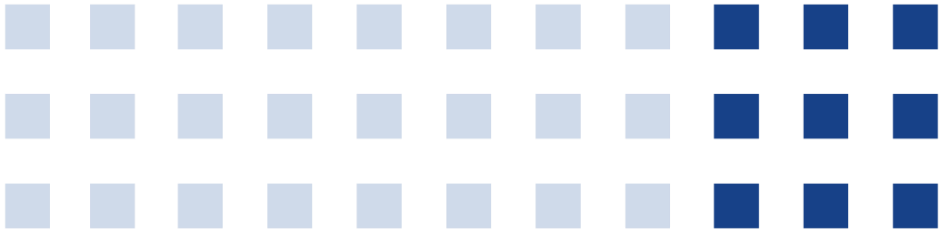


CubeServ®

Legacy Code

Test Seams

Automated Sketches

Pharo Smalltalk

Referent: **Rainer Winkler**

Date: **25.02.2107**

Martin Fowler
@martinfowler

Folge ich

Good software architecture advice is always worth considering, but it should never be always carried out

Original (Englisch) übersetzen

RETWEETS	GEFÄLLT
264	182

12:07 - 5. Juni 2015

8 264 182

Coding:
constructing a machine

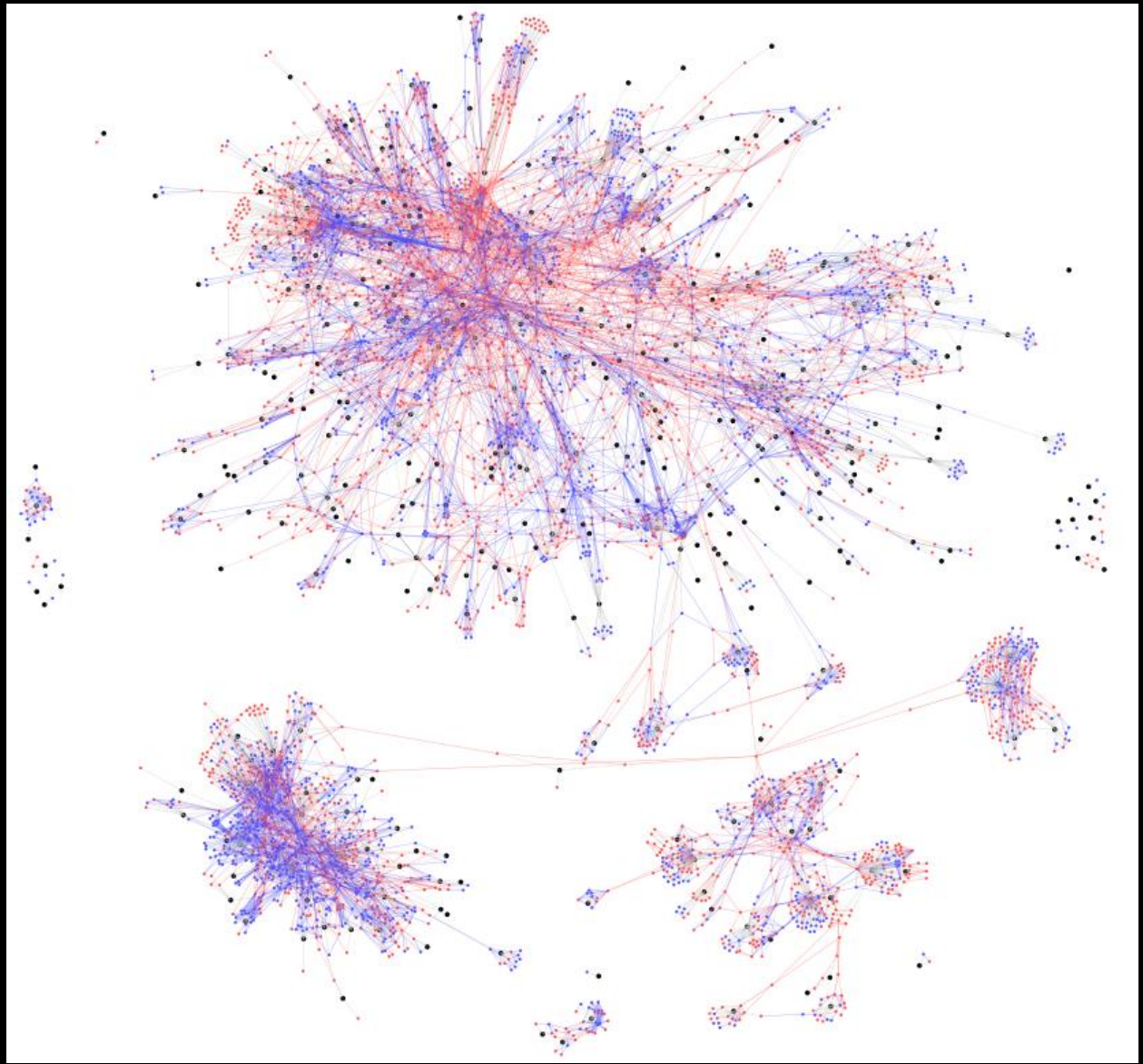
There are many tips to construct well

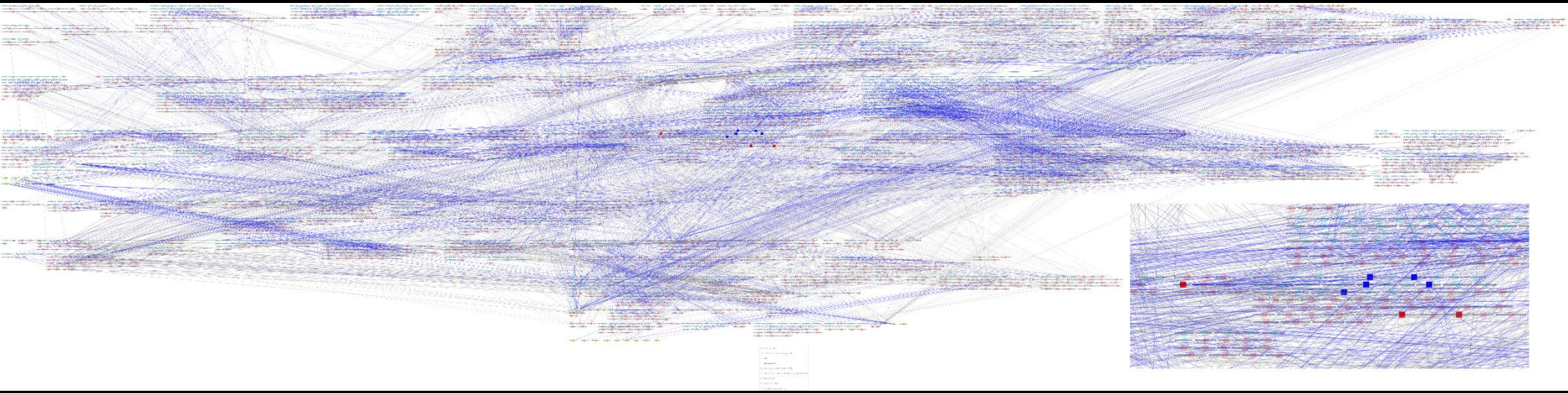
There is no simple way to construct a good machine

A machine has to fullfill many requirements that contradict each other

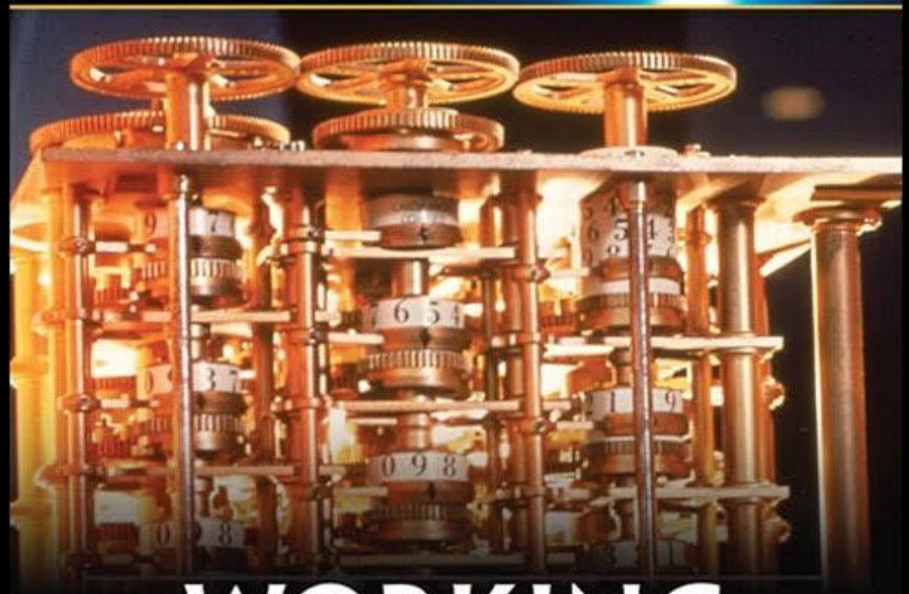
Low weight <-> robust

Coding becomes
complex quite soon





Robert C. Martin Series



WORKING EFFECTIVELY WITH LEGACY CODE

Michael C. Feathers

legacy code

is

code without tests

UNIT Tests
Automatic tests

Know your bugs before your customer finds them

Do not use much time with finding errors

But there are dependencies?

SELECT
INSERT
Calls to Code that changes ...

