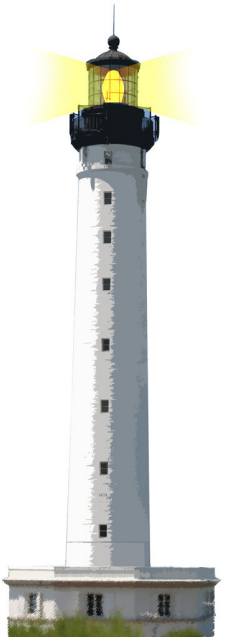
Debugging in Pharo

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W5S05

Pharo

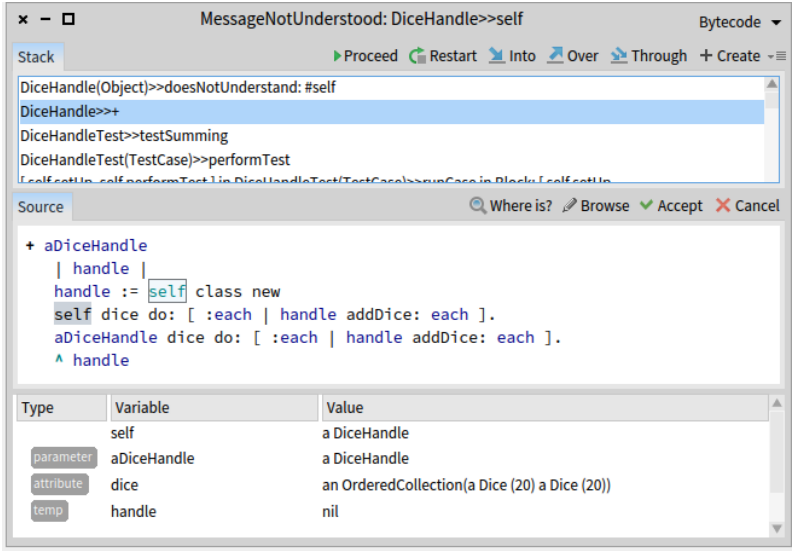
http://www.pharo.org



What You Will Learn

- The system is alive: Communicate with it
- The debugger is your best friend
- Don’t be afraid of it

Debugging


MessageNotUnderstood: DiceHandle>>self
Bytecode
Stack
Proceed
Restart
Into
Over
Through
Create
DiceHandle(Object)>>doesNotUnderstand: #self
DiceHandle>>+
DiceHandleTest>>testSumming
DiceHandleTest(TestCase)>>performTest
Source
Where is?
Browse
Accept
Cancel
+ aDiceHandle
| handle |
handle := self class new
self dice do: [:each | handle addDice: each].
aDiceHandle dice do: [:each | handle addDice: each].
^ handle
Type
Variable
Value
self
a DiceHandle
parameter
aDiceHandle
a DiceHandle
attribute
dice
an OrderedCollection(a Dice (20) a Dice (20))
temp
handle
nil

Debugging

- Closing the debugger does not solve bugs
- The debugger is your best friend
  - communicate with objects of the context
  - check state
  - send messages to specific objects
  - compile code on the fly
  - continue without restarting from scratch

Watch the videos and practice

Simple Trace

```
Transcript show: 'x = ', x printString
```

- used when you don’t have tools
- often inefficient
- we can do better

Defining a Breakpoint

```
...
Halt now.
```

Halt now (or self halt)

- pause the program
- invoke the debugger

Single-Shot Halt

```
...
Halt once.
```

To enable it, evaluate

```
Halt enableHaltOnce
```

Halt once, if enabled :

- pauses the program
- opens a debugger
- disables itself

Halt After n Iterations

```
Halt onCount: 10
```

Conditional Halt

- if: aSelector stops when invoked from a aSelector
- if: aBlock stops if the block evaluates to true

faces will stop only when invoked from printString

```
Dice >> faces
  ...
  Halt if: #printString
```

Conditional Halt

The parameter passed to if: can be a test name too:

```
Dice >> faces
  ...
  Halt if: #testLargeDice
```

faces will stop only when invoked from testLargeDice

Create Your Own Breakpoints

- now, once, onCount: and if: are methods in Halt class
- you can add your own methods, e.g.,

```
Halt class >> between: minTime and: maxTime
  (Time current
    between: minTime asTime
    and: maxTime asTime)
      ifTrue: [ self signal ]

Dice >> faces
  ...
  Halt between: '00:00' and: '02:00'
```

faces will halt only between midnight and 2am.

What You Should Know

- The debugger is a powerful tool
- You should communicate with objects
- Breakpoints are powerful and customizable

A course by

and

in collaboration with