

소프트웨어공학개론

오유수, Ph.D.

[Ref] “소프트웨어 공학의 소개”, 한혁수, 홍릉과학출판사

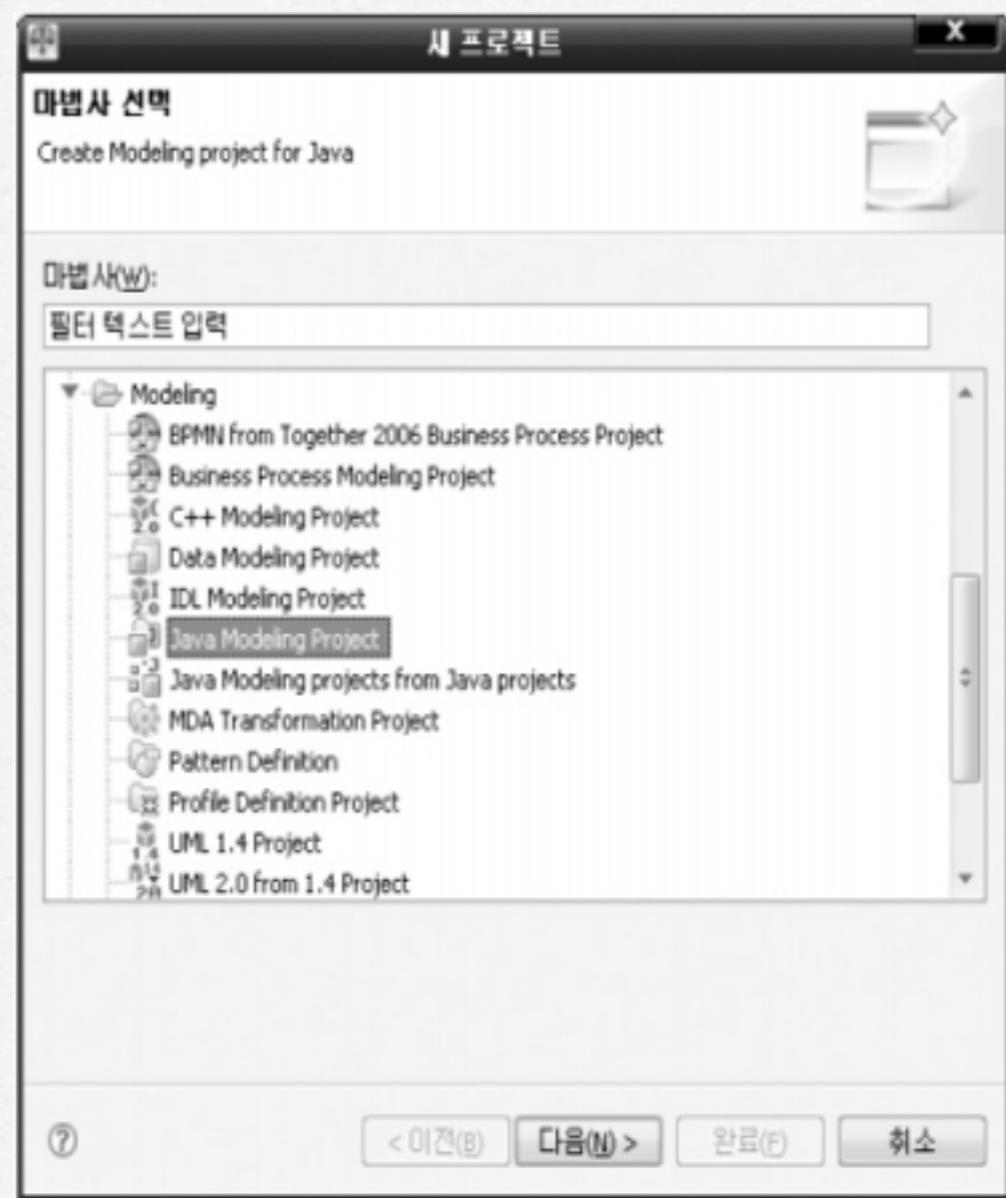
“성공적인 소프트웨어 개발 방법론”, 윤청, 생능출판사

오늘의 강의 목차

- 오늘의 강의 내용
 - UML 다이어그램 예제: 자판기 프로그램
 - 유스케이스 다이어그램
 - 클래스 다이어그램
 - 순차 다이어그램
 - 오늘의 강의 요약
 - 요약, Homework, 다음 강의 소개

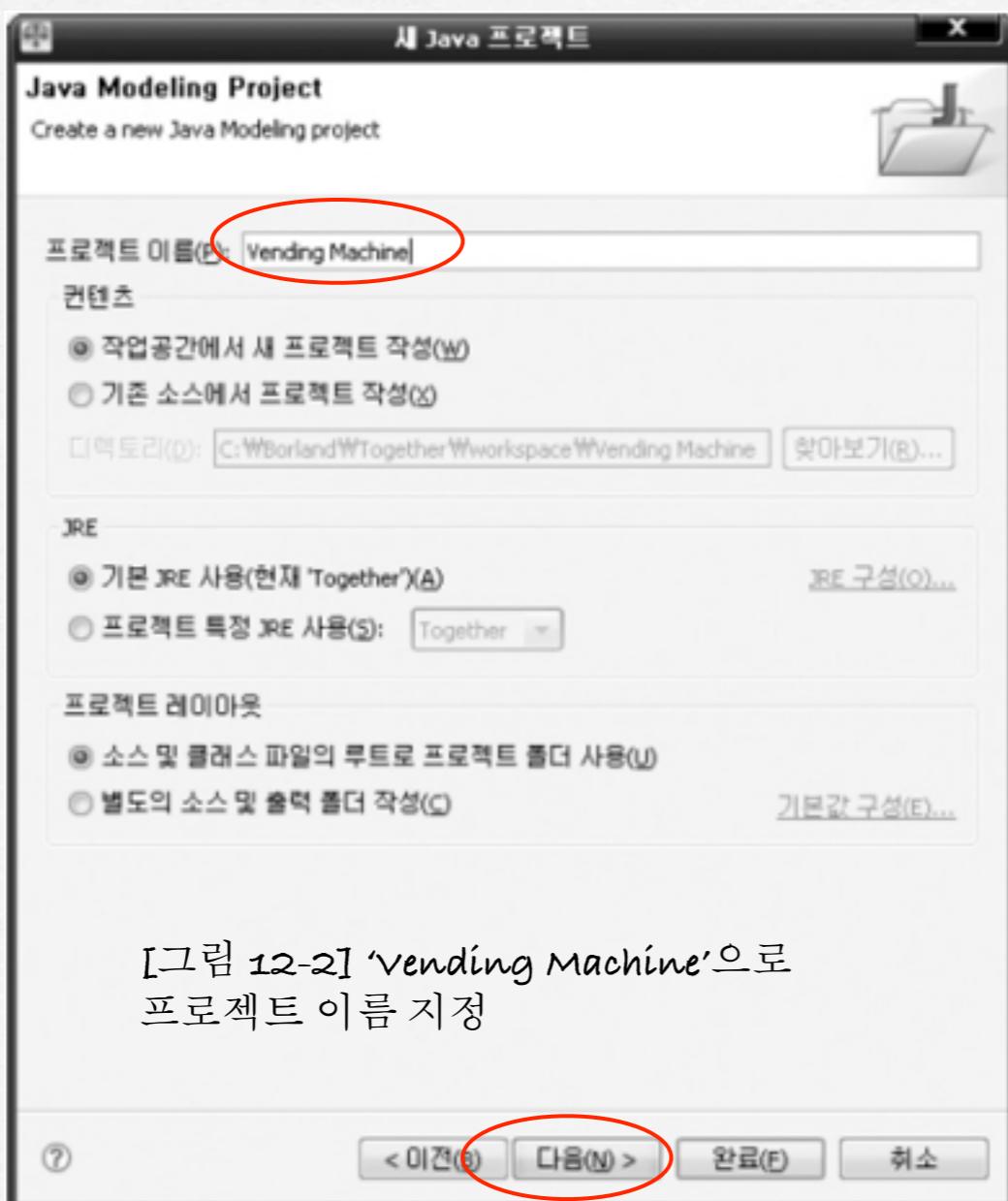
자판기 프로그램 개요

- Situation
 - 개발하고자 하는 시스템은 작은 자판기 시스템
 - 제품명 : 'vendingMachine'
- Task 1
 - 'vendingMachine'이라는 이름을 가진 자바 프로젝트(Java Project)를 투게더(Together)에서 생성
- Activity Steps
 - 투게더의 메인 메뉴에서 '파일 → 새로 작성 → 프로젝트'메뉴를 클릭
 - 마법사 선택에서 Modeling 폴더를 클릭하고, 그 안에 있는 'Java Modeling Project'를 선택한 후 '다음' 버튼을 클릭

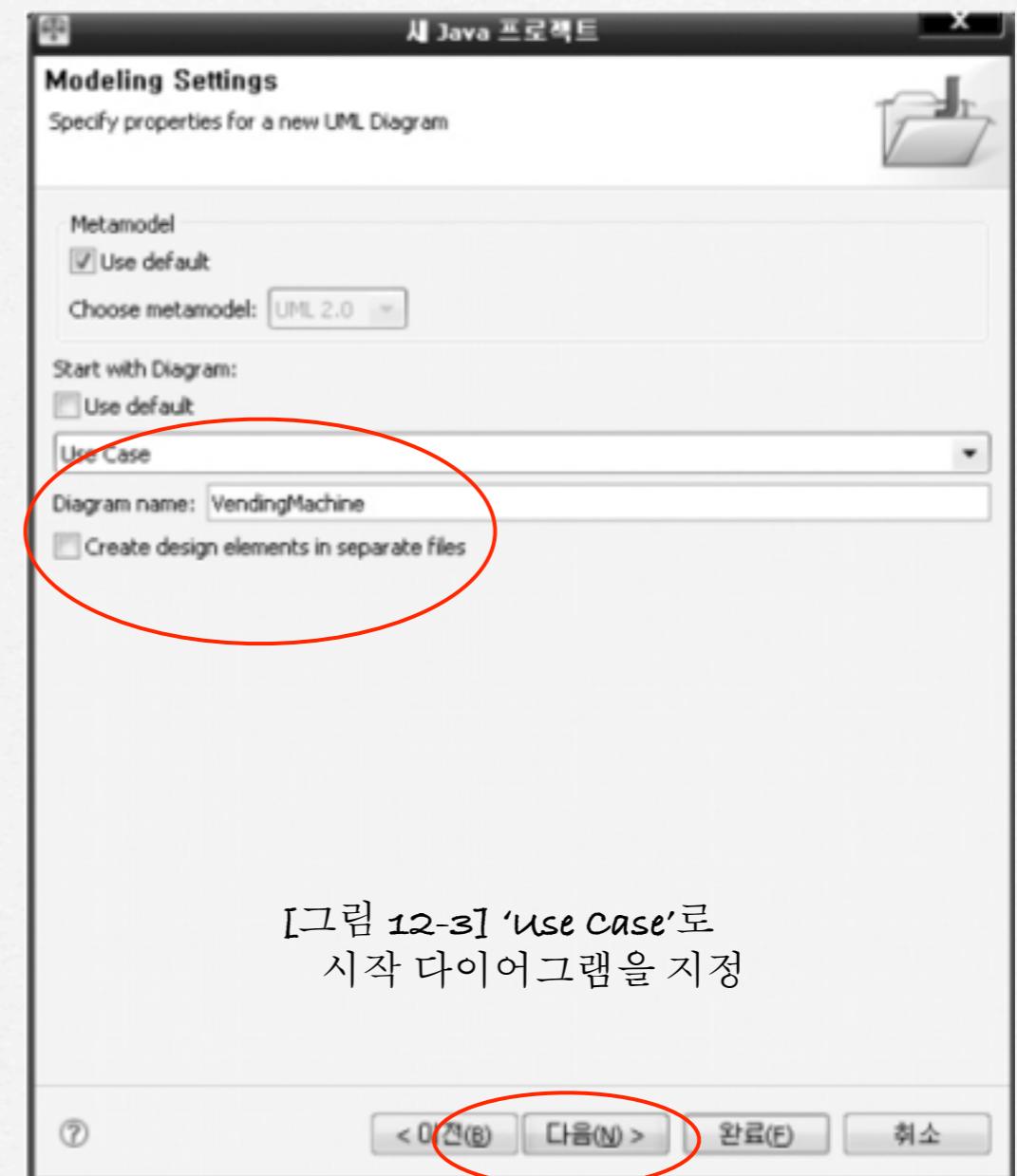


[그림 12-1] Java Modeling Project 선택

- 프로젝트의 이름을 'vendingMachine'이라는 이름으로 지정하고 나머지 부분은 디폴트(default) 상태로 놓는다. '다음'버튼을 클릭
- 기존에 check되어 있는 'Start with Diagram'의 check를 풀 후 시작 다이어그램을 'use case'로 지정, 유스케이스의 이름은 'vendingMachine'이라 한 후 '다음' 버튼 클릭