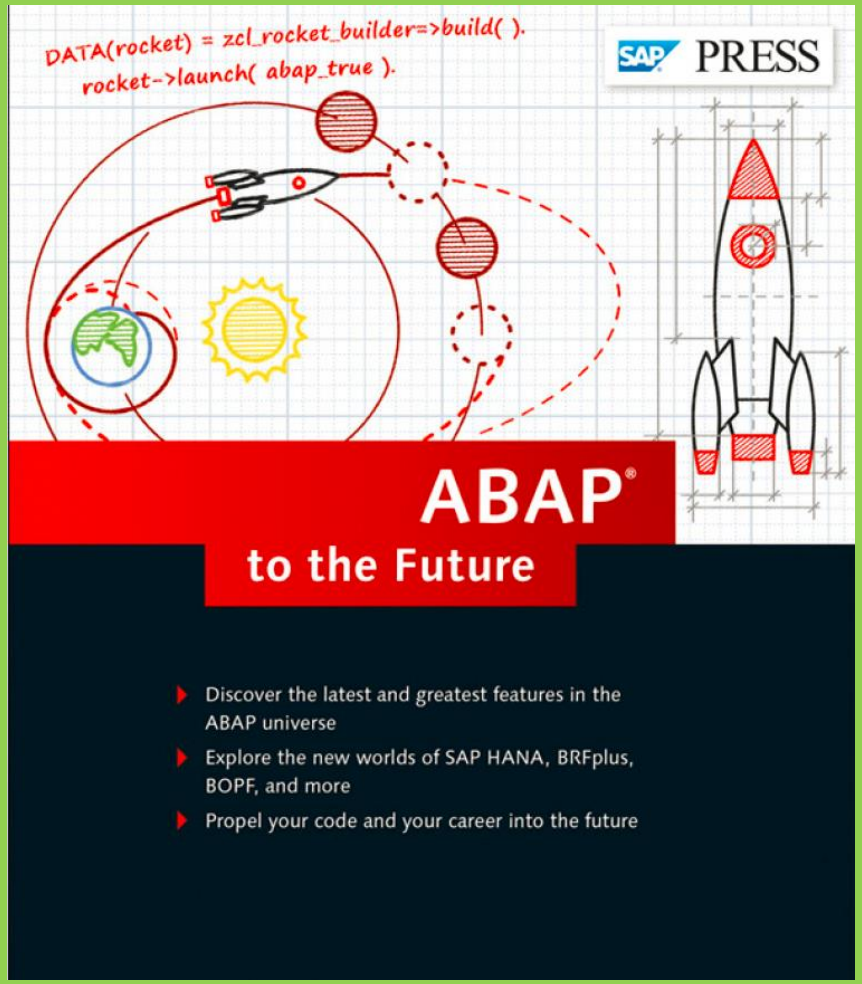
Common advice

Use interfaces

But using interfaces is often

not easy

Paul Hardy


DATA(rocket) = zcl_rocket_builder=>build( ).
rocket->launch(abap_true).
SAP PRESS
ABAP®
to the Future
Discover the latest and greatest features in the ABAP universe
Explore the new worlds of SAP HANA, BRFplus, BOPF, and more
Propel your code and your career into the future

Use instance methods
and
inherit mock classes

Simpler

Difficult and
dangerous to add
to existing code
that has no tests

Even simpler

TEST-SEAMS in ABAP 7.50

Works on static methods

Needs no additional classes or methods

Add to existing code without risk

Read scn blogs

ABAP News for Release 7.50 – Test Seams and Test Injections

October 23, 2015 | 1,011 Views |

Horst Keller

more by this author

ABAP Development

Working effectively with ABAP legacy code – ABAP Test Seams are your friend

February 6, 2016 | 290 Views |

Klaus Ziegler

more by this author

ABAP Testing and Analysis

Replace accesses to persistent data

Test Seam

```
TEST-SEAM read_content_seam.
  SELECT *
         FROM sflight
         WHERE  carrid IN @carrid_range AND
                fldate EQ @sy-datum
      INTO TABLE @flights.
END-TEST-SEAM.
```

Injection

```
TEST-INJECTION read_content_seam.
  flights =
    VALUE #( ( carrid = 'LHA'
               connid = 100 )
             ( carrid = 'AFR'
              connid = 900 ) ).
END-TEST-INJECTION.
```

Test Seam	Injection

TEST-SEAM store_content_seam.
  MODIFY sflight
  FROM TABLE @new_flights.
END-TEST-SEAM.
	TEST-INJECTION store_content_seam.
	  cl_abap_unit_assert=>assert_equals(
	  act = new_flights
	  exp = global_buffer=>exp_flights).
	END-TEST-INJECTION.

Replace
authority checks

Test Seam	Injection

TEST-SEAM authorization_seam.
  AUTHORITY-CHECK OBJECT 'S_CTS_ADMI'
  ID 'CTS_ADMFCT' FIELD 'TABL'.
END-TEST-SEAM.
 	
IF sy-subrc = 0.
  is_authorized = abap_true.
ENDIF.
	TEST-INJECTION authorization_seam.
	  sy-subrc = 0.
	END-TEST-INJECTION.

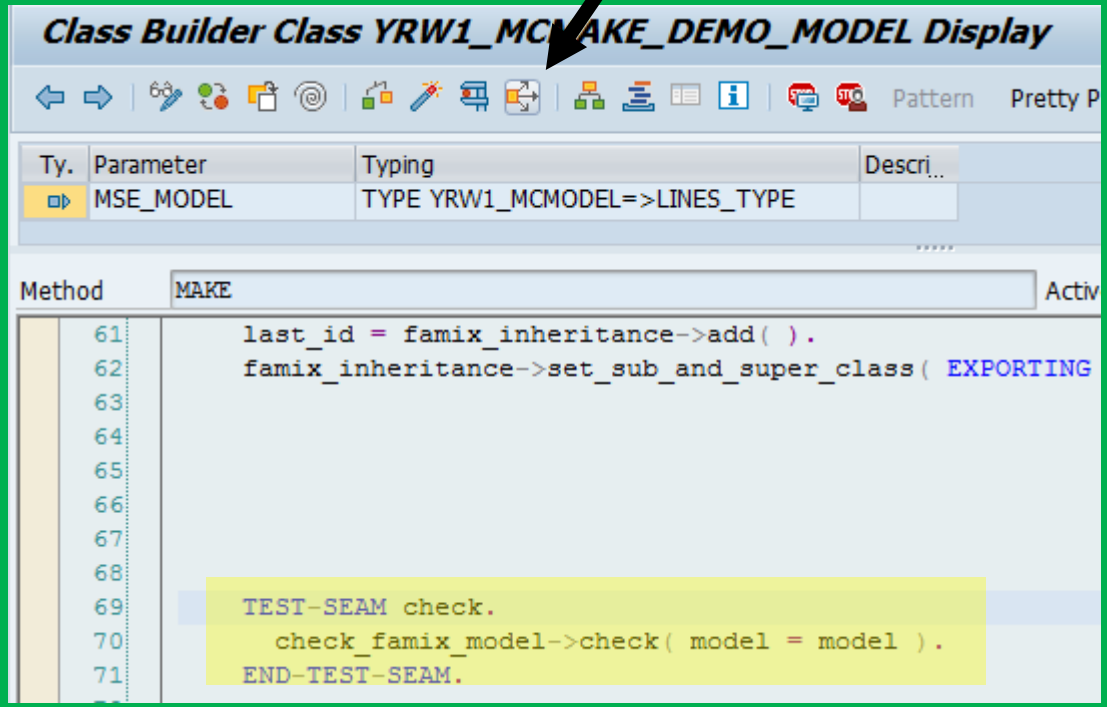
Inject
test doubles

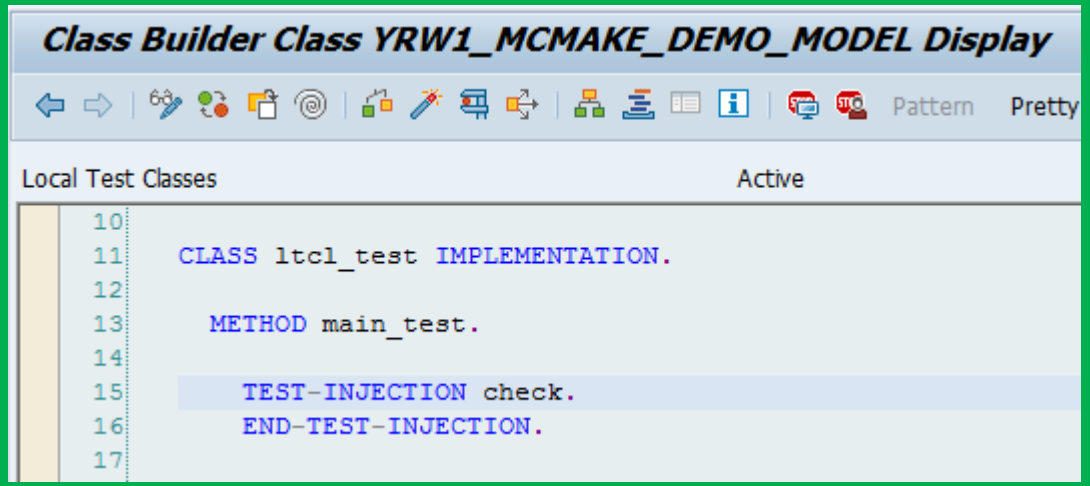
Test Seam	Injection

TEST-SEAM instantiation_seam.
  me->oref = NEW #( ).
END-TEST-SEAM.
	TEST-INJECTION instantiation_seam.
	  me->oref = NEW dummy_class( ).
	END-TEST-INJECTION.

Sometimes small changes needed so that TEST-SEAMS can be added

Where Used is working





A simple way to switch coding off during test

TEST-SEAMS for new coding?

Only way to add Unit tests to code that accesses the outside world

Simple

Better than no Unit Tests

I use TEST-SEAMS in
new coding
when it is the best
option

Less coding – short! ☺

Ask colleagues for help –
Starting with Unit Tests is
not easy

Feeling more save

Understanding how a method is to be used

Tools – Automatic diagram generation

What depends on this?

What is it influencing?

Where am I?

What is the big picture?

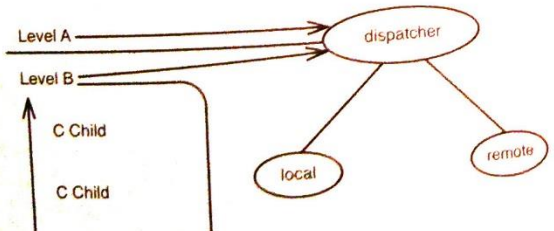
Notes/Sketching

When reading through code gets confusing, it pays to start drawing pictures and making notes. Write down the name of the last important thing that you saw, and then write down the name of the next one. If you see a relationship between them, draw a line. These sketches don't have to be full-blown UML diagrams or function call graphs using some special notation—although, if things get more confusing, you might want to get more formal or neater to organize your thoughts. Sketching things out often helps us see things in a different way. It's also a great way of maintaining our mental state when we are trying to understand something particularly complex.

