

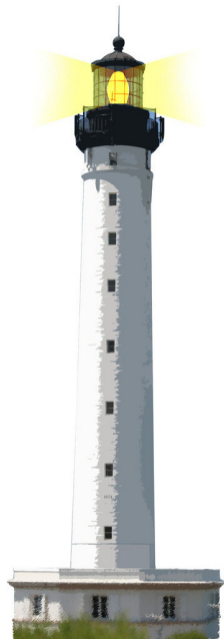
Xtreme Test Driven

S. Ducasse

Master Class



<http://www.pharo.org>



Outline

- TDD on steroids
- Live programming
- Smart tools
- Great developing flow



Principle

- Do not break the flow
- Write a test
- When it breaks, compile method on the fly in the debugger
- Resume and continue until test is green

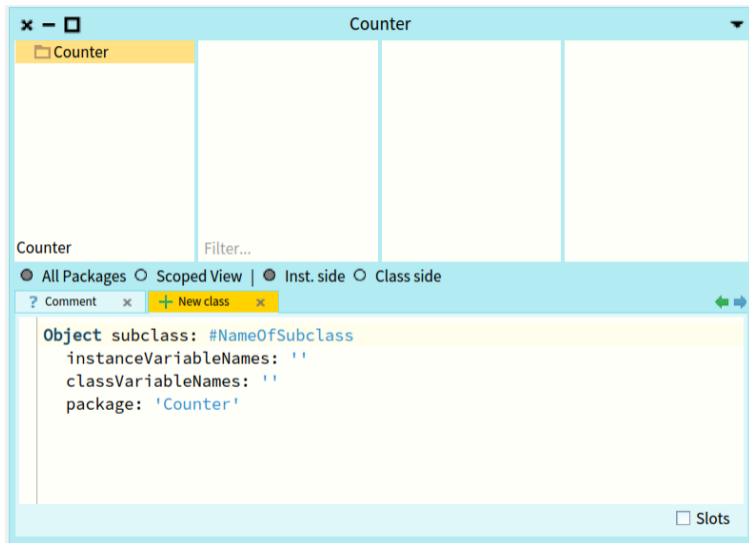


Studying an example

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



An Empty Package



The screenshot shows the Smalltalk IDE interface for a package named "Counter". The package is currently empty. The interface includes a toolbar with "All Packages", "Scoped View", "Inst. side", and "Class side" options. A "New class" button is also visible. The main editor area displays the following code:

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

At the bottom right of the editor, there is a checkbox labeled "Slots".

An Empty Test Case Class

The screenshot shows an IDE window titled "CounterTest". The interface is divided into several panes. At the top, there are tabs for "Counter" and "CounterTest", with "CounterTest" selected. Below the tabs, there are three main panes: the left pane shows the "Counter" class, the middle pane shows a "Filter..." input, and the right pane shows "instance side" with a search icon. Below these panes is a toolbar with radio buttons for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side" (selected), "Class side", "Methods", and "Vars". Below the toolbar is a tab bar with "New class", "Comment", "CounterTest" (selected), "setUp", and "Inst. side meth". The main editor area displays the following code:

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

At the bottom of the editor, there is a warning message: "Test class not in a package with name ending with '-Tests'".

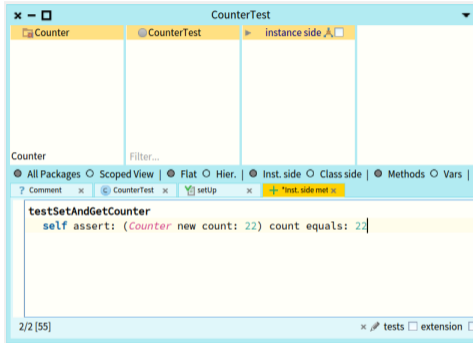
A first test

The screenshot shows an IDE window titled "CounterTest". At the top, there are three tabs: "Counter", "CounterTest", and "instance side". Below the tabs, there are three columns. The first column is labeled "Counter", the second "Filter...", and the third "instance side". Below these columns is a toolbar with radio buttons for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", and "Vars". Below the toolbar is a tab bar with tabs for "? Comment", "CounterTest", "setUp", and "*Inst. side met". The main editor area contains the following code:

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

At the bottom of the editor, there is a status bar showing "2/2 [55]" and "tests extension".

A first test



The screenshot shows an IDE window titled "CounterTest". The interface includes a breadcrumb "Counter > CounterTest" and a view selector "instance side". Below this is a toolbar with options: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side" (selected), "Class side", "Methods", and "Vars". The main editor area contains the following code:

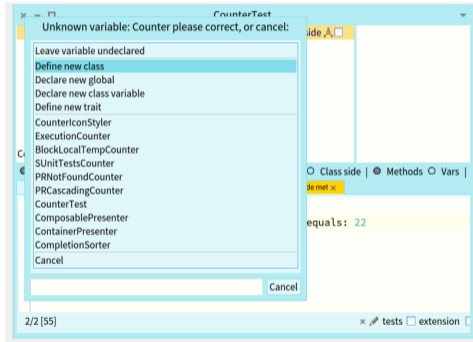
```
testSetAndGetCounter
self assert: (Counter new count: 22) count equals: 22
```

At the bottom of the window, there is a status bar showing "2/2 [55]" and icons for "tests" and "extension".