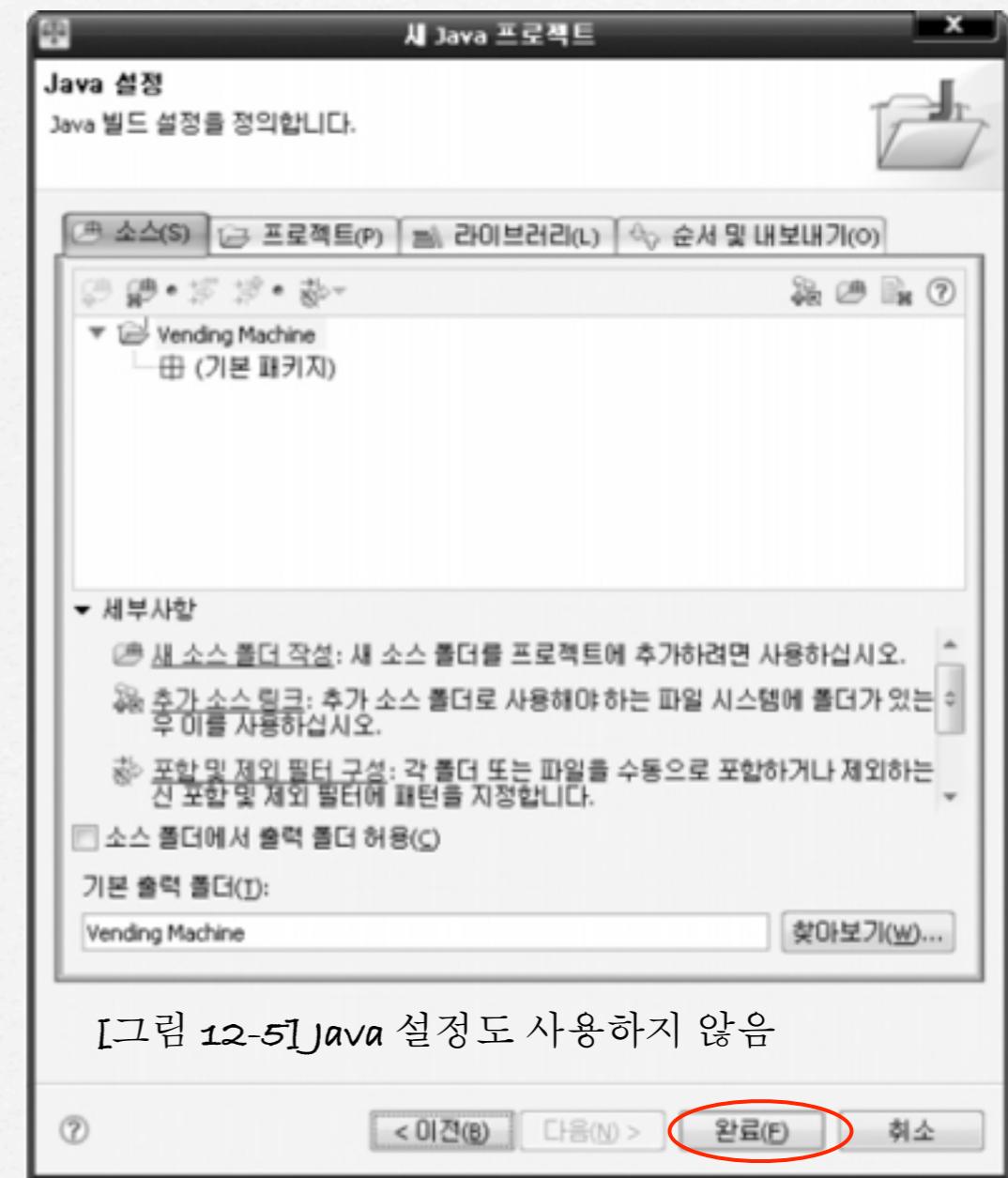


[그림 12-2] 'vending Machine'으로
프로젝트 이름 지정



[그림 12-3] 'use case'로
시작 다이어그램을 지정

- Profiles 부분 : 기존의 다이어그램에 추가적으로 설계할 다른 기능들을 불러와 사용하는 것. 이 장에서는 사용하지 않으므로 바로 '다음'버튼을 클릭
- Java 설정도 마찬가지로 사용하지 않으므로 '완료' 버튼을 클릭



유스케이스 다이어그램 만들기

□ *situation*

- 유스케이스 다이어그램의 구성 요소는
 - 액터, 유스케이스, 시스템 바운더리, 유스케이스 간의 관계 표현

□ *Task 1*

- 문제 기술서를 작성하기

□ *Activity Steps*

- 문제 기술서 작성

자판기 시스템은 사용자가 동전이나 지폐를 투입하면 시스템이 실행된다. 음료수 가격은 500원, 600원, 700원 세 종류이다. 따라서 각각 기준 가격 이상의 돈이 투입되면 자동으로 음료수선택 버튼들에 불이 들어온다. 음료수를 선택하고 남은 잔액은 잔액표시 화면에 표시된다. 사용자는 잔액반환 버튼을 누르면 잔액이 반환된다. 잔액이 음료수 가격 이하의 돈이면 음료수선택 버튼들에 불이 커진다.

유스케이스 다이어그램 만들기

□ Task 2

□ 유스케이스 구성 요소 생성

□ 액터 : user(사용자)

□ 유스케이스 : Input(투입구), DisplayMoney(잔액표시 화면), choice(음료수선택), ReturnMoney(잔액반환)

□ 관계 표현 : user와 Input 간에, user와 choice 간에, user와 ReturnMoney 간에, Input과 DisplayMoney 간에, choice와 DisplayMoney 간에, ReturnMoney와 DisplayMoney 간에 생성

□ 시스템 바운더리 : vendingMachine System

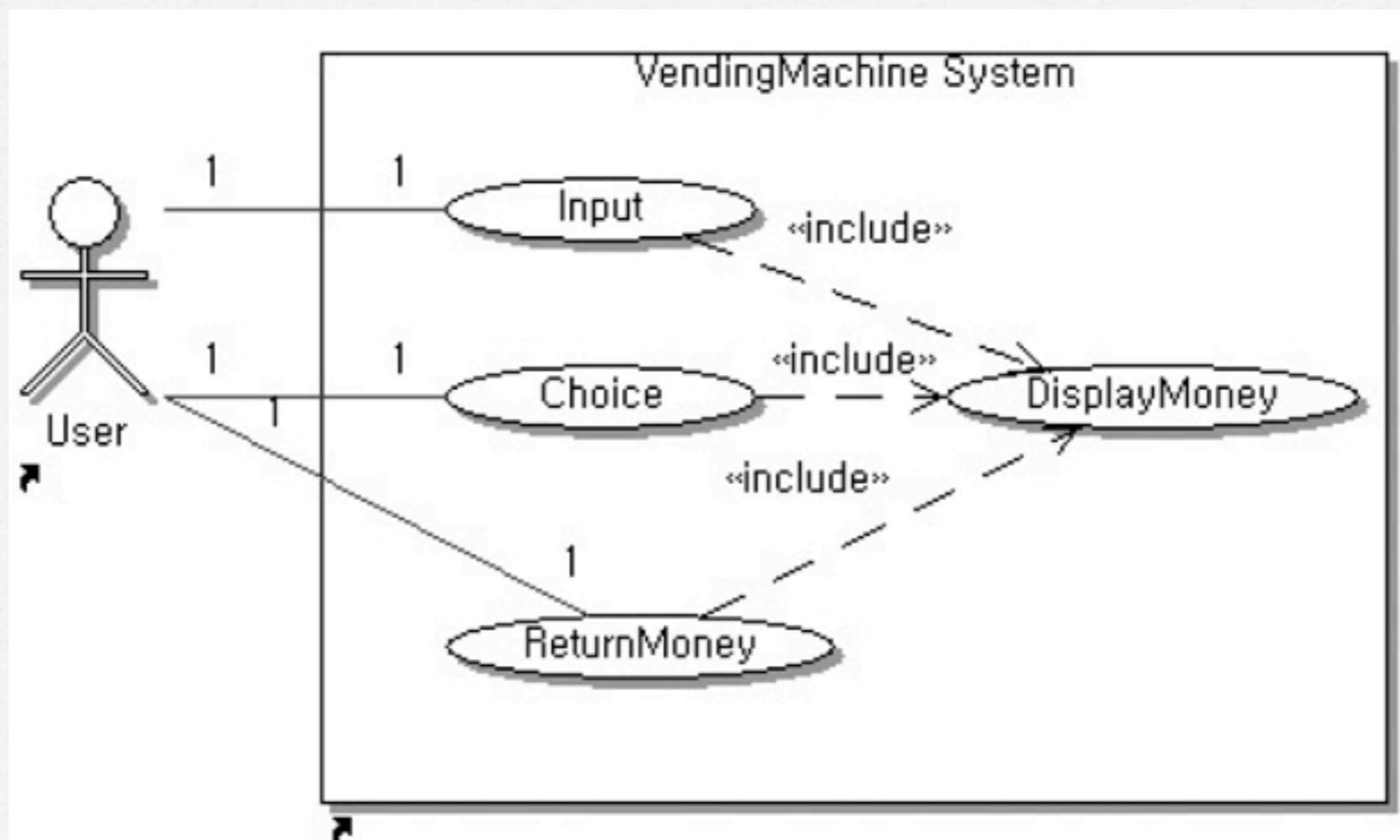
□ Activity Steps

□ (1) 디자이너 페인(Designer Pane)의 왼쪽의 다이어그램 툴바(Diagram Toolbar)에서 액터 버튼을 클릭하고 다이어그램을 클릭한다. 액터의 이름을 'User'로 변경

□ (2) 다이어그램 툴바에서 시스템 바운더리 버튼을 클릭하고 다이어그램을 클릭한다. 이름은 'vendingMachine System'으로 바꾼다.

□ (3) 5개의 유스케이스를 생성하기 위해 다이어그램 툴바의 유스케이스 버튼을 클릭하고 'vendingMachine System'을 클릭한다. 이름은 'Input', 'Choice', 'ReturnMoney', 'DisplayMoney'이라고 한다.

