http://stephane.ducasse.free.fr

Live tools are developers' best friends

http://www.pharo.org

Novi Sad - 2024

SCHMIDT
Ingenieurbüro für Bauwesen

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Lifeware

ENSTA Bretagne

THALES

ZWEIDENKER

Yesplan
Let's make it happen

Lam RESEARCH

esug
EUROPEAN SMALLTALK USER GROUP

projector software

GemTalk SYSTEMS

SpesenFuchs

InfOil

feenk

netstyle.ch

oco
osoco software company

inspired!

BetaNine
software engineering

telna

HRworks

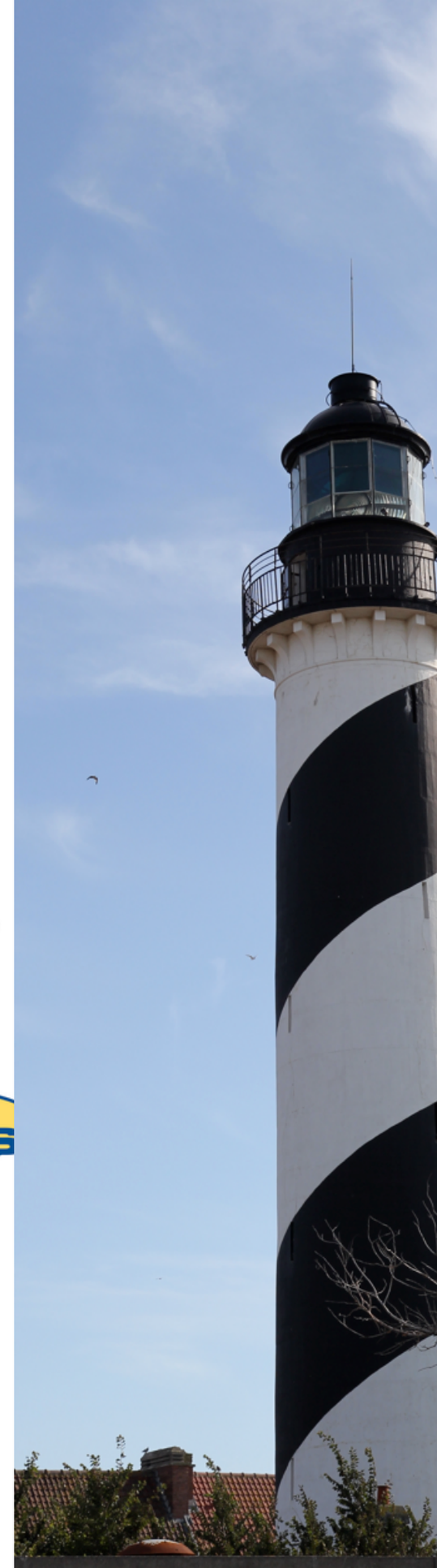
FINWORKS
INFORMATION TECHNOLOGY SOLUTIONS

Systems that make sense
Sensus

OBJECT PROFILE

OBJECT GUILD

TA MÈRE SCRL
BADASS MOBILE DEVELOPMENT



A journey in a live environment and its companion tools

- Pharo in 5 min
- Some advanced features
- Cool tools

Pharo!

- System: Pure object language + full IDE
- Powerful, elegant and fun to program
- Living system under your fingers
- Works on 11 oses, 32/64, X86, ARM, M1
- 100% MIT

Pharo in Numbers

17 releases since 2008
Language Core + IDE + Tools + Frameworks
~ 800 packages (tests included)

~ 27000 tests
# 10k classes
# 137k methods
61 Mb (64 bits)

http://github.com/pharo-project/Pharo (~300 forks, 15/123 international contributors)

Ecosystem
- polymath
- pharo-graphics
- pharo-gis
- pharo-container
- pharo-ai

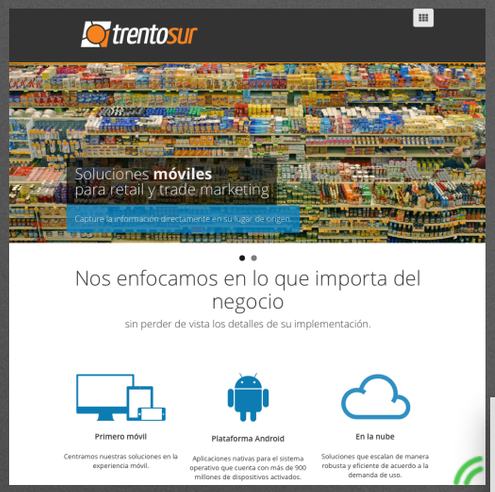
Pharo is our vehicle
We improve it everyday

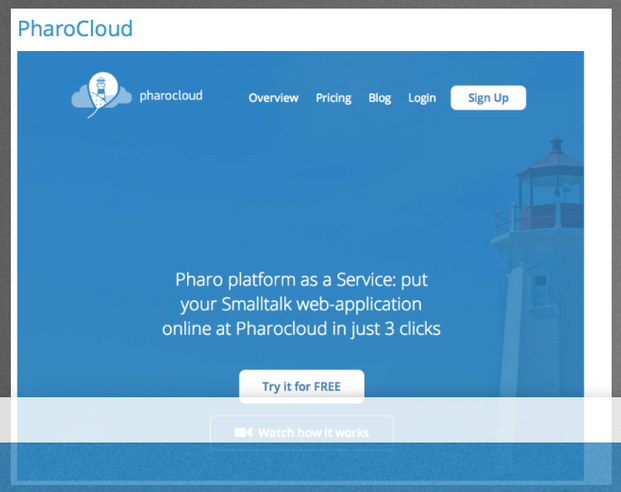
A powerful engine to invent (y)our future

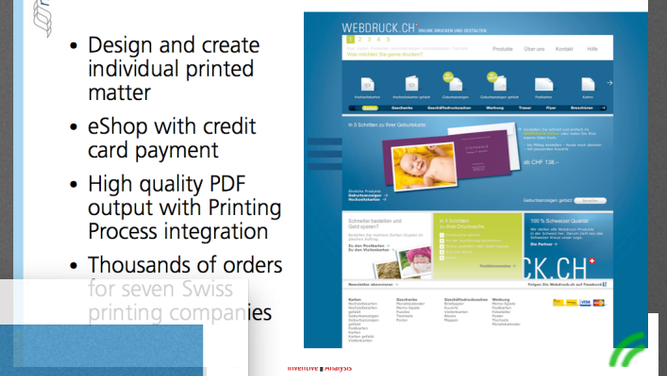
An ecosystem where innovation/business bloom

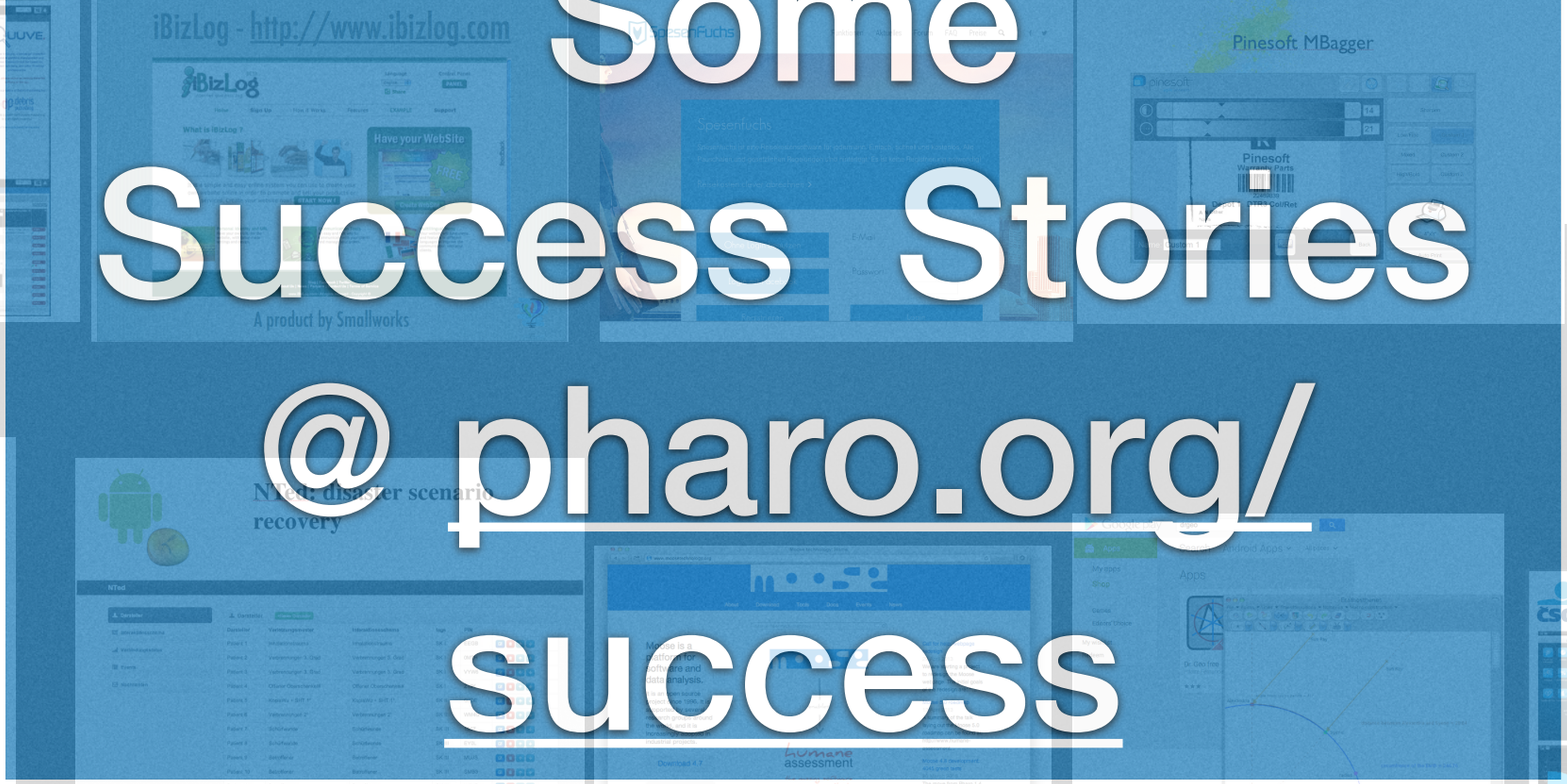
Some Success Stories

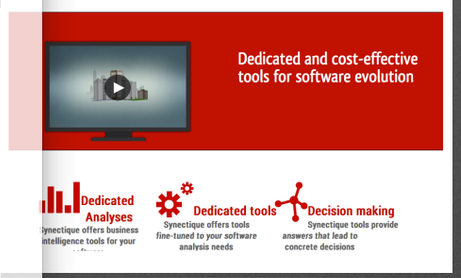
@ pharo.org/success

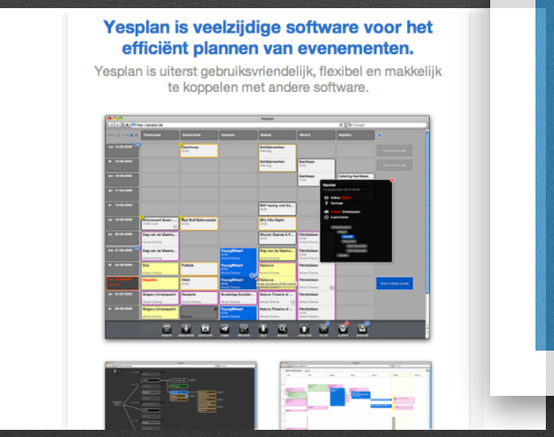


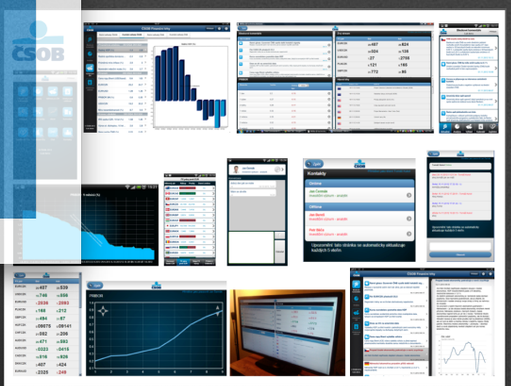


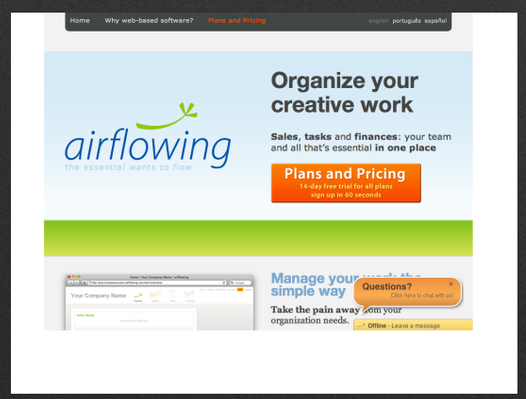


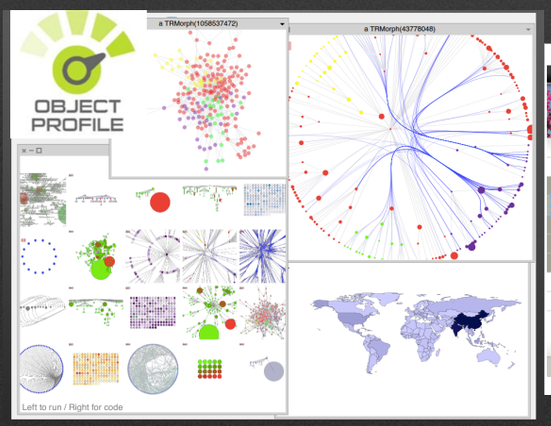


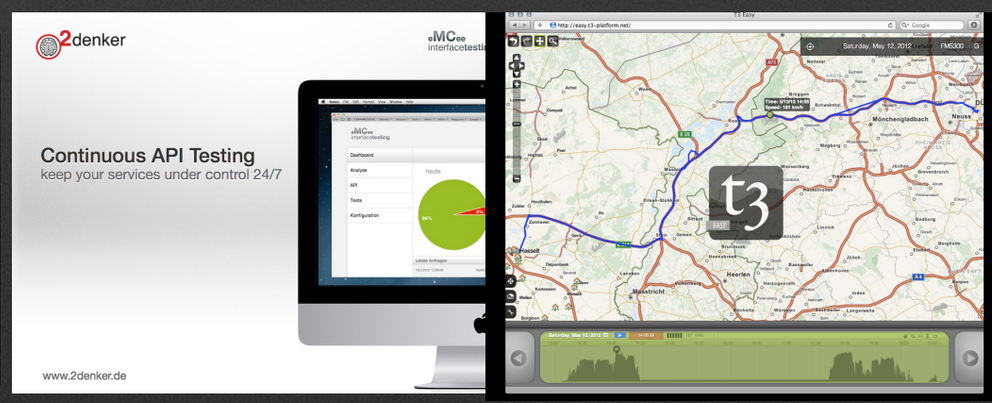












Elegant!

- Full syntax on a postcard
- Simple and powerful objet model

Pharo

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |
	true & false not & (nil isNil)
	 ifFalse: [ self perform: #add: with: x ].
	y := thisContext stack size + super size.
	byteArray := #[2 2r100 8r20 16rFF].
	{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
	 do: [ :each |
		| var |
		var := Transcript
		 show: each class name;
		 show: each printString ].
	^ x < y
```

https://www.pharo.org

PLACE STAMP HERE

.....

.....

.....

.....

Yes the full Syntax!

Yes there is nothing else…

Pharo

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |
	true & false not & (nil isNil)
	 ifFalse: [ self perform: #add: with: x ].
	y := thisContext stack size + super size.
	byteArray := #[2 2r100 8r20 16rFF].
	{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
	 do: [ :each |
		| var |
		var := Transcript
		 show: each class name;
		 show: each printString ].
	^ x < y
```

https://www.pharo.org

PLACE STAMP HERE

.....

.....

.....

.....

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |
	true & false not & (nil isNil)
		ifFalse: [ self perform: #add: with: x ].

	y := thisContext stack size + super size.

	byteArray := #[2 2r100 8r20 16rFF].

	{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
		do: [ :each |

			| var |
			var := Transcript
				show: each class name;
				show: each printString ].

	^ x < y
```

method

annotation

```
exampleWithNumber: x
```

parameter

```
<syntaxOn: #postcard>
"A ""complete"" Pharo syntax"
| y |

true & false not & (nil isNil)
 ifFalse: [ self perform: #add: with: x ].

y := thisContext stack size + super size.

byteArray := #[2 2r100 8r20 16rFF].
```

comment

```
{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
 do: [ :each |

  | var |
  var := Transcript
   show: each class name;
   show: each printString ].

^ x < y
```

return

```
exampleWithNumber: x
```

parameter

```
<syntaxOn: #postcard>
"A ""complete"" Pharo syntax"
| y |
```

local var

```
true & false not & (nil isNil)
 ifFalse: [ self perform: #add: with: x ].
```

affectation

```
y := thisContext stack size + super size.
```

instance var

```
byteArray := #[2 2r100 8r20 16rFF].
```

```
{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
 do: [ :each |
```

local var

```
  | var |
  var := Transcript
   show: each class name;
   show: each printString ].
```

```
^ x < y
```

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |

	true & false not & (nil isNil)
	 ifFalse: [ self perform: #add: with: x ].

	y := thisContext stack size + super size.

	byteArray := #[2 2r100 8r20 16rFF].

	{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
	 do: [ :each |

	  | var |
	  var := Transcript
	   show: each class name;
	   show: each printString ].

	^ x < y
```

closure

arg closure

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |
	true & false not & (nil isNil)
	 ifFalse: [ self perform: #add: with: x ].

	y := thisContext stack size + super size.

	byteArray := #[2 2r100 8r20 16rFF].
```

binary msg

keyword msg

unary msg

```
{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
 do: [ :each |

   | var |
   var := Transcript
    show: each class name;
    show: each printString ].

^ x < y
```

cascade

separator

binary msg

```
exampleWithNumber: x

	<syntaxOn: #postcard>
	"A ""complete"" Pharo syntax"
	| y |

	true & false not & (nil isNil)
	 ifFalse: [ self perform: #add: with: x ].

	y := thisContext stack size + super size.

	byteArray := #[2 2r100 8r20 16rFF].

	{ -42 . #($a #a #'I''m' 'a' 1.0 1.23e2 3.14s2 1) }
	 do: [ :each |

	  | var |
	  var := Transcript
	   show: each class name;
	   show: each printString ].

	^ x < y
```

array character

symbol

string

number

A Pure World of Objects

Only

objects + messages +

closures

mouse, booleans, arrays, numbers, strings, windows, scrollbars, canvas, files, trees, compilers, sound, url, socket, fonts, text, collections, stack, shortcut, streams, …

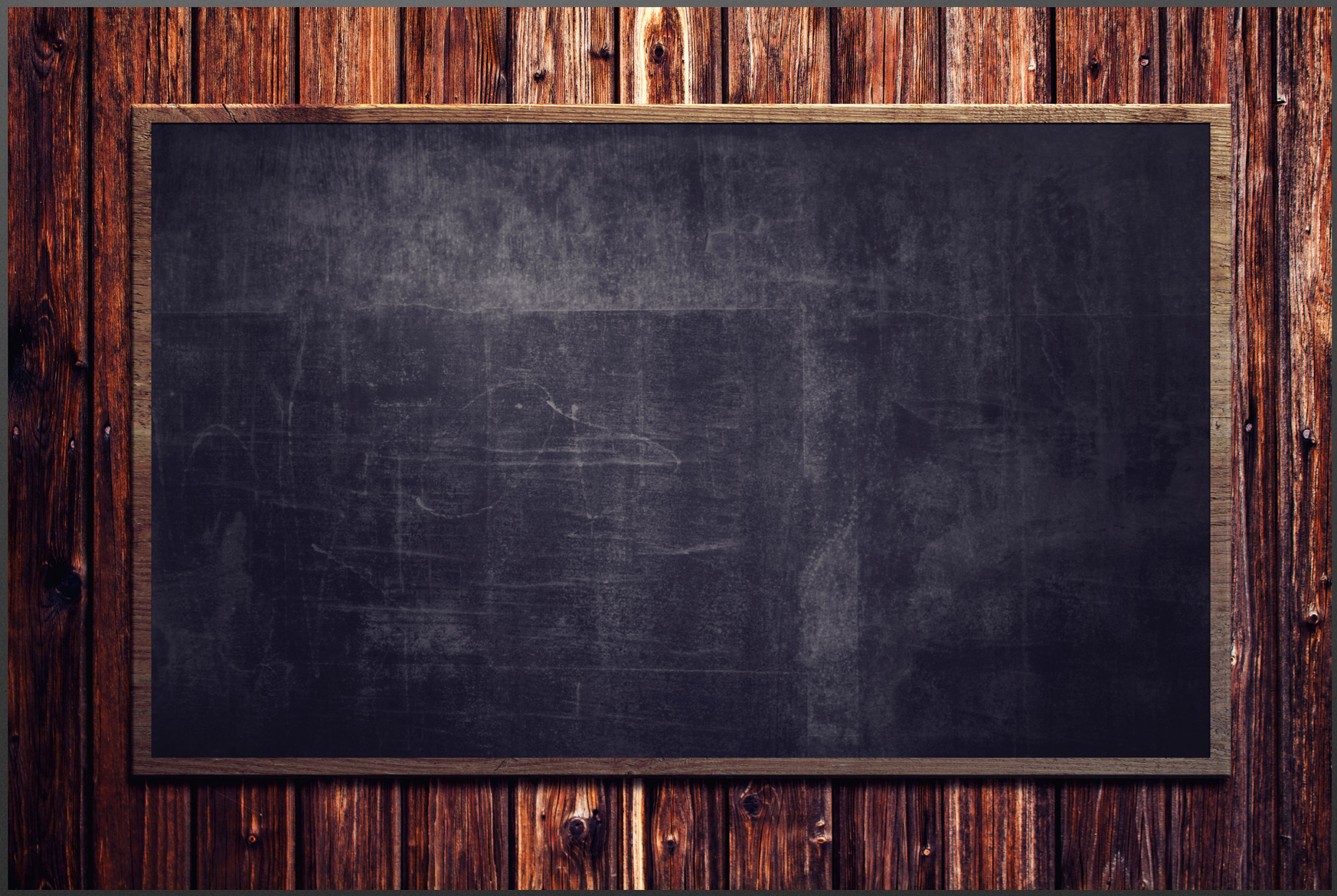
A Fully Uniform Model

- Dynamically typed
- Everything is an object instance of a class
- All methods are public virtual
- All attributes are protected
- Single inheritance with traits

Pharo is highly immersive

if a compiler would be an aquarium

compiling a program would like writing on a black board



Immersing...

Pharo is not a blackbox

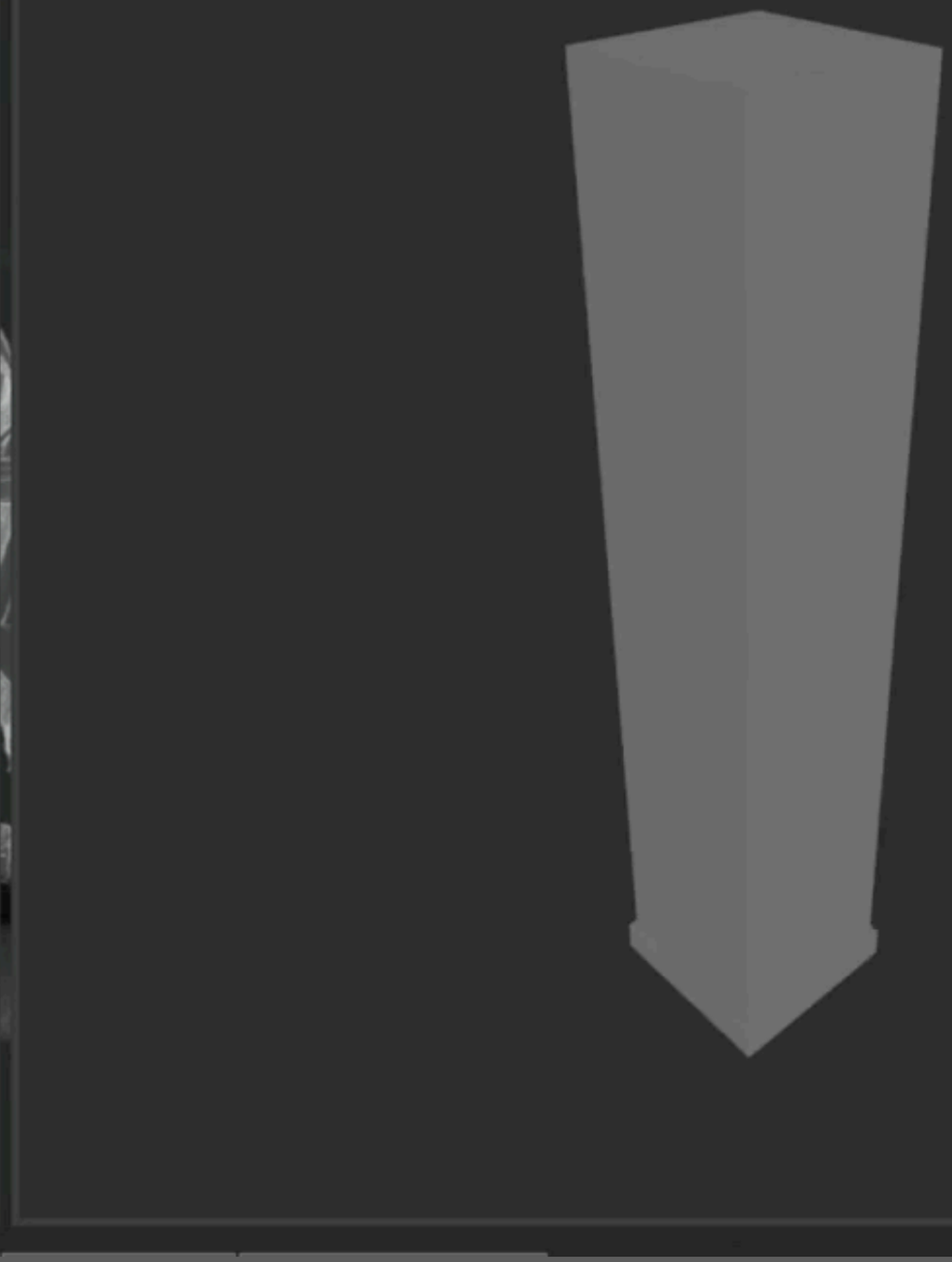
Everything is fully inspectable and reflective

You are immersed and interacting with objects

Workspace

```
| elements lay |


elements := (1 to: 5) collect: [ :ob |
  (R3CubeShape new) elementOn: ob ].


lay := R3WallLayout new.
lay on: elements.

