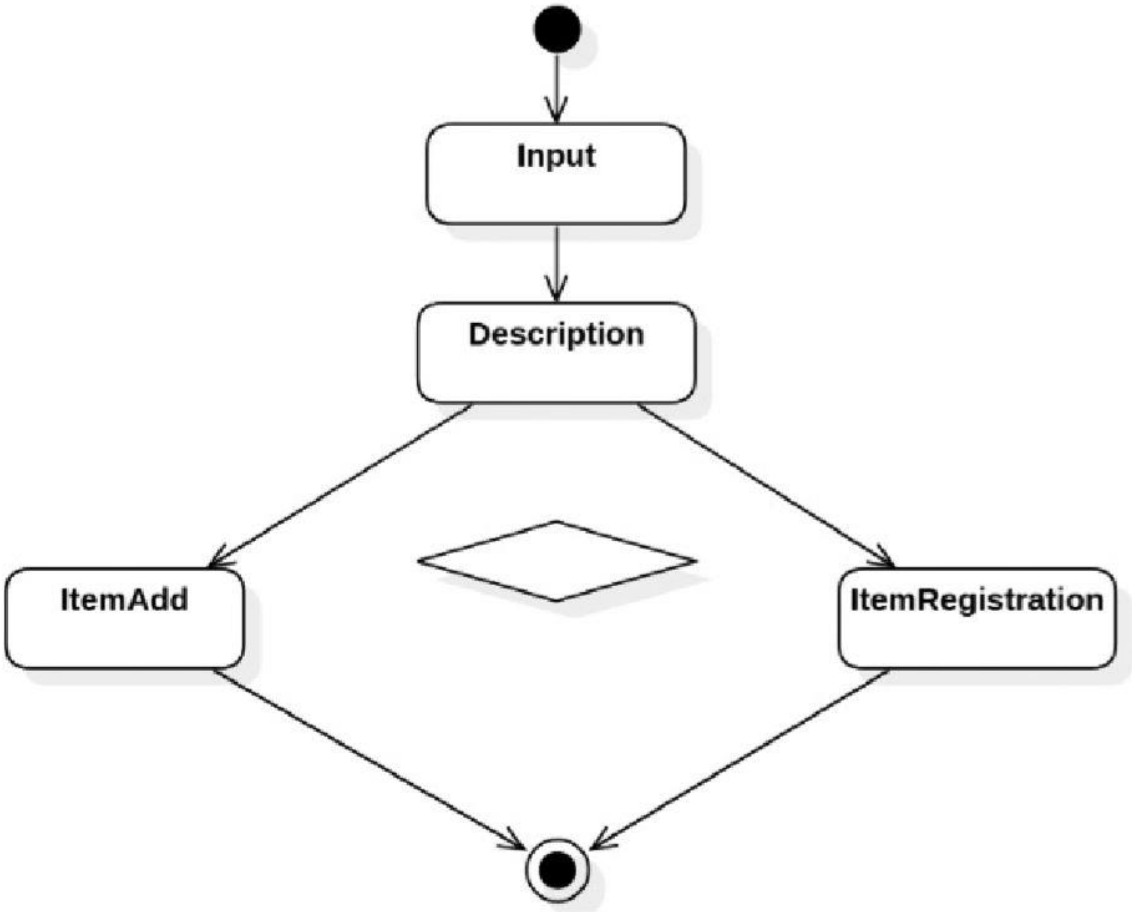


그림 12-16 활동 다이어그램 완성

프로젝트 생성

- StarUML을 실행 후, [Save AS] 메뉴를 이용해 프로젝트를 VendingMachine으로 저장

그림 12-17 VendingMachine 프로젝트 생성

3. 자판기 시스템

■ 유스케이스 다이어그램 작성

① 자판기 시스템은 사용자가 동전이나 지폐를 투입하면 시스템이 실행

② 음료수 가격은 500원, 600원, 700원 세 종류

③ 각각 기준 가격 이상의 돈이 투입되면 자동으로 각 음료수 선택 버튼에 불이 점등

④ 음료수를 선택하고 남은 잔액은 잔액 표시 화면에 표시

⑤ 사용자가 잔액 반환 버튼을 누르면 잔액이 반환

⑥ 잔액이 음료수 가격 미만이면 음료수 선택 버튼에 불이 소등

유스케이스 다이어그램 작성

- 액터 : User(사용자)
- 유스케이스 : Input(투입구), Choice(음료수 선택), ReturnMoney(잔액 반환), DisplayMoney(잔액 표시 화면)