- (4) 'User'와 'Input'간에, 'User'와 'Choice'간에, 'User'와 'ReturnMoney'간에 연결 link를 생성하기 위해 툴바에서 Association 버튼을 선택하고 각각을 드래그하여 연결
- (5) include link를 생성하기 위해 'Input' 유스케이스를 선택한 상태에서 툴바의 includes 버튼을 클릭하고 'DisplayMoney' 유스케이스로 드래그
- (6) 같은 방법으로 'Choice'와 'DisplayMoney' 간에, 'ReturnMoney'와 'DisplayMoney' 간에 include link를 생성



[그림 12-6] 포함(include)관계 생성

패키지 만들기

□ Situation

- 3개의 패키지를 추가하여 이후에 만들어질 구성 요소(소스, 다이어그램 등)들을 쉽게 관리할 수 있도록 한다.
- 'vendingMachine', 'Data', 'UserInterface'

□ Task 1

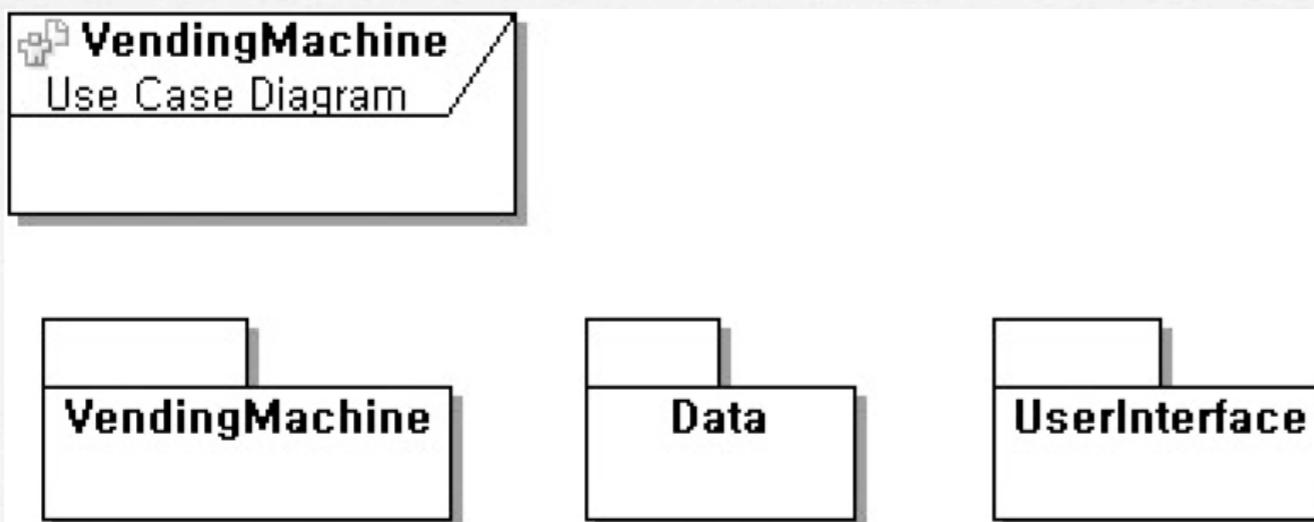
- 패키지와 패키지 다이어그램 생성하기

□ Activity Steps

- (1) 투게더의 Model Navigator에서 'vendingMachine'이라 되어 있는 'Default Diagram'을 더블 클릭
- (2) 투게더에서 패키지 아이콘(Package icon)을 클릭하고, 다이어그램(Diagram)을 클릭하면 새로운 패키지가 생성된다. 패키지의 이름을 'vendingMachine'으로 바꾼다.

□ Task 2

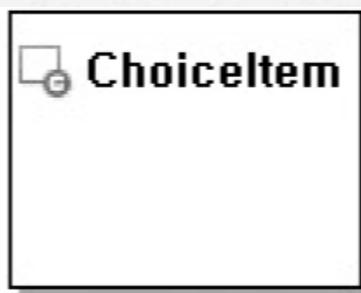
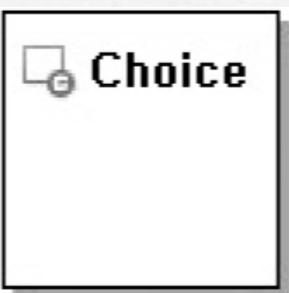
- 위와 같은 방법으로 'Data'와 'UserInterface'라는 2개의 패키지를 생성
- Activity Steps
 - Control 키를 누른 상태에서, 디자인 툴바에서 패키지 아이콘을 클릭
 - 다이어그램 바탕을 클릭하면 패키지가 생성되며 이름을 'Data'으로 바꾼다.
 - 같은 방식으로 패키지 'UserInterface'를 만든다.
 - Multiple Mode를 해지하지 위해서 툴바에서 다른 툴(icon)을 선택하거나 Esc 키를 누르면 된다.
 - 패키지를 열고 싶으면, 다이어그램에서 패키지를 더블클릭하거나, 패키지의 빠른 메뉴에서 'Open'을 선택



[그림 12-7] 패키지 생성

클래스 다이어그램 만들기

- Situation
 - 'vendingMachine'에 3개의 클래스를 만든다.
 - 'choice(선택처리)', 'choiceltem(선택음료수)', 'ItemExplain(음료수설명)'
- Task 1
 - 클래스 다이어그램을 생성하고 투게더가 자동으로 생성하여 주는 코드 보기
- Activity Steps
 - (1) 디자이너 페인(Designer Pane)에서 'vendingMachine' 패키지 열기
 - (2) 클래스 버튼을 클릭하고 다이어그램을 선택하여 새로운 클래스를 생성한다. 클래스의 이름을 'choice'로 바꾼다.
 - (3) 위와 같은 방법으로 'Choiceltem'과 'ItemExplain'이라는 2개의 클래스를 더 생성
 - (4) 투게더는 초기 클래스 소스 코드를 자동으로 생성하여 준다. 만약 에디터 페인(Editor Pane)이 보이지 않는다면, 해당 클래스를 더블클릭하면 해당 클래스의 소스를 볼 수 있다.



[그림 12-8] 클래스 생성

클래스에 attribute와 operation 추가하기

□ Situation

- 클래스들을 좀 더 상세하게 채워보도록 하자. 'Choice' 클래스에서부터 시작

□ Task 1

- 디자이너 폐인과 빠른 메뉴를 사용하여 어트리뷰트(Attribute) 더하기

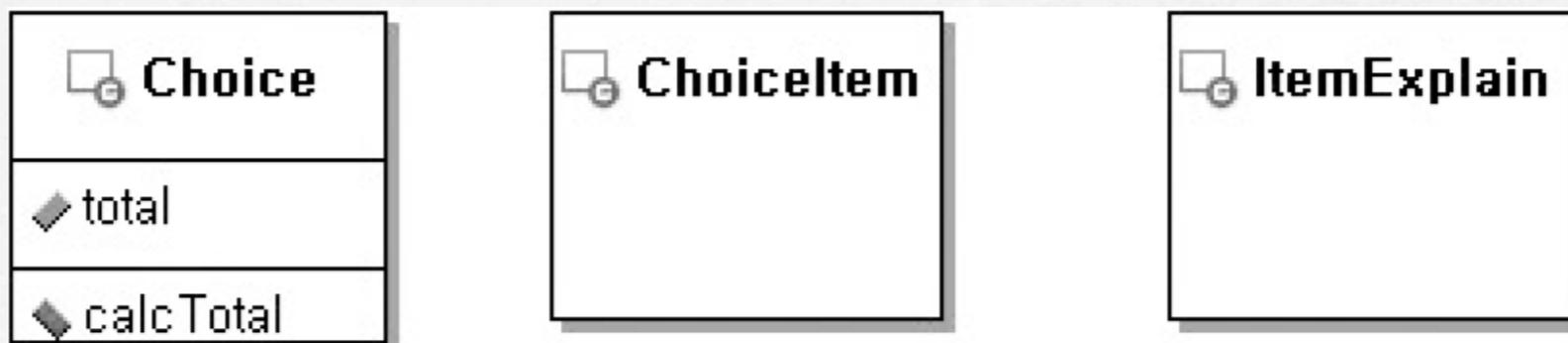
□ Activity Steps

- (1) 'VendingMachine' 디어그램 내의 'Choice' 클래스를 선택
- (2) 클래스에서 오른쪽 마우스를 클릭하여 빠른 메뉴를 띄운 후 'New → Attribute(단축키 Insert 누른 후 선택)'를 선택
- (3) 디자이너 폐인에서 attribute1을 더블클릭하여 'total:int'로 바꾼다.

클래스에 attribute와 operation 추가하기

□ Task 2

- 디자이너 폐인과 빠른 메뉴를 사용하여 오페레이션(Operation) 더하기
- Activity Steps
 - (1) 디자이너 폐인에서 작업한 'Choice' 클래스를 선택
 - (2) 클래스의 빠른 메뉴를 통하여 'New → Operation(단축키 Insert 누른 후 선택)'을 사용하여 새로운 오페레이션을 생성
 - (3) 디자이너 폐인에서 생성된 오페레이션을 선택하여 'calcTotal:int'로 바꾼다.



[그림 12-9] 'Choice' 클래스에 어트리뷰트와 오페레이션 추가

클래스에 attribute와 operation 추가하기

□ Task 3

- 에디터 페인(Editor pane)과 코드(Code)에서 어트리뷰트를 추가하기
- Activity Steps
 - (1) 'Choice' 클래스를 더블 클릭
 - (2) choice.java의 소스를 보여주는 텍스트 에디터(Text Editor)를 이용하여 아래의 어트리뷰트를 추가한 후 '저장(단축키 Ctrl+S)'을 클릭한다. 투게터는 소스파일이 바뀌면 바로 업데이트를 해준다.

```
private String cashRegister;
```

- (3) 디자이너 페인을 클릭하면, 텍스트 에디터에서 추가한 작업이 반영되는 것을 확인할 수 있다. (* cashRegister은 ... 상품 및 가격을 등록하기 위한 멤버변수)
- (4) 위와 같은 방법으로 choice.java 파일에 'public double calcTex() {return 0;}' 오퍼레이션을 추가 (* calcTex()은 .. 잔액을 계산하기 위한 멤버함수)

클래스간의 관계 설정하기

□ situation

- 어소시에이션(Association)은 클래스 다이어그램에서 연관관계를 나타낸다.
- 'vendingMachine' 다이어그램 내의 클래스 간에 관계 설정
- 관계들의 상세사항(cardinality, navigability, type)들도 함께 살펴보자.

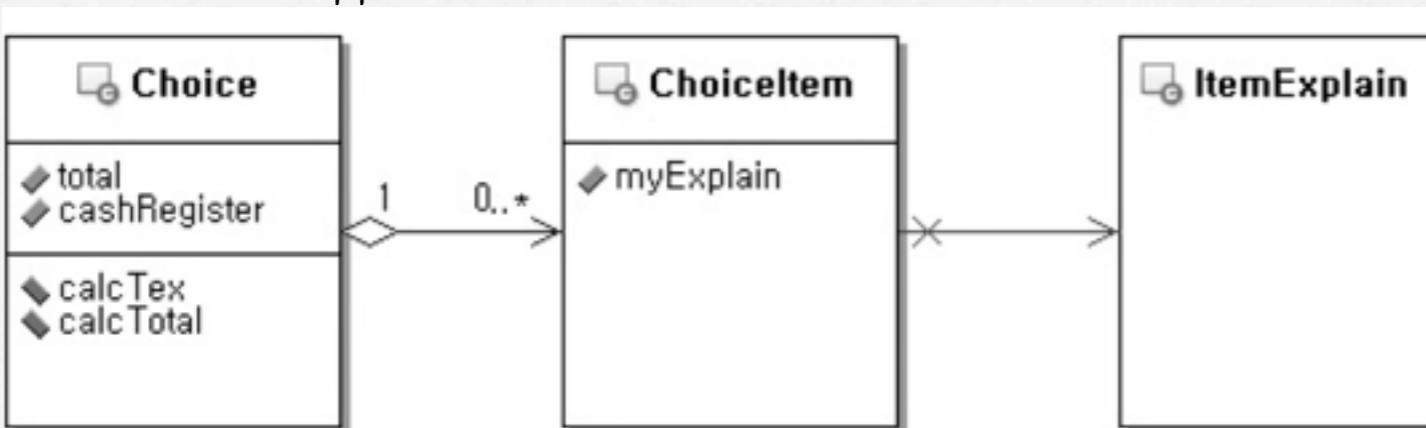
□ Task 1

- links의 프로퍼티 대화상자를 사용하여 어소시에이션의 프로퍼티를 생성하고 바꿔본다.
- Diagram Options를 사용하여 여러분들에게 보이는 links의 모습을 바꿔보도록 한다.

□ Activity Steps

- (1) 디자이너 폐인에서 'vendingMachine' 패키지를 선택하고 더블클릭
- (2) 'ChoiceItem' 클래스를 선택하고, 툴바에서 어소시에이션 아이콘을 클릭하고 'ItemExplain'으로 마우스로 드래그하여 'ChoiceItem'에서 'ItemExplain'까지 어소시에이션 관계를 생성
- (3) 에디터 폐인에서 'ChoiceItem' 클래스의 'linkItemExplain'라는 클래스 멤버의 이름을 'myExplain'으로 바꾼다.

