

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

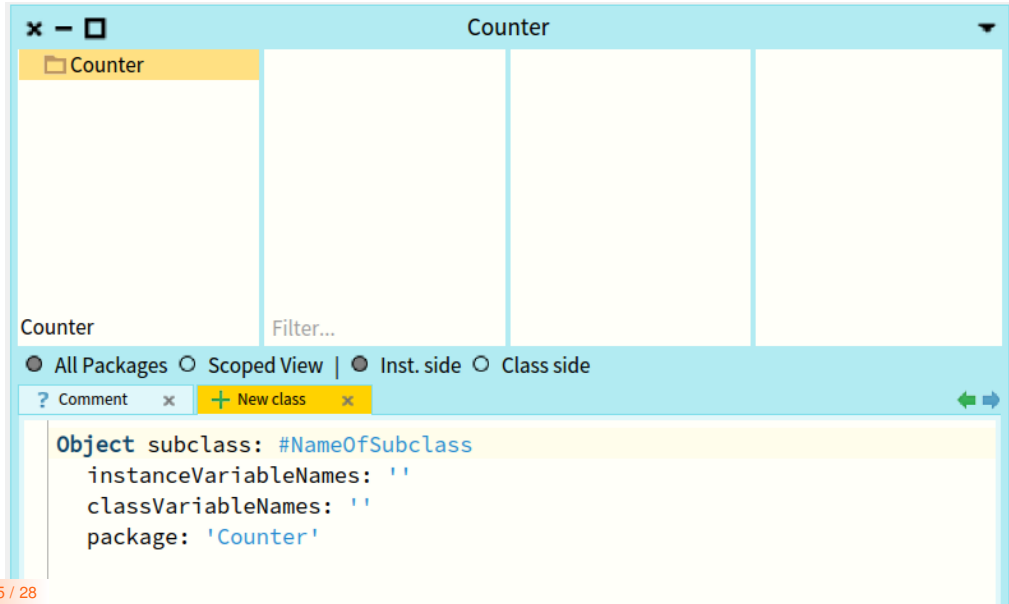


Studying an example

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



An empty package



An empty test case class

The screenshot shows the IntelliJ IDE interface with a project named 'CounterTest'. The top toolbar includes icons for 'x', '-', and a square. Below the toolbar, the 'Counter' package is expanded, showing the 'CounterTest' class. The 'instance side' view is active, displaying a search bar labeled 'Filter...'. The bottom toolbar contains buttons for 'All Packages', 'Scoped View', 'Flat', 'Hier.', 'Inst. side' (selected), 'Class side', 'Methods', and 'Vars'. Below the toolbar, a tab bar shows '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'
```

A first test

The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for file operations and a search icon. The main window is titled "CounterTest" and is divided into four panes. The left pane shows a project structure with a folder "Counter". The second pane shows a "CounterTest" class with a "Filter..." input field. The third pane shows the "instance side" view. The bottom pane displays the test code for "testSetAndGetCounter".

CounterTest

Counter

CounterTest

instance side

Counter

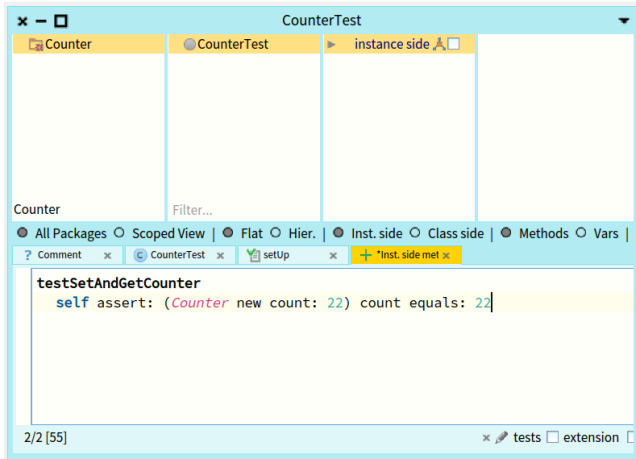
Filter...

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

? Comment x C CounterTest x ✓ setUp x + *Inst. side met x

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

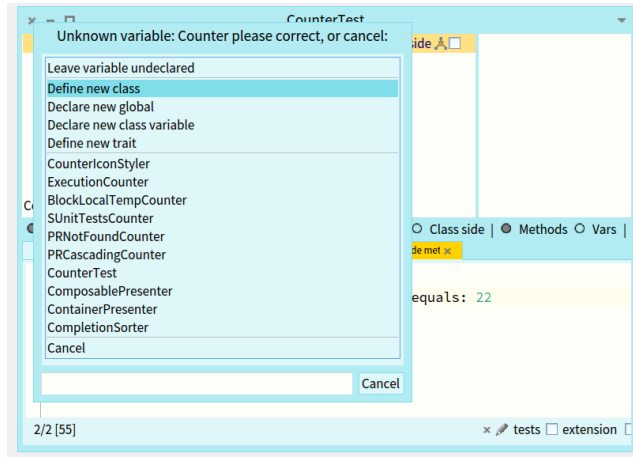
A first test (II)



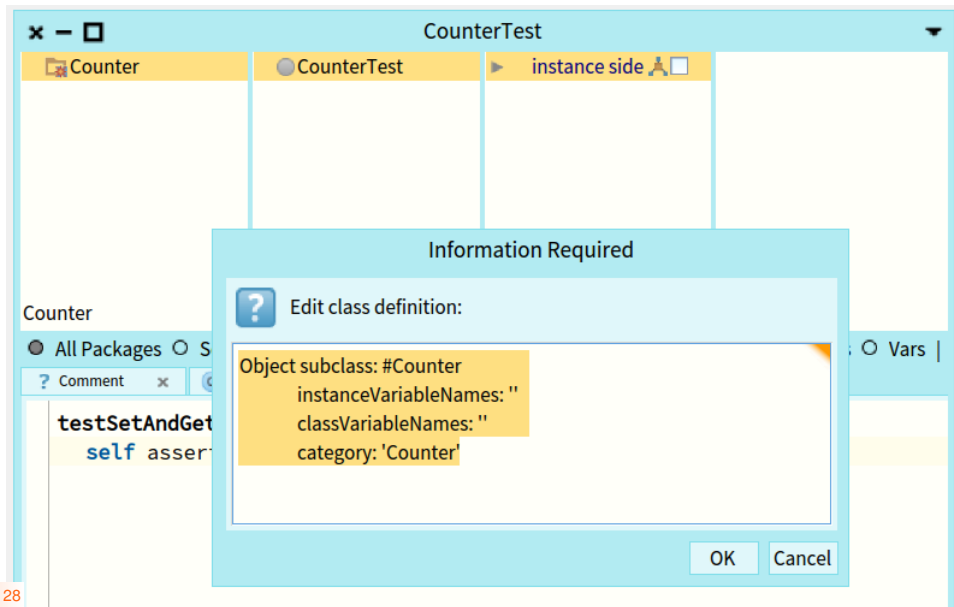
- Method is about to be compiled
- The system knows the class does not exist!

Define a class

