

CS 535 Object-Oriented Programming & Design
Fall Semester, 2010
Doc 5 Control Messages
Sept 6 2011

Copyright ©, All rights reserved. 2011 SDSU & Roger Whitney, 5500
Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/openpub/>) license defines the copyright on this document.

References

Ralph Johnson's University of Illinois, Urbana-Champaign CS 497 lecture notes,
<http://st-www.cs.uiuc.edu/users/cs497/>

Smalltalk Best Practice Patterns, Beck

Smalltalk With Style, Klimas, Skublics, Thomas

Reading

Smalltalk by Example, Alex Sharp,
Chapter 2 Methods
Chapter 8 Control Structures

if

(boolean expression) ifTrue: trueBlock

(boolean expression) ifFalse: falseBlock

(boolean expression) ifFalse: falseBlock ifTrue: trueBlock

(boolean expression) ifTrue: trueBlock ifFalse: falseBlock

```
a < 1 ifTrue: [Transcript show: 'hi mom' ]
```

```
difference := (x > y)
```

```
    ifTrue: [ x - y]
```

```
    ifFalse: [ y - x]
```

Boolean Expressions

	Symbol	Example
Or		a b
And	&	a & b
Exclusive or	xor:	a xor: (b > c)
Negation	not	(a < b) not

Lazy Logical Operations

	Message	Example
Or	or: aBlock	a or: [b > c]
And	and: aBlock	a and: [c b]

This is not C

This is a runtime error

```
5 ifTrue: [1 + 3]
```

A Style Issue

Both do the same thing

```
difference := (x > y)
  ifTrue: [ x - y]
  ifFalse: [ y - x]
```

```
(x > y)
  ifTrue: [difference := x - y]
  ifFalse: [difference := y - x]
```

isNil

Answers true if receiver is nil otherwise answers false

x isNil

ifTrue: [do something]

ifFalse: [do something else]

Shortcuts

ifNil:ifNotNil:

ifNotNil:ifNil:

ifNil:

ifNotNil:

x

ifNil: [do something]

ifNotNil: [do something else]

Blocks

A deferred sequence of actions – a function without a name

Can have 0 or more arguments

Executed when sent the message 'value'

Similar to

- Lisp's Lambda- Expression

- Erlang's funs

- Ruby's Blocks

- Python's lambda

- Anonymous functions

```
[:variable1 :variable2 ... :variableN |  
 | blockTemporary1 blockTemporary2 ... blockTemporaryK |  
 expression1.  
 expression2.  
 ...]
```

Blocks and Return Values

Blocks return the value of the last executed statement in the block

```
| block x |
```

```
block := [:a :b |
```

```
  | c |
```

```
  c := a + b.
```

```
  c + 5].
```

```
x := block value: 1 value: 2.
```

x has the value 8

Blocks know their Environment

```
| a b |  
a := 1.  
b := 2.  
aBlock := [a + b].  
result := aBlock value
```

result is now 3

```
| a b |  
a := 1.  
b := 2.  
aBlock := [a + b].  
a := 5  
result := aBlock value
```

result is now 6

Blocks and Arguments

Using the value: keyword message up to 4 arguments can be sent to a block.

```
[2 + 3 + 4 + 5] value
```

```
[:x | x + 3 + 4 + 5 ] value: 2
```

```
[:x :y | x + y + 4 + 5] value: 2 value: 3
```

```
[:x :y :z | x + y + z + 5] value: 2 value: 3 value: 4
```

```
[:x :y :z :w | x + y + z + w] value: 2 value: 3 value: 4 value: 5
```

valueWithArguments: can be used with 1 or more arguments

```
[:a :b :c :d :e | a + b + c + d + e ] valueWithArguments: #( 1 2 3 4 5)
```

```
[:a :b | a + b ] valueWithArguments: #( 1 2 )
```

Where is the Value Message

```
difference := (x > y)
  ifTrue: [ x - y]
  ifFalse: [ y - x]
```

In the False class we have:

```
ifTrue: trueAlternativeBlock ifFalse: falseAlternativeBlock
  ^falseAlternativeBlock value
```

In the True class we have:

```
ifTrue: trueAlternativeBlock ifFalse: falseAlternativeBlock
  ^trueAlternativeBlock value
```

While Loop

aBlockTest while True

aBlockTest while True: aBlockBody

aBlockTest while False

aBlockTest while False: aBlockBody

The last expression in aBlockTest must evaluate to a boolean

```
| x y difference |
```

```
x := 8.
```

```
y := 6.
```

```
difference := 0.
```

```
[x > y] while True:
```

```
    [difference := difference + 1.
```

```
    y := y + 1].
```

```
^difference
```

```
| count |
```

```
count := 0.
```

```
[count := count + 1.
```

```
count < 100] while True.
```

```
Transcript
```

```
    clear;
```

```
    show: count printString
```

More Loops

Transcript

```
clear.
```

```
3 timesRepeat:
```

```
[Transcript
```

```
  cr;
```

```
  show: 'Testing!'].
```

```
1 to: 3 do:
```

```
[:n |
```

```
  Transcript
```

```
    cr;
```

```
    show: n printString;
```

```
    tab;
```

```
    show: n squared printString].
```

```
9 to: 1 by: -2 do:
```

```
[:n |
```

```
  Transcript
```

```
    cr;
```

```
    show: n printString].
```

Transcript

```
Testing!
```

```
Testing!
```

```
Testing!
```

```
1 1
```

```
2 4
```

```
3 9
```

```
9
```

```
7
```

```
5
```

```
3
```

```
1
```