11. A bit of Smalltalk

Oscar Nierstrasz

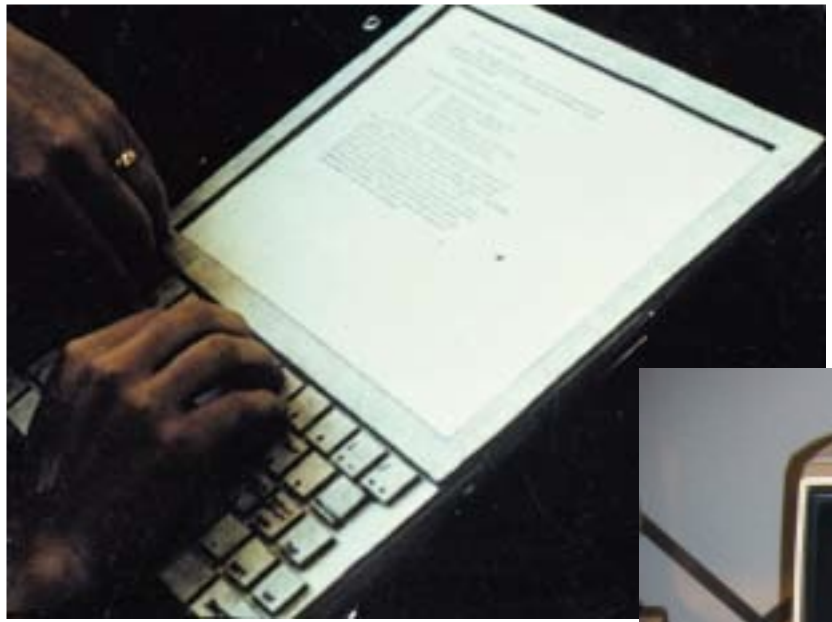
Roadmap

- The origins of Smalltalk
- What is Smalltalk?
- Syntax in a nutshell
- Seaside — web development with Smalltalk

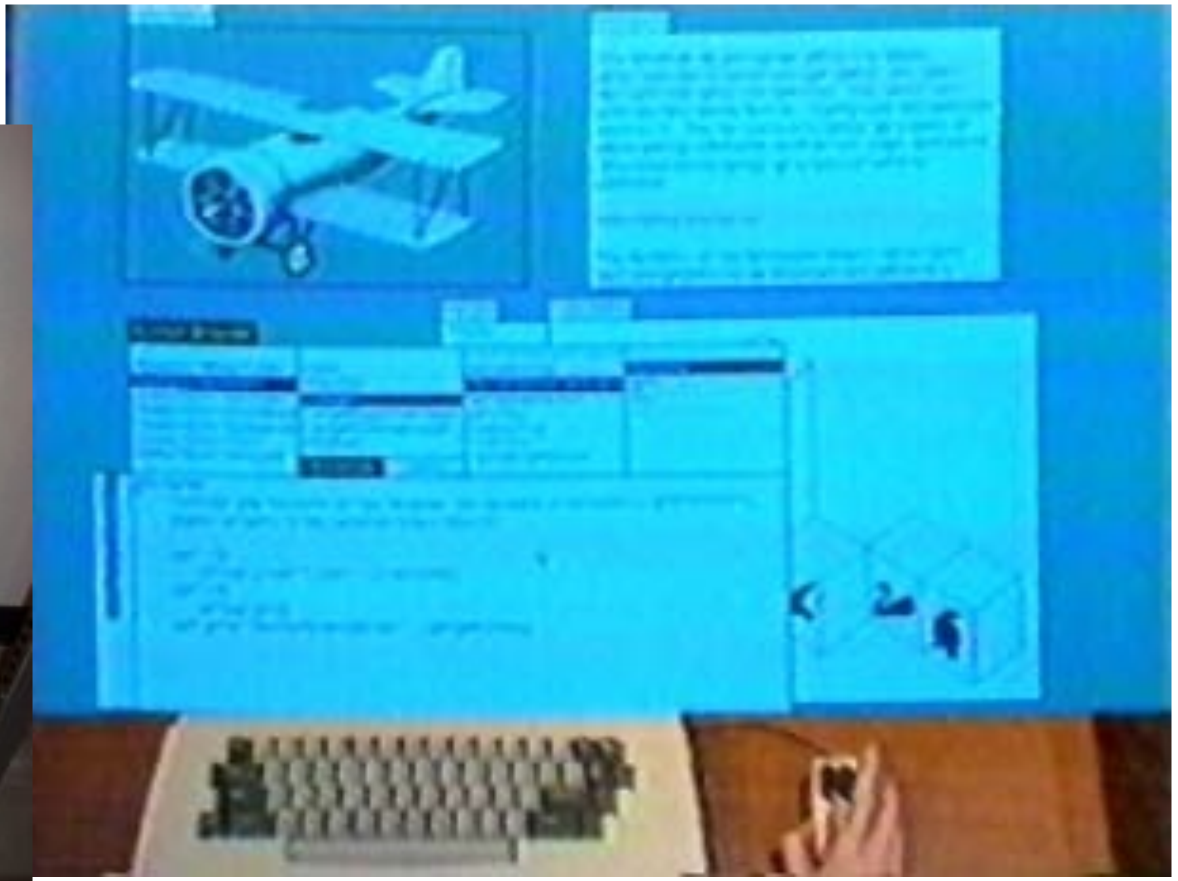
Roadmap

> **The origins of Smalltalk**
> What is Smalltalk?
> Syntax in a nutshell
> Seaside — web development with Smalltalk

The origins of Smalltalk



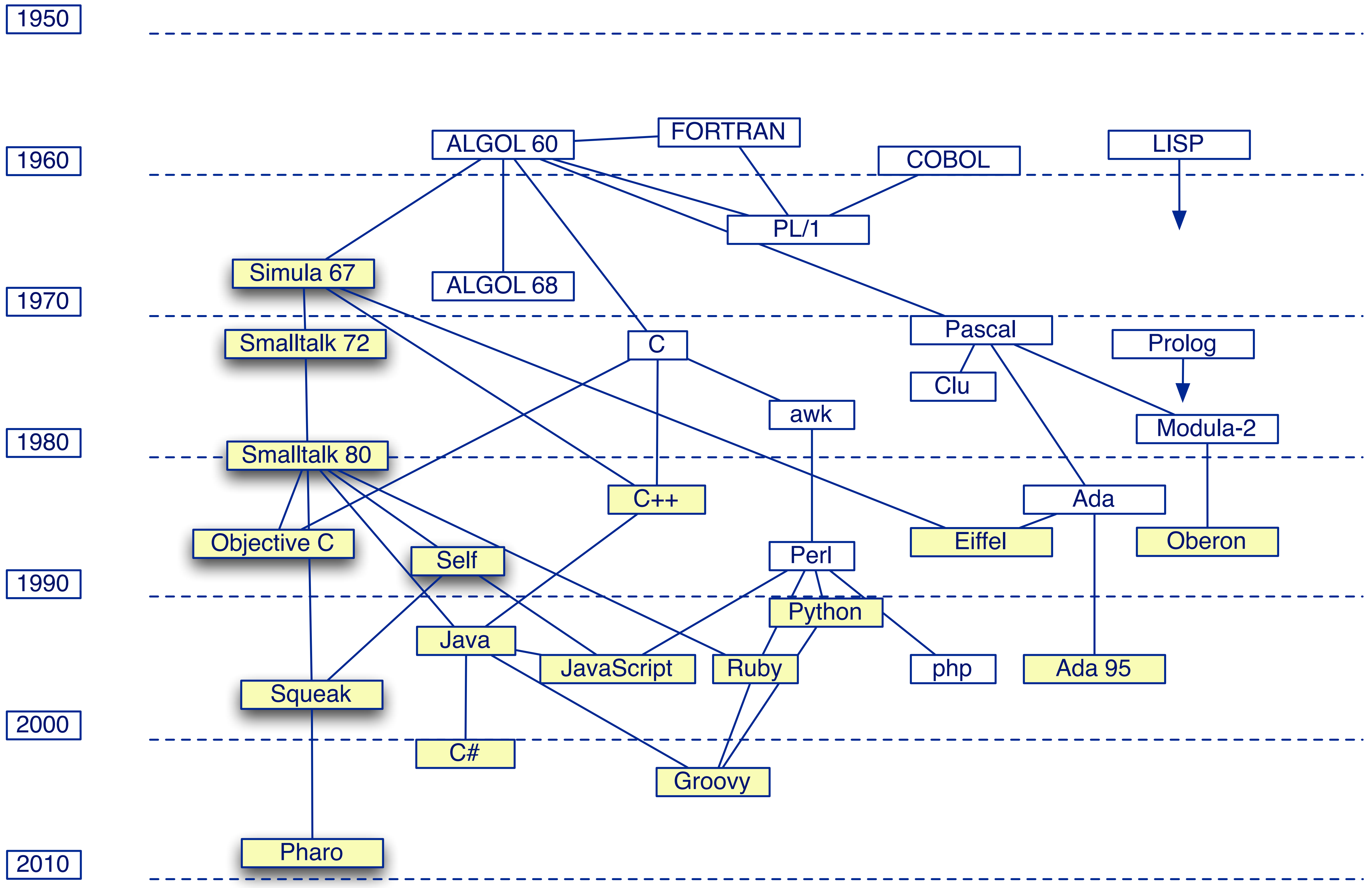
Alan Kay’s Dynabook project (1968)



Alto — Xerox PARC (1973)

gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html

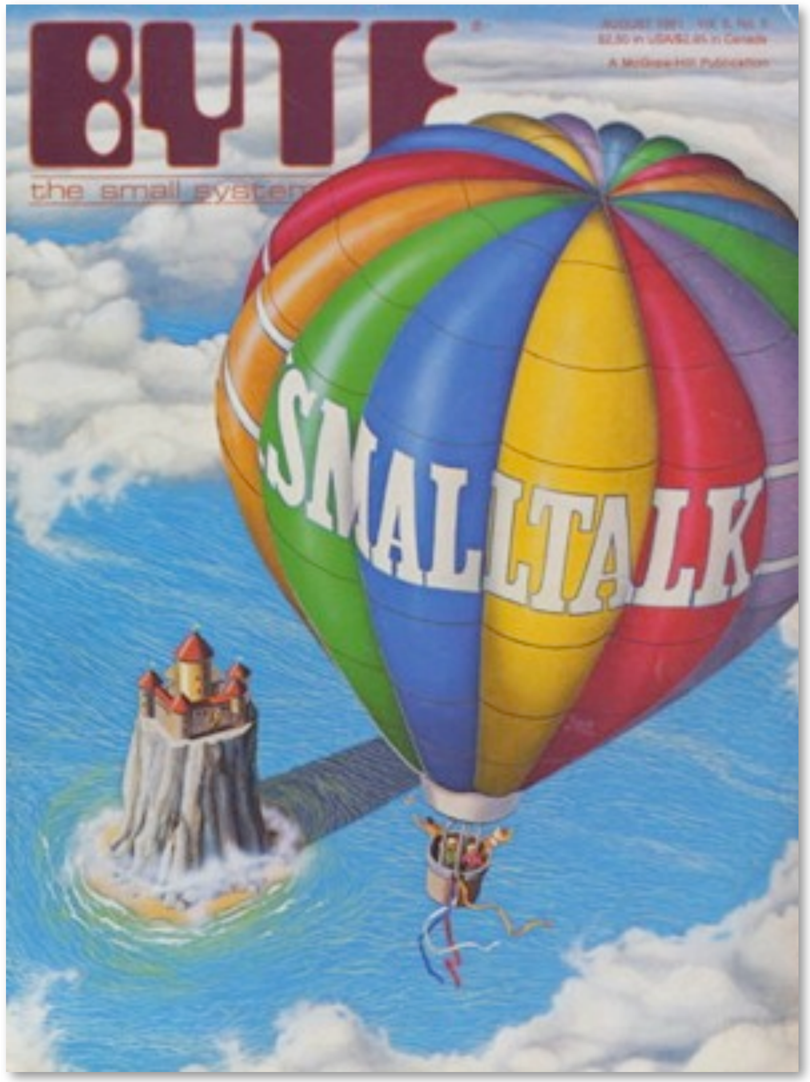
Object-oriented language genealogy



Smalltalk vs. Java vs. C++

	Smalltalk	***Java***	***C++***
Object model	Pure	Hybrid	Hybrid
Garbage collection	Automatic	Automatic	Manual
Inheritance	Single	Single	Multiple
Types	Dynamic	Static	Static
Reflection	Fully reflective	Introspection	Introspection
Concurrency	Semaphores	Monitors	Some libraries
Modules	Categories, namespaces	Packages	Namespaces