UberPresenter present: elements
```

Uber Presenter



We can do the same with web app, sockets, networks, sensors, living programing….

Hackers scripting live the coffee machine

Selected Infrastructure

Fully Written in Itself

Selected features

- First class instance variables (daemons, relationships…)
- Fast resumable exceptions
- Runtime classes and objects migration
- Customizable compiler
- Serializable and shareable execution stack
- Optional system virtualization
- Fully bootstrapped kernel(s) (down to 200kb)

Advanced reflective layer

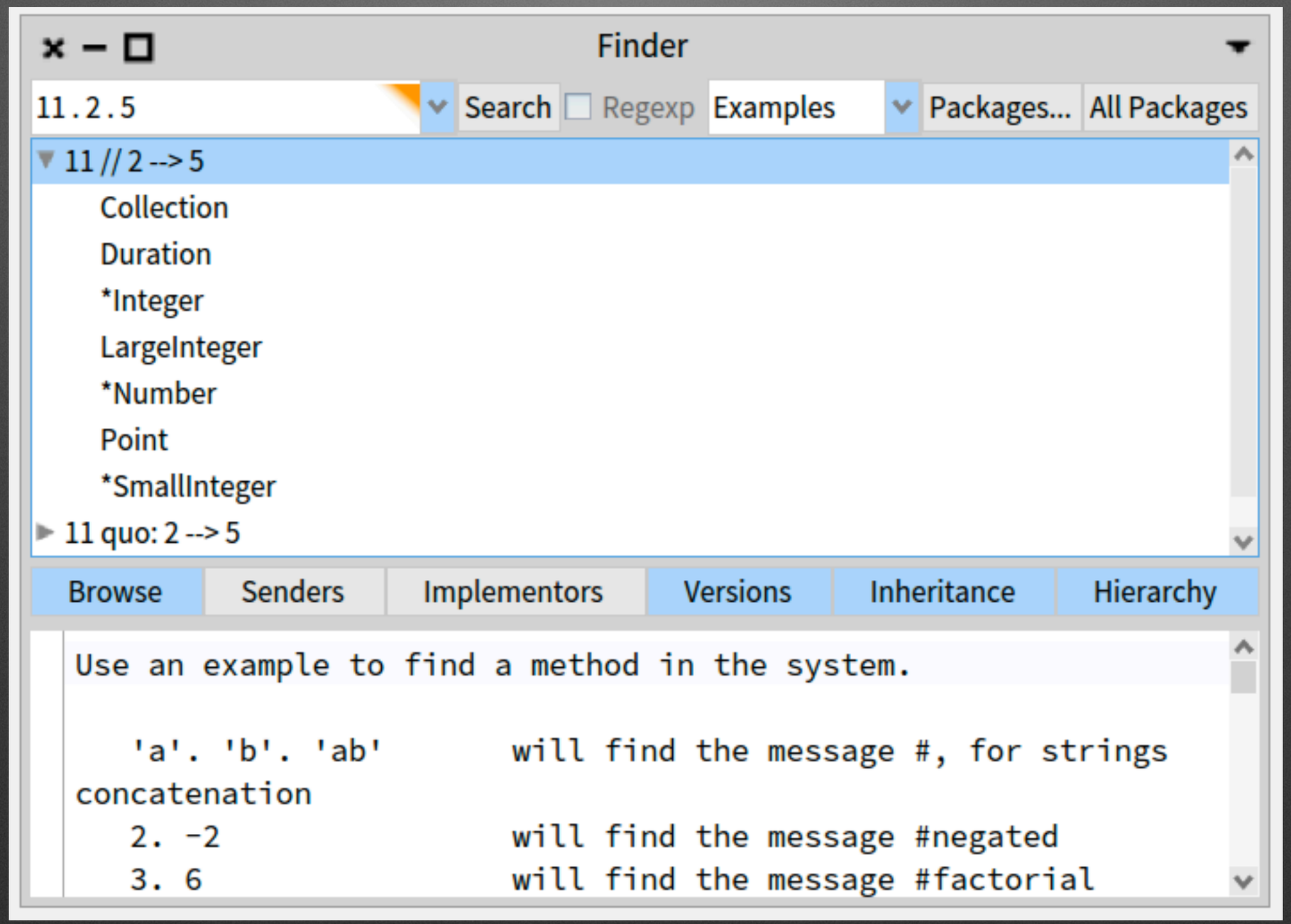
- Versatile AST annotations and transformations @ runtime
- Full stack reification (continuations, exceptions…)
- Instance enumeration
- Causally connected “Software as Objects”
- Atomic bulk object swapping
- … more but no time for that

Tools are our friends
But they shape our mind

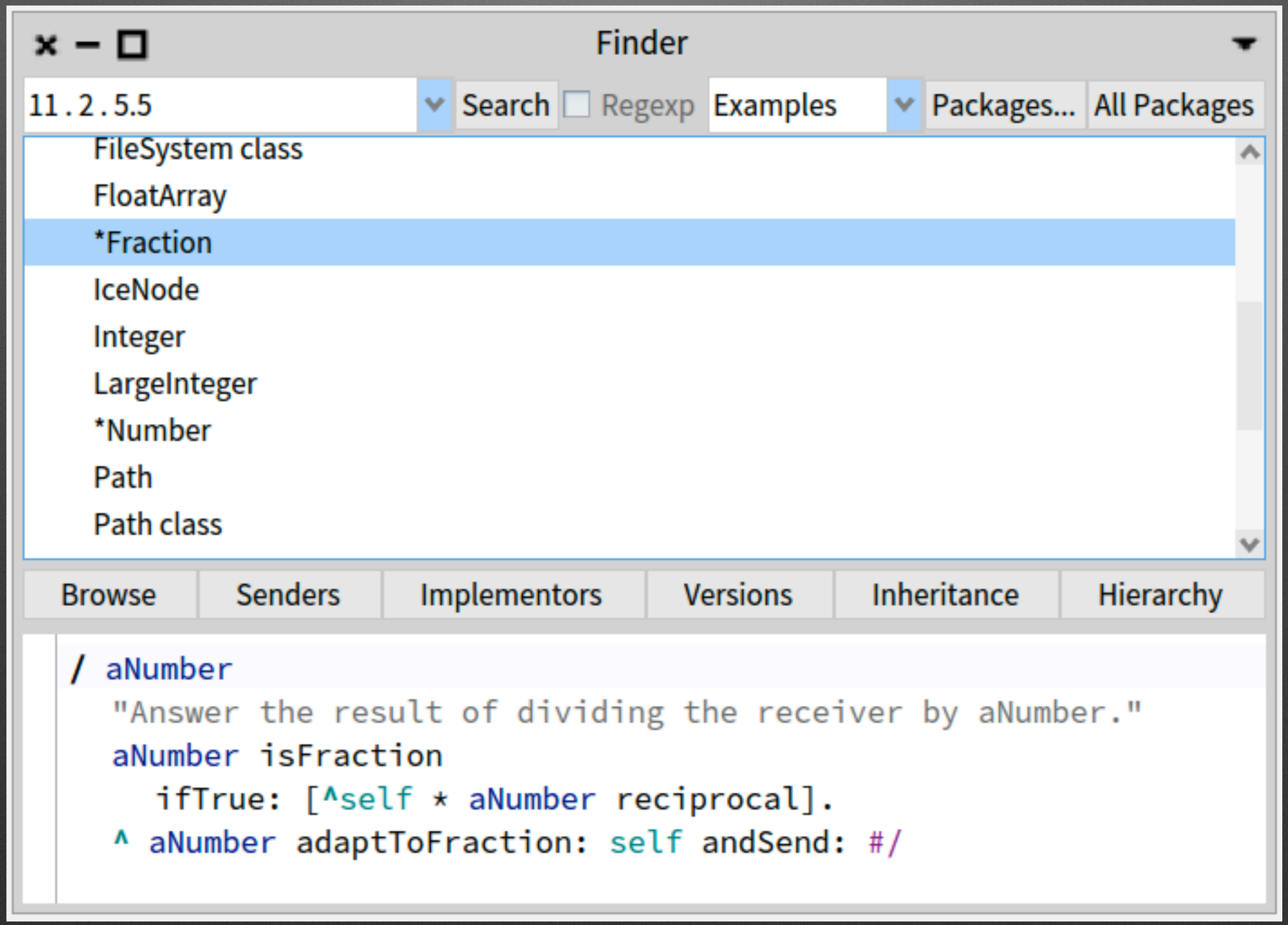
How to find information?

- Libraries are large
- You know what you want
- You do not know how to express it

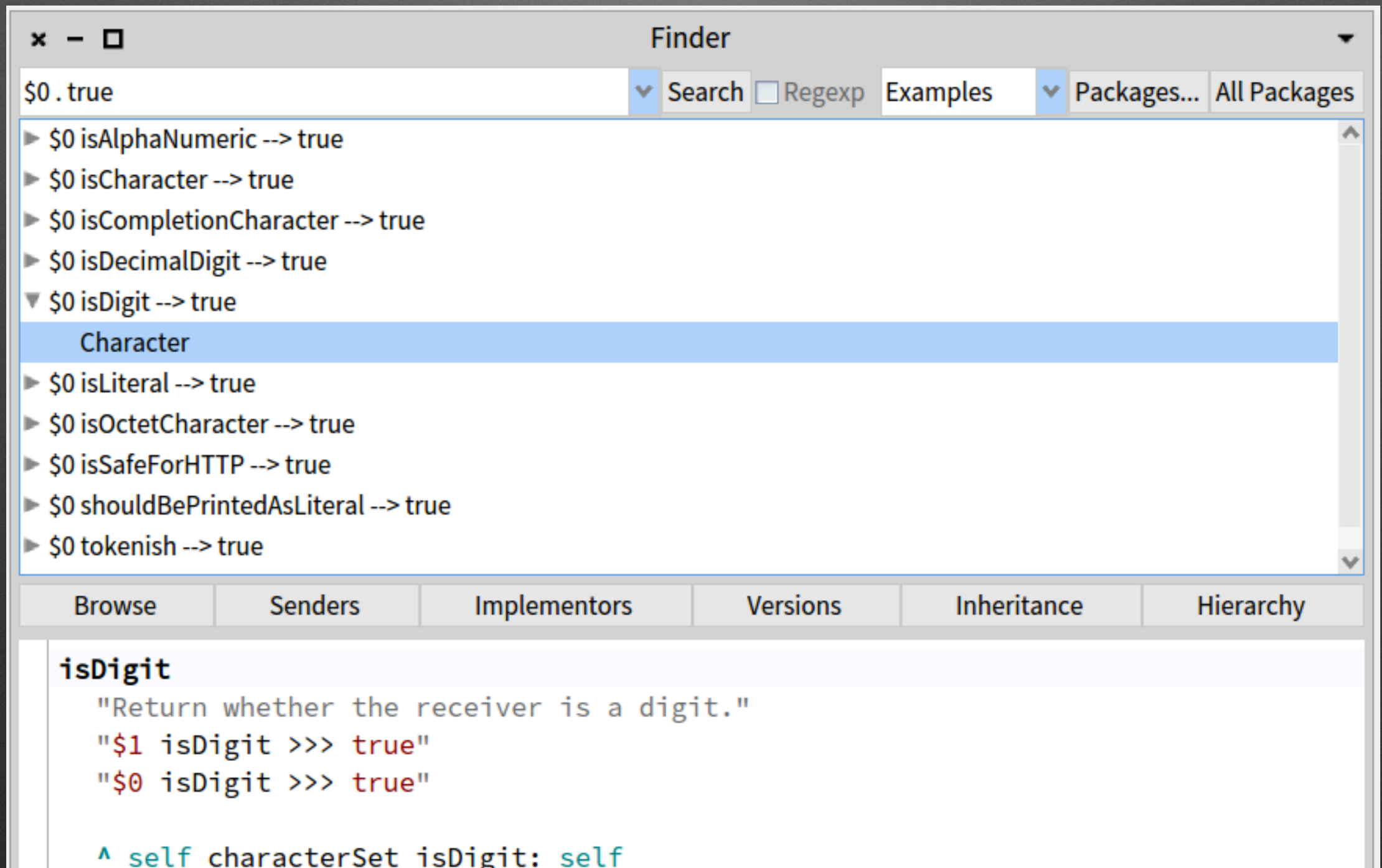
11 ??? 2 should give 5



11 ??? 2 should give 5.5



What are the messages send to $0 that return true



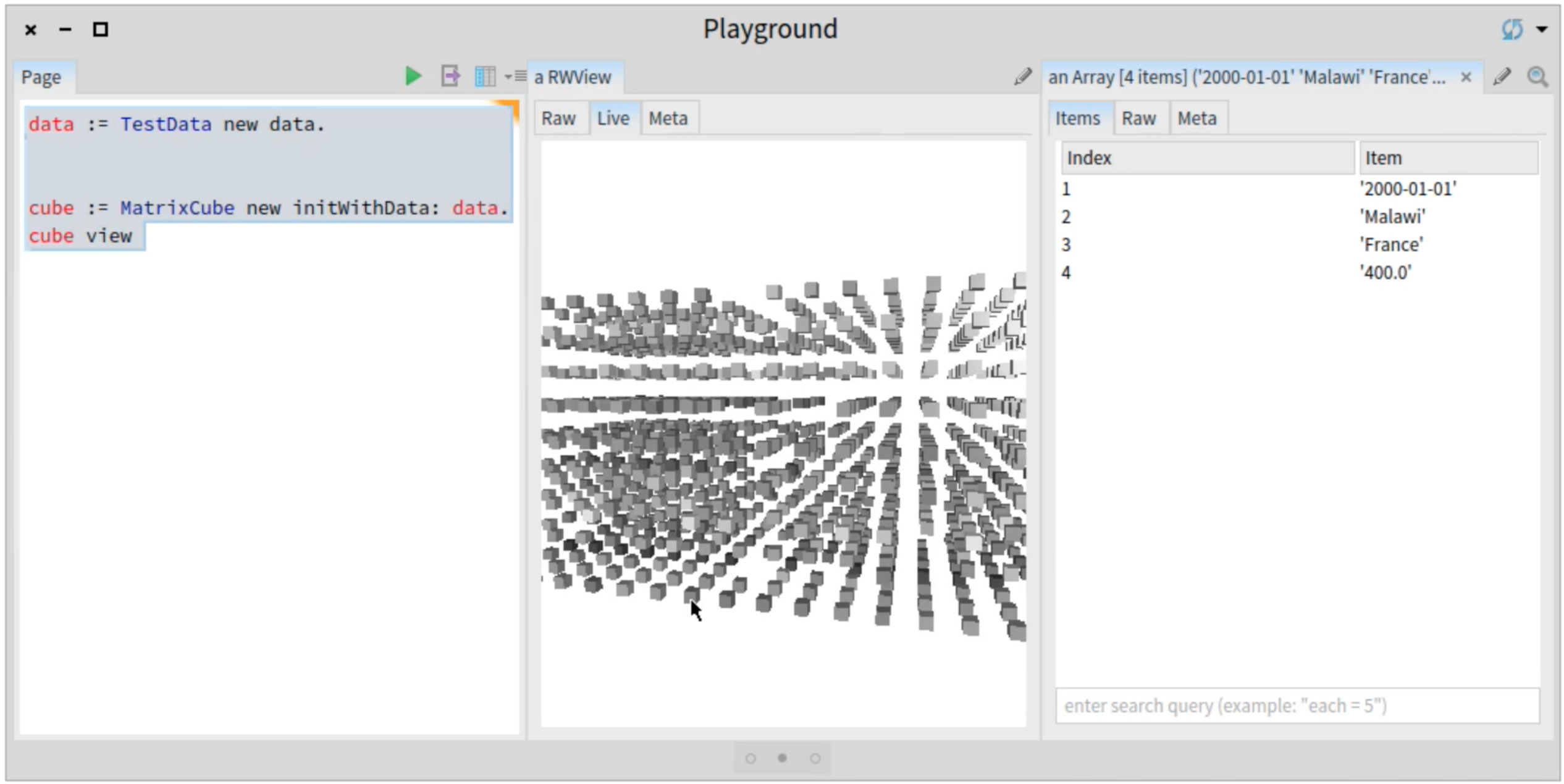
Tools

- Shape our mind…
- Pharo has moldable tools: you CAN adapt them to you and your process and not the inverse
- Build fast your own tools

Pharo has amazing moldable tools

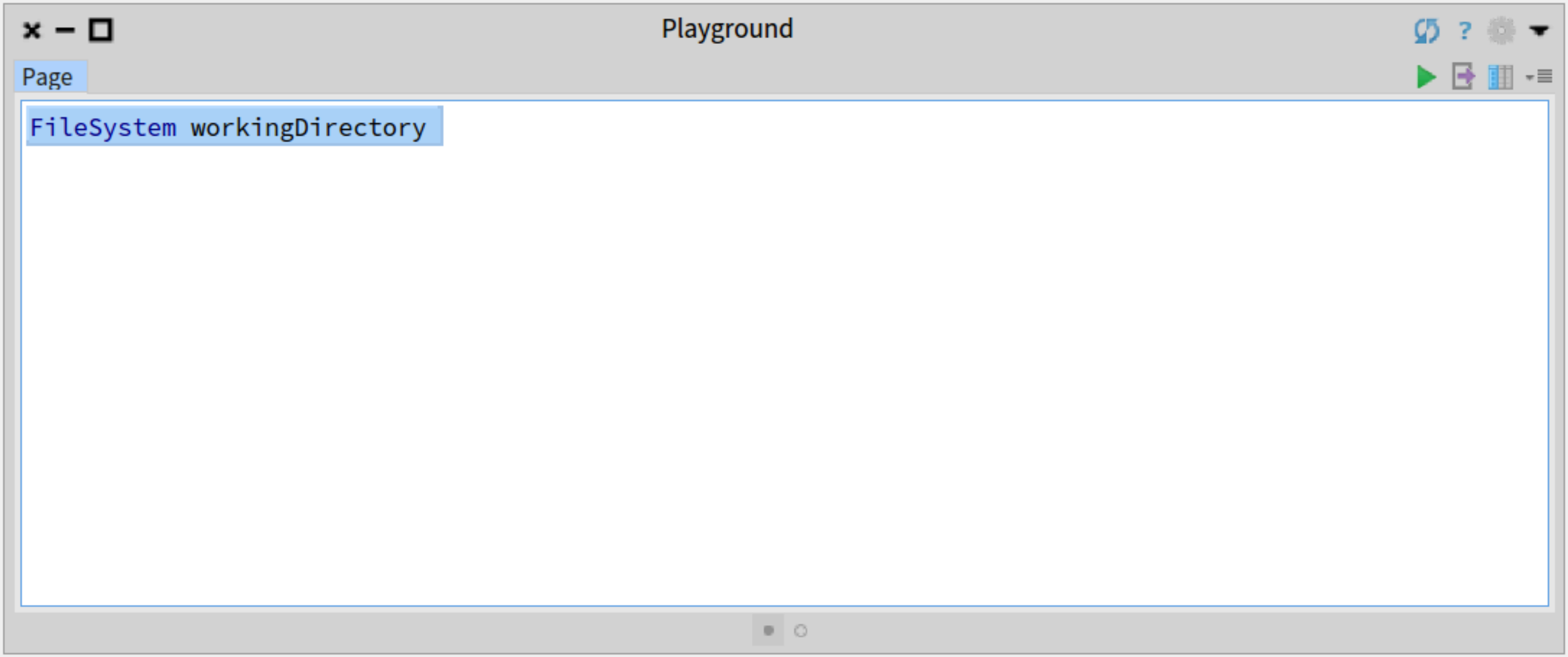
Customizable object interaction/presentations

Inspecting live a 3D object

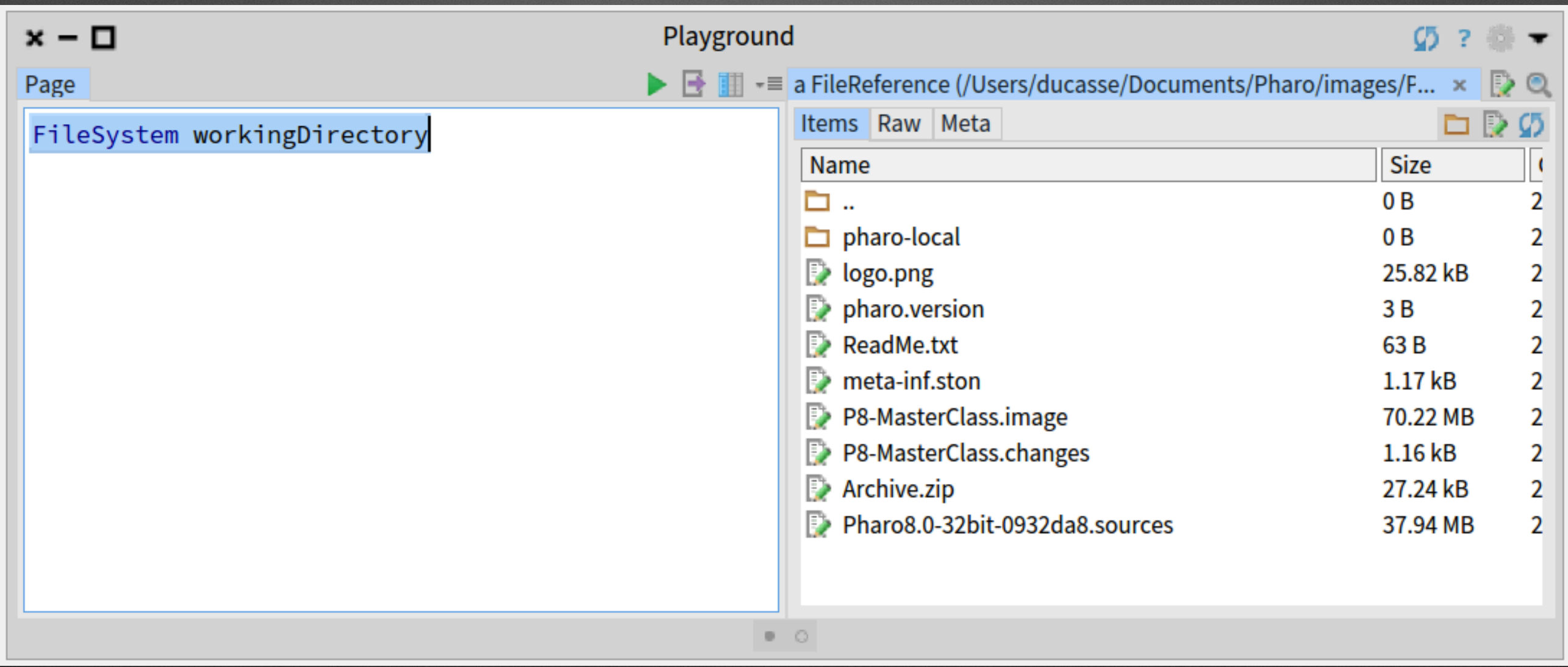


The views of a file reference

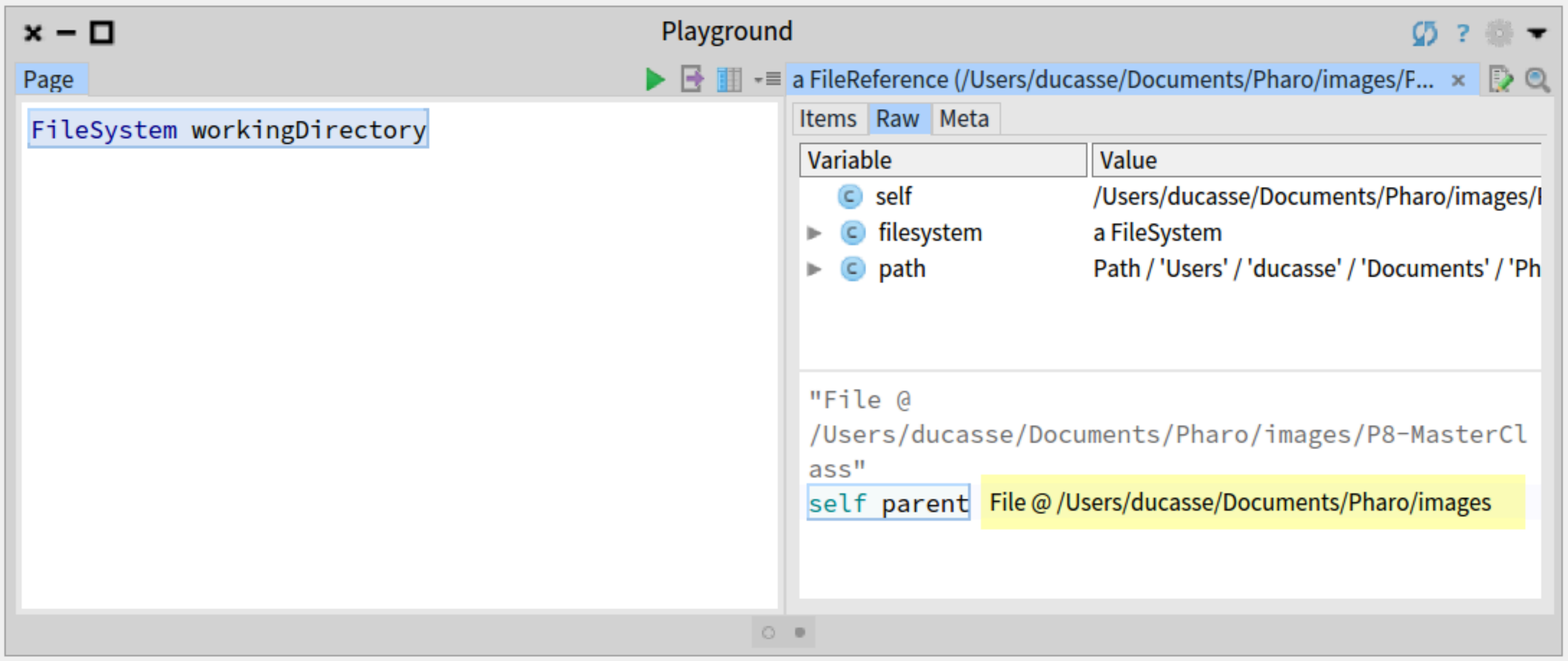
Looking at a file reference



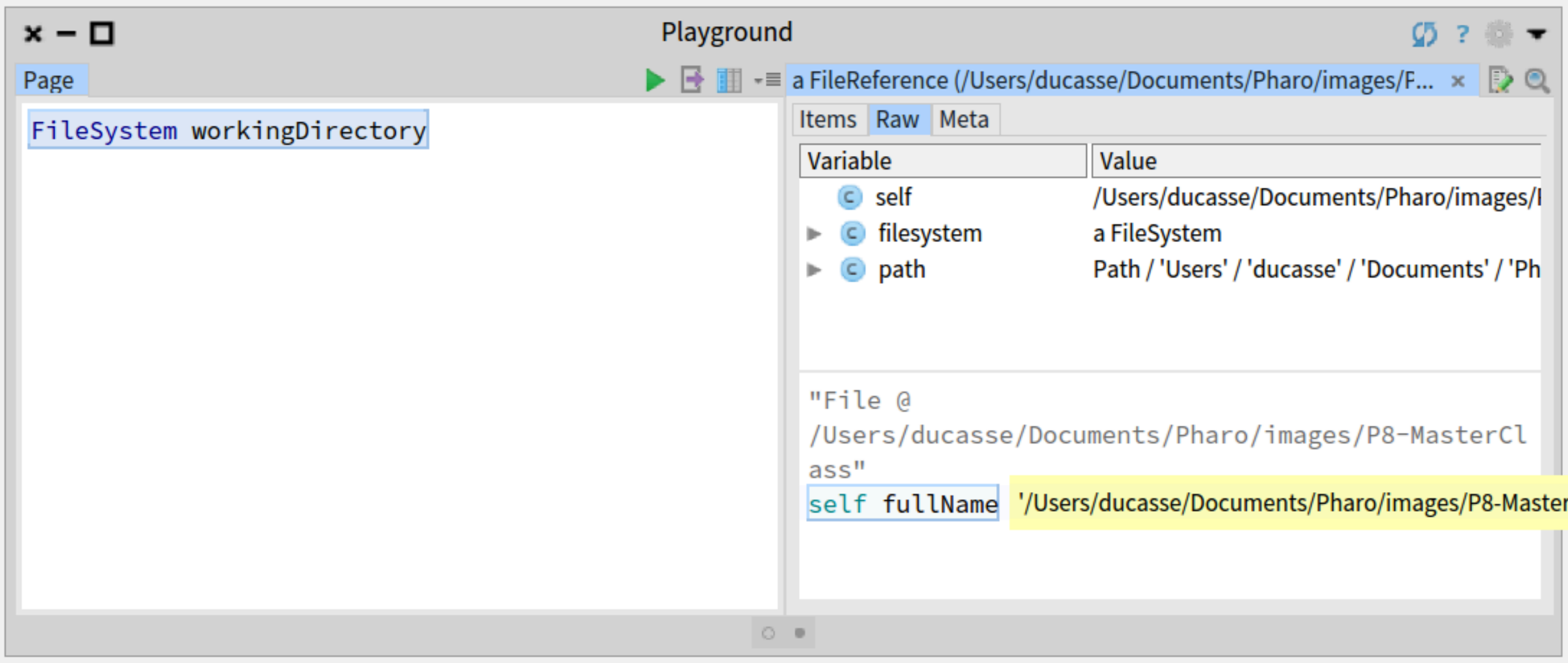
Oh! a file browser in my inspector!



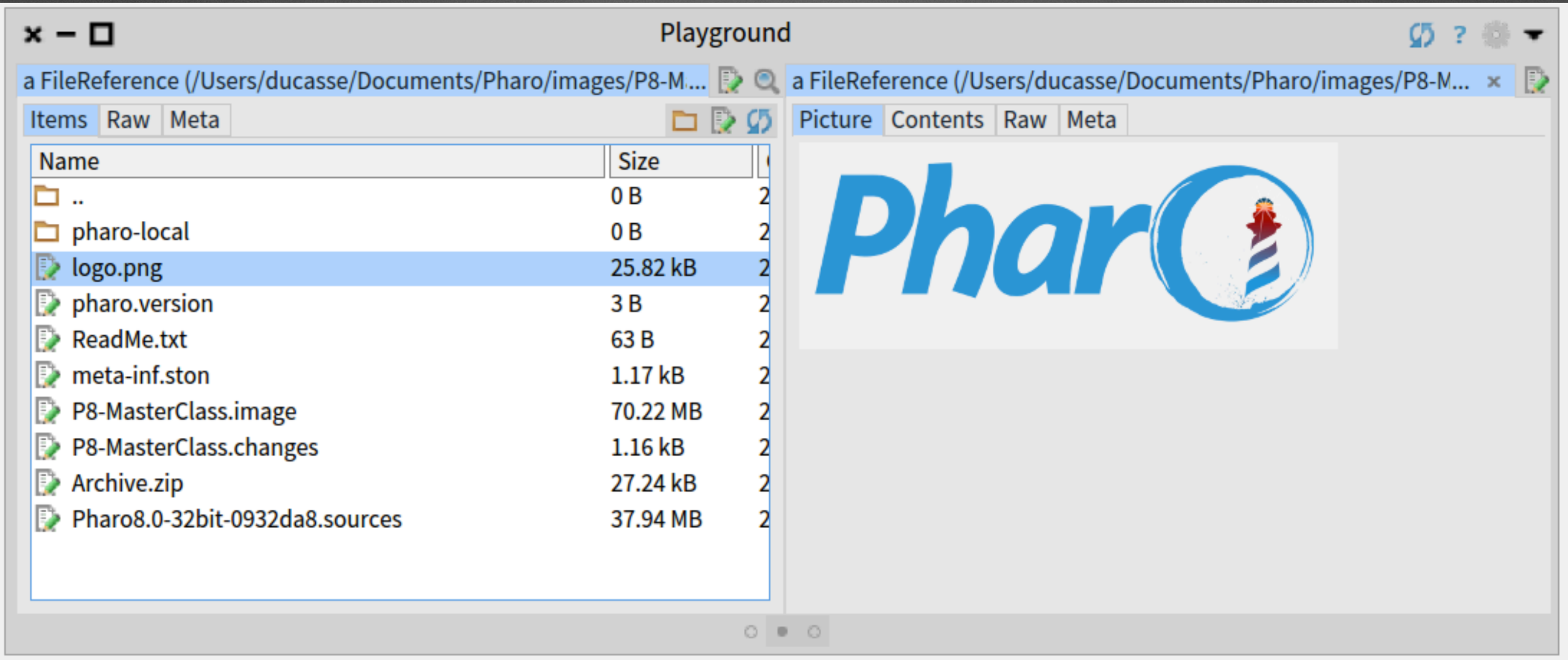
But I have a file reference: a dull object



Quite boring object



We can see the png ;)



Looking inside that PNG file

Playground

a FileReference (/Users/ducasse/Documents/Pharo/images/P8-M...

Items | Raw | Meta

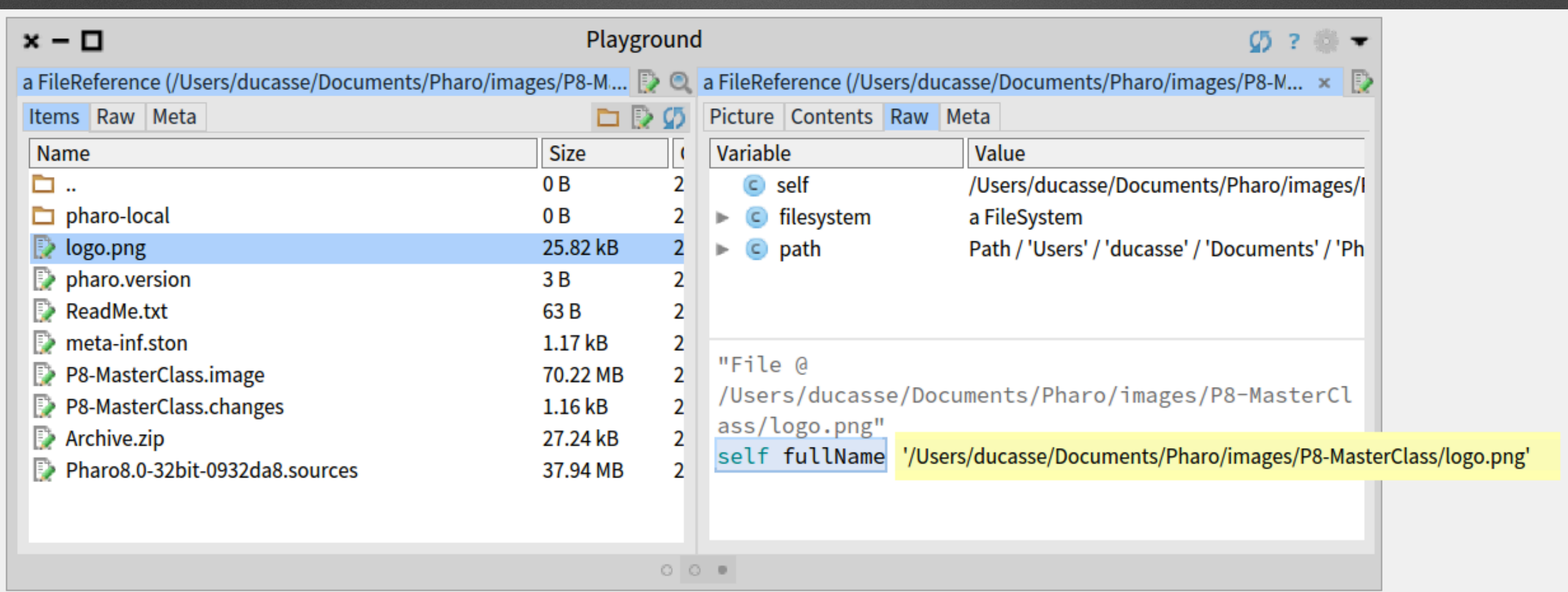
Name	Size	
..	0 B	2
pharo-local	0 B	2
logo.png	25.82 kB	2
pharo.version	3 B	2
ReadMe.txt	63 B	2
meta-inf.ston	1.17 kB	2
P8-MasterClass.image	70.22 MB	2
P8-MasterClass.changes	1.16 kB	2
Archive.zip	27.24 kB	2
Pharo8.0-32bit-0932da8.sources	37.94 MB	2

a FileReference (/Users/ducasse/Documents/Pharo/images/P8-M...

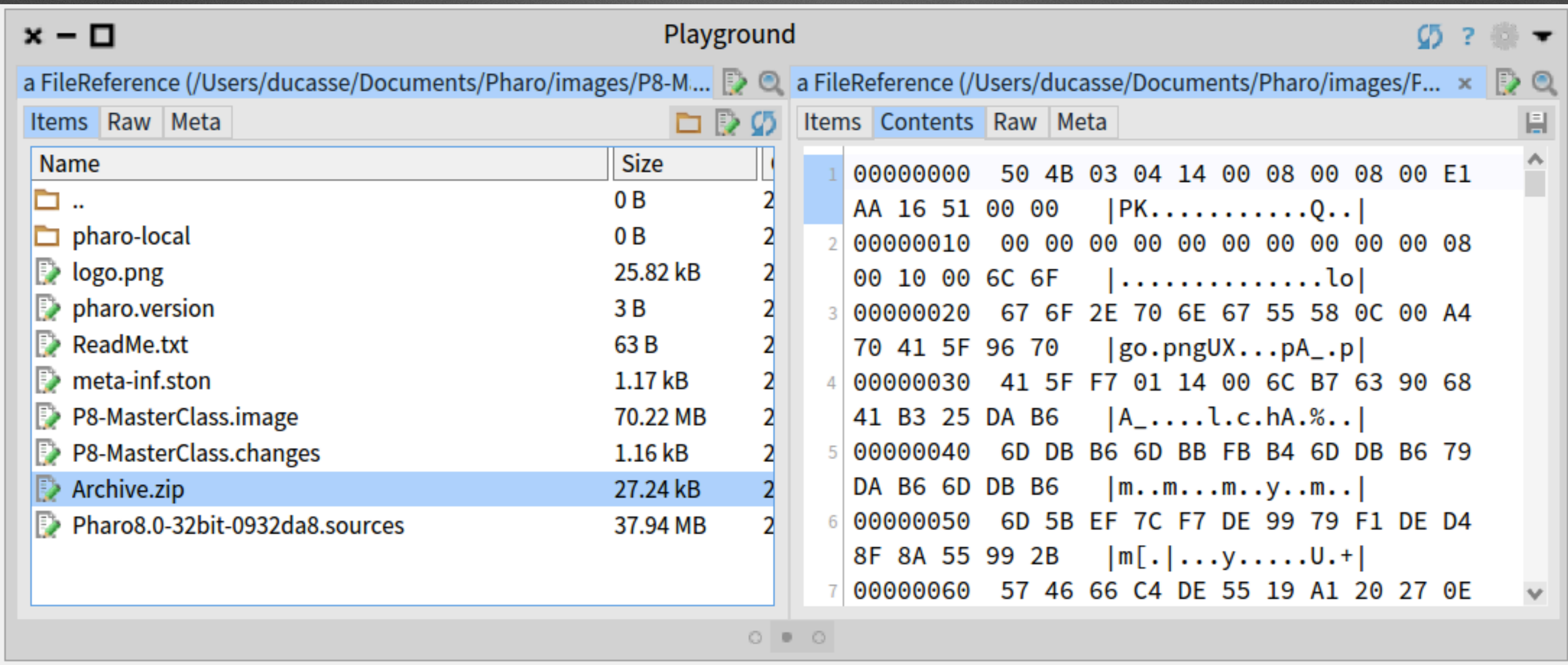
Picture | Contents | Raw | Meta