- 관계 : User와 Input, User와 Choice, User와 ReturnMoney는 연관 관계
  Input과 DisplayMoney, Choice와 DisplayMoney, ReturnMoney와 DisplayMoney는 포함 관계
- 시스템 경계 : Vending Machine System

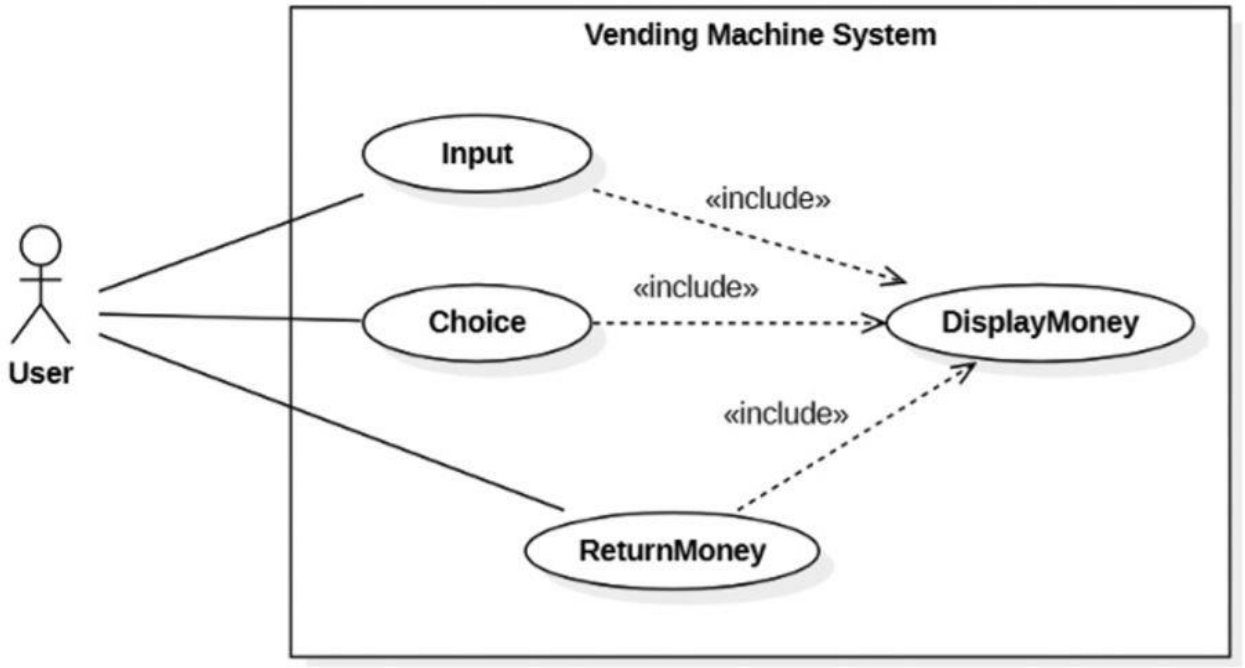


그림 12-18 유스케이스 다이어그램 작성

패키지 다이어그램 작성

- 이후에 만들 소스나 다이어그램을 쉽게 관리할 수 있도록 VendingMachine, Data, UserInterface 패키지 추가

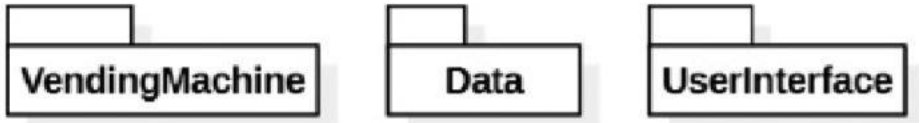


그림 12-19 패키지 다이어그램 작성

클래스 다이어그램 작성

- Model Explorer의 VendingMachine 패키지에서 마우스 오른쪽 버튼을 누른 다음 [Add Diagram]-[Class Diagram] 메뉴를 선택
- Choice, ChoiceItem, ItemExplain 클래스를 추가