Figure 16.1 is a re-creation of a sketch that I drew with another programmer the other day as we were browsing code. We drew it on the back of a memo (the names in the sketch have been changed to protect the innocent).

The sketch is not very intelligible now, but it was fine for our conversation. We learned a bit and established an approach for our work.

Doesn't everyone do this? Well, yes and no. Few people do it frequently. I suspect that the reason is because there really isn't any guidance for this sort of thing, and it's tempting to think that every time we put pen to paper, we should be writing a snippet of code or using UML syntax. UML is fine, but so are blobs and lines and shapes that would be indecipherable to anyone who wasn't there when we drew them. The precision doesn't have to be on paper. The paper is just a tool to make conversation go easier and help us remember the concepts we're discussing and learning.



The really great thing about sketching parts of a design as you are trying to understand them is that it is informal and infectious. If you find this technique useful, you don't have to push for your team to make it part of its process. All you have to do is this: Wait until you are working with someone trying to understand some code, and then make a little sketch of what you are looking at as you try to explain it. If your partner is really engaged in learning that part of the system too, he or she will look at the sketch and go back and forth with you as you figure out the code.

When you start to do local sketches of a system, often you are tempted to take some time to understand the big picture. Take a look at Chapter 17, *My Application Has No Structure*, for a set of techniques that make it easier to understand and tend a large code base.

Listing Markup

Sketching isn't the only thing that aids understanding. Another technique that I often use is *listing markup*. It is particularly useful with very long methods. The idea is simple and nearly everyone has done it at some time or another, but, frankly, I think it is underused.

The way to mark up a listing depends on what you want to understand. The first step is to print the code that you want to work with. After you have, you can use *listing markup* as you try to do any of the following activities.

Separating Responsibilities

If you want to separate responsibilities, use a marker to group things. If several things belong together, put a special symbol next to each of them so that you can identify them. Use several colors, if you can.

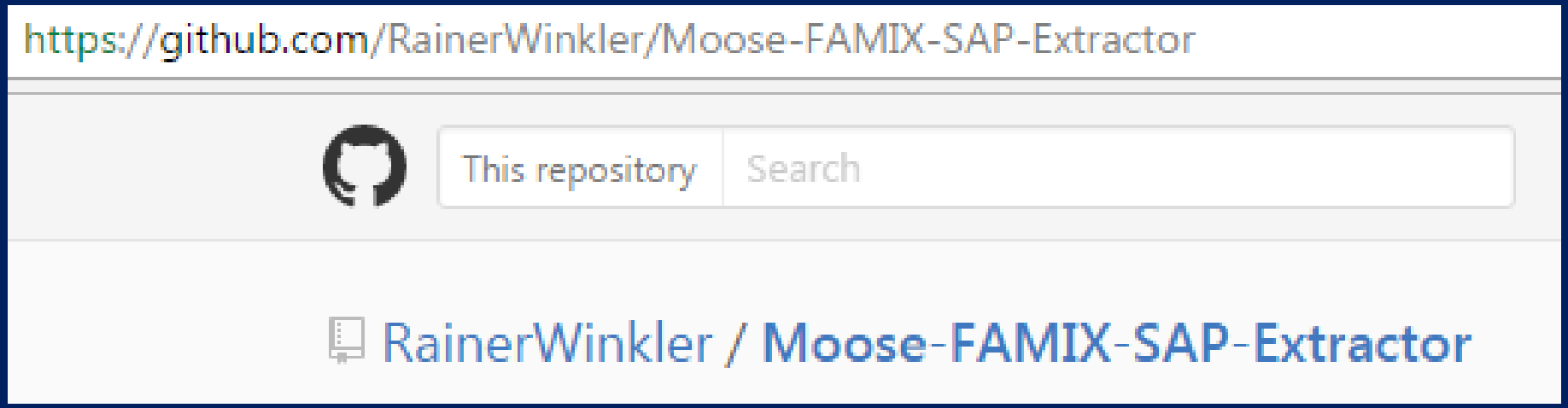
Understanding Method Structure

If you want to understand a large method, line up blocks. Often indentation in long methods can make them impossible to read. You can line them up by drawing lines from the beginnings of blocks to the ends, or by commenting the

https://youtu.be/0jLN-2AVIvo

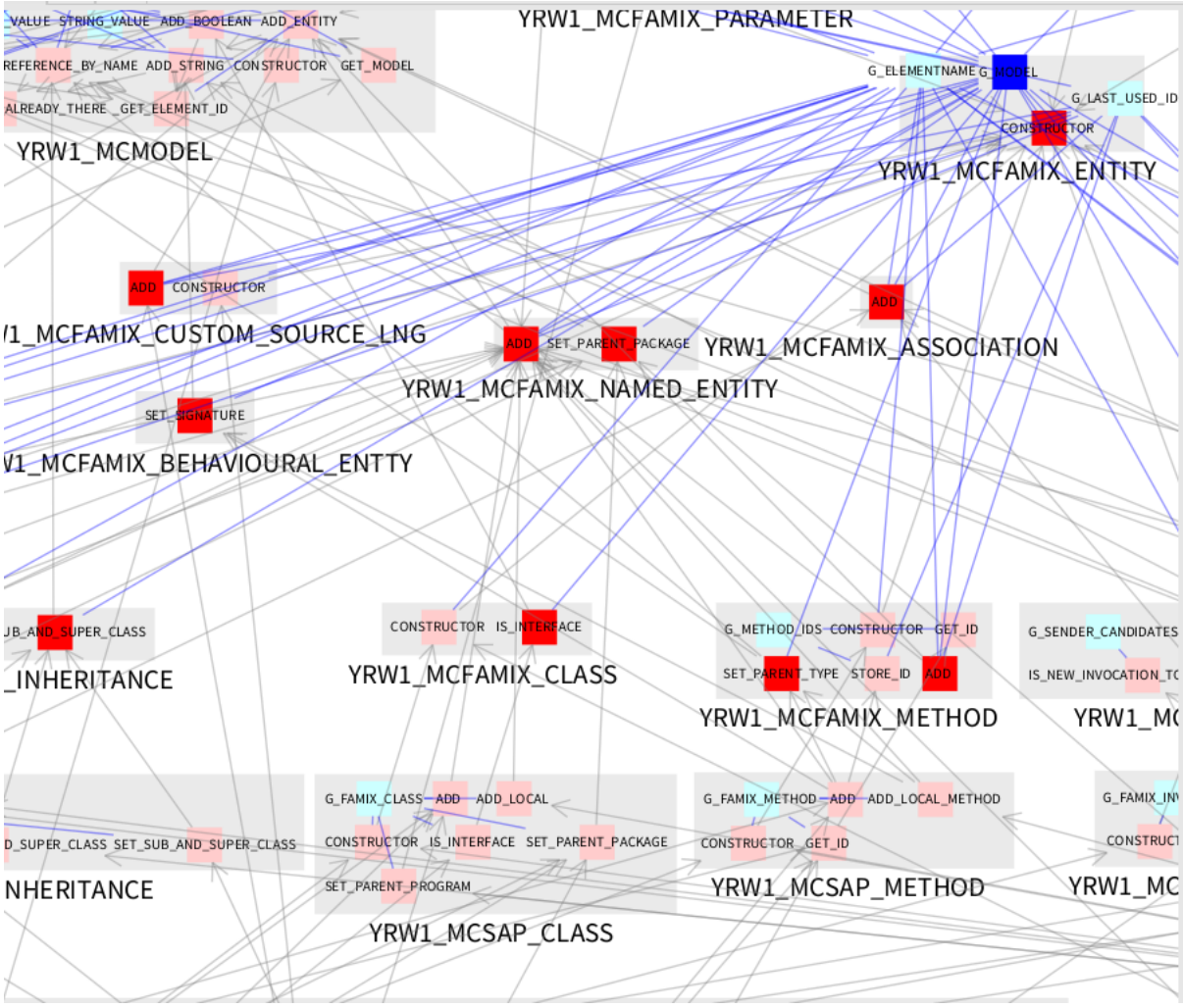
Open Source

Part 1 - Extractor



Open Source

Part 2 – Improved visualization



Variable	Value
self	a RWDiagramElement YRW1_MCFAMIX_ENTITY.G_MODEL (Attribute)
▶ attributeList	a RWDiagramAttributeList
▼ famixElement	YRW1_MCFAMIX_ENTITY.G_MODEL (Attribute)
self	YRW1_MCFAMIX_ENTITY.G_MODEL (Attribute)
▶ annotationInstances	a FMNullMultivalueLink [0 items] ()
▶ comments	a FMNullMultivalueLink [0 items] ()
▶ declaredSourceLanguage	nil
▶ declaredType	nil
▶ hasClassScope	false
▼ incomingAccesses	a FMMultivalueLink [22 items] (a FAMIXAccess #noname (Access) a FAMIXAcc
self	a FMMultivalueLink [22 items] (a FAMIXAccess #noname (Access) a FAMI
▶ opposite	#variable:
▶ owner	YRW1_MCFAMIX_ENTITY.G_MODEL (Attribute)
▶ values	an OrderedCollection [22 items] (a FAMIXAccess #noname (Access) a FAM
▶ isStub	false
▶ modifiers	a Set [0 items] ()
▶ mooseID	2406
▶ name	'G_MODEL'

```
"a RWDiagramElement YRW1_MCFAMIX_ENTITY.G_MODEL (Attribute)"
RWDiagram suppressWithChildren: true to: self.
RWDiagram highlightIncludingMe: true usedByLayers: 1 usingLayers: 1 to: self.
```

Why Pharo?
Why Smalltalk?

http://modeling-languages.com/openponk-metamodeling-platform/

OpenPonk (meta)modeling platform

By **Peter Uhnak** 16/02/2017 | 8:08 Posted in (meta)modeling, tools 8

Peter Uhnak

Pharo makes for a great platform for rapid prototyping
thanks to the combination of a
dead-simple language (the grandparent of OO languages),
powerful metaprogramming capabilities and
short feedback loops;
hey, you can even write your entire application in a debugger