```
1 00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00
  0D 49 48 44 52   |.PNG........IHDR|
2 00000010  00 00 01 77 00 00 00 90 08 06 00
  00 00 F3 F6 2B   |...w...........+|
3 00000020  70 00 00 0A D1 69 43 43 50 49 43
  43 20 50 72 6F   |p....iCCPICC Pro|
4 00000030  66 69 6C 65 00 00 48 89 95 97 07
  54 53 69 16 C7   |file..H....TSi..|
5 00000040  BF F7 D2 43 42 4B 08 45 4A E8 4D
  90 5E A5 84 1E   |...CBK.EJ.M.^...|
6 00000050  40 41 3A D8 08 49 48 42 09 21 05
  15 3B 32 38 02   |@A:..IHB.!..;28.|
7 00000060  23 8A 8A 08 96 01 1D 8A 28 38 16
```

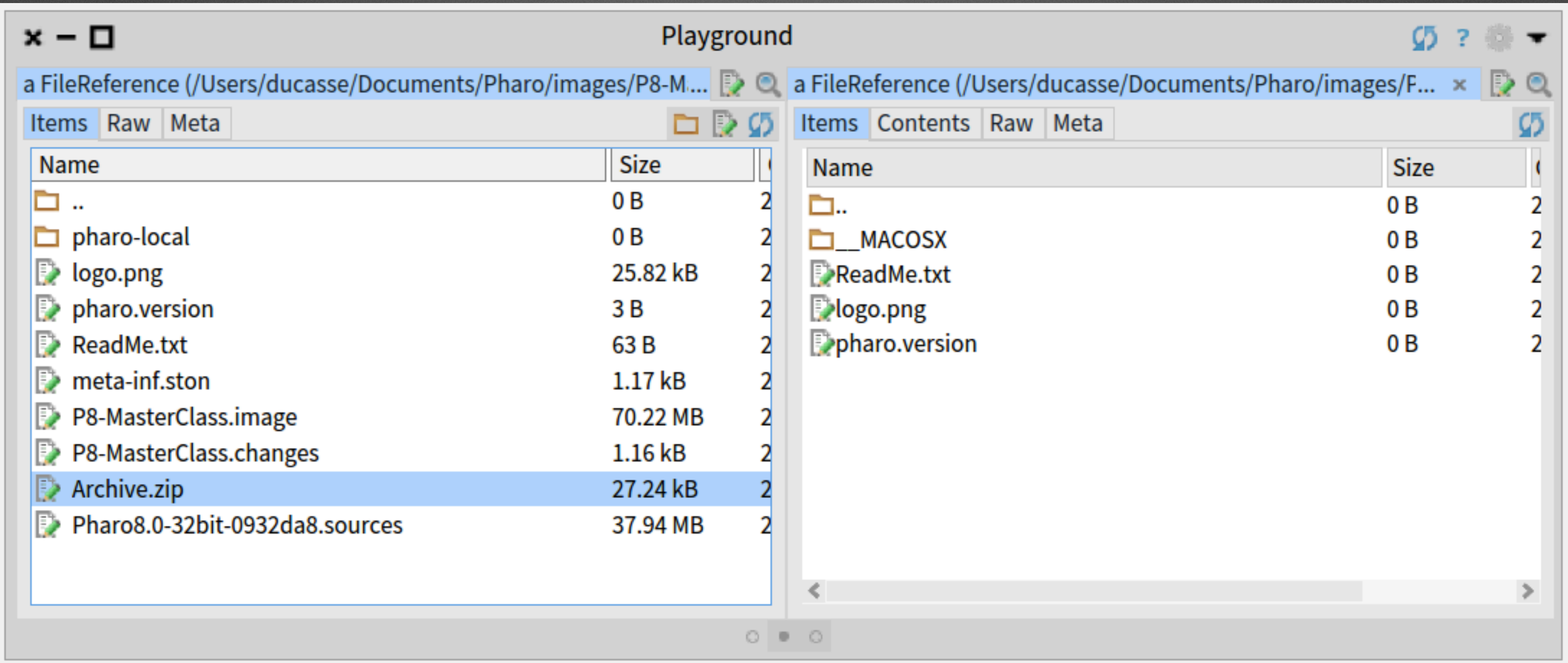
But still a file reference!



See! an archive ‘.zip’

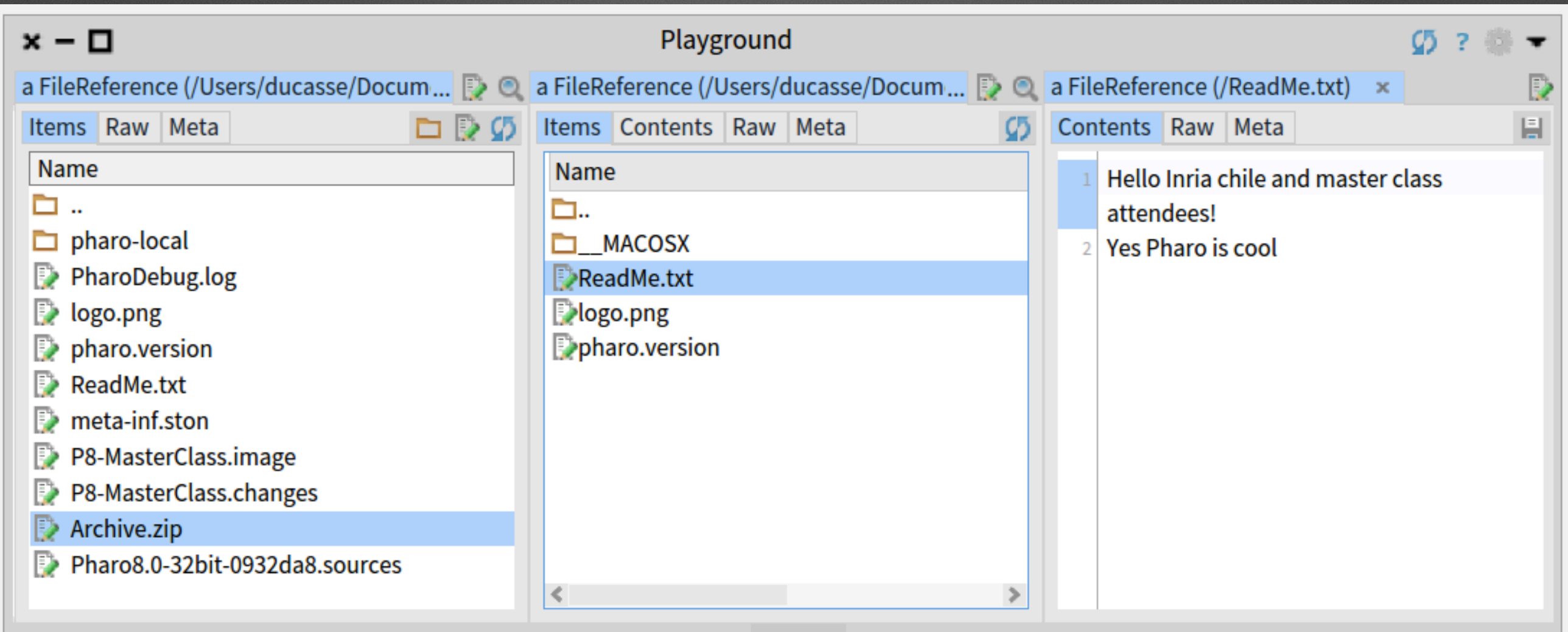


Kind of clear…



An object can expose multiple interactive views!

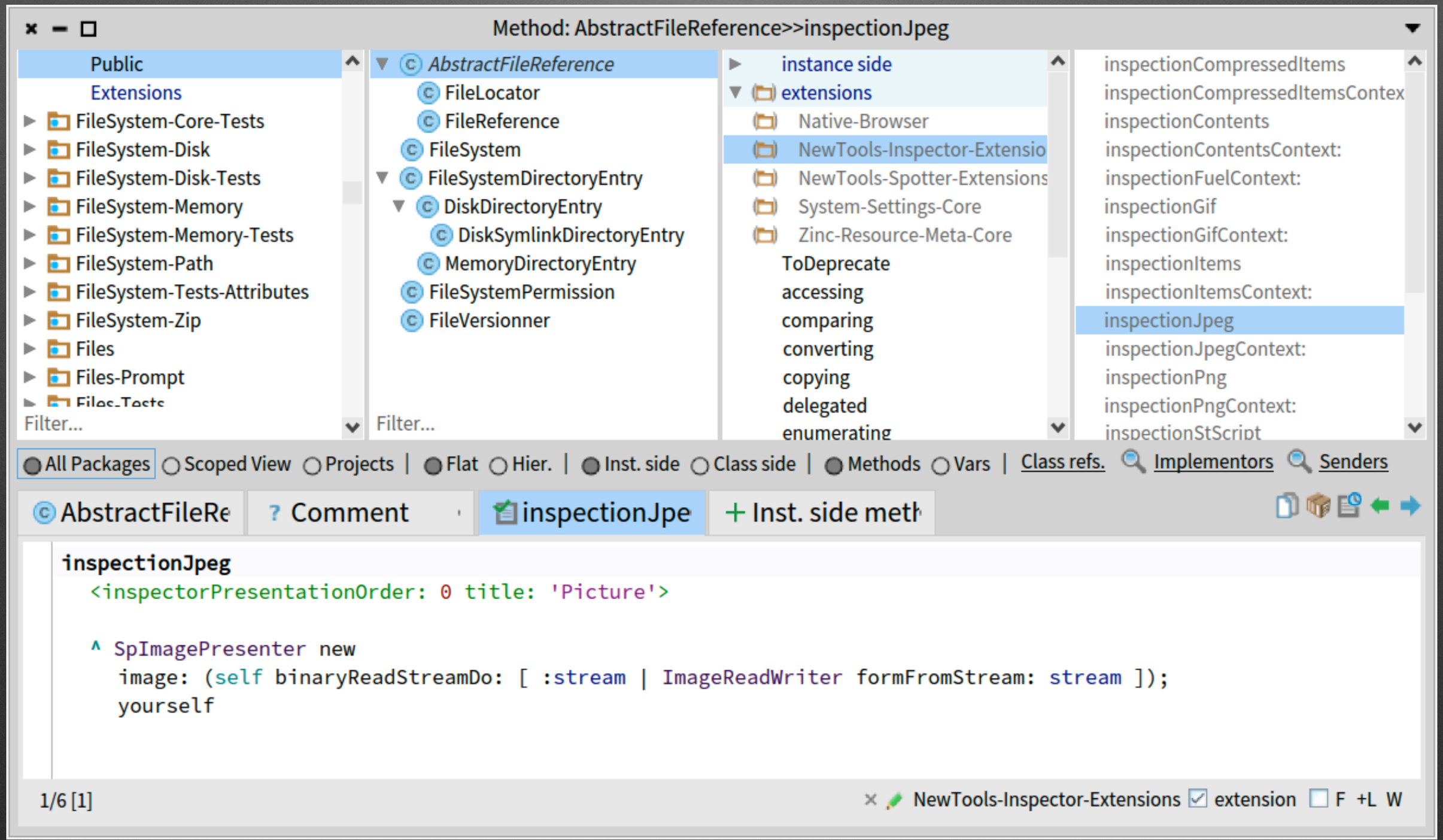
- You can use the best view for your task!
- You can add views to your domain objects



It is cool but it is not magic

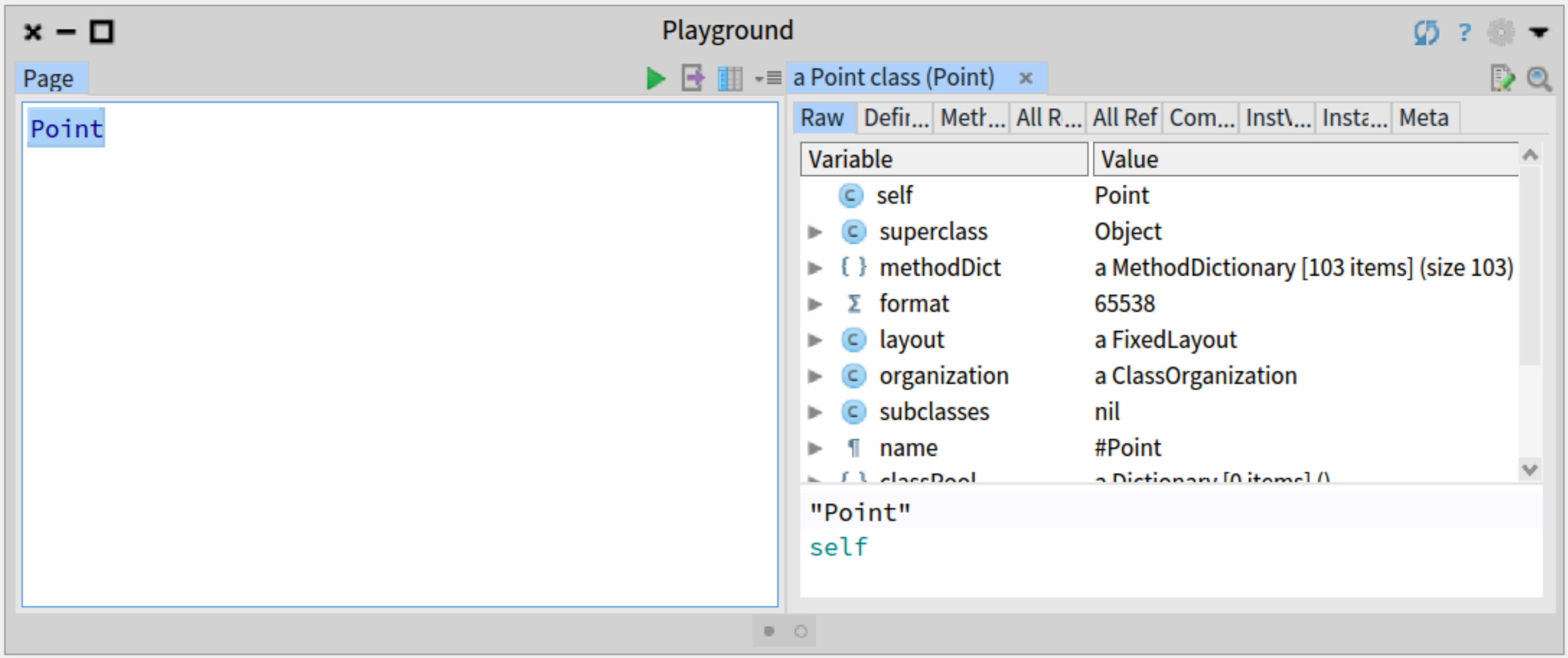
You can define your own

Implementing a pane!

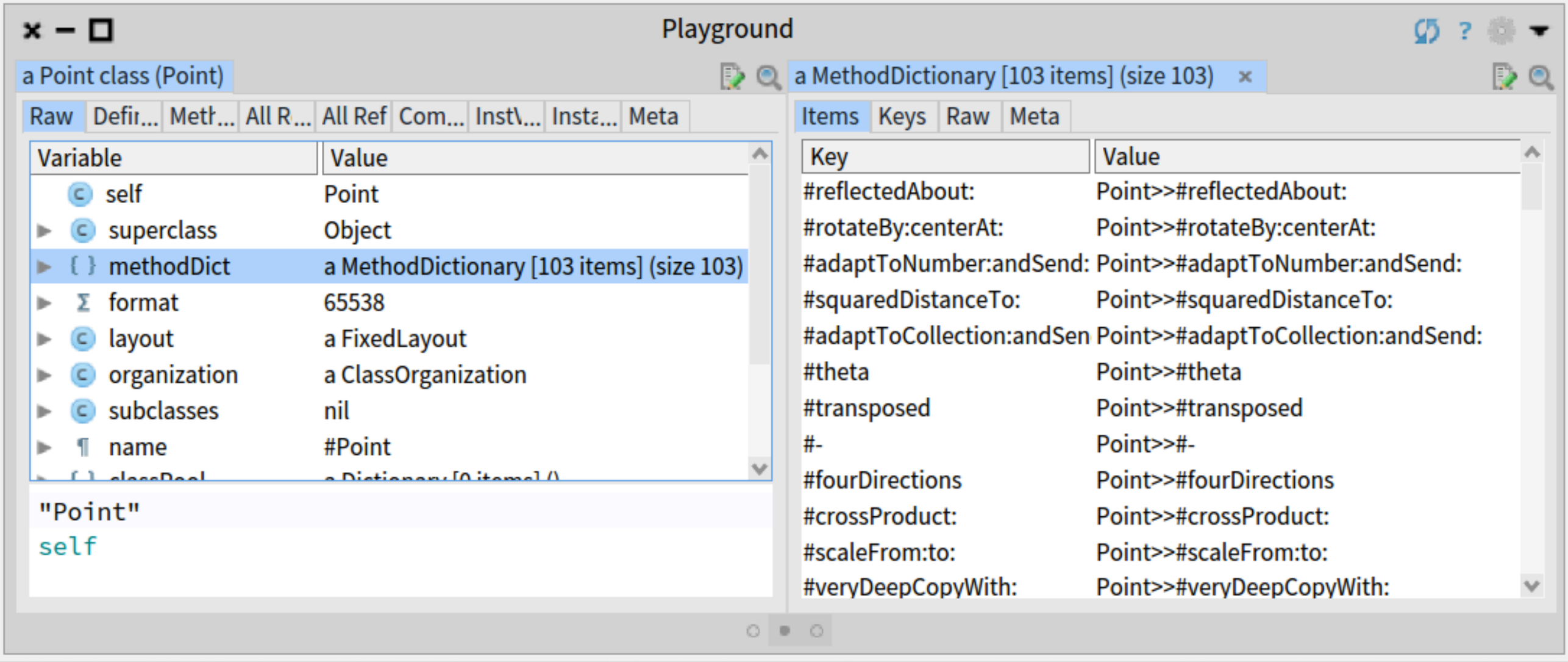


Files are boring… What about *inside* the system?

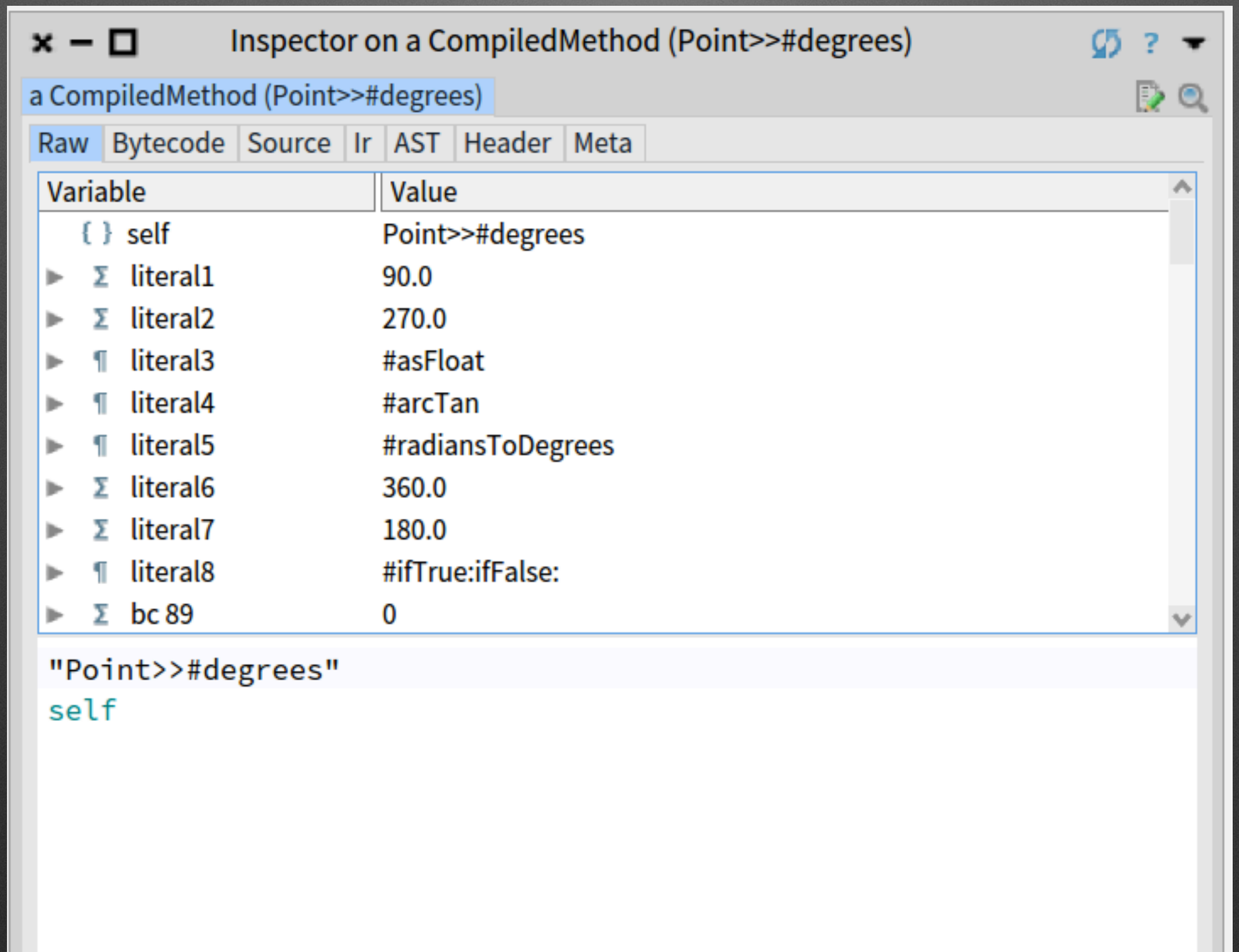
A class is an object we can inspect!



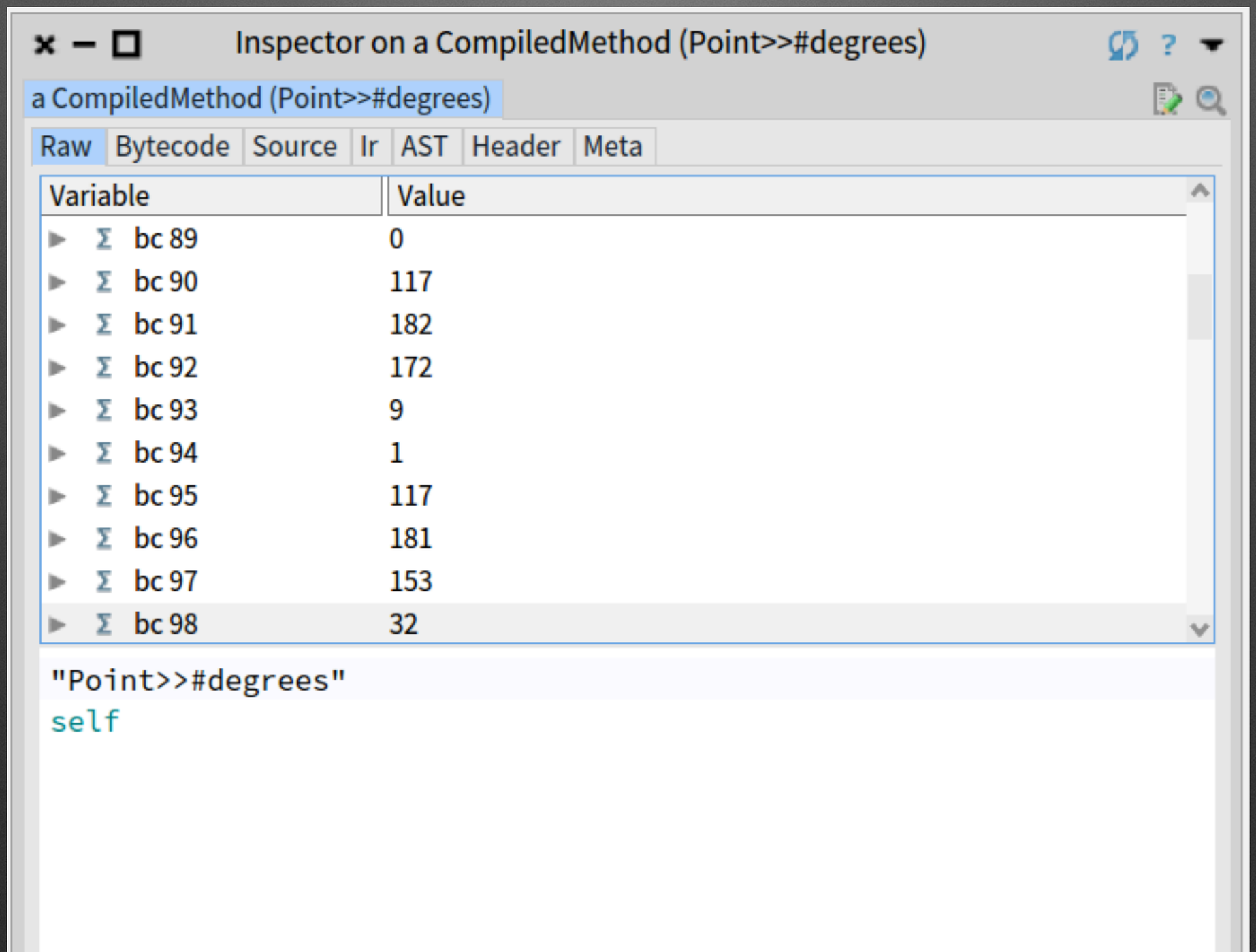
“A class has a method dictionary”
they said… let us verify



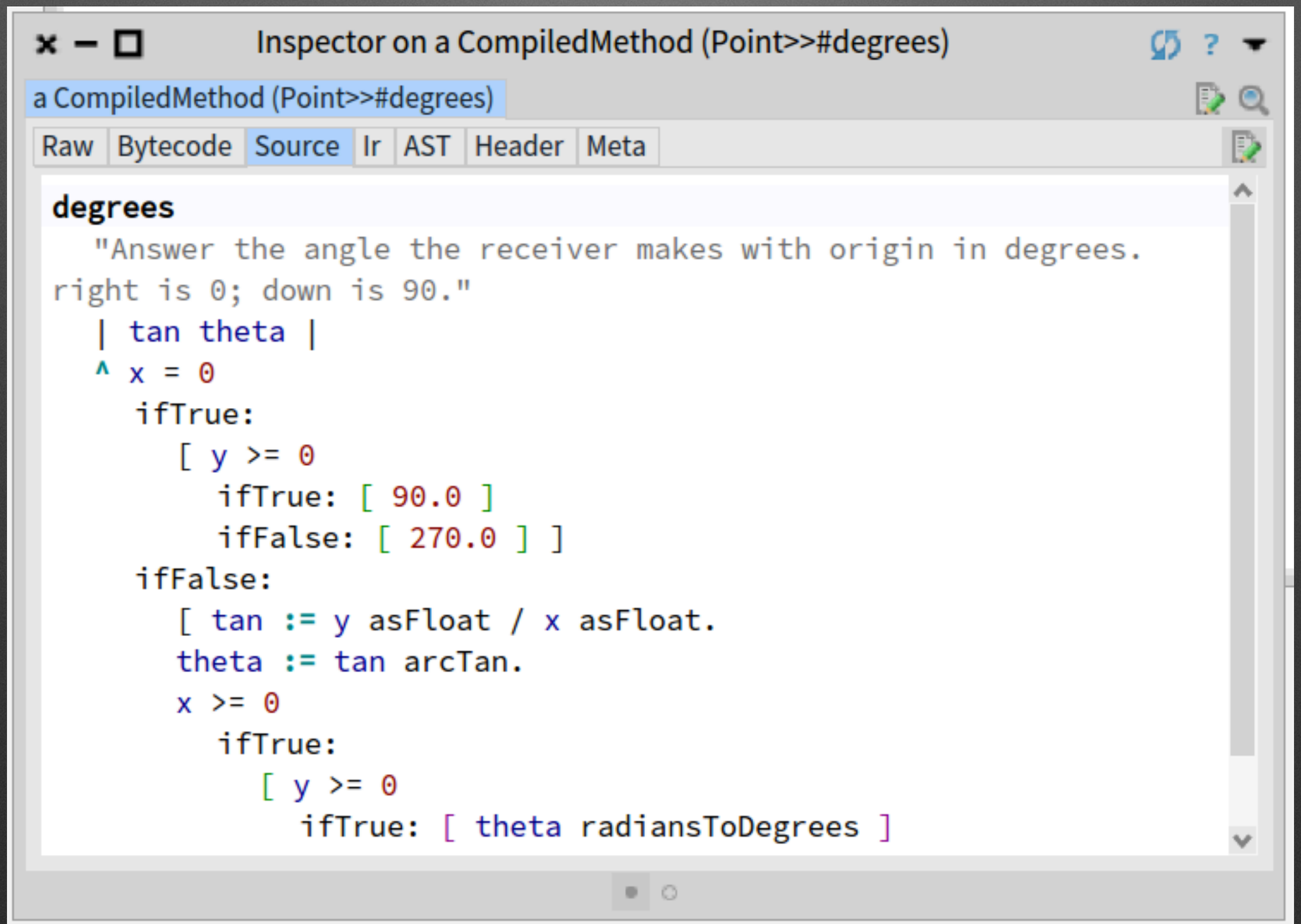
Dissecting one method object



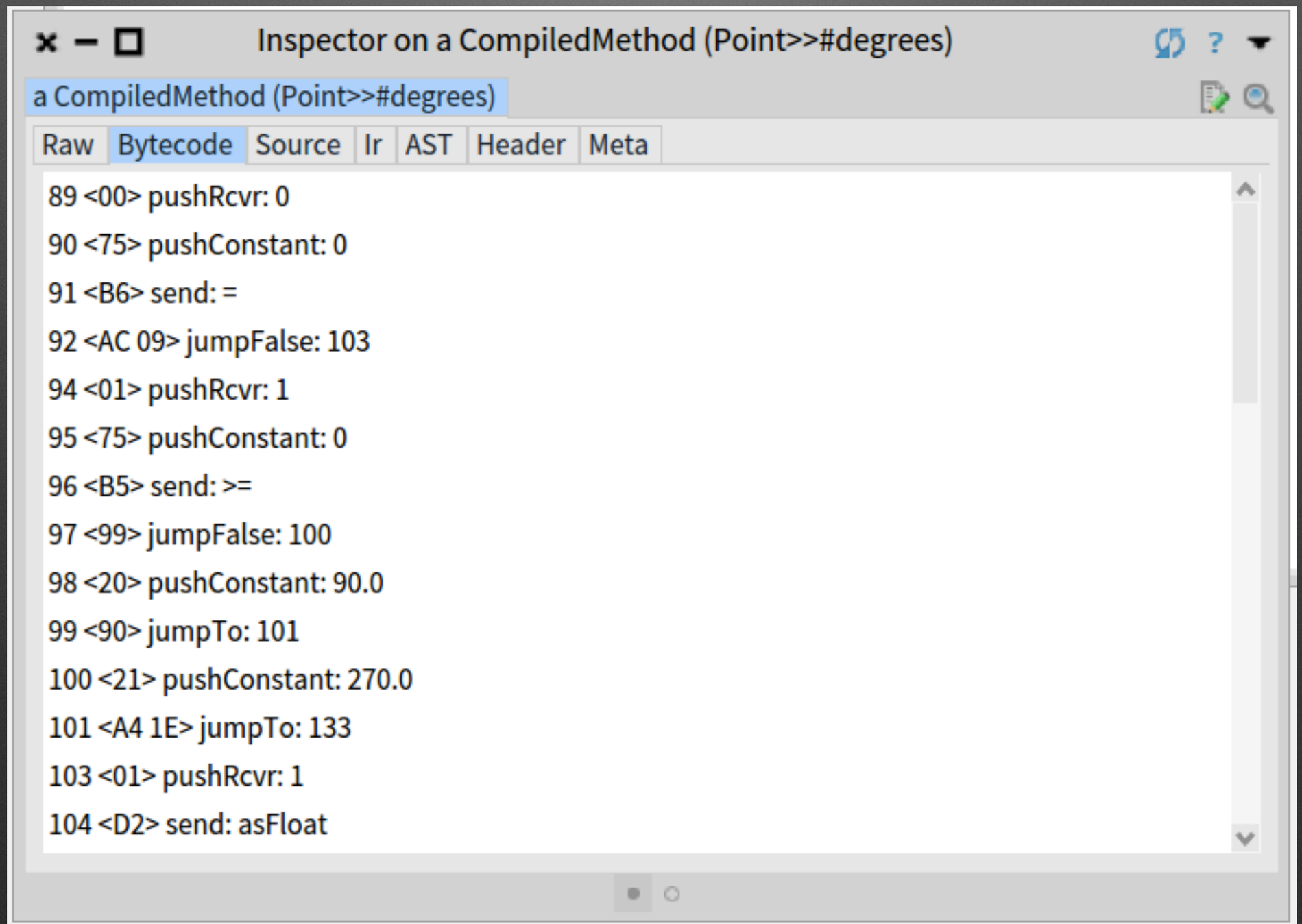
I do not want to be a compiler!



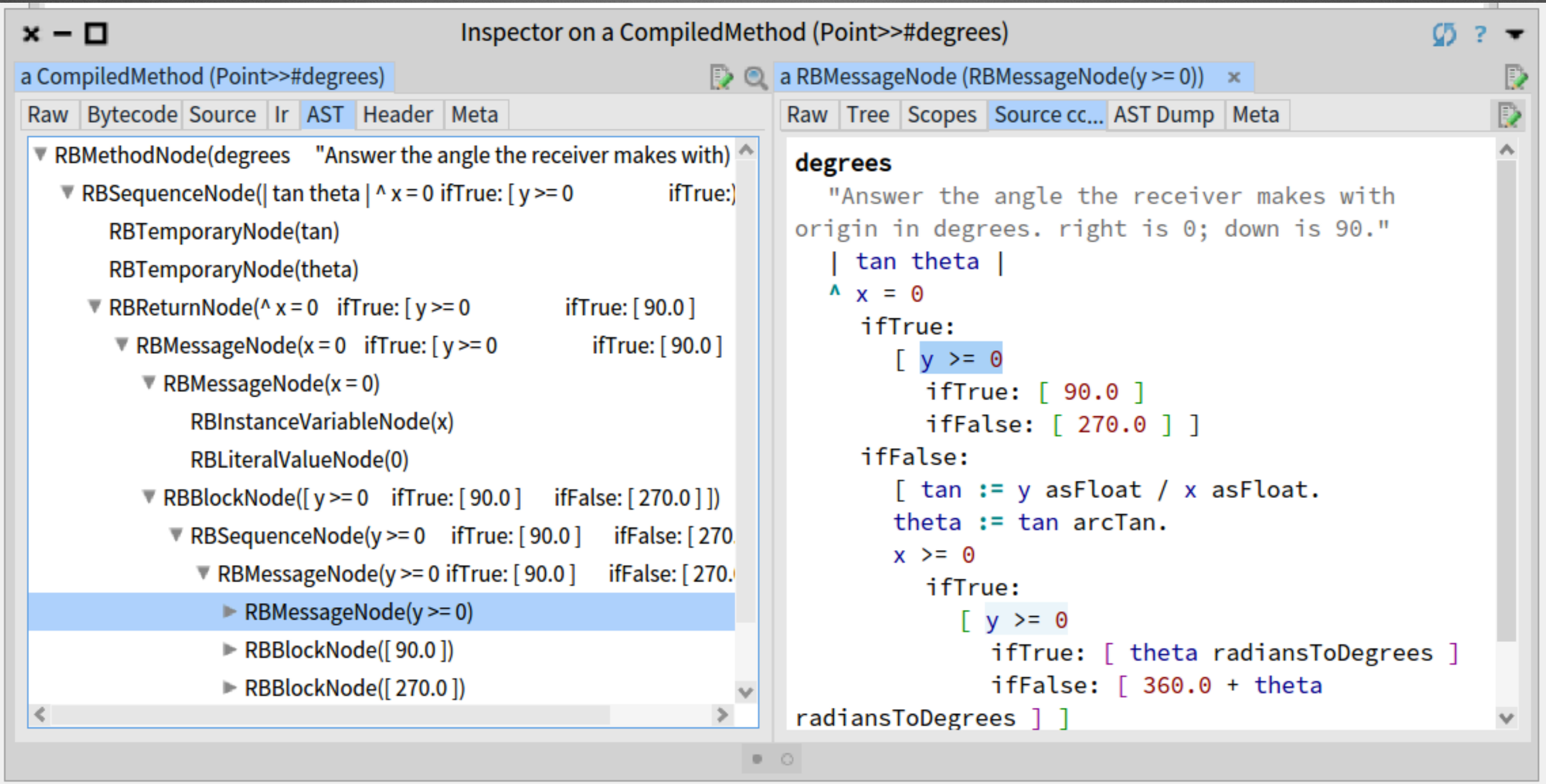
It looks like a method



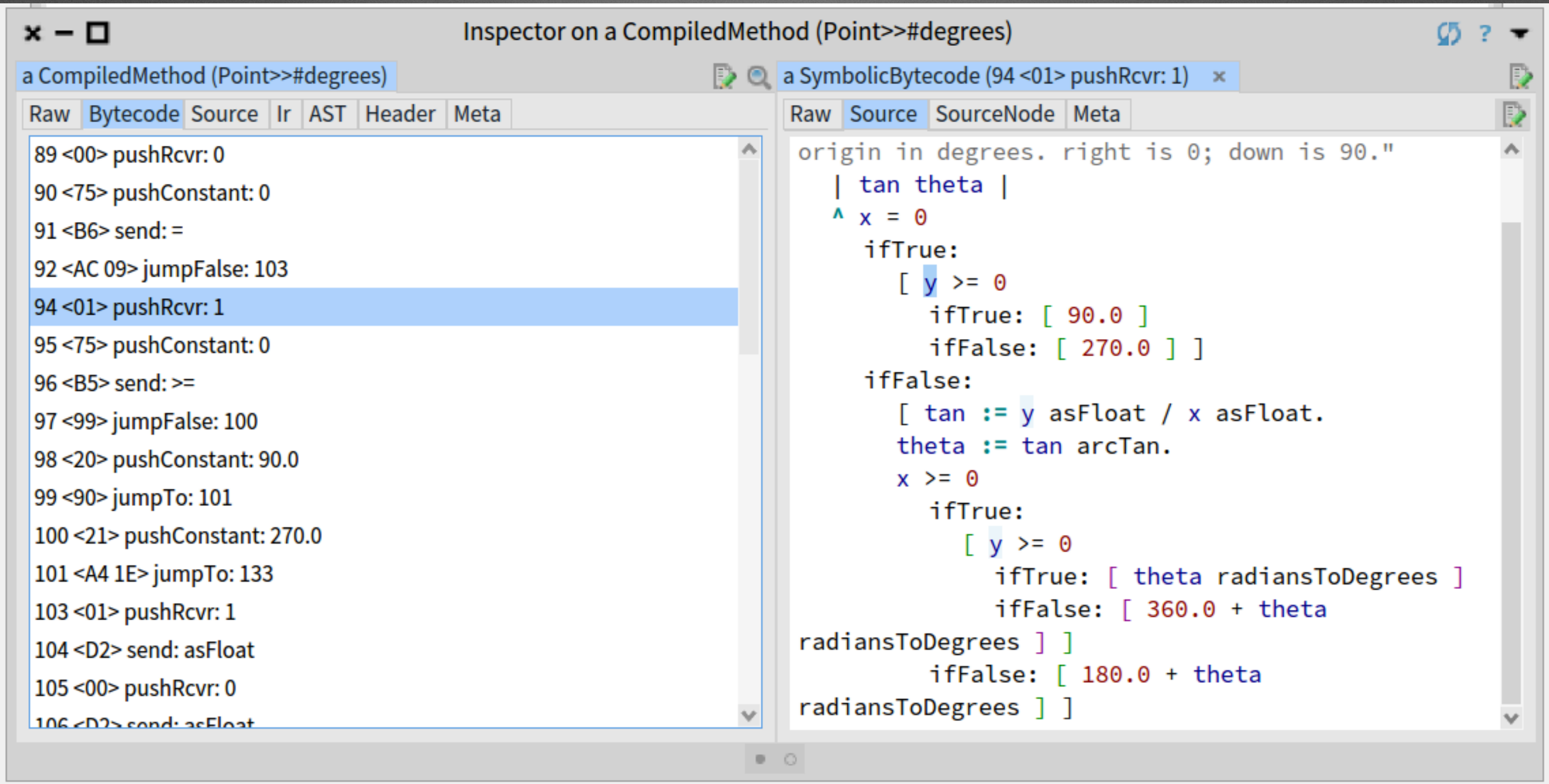
Numbers are not that obscure



And mapping them to the good abstraction helps