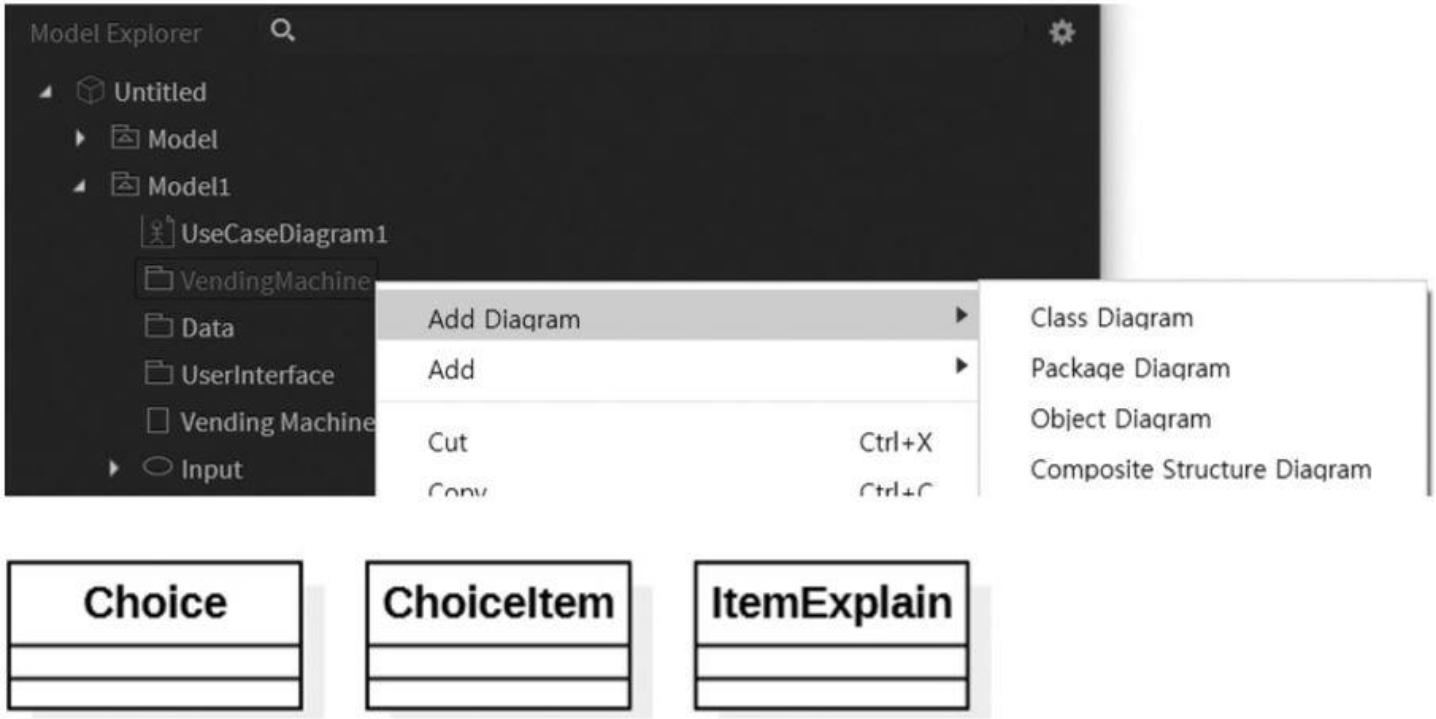


그림 12-20 VendingMachine 패키지에 클래스 다이어그램 추가

클래스 다이어그램 작성

- Choice 클래스에 총 합계를 저장하기 위한 private total 속성 추가
- 총 합계를 계산하기 위한 public calcTotal() 오퍼레이션 추가
- 상품 및 가격을 등록하기 위한 private cashRegister 속성 추가
- 잔액을 계산하기 위한 public calcTax() 오퍼레이션 추가
- ChoiceItem 클래스에는 private myExplain 속성 추가

