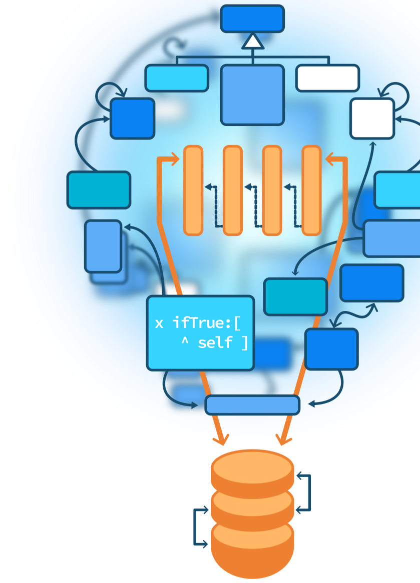
Advanced Object-Oriented Design

Xtreme Test Driven Development

Getting a productivity boost

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone



http://www.pharo.org

Outline

- TDD on **steroids**
- Live programming at **its best**
- Smart tools
- Absolutely **gorgeous** development flow

Principle

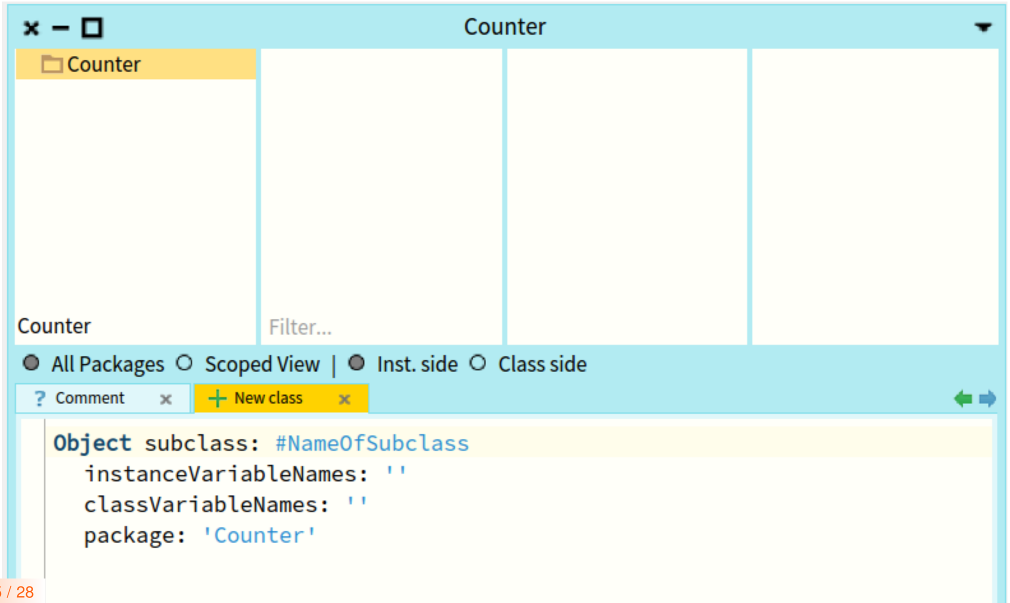
Do **not break** the flow

- Write a test
- When it breaks, define the method **on the fly in the debugger**
- **Resume and continue** until the test is green

Studying an example

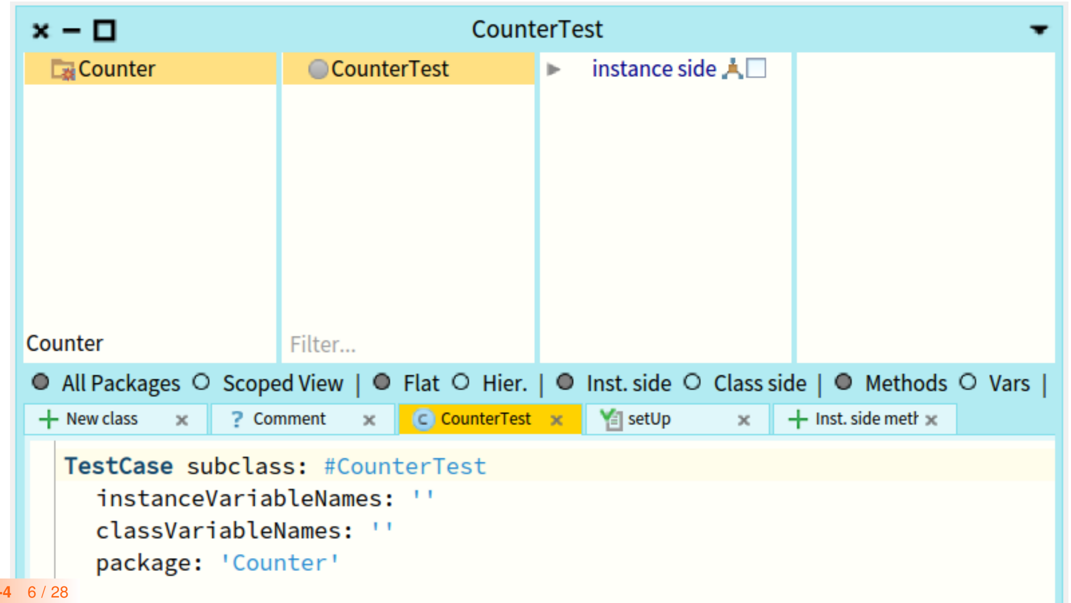
- A dead simple counter. Nothing simpler.
- Focus on the essence of the process!
- You can do it.