- (4) 위와 같은 방법으로 'Choice'에서 'Choiceltem'으로 어소시에이션 관계 생성
- (5) 방향성을 가질 수 있도록 설정하기 위해 해당 어소시에이션을 클릭하고 아래 특성 부분에서 **Directed** 체크 박스를 체크
- (6) 4번에서 그린 어소시에이션의 빠른 메뉴에서 **Link type → Aggregation**을 클릭하여 **Aggregation** 관계로 바꾼다. 투게더에서는 어소시에이션의 빠른 메뉴에서 **Association**, **Aggregation**, **Composition** 관계를 선택하여 표현
- (7) 어소시에이션의 빠른 메뉴에는 링크 **cardinalities**를 표시할 수 있는 기능이 있으며, 크게 **client cardinalities**와 **Supplier cardinalities** 두 가지의 **ardinalities**를 표현할 수 있다. 'Choice'에서 'Choiceltem'으로의 어소시에이션의 **cardinality**를 설정하기 위해 'Choice'쪽에서 어소시에이션 라인 가까이에서 마우스 오른쪽 버튼을 클릭한 후 어소시에이션의 빠른 메뉴가 뜨면, **client cardinalities**에서 '**1**'을 선택
- (8) 위와 같은 방법으로 'Choiceltem' 클래스 쪽에서 어소시에이션에서 빠른 메뉴를 실행하고 **Supplier cardinalities**에서 '**0...***'을 선택



[그림 12-10] 클래스의 연관관계 생성

클래스에 constructor와 property 추가하기

□ Situation

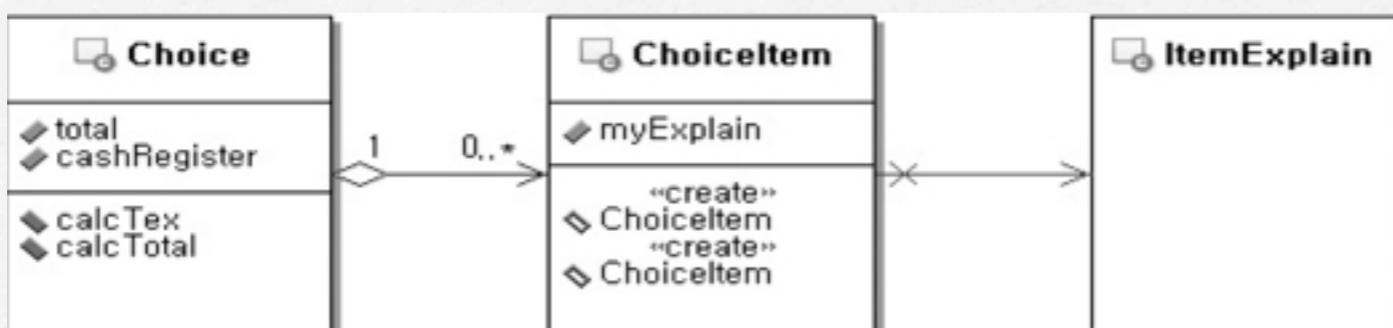
- 'Choiceltem'과 'ItemExplain' 클래스에 몇 개의 컨스트럭터(Constructor, 생성자)들을 생성하고, 그리고 그들의 프로퍼티(Property)를 바꿔보도록 하자.

□ Task 1

- 빠른 메뉴를 사용하여 컨스트럭터 생성하기

□ Activity Steps

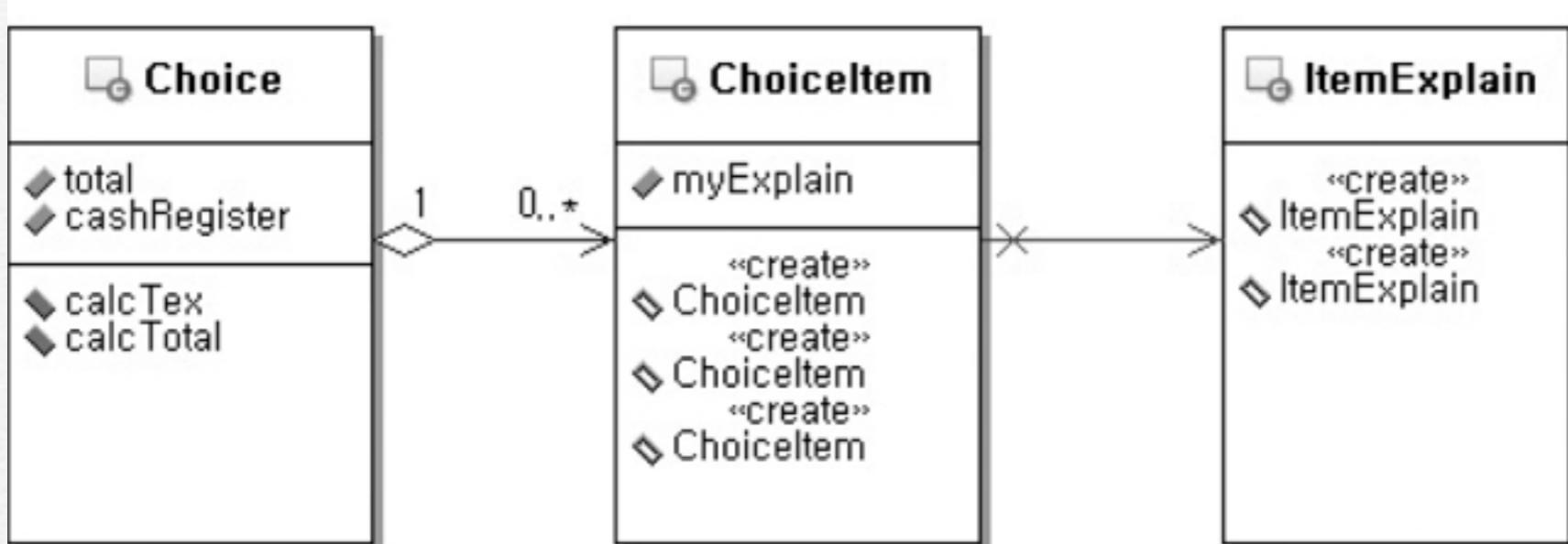
- (1) 'vendingMachine' 다이어그램 내의 'Choiceltem'을 선택하고 클래스의 빠른 메뉴에서 'New → Constructor'를 사용하여 Default Constructor를 만든다.
- (2) 'Choiceltem' 클래스의 빠른 메뉴에서 'New → Constructor'를 또 한 번 선택하여 하나의 파라미터(Parameter)를 가지고 있는 두 번째 컨스트럭터를 만든다. 투게더는 두 번째 생성되는 컨스트럭터는 하나의 파라미터를 갖는 컨스트럭터로 자동 생성
- (3) 에디터 페인에서 두 번째 생성한 컨스트럭터의 파라미터를 'int i01'에서 'ItemExplain desc'로 바꿔준다.



[그림 12-11] 'Choiceltem' 클래스의
컨스트럭터 생성

클래스에 constructor와 property 추가하기

- Task 2
 - 만약 하나의 컨스트럭터가 존재한다면, 복사하여 다른 컨스트럭터 만들기
- Activity Steps
 - (1) 'ChoiceItem' 클래스에서 'ItemExplain'을 파라미터로 가지고 있는 컨스트럭터를 선택하고 빠른 메뉴를 실행하여 'Copy(단축키 Ctrl+C)'를 선택
 - (2) 'ChoiceItem'을 활성화시킨 뒤에 클래스의 빠른 메뉴에서 'Paste(단축키 Ctrl+V)'를 선택한다. 그리고 나면 에디터 폐인에 세 번째의 컨스트럭터가 생성될 것이고, 디폴트로 정의된 두 번째의 파라미터인 'int i01'을 에디터 폐인에서 'int quantity'로 바꿔준다.
 - (3) 'ChoiceItem(desc:ItemExplain)'을 'ItemExplain' 클래스로 복사한다. 투게더는 자동적으로 컨스트럭터라는 속성에 따라서 컨스트럭터의 이름을 바꿔준다. 생성된 'ItemExplain'의 컨스트럭터를 파라미터를 더블클릭한 후 에디터 폐인에서 'String name'으로 바꿔준다.
 - (4) 디자이너 폐인 내의 클래스 다이어그램의 'ItemExplain(name:String)'으로 정의되어 있는 컨스트럭터를 복사한 후 Paste 한다.
 - (5) 'ItemExplain' 클래스에 다른 컨스트럭터가 더해질 것이다. 그러면 새롭게 생성된 컨스트럭터의 'int i02'파라미터를 'price:double'로 바꿔준다.



[그림 12-12] 컨스트럭터 복사

□ Task 3

- 빠른 메뉴를 사용하여 프로퍼티(Property)를 생성하고 getter와 setter를 만들기 위해서 자바빈 (Java Bean) 기능을 사용하기

□ Activity Steps

- (1) 'Choiceltem' 클래스의 빠른 메뉴를 실행시킨 다음 'New → Property(단축키 ctrl+B)'를 선택 한다. 'property1:int'인 프로퍼티의 기본값을 'quantity:int'로 바꿔준다. 프로퍼티를 생성하고 나면, 투게더가 자동으로 getter와 setter를 생성해주는 것을 에디터 폐인에서 확인할 수 있다.
- (2) 'ItemExplain' 클래스를 선택하여 빠른 메뉴를 실행시킨 다음 'New → property'를 선택 한다. 그리고 기본적인 'property:int'을 'name:String'으로 바꿔준다.
- (3) 위와 똑같은 방법으로 'ItemExplain' 클래스에 'price:double'인 프로퍼티를 추가

□ Task 4

- 'ItemExplain' 클래스 컴파일하기

□ Activity Steps

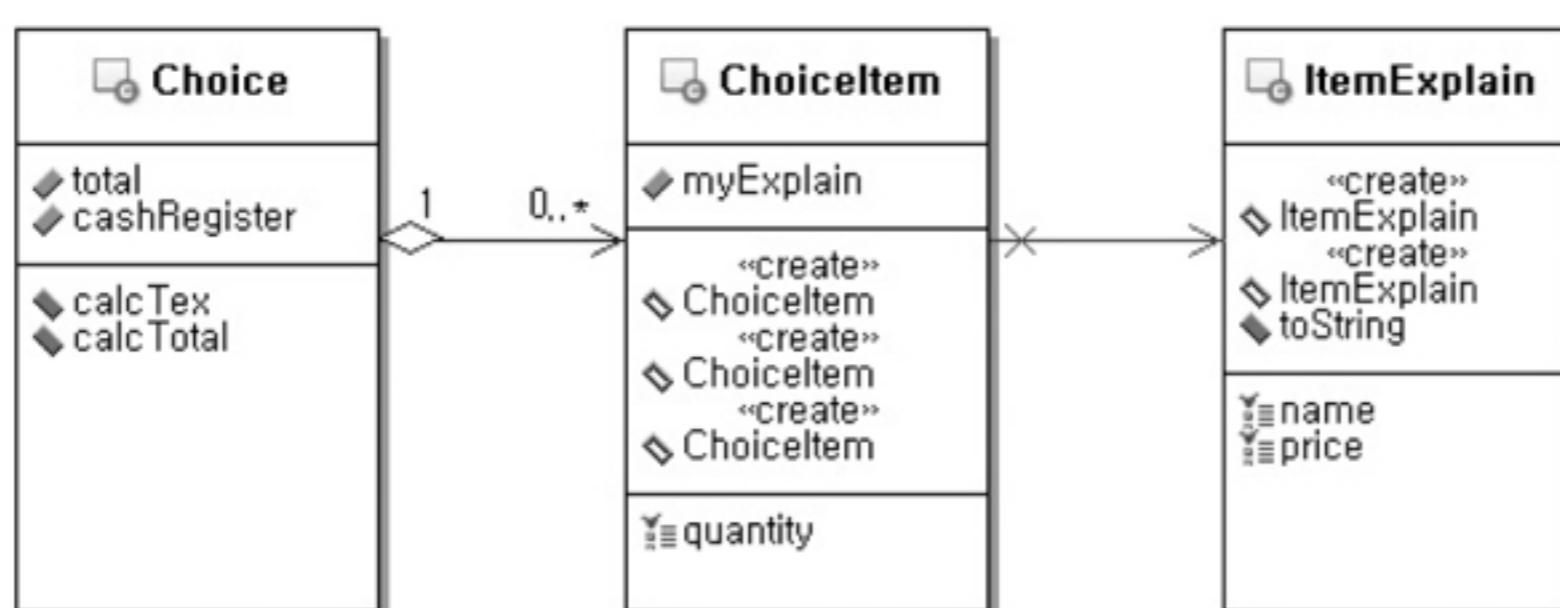
- (1) 에디터 폐인에서 'ItemExplain' 클래스의 `ItemExplain(name:String)` 컨스트럭터를 선택하고 이 컨스트럭터의 바디(Body) 부분에 아래 부분을 추가

```
this(name, 0.0);
```

- (2) `ItemExplain(name:String, price:double)` 컨스트럭터를 선택한다. 그리고 Editor에서 컨스트럭터의 바디 부분에 아래 부분을 추가

```
this.name = name;  
this.price = price;
```

- (3) 'ItemExplain' 클래스에 새로운 메서드(Method)를 추가



```
public String toString(){  
    return name + price + "원";  
}
```

그림 12-13] 프로퍼티와 메서드추가

순차 다이어그램을 이용하여 클래스 모델 다듬기

□ Situation

- 'vendingMachine' 내에 3개의 클래스를 구성하였다. 이번에는 그 클래스 간의 상호관계를 표현하고 다른 클래스들을 새롭게 추가하도록 하겠다.