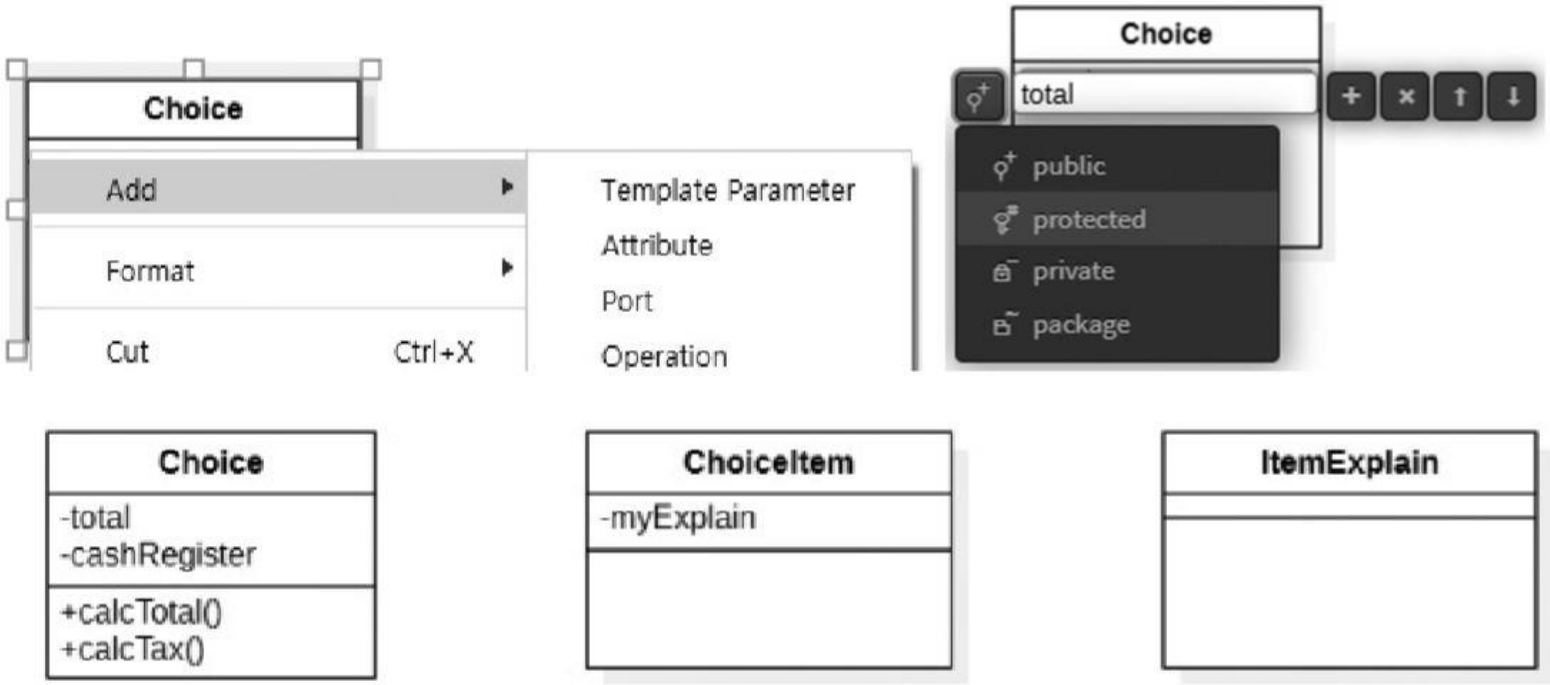
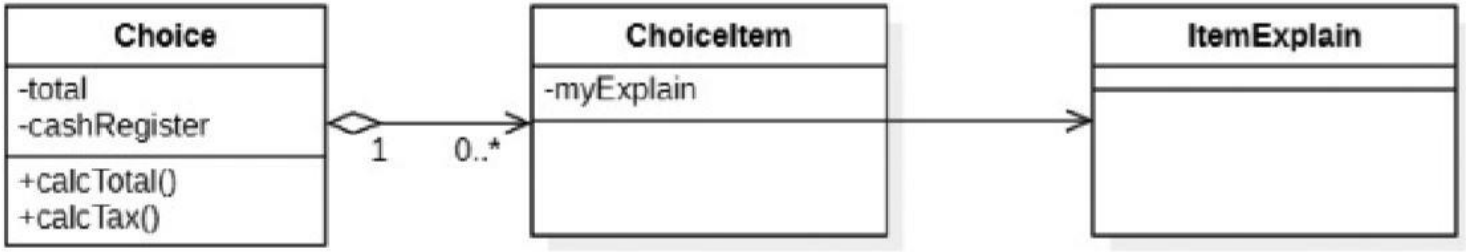


