

3장. 실체(Instance) II

(2023)



시작하기 전에

이 장은 실체에 대한 표현에 대해서 설명합니다. 실체는 텍스트와 그래픽으로 표현됩니다.

텍스트 표현을 설명하기 위해 팩트(fact)에 대한 개념을 설명합니다. 링크에 대해 좀 더 깊은 수준으로 설명합니다.

이번 장을 통해 실체에 대해 텍스트와 그래픽으로 표현할 수 있어야 합니다.

1절. 팩트

1.1 실체는 도메인에 존재하는 사실, 팩트(fact)입니다.

물리적이든 개념적이든 도메인에 존재하는 실체는 도메인 전문가에 의해 소프트웨어 개발자에게 언어적으로 전달됩니다.

도메인 전문가는 도메인에 존재하는 물리적 또는 개념적 실체를 추상화하고 개념화하고 구조화합니다.

도메인 전문가와 소프트웨어 개발자는 언어적 실체를 통해 커뮤니케이션 합니다. 도메인에 실제적으로 존재하는 실체들을 가지고 직접적으로 커뮤니케이션 하는 것이 아닙니다.

소프트웨어 개발자는 실체에 대한 사실(fact)에만 관심이 있습니다. 거짓에는 관심이 없습니다.

우리는 도메인 전문가에 의해 소프트웨어 개발자에게 전달되는 사실적 실체, 언어적 실체를 **팩트(fact)**라고 합니다.

2절. 객체의 그래픽 표현

2.1 객체와 값은 링크와 함께 UML 객체 다이어그램에 작성됩니다. 값은 객체의 특성을 나타내기 위해 사용됩니다.

2.2 객체는 이름 부분과 특성 부분을 갖는 사각형으로 작성합니다. 특성 부분에는 다수의 특성이 작성될 수 있습니다.

2.3 객체 이름 부분은 다음과 같은 형식으로 작성합니다.

[<객체 이름>] ':' <클래스 이름>

클래스는 여기에서 설명되지 않으니 '콜론(:) 뒤에 클래스 이름을 작성하는구나!' 정도만 확인하고 넘어가면 됩니다.

객체 이름은 보통 객체를 식별할 수 있는 이름을 사용합니다.

이름은 생략할 수 있습니다. 이름이 생략된 객체를 익명 객체라 합니다. '어떤 객체가 있어'라고 하는 것입니다.

그림 3.1에서,

이름에 밑줄이 있는 것을 보고 객체임을 알 수 있습니다. 객체 이름은 101이고 클래스 이름은 CourseOffering입니다.

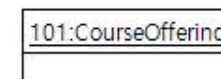


그림 3.1

2.4 특성은 다음과 같은 형식으로 작성합니다.

<특성 이름> [':' <데이터타입 이름>] '=' <특성 값>

데이터타입도 여기에서 설명되지 않았습니다.

특성 값이 문자열인 경우에는 " 또는 "와 같은 인용부호를 사용합니다. 그렇지 않은 경우에는 값을 직접 작성합니다.

예) name='n2', number=101

그림 3.2에서

CourseOffering 101 객체의 number는 데이터타입이 int이고 특성 값은 101입니다.

101은 객체의 특성 값인데 객체 이름으로도 사용되었습니다. 객체의 특성들 중에 객체를 식별할 수 있는 특성이 있는 경우, 일반적으로 그 특성 값을 객체이름으로 사용합니다.

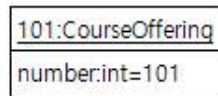


그림 3.2

의미적으로 가까운 특성들을 그룹으로 묶어 다룰 수 있습니다. 이를 **데이터값**이라 합니다.

위치를 표현할 때, 위치_x, 위치_y로 표현할 수도 있고, 위치(x, y)라고 표현할 수도 있습니다. 사람들은 후자를 선호하고, 이미 이런 방식을 사용하고 있습니다.

데이터값인 경우 값 부분은 '(<특성>[', <특성>]*')와 같이 작성됩니다.

x좌표와 y좌표가 100인 location 특성이 있을 때, 데이터타입을 생략한다면, location=(x=100, y=100)와 같이 작성 합니다.

UML에서는 데이터값을 객체와 동일하게 표현하는 것도 지원합니다. 이렇게 할 경우, 객체와 데이터값을 링크해야 하는데, 이는 링크 개

념에 모순됩니다. 우리는 이러한 문제를 근본적으로 막기 위해 객체 표현과 같이 데이터값을 표현하는 것을 금합니다.

특성 값이 다수인 경우에는 다음과 같이 작성합니다.

'{<특성 값>[, <특성 값>]*}'

예) favoriteColors = {"Green", "Red"}

데이터타입을 표현해야 한다면, 다중성을 표현해야 합니다.

