09장 자료 추상화와 객체지향

【9.6.1】 스몰토크의 객체 모델

스몰토크는 프로그래밍 환경에 완전히 통합되어 있다. 사실 이 책에서 언급한 다른 모든 언어와 달리 스몰토크 프로그램은 문자의 단순한 나열로 구성되지 않는다. 오히려 스몰토크 프로그램은 스몰토크 구현의 브라우저 내에 보지 않으면 안 된다. 여기서 주어진 프로그램 단위의 다른 부분을 식별하기 위해 폰트 변화와 스크린 위치를 사용할 수 있다. 동시대의 인터리스프와 PARC의 파일럿/메사 프로젝트와 함께 스몰토크 그룹은 비트로 매핑된 스크린, 창, 메뉴와 마우스 등 지금의 보편적인 개념을 개발한 것을 공유한다.

스몰토크는 모든 변수에 대해 유형을 지정하지 않은 참조 모델을 사용한다. 모든 변수는 객체를 참조한다. 하지만 객체의 클래스를 정적으로 알 필요가 없다. 9.3.1절에 설명한 것처럼(그리고 그림 9.2에서 예를 들어 설명한 것처럼) 모든 스몰토크 객체는 `Object`라는 단일 기본 클래스로부터 내려온 클래스의 인스턴스다. 모든 자료는 객체에 포함된다. 이 중 가장 명백한 것은 (클래스 `Boolean`에 속한) `true`와 (클래스 `Integer`에 속한) `3`과 같은 단순한 불변하는 객체다.

예 9.64
스몰토크에서의 메시지로서의 연산

스몰토크에서는 연산을 모두 객체에 보낸 메시지로 개념화한다. 예를 들어 수식 `3 + 4`는 `a +` 메시지를 인자로 객체 `4`로의 참조와 함께 (불변의) 객체 `3`에 전달하는 것을 뜻한다. 이 메시지에 대한 응답으로 객체 `3`은 (불변의) 객체 `7`로의 참조를 만들어 반환한다. 비슷한 예로 `a`와 `b`가 변수면 수식 `a + b`는 `a +` 메시지를 인자로 `b`에 있는 참조와 함께 `a`가 참조하는 객체에 전달하는 것을 의미한다. `a`가 `3`을 참조하고 `b`가 `4`를 참조한다면 그 효과는 상수일 때와 동일할 것이다.

예 9.65
혼합위 메시지

6.1절에서 언급한 것처럼 여러 개의 인자를 가진 메시지는 다단어("혼합위") 이름을 가진다. 각 단어는 콜론으로 끝나고 각 인자는 단어를 따라온다. 다음 수식은 `display`

On: at: 메시지를 인자로 myScreen과 location이 참조하는 객체와 함께 변수 myBox가 참조하는 객체에 보낸다.

```
myBox displayOn: myScreen at: location
```

예 9.66

ifTrue: ifFalse: 메시지로서의 선택

스몰토크의 제어 흐름을 메시지로 개념화하기까지 한다. 다음 선택 구성소를 살펴보자.

```
n < 0
    ifTrue: [abs <- n negated]
    ifFalse: [abs <- n]
```

이 코드는 a < 0 메시지(인자로 0을 가진 a < 메시지)를 n이 참조하는 객체에 전송함으로써 시작한다. 이 메시지에 대한 응답으로 n이 참조하는 객체는 두 불변 객체 true와 false 중 하나로의 참조를 반환할 것이다. 이 참조는 n < 0 수식의 값이 된다.

스몰토크는 우선순위나 결합법칙 없이 왼쪽부터 오른쪽으로 수식을 계산한다. 그러므로 n < 0의 수식은 ifTrue:ifFalse: 메시지의 수령인이 된다. 이 메시지는 두 인자를 갖고 각기 블록이다. 스몰토크의 한 블록은 각괄호로 둘러싸인 코드의 일부다. 그것은 리스프에 람다 수식의 의미와 거의 필적하는 의미를 가진 불변 객체다. 한 블록을 실행하려면 value 메시지를 보낸다.

ifTrue:ifFlase: 메시지를 보낼 때 불변 객체 true는 value 메시지를 (블록인 것이 더 나은) 첫 번째 인자에 전달하고 나서 그 결과를 반환한다. 반면에 동일한 메시지에 대한 응답으로 객체 false은 value 메시지를 두 번째 인자(ifFalse:에 뒤따라오는 블록)로 전달한다. 각 블록에서 왼쪽 화살표(<-)는 대입 연산자다. 대입은 메시지가 아니다. 그것은 오른편에 있는 값 계산의 부수효과다. 알골 68과 같은 수식 기반 언어에서처럼 대입 연산의 값은 오른편의 값이다. 선택 수식의 전체 값은 블록 중 하나의 값, 즉 그것이 음수가 아니냐에 따라 n으로나 그것의 덧셈 역원으로의 참조일 것이다. 편리함을 위해 스몰토크의 논리형 객체도 ifTrue:, ifFlase:, ifFalse:ifTrue: 메소드 등을 구현한다.