그림 12-21 클래스에 속성과 오퍼레이션 추가

클래스 다이어그램 작성

- 툴박스의 [Directed Association]를 이용해 클래스 간의 연관 관계를 설정



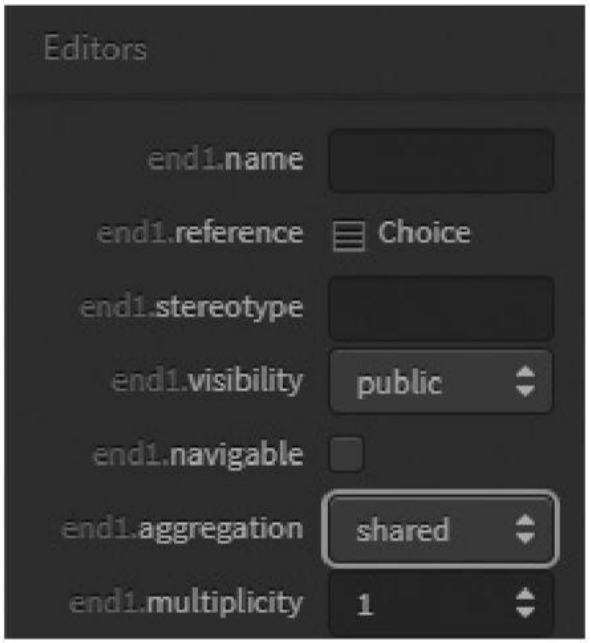


그림 12-22 클래스 생성

클래스 다이어그램 작성

- hoice 클래스, ChoiceItem 클래스, ItemExplain 클래스에 생성자 추가
- Choice 클래스에 기본 생성자 추가
- ChoiceItem 클래스에는 기본 생성자, 매개변수가 ItemExplain인 생성자, 매개변수가 quantity인 생성자 추가
- ItemExplain 클래스에는 매개변수가 name인 생성자와 매개변수가 name, price인 생성자를 추가
- 속성과 countMoney 클래스를 추가

