

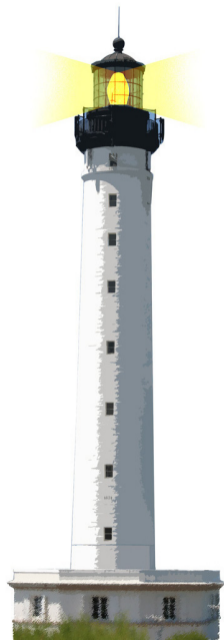
Common Errors

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W5S03



<http://www.pharo.org>



What You Will Learn

Find and fix common mistakes faster



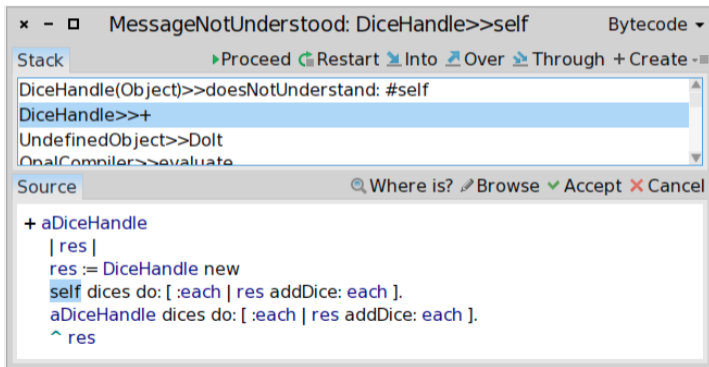
A Problem

```
DiceHandle >> + aDiceHandle
```

```
| res |
```

```
res := DiceHandle new
```

```
self dices do: [ :each | ... ].
```



The screenshot shows an IDE window titled "MessageNotUnderstood: DiceHandle>>self". The "Stack" pane shows the following frames from top to bottom: `DiceHandle(Object)>>doesNotUnderstand: #self`, `DiceHandle>>+` (highlighted), `UndefinedObject>>Dolt`, and `OpalCompiler>>evaluate`. The "Source" pane shows the code for the `+ aDiceHandle` message:

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

Missing Period

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

- Separate instructions with period (.)

A Problem

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

Error: Message includes:ifTrue: 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 messages are built out of fragments
- the message is the longest sequence of fragments
- use parentheses to delimit multiple keyword messages

A Problem

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

Error: numbers is the number 35 and not a collection

```
Dice >> setFaces: aNumber  
^ faces := aNumber
```

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

Error: Dice new returns 6 instead of a dice



Forgotten yourself

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

is equivalent to

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



Forgotten yourself

Solutions

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

```
Dice 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



A Problem

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

Error: nil does not understand ifFalse:

A Problem

```
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



A Problem

```
Dice >> roll  
faces atRandom
```

Error: aDice roll returns aDice instead of a number



A Problem

```
Dice >> roll  
faces atRandom
```

is equivalent to

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

A Problem

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

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



A Problem

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

is equivalent to

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

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



Forgetting to Return the Result

Solutions

```
Dice >> roll  
  ^ faces atRandom
```

```
Dice 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

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

Error: System is frozen



Infinite Loops in Overridden Methods

Solution

```
Dice 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



A course by



and



in collaboration with



Inria 2016

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/>