

Pure Object Oriented Programming in

Smalltalk

Lukas Renggli

www.lukas-renggli.ch

Lukas Renggli

- Academics
 - PhD Student, University of Bern
- Industrial
 - Independent Software Consultant
- Communities
 - Contributor to Squeak and Seaside
 - Author of Magritte and Pier

Everything
is an Object

**Object communicate by
message passing**

**Method invocation
is late bound**

Methods are public

Single Inheritance

**Instance variables
are private to the object**

Garbage Collection

**Everything is
written in itself**

**Everything happens
on the fly**

What you need

VM

Image

Sources

Changes

www.squeak.org

Community

www.squeak.org

License

APSL 2.0

Bug Tracking

bugs.squeak.org

Community

Squeak Foundation
7 Elected Board Members
Promotion of Squeak

Smalltalk Language

example: aNumber

"This is a method that illustrates every part of the Smalltalk method syntax. It has unary, binary, and keyword messages, declares arguments and temporaries, accesses an instance variable and a global variable, uses literals (array, character, symbol, string, integer, float), uses the pseudo variables true, false, nil, self, and super, and has sequence, assignment, return and cascade. It has both zero argument and one argument blocks. It doesn't do anything useful, though."

```
| x |
true & false not & (nil isNil) iffFalse: [ self halt ].
x := y + super size.
#( $a #a 'a' 1 1.0 ) do: [ :each |
    Transcript
        show: each class name;
        show: each printString ].
^ x < aNumber
```

Syntax (I)

Comment	"a comment"
Character	\$a
String	'a nice string'
Symbol	#something
Integer	123
Float	3.14
Array	#{ 1 2 3 }
Boolean	true false
Undefined	nil

Syntax (II)

Temporary Variable	<code>var</code>
Assignment	<code>var := aNumber</code>
Block	[<code>expr ...</code>] [<code>:arg1</code> <code>expr ...</code>] [<code>:arg1 :arg2</code> <code>expr ...</code>]
1. Unary message	receiver selector
2. Binary message	receiver selector argument
3. Keyword message	receiver selector: argument receiver add: <code>2</code> after: <code>1</code>

Syntax (II)

Cascade

receiver selector1; selector2

receiver add: 2 after: 1

Statement

expr1. expr2

expr1. expr2. expr3

Return

^ expr

Parenthesis

(expr)

Sending Messages to Object

-2 abs

1 + 2

1 + (2 * 3)

2 raisedTo: 3

Transcript

show: (1 + 2);

cr

Conditionals

```
(1 < aNumber)  
  ifTrue: [ self foo ].
```

```
(aNumber = 5)  
  ifFalse: [ self bar ].
```

```
aBoolean  
  ifTrue: [ self doThis ]  
  ifFalse: [ self doThat ]
```

Interval-Loops

```
1 to: 10 do: [ :each |  
    Transcript show: each ].
```

```
1 to: 10 by: 2 do: [ :each |  
    Transcript show: each ].
```

```
10 to: 1 by: -1 do: [ :each |  
    Transcript show: each ]
```

While-Loops

```
| x |  
x := 0.
```

```
[ x < 10 ] whileTrue: [  
  Transcript show: x.  
  x := x + 1 ].
```

```
[ x < 0 ] whileFalse: [  
  Transcript show: x.  
  x := x - 1 ]
```

Classes

Class Definition

```
NameOfSuperclass subclass: #NameOfClass  
  instanceVariableNames: 'instVar1 instVar2'  
  classVariableNames: ''  
  poolDictionaries: ''  
  category: 'Category-Name'
```

System Browser: Set

Collections-Unordered	Bag	-- all --
Collections-Weak	IdentityBag	*Tools-Inspector
Collections-Stack	Matrix	*ob-tools-inspector
CollectionsTests-Abstrac	Set	accessing
CollectionsTests-Arrayec	Dictionary	adding
CollectionsTests-Sequen	IdentityDictionary	converting
CollectionsTests-Stream	PluggableDictionary	copying
CollectionsTests-Support	IdentitySet	enumerating
CollectionsTests-Text	KeyedSet	explorer
CollectionsTests-Unorde	instance	objects from disk
	?	class

browse | hierarchy | variables | implementors | inheritance | senders | versions

Collection subclass: #Set
instanceVariableNames: 'tally array'
classVariableNames: ''
poolDictionaries: ''
category: 'Collections-Unordered'

Method Definition

factorial

```
self = 1 if True: [ ^ self ].
```

```
self > 0 if True: [ ^ self * ((self - 1) factorial) ].
```

```
self error: 'No negative number please'
```

System Browser: Set

<ul style="list-style-type: none"> Collections-Unordered Collections-Weak Collections-Stack CollectionsTests-Abstrac CollectionsTests-Arrayec CollectionsTests-Sequen CollectionsTests-Stream CollectionsTests-Support CollectionsTests-Text CollectionsTests-Unorde 	<ul style="list-style-type: none"> Bag IdentityBag Matrix Set Dictionary IdentityDictionary PluggableDictionary IdentitySet KeyedSet 	<ul style="list-style-type: none"> -- all -- *Tools-Inspector *ob-tools-inspector accessing adding converting copying enumerating explorer objects from disk 	<ul style="list-style-type: none"> collect: do: doWithIndex: union:
--	--	---	--

instance ? class

browse hierarchy variables implementors inheritance senders versions

collect: aBlock

"Evaluate aBlock with each of the receiver's elements as the argument. Collect the resulting values into a collection like the receiver. Answer the new collection."

```

| newSet |
newSet _ Set new: self size.
array do: [:each | each ifNotNil: [newSet add: (aBlock value: each)]].
^ newSet

```

Instance Creation

- Literals:

123456

'abc'

- Basic class creation messages:

Monster new

Array new: 10

- Specific constructor messages:

Array with: 1 with: 2

ReadStream on: 'abc'

Tools

Monticello

Source Code Management

Squeak Source

Public Code Repository

OmniBrowser

Meta-Driven Code Browser

Other Software Engineering Tools

Refactoring Tools
Unit Testing
Code Coverage
Code Critics
Shout
eCompletion

Seaside

Web Application Framework

Pier

Content Management System

Magritte

Generic Meta Framework

Join and have Fun

www.squeak.org