□ Task 1

- 순차 다이어그램 생성하기

□ Activity Steps

- (1) 디자이너 페인에서 'vendingMachine' 다이어그램을 선택하고 메인 메뉴에서 '파일→새로 작성 → Diagram'을 선택한다. 뒤이어 나오는 다이어그램 설정 타입(Type)에서 'Sequence Diagram'을 선택
- (2) 다이어그램의 이름을 'startvendingMachine'로 입력한 후 완료 버튼을 클릭한다. 투게 더 2006에서는 기존의 1.4 버전에서는 없었던 Intercation 영역을 넣었으며, 이 영역 안에서만 순차 다이어그램을 설계할 수 있게끔 되어 있다.
- (3) 작업을 하는 동안 순차 다이어그램에서 메시지의 번호가 보이는 것이 훨씬 편리하다. 이것이 기본적으로 설정되어 있다.

[그림 12-14] 순차 다이어그램 생성

- Task 2

- 현재의 순차 다이어그램에서 툴바의 오브젝트 버튼을 사용하도록 하겠다.

- Activity Steps

- (1) 다이어그램 툴바에서 오브젝트 아이콘 버튼을 사용하여 'aCustomer'라는 이름의 오브젝트를 만든다.
 - (2) 'inputMoney', 'currentTransaction', 'anItem'이라는 이름의 3개의 오브젝트를 추가로 만든다.
 - (3) 단축 버튼 'Ctrl + K'를 누르거나 다이어그램 빠른 메뉴의 'Layout → All'을 사용하여 다이어그램을 깔끔하게 정렬한다.

□ Task 3

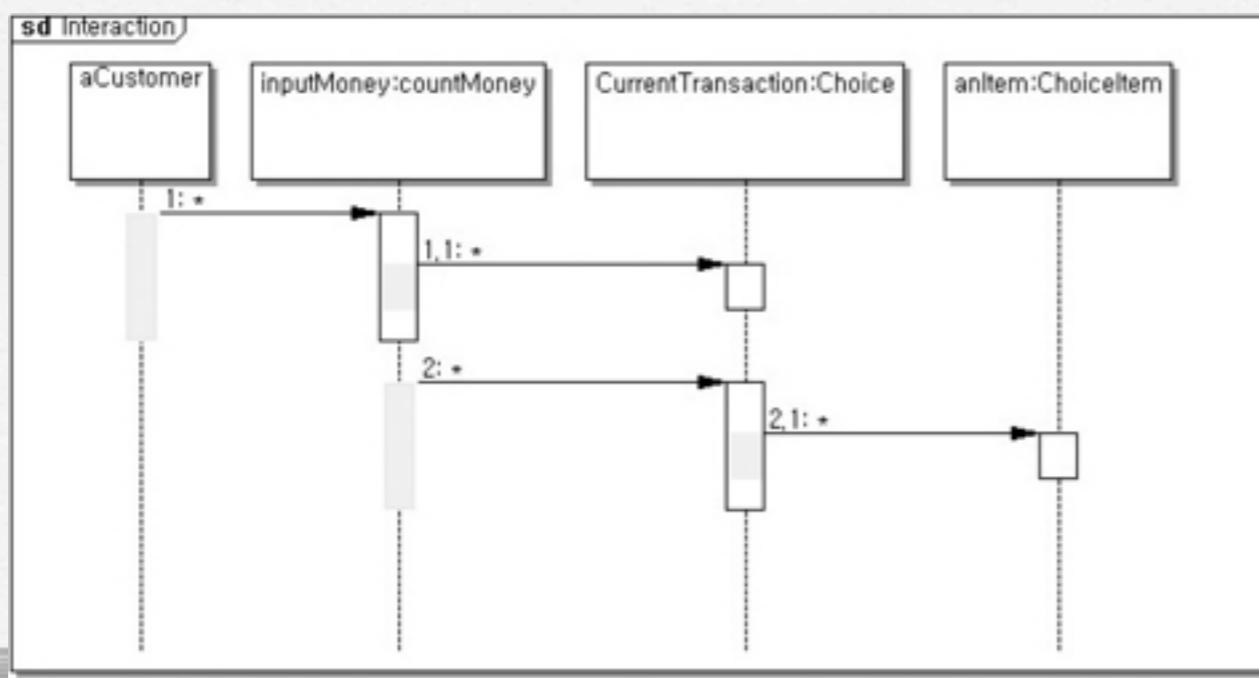
- 빠른 메뉴를 사용하여 각각의 오브젝트(클래스)의 타입 정의하기

□ Activity Steps

- (1) 'inputMoney' 오브젝트를 선택하여 빠른 메뉴를 띄운 후, 'New → Type → class'를 선택한다. 디폴트로 정의되어 있는 'class'라는 클래스의 이름을 'countMoney'로 바꿔준다. 이후 완료 버튼을 클릭한다.
- (2) 'vendingMachine' 다이어그램에서 'countMoney'를 선택하고, 'choice'타입의 'currentTransaction'이라 부르는 새로운 어트리뷰트를 추가한다.
- (3) 'startvendingMachine' 순차 다이어그램에서 'currentTransaction' 오브젝트를 선택하고 오른쪽 마우스를 클릭하여 오브젝트의 빠른 메뉴를 띄운 후, 'Choose Type'를 선택한다. 'currentTransaction' 오브젝트를 위하여 'choice'클래스를 선택한다.
- (4) 위와 똑같은 방법으로 'anItem' 오브젝트를 위하여 'ChoiceItem' 클래스를 선택한다.

□ Task 4

- 툴바에서 메시지(Message) 버튼을 선택하여 메시지 또는 컨스트럭터를 만들어 보도록 한다.
- Activity Steps
 - (1) 'startvendingMachine' 다이어그램을 선택한다.
 - (2) 툴바에서 'Message' 아이콘을 클릭하고 'aCustomer'에서 'inputMoney' 오브젝트로 메시지를 드래그한다. 이 메시지의 번호는 1번으로 표시된다.
 - (3) 위와 같은 방법으로 'inputMoney'에서 'currentTransaction'로 2개의 메시지를 생성한다. 1.1의 메시지는 'inputMoney' 오브젝트의 흰색 activation bar에서 시작한다. 2의 메시지도 위와 같은 방법으로 생성한다.
 - (4) 위와 같은 방법으로 'currentTransaction'에서 'anItem'으로 가는 번호 2.1의 메시지를 생성한다.



[그림 12-15] 메시지 생성

□ Task 5

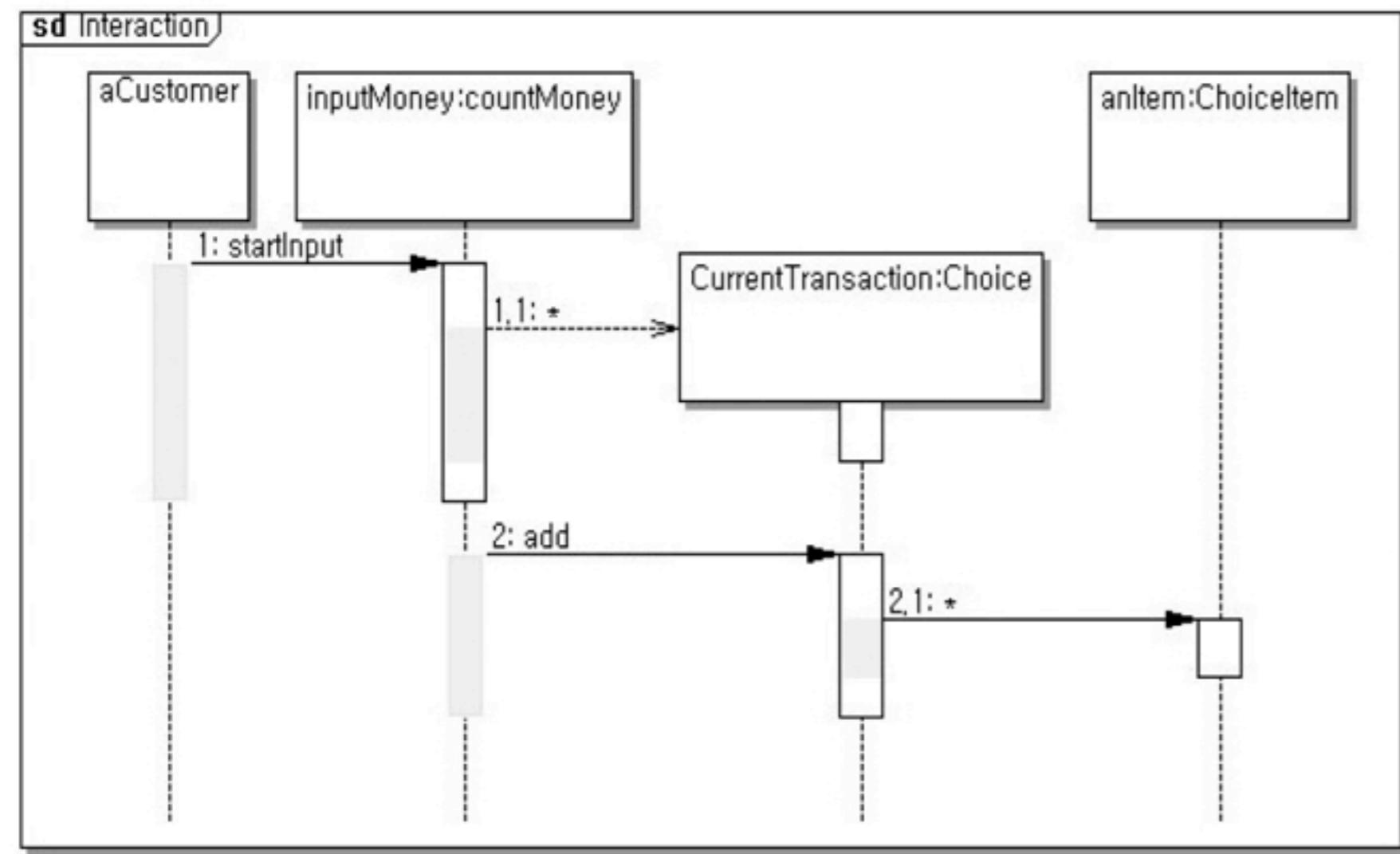
- 만약 순차 다이어그램에서 메시지에 해당하는 새로운 메서드를 생성하고자 한다면, 메시지의 빠른 메뉴에서 'New → Operation' 또는 'New → Constructor'를 사용하면 된다.
- 투게더에서 생성한 오퍼레이션들은 Explorer pane의 Model 탭에서 각 클래스의 노드를 확장하면 확인할 수 있다.

□ Activity Steps

- 'aCustomer'와 'inputMoney' 사이에 위치하는 메시지의 빠른 메뉴에서 'New → Operation'을 선택한다.
- 'inputMoney'와 'currentTransaction' 사이에 위치하는 메시지 중 위(1.1)의 메시지의 빠른 메뉴를 실행시키고 'New → Constructor'를 선택한다.
- 'inputMoney'와 'currentTransaction' 사이에 위치하는 메시지 중 아래(2)의 메시지의 빠른 메뉴를 실행시키고 'New → Operation'을 선택한다.