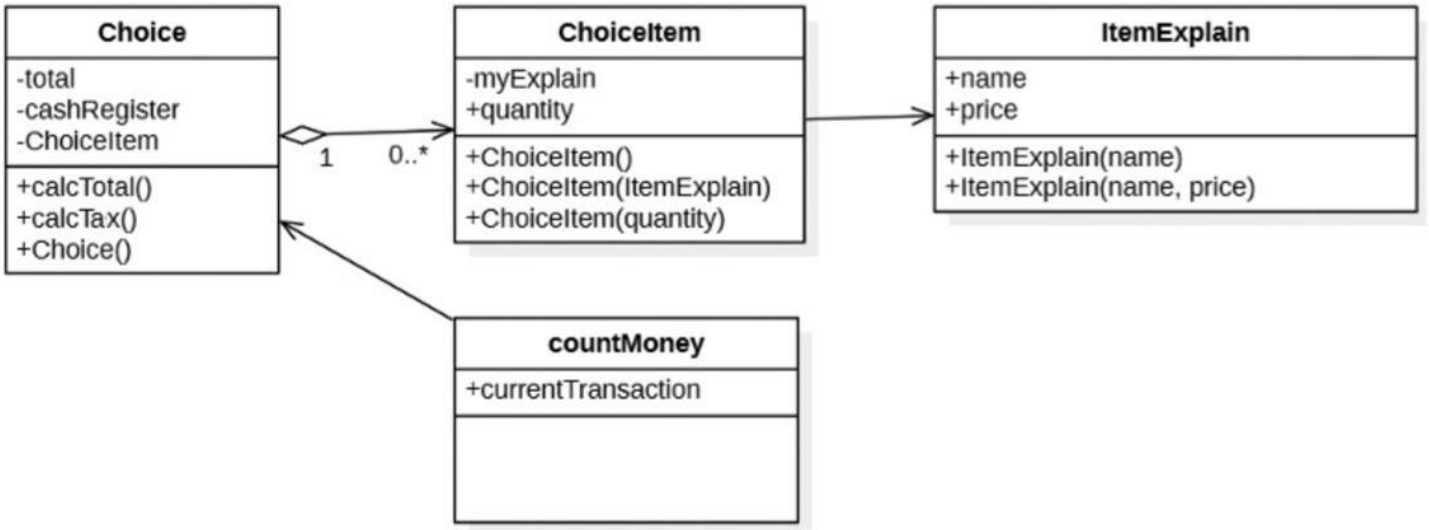


그림 12-23 클래스 다이어그램 작성

순차 다이어그램 작성

- 순차 다이어그램을 추가하고 이름을 startVendingMachine으로 지정
- aCustomer, inputMoney, currentTransaction, anItem 객체를 추가
- inputMoney 객체는 countMoney 클래스, currentTransaction 객체는 Choice 클래스, anItem 객체는 ChoiceItem 클래스를 선택
- aCustomer 객체에서 inputMoney 객체로 보내는 메시지 1개 생성
  currentTransaction 객체에서 anItem 객체로 보내는 메시지 생성, inputMoney 객체에서 currentTransaction 객체로 보내는 메시지를 2개 생성하는데 2번 메시지는 inputMoney 객체의 활성바에서 시작

