6장. **실체**(Instance) III

(2023)

시작하기 전에

이번 장은 실체에 대한 기본적인 것들 중 일부 내용을 확장하고, 실체에 대해 알아야 하는 좀 더 깊이 있는 것들을 추가적으로 설명합니다.

이번 장은 2장과 3장에 더해 실체에 대한 완성된 개념을 갖는 것을 목표로 합니다.

1절. 전체와 부분

1.1 객체가 존재하기 위해서는 이미 존재하는 어떤 것이 필요합니다.
존재한다는 것은 어딘가에 존재한다는 의미를 함축하고 있습니다.
'객체가 존재해야 하는 어딘가'라고 할 때 어딘가 또한 이미 독립적으로 인식된 것으로 객체입니다.

1.2 객체는 다른 객체가 존재하는 공간으로서의 객체와 그 공간에 존재하는 객체로 구분됩니다.
객체가 존재하는 공간은 객체의 입장에서 보면 자신 밖에 있는 외부 환경(environment)입니다.

1.3 외부환경에 해당하는 객체를 전체(whole)라 하고, 그 공간 안에 존재하는 객체를 부분(part)이라 합니다.

1.4 전체와 부분은 부분들을 모아 전체를 이루는 구성(構成)의 의미로 사용됩니다.
그림 6.1의 삼각형 T1은 세 점 P1, P2, P3로 구성됩니다.

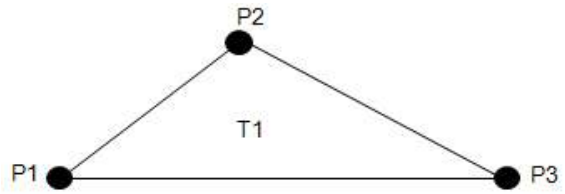

그림 6.1

1.5 전체와 부분은 전체가 부분을 담는 포함의 의미로도 사용됩니다.
한 폴더에는 다수의 파일들이 포함됩니다.

1.6 전체와 부분 사이에도 링크가 필요합니다.

전체도 부분도 독립적으로 인식되는 객체입니다.

전체도 부분도 독립적으로 인식되는 상황에서 어떤 것이 전체이고 어떤 것이 부분임을 나타내는 것이 필요한데 이를 링크로 표현합니다.

2절. 전체 부분의 표현

2.1 전체 부분은 링크로 표현됩니다.

2.2 링크의 팩트 표현에 '포함'이나 '구성'의 의미가 추가됩니다. 전체가 부분을 '갖는다(has)'는 의미가 추가되기도 합니다.

구성의 의미는 '구성', '조합', '복합'과 같은 단어들로 표현됩니다.

3절. 상태

3.1 상태는 변화를 겪는 객체가 그때 어떠했는지 또는 지금 어떠한지에 대한 표현입니다.

3.2 객체는 양적인 변화를 겪습니다.
사람은 매년 나이를 먹습니다.
고객의 구매에 따라 포인트가 누적됩니다.
해바라기의 꽃잎 색은 노랑에서 더 노랑이 됩니다.

3.3 객체의 양적변화를 표현하는 상태를 양적상태라 합니다.

3.4 특성의 변화로 객체의 양적상태를 표현합니다. 양적상태는 객체 자체를 대상으로 표현됩니다.
나이 값으로 나이의 변화를 겪는 사람의 양적상태를 표현합니다.
포인트 값으로 포인트 변화를 겪는 고객의 양적상태를 표현합니다.
꽃잎 색으로 꽃잎 색의 변화를 겪는 해바라기의 양적상태를 표현합니다.

3.5 특성은 객체에게 일어난 양적변화들에 대한 현재시점에서의 누적치 만을 갖습니다.

3.6 특성으로는 현재시점에서의 객체의 양적상태만을 알 수 있습니다.
포인트 값으로 홍길동이 지금 몇 포인트를 갖고 있는지를 알 수 있습니다. 홍길동의 포인트가 어떤 변화를 겪었는지는 알 수 없습니다.