□ Task 6

- 순차 다이어그램에서 메시지와 연결되어 있는 메서드의 이름들을 바꾸는 작업은 'Property Inspector'를 사용하여 바꿀 수 있다. 단지 생성된 메서드의 이름을 바꾸기 위한 것이다.
- Activity Steps
 - 'aCustomer'와 'inputMoney' 사이의 메시지 빠른 메뉴에서 'Properties'를 선택한다. 투게더의 화면 왼쪽에 위치한 Editor 아래쪽에 특성이 나타날 것이다. 특성 내에 있는 Properties에서 'Signature'에 값을 넣는다.
 - 기본값으로 된 오퍼레이션의 이름인 '*'을 'startInput'로 바꾸고 'Enter'를 클릭한다.
 - 'inputMoney'와 'currentTransaction'에서 위 (1.1)에 위치한 메시지를 선택한다. 그리고 특성 내에 있는 Properties에서 이 메시지의 Properties 내에 있는 'Creation' 박스를 체크한다. 그리고 순차 다이어그램에서 'currentTransaction' 오브젝트의 위치가 바뀌는 것을 확인한다.
 - 'inputMoney'와 'currentTransaction'에서 아래 (2)에 위치한 메시지를 선택한다. 특성 내에 있는 Properties에서 Signature의 값을 'add():void'로 넣고 'Enter'를 클릭한다. 그 후 확인을 위해 나타나는 대화상자에서 'Save as Text'을 선택한다.
 - 디자이너 페인에서 빠른 메뉴를 실행한 후, 'Layout → All'을 사용하여 다이어그램을 정렬한다.



[그림 12-16] 메시지의 'Properties'에서 메서드 이름 바꾸기

□ Task ↗

- 존재하는 메서드를 메시지에 할당하기

□ Activity Steps

- 'currentTransaction'과 'anItem' 사이에 메시지의 빠른 메뉴에서 'Choose signature'을 선택한다. 'Choiceltem(desc:itemExplain,quantity:int)'를 선택한다.

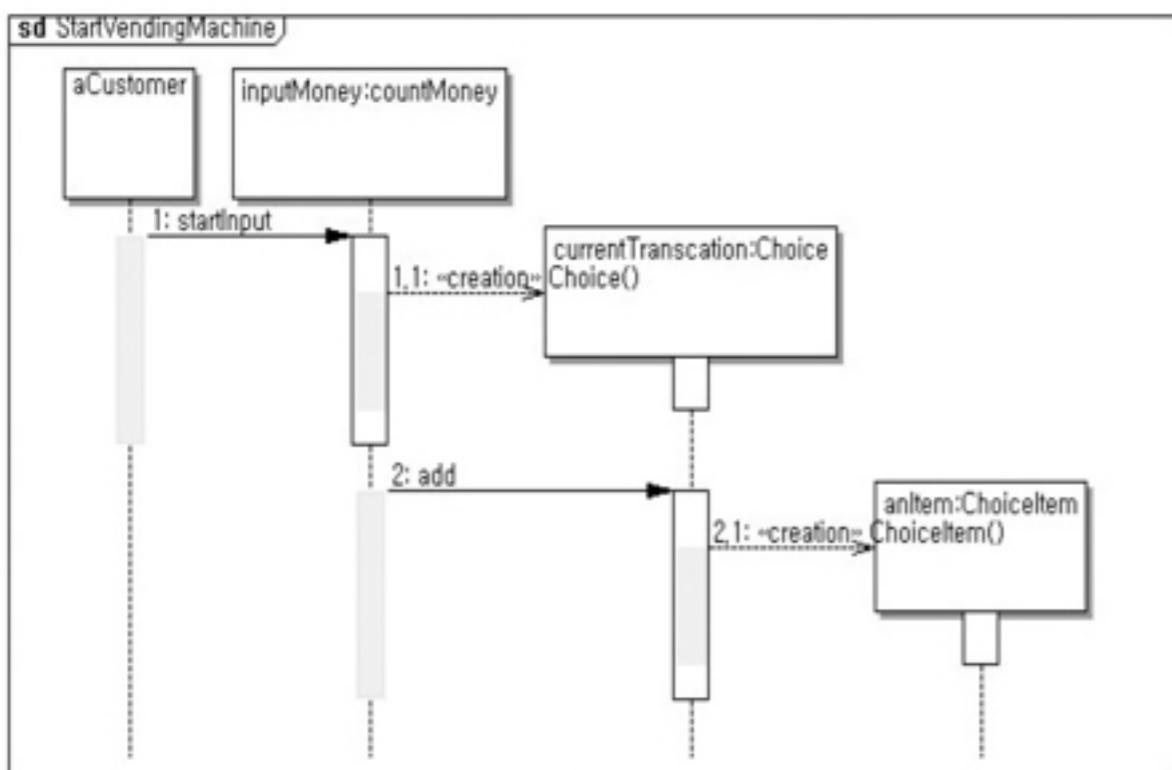
□ Task 8

- 메시지의 *Property Inspector*의 *Operation* 탭을 사용하여 파라미터 바꾸기. 단, 이 *Operation* 탭에서 사용되는 파라미터의 타입은 자바 포맷(<type>, <name>)을 따른다.
- Activity Steps
 - (1) 'acustomer'와 'inputMoney' 사이에 있는 'startInput' 오퍼레이션을 선택한다. 특성 내에 있는 *Properties*에서 어트리뷰트란에 아래와 같은 값을 입력한다.

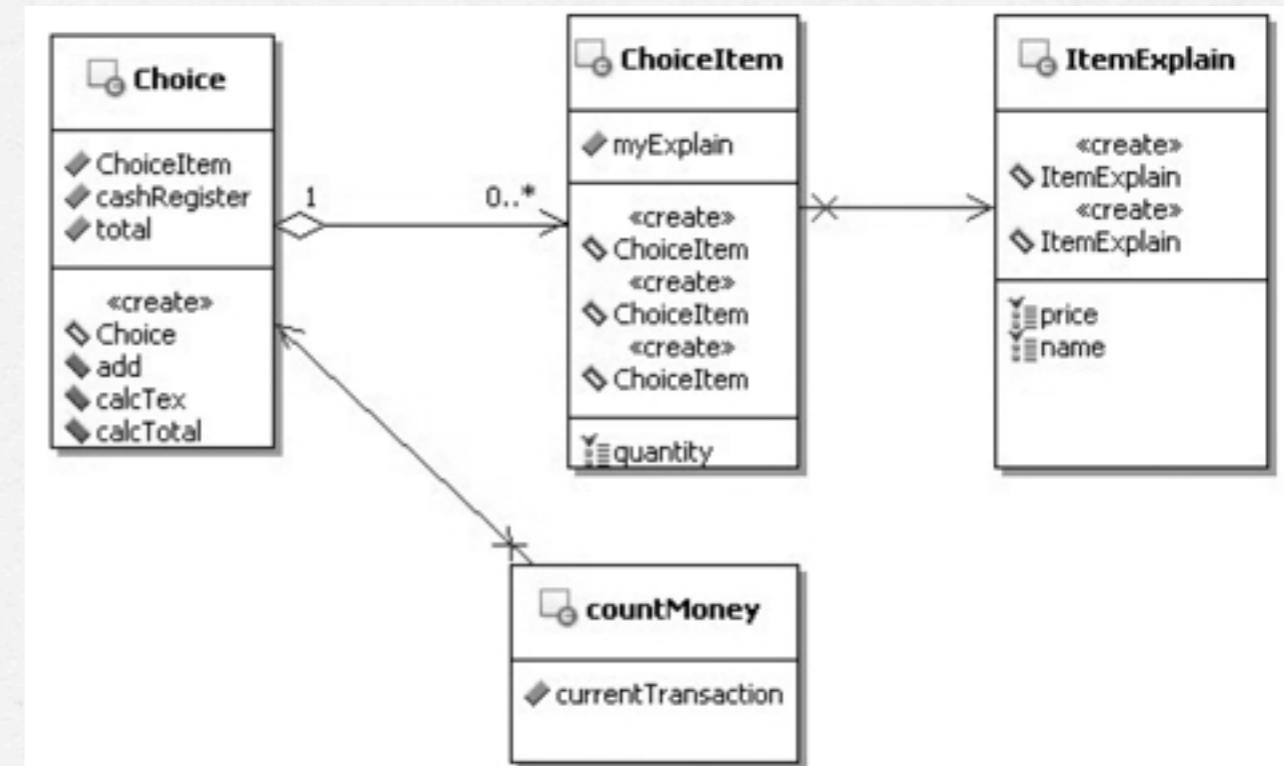
ItemExplain desc, int quantity
 - (2) 'inputMoney'와 'currentTransaction' 사이에 있는 'add' 오퍼레이션의 빠른 메뉴에서 'Properties'를 선택한다. 특성 내에 있는 *Properties*에서 어트리뷰트란에 아래와 같은 값을 입력한다.

ItemExplain desc, int quantity
 - (3) 'inputMoney'와 'currentTransaction' 사이의 1.1 메시지를 선택한 후, 특성 내에 있는 *Properties* 내에 있는 *Return value*에 'currentTransaction'을 입력하고 'Enter'를 클릭한다.
 - (4) 지금 한 작업은 다음 스텝에서 코드로 *Generate*될 때, 'countMoney' 클래스 내의 인스턴스 변수와 함께 새로운 'choice' 오브젝트로 연결될 것이다.
 - (5) 디자이너 페인에서 빠른 메뉴를 실행한 후, 'Layout → All'을 사용하여 디어그램을 정렬한다.

- Task 9
- 새로 추가된 오퍼레이션이 반영된 Code Generate하기
- Activity Steps
 - (1) 순차 다이어그램의 회색 바탕에서 빠른 메뉴를 띄우고, 'Generate implementation' 메뉴를 선택한다.
 - (2) 각각의 메서드가 어떻게 구현되었는지 에디터 페인에서 살펴보자. 그리고 'vendingMachine' 클래스 다이어그램에서도 확인해 보자.



[그림 12-17] 추가된 오퍼레이션이 반영된 코드 생성하기



[그림 12-18] 구현된 메서드 확인하기

클래스간의 상속관계 만들기 및 인터페이스 추가하기

□ Situation

- 'countMoney'를 포함해서 클래스 다이어그램의 범위를 확장해보자. 먼저 'countMoney'클래스에 좀 더 일반적인 관계들을 추가해보자.

□ Task 1

- 'Ichoice'라는 이름의 인터페이스 생성하기. 'countMoney'클래스는 인터페이스의 구현 클래스 만들기

□ Activity Steps

- (1) 'vendingMachine' 다이어그램을 연다.
- (2) 'countMoney' 클래스에 빠른 메뉴의 'New → Operation'를 실행하여 아래의 두 오퍼레이션을 추가한다.

```
addItem(desc:ItemExplain, quantity:int):void  
calculateTotal():String
```

- (3) 'countMoney' 클래스의 빠른 메뉴에서 '리펙터 → 인터페이스 추출'메뉴를 실행한다.

- (4) 'Extract interface' 대화상자에서 아래와 같이 설정한다.