An empty package



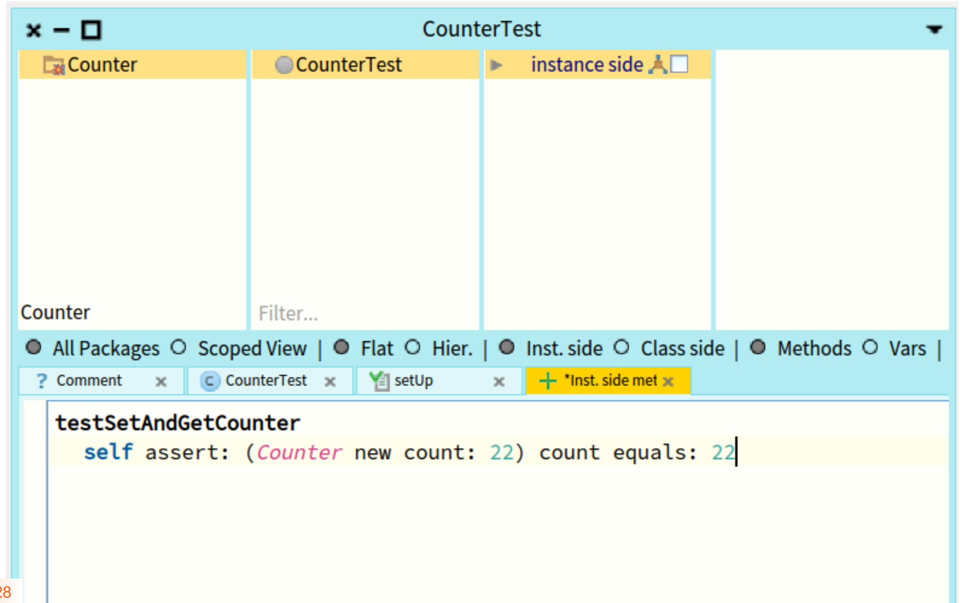
```
Object subclass: #NameOfSubclass
   instanceVariableNames: ''
   classVariableNames: ''
   package: 'Counter'
```

An empty test case class



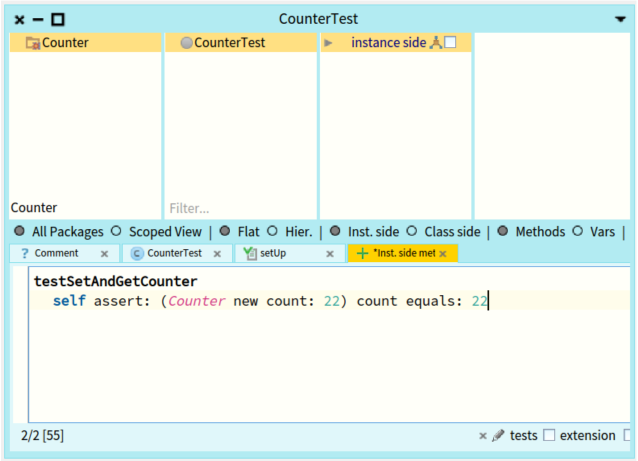
```
TestCase subclass: #CounterTest
   instanceVariableNames: ''
   classVariableNames: ''
   package: 'Counter'
```

A first test



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

A first test (II)