예 9.67

메시지를 이용한 반복

반복을 비슷한 형태로 모델화한다. 열거 제어형 루프에 대해 클래스 Integer는 timesRepeat:와 to: by: do: 메소드를 구현한다.

```
pow <- 1.
10 timesRepeat:
    [pow <- pow * n]

sum <- 0.
1 to: 100 by: 2 do:
    [:i | sum <- sum + (a at: i)]
```

이 코드의 처음은 n^{10}을 계산한다. timesRepeat: 메시지에 대한 응답으로 정수 k는 value 메시지를 인자(블록)에 k번 전달한다. 두 번째 코드는 a가 참조하는 배열의 홀

수 인덱스를 가진 원소의 합을 구한다. to: by: do: 메시지에 대한 응답으로 정수 k는 기대하는 것처럼 행동한다. t가 첫 번째 인자이고 b가 두 번째 인자라고 하면 그것은 value: 메시지를 세 번째 인자(블록)에 ⌊(t-k+b)/b⌋번 보낸다. value:의 끝에 있는 콜론에 유의하자. 평문 value 메시지는 인자가 1개다. value: 메시지는 하나의 인자를 갖는다. 즉, (하나의) 형식 매개변수를 가진 블록은 그것을 받아들일 수 있다. 앞선 루프의 예에서 정수 1은 메시지 value: 1, value: 3, value: 5 등을 블록 [:i | sum<- sum+(a at:i)]에 보낸다. 블록의 시작 부분에 :i |는 형식 매개변수다. at: 메시지를 배열로 취급한다. 1의 스텝 크기를 이용한 반복에 대해 정수는 to: do: 메소드를 제공한다.

예 9.68
클로저로서의 블록

그것이 객체이기 때문에 한 변수는 블록을 참조할 수 있다.

```
b <- [n <- n + 1].          "b는 이제 클로저임"
c <- [:i | n <- n + i].     "c도 마찬가지"
...
b value.                    "n을 1만큼 증가시킴"
c value: 3.                 "n을 3만큼 증가시킴"
```

두 매개변수를 가진 블록은 value: value: 메시지를 기대한다. j개의 매개변수를 가진 블록은 j번의 value:로 구성된 이름을 가진 메시지를 기대한다. 스몰토크의 주석은 큰따옴표로 둘러싸여 있다(문자열은 작은따옴표로 둘러싸임).

예 9.69
메시지로 함께 논리적인 루프 실행

논리 제어형 루프에 대해 스몰토크는 whileTrue: 메시지에 의존하고 블록은 그것을 이해한다.

```
tail <- myList.
[tail next ~~ nil]
    whileTrue: [tail <- tail next]
```

이 코드는 tail을 myList의 마지막 원소로 설정한다. 이중틸드(~~) 연산자는 "동일한 객체를 참조하지 않음"을 의미한다. 메소드 next가 그것의 수령인 뒤에 따라오는 원소로의 참조를 반환한다고 가정한다. whileTrue: 메시지에 대한 응답으로 한 블록은 자신에게 value 메시지를 전달한다. 그 메시지의 결과가 true에 대한 참조면 그 블록은 value 메시지를 원래 메시지의 인자에게 전달하고 반복한다. 블록은 whileFalse: 메소드도 구현한다.

예 9.70
제어 추상화 정의

스몰토크의 블록은 프로그래머가 거의 임의의 제어 흐름 구성소를 만들게 해준다. 간단한 구문 때문에 스몰토크의 블록은 리스프의 람다 수식보다 조작하기 훨씬 쉽다. 사실 to: by: do: 메시지는 반복을 "뒤집어서" 루프의 몸체를 to: by: do: 메소드의 몸체 내에서 (그것에 value 메시지를 보냄으로써) 실행할 수 있는 간단한 메시지 인자로 만든다. 스몰토크 프로그래머는 다른 컨테이너 클래스에 대해 비슷한 메소드를 정의할 수 있어서 반복자의 모든 능력(6.5.3절)과 call_with_current_continuation

(8.5.4절)의 능력 대부분을 얻을 수 있다.

```
myTree inorderDo: [:node | whatever ]
```

스몰토크에서 계산의 정규 객체 모델이 일정한 구현을 의미할 필요가 없음을 언급할 만한 가치가 있다. 클루 구현이 고유 불변 객체를 값으로 구현하는 것과 마찬가지로 참조의 의미(6.1.2절)에도 불구하고 스몰토크 구현은 실제로 정수로 메시지를 보내기보다는 컴퓨터 계산에 대해 보통 기계 명령어를 사용할 것이다. 비슷한 예로 스몰토크 해석기는 가장 흔한 제어 흐름 구성소(`ifTrue: ifFalse:`, `to: by: do:`, `whileTrue:` 등)를 인식하고 특별하고 더 빠른 코드로 구현할 것 같다.

예 9.71
스몰토크의 재귀

명령형 언어에서 그런 것처럼 스몰토크에서도 적어도 재귀가 동작함을 말하면서 이 하위 절을 마칠 것이다. 다음은 유클리드 알고리즘의 재귀 구현이다.

```
gcd: other                                          "other은 형식 매개변수임"
    [self = other]
       ifTrue: [↑ self].                            "종료 조건"
[self < other]
    ifTrue: [↑ self gcd: (other − self)]      "재귀"
    ifFalse: [↑ other gcd: (self − other)]    "재귀"
```

위 화살표(↑) 기호는 C나 알골의 `return`과 비슷하다. 키워드 `self`는 C++의 `this`와 비슷하다. 스몰토크 브라우저에서 나타나는 것과 흡사하게 혼합된 글씨체로 된 코드를 보여줬다. 그 메소드의 헤더를 굵은 글씨로 식별한다.

✓ 확인문제

48. 현대 GUI 기반 개인용 컴퓨터를 개척한 1970년대 제록스 PARC에서의 3가지 프로젝트를 제시하라.

49. 스몰토크에서 메시지의 개념을 설명하라.

50. 스몰토크는 여러 개의 메시지 인자를 어떻게 표시하는가?

51. 스몰토크에서 블록은 무엇인가? 리스프의 어떤 기법과 비슷한가?

52. 스몰토크가 제어 흐름을 메시지 계산으로서 모델화하는 방법에 대해 3가지 예를 제시하라.

53. 스몰토크에서 유형 검사가 어떻게 작동하는지를 설명하라.