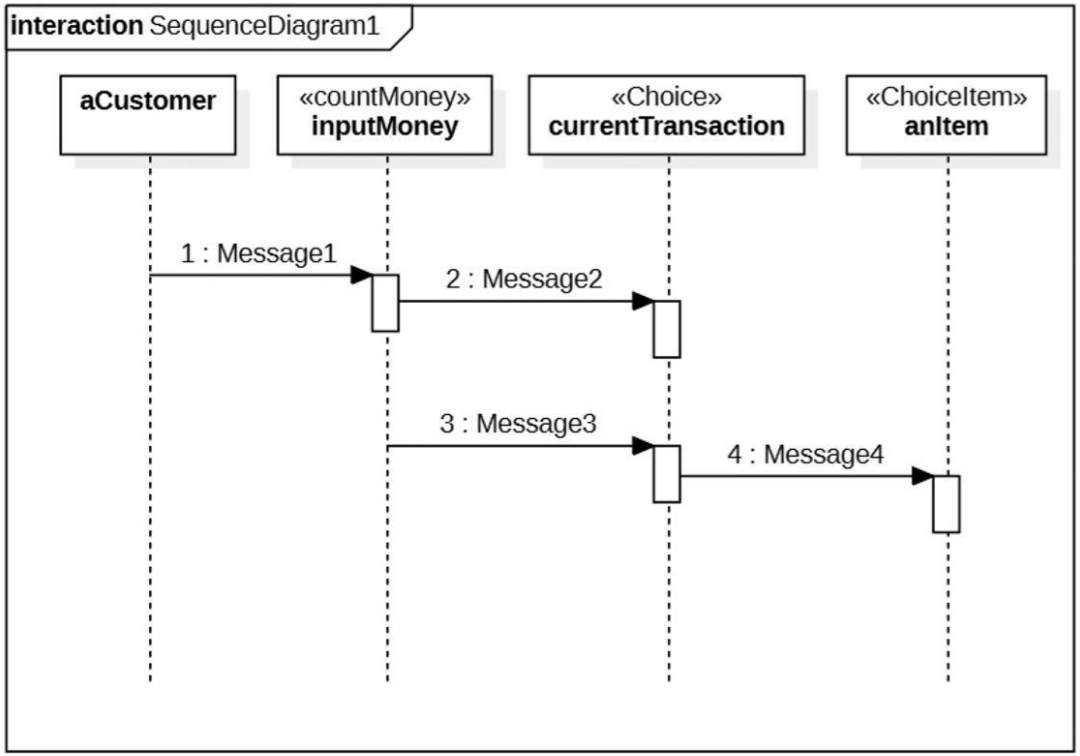


그림 12-24 순차 다이어그램 작성

자바 코드로 변환

- StarUML에서는 추가 플러그인을 설치하면 생성한 다이어그램을 소스 코드로 변환할 수 있음
- StarUML의 [Tools]-[Extension Manager] 메뉴를 선택 후, JAVA를 검색한 다음 [Install]를 눌러 설치

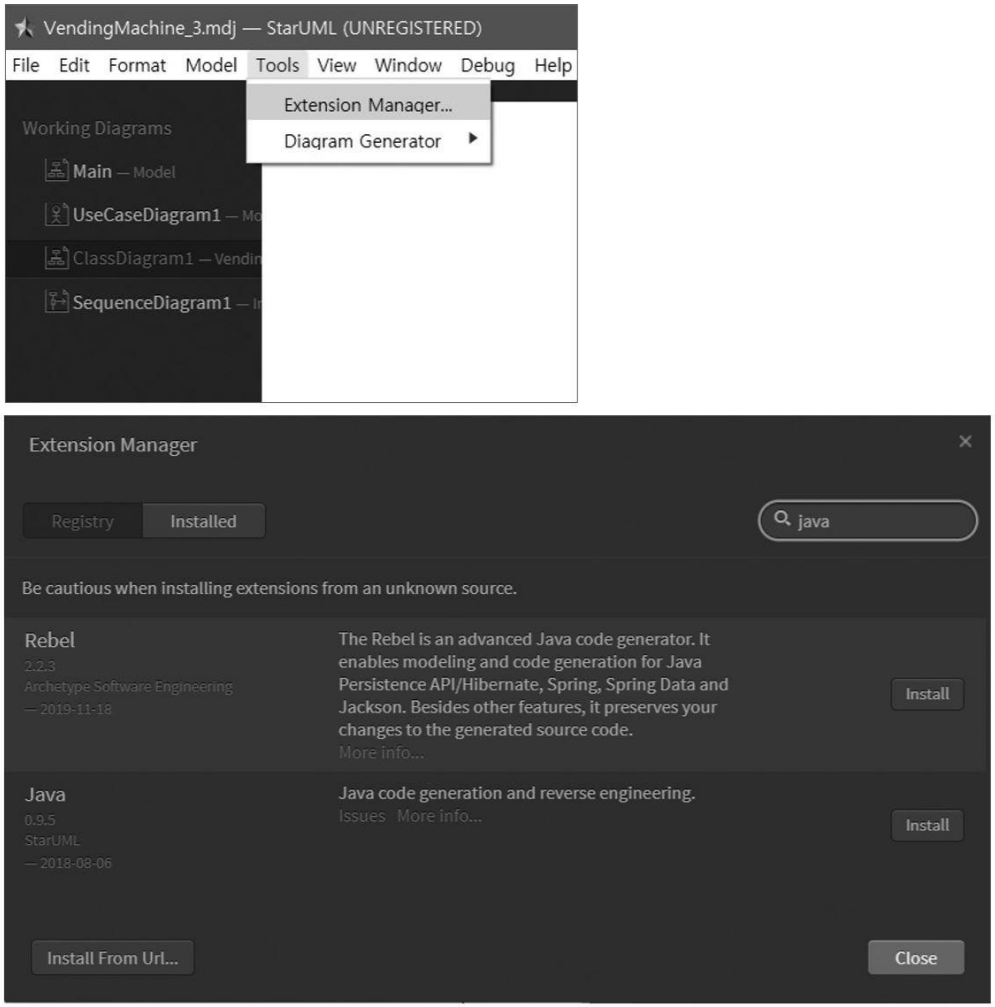


그림 12-25 자바 플로그인 설치

자바 코드로 변환

- StarUML을 다시 실행하면 [Tools]에 [Java] 메뉴가 추가된 것을 확인할 수 있음
- 이 메뉴를 이용해 자바 코드로 변환할 수 있으나 변환된 코드는 실제 구현과는 맞지 않을 수 있음