예) favoriteColors : string[*] = {"Green", "Red"}

위 예는 favoriteColors로 string 타입의 값이 여러 개 작성될 수 있음을 나타냅니다.

다중성도 여기에서 설명되지 않았습니다.

3절. 객체의 팩트 표현

3.1 객체는 독립적 존재를 나타내는 것이고, 독립적 존재는 ‘있다’ 또는 ‘이다’로 표현됩니다.

<객체 이름> ‘는’ <클래스 이름> ‘이다.’
 <클래스 이름> <객체 이름> ‘(이)가 있다.’
 예) 홍길동은 사람이다.
 사람 홍길동이 있다.

클래스 이름이 생략될 수 있습니다.

3.2 특성은 객체가 어떠한지를 설명하는 구체적인 값으로 표현됩니다.

<객체 이름> ‘의’ <특성이름> ‘는’ <구체적인 값>
 예) 홍길동의 나이는 20이다.

특성이름이 생략될 수 있습니다.

4절. 링크

4.1 링크는 원하는 시점에 연결된 대상을 찾을 수 있도록 하는 객체와 객체 사이의 강한 연결입니다.

그림 3.3에서

Justine과 연결된 사람들이 누구인지를 알고 싶을 때, Justine을 기준으로 연결된 링크를 따라가 보면 됩니다. Justine에는 Poll, Mary, Tom이 연결되어 있습니다.

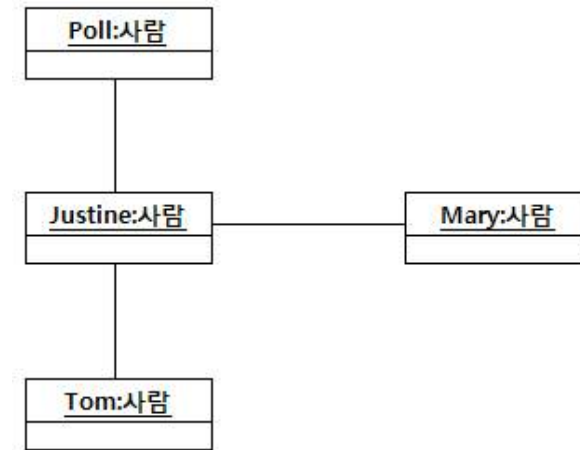


그림 3.3

링크를 굵은 동아줄로 생각합니다.

링크된 객체를 찾는 것은 연결된 동아줄을 움켜쥐고 힘껏 당기는 것으로 생각합니다. 동아줄을 당기면 그 동아줄에 연결된 객체들이 짝 떨어집니다.

그림 3.3에서 Justine에 연결된 사람이 누구인지를 알고 싶을 때, Justine에 연결된 동아줄을 움켜쥐고 당깁니다. 그러면 Poll과 Mary와 Tom이 짝 떨어집니다.

그림 3.3에 등장하는 객체들에 대해서 좀 더 자세히 알아보니, Poll과 Tom은 Justine의 employee이고 Mary는 wife라고 합니다.

그림 3.3은 'Justine에 연결된 사람들은 누구인가?'라는 물음에 답할 수 있지만 'Justine의 employee는 누구인가?'와 'Justine의 wife는 누구인가?'라는 물음에는 답할 수 없습니다.

이 각각의 물음에 답하기 위해 그림 3.4와 같이 동아줄(링크) 끝에 꼬리표를 붙입니다.

이렇게 함으로, 'Justine의 employee는 누구인가?'와 'Justine의 wife는 누구인가?'라는 각각의 질문에 답할 수 있습니다. Justine의 wife를 찾는다면 Justine에서 시작해서 husband와 wife라는 꼬리표를 가진 동아줄을 찾고 그것을 당기면 됩니다.

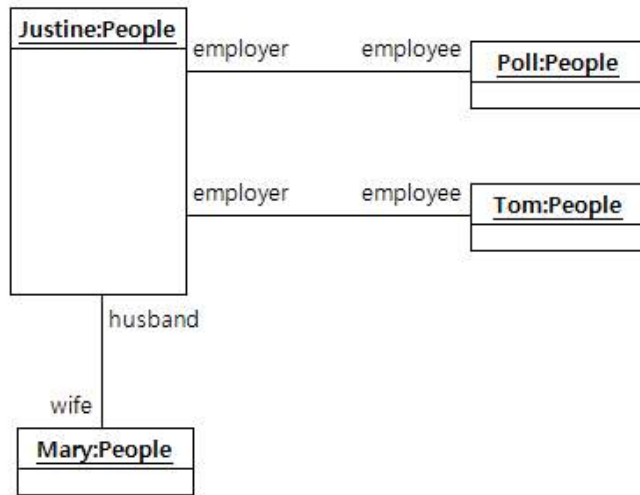


그림 3.4

이 질문 외에도 그림 3.4를 통해 'Mary의 husband는 누구인가?', 'Poll의 employer는 누구인가?', 'Tom의 employer는 누구인가?' 라는 질문들에도 답할 수 있습니다.