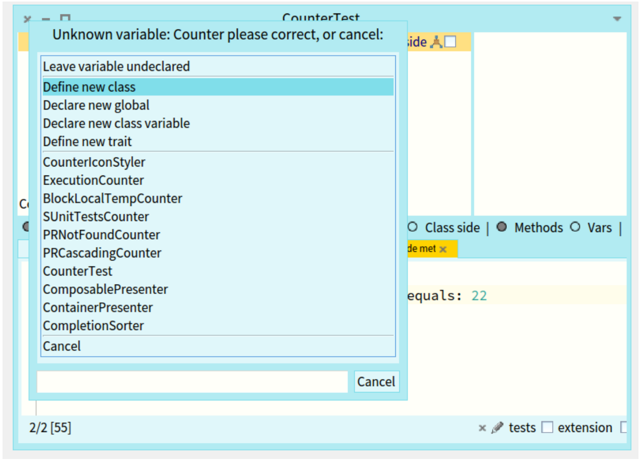


```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

- Method is about to be compiled
- The system knows the class does not exist!

Define a class

- At compile time...



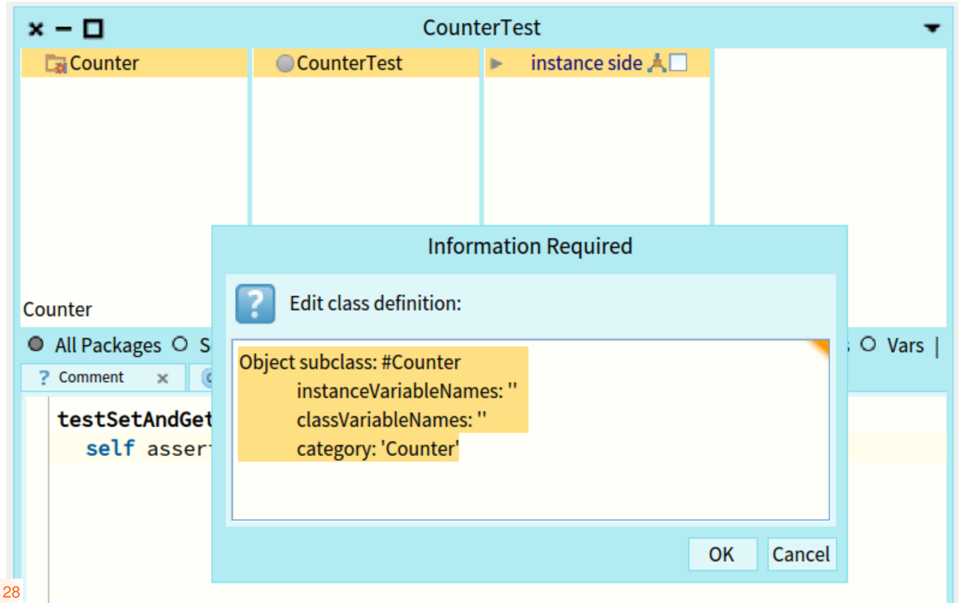
CounterTest

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

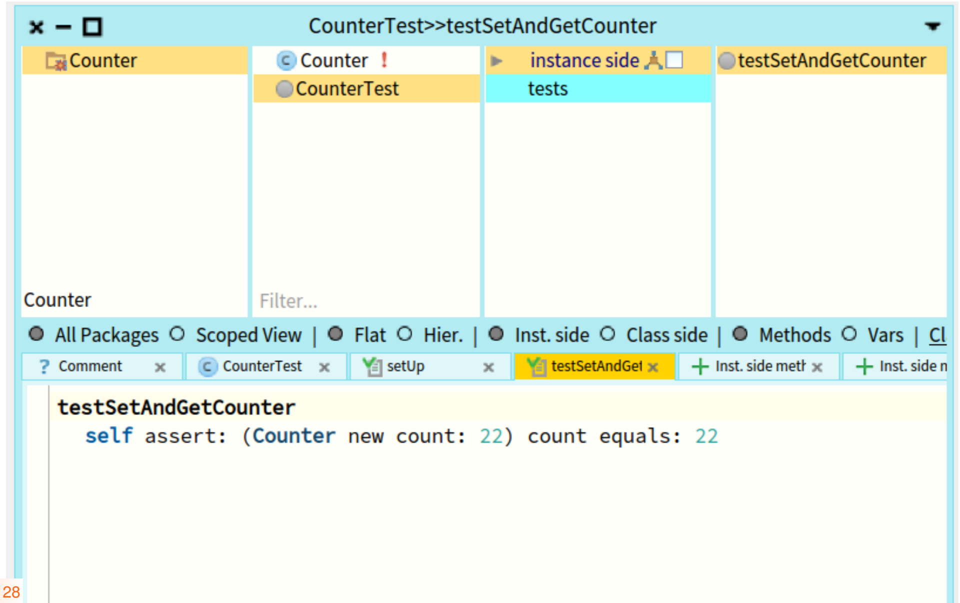
```
equals: 22
```

Define a class (II)



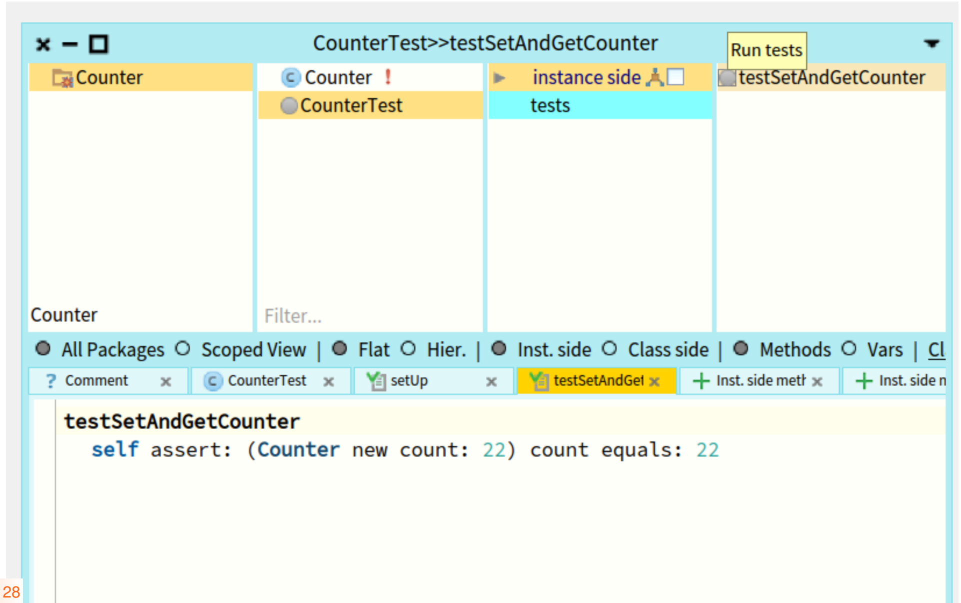
```
Object subclass: #Counter
        instanceVariableNames: ''
        classVariableNames: ''
        category: 'Counter'
```

Test defined but not executed



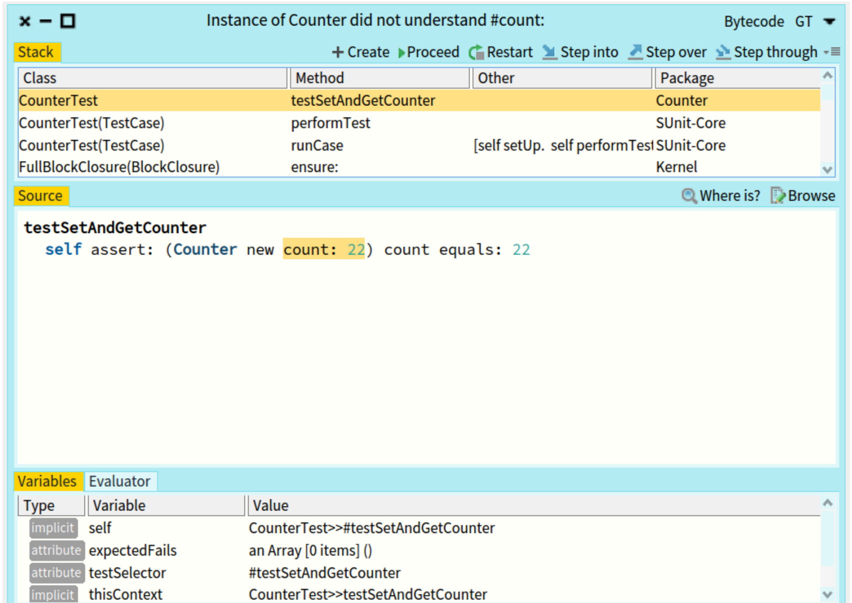
```
testSetAndGetCounter
