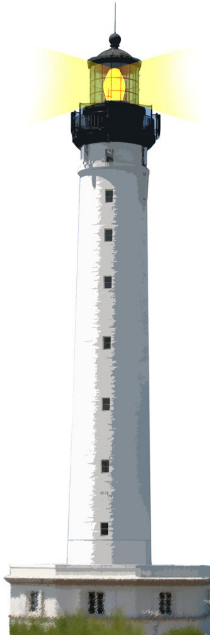
Learning Object-Oriented
Programming and Design with TDD



Common Errors

Stéphane Ducasse
http://stephane.ducasse.free.fr

http://www.pharo.org

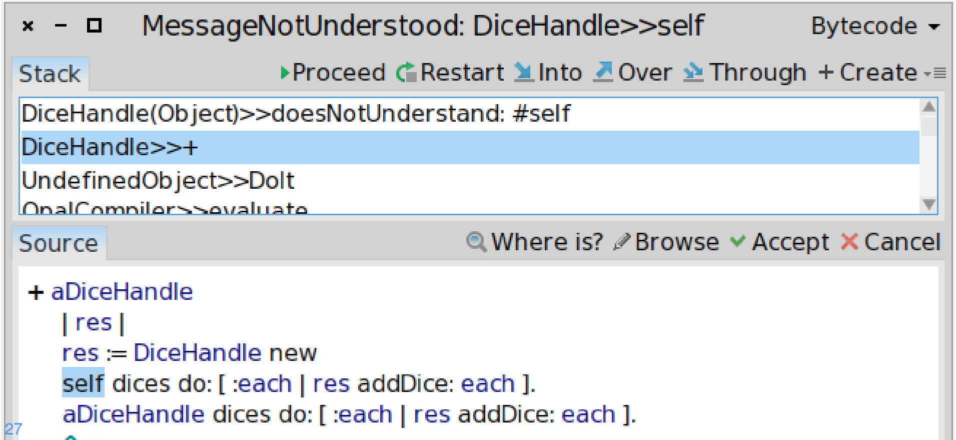
W4S05

What You Will Learn

Find and fix common mistakes faster!

Problem: aDie does not understand self

```
DieHandle >> + aDieHandle
  | res |
  res := DieHandle new
  self dices do: [ :each | ... ].
```



```
DiceHandle(Object)>>doesNotUnderstand: #self
DiceHandle>>+
UndefinedObject>>DoIt
OpalCompiler>>evaluate
```

```
+ aDiceHandle
    | res |
    res := DiceHandle new
    self dices do: [ :each | res addDice: each ].
    aDiceHandle dices do: [ :each | res addDice: each ].
```

Missing Period

```
DieHandle >> + aDieHandle
  | res |
  res := DieHandle new.
  self dices do: [ :each | ... ].
```

- Separate instructions with period (.)

Problem: Message includes:ifTrue: does not exist

```
x includes: 33
  ifTrue: [ self do something ]
```

Error: Message includes:ifTrue: does not exist

Solution: Disambiguate Messages Using Parenthesis

```
(x includes: 33)
  ifTrue: [ self do something ]
```

Problem: Message assert:includes: does not exist

```
self assert: players includes: aPlayer
```

Error: Message assert:includes: does not exist

Keyword-Based Messages

Solution

```
self assert: (players includes: aPlayer)
```

- Keyword-based messages are built out of fragments
- The message is the longuest sequence of fragments
- Use parentheses to delimit multiple keyword messages

Problem: Got an element insted of the collection

```
numbers := OrderedCollection new
        add: 35
```

Error: numbers is the number 35 and not a collection

Forgotten yourself

```
numbers := OrderedCollection new
        add: 35;
        yourself
```

is equivalent to

```
| numbers |
numbers := OrderedCollection new.
numbers add: 35.
numbers
```

Problem: Got 6 instead a Die

```
Die >> setFaces: aNumber
  faces := aNumber

Die class >> new
  ^ super new setFaces: 6
```

Error: Die new returns 6 instead of a dice

Access the receiver of the message with yourself

```
Die class >> new
  ^ super new setFaces: 6; yourself
```

- add: and setFaces: return their argument, not the receiver
- Send yourself after a sequence of messages if you want the receiver

Problem: nil does not understand ifFalse:

```
Book >> borrow
  inLibrary ifFalse: [ ... ].
  ...
```

Error: nil does not understand ifFalse:

Solution: initialize your objects!

```
Book >> initialize
  inLibrary := false
  ...
```

Problem: Booleans vs. Boolean classes

```
Book >> initialize
  inLibrary := True
```

```
Book >> borrow
  inLibrary ifFalse: [ ... ].
 ...
```

Error: Class True does not understand ifFalse:

True vs. true

Solution

```
Book >> initialize
  inLibrary := true
```

- nil is the unique instance of the class UndefinedObject
- true is the unique instance of the class True
- Class names start with an uppercase letter

Problem returns aDie instead of a number

```
Die >> roll
  faces atRandom
```

Error: aDie roll returns aDie instead of a number

Problem Analysis

```
Die >> roll
  faces atRandom
```

is equivalent to

```
Die >> roll
  faces atRandom.
  ^ self
```

Do not forget to return result

```
Die >> roll
  ^ faces atRandom
```

A Problem

```
Die class >> new
  super new
    setFaces: 0;
    yourself
```

Error: Die new returns the class instead of the new instance

Problem Analysis

```
Die class >> new
  super new
    setFaces: 0;
    yourself
```

is equivalent to

```
Die class >> new
  super new
    setFaces: 0;
    yourself.
  ^ self
```

- `new` is sent to a class
- `self` is the class `Die`
- returns `Die` and not its newly created instance

Forgetting to Return the Result

```
Die class >> new
  ^ super new
    setFaces: 0;
    yourself
```

- in a method, self is returned by default
- do not forget the caret ^ to return something else

A Problem

```
Die class >> new
  ^ self new
    setFaces: 0;
    yourself
```

Error: System is frozen

Infinite Loops in Overridden Methods

Solution

```
Die class >> new
  ^ super new
    setFaces: 0;
    yourself
```

- use super in overridden methods

What You Should Know

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

Resources

- Pharo mooc - Videos W5S03: `http://mooc.pharo.org`
- Pharo by Example: `http://books.pharo.org`

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

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
http://mooc.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/