- At compile time...



Define a class (II)

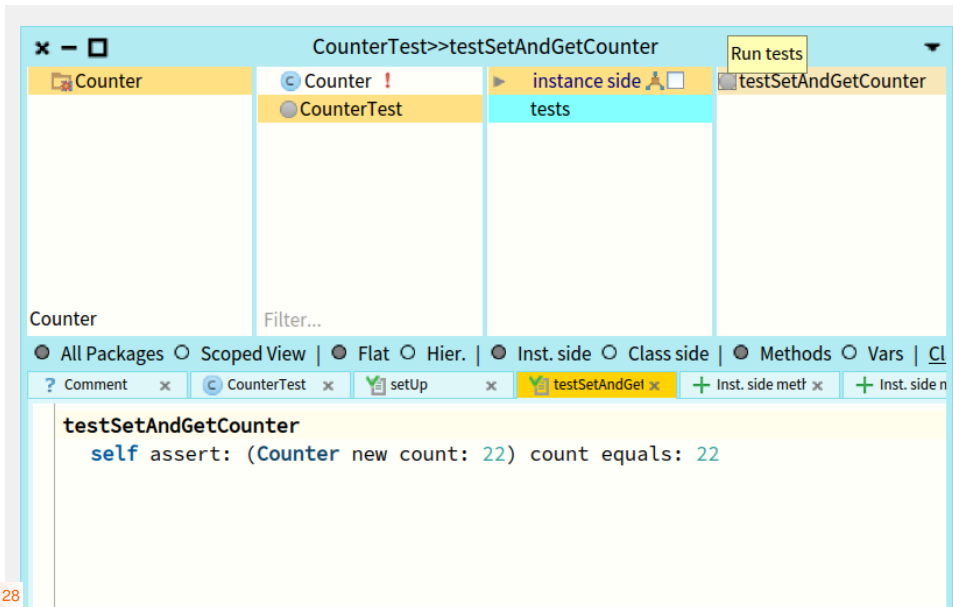


Test defined but not executed

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes. On the left, a "Counter" package is shown. The middle pane displays a tree view with "Counter" (marked with a red exclamation mark) and "CounterTest". The right pane shows a list of tests, with "testSetAndGetCounter" selected. Below the panes, a toolbar contains options for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", "Vars", and "CL". A tab bar at the bottom shows open files: "Comment", "CounterTest", "setUp", "testSetAndGet", "Inst. side mettr", and "Inst. side n". The main editor area displays the code for the "testSetAndGetCounter" test:

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

Running the test



The screenshot shows the IntelliJ IDE interface for running a test. The title bar indicates the current test is `CounterTest>>testSetAndGetCounter`. The interface is divided into several panes:

- Left Pane:** A tree view showing the package structure. `Counter` is expanded, and `CounterTest` is selected.
- Middle Pane:** A list of test methods. `testSetAndGetCounter` is selected and highlighted in yellow.
- Right Pane:** A button labeled `Run tests` is visible.
- Bottom Pane:** The test code is displayed. The method `testSetAndGetCounter` is shown, containing the assertion: `self assert: (Counter new count: 22) count equals: 22`.

The bottom of the IDE shows the `Run` toolbar with various options like `All Packages`, `Scoped View`, `Flat`, `Hier.`, `Inst. side`, `Class side`, `Methods`, and `Vars`. The `testSetAndGetCounter` tab is active in the toolbar.

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
```

VariablesEvaluator

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

Type	Variable	Value
implicit	self	a Counter

Edit the method in the debugger

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
implicit	thisContext	Counter>>count:
implicit	stack top	22

Add an instance variable on the fly




The screenshot shows the IntelliJ IDEA IDE with a runtime error dialog box open. The error message is "Unknown variable: count please correct, or cancel:". The dialog box has three options: "Declare new temporary variable", "Declare new instance variable" (which is highlighted), and "Cancel". Below these options is a text input field and a "Cancel" button. The background shows the IDE's interface with the "Source" tab selected, displaying the following code:







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

Below the code editor is the "Variables" tab, which shows the current state of the program's variables:



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

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

Test is green

The screenshot shows the RubyMine IDE interface. The top toolbar indicates the current view is 'Instance side' and 'Tests'. The left sidebar shows the project structure with 'Counter' and 'CounterTest' (highlighted). The main editor area displays the 'testSetAndGetCounter' test method, which is highlighted in yellow. The test code is as follows:

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

The bottom status bar shows the file path 'M2-4 24 / 28'.

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



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/>