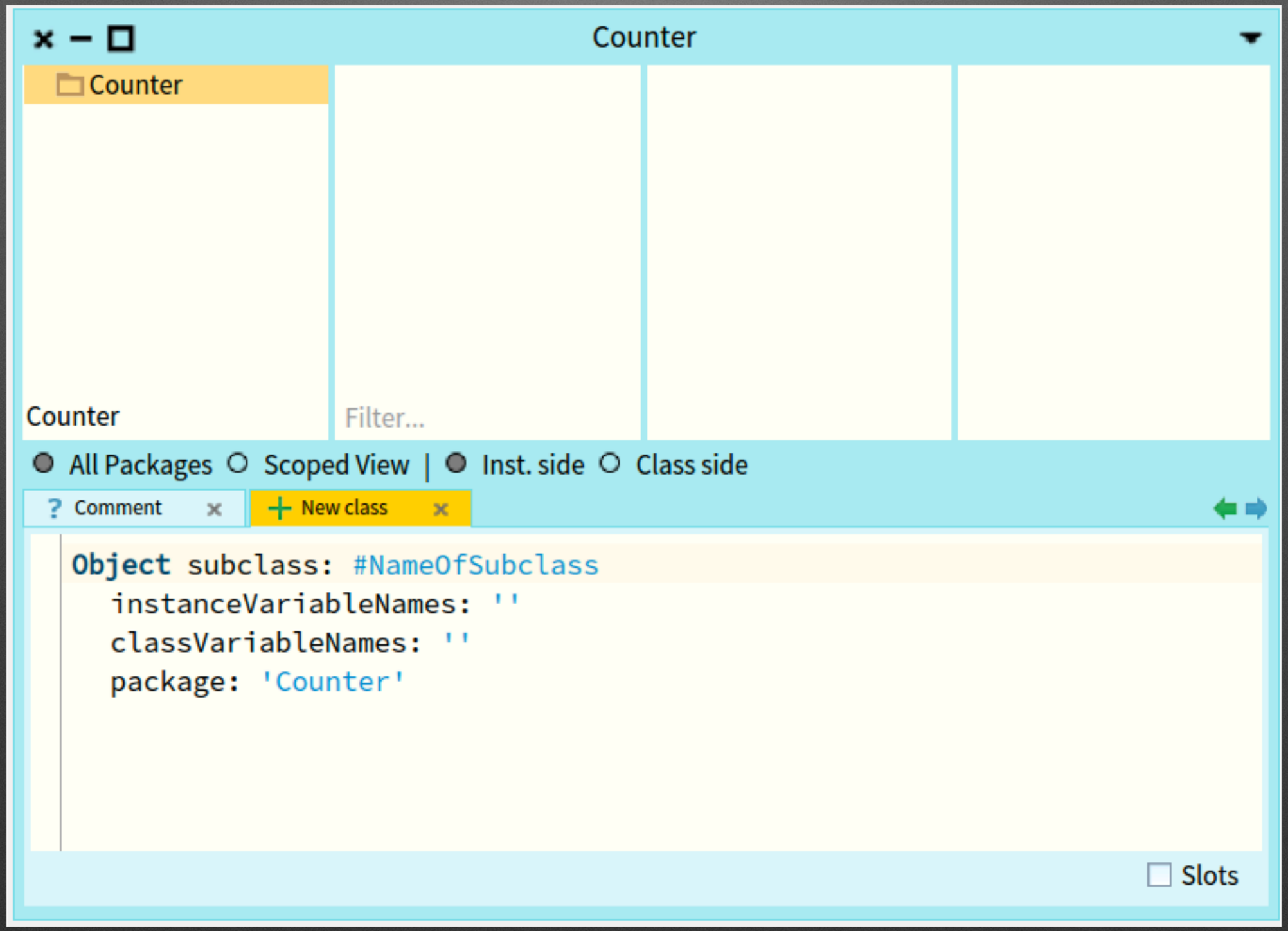


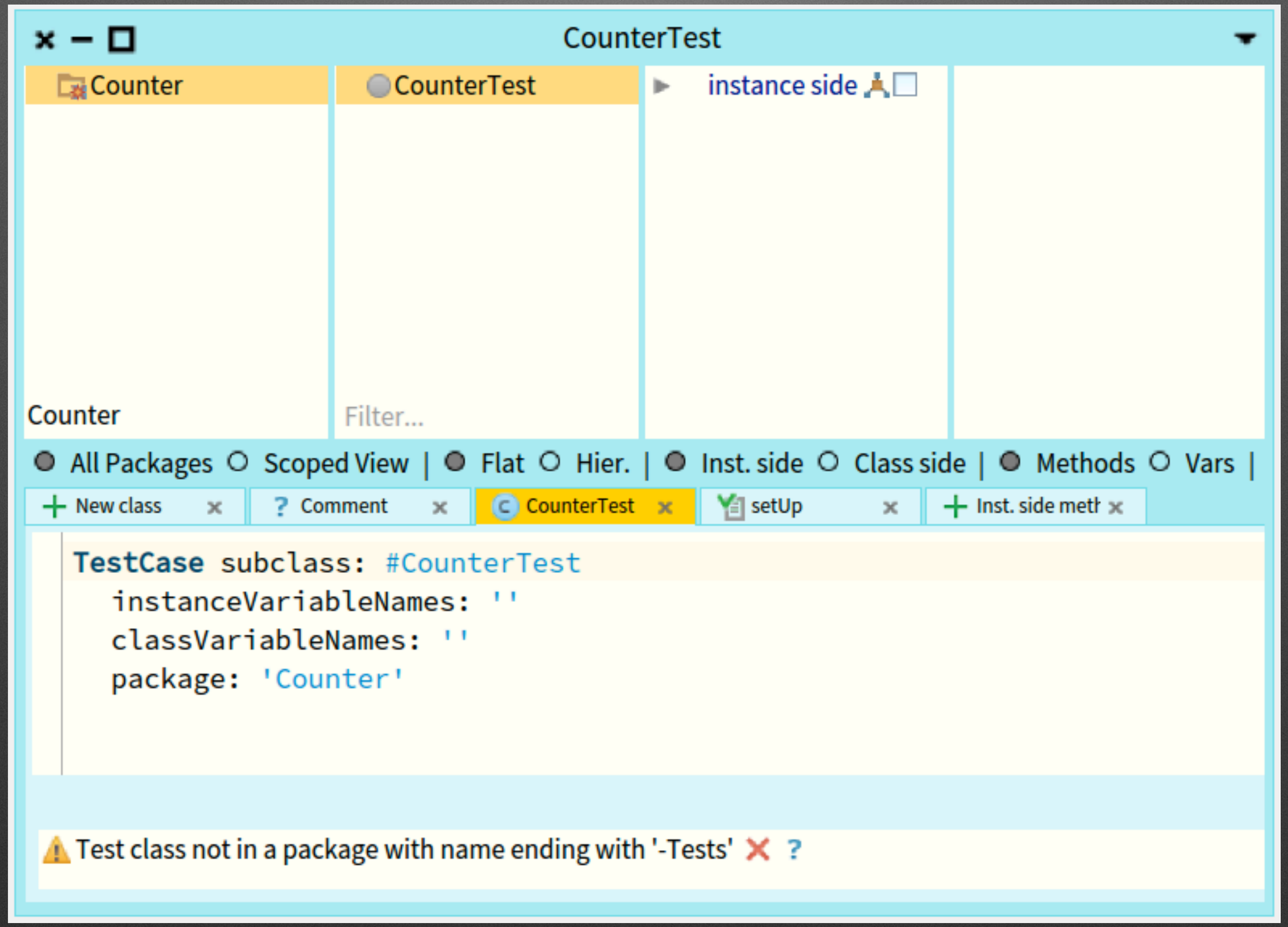
Yes pushRcvr: 1 means the second field!



Pharo Pro devs do XtremeTDD

- Get productivity boost
- Xtreme TDD
  - write test, test fails and
  - code in debugger


Counter
Counter
Counter
Filter...
All Packages
Scoped View
Inst. side
Class side
Comment
New class
Object subclass: #NameOfSubclass
instanceVariableNames: ''
classVariableNames: ''
package: 'Counter'
Slots


CounterTest
Counter
CounterTest
instance side
Counter
Filter...
All Packages
Scoped View
Flat
Hier.
Inst. side
Class side
Methods
Vars
New class
Comment
CounterTest
setUp
Inst. side meth
TestCase subclass: #CounterTest
instanceVariableNames: ''
classVariableNames: ''
package: 'Counter'
Test class not in a package with name ending with '-Tests'

CounterTest

Counter | CounterTest | instance side

Counter

Filter...

All Packages | Scoped View | Flat | Hier. | Inst. side | Class side | Methods | Vars

Comment | CounterTest | setUp | *Inst. side met

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

2/2 [55]

tests | extension

CounterTest

Unknown variable: Counter please correct, or cancel:

Leave variable undeclared

Define new class

Declare new global

Declare new class variable

Define new trait

CounterIconStyler

ExecutionCounter

BlockLocalTempCounter

SUnitTestsCounter

PRNotFoundCounter

PRCascadingCounter

CounterTest

ComposablePresenter

ContainerPresenter

CompletionSorter

Cancel

Cancel

ide

Class side | Methods Vars |

de met

equals: 22

2/2 [55]

tests extension

CounterTest

Counter | CounterTest | instance side

Counter

All Packages | S | Vars |

Comment