Peter Uhnak

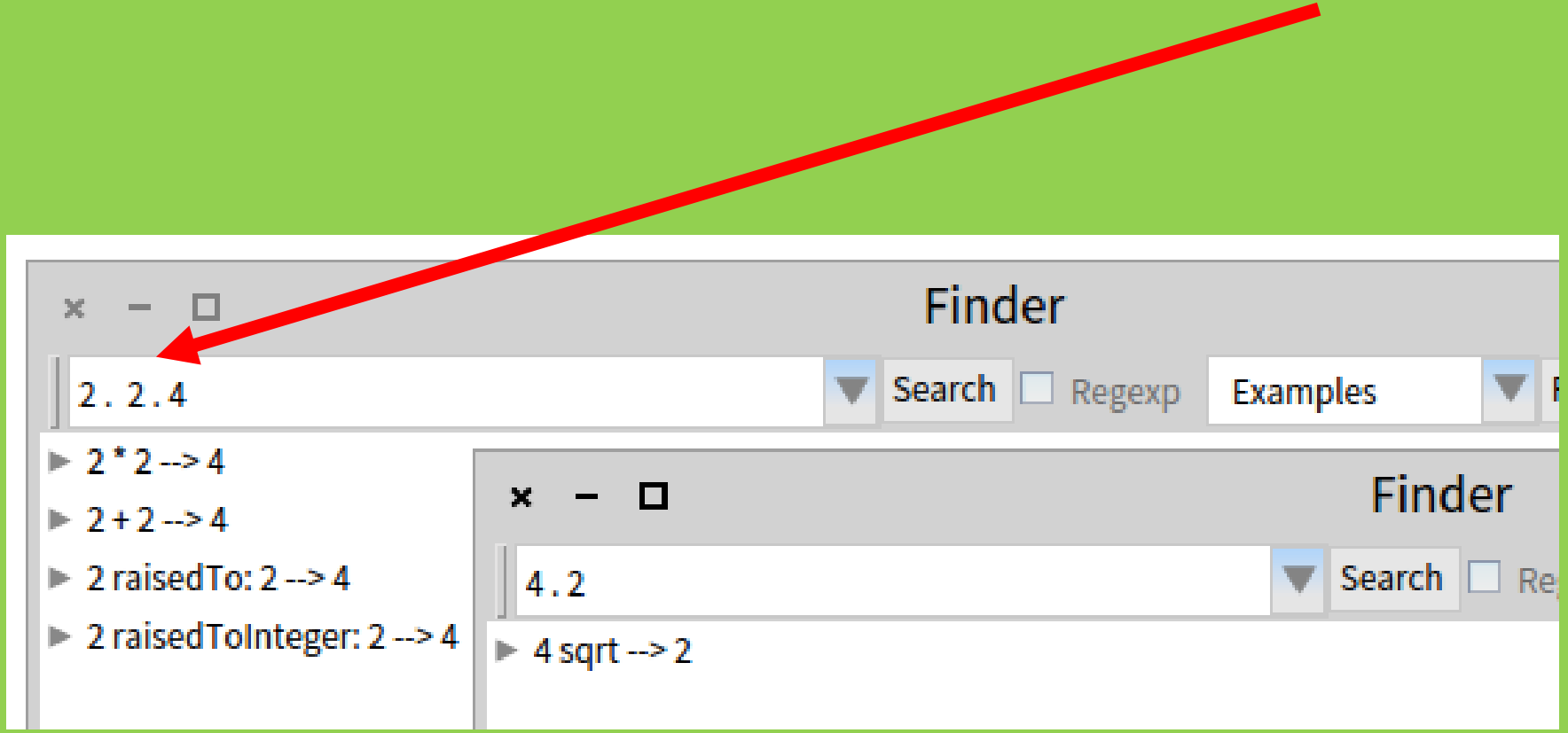
What about other tools?

... layers of abstractions to facilitate all the generic tools and needs, and overall architecture we simply didn't see it as a good fit for our long-term needs.

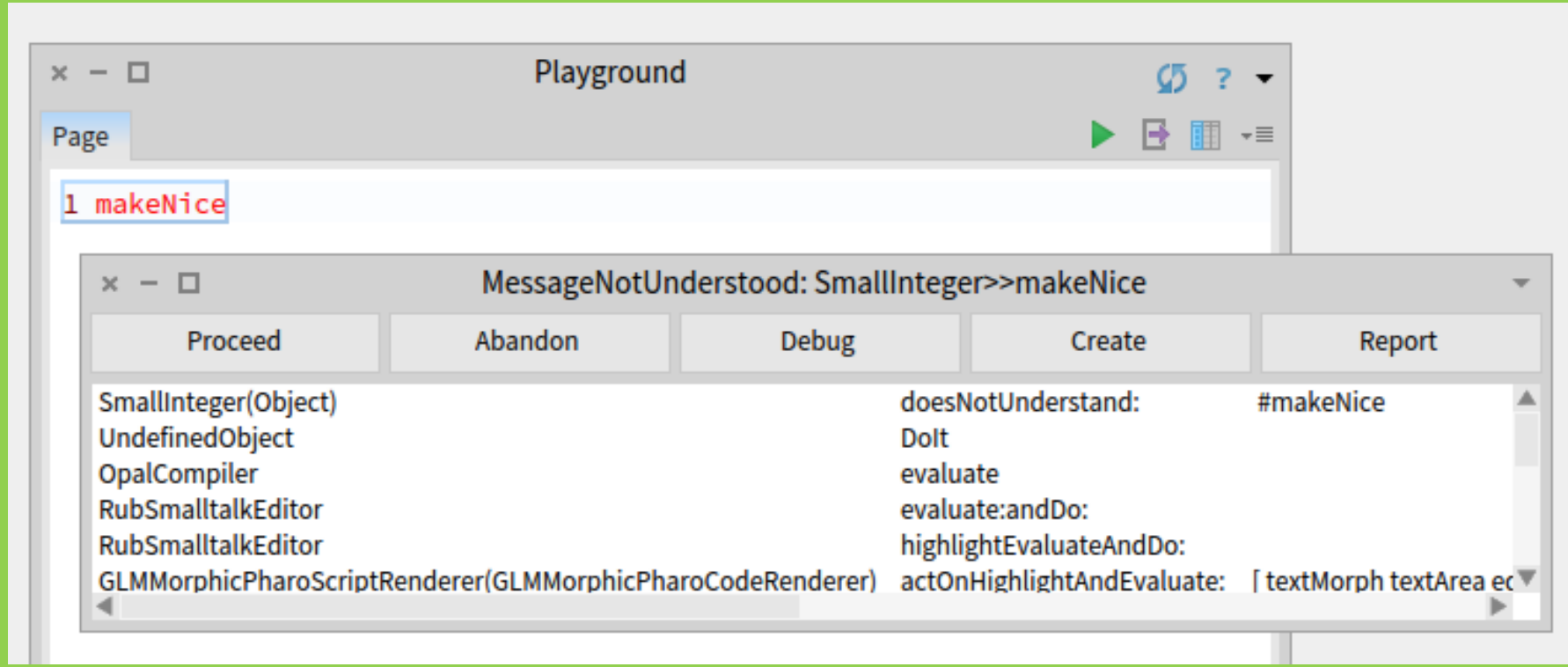
Information hiding, user hand-holding, and over-engineering are certainly valuable, if you seek a robust solution for end-users.

But for a poweruser any non-essential complexity is problematic, or prohibits different use-cases (in part this also relates to Java).