   self assert: (Counter new count: 22) count equals: 22
```

Running the test



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

First Error

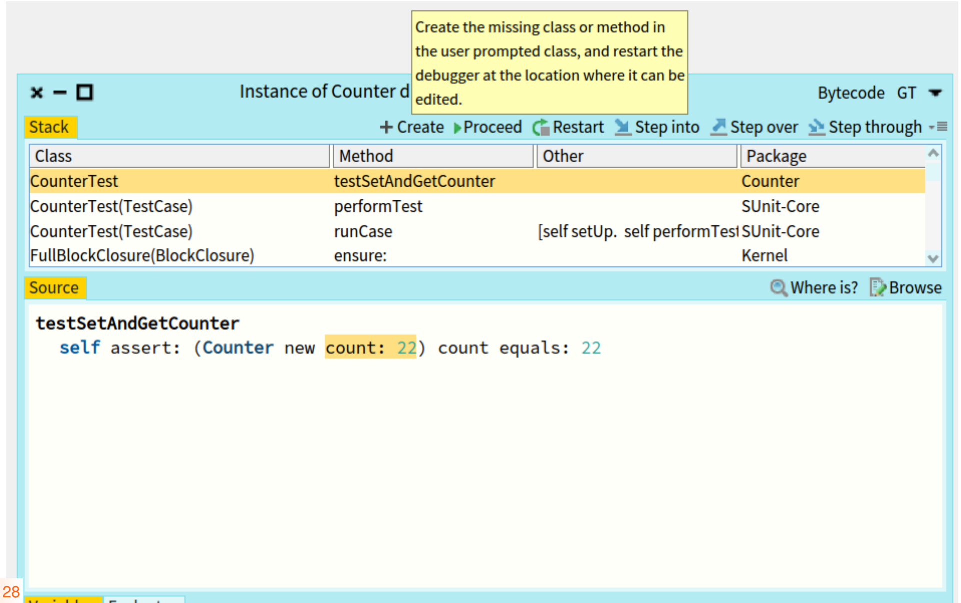

Instance of Counter did not understand #count:

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

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

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

Create a method on the fly



Instance of Counter d

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

Bytecode GT

Stack

+ Create ▶ Proceed Restart Step into Step over Step through

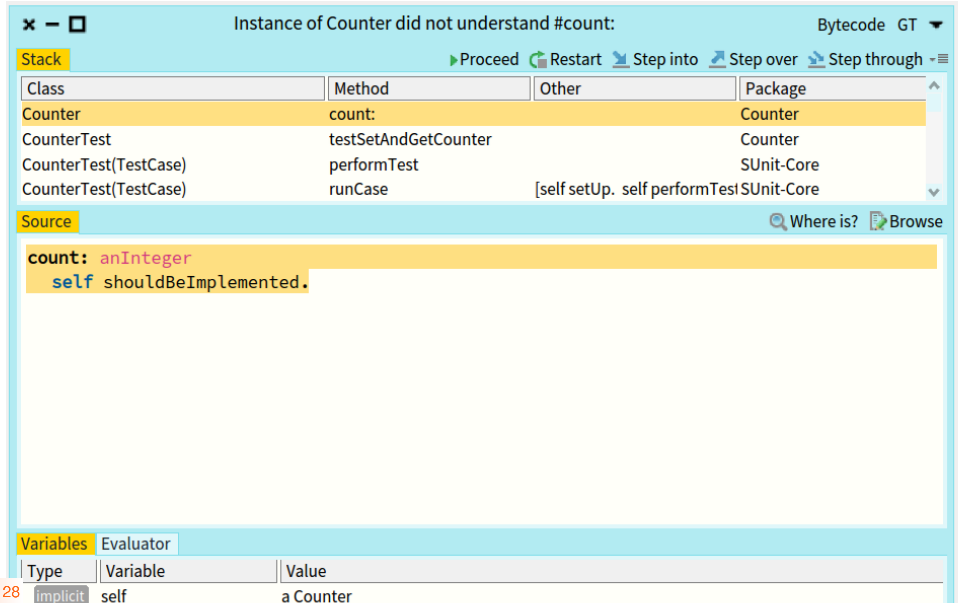
Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

Where is? Browse