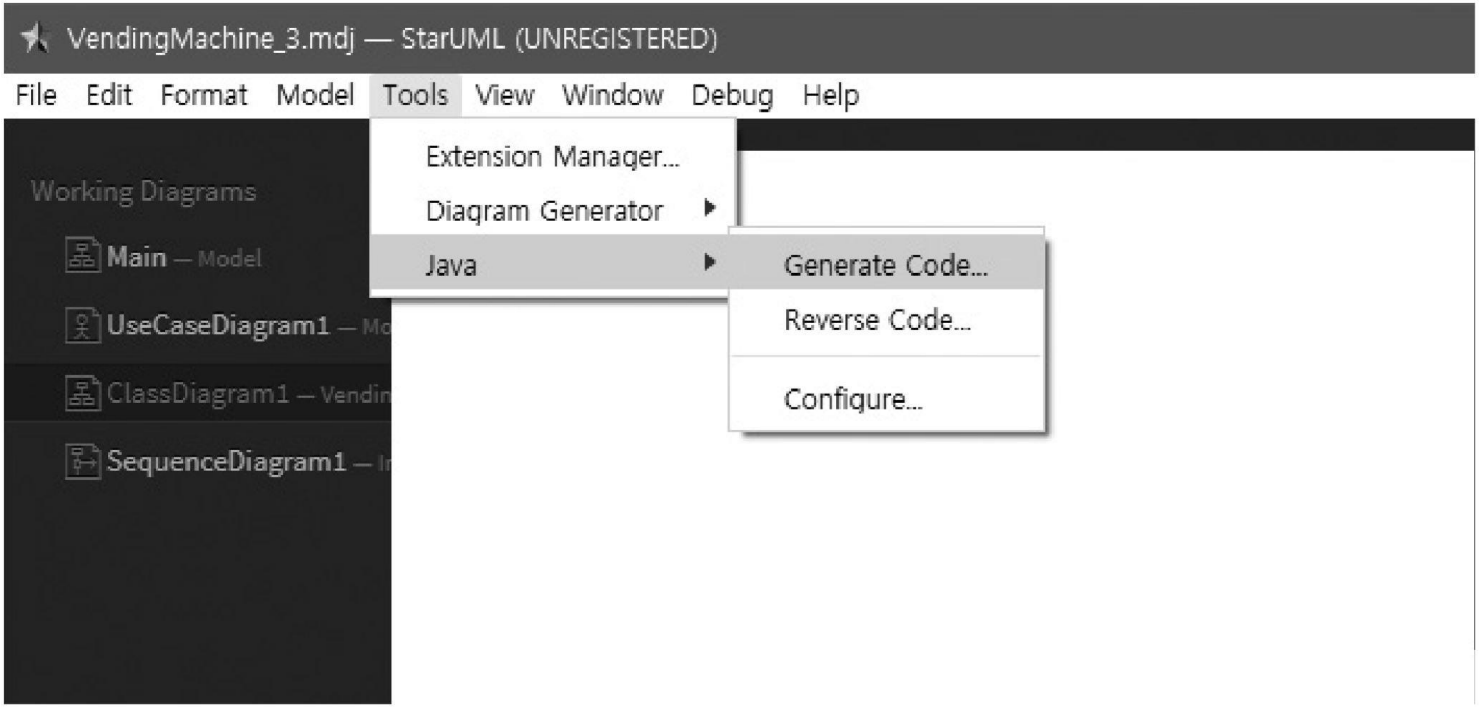


그림 12-26 자바 코드 변환