```
testSetAndGet
    self asser
```

Information Required

Edit class definition:

```
Object subclass: #Counter
        instanceVariableNames: ''
        classVariableNames: ''
        category: 'Counter'
```

OK | Cancel

2/2 [55]

tests | extension

CounterTest>>testSetAndGetCounter

Run tests

Counter | Counter ! | instance side | testSetAndGetCounter

CounterTest | tests

Counter

Filter...

All Packages | Scoped View | Flat | Hier. | Inst. side | Class side | Methods | Vars | Cl

Comment | CounterTest | setUp | testSetAndGet | Inst. side meth | Inst. side n

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

1/2 [1]

Instance of Counter did not understand #count:

Bytecode GT

Stack

+ Create | Proceed | Restart | Step into | Step over | Step through

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTes	SUnit-Core
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

Where is? | Browse

```
testSetAndGetCounter
   self assert: (Counter new count: 22) count equals: 22
```

Variables | Evaluator

Type	Variable	Value
implicit	self	CounterTest>>#testSetAndGetCounter
attribute	expectedFails	an Array [0 items] ()
attribute	testSelector	#testSetAndGetCounter
implicit	thisContext	CounterTest>>testSetAndGetCounter

Create the missing class or method in the user prompted class, and restart the debugger at the location where it can be edited.

Instance of Counter d

Bytecode GT

Stack

+ Create | Proceed | Restart | Step into | Step over | Step through

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTes	SUnit-Core
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

Where is? | Browse

```
testSetAndGetCounter
   self assert: (Counter new count: 22) count equals: 22
```

Variables | Evaluator

Type	Variable	Value
implicit	self	CounterTest>>#testSetAndGetCounter
attribute	expectedFails	an Array [0 items] ()
attribute	testSelector	#testSetAndGetCounter
implicit	thisContext	CounterTest>>testSetAndGetCounter

Instance of Counter did not understand #count:

Bytecode GT

Stack

Proceed Restart Step into Step over Step through

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTes	SUnit-Core

Source

Where is? Browse

```
count: anInteger
   self shouldBeImplemented.
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
paramete	anInteger	22
implicit	thisContext	Counter>>count:
implicit	stack top	22

Instance of Counter did not understand #count:

Bytecode GT

Stack

Proceed Restart Step into Step over Step through

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

Where is? Browse

```
count: anInteger
   count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
paramete	anInteger	22
implicit	thisContext	Counter>>count:
implicit	stack top	22

Instance of Counter did not understand #count:

Bytecode GT

Unknown variable: count please correct, or cancel:

Declare new temporary variable
Declare new instance variable
Cancel

Cancel

Restart Step into Step over Step through

Other	Package
	Counter
	Counter
	SUnit-Core
[self setUp. self performTes	SUnit-Core

Source

Where is? Browse

```
count: anInteger
    count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
paramete	anInteger	22
implicit	thisContext	Counter>>count:
implicit	stack top	22

Instance of Counter did not understand #count:

Bytecode GT

Stack

Proceed | Restart | Step into | Step over | Step through

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

Where is? | Browse

```
count: anInteger
   count := anInteger
```

Variables | Evaluator

Type	Variable	Value
implicit	self	a Counter
paramete	anInteger	22
attribute	count	nil
implicit	thisContext	Counter>>count:

Relinquish debugger control and proceed execution from the current point of debugger control.cmd+r

Instance of Counter did not und

Bytecode GT

Stack

Proceed Restart Step into Step over Step through

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTes	SUnit-Core

Source

Where is? Browse

```
count: anInteger
   count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
paramete	anInteger	22
attribute	count	nil
implicit	thisContext	Counter>>count:

Instance of Counter did not understand #count

Bytecode GT

Stack

+ Create | Proceed | Restart | Step into | Step over | Step through

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Cor
CounterTest(TestCase)	runCase	[self setUp. self performTes	SUnit-Cor
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

Where is? | Browse

```
testSetAndGetCounter
   self assert: (Counter new count: 22) count equals: 22
```

Variables | Evaluator

Type	Variable	Value
implicit	self	CounterTest>>#testSetAndGetCounter
attribute	expectedFails	an Array [0 items] ()
attribute	testSelector	#testSetAndGetCounter
implicit	thisContext	CounterTest>>testSetAndGetCounter

Instance of Counter did not understand #count

Bytecode GT

Stack

Proceed Restart Step into Step over Step through

Class	Method	Other	Package
Counter	count		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Cor
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Cor

Source

Where is? Browse

```
count
    ^ count
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
attribute	count	22
implicit	thisContext	Counter>>count
implicit	stack top	nil

CounterTest>>testSetAndGetCounter

Counter
Counter
Filter...