4.2 역할은 쌍을 이룹니다.

이 꼬리표는 한 객체에 연결된 다른 객체의 상대적인 역할입니다. 역할은 상대적이므로 쌍을 이룹니다. Poll과 Justine, Tom과 Justine은 employee-employer라는 역할 쌍이 같은 동아줄로 연결된 것입니다.

5절. 링크의 그래픽 표현

5.1 링크는 연결되는 객체와 객체 사이에 실선으로 작성합니다. 실선의 양쪽 끝에는 객체의 상대적인 역할이 작성됩니다.

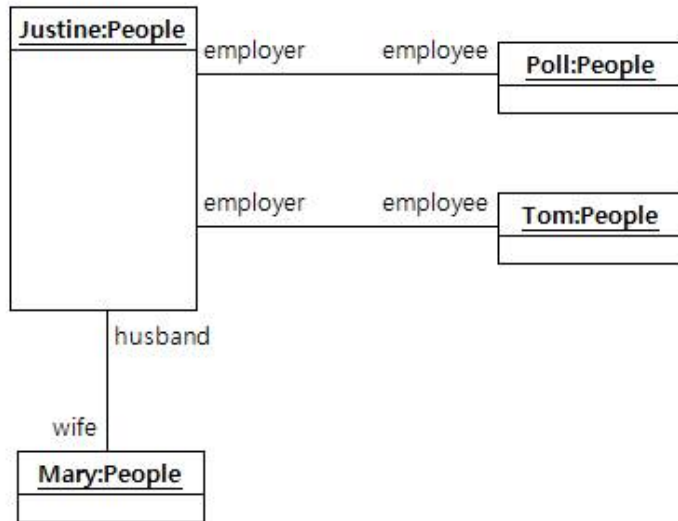


그림 3.5

5.2 역할 이름이 클래스 이름과 같은 경우에는 역할 이름을 생략할 수 있습니다.



그림 3.6

영문으로 작성하는 경우에는 생략하지 않고 클래스 이름의 첫 글자를 소문자로 해서 작성할 수도 있습니다.



그림 3.7

6절. 링크의 팩트 표현

6.1 링크는 역할 쌍으로 표현됩니다.

<객체 이름> '는' <객체 이름>'의 ' <역할 이름> 이고,

<객체 이름> '는' <객체 이름>'의 ' <역할 이름> 이다.

예) 저스틴은 메리의 남편이고, 메리는 저스틴의 아내이다.

상대적인 역할이 분리되어 독립적으로 작성될 수도 있고, 한 객체가 여러 객체들과 연결될 수 있으므로 객체이름이 여러 개 열거될 수 있습니다.

7절. 객체 다이어그램 작성

7.1 주어진 팩트에서 객체를 식별합니다.

Poll, Justine, Tom, Mary는 People이다.

Poll, Justine, Tom, Mary는 People 클래스의 객체입니다.

7.2 주어진 사실을 통해 역할 쌍을 식별합니다.

Justine은 Poll과 Tom의 employer이고, Poll과 Tom은 Justine의 employee이다. Justine은 Mary의 husband이고, Mary는 Justine의 wife이다.

Justine의 입장에서 보면,

Poll과 Tom과의 역할 쌍은 employer - employee이고,

Mary와의 역할 쌍은 husband - wife입니다.

Poll과 Tom의 입장에서 보면,

Justine과의 역할 쌍은 employee - employer입니다.

Mary의 입장에서 보면,

Justine과의 역할 쌍은 wife - husband입니다.

employer - employee와 employee - employer는 방향만 뒤집어진 것이지 같은 역할 쌍입니다. husband - wife, wife - husband도 마찬가지입니다.

7.3 팩트는 텍스트 표현이고, 객체 다이어그램은 그래픽 표현입니다. 같은 사실을 표현합니다.

7.4 팩트로 식별된 객체를 객체 다이어그램의 객체로 작성합니다.



그림 3.8

7.5 역할 쌍에 대한 링크를 준비합니다.

참고에 역할 쌍에 해당하는 태그가 붙은 동아줄이 가득하다고 생각합니다.



그림 3.9

7.6 주어진 사실을 통해 객체들 사이에 링크를 연결합니다.