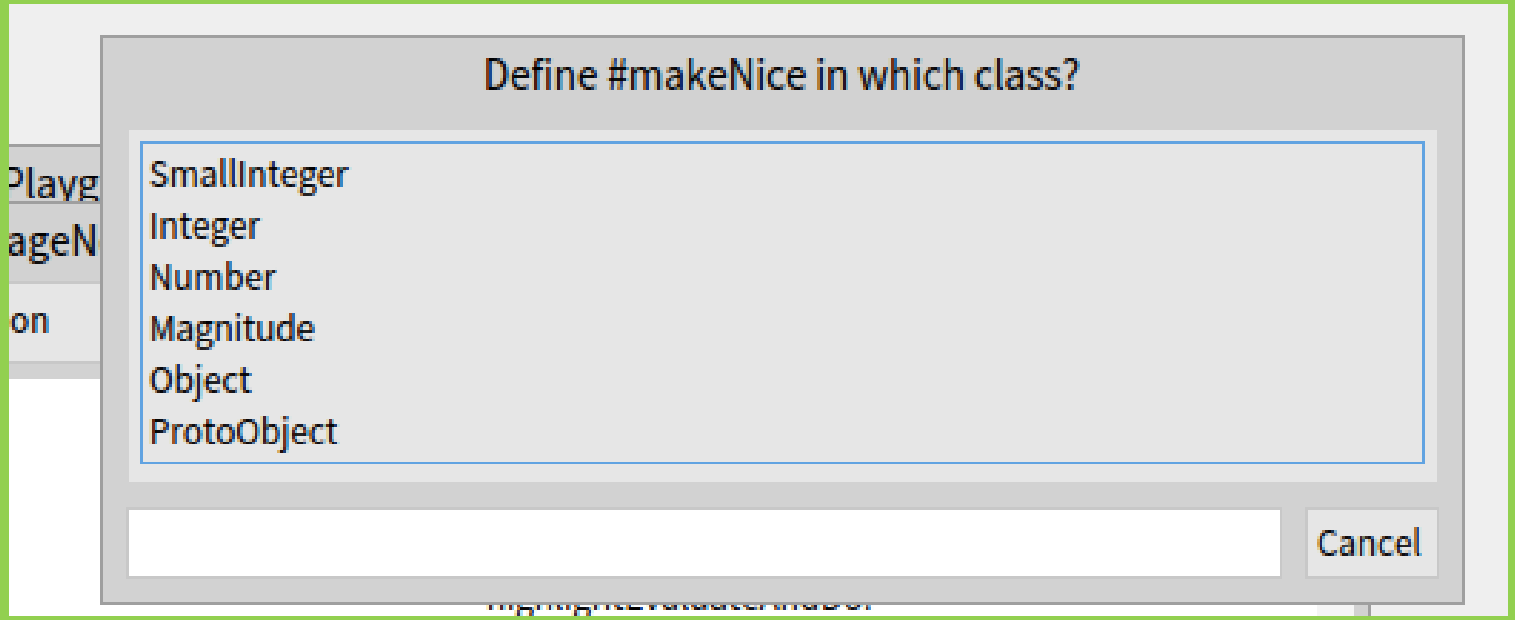
Search for statements by example



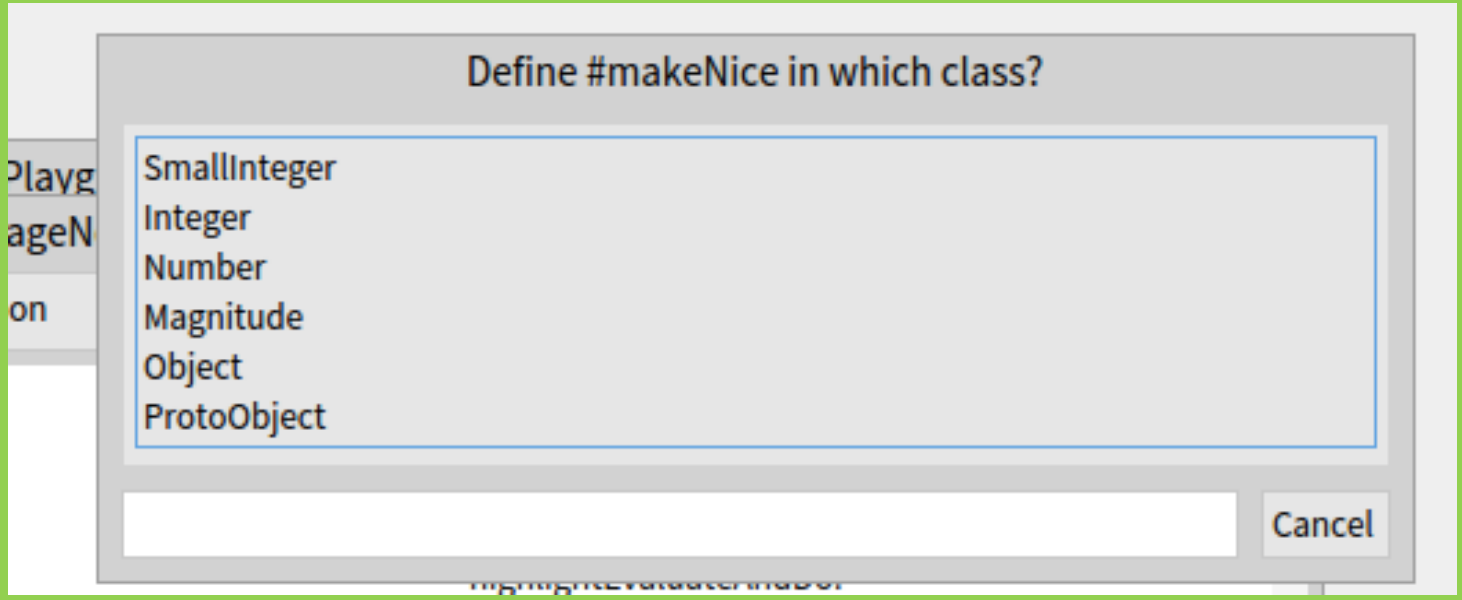
Write code in debugger



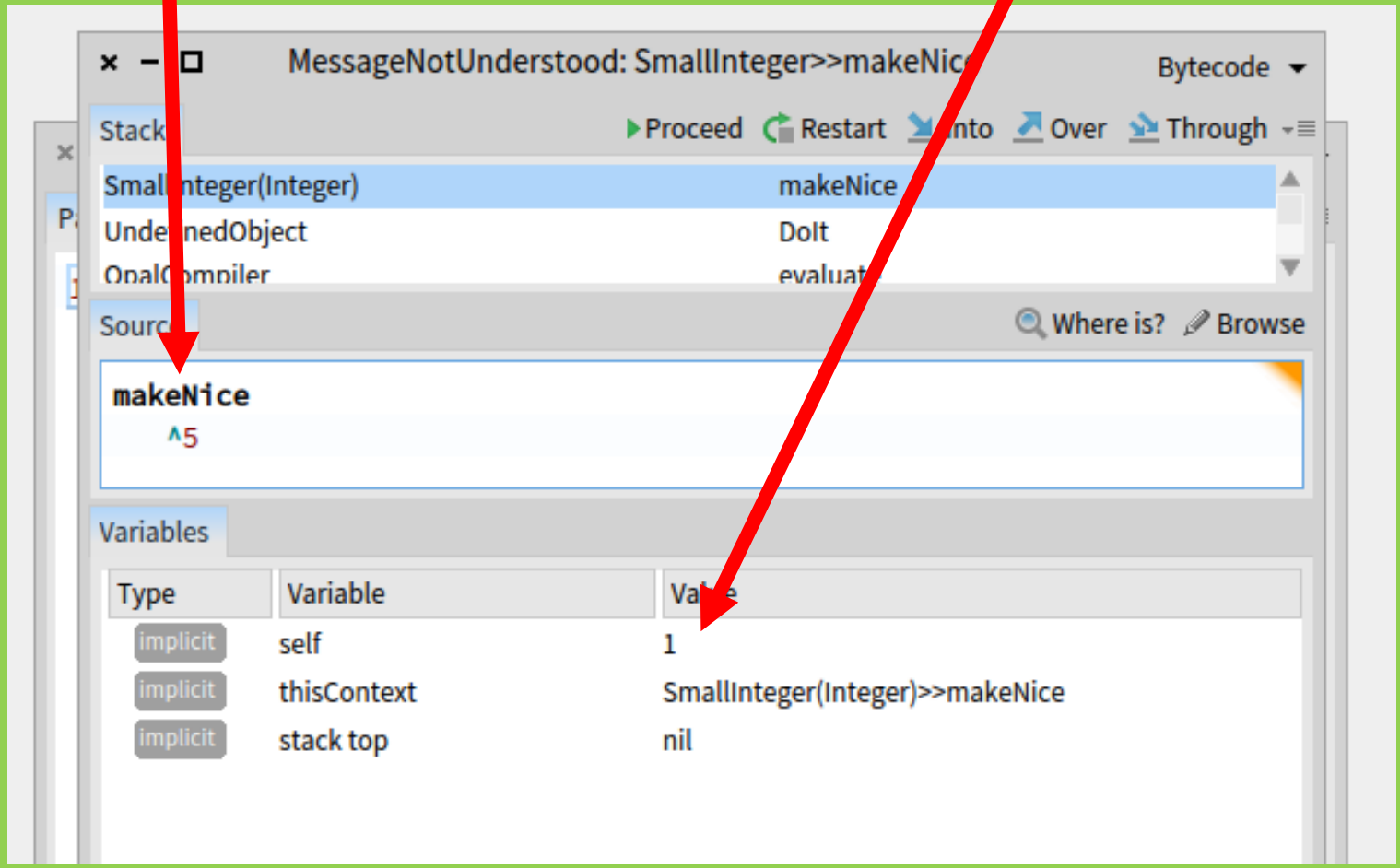
Write code in debugger



Write code in debugger

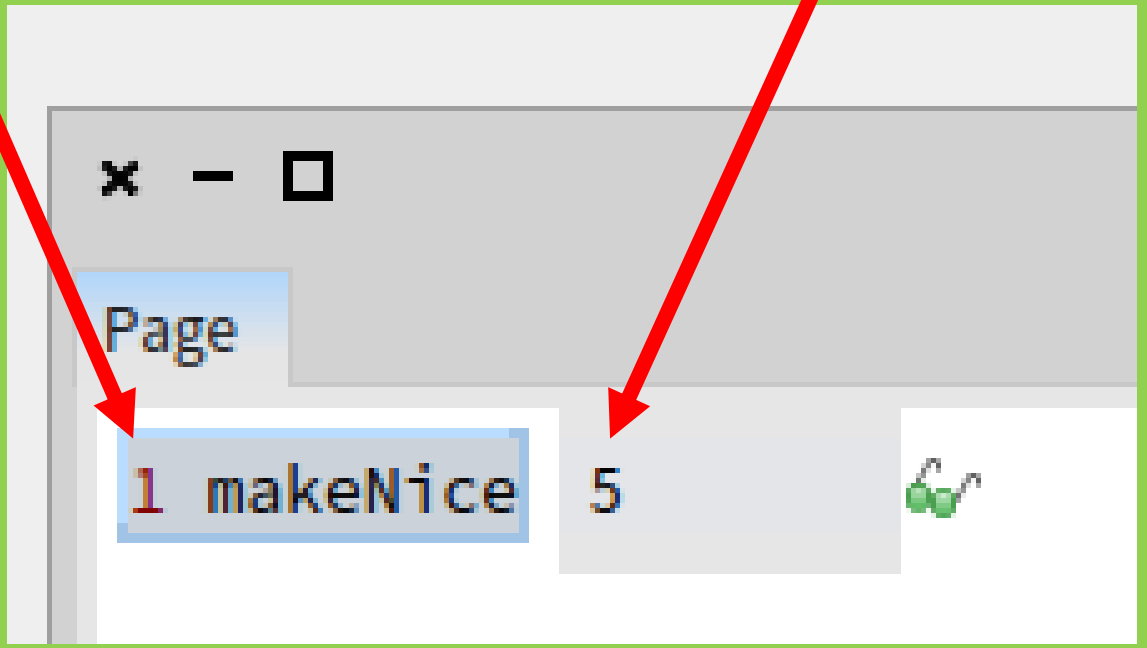


