

# Xtreme Test Driven Development

Getting a productivity boost

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



# 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

The screenshot shows an IDE window titled "Counter". The left sidebar displays a project tree with a folder named "Counter". The main editor area is divided into three vertical panes, all of which are empty. Below the editor panes, there is a toolbar with several options: "All Packages" (selected), "Scoped View", "Inst. side" (selected), and "Class side". Below the toolbar, there are two buttons: "Comment" and "New class". The bottom pane of the editor contains the following code:

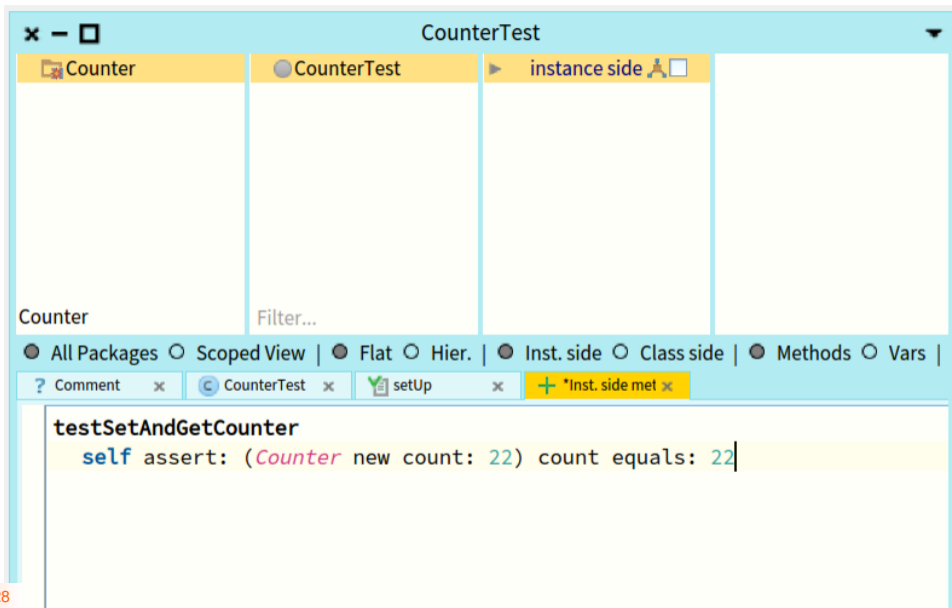
```
Object subclass: #NameOfSubclass
  instanceVariableNames: ''
  classVariableNames: ''
  package: 'Counter'
```

# An empty test case class

The screenshot shows an IDE window titled "CounterTest". The interface is divided into several panes. On the left, a "Counter" package is visible. The main area shows the "CounterTest" class selected, with the "instance side" view active. Below the panes, there are navigation options: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side" (selected), "Class side", "Methods", and "Vars". A toolbar contains buttons for "New class", "Comment", "CounterTest", "setUp", and "Inst. side metr". The code editor at the bottom displays the following code:

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

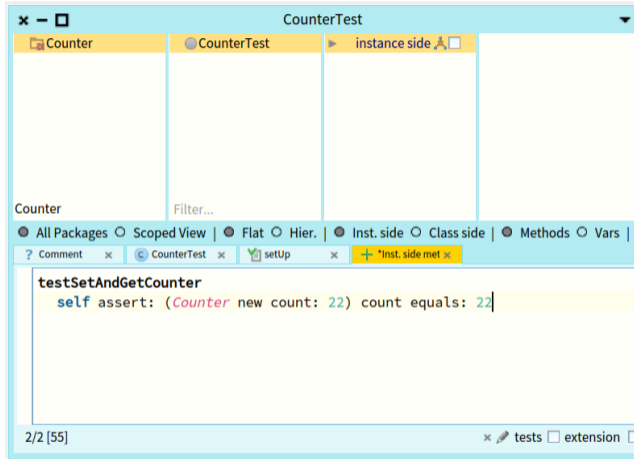
# A first test



The screenshot shows an IDE window titled "CounterTest". The interface is divided into several panes. At the top, there are three tabs: "Counter", "CounterTest", and "instance side". Below these tabs, the "CounterTest" pane is active, showing a search filter "Filter...". Below the filter, there are radio buttons for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", and "Vars". The "Inst. side" radio button is selected. Below the radio buttons, there are several tabs: "? Comment", "CounterTest", "setUp", and "+ \*Inst. side met". The "+ \*Inst. side met" tab is highlighted in yellow. The main editor area shows the following code:

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

# A first test (II)



The screenshot shows an IDE window titled "CounterTest". The interface includes a breadcrumb "Counter > CounterTest" and a view selector set to "instance side". Below the breadcrumb is a search bar labeled "Filter...". The main editor area displays the following code:

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

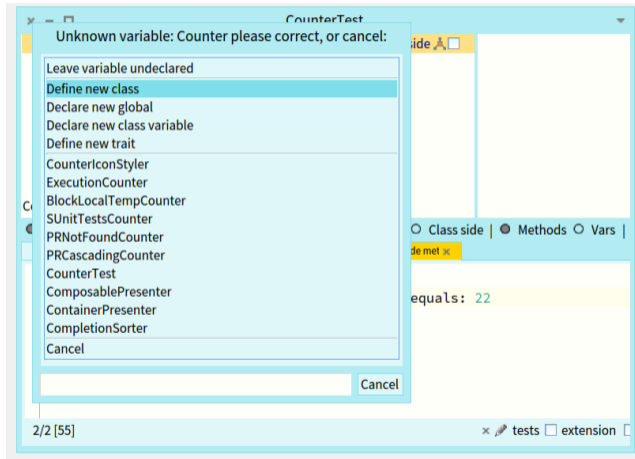
The IDE's status bar at the bottom shows "2/2 [55]" and "tests extension".