Justine은 Poll의 employer이고, Poll은 Justine의 employee이다.

employer - employee 역할 쌍을 가진 링크를 가져다가 employer쪽을 Justine으로 하고, employee쪽을 Poll로 해서 연결합니다.

Justine은 Tom의 employer이고, Tom은 Justine의 employee이다.

employer - employee 역할 쌍을 가진 링크를 가져다가 employer쪽을 Justine으로 하고, employee쪽을 Tom으로 해서 연결합니다.

Justine은 Mary의 husband이고, Mary는 Justine의 wife이다.

husband - wife 역할 쌍을 가진 링크를 가져다가 husband쪽을 Justine으로 하고, wife쪽을 Mary로 해서 연결합니다.

그림 3.10은 링크를 모두 작성한 모습입니다. 태그(역할 이름 테두리 박스)는 이해를 돕기 위한 것입니다. 링크 끝에는 역할 이름만 작성하면 됩니다.

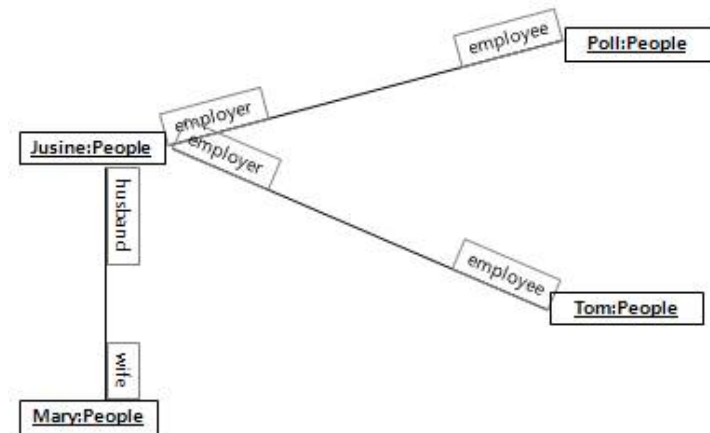


그림 3.10

7.7 주어진 팩트에서 특성을 식별합니다.

Poll, Justine, Tom, Mary는 People이다.

객체에 대한 직접적인 설명부분은 없습니다. 객체의 이름으로 사용된 부분을 보고, 어떤 특성을 사용한 것인지를 파악합니다.

이를 통해 Poll, Justine, Tom, Mary은 People 객체의 이름임을 알 수 있습니다.

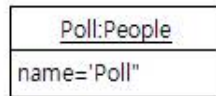


그림 3.11

7.8 객체 다이어그램에 점차적으로 표현이 더 늘어나는 것은 도메인의 사실이 점차적으로 더 많이 다이어그램 작성자에게 알려졌다는 것을 의미합니다.

7.9 객체 다이어그램은 다이어그램이 작성된 시점까지의 사실들을 모아 놓은 스냅샷(snapshot)입니다.

마치기 전에

이번 장에서는 객체와 값과 링크에 대한 텍스트 표현과 그래픽 표현에 대해서 설명했습니다.

단순한 표현들이지만 무시하지 마시고 직접 손으로 그려보십시오.

대부분의 큰 잘못은 단순하고 쉬운 것들을 무시하는 것이 습관이 되어 일어납니다.

7절의 내용을 여러 번 반복해서 해 보십시오.

팩트로 객체 다이어그램을 작성해 보기도 하고, 객체 다이어그램을 보고 팩트를 작성해 보기도 해봅니다. 객체 다이어그램이 무엇을 표현하고 있는지에 대한 분명한 이해가 생길 때까지 반복합니다.

링크와 역할 쌍 사이의 의미에 대해서도 감을 잡도록 주의를 기울입니다.

객체 다이어그램은 중요합니다.

보통 현장에서는 객체 다이어그램을 잘 작성하지 않습니다. 하지만 우리는 실체를 작성하는 객체 다이어그램을 중요하게 다룹니다.

객체 다이어그램에 실체를 직접적으로 표현해 보는 것은 실체에 대한 감을 잡는데 도움이 됩니다.

실체에 대한 이해 없이 실체에 대응하는 개념을 이해하는 것은 대단히 어렵습니다. 예를 들어 객체를 잘 모르고 클래스에 대해서 잘 알 수 없습니다. 기존의 설명 방식 중에 많은 경우가 클래스로 시작해서 객체로 넘어갑니다. 이것은 구체성을 통하여 더 잘 이해할 수 있는 인간의 특성을 무시한 것입니다. 혼돈의 지름길입니다.

우리는 실체에 대한 이해를 그대로 실체에 대응하는 개념에 사용합니다.

우리는 객체 다이어그램을 스냅샷으로 사용하여 모델링 결과를 시물레이션하거나 검증하는데도 사용합니다.