Counter !
CounterTest

instance side
tests

testSetAndGetCounter

All Packages | Scoped View | Flat | Hier. | Inst. side | Class side | Methods | Vars | Cl

Comment | CounterTest | setUp | testSetAndGet | Inst. side meth | Inst. side n

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

1/2 [1]

Micro commits are your friends

Micro commits?

- Record all your activities
- Support the replay, query and reload of actions

Going back in time

Recent versions of Counter>>#increment

Browse | References | Senders | Implementors | Revert | Side By Side

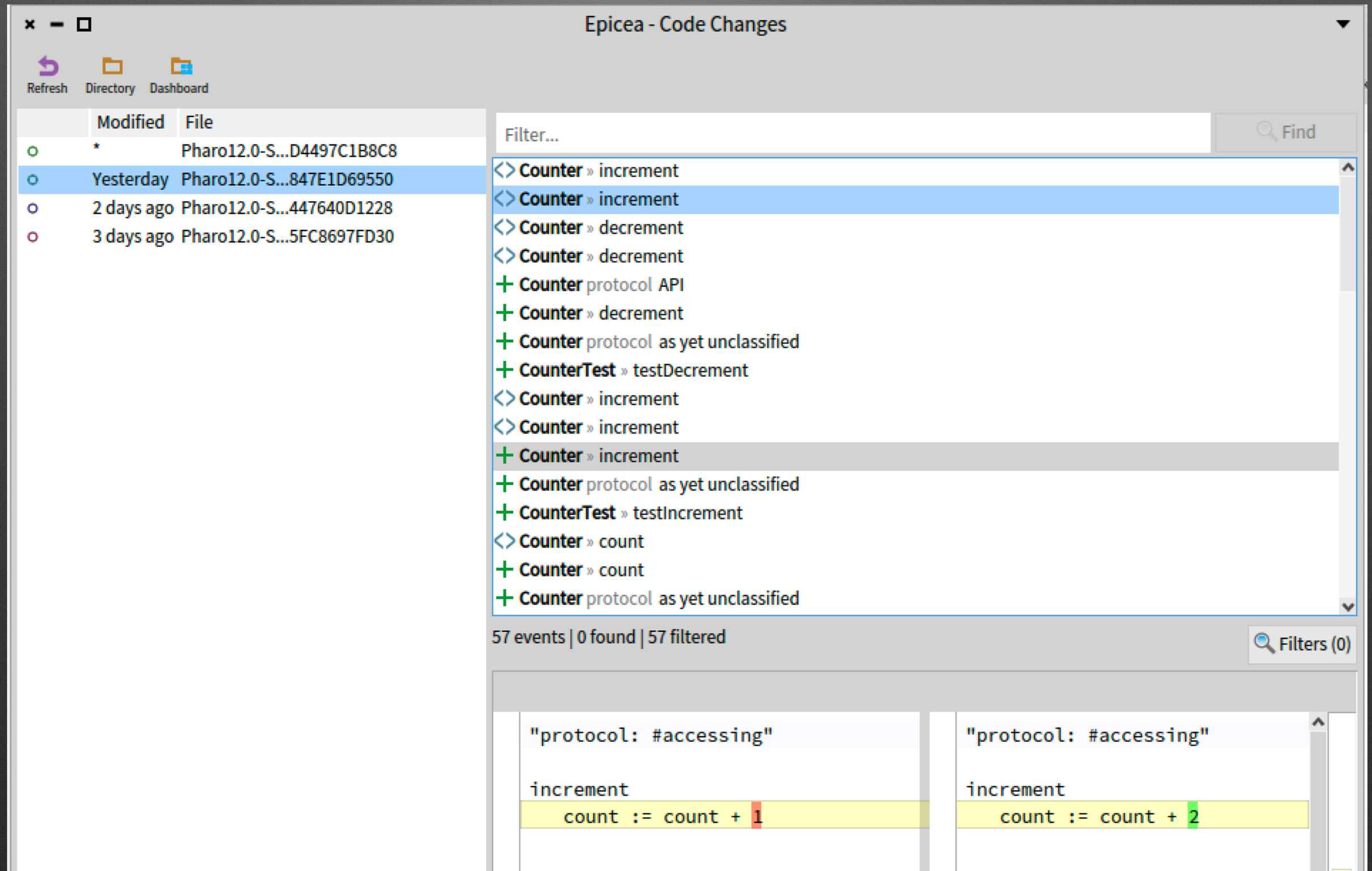
Location	Selector	Package	Timestamp
Counter (accessing)	increment	[]	1901-01-01T00:00:00+00:00
Counter (accessing)	increment	[]	1901-01-01T00:00:00+00:00
Counter (accessing)	increment	[]	1901-01-01T00:00:00+00:00
Counter (accessing)	increment	[]	1901-01-01T00:00:00+00:00
Counter (accessing)	increment	[]	1901-01-01T00:00:00+00:00

Pretty print

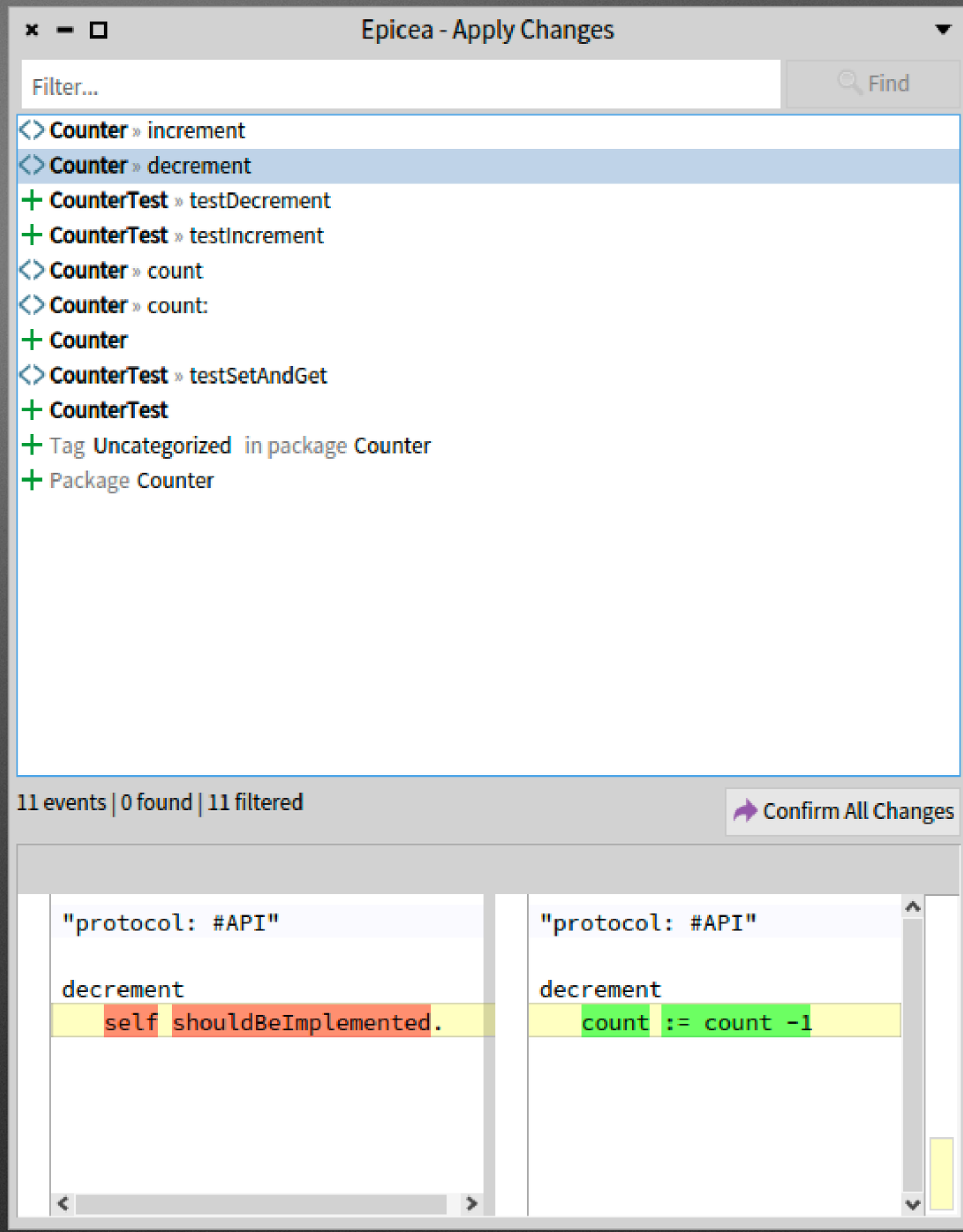
```
increment
    self halt.
    count := count + 2
```

```
increment
    count := count + 1
```

Navigating/Querying past activities



Selective Replay

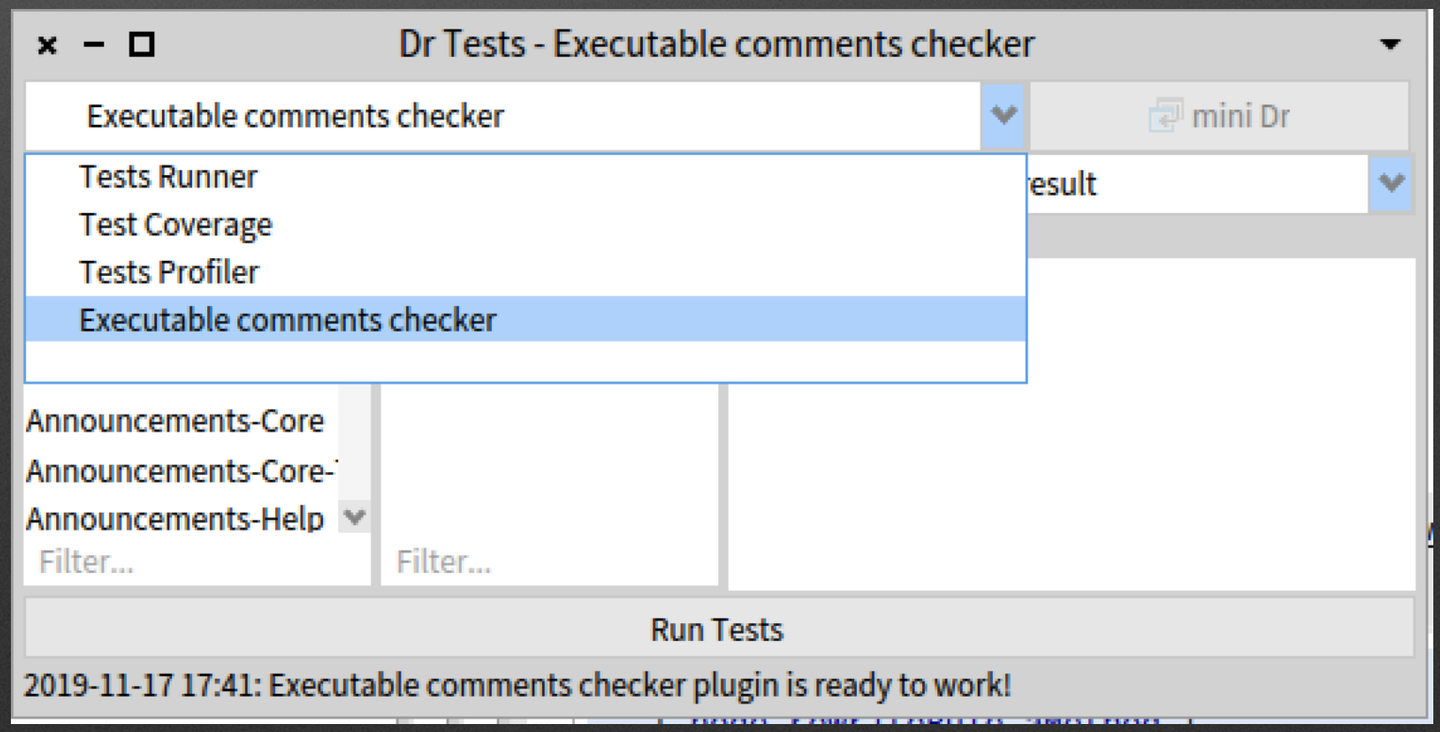


What about tests?

Tests are also our best friends

- DrTests
  - Plugin for Rotten Green tests
    - Executable comments
- Mutalk: Mutation analyser framework

DrTests: a plugin-based architecture to plug test analyses



We validated our 27000 tests for Rotten Green Tests (ICSE'19)

Hot update on the fly customizable debugger

QuittanceGenerator class>>#exampleJustMonthAndRenter

Halt

Bytecode

Stack

Proceed Restart Into Over Through

PDFCellElement	getSubElementsWith:styleSheet:
PDFCellElement(PDFComposite)	generateCodeSegmentsCollectionWi
PDFCellElement(PDFComposite)	generateCodeSegmentWith:styleShe
PDFDataTableElement(PDFComposite)	generateCodeSegmentsCollectionWi [:aSubElement | aSubElement generateCodeSe
Array(SequenceableCollection)	collect:

Source

Where is? Browse

```
generateCodeSegmentsCollectionWith: aPDFGenerator styleSheet: compositeStyleSheet format: aFormat
    ^ (self getSubElementsWith: aPDFGenerator styleSheet: compositeStyleSheet)
        collect: [ :aSubElement |
            aSubElement
                generateCodeSegmentWith: aPDFGenerator
                styleSheet: (aSubElement buildCompositeStyleSheetFrom: compositeStyleSheet)
                format: aFormat ]
```

Variables

Type	Variable	Value
implicit	self	a PDFCellElement
parameter	aFormat	a PDFA4Format
parameter	aPDFGenerator	a PDFGenerator
parameter	compositeStyleSheet	a StyleSheet

```
        dimension: 80 mm @ 20 mm;
```

debug it

Bytecode GT

Parser Stack | Through Action | Into Reduce | Next Token

Symbol	Value
Expression	a JSUnaryExpressionNode
")"	{)(238498,238498,#(23))}
"{"	{{(238499,238499,#(25))}
StatementList	an OrderedCollection(a JSExpressionStatementNode a JSI
"("	{((238678,238678,#(28))}
"function"	{function(238679,238686,#(53 82))}
Identifieropt	nil
"("	{((238687,238687,#(28))}
FormalParameterListopt	an Array(an OrderedCollection() an OrderedCollection())
")"	{)(238688,238688,#(23))}
"{"	{{(238689,238689,#(25))}
SourceElements	an OrderedCollection(a JSVariableStatementNode a JSVar
MemberExpression	a JSNameExpressionNode
"."	{.(274447,274447,#(38))}

Possible Actions

Symbol	Action
<identifier>	Shi
IdentifierName	Shi

GLR

1

A specific parser compiler debugger

Input | Step to Cur | Next Character

```
            return t._stash(this); // dojo.NodeList
        }

    });

    nl.events = [
        // summary:
        //          list of all DOM events used in NodeList
        "blur", "focus", "change", "click", "error", "keydown", "keypress",
        "keyup", "load", "mousedown", "mouseenter", "mouseleave", "mousemove",
        "mouseout", "mouseover", "mouseup", "submit"
    ];

    // FIXME: pseudo-doc the above automatically generated on-event functions

    // syntactic sugar for DOM events
    d.forEach(nl.events, function(evt){
```

Nam	Value
State	default

Stack | Proceed | Restart | Into | Over | Through

JSParser(SmaCCGLRParser)	performParsingStep	
JSParser(SmaCCGLRParser)	performParsingLoop	
JSParser(SmaCCParser)	parse	
JSParser class(SmaCCParser class)	parseStream:startingAt:	
JSParser class(SmaCCParser class)	parseFile:	[self parseStrean
BlockClosure	ensure:	
JSParser class(SmaCCParser class)	parseFile:	
UndefinedObject	DoIt	
CompiledMethod	valueWithReceiver:arguments:	
RubSmalltalkEditor	debug:receiver:in:	[aCompiledMethc
BlockClosure	newProcess	[self value.Proces

Source | Where is? | Browse

```
    (tryAllStates and: [ self isEOFToken ])
        ifTrue: [ ^ self processEOFToken ].
    currentToken isPattern
        ifTrue: [ ^ self splitForPatternToken ].
    [ actions := self actionsForCurrentToken.
    actions isEmpty
        ifTrue: [ self killState.
            currentToken := nil.
            ^ self ].
    2 to: actions size do: [ :i | self splitWithAction: (actions at: i) ].
    action := actions first.
    action = self acceptAction
        or: [ self performAction: action.
            currentToken isNil ] ] whileFalse.
    action = self acceptAction
        ifTrue: [ currentState markAccepted ]
```

Halt in OCDBox>>name:

ack

Class	Method	Context
DBox	name:	UndefinedObject>>DoIt
definedObject	DoIt	CompiledMethod>>valueWithRec
mpiledMethod	valueWithReceiver:arguments:	[aCompiledMethod
bSmalltalkEditor	debug:receiver:in:	[self value.

ceed | Into | Over | Through | Run to | Restart | Return | Where is? | Create

lt

```
1 name: anObject
2     self halt.
3     name := anObject
```

Bytecode | Breakpoints | Sindari

Type	Target	Method
Breakpoint	self	Reflecti
Breakpoint	self	OCDBox>>initialize
Breakpoint	self	OCDBox>>name:
Breakpoint	self	OCDBox>>name:
Halt	OCDBox	OCDBox>>name:
Breakpoint	OCDBox	OCDBox>>name:
Breakpoint	self	OCDBox>>removeElement:
Breakpoint	self	OrderedCollection>>removeI
Halt	StHaltCacheTest	StHaltCacheTest>>testInitial
Halt	StHaltCacheTest	StHaltCacheTest>>testInitial

```
1 name: anObject
2     self halt.
3     name := anObject
```

Specific view

Bytecode | Breakpoints | Sindarin

```
33 <4C> self
34 <80> send: halt
35 <D8> pop
36 <40> pushTemp: 0
37 <C9> popIntoRcvr: 1
38 <58> returnSelf
```

ceiver in: a StDebuggerContext (OCDBox>>name:)

an OCDBox

Variable	Value
[arg] anObject	'i'
self	an OCDBox
{ } elements	an OrderedCollection [0 items] ()
name	''
{ } Temps	a Dictionary [1 item] (#anObject->'i')
anObject	'i'
stackTop	an OCDBox

Raw | Breakpoints | Meta

Type	Target	Method
Halt	OCDBox	OCDBox>>name:

```
1
```

```
1 "an OCDBox"
2 self
```

Variable	Value
stackTop	an OCDBox
0 [anObject]	'i'
{ } rcvr: 0 [elements]	an OrderedCollection [0 iten
rcvr: 1 [name]	''

object centric + adv scripting (AST + call stack)

Rece n: a StDebuggerContext (OCDBox>>name:)

an OCDBox

Variable	Value
[arg] anObject	'i'

Raw | Breakpoints | Meta

Variable	Value

Debug actions

- Since Pharo 12
- Toggle set of stop points:
  - toggle off + run the tests + toggle on and code
- Chain stop points: only stop here if you stopped there

Debug Point Browser

(De)activate all

Search by name

Type	Target	Name	Scope
Breakpoint	RBInlineMethodRefactoring>>#pr	Breakpoin	class RBI

Refresh Remove

```
1 privateTransform
2     self
3         renameConflictingTemporaries;
4         insertInlinedMethod;
5         compileMethod
```

enabled: (de)activates debug point

Condition: Hit when the condition evaluates to true

```
1
```

Test Environment Only: Hits only when executing tests

Chain: Each debug point is hit once in sequential order

Counter: Tracks how many times the debug point was reached

Once: Deactivates debug point after one hit

Script: Executes a script at each hit

```
1
```

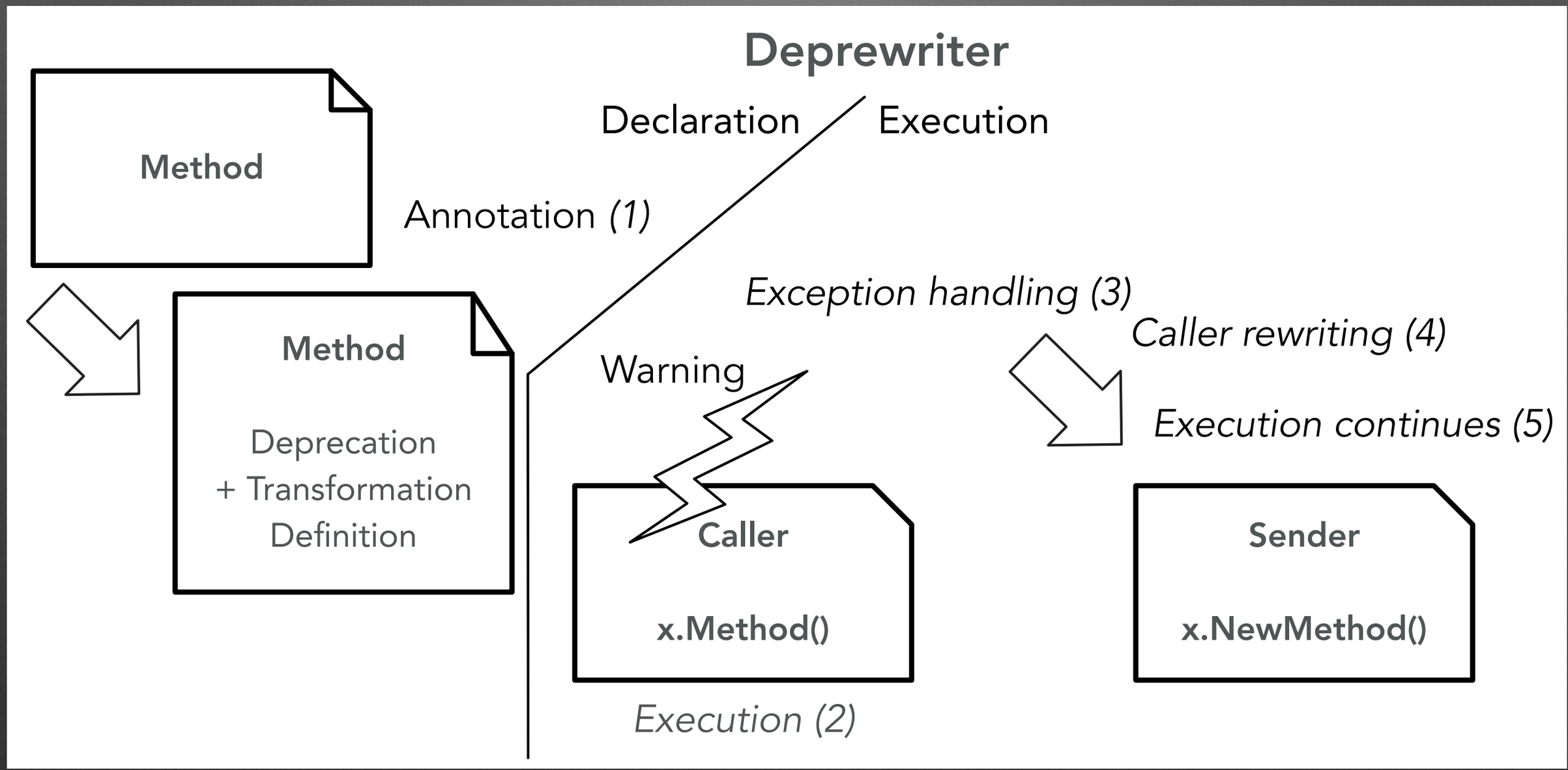
Transcript: Logs to transcript at each hit

Break

Dynamically rewriting deprecated calls @ runtime

How to support migration to new versions

We deprecate API
How to help our users to migrate?


Deprecation
Method
Annotation (1)
Declaration
Execution
Method
Deprecation
+ Transformation
Definition
Exception handling (3)
Warning
Caller
x.Method()
Execution (2)
Caller rewriting (4)
Execution continues (5)
Sender
x.NewMethod()

Rewriting deprecation

```
crLog: aString

    self
        deprecated: 'Please use trace* methods instead.'
        transformWith:
            '`@receiver crLog: `@statements1'
                -> '`@receiver crTrace: `@statements1'.
    self crTrace: aString
```

Run your tests. Now your code and tests use the new API!