Smalltalk-80 and Pharo

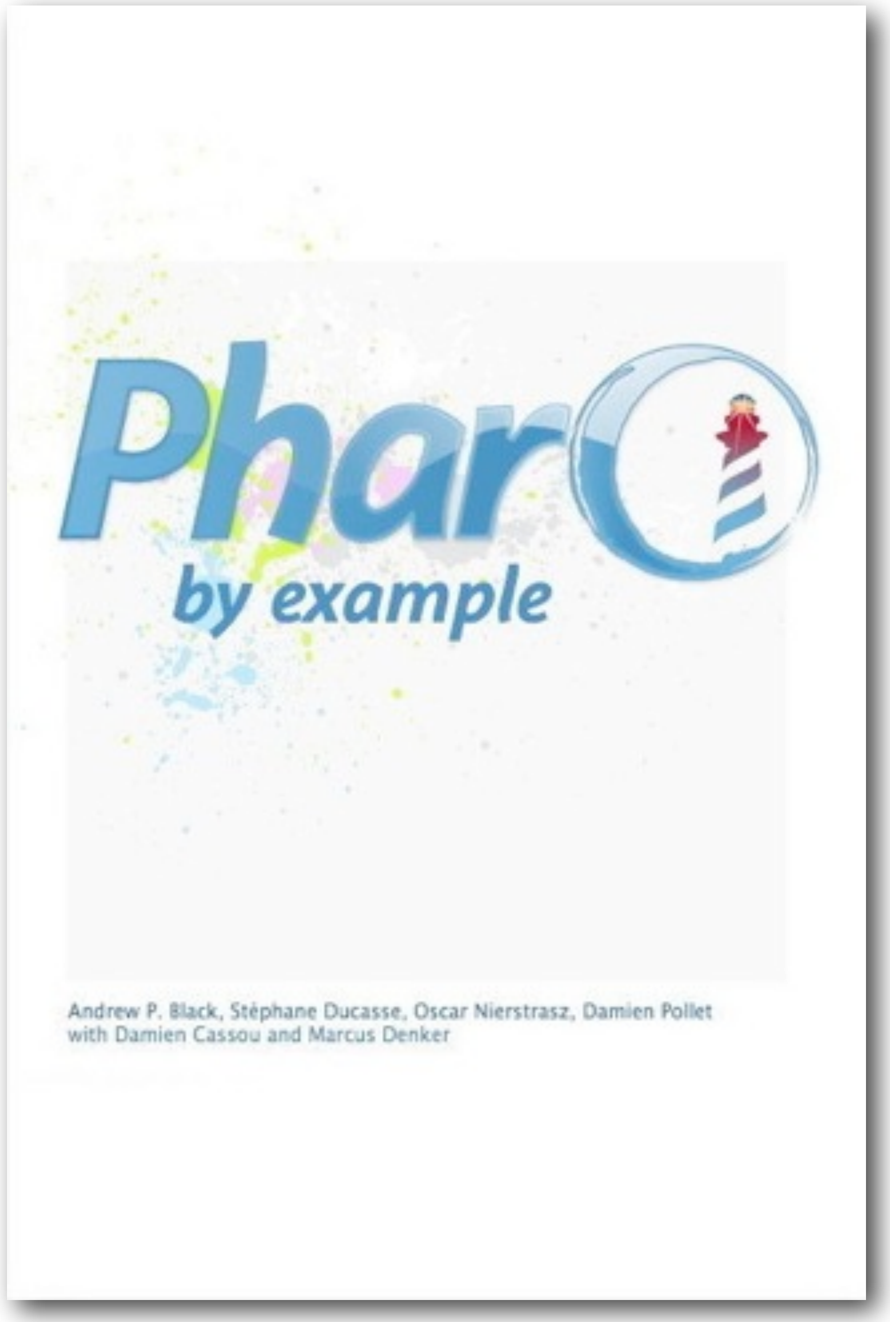


- Everything is an object
- Everything is there, all the time
- First windowing system with mouse
- First graphical IDE

What are Squeak and Pharo?

- Squeak is a modern, open-source, highly portable, fast, full-featured Smalltalk implementation
  - —Based on original Smalltalk-80 code
- Pharo is a lean and clean fork of Squeak
  - —www.pharo-project.org

Pharo by Example



Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz, Damien Pollet
with Damien Cassou and Marcus Denker

http://pharobyexample.org/

- Free download
- Open-Source
- Print-on-demand

Don’t panic!

New Smalltalkers often think they need to understand all the details of a thing before they can use it.

Try to answer the question

“How does this work?”

with

“I don’t care”.

Alan Knight. Smalltalk Guru

Roadmap

> The origins of Smalltalk
> **What is Smalltalk?**
> Syntax in a nutshell
> Seaside — web development with Smalltalk

Two rules to remember

Everything is an object

Everything happens by sending messages

What is Smalltalk?

Image

Welcome to Squeak - an open Smalltalk system.

What is this?
Squeak is an open source implementation of Smalltalk with an expansive and rapidly developing world of objects included.
Squeak includes a fully integrated development environment, networking, sound synthesis and sampling, speech synthesis, 2 & 3D graphics, arithmetic and data structure libraries.
The SqueakMap tool (click on -> "SMLoader open" to start it) tool provides easy access to a large range of packages and projects.
Even the tools to produce the its own virtual machine (VM) are available so that you can build your own - and most of the VM is written in Smalltalk. Not only is all the Smalltalk source code included and changeable at will, it is also completely open and free. The Squeak system image runs bit-identically across all platforms, and VMs are available for just about every computer and operating system available.

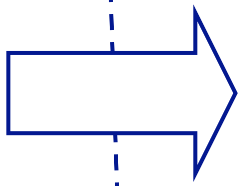
To find out exactly which version of Squeak this is click on -> "SystemVersion current"

Further Documentation

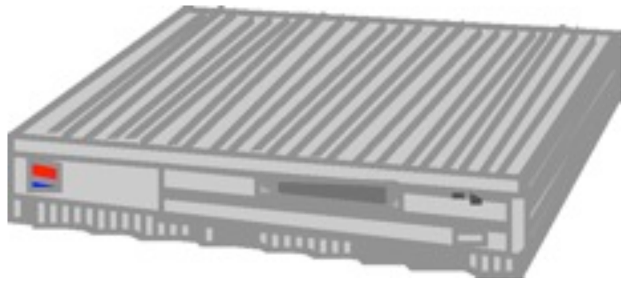
[illegible]

