Programação Orientada à Objetos

Estruturas e Exceções

Prof. Edson Alves

Faculdade UnB Gama

2020

Sumário

Estruturas de seleção

- Smalltalk permite a escrita de blocos de código que só serão executados caso uma condição seja verdadeira (ou falsa)
- Isto é feito de acordo paradigma da orientação aos objetos: a condição gera um objeto do tipo booleano, o qual recebe uma mensagem cujo argumento é um bloco, que também é um objeto
- De fato, um bloco é um objeto composto por *statements* executáveis
- Assim, a estrutura de seleção `if-else`, comum em outras linguagens, de fato não é uma forma da linguagem Smalltalk, e sim um comportamento dos objetos booleanos

Métodos ifTrue e ifFalse

- O método `ifTrue` recebe como argumento um bloco de comandos e o executa, caso o objeto seja `true`, ou retorna sem executá-lo, caso o objeto seja `false`

```
booleana ifTrue: [ bloco_de_comandos ]
```

- O método `ifFalse` tem comportamento análogo, executando o bloco caso o objeto seja `false` e retornando sem executar se o objeto é `true`

```
booleana ifFalse: [ bloco_de_comandos ]
```

- Eles podem ser usados para simular o comportamento do `if-else` padrão das linguagens imperativas:

```
condicao ifTrue: [ bloco_true ]; ifFalse: [ bloco_false ]
```

Exemplo de uso de condicionais em SmallTalk

```
1  Object subclass: QuadraticPolynomial [
2      | a b c |
3
4      QuadraticPolynomial class >> coefA: av coefB: bv coefC: cv [
5          | r |
6          r := super new . r setA: av . r setB: bv . r setC: cv . ^r
7      ]
8
9      setA: av [
10         (a == 0) ifTrue: [ ^self error: 'a must be non-zero' ] .
11         a := av
12     ]
13
14     setB: bv [ b := bv ]
15     setC: cv [ c := cv ]
16
17     printOn: stream [
18         (a < 0) ifTrue: [ stream nextPutAll: '- ' ] .
19         (a abs ~= 1) ifTrue: [ (a abs) printOn: stream ] .
20         stream nextPutAll: 'x^2 ' .
```

Exemplo de uso de condicionais em SmallTalk

```
21          (b ~= 0) ifTrue: [
22              (b < 0) ifTrue: [ stream nextPutAll: '- ' ];
23                      ifFalse: [ stream nextPutAll: '+ ' ] .
24              (b abs ~= 1) ifTrue: [ (b abs) printOn: stream ] .
25              stream nextPutAll: 'x '
26          ] .
27          (c ~= 0) ifTrue: [
28              (c < 0) ifTrue: [ stream nextPutAll: '- ' ];
29                      ifFalse: [ stream nextPutAll: '+ ' ] .
30              (c abs) printOn: stream
31          ]
32      ]
33 ]
34
35 p := QuadraticPolynomial coefA: 1 coefB: -5 coefC: 6 .
36 p printNl .
37 q := QuadraticPolynomial coefA: -2 coefB: 0 coefC: -1 .
38 q printNl
```

Laços

- Smalltalk possui métodos que permitem executar um bloco repetidas vezes
- Os blocos possuem o método `whileTrue`, que recebe um bloco de comandos que será executado enquanto o bloco que o invocou seja avaliado como verdadeiro

```
[ bloco_condicao ] whileTrue: [ bloco_comandos ]
```

- O bloco de comandos deve, em algum momento, modificar as variáveis que compõem a condição, caso contrário o laço executará indefinidamente
- O método `whileFalse` tem comportamento semelhante, executando o bloco de comandos enquanto a condição for falsa

```
[ bloco_condicao ] whileFalse: [ bloco_comandos ]
```

Exemplo de uso de laços em SmallTalk

```
1  Object subclass: Math [
2      Math class >> numDigits: x [
3          | count n |
4          count := 1 .
5          n := x quo: 10 .
6          [ n > 0 ] whileTrue: [ count := count + 1 . n := n quo: 10 ] .
7          ^count
8      ]
9  ]
10
11 (Math numDigits: 0) printNl .
12 (Math numDigits: 7) printNl .
13 (Math numDigits: 123) printNl .
14 (Math numDigits: 123456789) printNl
```

Repetição

- Se um bloco de comandos deve ser executado exatamente n vezes, uma forma mais concisa e apropriada do que `whileTrue` é o método `timesRepeat` dos inteiros

```
n timesRepeat: [ bloco_de_comandos ]
```

- De fato, os códigos

```
x := n .
[ x > 0 ] whileTrue: [ c1 . c2 . "..." . cN . x := x - 1]
```

e

```
n timesRepeat: [ c1 . c2 . "..." . cN ]
```

são equivalentes

- Os blocos também podem, opcionalmente, ter variáveis locais ou, no caso de coleções, capturar argumentos
- A sintaxe de um bloco é

```
[ :arg1 :arg2 "..." :argN | var1 var2 "..." varM | comandos ]
```

Exemplo de uso de blocos repetidos em SmallTalk

```
1  Object subclass: Math [
2
3      Math class >> fib: n [
4          | a b |
5          a := 0 .
6          b := 1 .
7          n timesRepeat: [ | c | c := a + b . a := b . b := c ] .
8          ^a
9      ]
10 ]
11
12 (Math fib: 1) printNl .
13 (Math fib: 2) printNl .
14 (Math fib: 3) printNl .
15 (Math fib: 4) printNl .
16 (Math fib: 5) printNl .
17 (Math fib: 6) printNl
```

Referências

1. GNU Operation System. GNU Smalltalk, acesso em 08/09/2020.
2. **RATHMAN**, Chris. Smalltalk notes, acesso em 08/09/2020.
3. Savannah. GNU Smalltalk – Resumo, acesso no dia 08/09/2020.
4. **SHALOM**, Elad. *A Review of Programming Paradigms Througout the History – With a Suggestion Toward a Future Approach*, Amazon, 2019.