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

Create a method on the fly (II)



Instance of Counter did not understand #count:

Stack

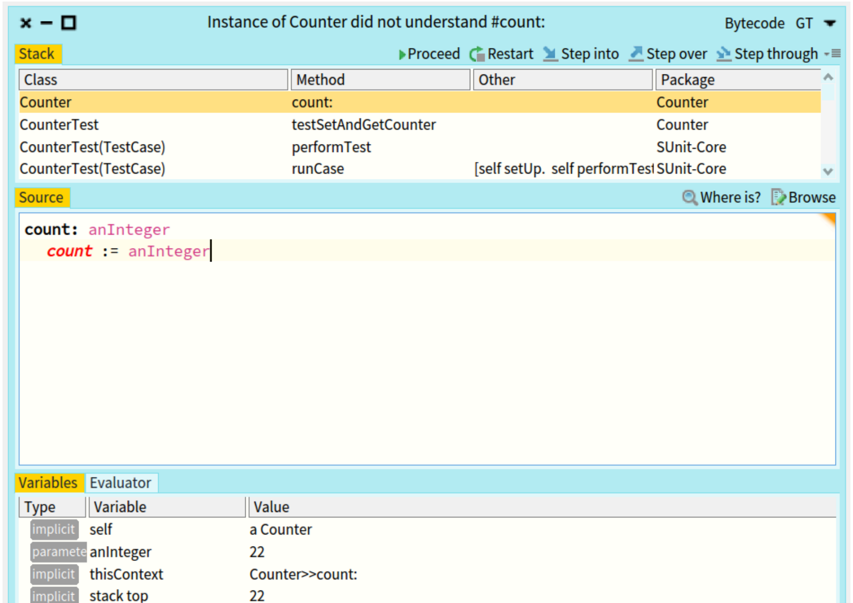
Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

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

Type	Variable	Value
implicit	self	a Counter

Edit the method in the debugger



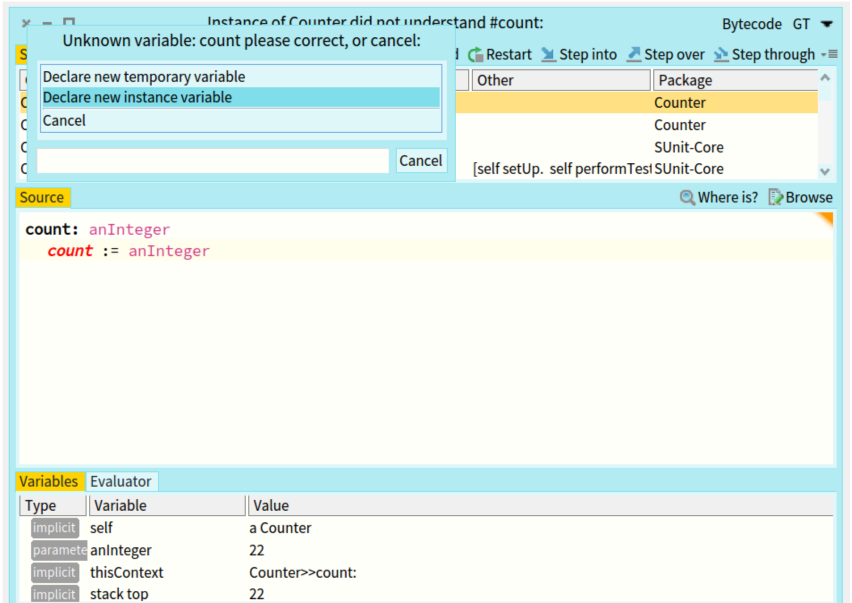
Instance of Counter did not understand #count:

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

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

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

Add an instance variable on the fly