+

Changes



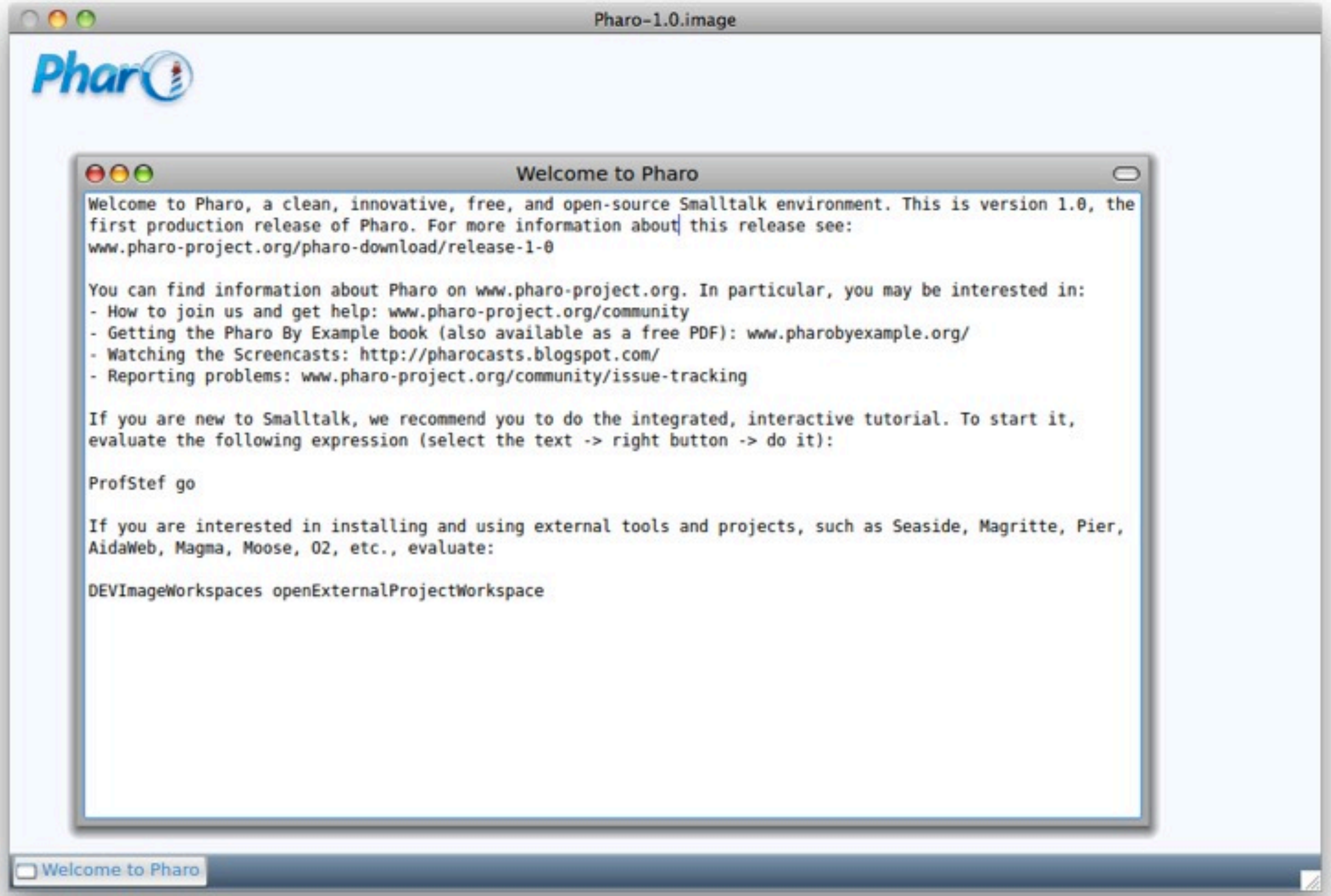
Virtual machine



+

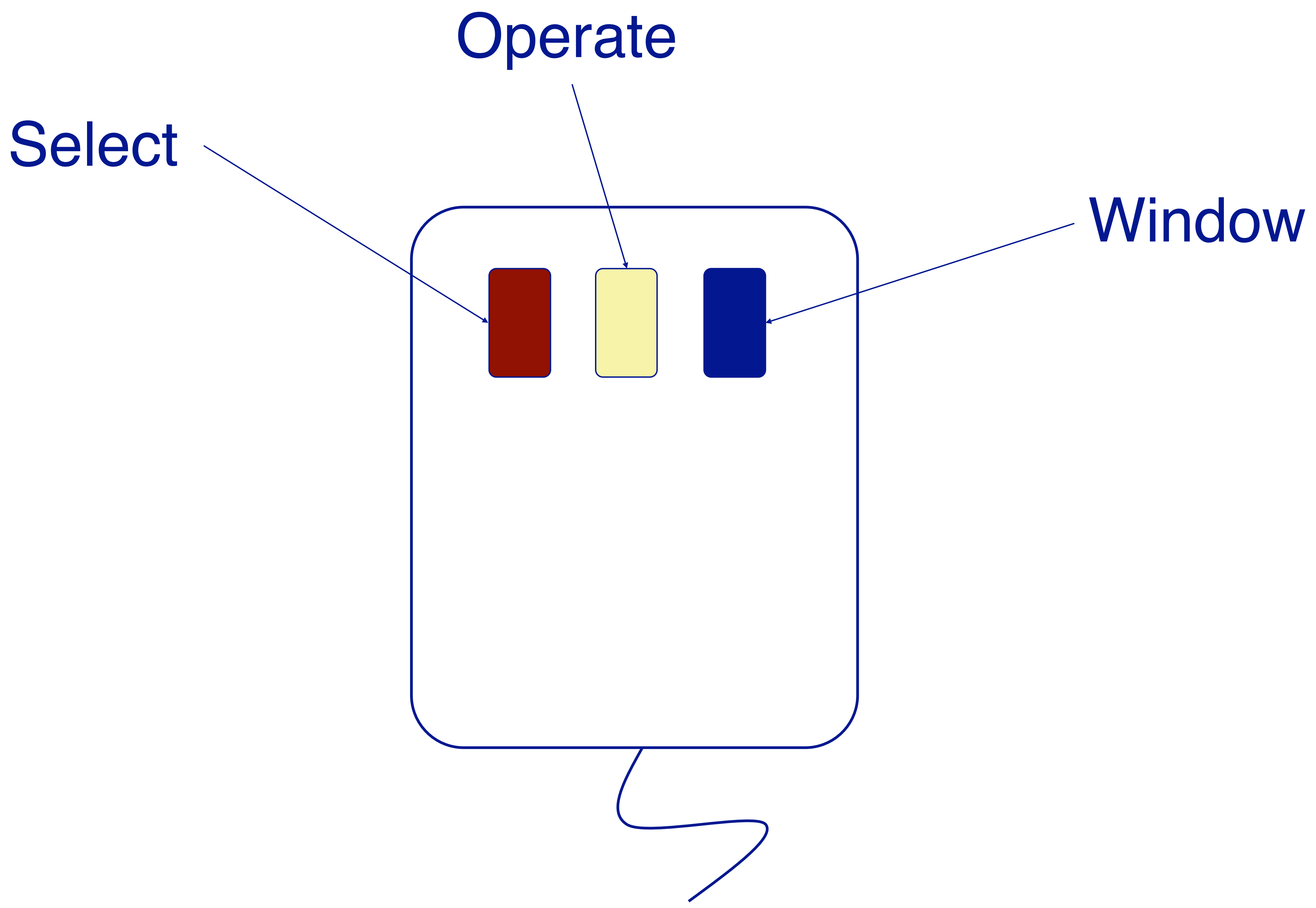
Sources

Demo: Running Pharo

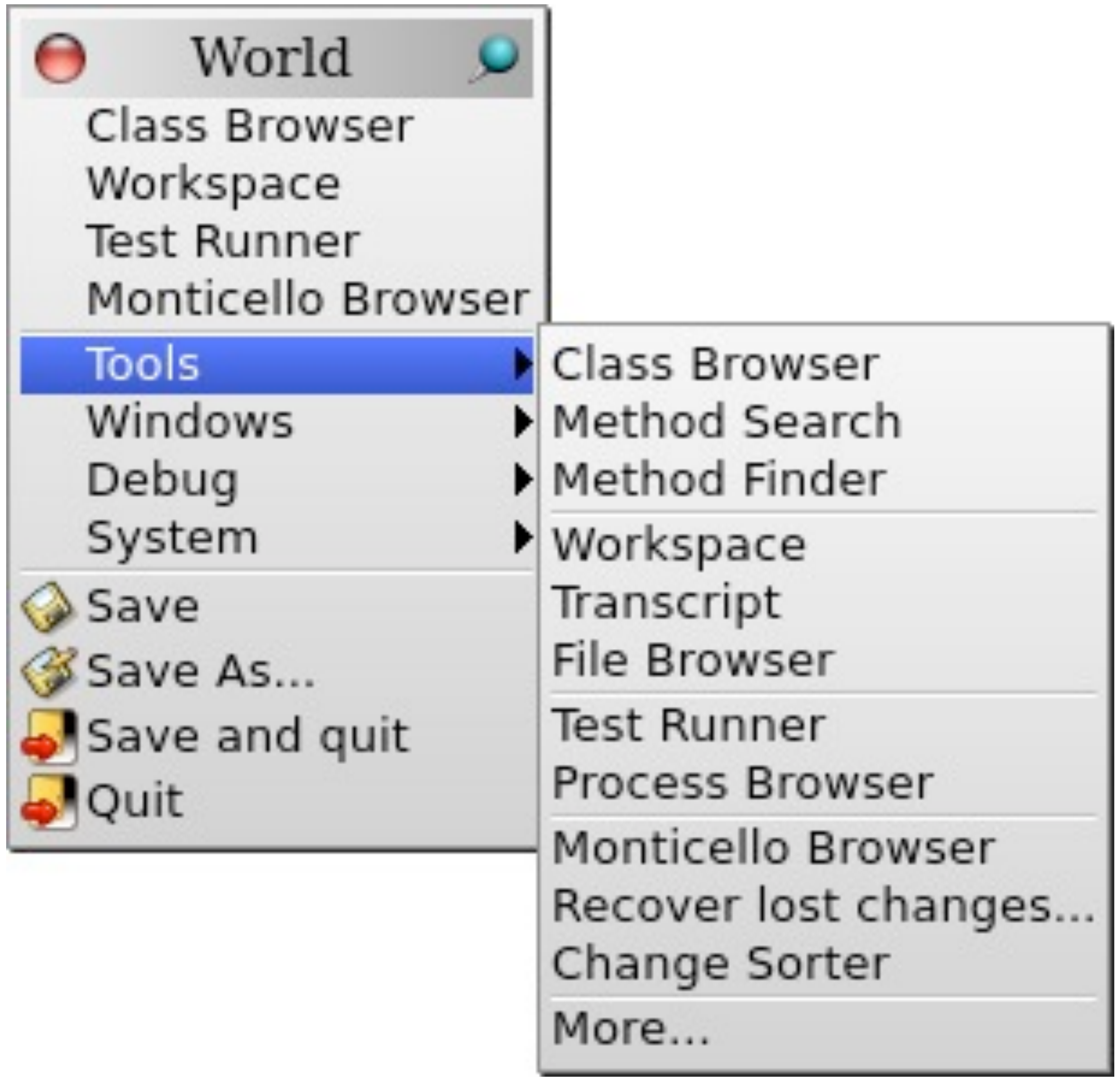


```
Welcome to Pharo, a clean, innovative, free, and open-source Smalltalk environment. This is version 1.0, the
first production release of Pharo. For more information about this release see:
www.pharo-project.org/pharo-download/release-1-0

You can find information about Pharo on www.pharo-project.org. In particular, you may be interested in:
- How to join us and get help: www.pharo-project.org/community
- Getting the Pharo By Example book (also available as a free PDF): www.pharobyexample.org/
- Watching the Screencasts: http://pharocasts.blogspot.com/
- Reporting problems: www.pharo-project.org/community/issue-tracking

If you are new to Smalltalk, we recommend you to do the integrated, interactive tutorial. To start it,
evaluate the following expression (select the text -> right button -> do it):

ProfStef go

If you are interested in installing and using external tools and projects, such as Seaside, Magritte, Pier,
AidaWeb, Magma, Moose, O2, etc., evaluate:

DEVImageWorkspaces openExternalProjectWorkspace
```

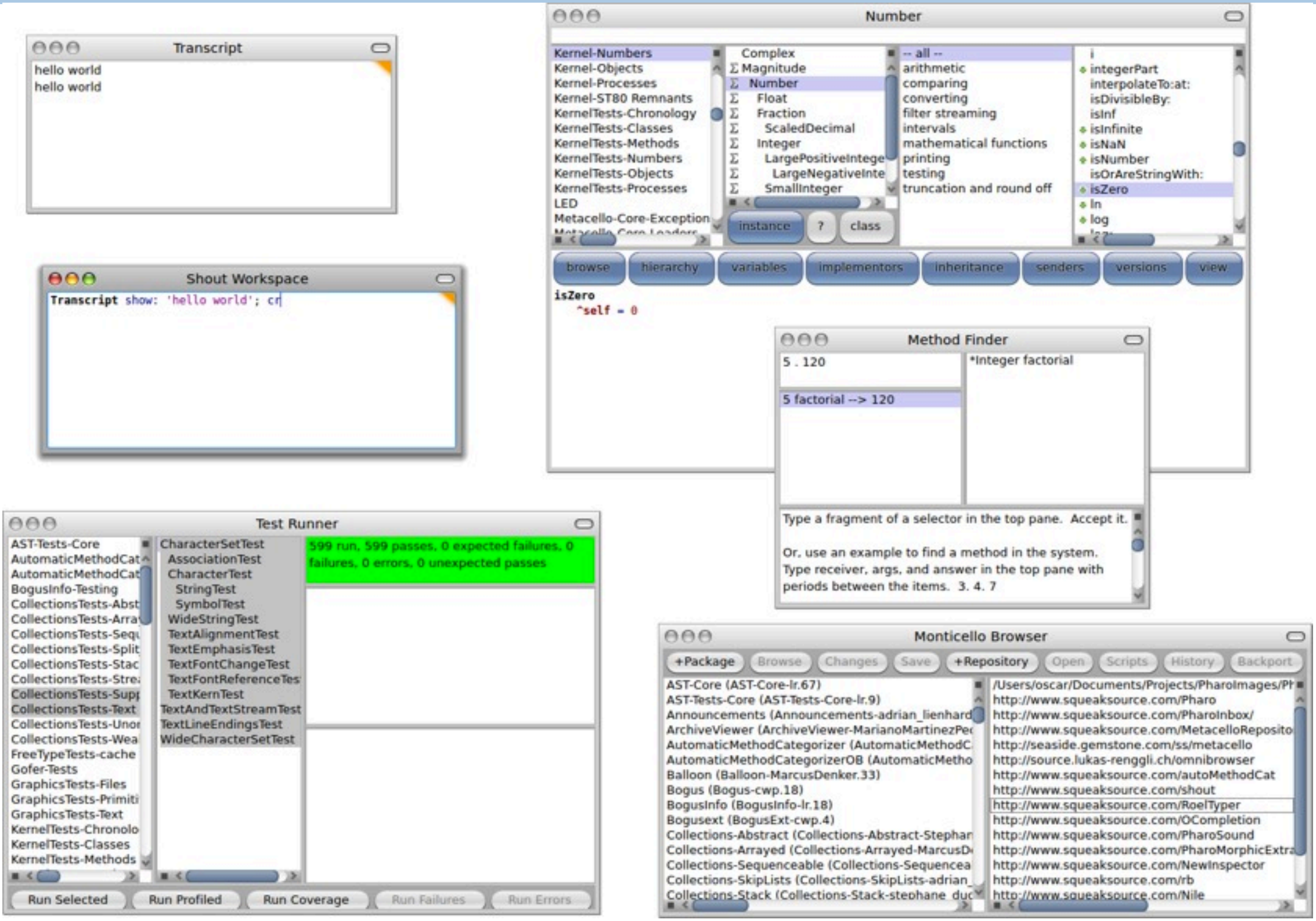
Mouse Semantics



World Menu


World
Class Browser
Workspace
Test Runner
Monticello Browser
Tools
Windows
Debug
System
Save
Save As...
Save and quit
Quit
Class Browser
Method Search
Method Finder
Workspace
Transcript
File Browser
Test Runner
Process Browser
Monticello Browser
Recover lost changes...
Change Sorter
More...

Standard development tools



hello world
hello world

```
Transcript show: 'hello world'; cr
```

```
isZero
    ^self = 0
```

5 . 120

5 factorial --> 120

*Integer factorial

Type a fragment of a selector in the top pane. Accept it.