Write new code while inspecting variables

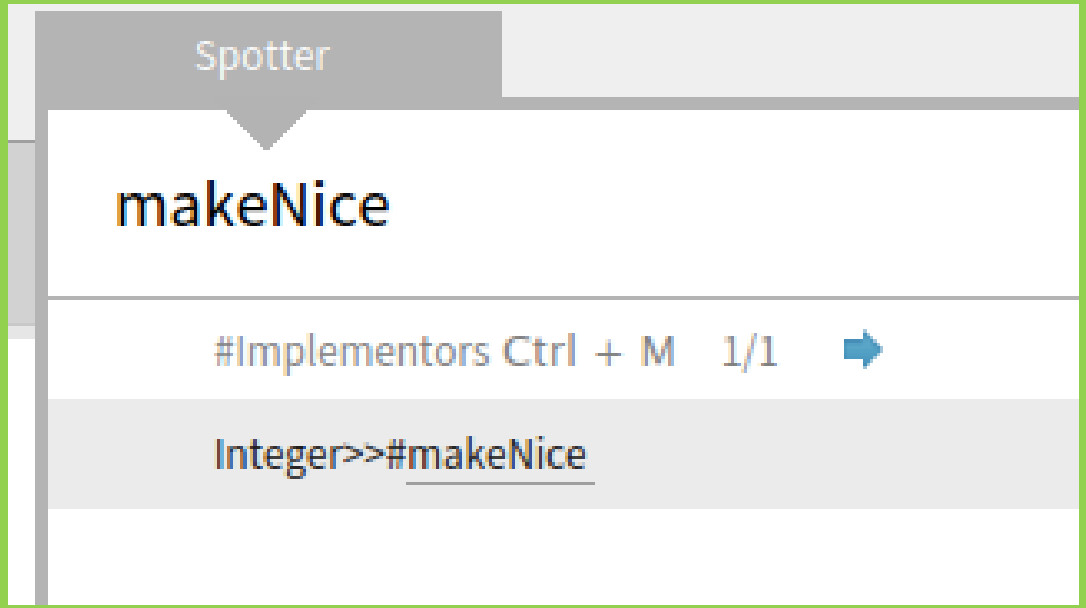


Send message „makeNice“ to 1

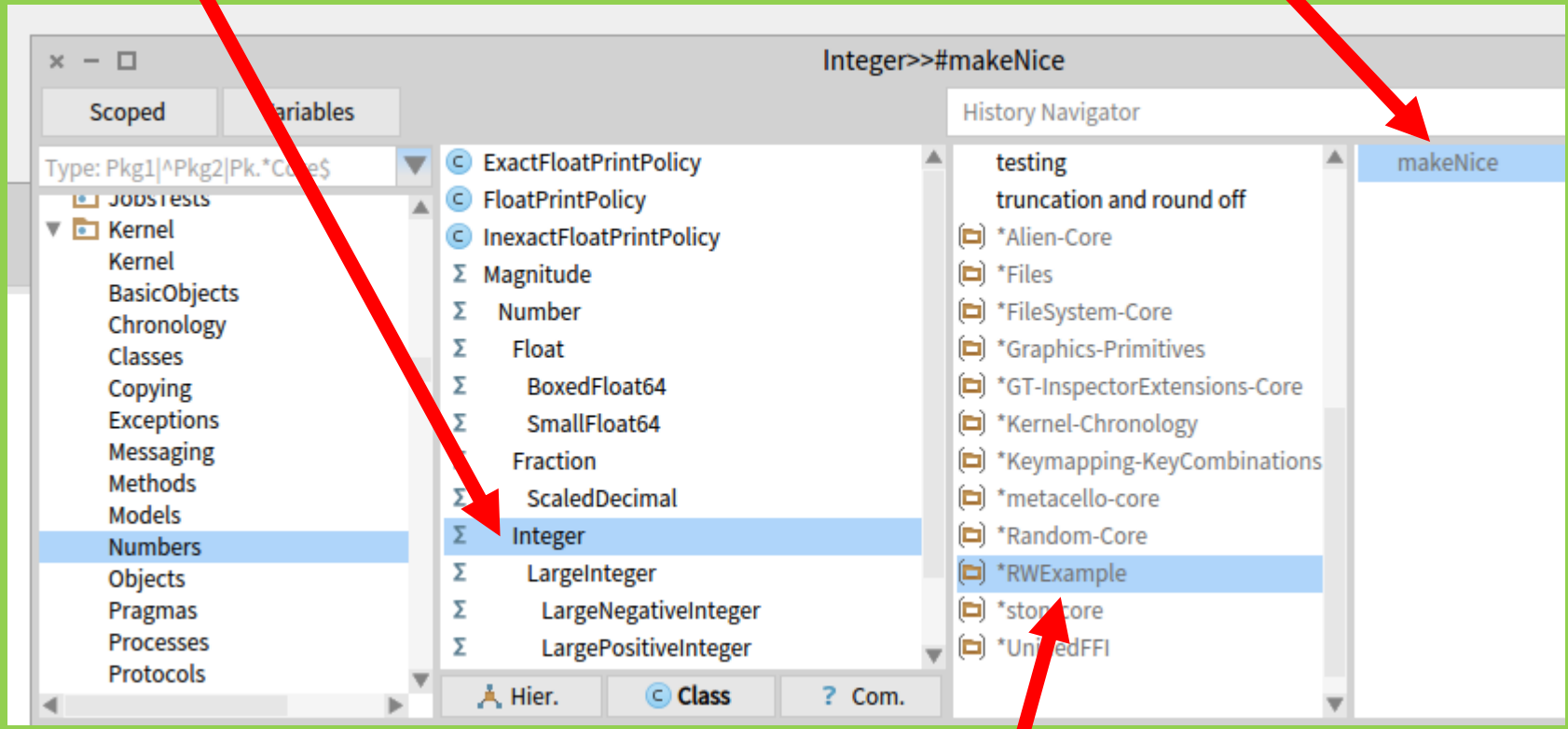
1 replies now with 5



Press Shift Enter : „Spotter“ Find things fast

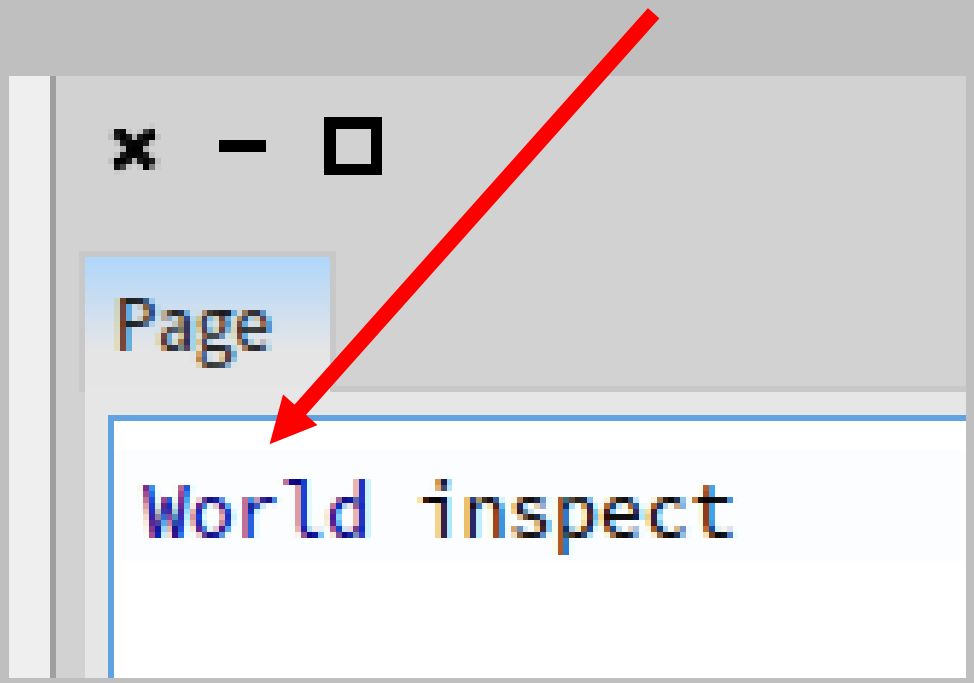


Integer has now a new method