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

Define a class

- At compile time...



Define a class (II)

The screenshot shows an IDE window titled "CounterTest" with a tab for "CounterTest" selected. A dialog box titled "Information Required" is open, prompting the user to "Edit class definition:". The dialog contains a text area with the following content:

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

At the bottom of the dialog are "OK" and "Cancel" buttons. The IDE background shows a code editor with the following code:

```
testSetAndGet  
self asser
```

The status bar at the bottom of the IDE window displays "2/2 [55]" and "tests extension".

Test Defined but Not Executed

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes:

- Left Pane:** Shows a project structure with "Counter" and "CounterTest".
- Top Middle Pane:** Shows a tree view with "instance side" selected and "tests" listed below it.
- Bottom Middle Pane:** Shows the source code for the `testSetAndGetCounter` method:

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```
- Bottom Right Pane:** Shows the execution progress, currently at "1/2 [1]".

The IDE's toolbar at the bottom indicates that the "testSetAndGetCounter" method is selected for execution, but the progress bar shows that only the first step of a two-step process has been completed.

Running the test

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes:

- Left Pane:** A tree view showing the package structure. "Counter" is selected, and "CounterTest" is highlighted.
- Middle Pane:** A view of the selected class, showing "instance side" and "tests".
- Right Pane:** A list of test methods, with "testSetAndGetCounter" selected.
- Bottom Pane:** A code editor showing the implementation of the test method:

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

At the bottom of the window, there is a status bar showing "1/2 [1]".

First Error

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 performTest	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

- Of course, we did not define any method, yet!

Create a method on the fly

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 performTest]	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 a method on the fly (II)

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  
self shouldBeImplemented.
```

Variables Evaluator

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

Edit the method in the debugger (III)

The screenshot shows a debugger window titled "Instance of Counter did not understand #count:". The window is divided into three main sections: Stack, Source, and Variables/Evaluator.

Stack: A table showing the call stack. The top frame is highlighted in yellow.

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: A code editor showing the source code for the `count:` method. The current line is highlighted in yellow.

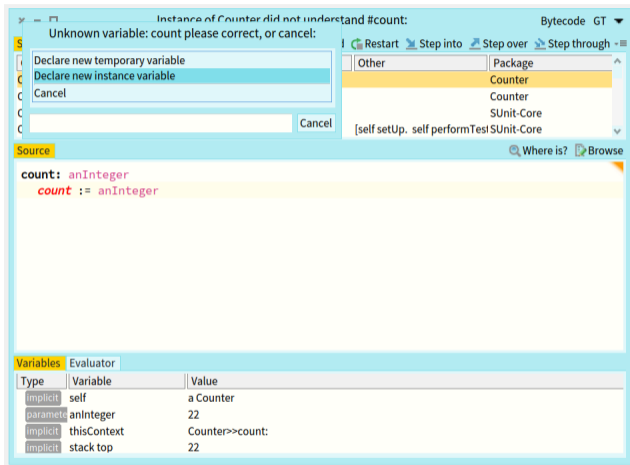
```
count: anInteger  
  count := anInteger |
```

Variables/Evaluator: A table showing the current state of variables.

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

- But there is no instance variable!
- So what?

Add an instance variable on the fly



The screenshot shows an IDE window titled "Instance of Counter did not understand #count:". The error message is "Unknown variable: count please correct, or cancel:". A context menu is open over the error, with "Declare new instance variable" selected. Other options include "Declare new temporary variable" and "Cancel".

The "Source" pane shows the following code:

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

The "Variables" pane shows the following table:

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

Compile....

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
parameter	anInteger	22
attribute	count	nil
implicit	thisContext	Counter>>count:

Continue the execution...

Instance of Counter did not un... Bytecode GT

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

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
parameter	anInteger	22
attribute	count	nil
implicit	thisContext	Counter>>count:

Stepping back

- The system created a new method
- Removed the stack element with Error
- Replace it with a call to the new method
- Relaunches execution
- We edited it and recompiled
- Continued execution



Stepping back (II)

- The system created a new method
- Removed the stack element with Error
- Replace it with a **call** to the new method

```
count: anInteger  
  self shouldBelImplemented
```

- `shouldBelImplemented` 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

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

Smart tools precompile methods

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
```

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

- A method with the same name than an attribute...
- Probably an accessor

Test is green

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes:

- Left Pane:** Shows a tree view with "Counter" selected.
- Top Middle Pane:** Shows a breadcrumb path: "Counter !", "CounterTest", "instance side", and "testSetAndGetCounter".
- Right Pane:** Shows a list of tests, with "testSetAndGetCounter" selected and marked with a green circle.
- Bottom Pane:** Displays the source code for the selected test method:

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

At the bottom left of the IDE window, the text "1/2 [1]" indicates that the test passed on the first attempt.

One Cycle

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



Stepping back

- Avoid guessing when coding
- Much much better context
 - inspecting that specific instance state access
 - talking to that specific object
- Inspectable / interactable context
- Tests are not a side effect artefacts but the driving force



Protip from expert Pharo developers

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

A course by



and



in collaboration with



Inria 2020

Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France

<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>