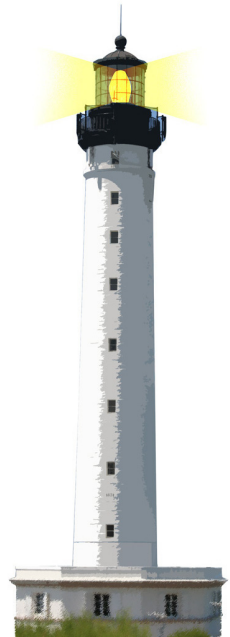
DoesNotUnderstand: a Precious Hook

Damien Cassou, Stéphane Ducasse and Luc Fabresse

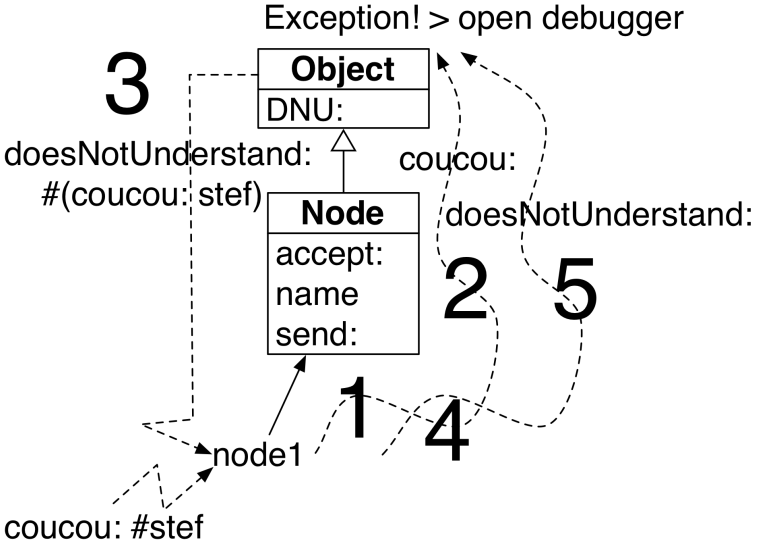
W7S05

Pharo

http://www.pharo.org



When No Method is Found



doesNotUnderstand: is a Message

- doesNotUnderstand: is a message
- Every class can customize this hook
- Important hook for automatic delegation, distributed programming and many other usages

Example 1: Delegation

Redirect unknown messages to another object (a target)

Example 1: Delegation

```
Object subclass: #Delegating
  instanceVariableNames: 'target'
```

```
Delegating >> doesNotUnderstand: aMessage
  ^ aMessage sendTo: self target
```

- Pay attention it blurs code understandability
- Only tools and specific parts should use such tricks

Example 2: A LoggingProxy

Basic idea:

- Create a 'minimal' object raising error to most messages
- Customize its doesNotUnderstand: method
- Swap an object and the proxy

Implementing LoggingProxy

```
ProtoObject subclass: #LoggingProxy
  instanceVariableNames: 'subject invocationCount'
  classVariableNames: ''
  package: 'LoggingProxy'
```

```
LoggingProxy >> initialize
  invocationCount := 0.
  subject := self
  "will be swapped by become:"
```

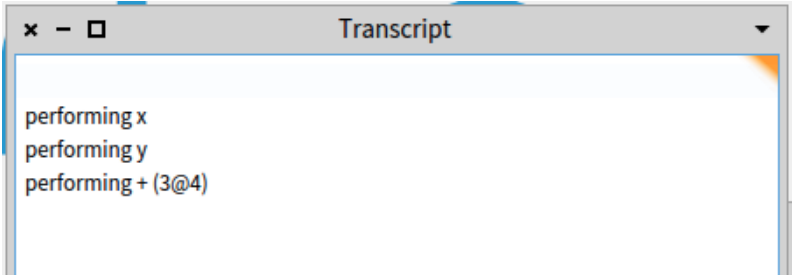
Customize doesNotUnderstand:

```
LoggingProxy >> doesNotUnderstand: aMessage
  Transcript show: 'performing ', aMessage printString; cr.
  invocationCount := invocationCount + 1.
  ^ aMessage sendTo: subject
```

```
Message >> sendTo: receiver
  ^ receiver perform: selector withArguments: args
```

Message Behavior

```
| point |
point := 1@2.
LoggingProxy new become: point.
self assert: point invocationCount = 0.
self assert: point x + point y = 3.
self assert: point + (3@4) = (4@6).
self assert: point invocationCount = 3.
```



Some Limits of such Minimal Objects

- Messages sent by the object to itself are not trapped!
- Class cannot be swapped
- What to do with messages that are understood by both the minimalObject and its subject?

Another Application

Scaffolding patterns: Generate code on the fly based of patterns

```
DynamicAccessors >> doesNotUnderstand: aMessage
  | messageName |
  messageName := aMessage selector asString.
  (self class instVarNames includes: messageName)
    ifTrue: [self class compile:
          messageName , String cr , ' ^ ' , messageName.
      ^ aMessage sendTo: self].
  super doesNotUnderstand: aMessage
```

Conclusion

Minimal Objects

- Basis for proxies
- Multiple usages (distribution, object loading, spying)

doesNotUnderstand:

- Powerful hook
- Only to be used when needed

A course by

and

in collaboration with