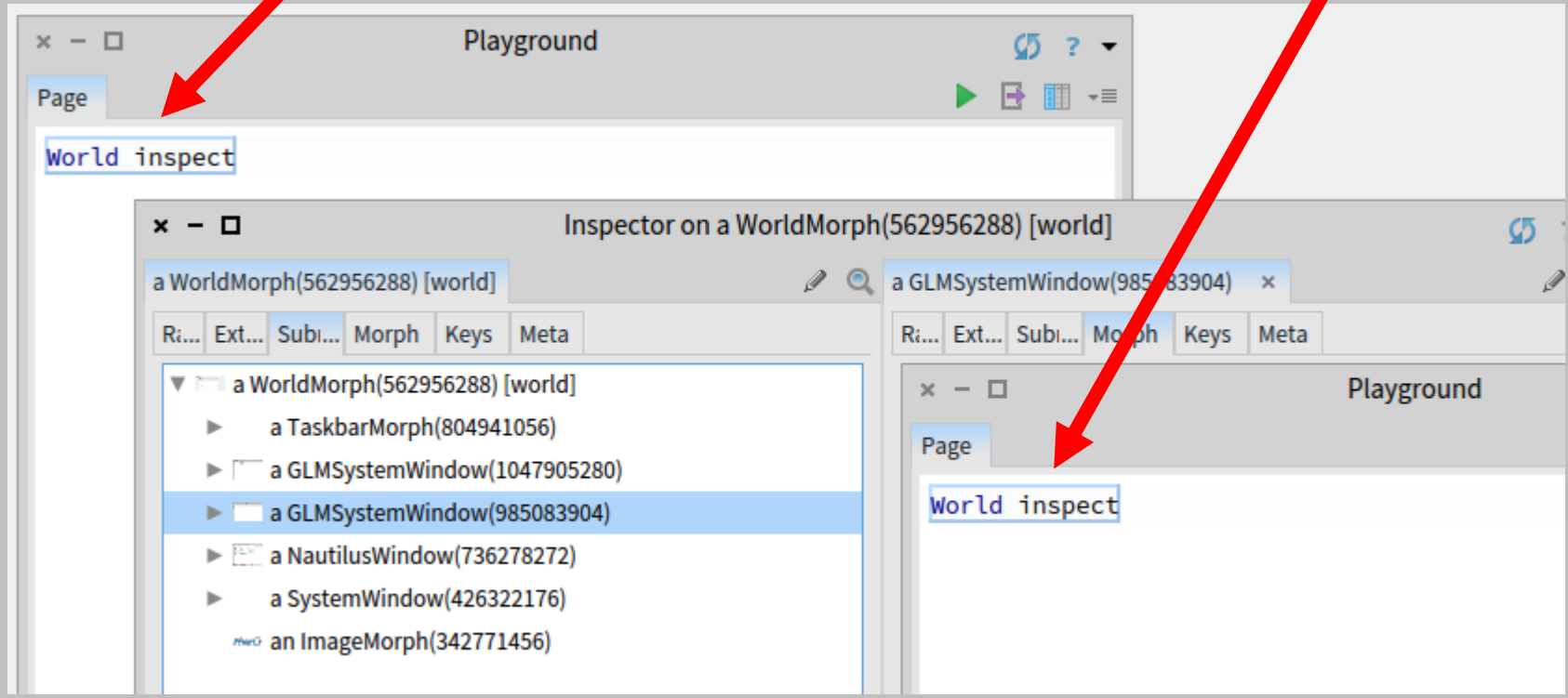


Will be delivered with package RWExample

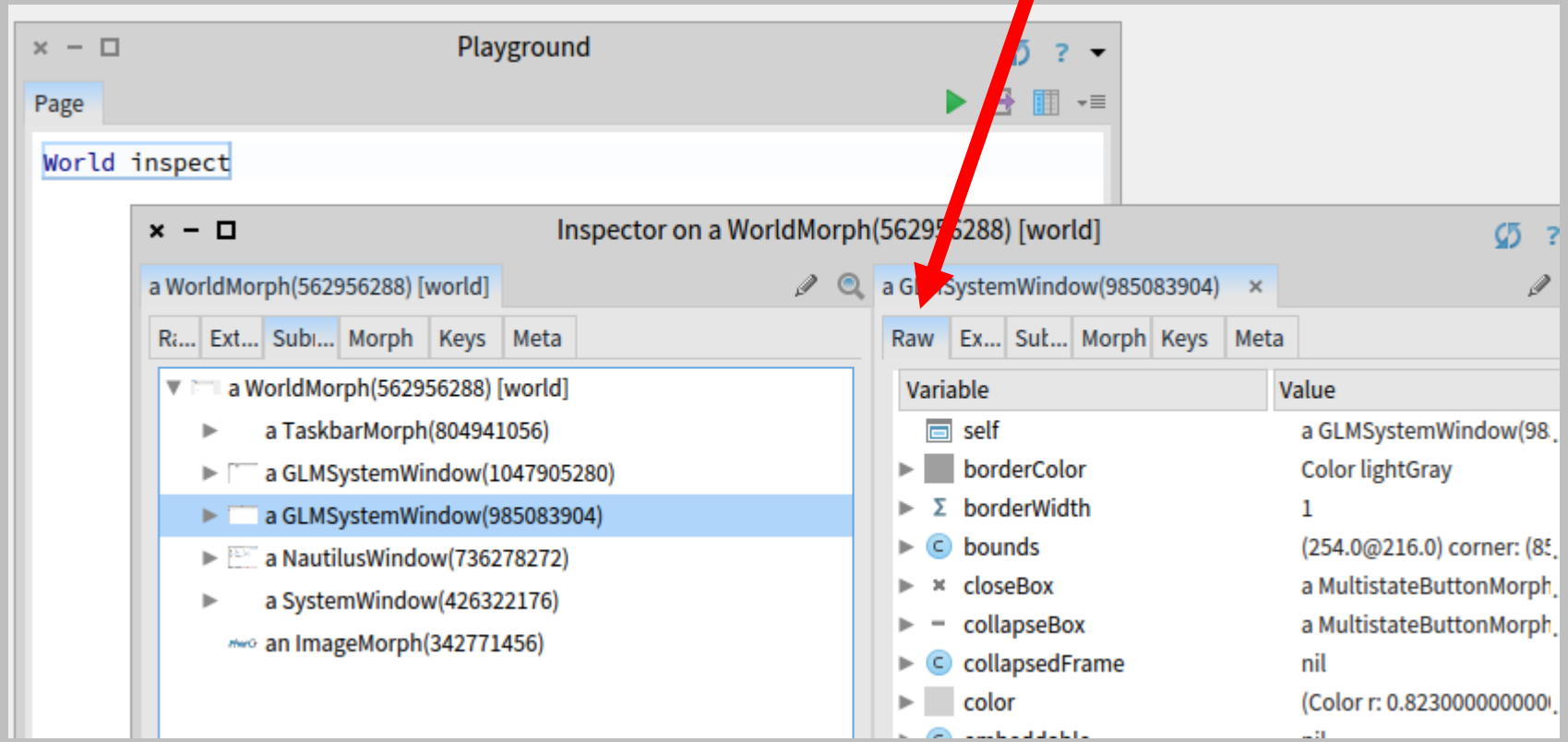
Inspect (nearto) everything



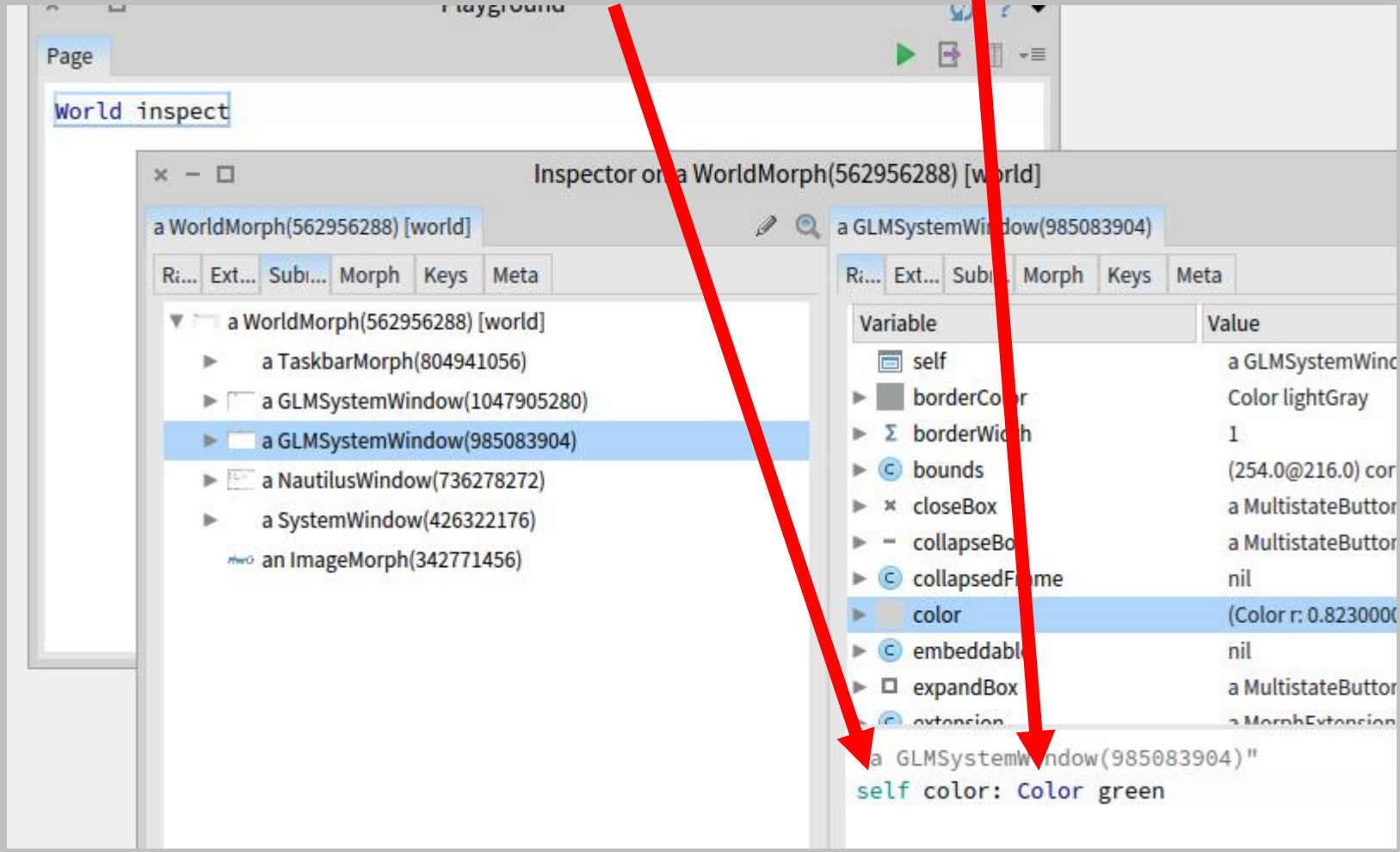
The window is shown in inspector



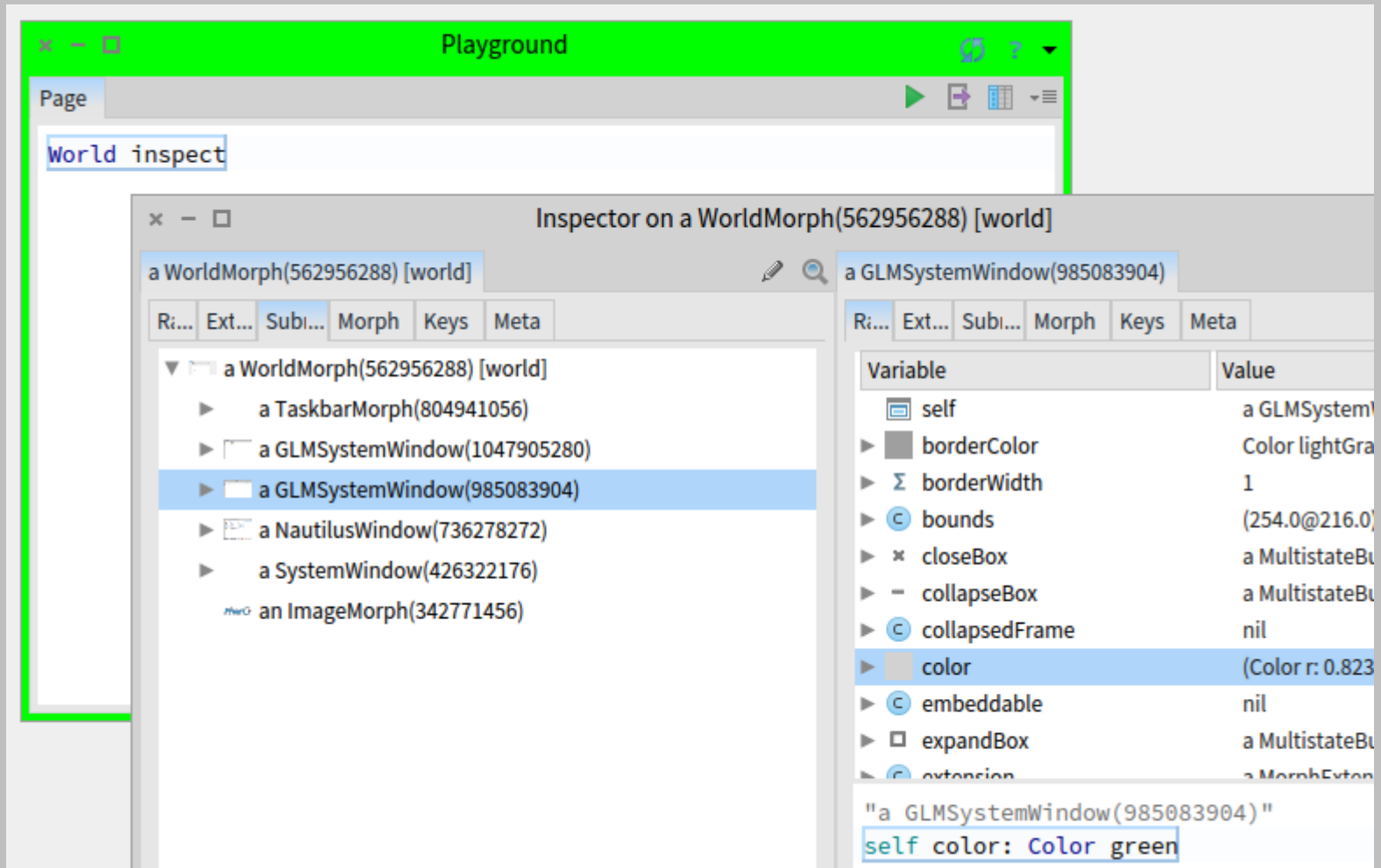
Internal variables on Raw Tab