Interface Name : Ichoice

Select members : startInput(), addItem(), calculateTotal()

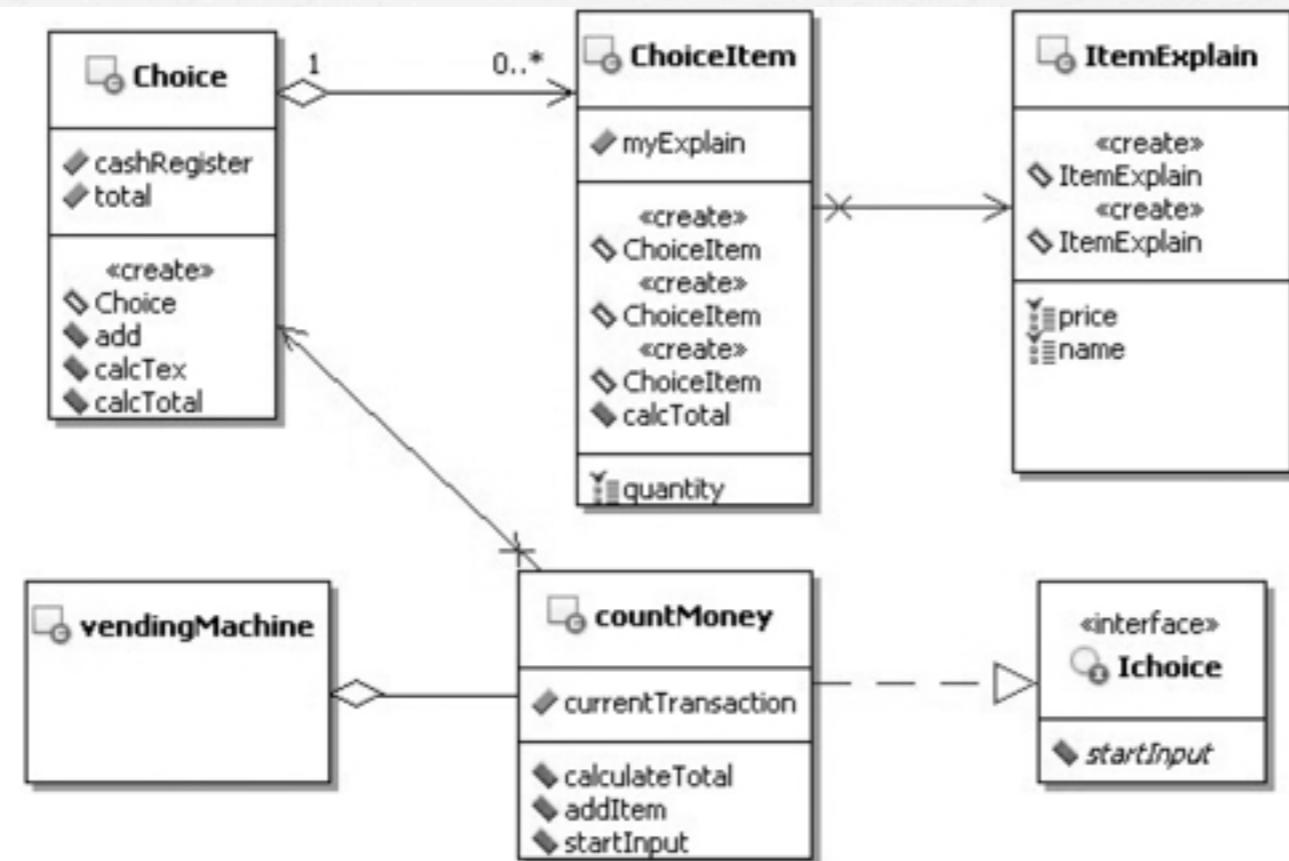
모두 Action을 abstract로 설정

- (5) '미리보기' 버튼을 눌러 변경되는 내용을 확인한 후, '완료'버튼을 클릭한다.
- (6) 'vendingMachine' 패키지에 'Ichoice'라는 인터페이스가 'countMoney' 클래스간에 'implementation' 관계가 표시되면서 생성된 것을 확인할 수 있다.

□ Task 2

- 새로운 클래스 생성 및 인터페이스와 다른 클래스와의 새로운 관계 생성하기
- Activity Steps

- (1) 다이어그램에서 'vendingMachine'이라는 새로운 클래스를 생성한다.
- (2) 툴바에서 'Association' 아이콘(화살표가 없는 실선의 아이콘)을 클릭하여 'vendingMachine'에서 'countMoney'로 어소시에이션을 그려준다. 그리고 생성된 어소시에이션의 빠른 메뉴에서 'Aggregation'으로 체크한다.



[그림 12-19] 'vendingMachine'에서 'countMoney'로 연관관계 생성

템플릿 패턴을 정의하고 사용하기

□ Situation

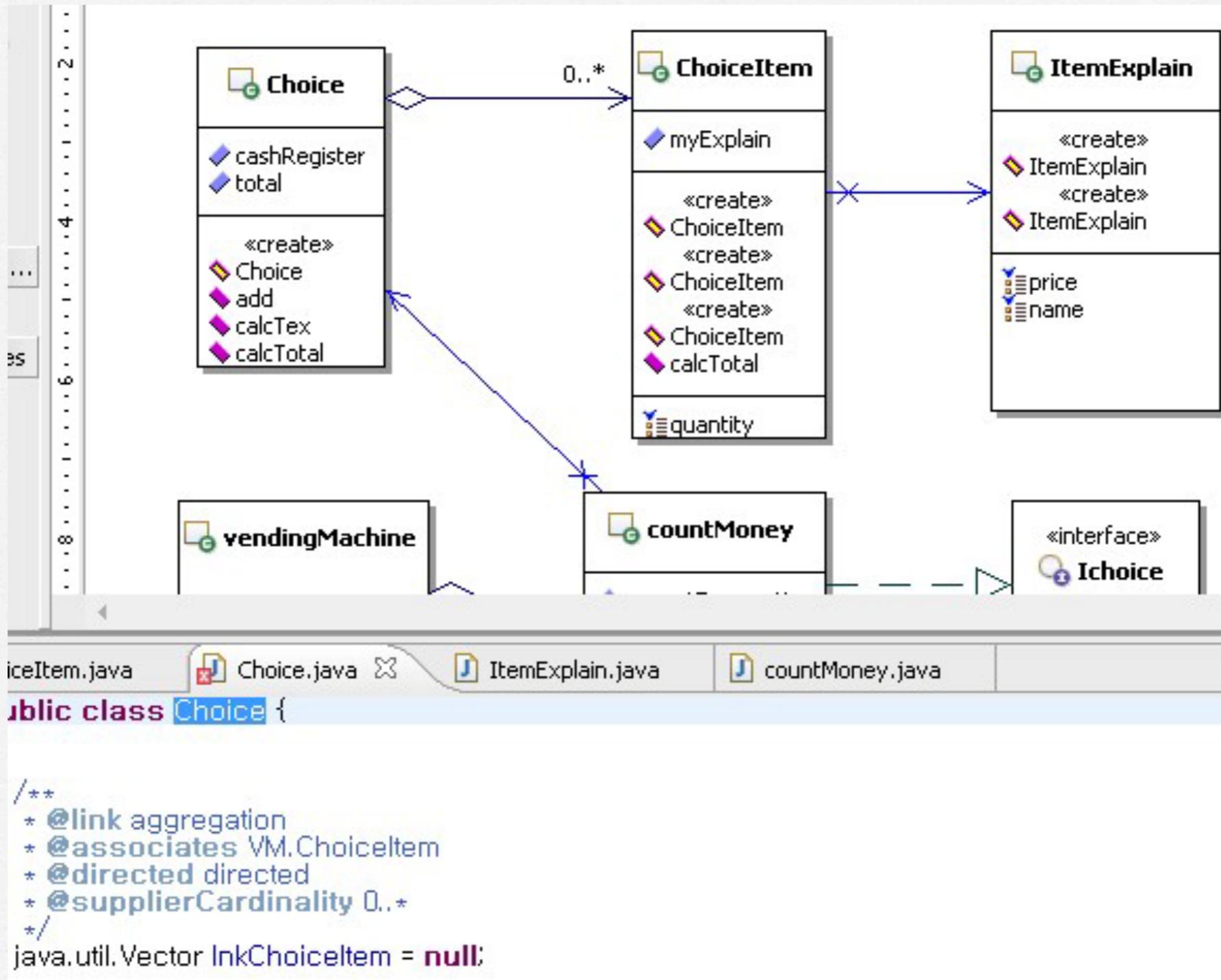
- 지금까지 프로젝트를 구성하면서 몇몇의 Aggregation 관계를 보았다. 하지만 이것이 구현될 때 어떻게 구현되는지에 대해서는 나타내지 않는다는 것을 알 것이다. 투게더에서는 클래스의 구현에서 명시되어 사용될 수 있는 템플릿 세트(Template Set)를 가지고 있다.

□ Task 1

- 이 작업의 목적은 다음과 같다.
- choice와 choiceltem 간의 Aggregation 관계에 패턴 적용시키기

□ Activity Steps

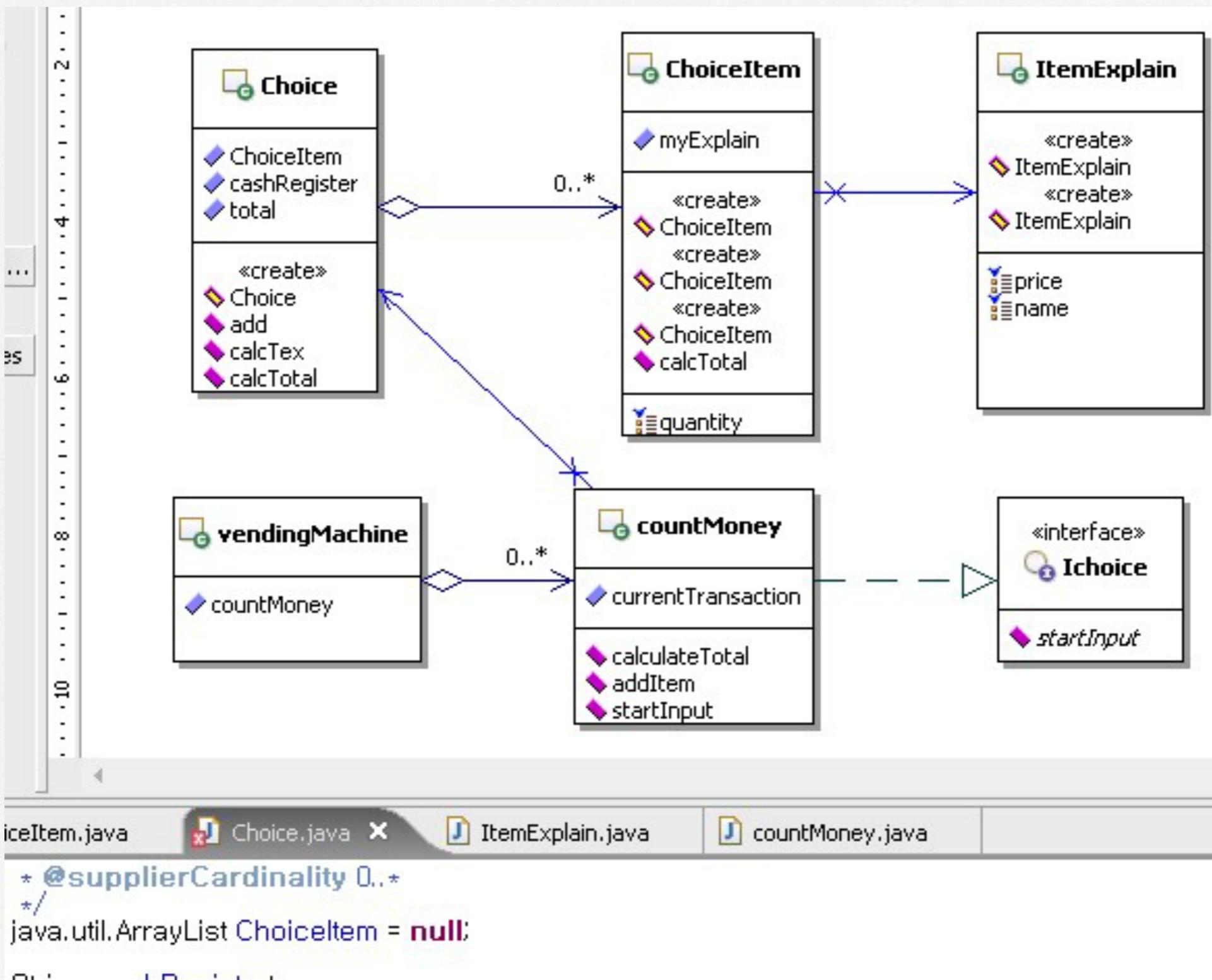
- (1) 'Choice'와 'Choiceltem' 사이의 'Aggregation'을 선택하고 빠른 메뉴에서 'Apply Template'을 클릭한다. 다음에 대화상자에서 'java.util.Collections → Aggregations' 노드의 'Aggregation as vector'를 선택한다.
- (2) choice 클래스를 더블 클릭하고 코드가 보이는 것을 확인한다.