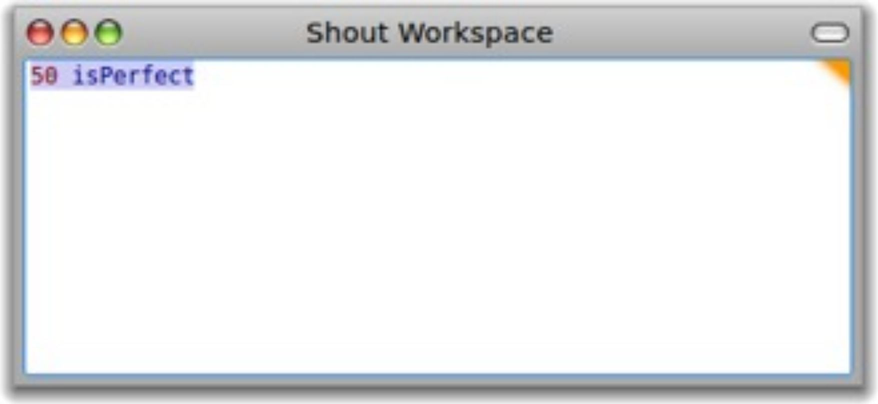
Or, use an example to find a method in the system. Type receiver, args, and answer in the top pane with periods between the items. 3. 4. 7

599 run, 599 passes, 0 expected failures, 0 failures, 0 errors, 0 unexpected passes

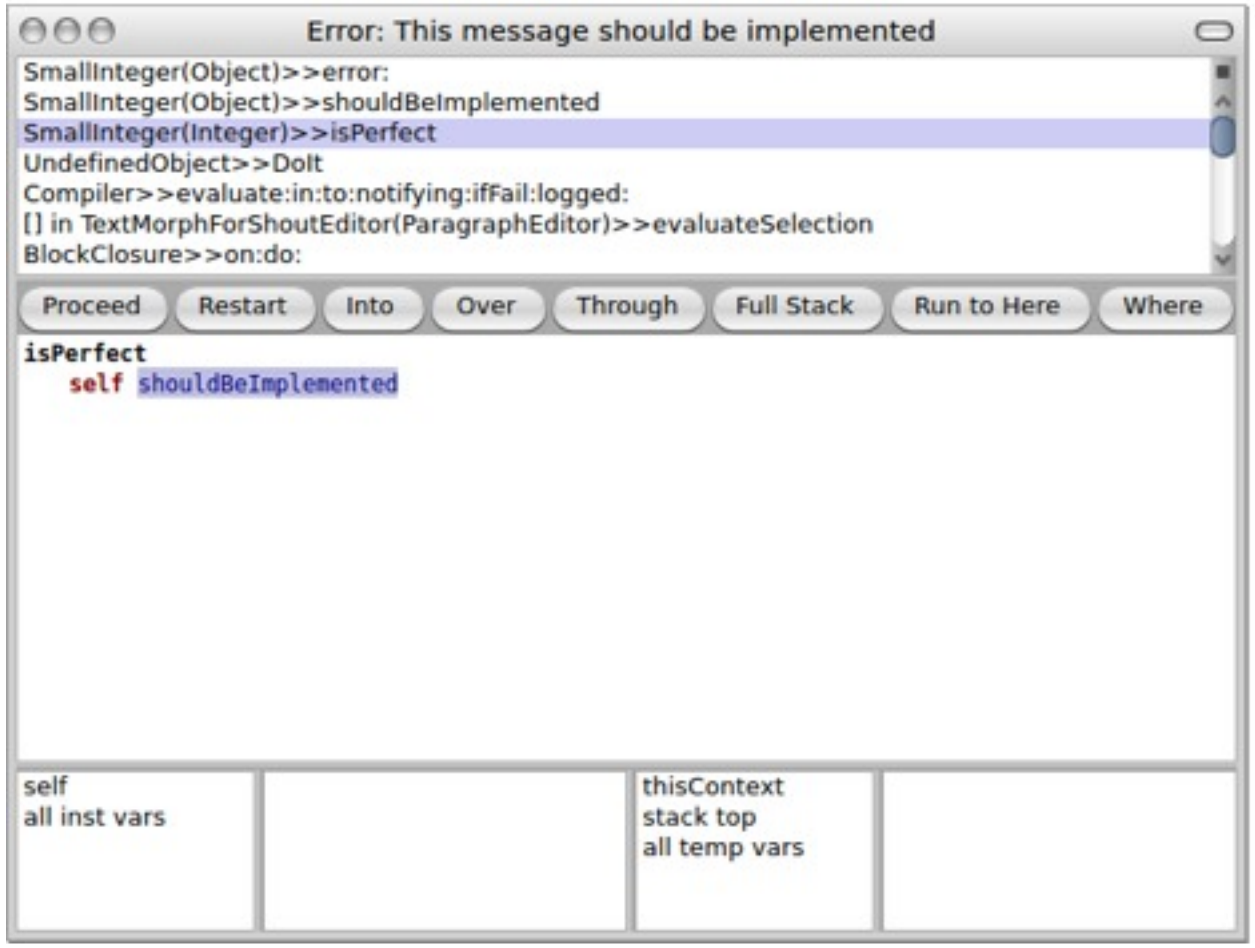
AST-Core (AST-Core-lr.67)
AST-Tests-Core (AST-Tests-Core-lr.9)
Announcements (Announcements-adrian_lienhard
ArchiveViewer (ArchiveViewer-MarianoMartinezPec
AutomaticMethodCategorizer (AutomaticMethodC
AutomaticMethodCategorizerOB (AutomaticMetho
Balloon (Balloon-MarcusDenker.33)
Bogus (Bogus-cwp.18)
BogusInfo (BogusInfo-lr.18)
Bogusext (BogusExt-cwp.4)
Collections-Abstract (Collections-Abstract-Stephan
Collections-Arrayed (Collections-Arrayed-MarcusD
Collections-Sequenceable (Collections-Sequencea
Collections-SkipLists (Collections-SkipLists-adrian_
Collections-Stack (Collections-Stack-stephane_duc

/Users/oscar/Documents/Projects/PharoImages/Ph
http://www.squeaksource.com/Pharo
http://www.squeaksource.com/PharoInbox/
http://www.squeaksource.com/MetacelloReposito
http://seaside.gemstone.com/ss/metacello
http://source.lukas-renggli.ch/omnibrowser
http://www.squeaksource.com/autoMethodCat
http://www.squeaksource.com/shout
http://www.squeaksource.com/RoelTyper
http://www.squeaksource.com/OCompletion
http://www.squeaksource.com/PharoSound
http://www.squeaksource.com/PharoMorphicExtra
http://www.squeaksource.com/NewInspector
http://www.squeaksource.com/rb
http://www.squeaksource.com/Nile

Debuggers, Inspectors, Explorers

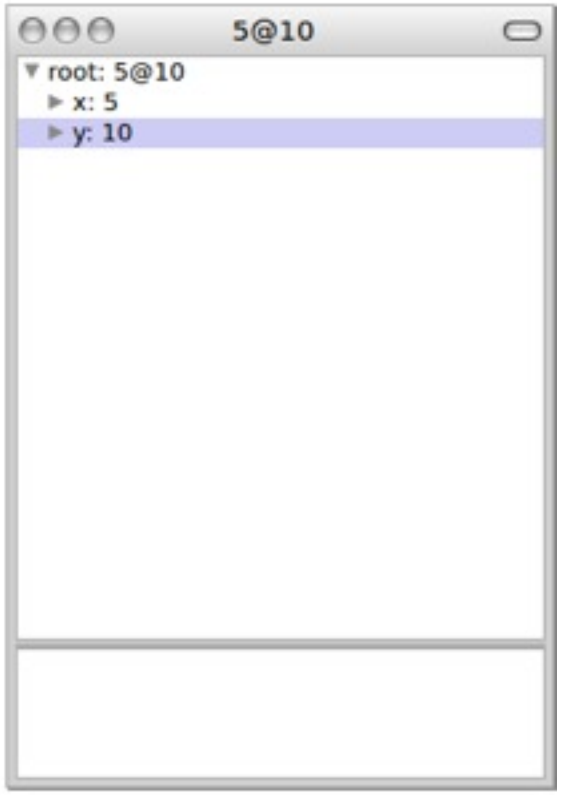


```
50 isPerfect
```



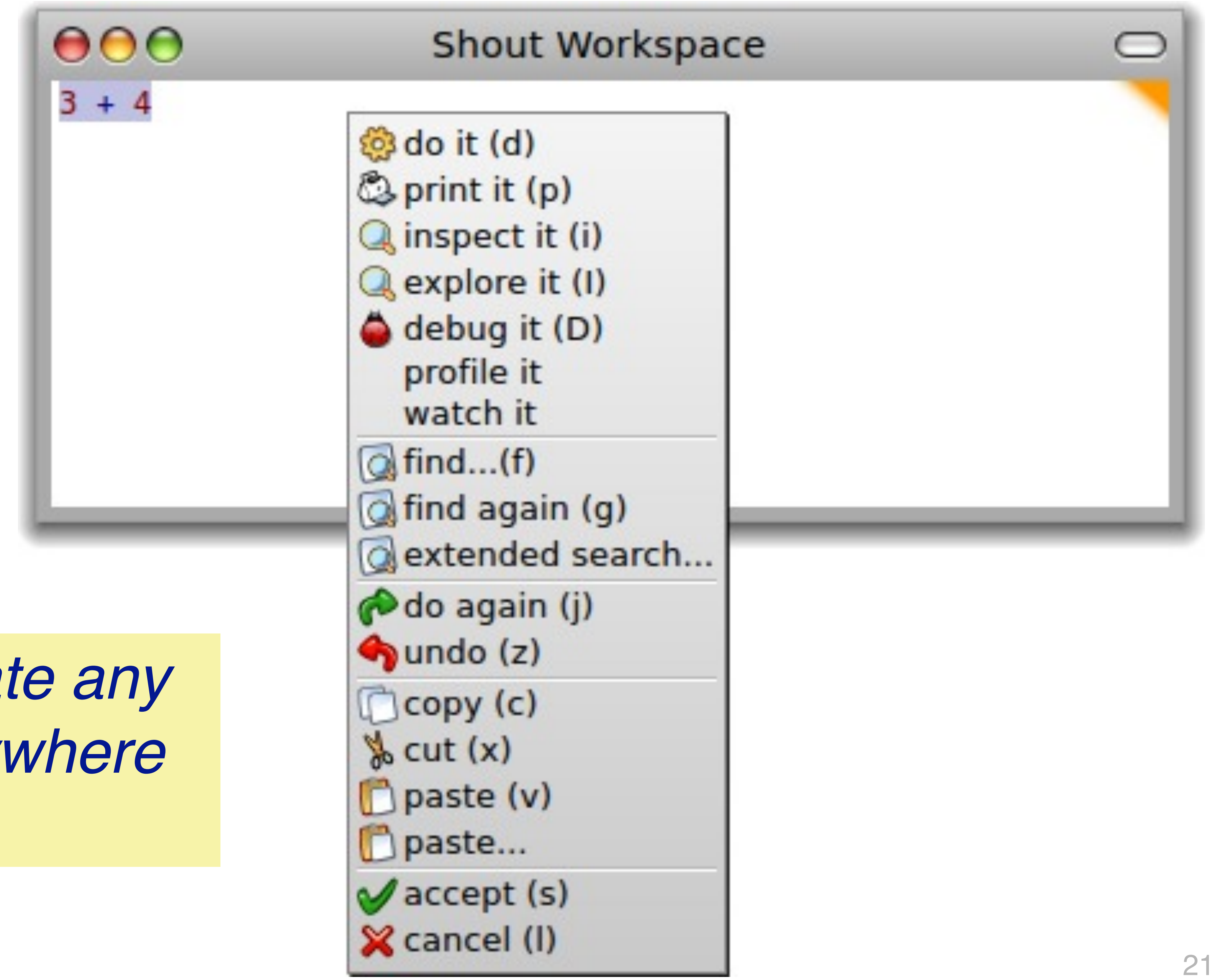
```
SmallInteger(Object)>>error:
SmallInteger(Object)>>shouldBeImplemented
SmallInteger(Integer)>>isPerfect
UndefinedObject>>DoIt
Compiler>>evaluate:in:to:notifying:ifFail:logged:
[] in TextMorphForShoutEditor(ParagraphEditor)>>evaluateSelection
BlockClosure>>on:do:
```

```
isPerfect
    self shouldBeImplemented
```



```
▼ root: 5@10
  ▶ x: 5
  ▶ y: 10
```

Do it, Print it, …



You can evaluate any expression anywhere in Smalltalk.

Roadmap

- The origins of Smalltalk
- What is Smalltalk?
- **Syntax in a nutshell**
- Seaside — web development with Smalltalk

Three kinds of messages

- *Unary messages*

```
5 factorial
Transcript cr
```

- *Binary messages*

```
3 + 4
```

- *Keyword messages*

```
3 raisedTo: 10 modulo: 5

Transcript show: 'hello world'
```

Precedence

First unary, then binary, then keyword:

```
2 raisedTo: 1 + 3 factorial
```

Same as:

```
2 raisedTo: (1 + (3 factorial))
```

Use parentheses to force order:

```
1 + 2 * 3
1 + (2 * 3)
```

Precedence

First unary, then binary, then keyword:

```
2 raisedTo: 1 + 3 factorial
```

```
128
```

Same as:

```
2 raisedTo: (1 + (3 factorial))
```

Use parentheses to force order:

```
1 + 2 * 3
1 + (2 * 3)
```

Precedence

First unary, then binary, then keyword:

```
2 raisedTo: 1 + 3 factorial
```

128

Same as:

```
2 raisedTo: (1 + (3 factorial))
```

Use parentheses to force order:

```
1 + 2 * 3
1 + (2 * 3)
```

9 (!)