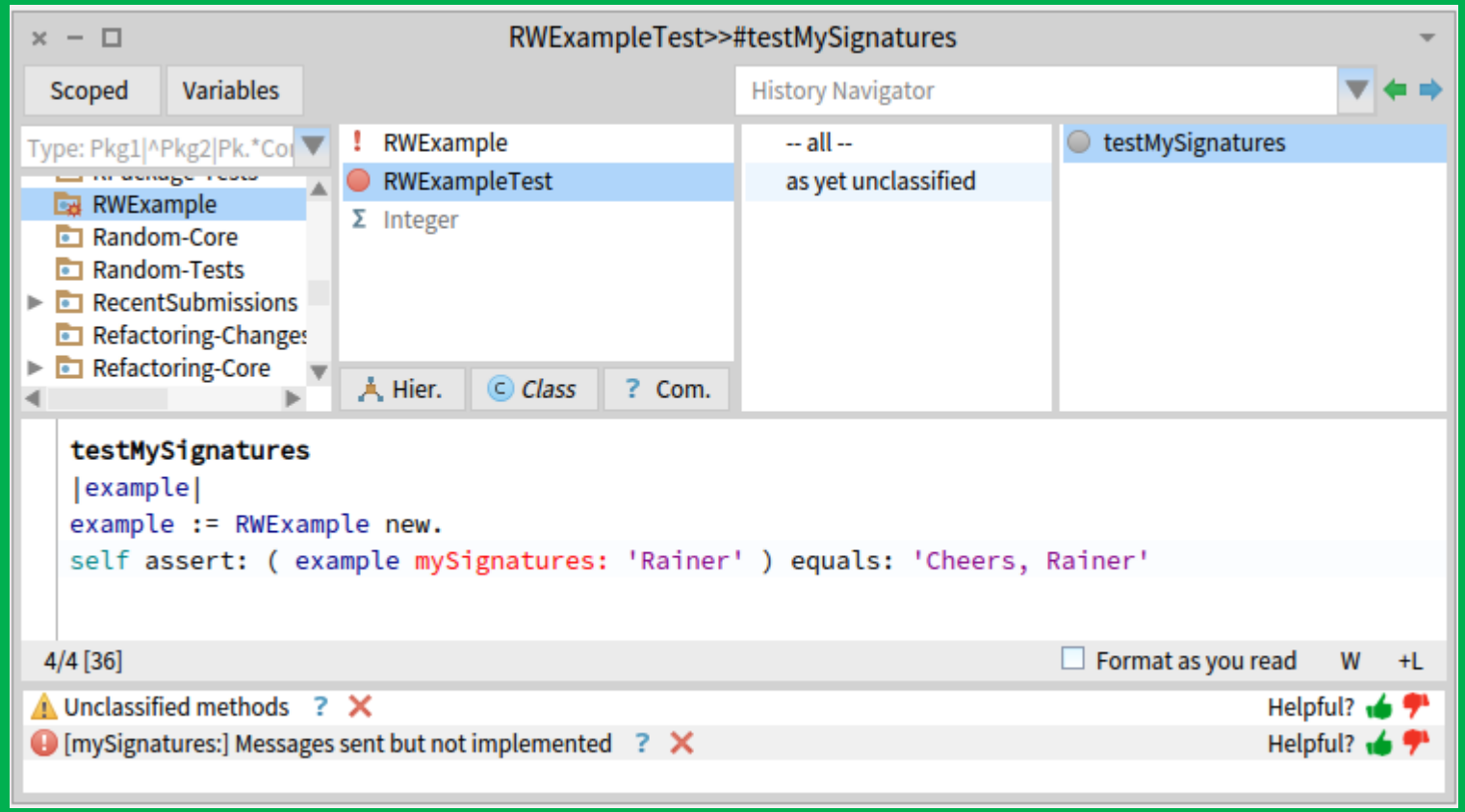
Send to the window a message



The window is now green

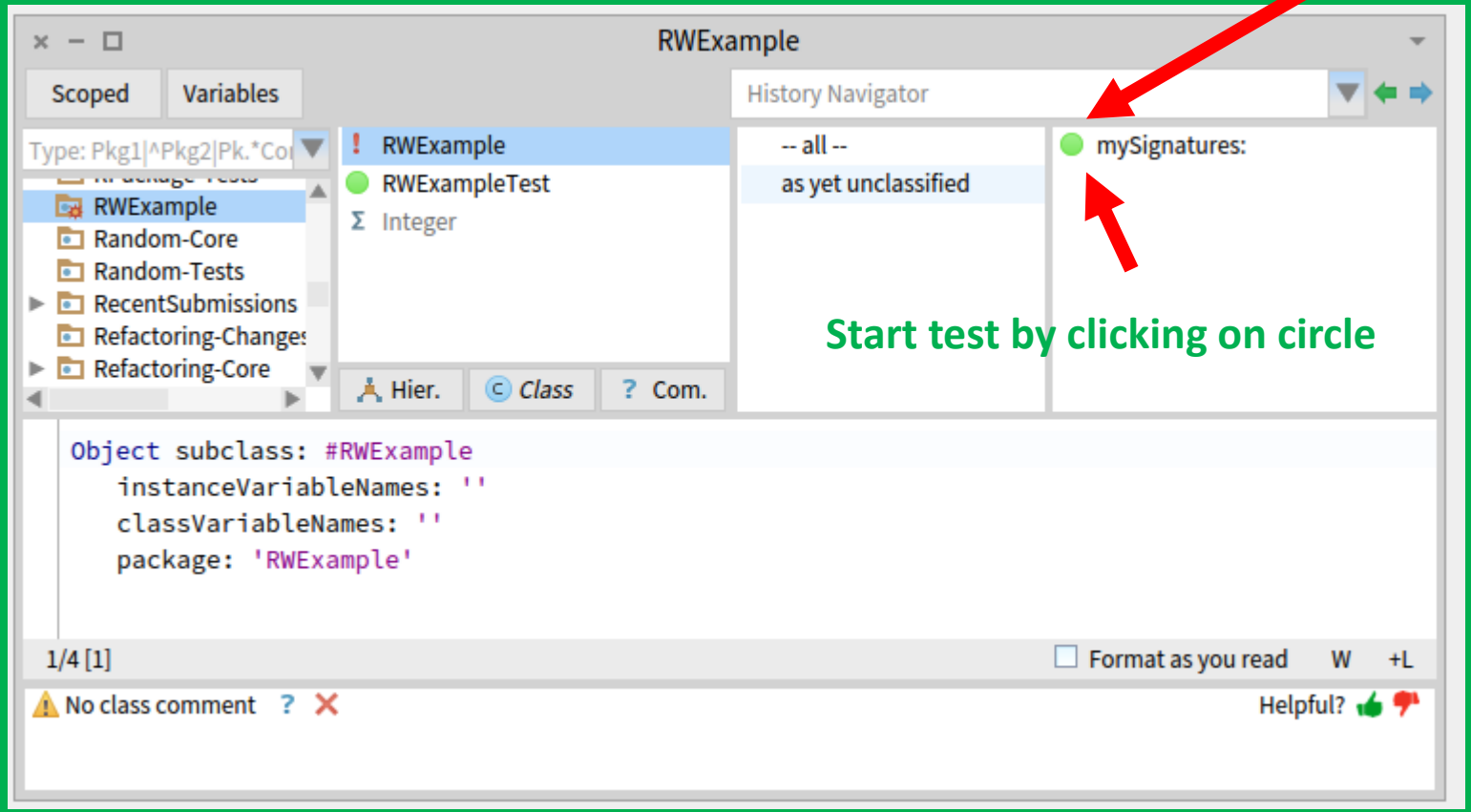


Write a test for a not existing method



Write program in debugger

Method has now a passing test



Ask me:
Rainer Winkler

Have fun ☺