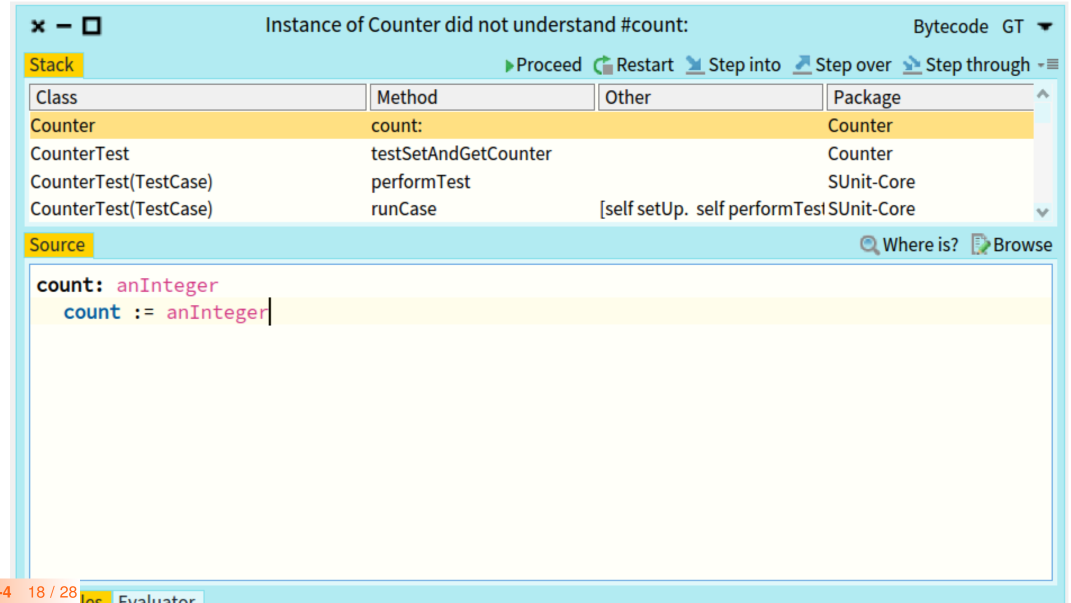
Instance of Counter did not understand #count:
Bytecode GT
Restart
Step into
Step over
Step through
Unknown variable: count please correct, or cancel:
Declare new temporary variable
Declare new instance variable
Cancel
Cancel
Other
Package
Source
Where is?
Browse
Variables
Evaluator

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

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

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

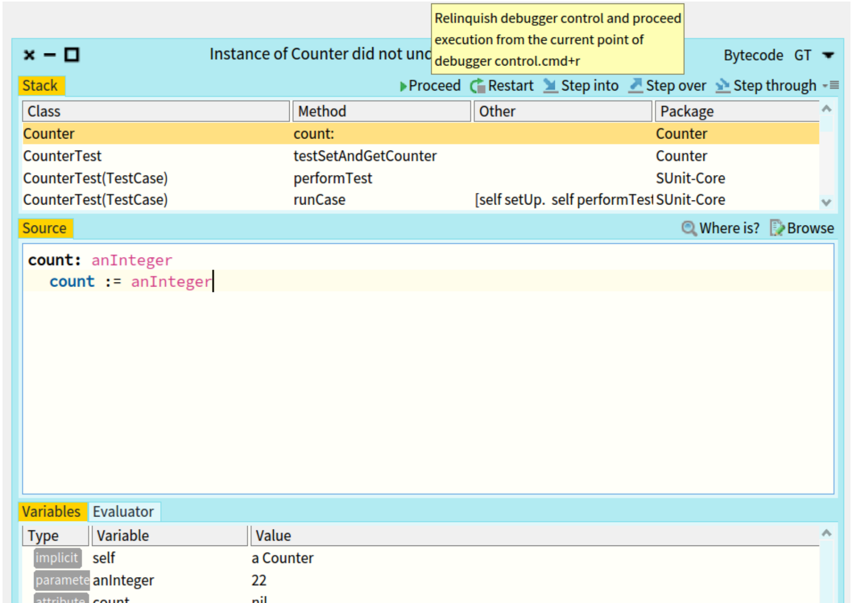
Compile....



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

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

Continue the execution...



Instance of Counter did not unc