At notification time:
Walk the stack
Get caller AST
If should be rewriten
Rewrite it and proceed execution

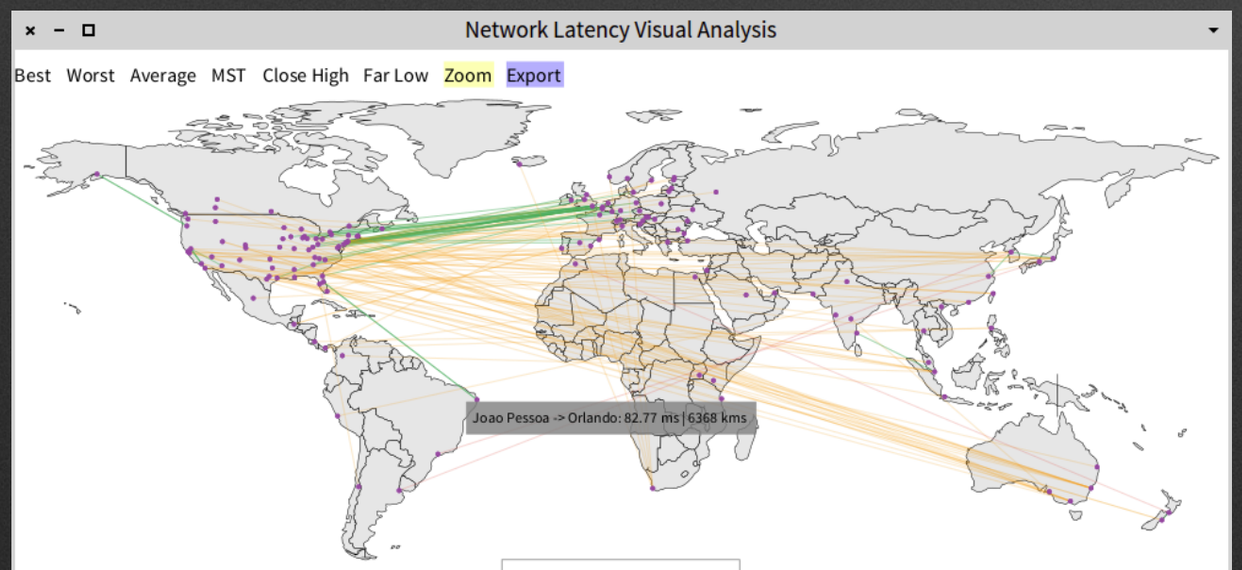
```
transform
    | node rewriteRule aMethod |
    aMethod := self contextOfSender method.

    node := self contextOfSender sourceNodeExecuted.
rewriteRule := self rewriterClass new
    replace: rule key with: rule value.
(rewriteRule executeTree: node)
    ifFalse: [ ^ self signal ].
node replaceWith: rewriteRule tree.
```

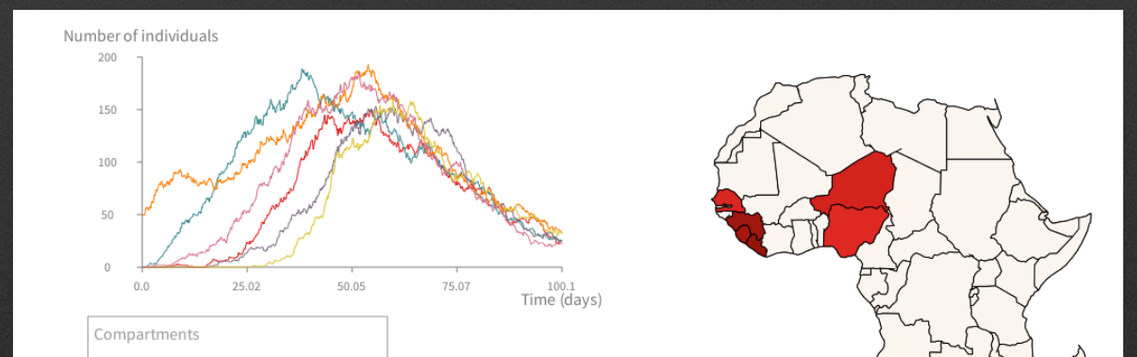
Live visualisation scripting

- Kind of another level
- Roassal 3.0 by Prof. A. Bergel
- Simply gorgeous
- Check http://agilevisualization.com

Includes a DSL for Scripting visualisations

```
b := RTMondrian new.
    b shape rectangle
        withBorder;
        width: [ :cls | cls numberOfVariables * 5 ];
        height: [ :cls | cls numberOfMethods ].

    b nodes: Collection withAllSubclasses.
    b edges connectToAll: [ :cls | cls subclasses ].
    b layout tree.
    b normalizer
        normalizeColorAsGray: [ :cls |
cls numberOfLinesOfCode ].
    b
```

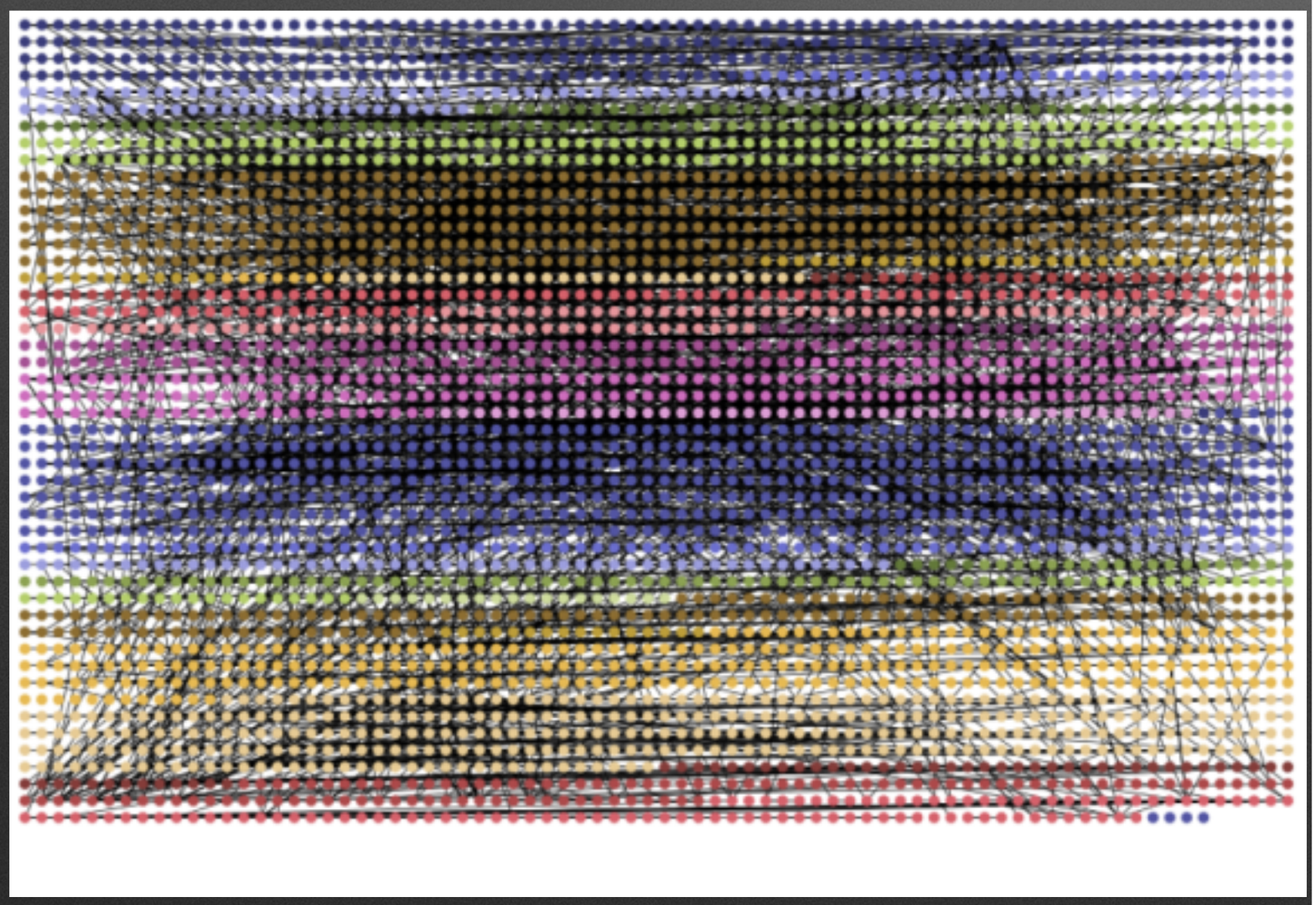


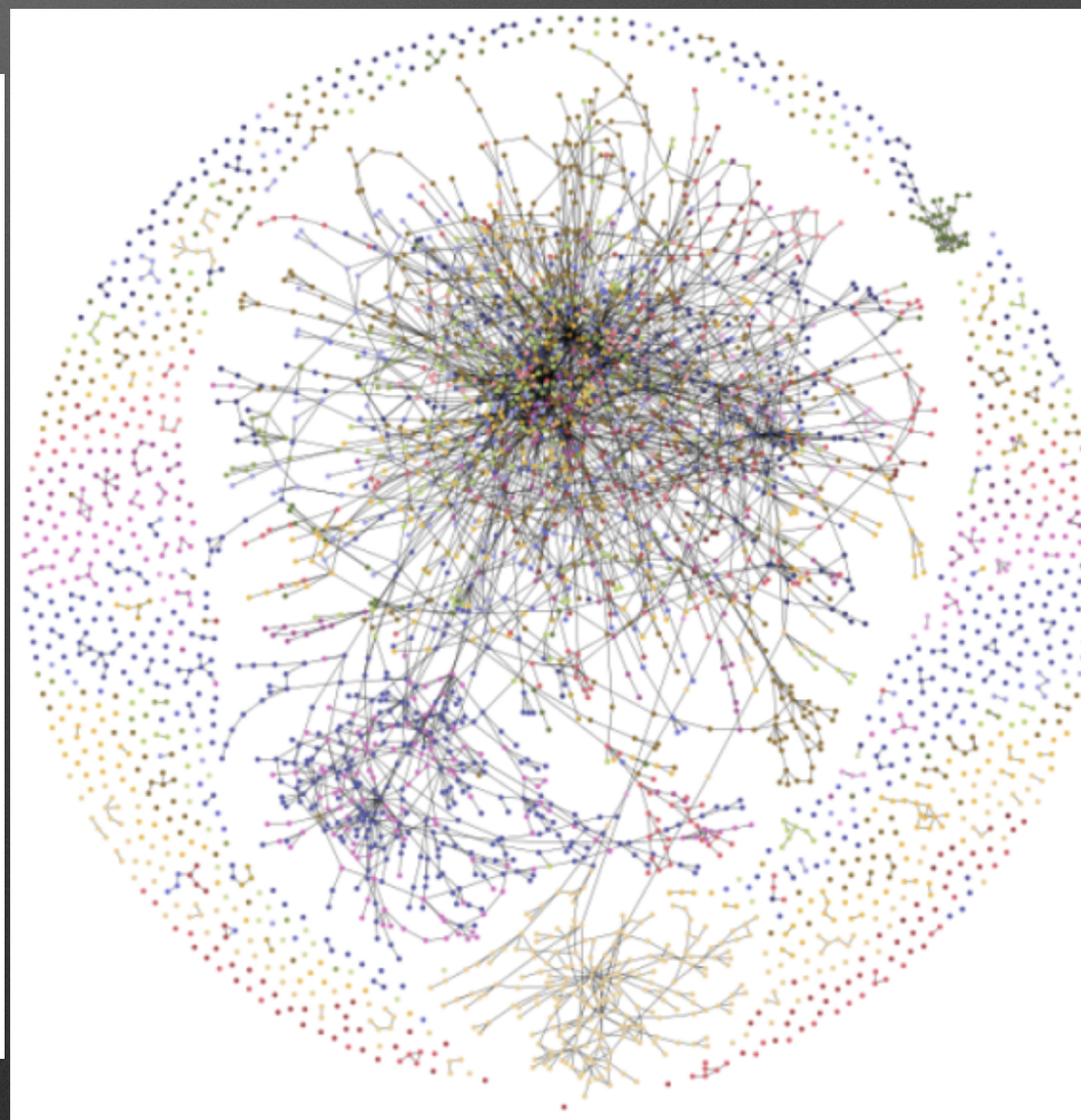


One hour about Basic

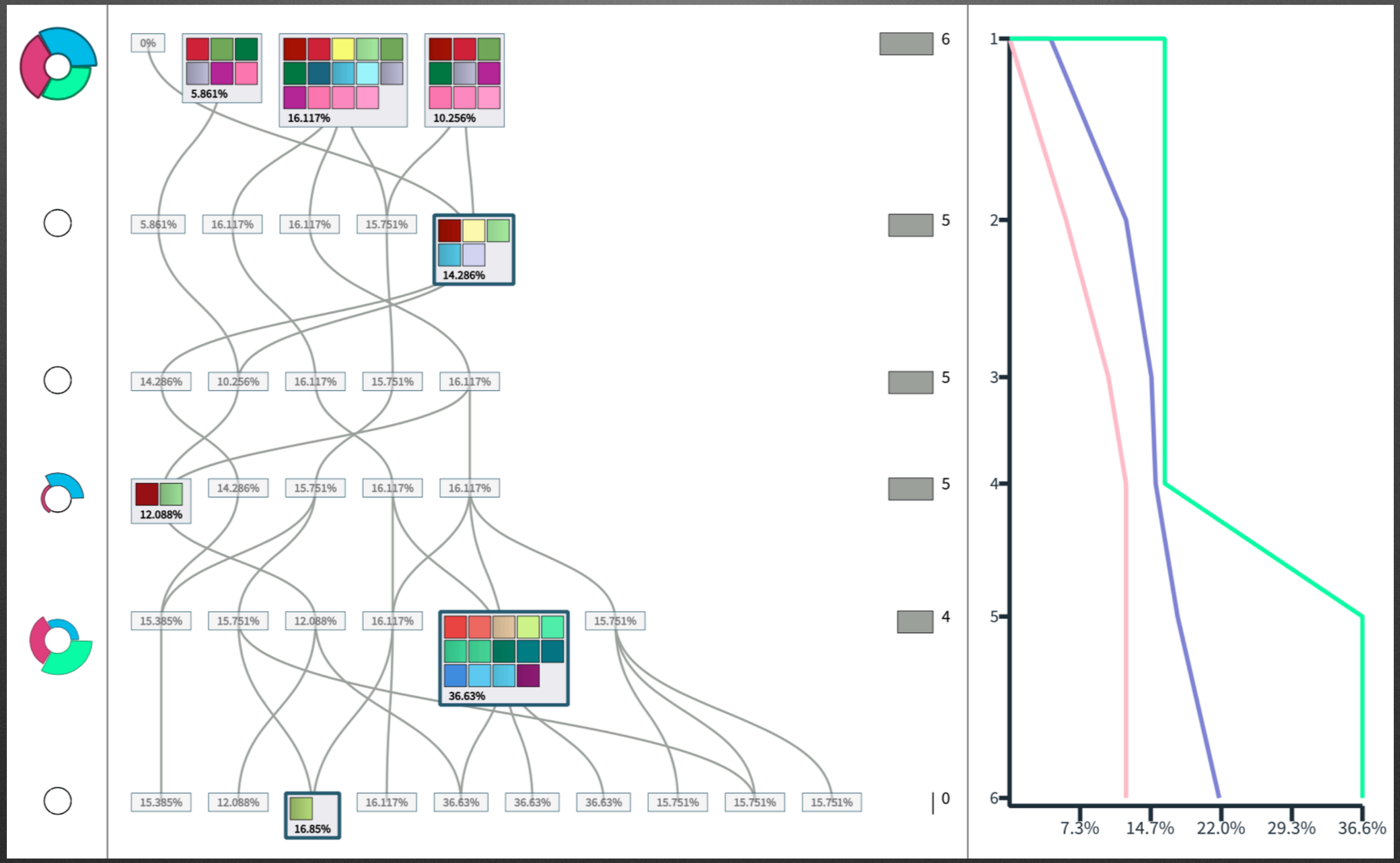
```
 ********** ****  ** **** ***** *** **** *****
 COPYRIGHT 1975 BY BILL GATES AND PAUL ALLEN
 ********** ****  ** **** ***** *** **** *****

 ORIGINALLY WRITTEN ON THE PDP-10 FROM
 FEBRUARY 9 TO  APRIL 9 1975

 BILL GATES WROTE A LOT OF STUFF.
 PAUL ALLEN WROTE A LOT OF OTHER STUFF AND FAST CODE.
 MONTE DAVIDOFF WROTE THE MATH PACKAGE (F4I.MAC).
```

https://pharoweekly.wordpress.com/2020/05/24/roassal-1-hour-xp-assembly-code-of-gwbasic/