[그림 12-20] 템플릿 패턴 적용

□ Task 2

- 나중을 위해 아래와 같은 작업을 완성해야 한다.
 - 클래스 멤버의 초기화를 위한 새로운 ArrayListpattern을 생성하기
 - 'vendingMachine'에서 'countMoney'의 Aggregation과 'ChoiceTransaction'에서 'ChoiceItem'의 Aggregation에 새로운 패턴 적용시키기
- Activity Steps
 - 'vendingMachine'과 'countMoney' 사이의 'Aggregation'을 선택하고 빠른 메뉴를 띄운 후, 'Apply Template'을 클릭한다. 'java.util.Collections → Aggregations' 노드로부터 'Aggregation as ArrayList'를 선택한다.
 - 기본값들을 바꿔보도록 하겠다. 'Field Name' 박스에서 'lnkcountMoney'를 'countMoney'로, 'Link destination' 박스에서 'problemdomain.countMoney'를 'countMoney'로 바꾼 후 'Finish'를 클릭한다.
 - Designer pane 내에서 'ArrayList countMoney'라는 새로운 attribute를 찾아본다.
 - 'Choice'와 'ChoiceItem' 사이의 Aggregation의 빠른 메뉴에서 'Apply Template'을 클릭한다. 'java.util.Collections → Aggregations' 노드로부터 'Aggregation as ArrayList'를 선택한다.
 - 대화상자에서 'Name'에서 'lnkchoiceitem'을 'ChoiceItem'으로 바꾼 후 'Finish'.

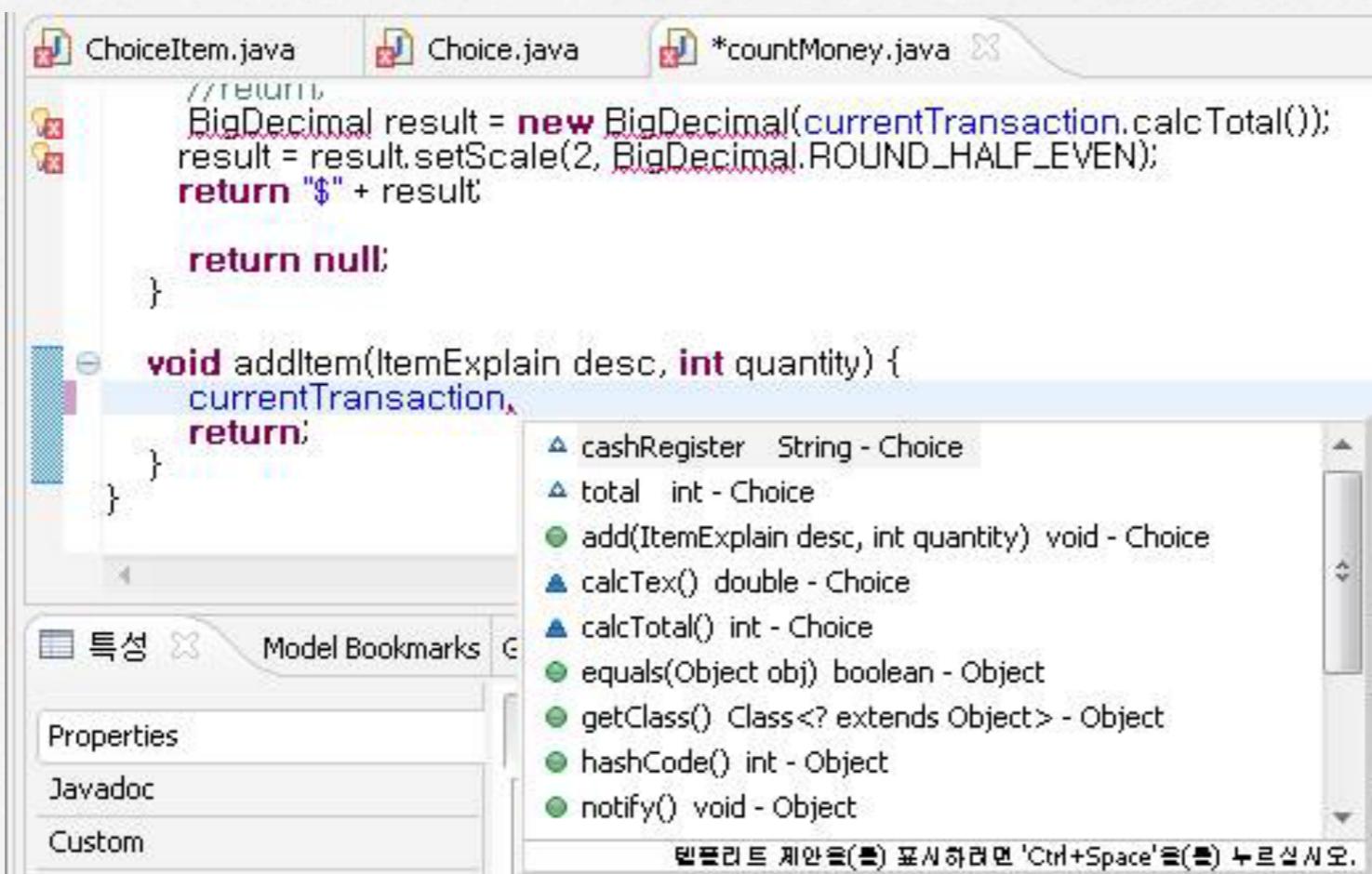


[그림 12-21] 연관관계에서 패턴 적용하기

Code Sense 사용하기

□ Situation

- 이번에는 소스 코드 부분을 자세히 보도록 하자. 우선 텍스트 에디터(Text Editor)를 우리가 사용하는 환경에 맞게 여러 가지 기능들을 더하고 바꿔보도록 하자. 예를 들어 투게더에서는 에디터에서 작업 시에 코드(code)의 완성도를 높여 줄 수 있는 팝업창을 함께 띄워주는 기능이 있다.



[그림 12-22] 코드 센스의 예

□ Task 1

- 다양한 방법으로 코드 센스 사용하기
- Activity Steps
 - (1) 'ChoiceItem' 클래스에서 'ChoiceItem(itemExplain desc, int quantity)' 컨структор를 클릭한다. 그리고 에디터 페인에서 아래의 부분을 컨структор에 추가

```
myExplain = desc;
this.quantity = quantity;
```
 - (2) 'ChoiceItem(itemExplain desc)' 컨структор의 바디부분에 아래 부분을 추가

```
this(desc,1);
```
 - (3) 'ChoiceItem' 클래스에 새로운 메서드를 추가하고 아래와 같이 코드를 추가

```
public double calcTotal(){
    double total = 0;
    total = myExplain.getPrice() * quantity;
    return total;}
```

- (4) 'Choice' 클래스의 'add(itemExplain desc, int quantity)' 메서드를 클릭하고 바디 부분을 완성하도록 한다. 코드의 첫 줄에 주석은 Aggregation 관계를 맺어 주고 생성된 주석이다.

```
// message #2.1 to anItem:problemdomain.ChoiceItem  
ChoiceItem anItem = new ChoiceItem(desc, quantity);  
listChoiceItem.add(anItem);
```

- (5) 'countMoney' 클래스의 'calculateTotal' 메서드를 클릭한다. 그리고 바디 부분을 완성한다.

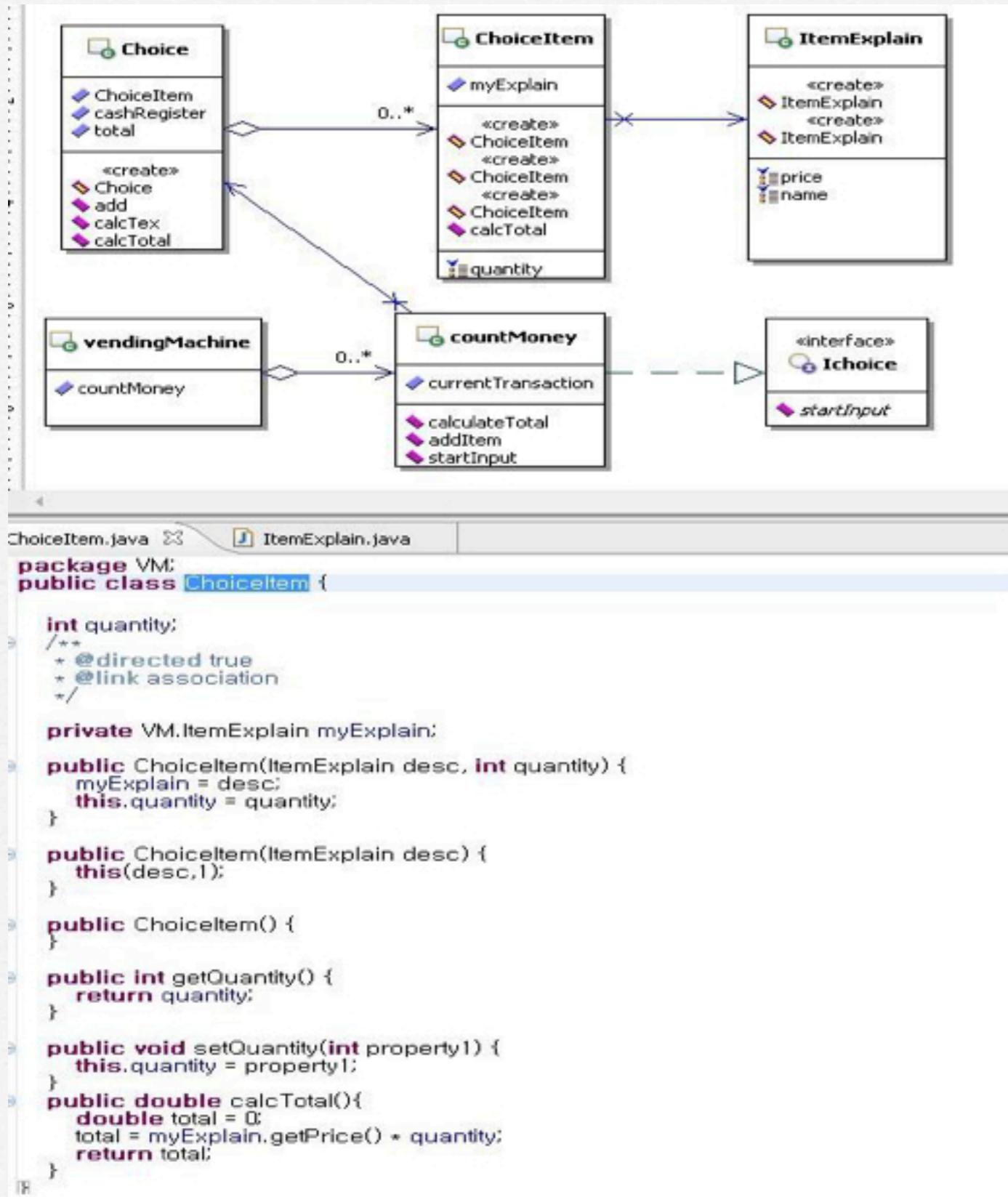
```
BigDecimal result = new  
BigDecimal(currentTransaction.calcTotal());  
result = result.setScale(2, BigDecimal.ROUND_HALF_EVEN);  
return "$" + result;
```

- (6) 'countMoney' 클래스에서 additem() 메서드를 클릭하고 아래 부분으로 바디 부분을 채운다.

```
currentTransaction.add(desc, quantity);
```

- (7) 'Choice.calcTotal()'을 아래와 같이 코드 센스와 snippet를 사용하여 채운다.

```
public double calcTotal() {  
    total = 10000; //1만원을 넣었다고 가정하자.  
    int sizeOfItems = listChoiceItem.size();  
    for (int k = 0; k < sizeOfItems; k++) {  
        ChoiceItem item = (ChoiceItem)listChoiceItem.get(k);  
        total -= item.calcTotal();  
    }  
    return total;  
}
```



[그림 12-23] 코드 센스를 이용한 클래스 다이어그램

오늘의 강의 요약

- UML 다이어그램 예제: 자판기 프로그램
- 유스케이스 다이어그램
- 클래스 다이어그램
- 순차 다이어그램

Q & A

More Information?

오유수, Ph.D.

yoosoo.oh@daegu.ac.kr

공대5호관 5506D호