Precedence

First unary, then binary, then keyword:

```
2 raisedTo: 1 + 3 factorial
```

```
128
```

Same as:

```
2 raisedTo: (1 + (3 factorial))
```

Use parentheses to force order:

```
1 + 2 * 3
1 + (2 * 3)
```

```
9 (!)
7
```

A typical method in the class Point

```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point

Method name

```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point

Method name *Argument*

```
<= aPoint
  "Answer whether the receiver is neither
  below nor to the right of aPoint."

  ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point

Method name *Argument* *Comment*

```
<= aPoint
   "Answer whether the receiver is neither
  below nor to the right of aPoint."

  ^ x <= aPoint x and: [y <= aPoint y]
```

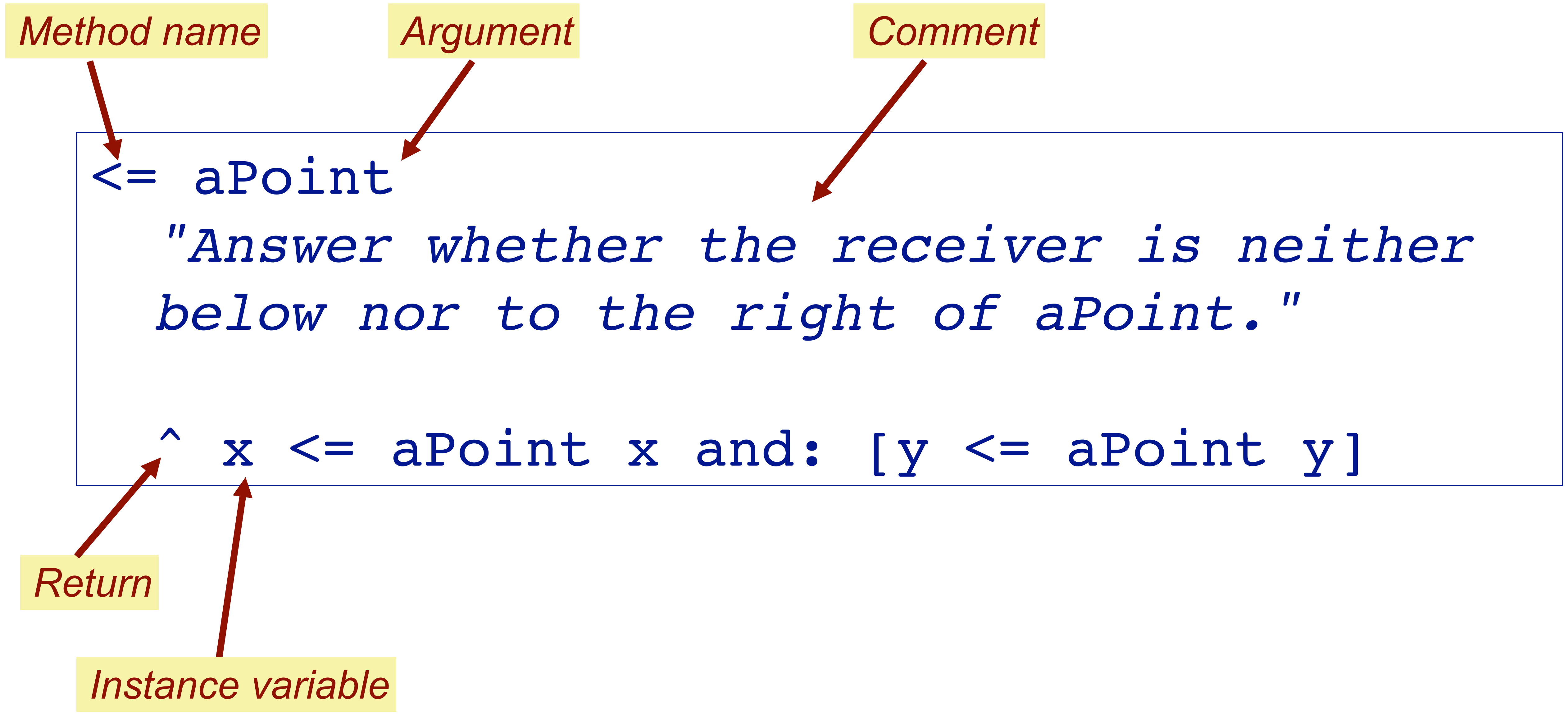
A typical method in the class Point

Method name *Argument* *Comment*

```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

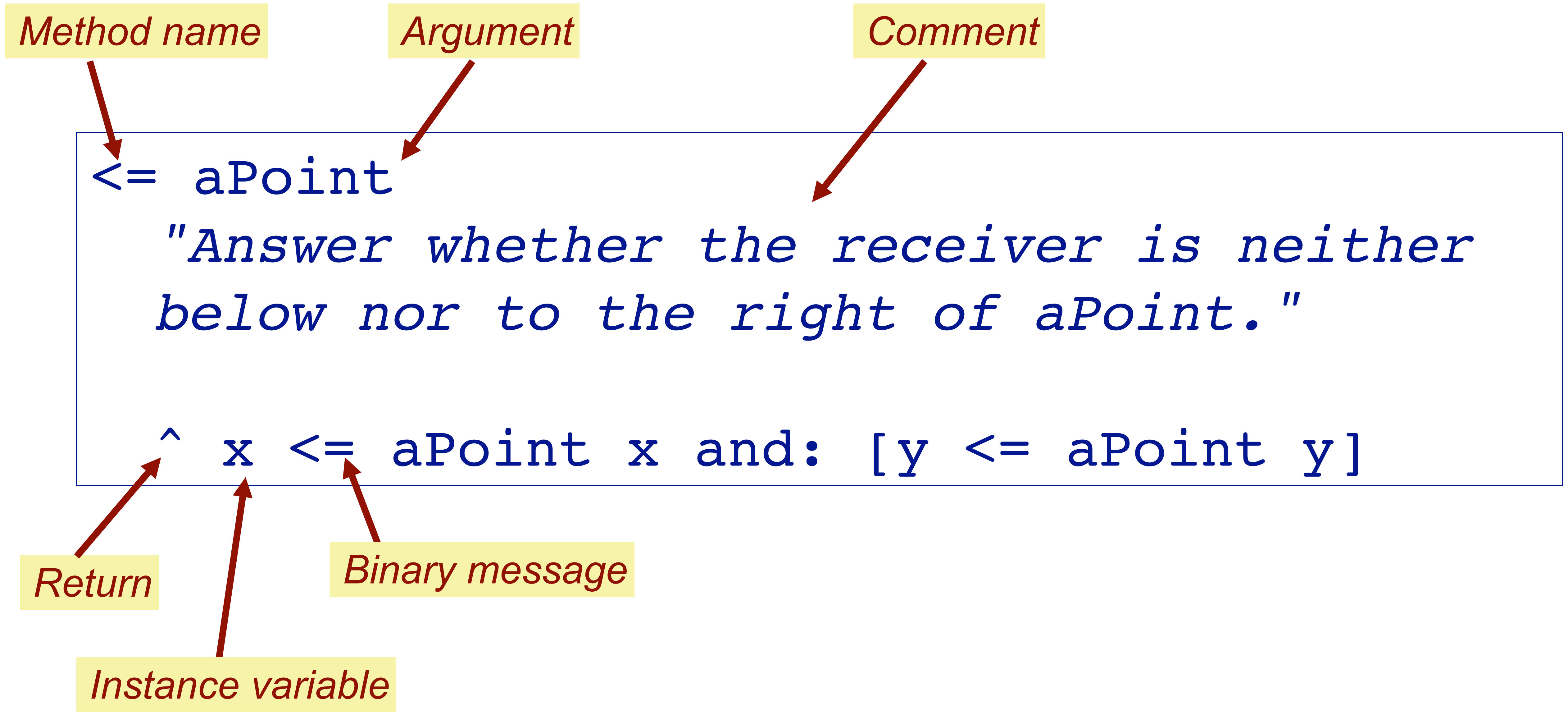
Return

A typical method in the class Point



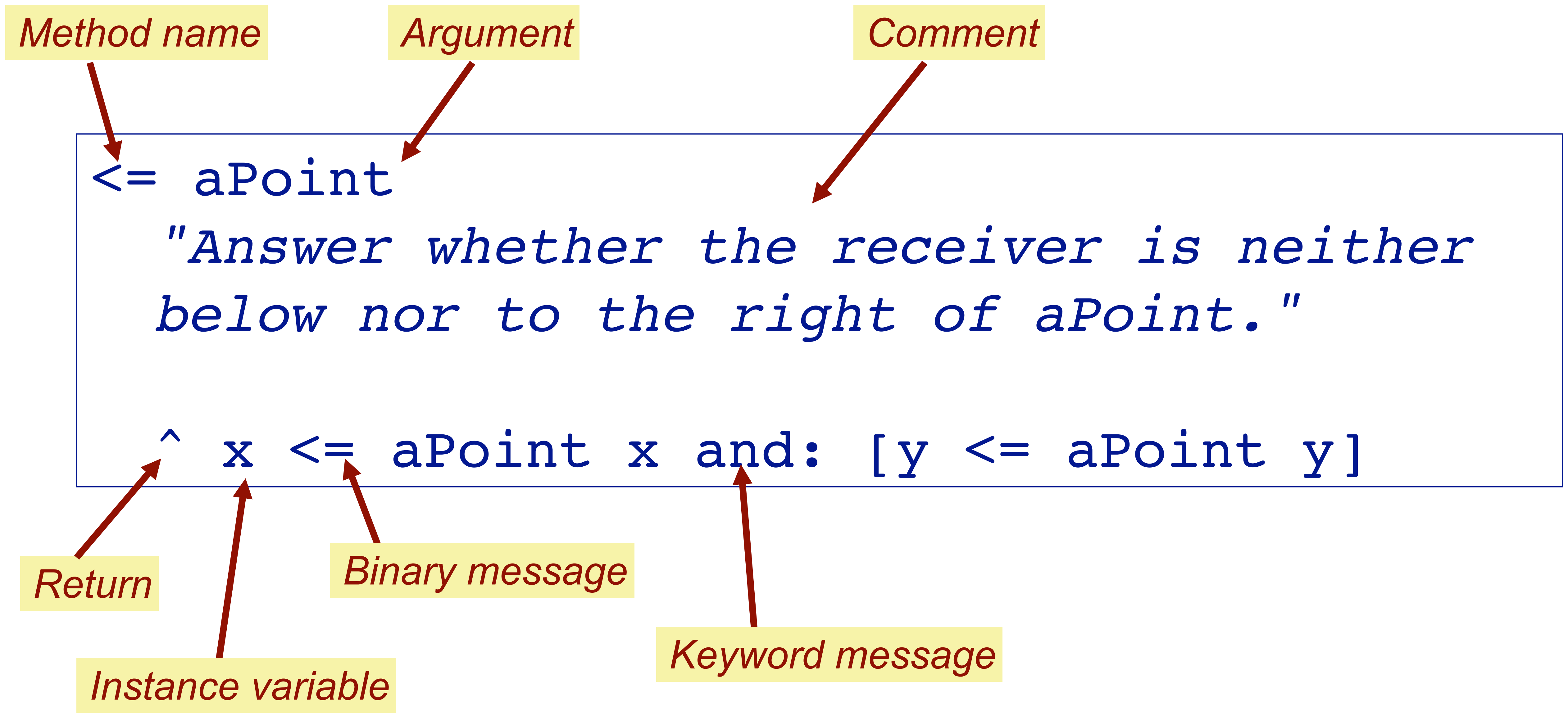
```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point



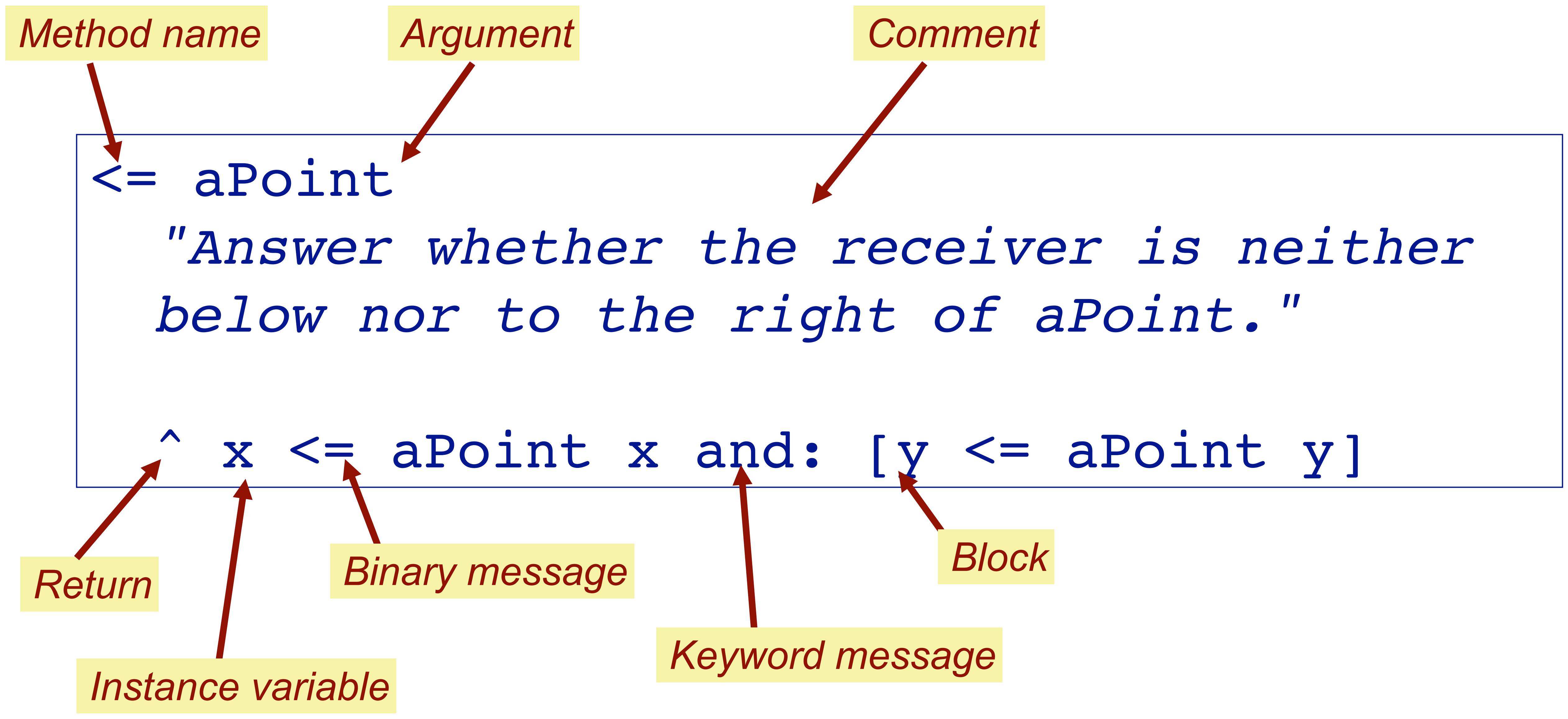
```
<= aPoint
  "Answer whether the receiver is neither
  below nor to the right of aPoint."

  ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point



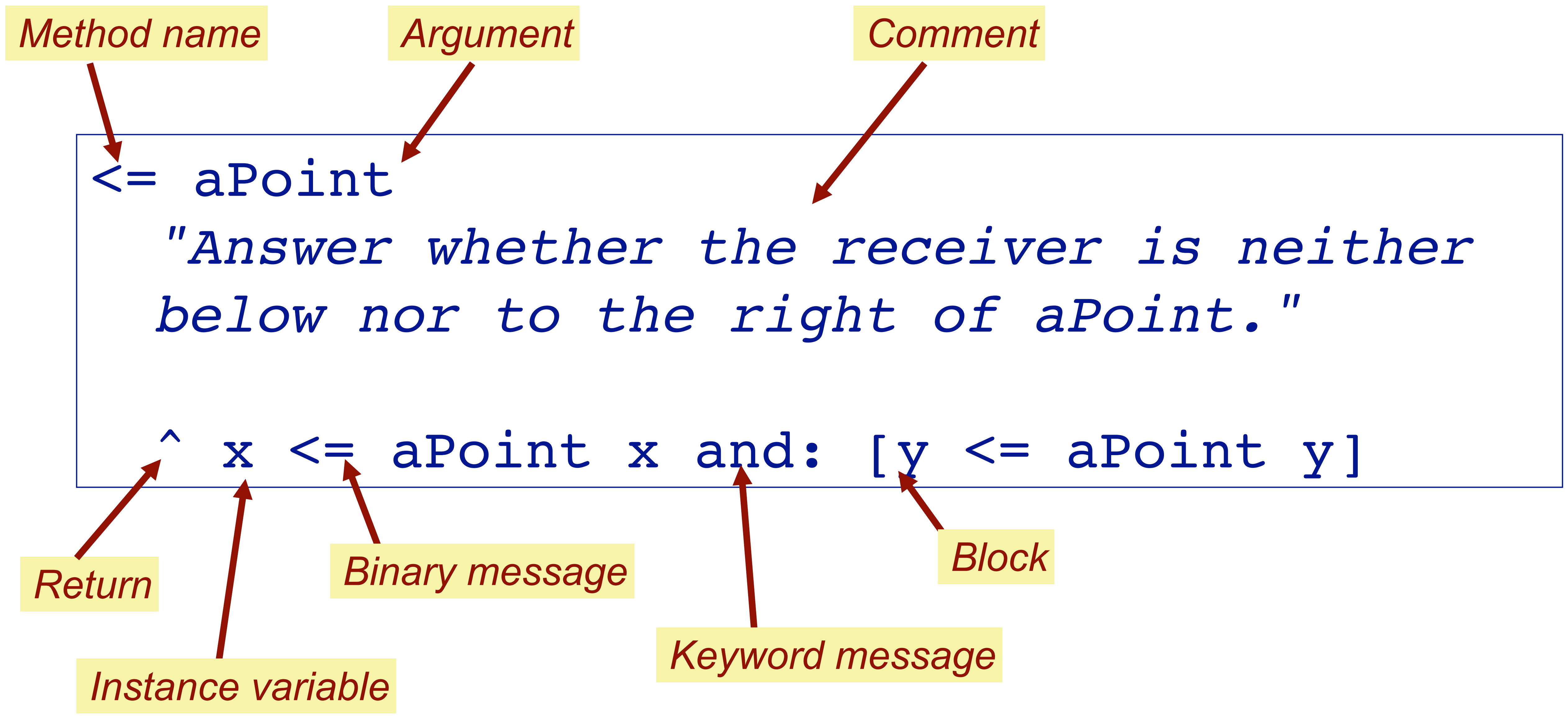
```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

A typical method in the class Point



```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

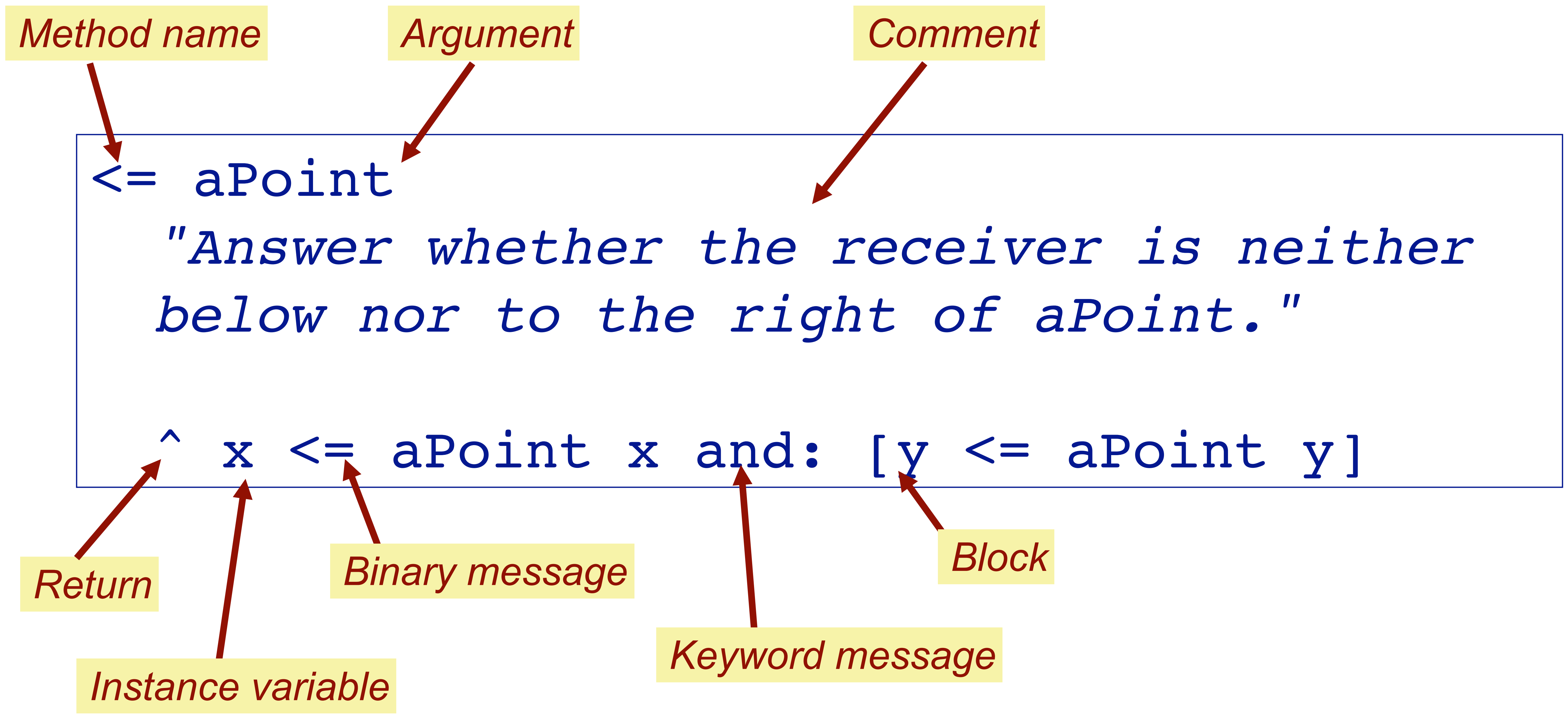
A typical method in the class Point



```
<= aPoint
  "Answer whether the receiver is neither
  below nor to the right of aPoint."

  ^ x <= aPoint x and: [y <= aPoint y]
```

```
(2@3) <= (5@6)
```

A typical method in the class Point



```
<= aPoint
   "Answer whether the receiver is neither
   below nor to the right of aPoint."

   ^ x <= aPoint x and: [y <= aPoint y]
```

```
(2@3) <= (5@6)
```

```
true
```

Statements and cascades

```
| p pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```

Statements and cascades

Temporary variables