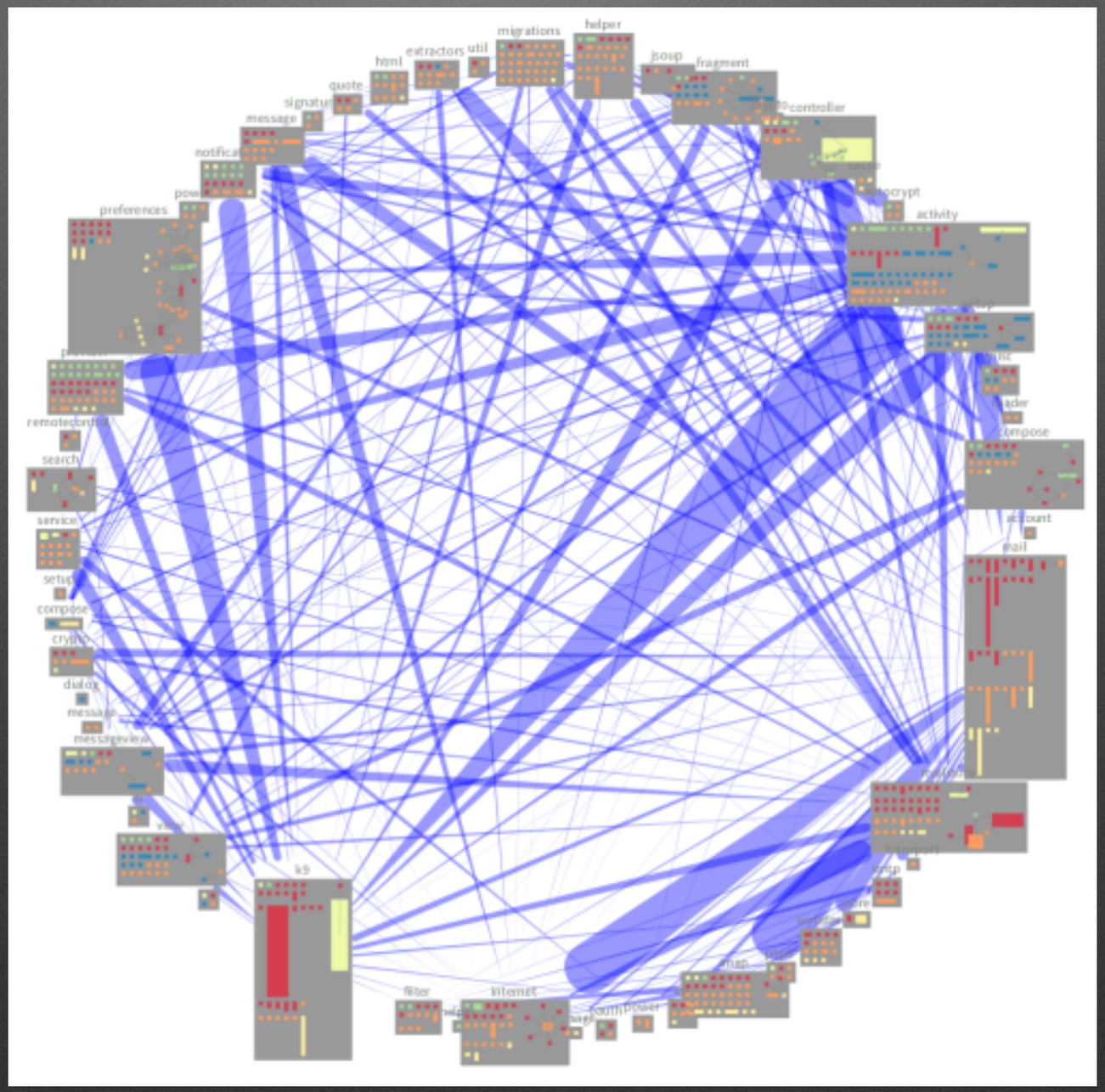




Execution of IA generating tests



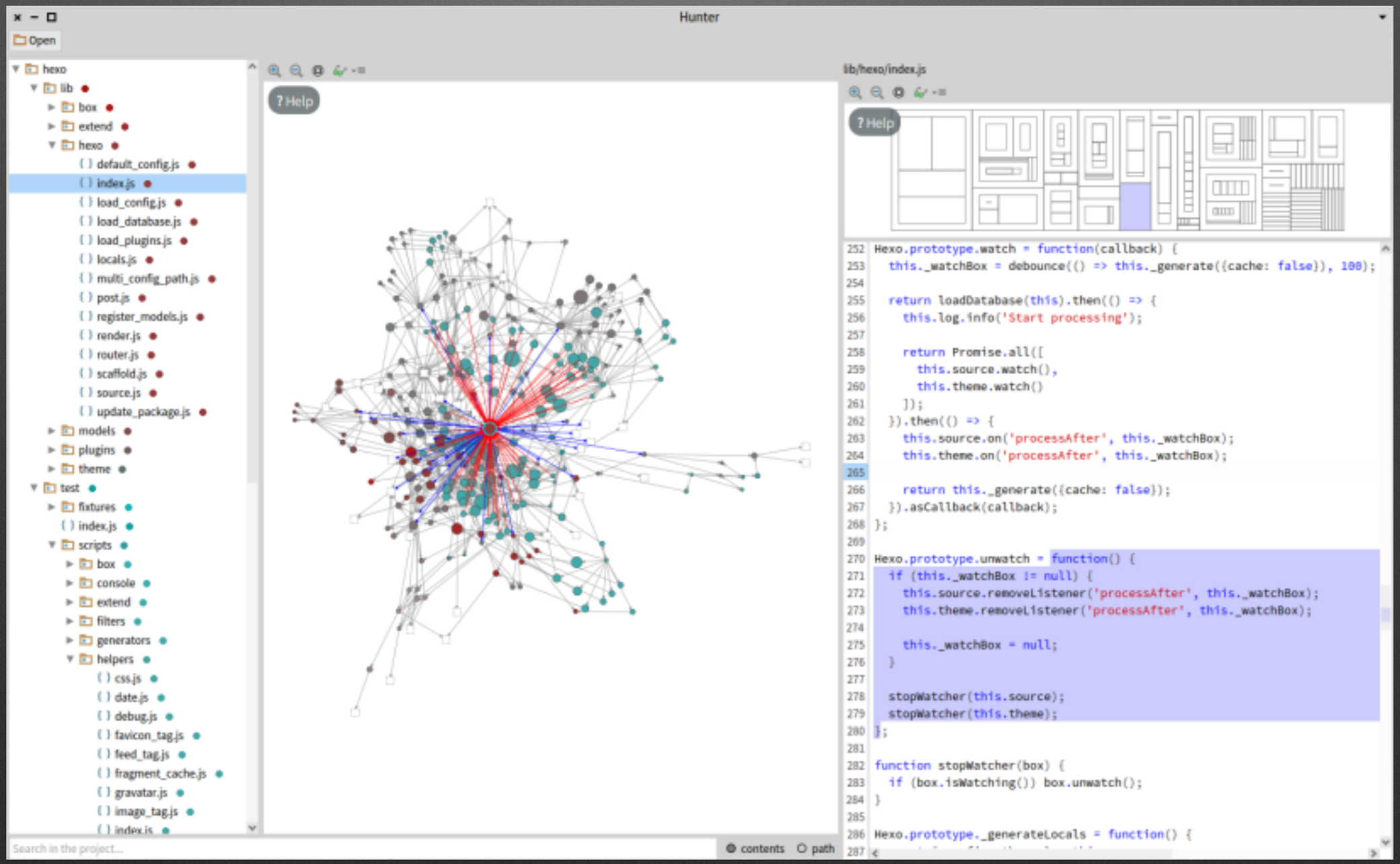
Analysis Android application



Often developers write their own tools

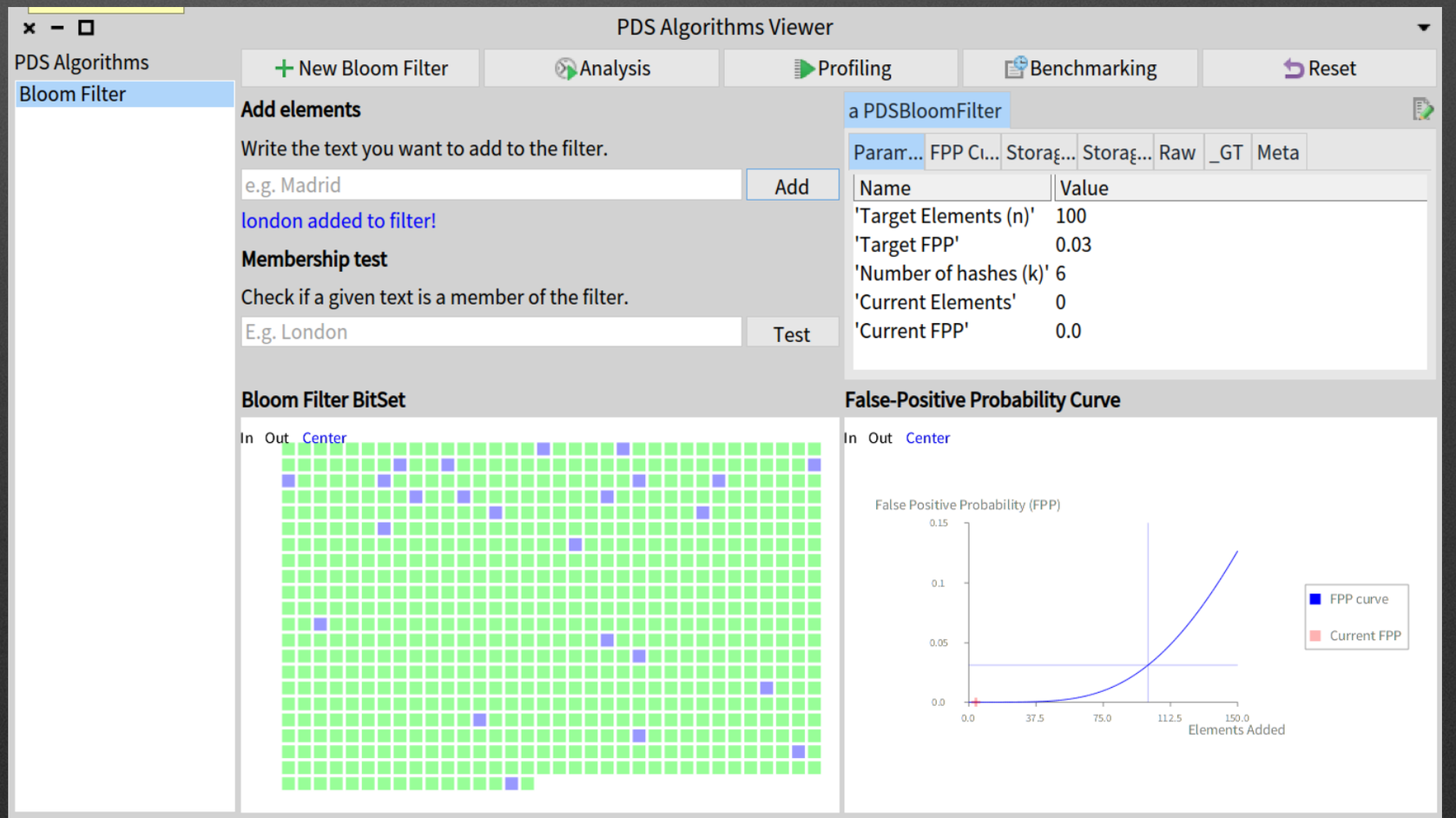
Building your own tool

- Example Javascript analysis



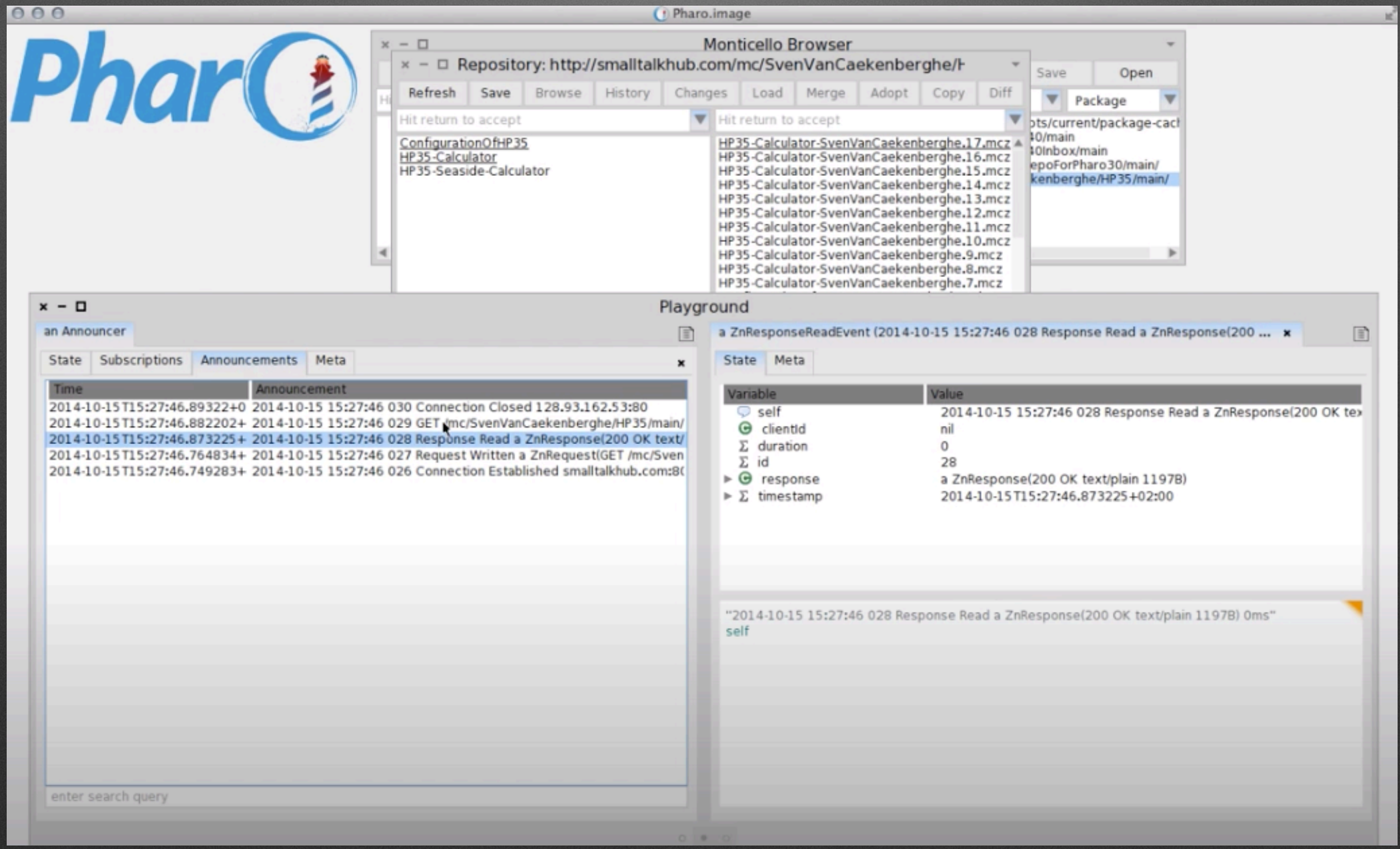
Probabilistic Data Structure

- https://github.com/osoco/PharoPDS
- Defined new data structure
- And the analysis tools



HTTP traffic analysis

- http://youtu.be/rIBbeMdFCys

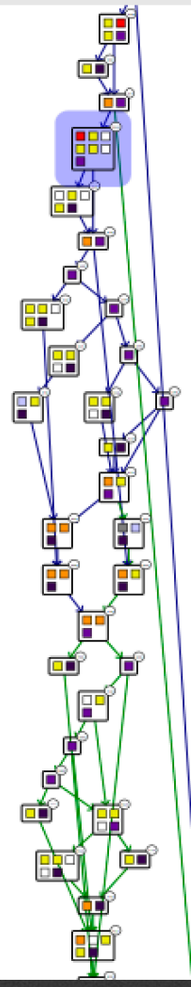


AOT compiler CFG

Inspector on a DRPrimitiveControlFlowGraph

a DRPrimitiveControlFlowGraph

CFG | Raw | Breakpoints | Meta



a DRBasicBlock (DRBasicBloc...

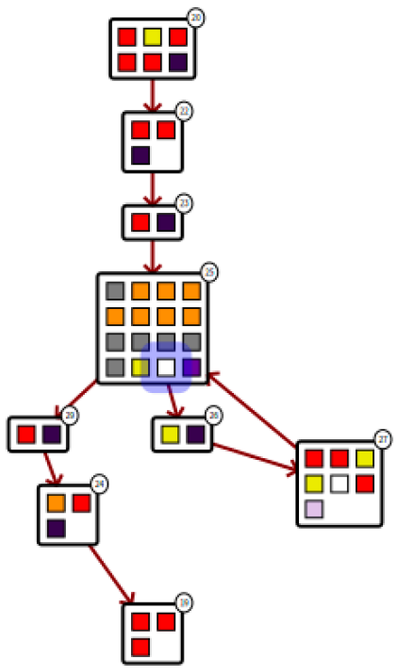
Instructions | Source code | Raw | Breakpoints | Meta

```
R4333 := Arg0Reg
R351 := RightShift R4333 3
R370 := LOAD 8 [ReceiverResultReg]
R420 := RightShift R370 24
R446 := BitAnd R420 31
R508 := LOAD 1 [ReceiverResultReg, offset=7]
Jump if R508 EqualsThan 255 -> 919 if false -> 918
```

Inspector on a DRControlFlowGraph

a DRControlFlowGraph

CFG | Raw | Breakpoints | Meta



9 blocks

44 instructions

0 paths

a DRGetConditionCode(R29 :=...

Source | Dependency Graph | Raw | Breakpoints | Meta

```
1  to: stop do: aBlock
2     "Normally compiled in-line, and therefore not overridable.
3     Evaluate aBlock for each element of the interval (self to: stop by: 1)."
4     | nextValue |
5     nextValue := self.
6     [nextValue <= stop]
7        whileTrue:
8           [aBlock value: nextValue.
9           nextValue := nextValue + 1]
```

A little binary stepper

VM Debugger

Address		ASM	Bytes
16r10000C0	➡	tst x23, #0x7	#['16rFF' '16rA'
16r10000C4		b.ne #760	#['16rC1' '16r1'
16r10000C8		mov x1, #1	#['16r21' '16r0'
16r10000CC		mov x22, x3	#['16rF6' '16r3'
16r10000D0		ands x1, x1, x22	#['16r21' '16r0'
16r10000D4		cmp x1, #0	#['16r3F' '16r0'
16r10000D8		b.eq #12	#['16r60' '16r0'
16r10000DC		mov x22, #0	#['16r16' '16r0'
16r10000E0		b.al #8	#['16r4E' '16r0'
16r10000E4		mov x22, #1	#['16r36' '16r0'
16r10000E8		cmp x1, #0	#['16r3F' '16r0'
16r10000EC		b.eq #12	#['16r60' '16r0'
16r10000F0		mov x22, #0	#['16r16' '16r0'
16r10000F4		b.al #4	#['16r2E' '16r0'
16r10000F8		cmp x22, #0	#['16rDF' '16r2
16r10000FC		b.ne #704	#['16r1' '16r16'
16r1000100		mov x22, x3	#['16rF6' '16r3'
16r1000104		asr x22, x22, #3	#['16rD6' '16rF
16r1000108		ldr x1, [x23]	#['16rE1' '16r2'
16r100010C		mov x25, x1	#['16rF9' '16r3'
16r1000110		asr x25, x25, #24	#['16r39' '16rFI
16r1000114		ands x25, x25, #	#['16r39' '16r1:
16r1000118		ldurb w19, [x23,	#['16rF3' '16r7:
16r100011C		ands x19, x19, #	#['16r73' '16r1I

lr		'16r143C000'
pc		'16r10000C0'
sp		'16r143BFC0'
fp		'16r143E000'
x28	vmStackPointer	'16r143DFF8'
x0		'16r0'
x1		'16r7FFFFFFFFFF
x2		'16r0'
x3		'16r9'
x4		'16r0'
x5		'16r0'
x6		'16r0'
x7		'16r0'
x8		'16r0'
x9		'16r0'
x10		'16r0'
x11		'16r0'
x12		'16r0'
x16		'16r143BFF8'
x19		'16r0'
x20		'16r0'
x21		'16r0'
x22	classRegister	'16r0'
x23	receiverRegister	'16r10B0B60'

SP	16r143DFF8	16r123B000
FP	16r143E000	16r0
	16r143E008	16r0
	16r143E010	16r0
	16r143E018	16r0
	16r143E020	16r0
	16r143E028	16r0
	16r143E030	16r0
	16r143E038	16r0
	16r143E040	16r0
	16r143E048	16r0
	16r143E050	16r0
	16r143E058	16r0
	16r143E060	16r0
	16r143E068	16r0
	16r143E070	16r0
	16r143E078	16r0
	16r143E080	16r0
	16r143E088	16r0
	16r143E090	16r0
	16r143E098	16r0
	16r143E0A0	16r0
	16r143E0A8	16r0
	16r143E0B0	16r0

Jump to

Step

Disassemble at PC

Moments of grace...

I want my halt to only stop when called from THAT test called testMe!

```
mycode
    self haltIf: #testMe
    ...
```

Use stack reification
Walk it
Halt if needed
(in 5 lines)

```
haltIf: aSelector
	| cntxt |
	cntxt := thisContext.
	[ cntxt isNil ] whileFalse: [
		cntxt selector = aSelector
		  ifTrue: [ self halt ].
		cntxt := cntxt sender ]
```

Pharo is

research friendly

International Research Groups

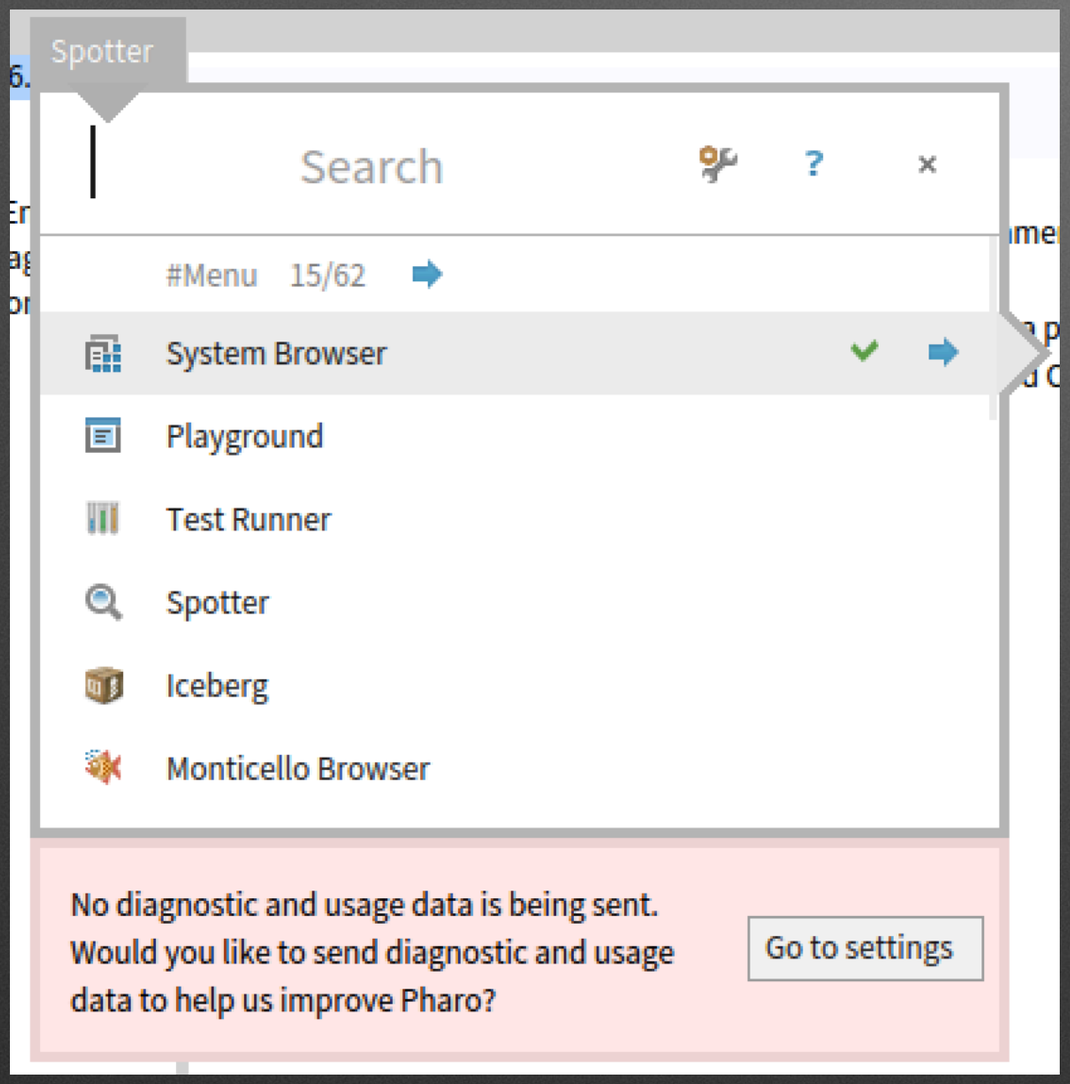
Lafhis (AR)
SCG (CH)
CAR (FR)
RMOD (FR)
Ummisco (IRD)
Reveal (CH)
Lysic (FR)
ENSTA-Bretagne (FR)
CEA-List (FR)
Ryerson (CAN)
OC (FR)
CCMI-FIT (CZ)

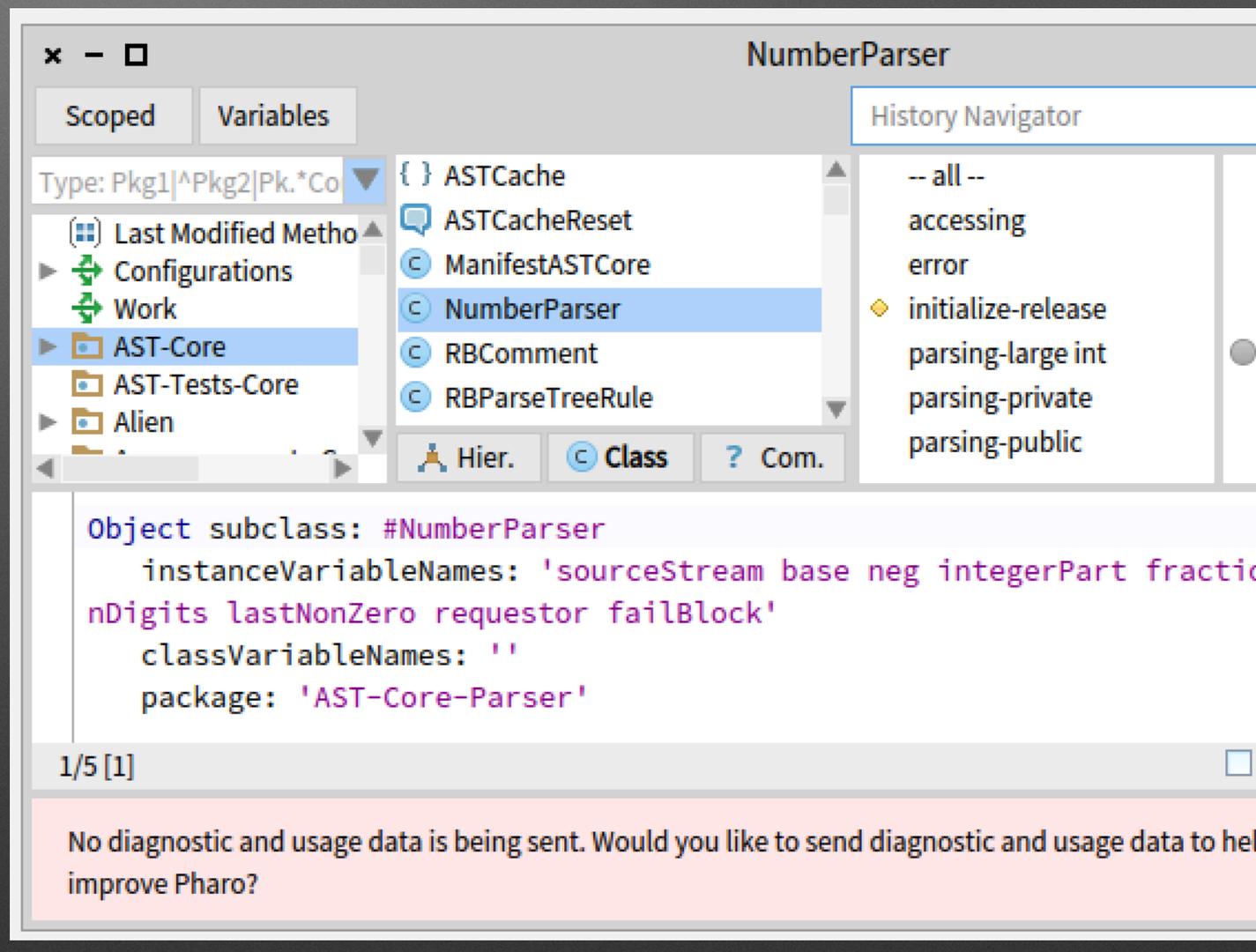
ASERG (BR)
Pleiad (CL)
Macau (UNO)
Cirad (FR)
USTH (Vietnam)
Soft-Qual (Serbia)
Uni. Quilmes (AR)
ENIT (FR)
CS (Bo)
Maroua (CAM)
ETS (CAN)

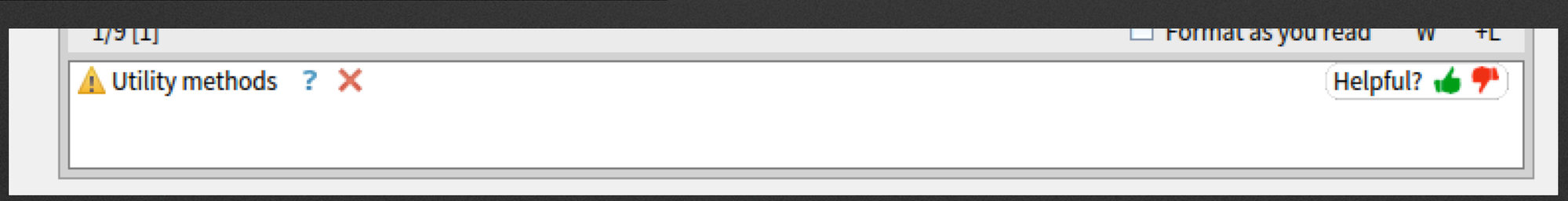
We can be your guinea pigs,...

(well kind of.... we have real users)

Was actively supported research e.g., SCG from Uni. Bern







Best Paper Award @ ICPC'19

Kubelka, Bergel, Robbes,
"Live Programming and Software Evolution: Questions during a Programming Change Task"

Map reduce debugging

by M. Marra and Prof. E. Gonzales Boix from Vrije Universiteit Brussel

Empowering is the right word

- Pharo is an energizing and creative environment
- Moldable tools are powerful
- Tried to share my feeling
- But “The idea of experience does not replace experience.” Alain

Fun with Us

- 1 hour from Paris, 1:30 from London, 35 min from Brussels
- Internships 3 to 6 months (right now we have 8 interns)
- Google Summer of Code
- PhDs / co supervision (e.g., B. Sarenac)
- Engineer position
- Visitors (Magadascar, Chile, Montreal, …)