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

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

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

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

Supporting the programmer flow

- The system
  - **created** a new method for us
  - **Removed** the stack element with Error
  - **Replaced** it with a call to the new method
  - **Relaunched** execution
- We edited it and recompiled the method
- The system **Continued** execution

New method

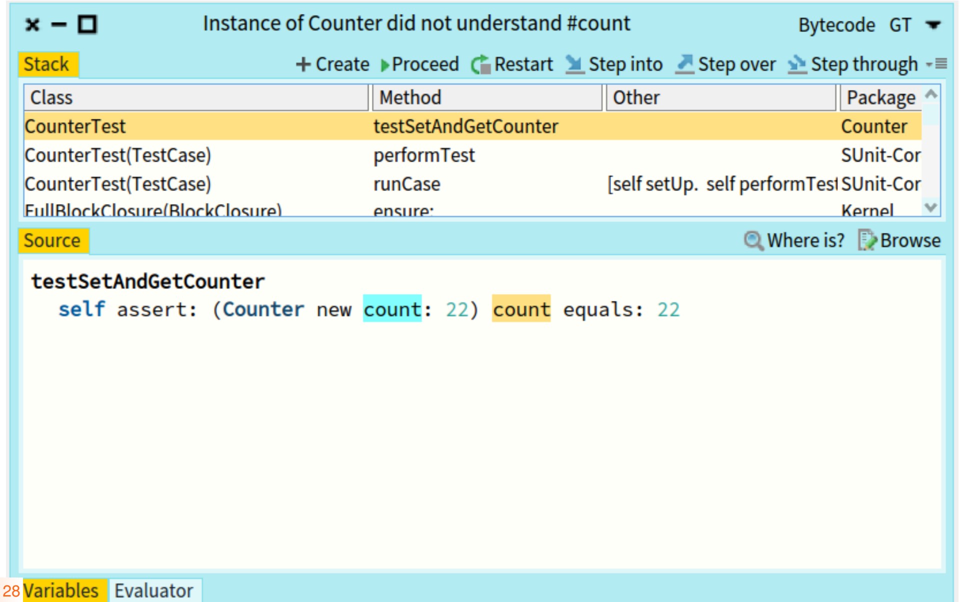
The system:

- Created a new method
- Removed the stack element with Error
- Replaced it with a **call** to the new method