3.7 객체에게 일어나는 모든 양적변화를 누적함으로 객체의 양적변화를 추적할 수 있습니다.

'홍길동의 포인트가 가장 높았던 때는 언제인가?'라는 질문에 답하려면 홍길동의 포인트 변화를 추적할 수 있어야 합니다.
홍길동의 포인트 변화를 추적하려면 먼저 포인트 변화가 있을 때마다 누적해 두어야 합니다.

객체의 모든 양적변화를 누적해 두면 특정 시점으로 객체를 되돌릴 수도 있게 됩니다. 과거로 돌아갈 수 있는 타임머신을 준비해 두는 것입니다.

3.8 객체는 질적인 변화를 겪습니다.

그림 6.2와 같이 홍길동이란 사람이 있습니다.
홍길동은 매년 나이를 먹습니다. 양적인 변화를 겪습니다. 홍길동은 나이라는 특성으로 홍길동에게 일어나는 양적인 변화를 표현합니다.

그림 6.2

홍길동은 19세까지는 나이라는 양적인 변화만을 겪었습니다. 20살부터 투표가 가능한 것으로 가정한다면, 20살이 되면서 홍길동에게 큰 변화가 일어났습니다. 투표를 할 수 없다가 투표를 할 수 있게 되는

질적인 변화가 일어난 것입니다.

3.9 객체의 질적변화를 표현하는 상태를 질적상태라 합니다.

3.10 질적인 변화는 행위수행가능여부에 변화가 생기는 것입니다. 객체가 행위를 할 수 없다가 할 수 있게 되거나, 행위를 할 수 있다가 행위를 할 수 없게 되는 변화를 겪는 것입니다.

3.11 질적상태는 상태머신(state machine)으로 표현됩니다.
상태머신은 객체가 생애주기 동안 겪는 모든 질적인 변화를 하나로 표현합니다.

3.12 객체는 시간의 흐름에 따라 또는 객체에 일어난 행위에 따라 양적인 변화와 질적인 변화를 겪습니다.

3.13 객체에게 일어나는 모든 변화는 객체 자체를 대상으로 표현됩니다.

4절. 정체성

4.1 정체성은 그것이 그것인 것을 아는 방법입니다.

4.2 정체성은 바뀔 수 없습니다.

객체는 양적변화와 질적변화를 겪습니다.
변화라는 것은 달라진다는 것입니다. 달라졌는데 어떻게 그것이 그것인지를 알 수 있습니까?

변하지 않는 것, 즉 바뀔 수 없는 것을 두고
이것을 기준으로 삼아
그것이 그것인지를 알 수 있는 것입니다.

4.3 정체성은 고유합니다.

다수의 객체들 중에서 그것이 그것인지를 안다는 것은 그것을 그것으로 여기게 하는 그것만의 고유함이 있다는 것입니다.

4.4 객체가 팩트로 표현될 때 정체성도 팩트로 표현되어야 합니다.

정체성은 객체를 객체이게 만드는 것으로 객체가 팩트로 표현된다면 당연히 정체성도 팩트로 표현되어야 합니다.

4.5 정체성에 대한 팩트 표현을 식별자라고 합니다.

4.6 식별자는 고유하고 바꿀 수 없으며 기억하기 쉬워야 합니다.

식별자는 정체성에 대한 팩트 표현이므로 정체성과 같이 고유하며 변하지 않아야 합니다.

팩트에서 객체를 식별해내려면 그 객체의 식별자를 기억하고 있어야 합니다.