```
| p pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```

Statements and cascades

Temporary variables

Assignment

```
| p pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```

Statements and cascades

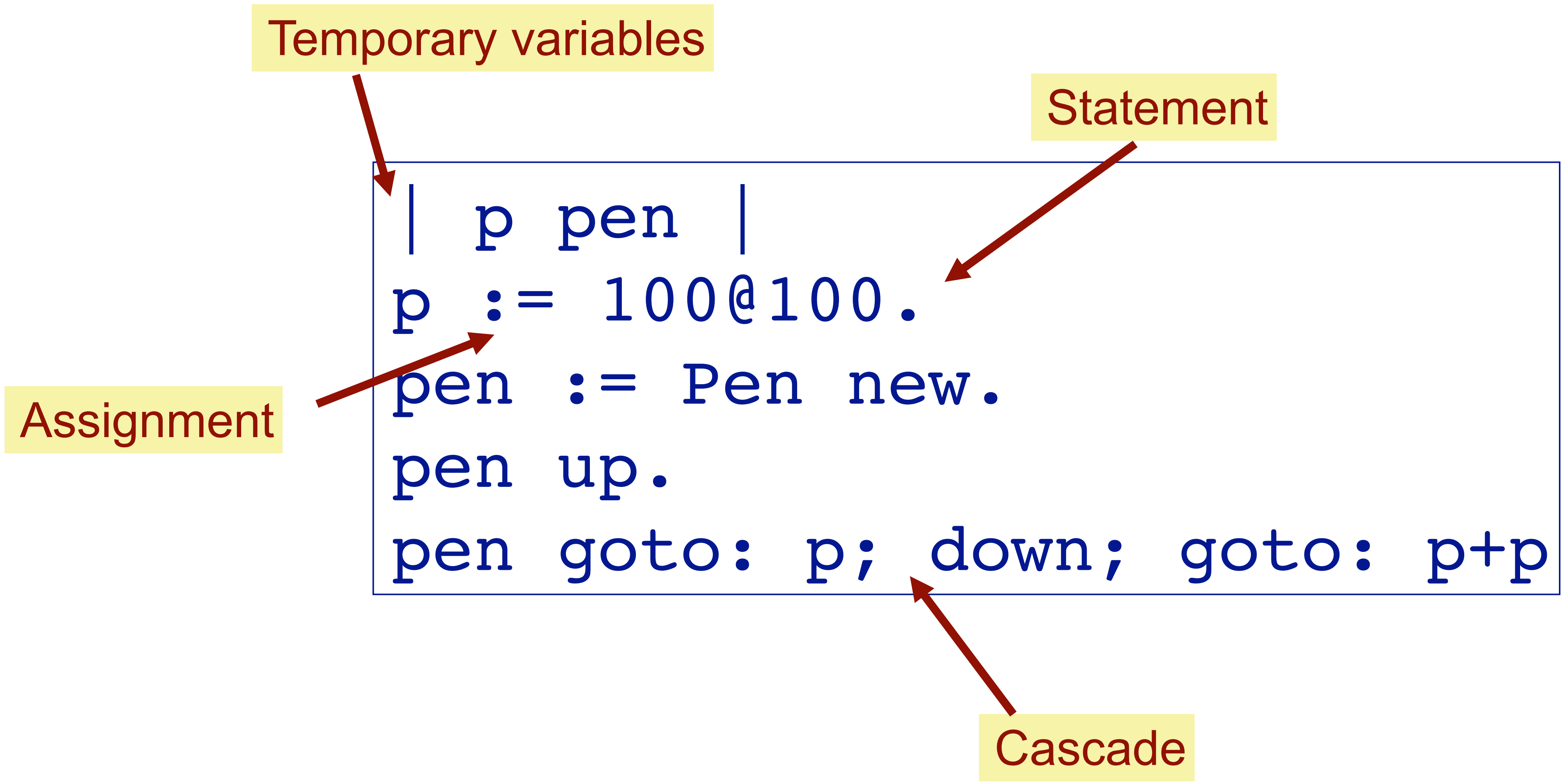
Temporary variables

Statement

Assignment

```
| p pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```

Statements and cascades



```
| p pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```

Literals and constants

Strings & Characters	`'hello'    $a`
Numbers	`1    3.14159`
Symbols	`#yadayada`
Arrays	`#(1 2 3)`
Pseudo-variables	`self super`
Constants	`true false`

Variables

> *Local variables* are delimited by `|var|`
> *Block variables* by `:var|`

```
OrderedCollection>>collect: aBlock
    "Evaluate aBlock with each of my elements as the argument."
    | newCollection |
    newCollection := self species new: self size.
    firstIndex to: lastIndex do:
        [ :index |
        newCollection addLast: (aBlock value: (array at: index))].
    ^ newCollection
```

Variables

> *Local variables* are delimited by `|var|`
> *Block variables* by `:var|`

```
OrderedCollection>>collect: aBlock
	"Evaluate aBlock with each of my elements as the argument."
	| newCollection |
	newCollection := self species new: self size.
	firstIndex to: lastIndex do:
		[ :index |
		newCollection addLast: (aBlock value: (array at: index))].
	^ newCollection
```

```
(OrderedCollection with: 10 with: 5) collect: [:each| each factorial ]
```

Variables

> *Local variables* are delimited by `|var|`
> *Block variables* by `:var|`

```
OrderedCollection>>collect: aBlock
    "Evaluate aBlock with each of my elements as the argument."
    | newCollection |
    newCollection := self species new: self size.
    firstIndex to: lastIndex do:
        [ :index |
        newCollection addLast: (aBlock value: (array at: index))].
    ^ newCollection
```

```
(OrderedCollection with: 10 with: 5) collect: [:each| each factorial ]
```

```
an OrderedCollection(3628800 120)
```

Control Structures

> *Every control structure is realized by message sends*

```
max: aNumber
   ^ self < aNumber
     ifTrue: [aNumber]
     ifFalse: [self]
```

```
4 timesRepeat: [Beeper beep]
```

Creating objects

> *Class methods*

```
OrderedCollection new
Array with: 1 with: 2
```

> *Factory methods*

```
1@2
1/2
```

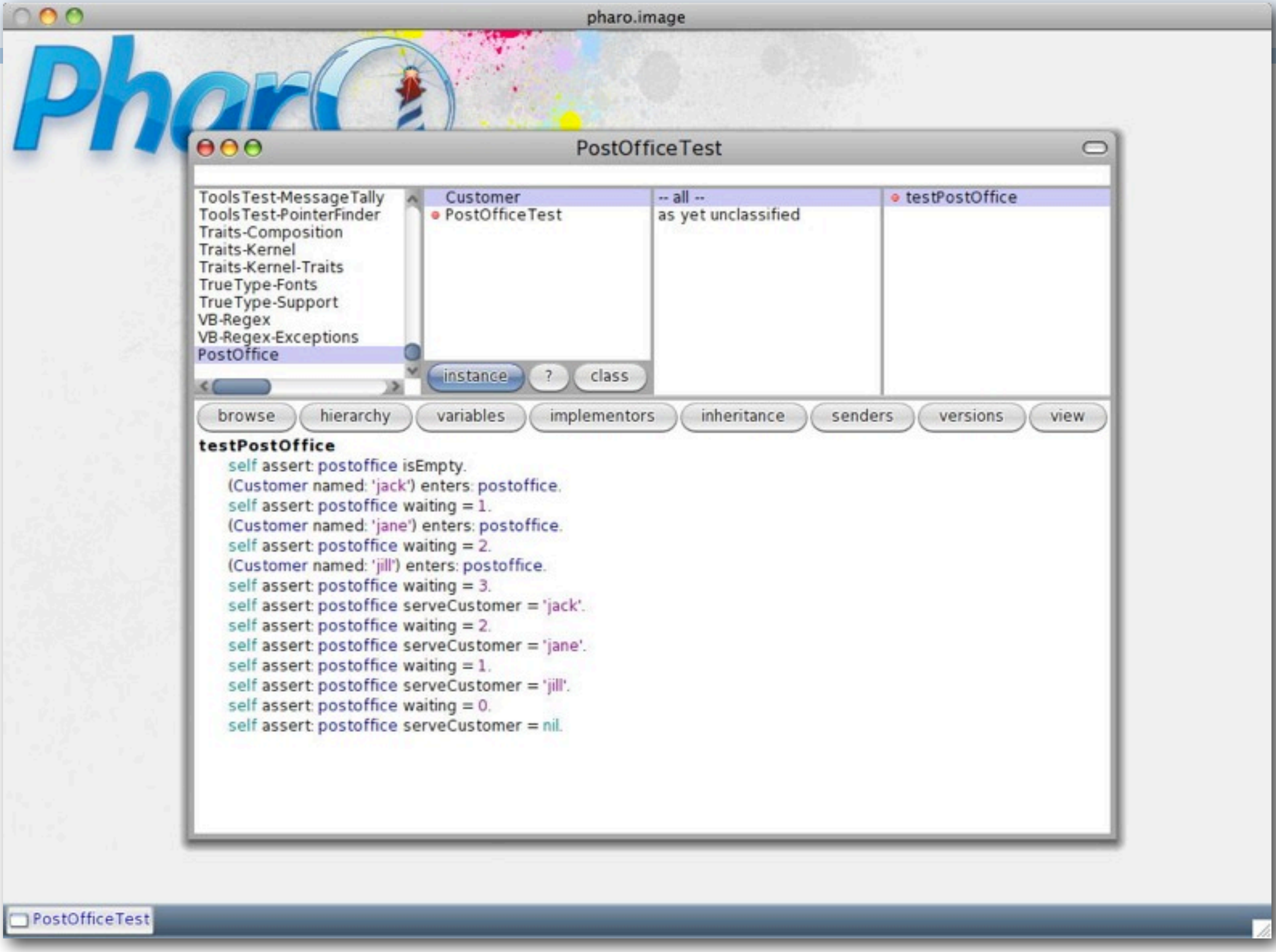
```
a Point
a Fraction
```

Creating classes

> Send a message to a class (!)

```
Number subclass: #Complex
   instanceVariableNames: 'real imaginary'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'ComplexNumbers'
```

Demo: Defining classes and methods

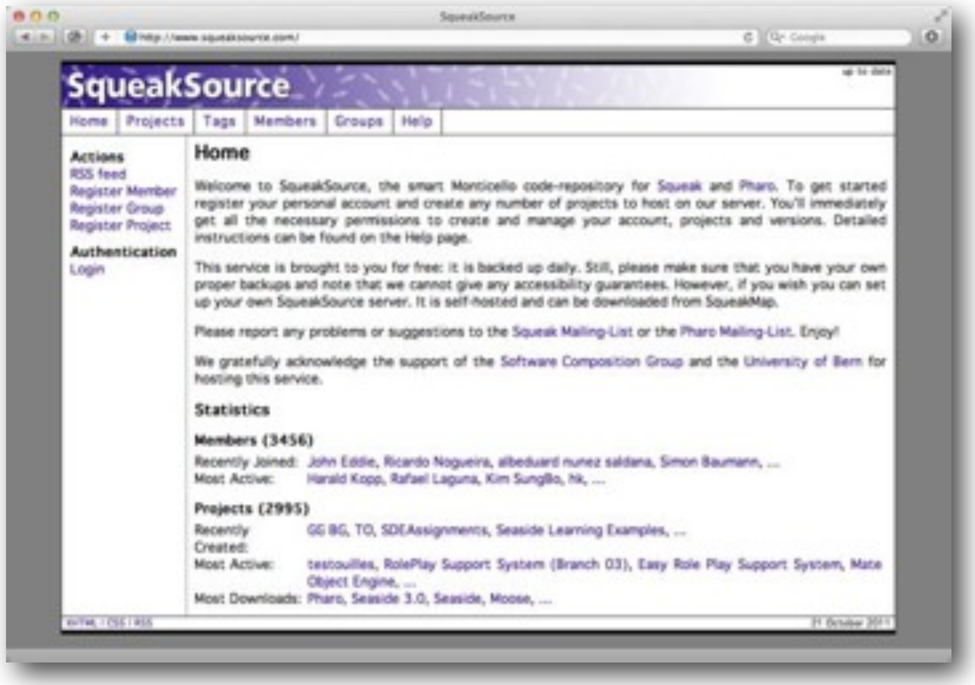