```
count: anInteger
  self shouldBeImplemented
```

- shouldBeImplemented is just an exception so that the debugger stops again

Same story....


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 performTest	SUnit-Cor
FullBlockClosure(BlockClosure)	ensure:		Kernel

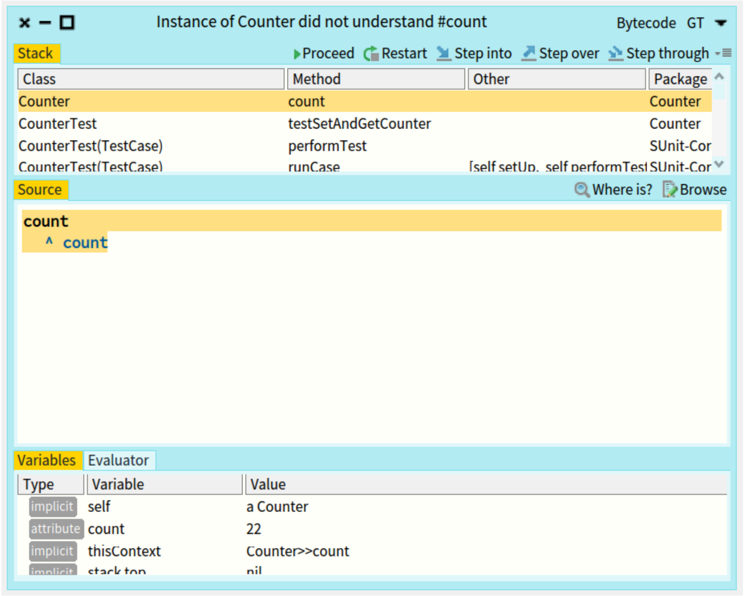
Source

Where is? Browse

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

Variables Evaluator

Debugger also precompiles methods

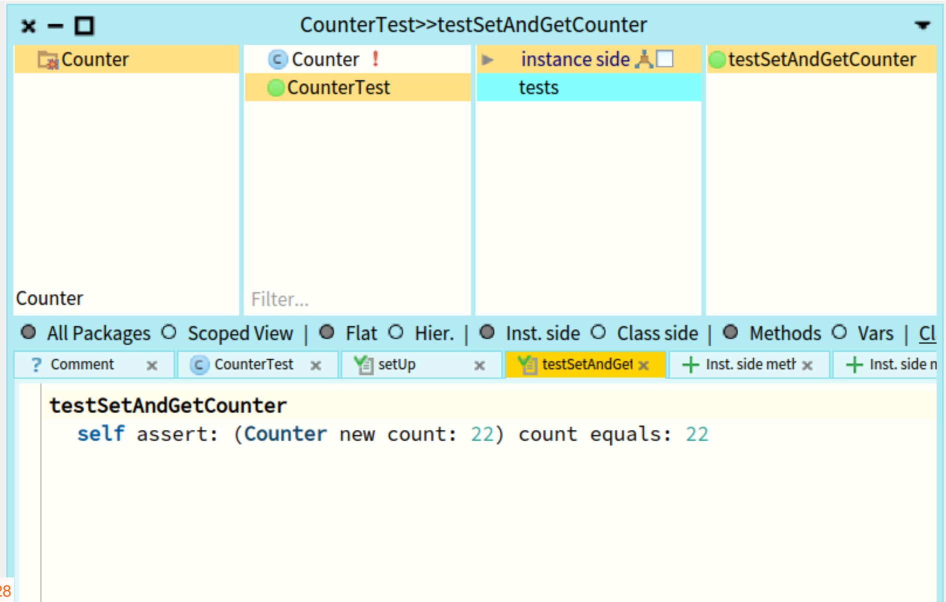


Class	Method	Other	Package
Counter	count		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Cor
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Cor

```
count
    ^ count
```

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

Test is green



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

One Cycle

- Run all the tests
- Ready to commit
- New test

Why XTDD is powerful

- Avoid **guessing** context when coding
- Much much better context
  - inspect that **specific** instance state
  - talk to that **specific** object
- Inspectable / interactable context
- Tests are not a side effect artifact but the **driving** force

Protip from expert Pharo developers

- Grab **as fast as** possible one object
- **Cristalize** your scenario with a test
- Xtreme TDD
- Loop

Advanced Object-Oriented Design and Development with Pharo

A course by

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone

2023