사람들은 정체성을 팩트로 표현하기 위해 식별자를 만들어냅니다. 가장 일반적인 방법이 이름을 부여하는 것입니다.
이름은 기억하기 쉽도록 짧게 만듭니다. 짧게 만든다는 것은 그만큼 경우가 수가 적다는 것입니다. 객체 수가 조금만 많아지면 같은 이름이 등장할 가능성이 높습니다.
사람들은 이름 충돌의 문제를 해결하기 위해, 회원번호, 학번, 주민등록번호와 같은 이름보다 좀 더 긴 코드 화된 식별자를 만들어내기도 합니다.

사람들은 식별자를 만들면서 중복의 가능성을 모두 살피지 않습니다. 객체가 사용되는 범위 내에서만 고유할 정도의 노력을 기울입니다. 이러한 이유로 객체가 사용되는 범위가 확장될 경우 식별자에 중복이 생길 가능성이 있습니다.
여러 대학들이 통합될 때 각각의 대학에서는 중복되지 않던 학번이 통합된 대학에서는 중복될 수 있습니다.

4.7 식별자는 바뀔 수 없습니다.

식별자는 객체를 나타내는 것으로 식별자가 바뀌었다는 것은 다른 객체가 되었다는 것입니다. 그 객체가 여전히 그 객체라는 것은 식별자가 같다는 것입니다.

4.8 특성은 식별자로 사용할 수 없습니다.

특성은 양적변화를 나타내는 것으로 의미 그대로 변하는 것입니다. 변할 수 있는 특성을 변할 수 없는 식별자로 사용할 수 없습니다.

4.9 소프트웨어 개발자는 사람을 위한 식별자와 기계를 위한 식별자를 고려해야 합니다.

사람을 위한 식별자는 사람의 기억의 한계로 한정된 길이를 가져야 합니다. 식별자가 너무 길면 기억할 수 없어 어디엔가 적어두어야 합니다. 자주 객체를 식별해야 하는 상황이라면 적어둔 것을 매번 봐야 하는 것은 불편합니다.

한정된 길이의 식별자는 가능성이 낮을 뿐 확장과 같은 이유로 중복이 발생할 수 있습니다. 식별자에게 있어 중복은 치명적이기 때문에 중복을 완전히 배제할 방법이 필요합니다.

UUID(Universally Unique IDentifier)나 GUID(Globally Unique IDentifier)는 중복을 거의 완전히 배제할 수 있는 형태로 식별자를 만드는 것을 가능하게 합니다. 기계는 이러한 형태의 식별자를 정확히 기억하고 정확히 비교할 수 있습니다.

4.10 사람을 위한 식별자는 사용자 인터페이스에 드러나고 기계를 위한 식별자는 내부적으로만 사용합니다.

4.11 사람을 위한 식별자는 여러 개 있을 수 있습니다.

이름으로 식별이 가능하면 이름으로 식별합니다. 이름 충돌이 발생하면 이름보다 더 고유함을 인정받는 주민등록번호로 식별합니다. 이름도 주민등록번호도 사람을 위한 식별자입니다.

4.12 객체는 식별자로 같은 객체인지를 비교할 수 있습니다. 값은 같은 값인지를 비교할 수 있는 방법이 필요합니다.

식별자는 정체성을 표현하는 것으로 식별자가 같으면 같은 객체입니다.

값은 정체성이 없기 때문에 어떤 경우에 같은 값으로 다룰 것인지를 정해야 합니다.

대부분의 프로그래밍 언어들은 기본적인 값타입들을 제공하고 값들의 동등비교(equal)에 대한 방법을 정해놓고 있습니다. 데이터값은 값들의 그룹으로 사용자가 정의해서 사용하는 타입으로 값의 같음을 알아야 한다면 동등비교방법을 정의해야 합니다.

마치기 전에

이번 장에서는 전체와 부분에 대한 기본적인 것들을 설명했습니다. 또한 상태와 정체성에 대해서 좀 더 깊이 있게 설명했습니다.

전체와 부분은 개념 III에서 좀 더 깊이 있게 다룹니다.

상태머신은 행위에서 좀 더 자세히 다룹니다.

정체성은 속성에서 완성됩니다.