

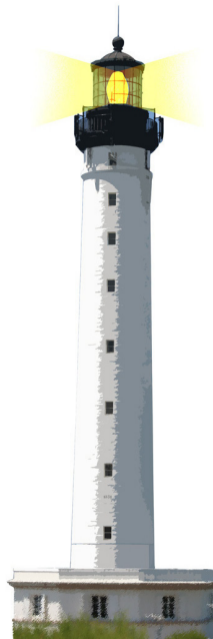


**Learning Object-Oriented
Programming and Design with TDD**

Understanding Mistakes

Stéphane Ducasse

<http://stephane.ducasse.free.fr>



What You Will Learn

Find and fix common mistakes faster

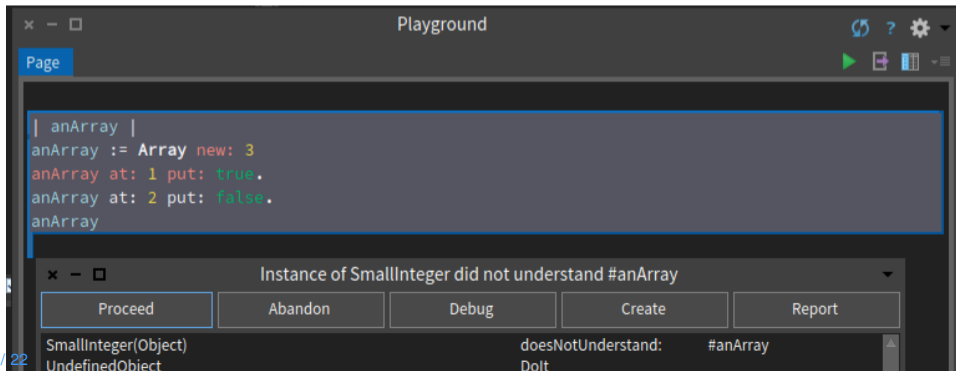
- An expert does mistakes
- But identifies and fixes them fast!



A Problem

```
| anArray |  
anArray := Array new: 3  
anArray at: 1 put: true.  
anArray at: 2 put: false.  
anArray
```

Message: An instance of SmallInteger did not understand anArray



The screenshot shows a REPL window titled "Playground" with the following code:

```
| anArray |  
anArray := Array new: 3  
anArray at: 1 put: true.  
anArray at: 2 put: false.  
anArray
```

Below the code, a message box displays the error: "Instance of SmallInteger did not understand #anArray". The message box contains buttons for "Proceed", "Abandon", "Debug", "Create", and "Report". Below the buttons, the error details are shown:

```
SmallInteger(Object) doesNotUnderstand: #anArray  
UndefinedObject Dolt
```

In the bottom left corner, there is a small lighthouse icon and the text "W2S09 3 / 22".

Missing Period

```
| anArray |  
anArray := Array new: 3.  
anArray at: 1 put: true.  
anArray at: 2 put: false.  
anArray
```

A Problem

```
| aStream |  
aStream := String new: 100 writeStream.  
aStream  
  nextPutAll: 'Today, '  
  nextPutAll: Date today printString;  
  contents
```

Message: An instance of SmallInteger did not understand writeStream



Missing Parenthesis

```
| aStream |  
aStream := (String new: 100) writeStream.  
aStream  
  nextPutAll: 'Today, '  
  nextPutAll: Date today printString;  
  contents
```

A Problem

```
CounterTest >> testCountIsSetAndRead
```

```
|c|
```

```
c := Counter new.
```

```
c count: 7.
```

```
c assert: c count = 7
```

Message: An instance of Counter did not understand assert:

Assert: is a testcase message

```
CounterTest >> testCountIsSetAndRead
```

```
|c|
```

```
c := Counter new.
```

```
c count: 7.
```

```
self assert: c count = 7
```


A Problem

```
CounterTest >> testIncrement  
|c|  
c := Counter new.  
c count: 0 ; increment; increment.  
self assert: self count = 2
```

Message: An instance of CounterTest did not understand count

Identify the object under test API

```
CounterTest >> testIncrement  
|c|  
c := Counter new.  
c count: 0 ; increment; increment.  
self assert: self count = 2
```

Message: An instance of CounterTest did not understand count

A Problem

```
item := Gameltem new  
  title: 'Final Fantasy XII';  
  hasDocumentation: True.
```



True vs. true

```
item := GameItem new  
      title: 'Final Fantasy XII';  
      hasDocumentation: true.
```

- True is the class True
- true is an instance of the class True

A Problem

```
MyExampleSetTest >> testAddTwice
```

```
|s|
```

```
s := Set new.
```

```
self assert: s isEmpty.
```

```
s add: $A.
```

```
(self assert: s size) equals: 1.
```

```
s add: $A.
```

```
(self assert: s size) equals: 1.
```



Assert:equals: is a message

```
MyExampleSetTest >> testAddTwice
|s|
s := Set new.
self assert: s isEmpty.
s add: $A.
self assert: s size equals: 1.
s add: $A.
self assert: s size equals: 1.
```

assert:equals: is a message with two arguments:

- the actual value
- the expected value



Check the argument of a message

Often we pass wrong information as argument

```
testRemoveFrom
| item |
collector := GameCollector smallCollection.
item := (collector collectionNamed: #owned) anyOne.
collector remove: item from: #owned.
self assert: (collector collectionNamed: #owned size) equals: 1
```

Use the debugger to look at the argument

The screenshot shows a debugger window with the title "KeyNotFound: key 5 not found in Dictionary". The "Stack" pane shows the following frames:

- Dictionary at:ifAbsent:
- Dictionary at:
- GameCollector collectionNamed:** (highlighted)
- GameCollectorTest testRemoveFrom

The "Source" pane shows the following code:

```
collectionNamed: aCollectionSymbol  
  ^ collectionDictionary at: aCollectionSymbol .
```

The "Variables" pane shows the following table:

Type	Variable	Value
implicit	self	a GameCollecto
parameter	aCollectionSymbol	5
attribute	collectionDictionary	a Dictionary [1 it

The "Evaluator" pane shows the following code:

```
"5"  
self
```


Check the argument of a message

Often we pass wrong information as argument

```
testRemoveFrom
| item |
collector := GameCollector smallCollection.
item := (collector collectionNamed: #owned) anyOne.
collector remove: item from: #owned.
self assert: (collector collectionNamed: #owned) size equals: 1
```

A problem

TemperatureLogger >> average

| sum |

sum := 0.

measures do: [:aMeasure | sum := sum + aMeasure].

sum / measures size



Missing return

```
TemperatureLogger >> average
```

```
| sum |
```

```
sum := 0.
```

```
measures do: [ :aMeasure | sum := sum + aMeasure ].
```

```
^ sum / measures size
```



What happened?

By default a method always returns the receiver `self`

```
TemperatureLogger >> average  
| sum |  
sum := 0.  
measures do: [ :aMeasure | sum := sum + aMeasure ].  
sum / measures size
```

is equivalent to

```
TemperatureLogger >> average  
| sum |  
sum := 0.  
measures do: [ :aMeasure | sum := sum + aMeasure ].  
sum / measures size.  
^ self
```



What You Should Know

- How to identify common errors faster
- Check periods .
- Check parentheses (and)
- Check carets ^
- Use the debugger to understand the problem



A course by Stéphane Ducasse
<http://stephane.ducasse.free.fr>

Reusing some parts of the Pharo Mocc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
<http://mocc.pharo.org>



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/>