- 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 breadcrumb trail: Counter > CounterTest > instance side. A dialog box titled "Information Required" is open, displaying the following class definition:

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

The dialog box has "OK" and "Cancel" buttons at the bottom right. In the background, a code editor shows the following code snippet:

```
testSetAndGet
self asser
```

# 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 tree view with "Counter" selected.
- Middle Pane:** Shows a tree view with "Counter !" and "CounterTest" listed. "CounterTest" is selected.
- Right Pane:** Shows a tree view with "instance side" and "tests" listed. "tests" is selected.

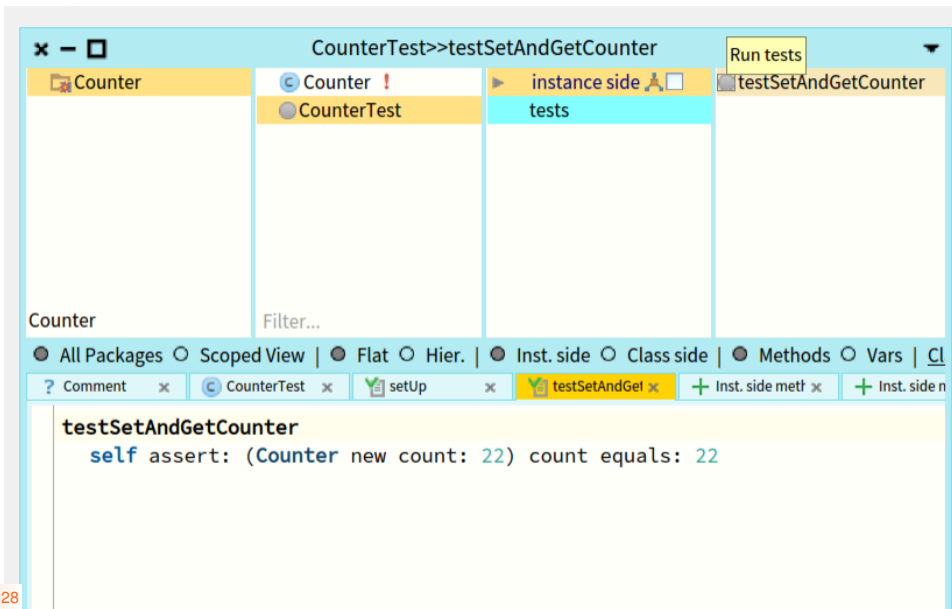
Below the panes, there are view options: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", and "Vars". The "Inst. side" option is selected.

At the bottom, there is a code editor showing the following code:

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

The code editor also shows a tab for "testSetAndGetCounter" and a "Filter..." input field.

# 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.
- Middle-Left Pane:** A list of classes. "Counter" (with a red exclamation mark) and "CounterTest" are listed.
- Middle-Right Pane:** A list of test methods. "instance side" and "tests" are visible.
- Right Pane:** A list of test cases. "testSetAndGetCounter" is selected.

Below the panes is a toolbar with various options: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", "Vars", and "Cl". Below the toolbar is a tabbed interface with tabs for "Comment", "CounterTest", "setUp", "testSetAndGet", "Inst. side meth", and "Inst. side n".

The main editor area displays the following code snippet:

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

# 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

# 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
```

# 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

# Edit the method in the debugger

The screenshot shows an IDE debugger window titled "Instance of Counter did not understand #count:". The window is divided into several sections:

- Stack:** A table showing the call stack. The top frame is highlighted in yellow and shows the `count:` method in the `Counter` class. Below it are frames for `testSetAndGetCounter` in `CounterTest`, `performTest` in `CounterTest(TestCase)`, and `runCase` in `CounterTest(TestCase)`.
- Source:** A text area showing the source code for the `count:` method. The code is:

```
count: anInteger  
  count := anInteger
```
- Variables:** A table showing the current state of variables in the scope. The table has columns for Type, Variable, and Value.

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



# Add an instance variable on the fly

The screenshot shows an IDE window titled "Instance of Counter did not understand #count:". A dialog box is open with the message "Unknown variable: count please correct, or cancel:". The dialog has three options: "Declare new temporary variable", "Declare new instance variable" (which is highlighted), and "Cancel".

Below the dialog, the "Source" view shows the following code:

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

The "Variables" view at the bottom 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 |
```

# Continue the execution...

Instance of Counter did not un...  
Relinquish debugger control and proceed execution from the current point of debugger control.cmd+r

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

# 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 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
```

# Debugger also precompiles methods

The screenshot shows a debugger window titled "Instance of Counter did not understand #count". The window is divided into several sections:

- Stack:** A table showing the call stack. The top frame is highlighted in yellow.
- Source:** A text editor showing the source code for the `count` method.
- Variables:** A table showing the current state of variables.

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**

```
count
  ^ count
```

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

# Test is green

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

- Left Pane:** Shows a project tree with "Counter" selected.
- Middle-Left Pane:** Shows a class hierarchy with "Counter" (marked with a red exclamation mark) and "CounterTest" (highlighted in yellow).
- Middle-Right Pane:** Shows the "instance side" view with "tests" listed.
- Right Pane:** Shows the "testSetAndGetCounter" method, which is highlighted in yellow.

Below the panes, there are navigation and filter options:

- Buttons for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", and "Vars".
- A search bar labeled "Filter...".

The bottom of the window shows a tabbed editor with the following tabs:

- Comment
- CounterTest
- setUp
- testSetAndGet (highlighted in yellow)
- Inst. side met
- Inst. side n

The main editor area displays the following code snippet:

```
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



Produced as part of the course on <http://www.fun-mooc.fr>

# Advanced Object-Oriented Design and Development with Pharo

A course by

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



Inria  
LearningLab



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/>