```
testPostOffice
    self assert: postoffice isEmpty.
    (Customer named: 'jack') enters: postoffice.
    self assert: postoffice waiting = 1.
    (Customer named: 'jane') enters: postoffice.
    self assert: postoffice waiting = 2.
    (Customer named: 'jill') enters: postoffice.
    self assert: postoffice waiting = 3.
    self assert: postoffice serveCustomer = 'jack'.
    self assert: postoffice waiting = 2.
    self assert: postoffice serveCustomer = 'jane'.
    self assert: postoffice waiting = 1.
    self assert: postoffice serveCustomer = 'jill'.
    self assert: postoffice waiting = 0.
    self assert: postoffice serveCustomer = nil.
```

Roadmap

> The origins of Smalltalk
> What is Smalltalk?
> Syntax in a nutshell
> **Seaside — web development with Smalltalk**

Seaside — a Smalltalk web development platform


SqueakSource

Home | Projects | Tags | Members | Groups | Help

Actions
RSS feed
Register Member
Register Group
Register Project

Authentication
Login

Home

Welcome to SqueakSource, the smart Monticello code-repository for Squeak and Pharo. To get started register your personal account and create any number of projects to host on our server. You'll immediately get all the necessary permissions to create and manage your account, projects and versions. Detailed instructions can be found on the Help page.

This service is brought to you for free: it is backed up daily. Still, please make sure that you have your own proper backups and note that we cannot give any accessibility guarantees. However, if you wish you can set up your own SqueakSource server. It is self-hosted and can be downloaded from SqueakMap.

Please report any problems or suggestions to the Squeak Mailing-List or the Pharo Mailing-List. Enjoy!

We gratefully acknowledge the support of the Software Composition Group and the University of Bern for hosting this service.

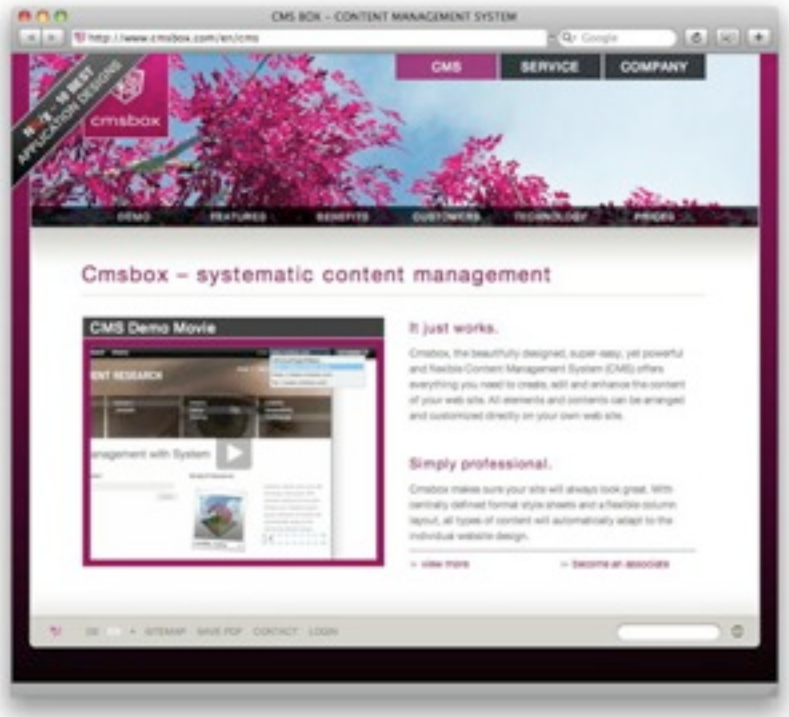
Statistics

Members (3456)

Recently Joined:	John Eddie, Ricardo Nogueira, albeduard nunez saldana, Simon Baumann, ...
Most Active:	Harald Kopp, Rafael Laguna, Kim SungBo, hk, ...

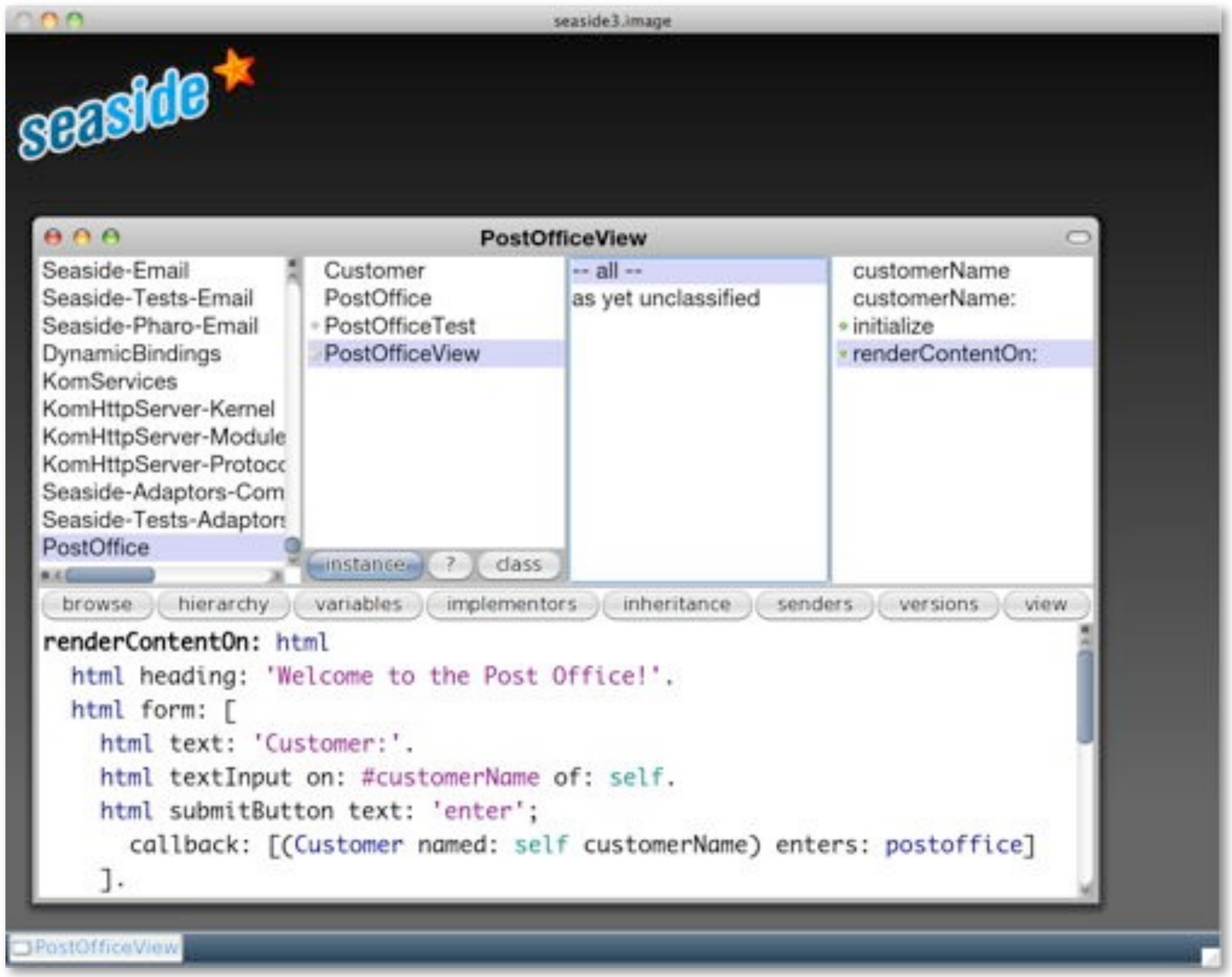
Projects (2995)

Recently Created:	GG BG, TO, SDEAssignments, Seaside Learning Examples, ...
Most Active:	testouilles, RolePlay Support System (Branch 03), Easy Role Play Support System, Mate Object Engine, ...
Most Downloads:	Pharo, Seaside 3.0, Seaside, Moose, ...

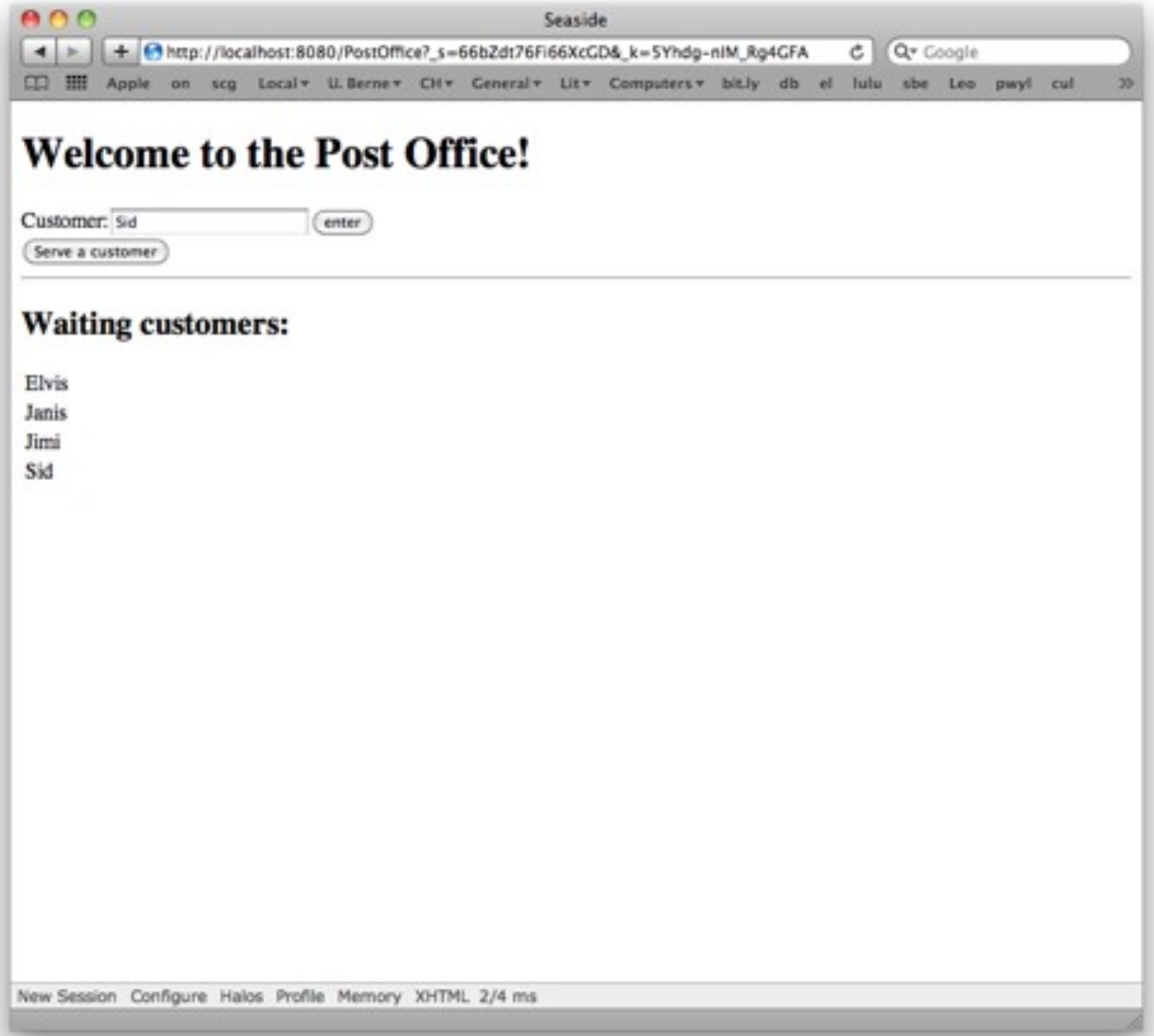

CMS BOX – CONTENT MANAGEMENT SYSTEM
CMS
SERVICE
COMPANY
cmsbox
Cmsbox – systematic content management
CMS Demo Movie
It just works.
Simply professional.

The 8 Minute Demo
dabble
Tired of reading? Try Dabble DB yourself for free. Create a database in minutes. No obligations, no hassles. (Honest.)

Demo: PostOffice in Seaside



```
renderContentOn: html
  html heading: 'Welcome to the Post Office!'.
  html form: [
    html text: 'Customer:'.
    html textInput on: #customerName of: self.
    html submitButton text: 'enter';
      callback: [(Customer named: self customerName) enters: postoffice]
    ].
```


Seaside
http://localhost:8080/PostOffice?_s=66bZdt76Fi66XcGD&_k=5Yhdg-nlM_Rg4GFA
Google
Apple on scg Local U. Berne CH General Lit Computers bit.ly db el lulu sbe Leo pwyl cul
New Session Configure Halos Profile Memory XHTML 2/4 ms

Welcome to the Post Office!

Customer: Sid [enter]

[Serve a customer]

Waiting customers:

Elvis
Janis
Jimi
Sid

What you should know!

- ✎ *What are the key differences between Smalltalk, C++ and Java?*
- ✎ *What is at the root of the Smalltalk class hierarchy?*
- ✎ *What kinds of messages can one send to objects?*
- ✎ *What is a cascade?*
- ✎ *Why does* `1+2/3 = 1` *in Smalltalk?*
- ✎ *How are control structures realized?*
- ✎ *How is a new class created?*
- ✎ *What are categories for?*
- ✎ *What are Factory methods? When are they useful?*

Can you answer these questions?

- ✎ *Which is faster, a program written in Smalltalk, C++ or Java?*
- ✎ *Which is faster to develop & debug, a program written in Smalltalk, C++ or Java?*
- ✎ *How are Booleans implemented?*
- ✎ *Is a comment an Object? How would you check this?*
- ✎ *What is the equivalent of a static method in Smalltalk?*
- ✎ *How do you make methods private in Smalltalk?*
- ✎ *What is the difference between = and ==?*
- ✎ *If classes are objects too, what classes are they instances of?*