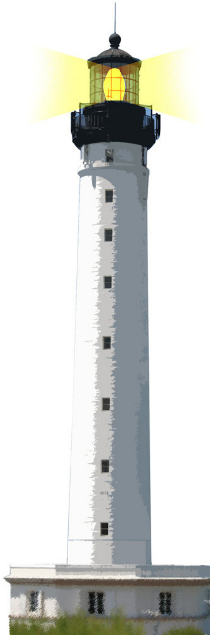
Advanced Object-Oriented Design

Global To Parameter

Basic but important

S. Ducasse



Pharo

http://www.pharo.org

Goal

- Globals are not a fatality
- Some can be turned as parameters (instance variables...)
- Pros and cons

Roadmap

- Analysis of Transcript usage
- Cure
- Analysis
- Related to Singleton Design Pattern Plague

The case: Transcript

```
myMethod
  Transcript show: 'foo' ; cr.
  self doSomething.
```

- Remember: Transcript is a global variable pointing to a log stream instance

Example in Microdown

If logging is part of your domain

```
MicAbstractBlock >> iterate
  ...
  Transcript
    nextPutAll: 'Start ';
    nextPutAll: step asString;
    cr.
  ...
  Transcript
    nextPutAll: 'Stop ';
    nextPutAll: step asString;
    cr.
```

- What if I would like to have a personal log for Microdown?
- What if we want to test that such logs are correct?

Analysis

- You do not want to get extra dependencies in your code
- Your log can be mixed with other logs
- You do not want to dirty build log without control

Far worse and more important

- You cannot reliably write tests to be sure that the log is correctly happening

The solution: Use locality and encapsulation

- Think self-contained
- Add an instance variable to hold a stream

```
MicAbstractBlock >>initialize
  super initialize.
  logStream := WriteStream on: (String new: 1000)
```

Write to THAT stream

```
MicAbstractBlock >>closeMe
  logStream << 'Closing ' << self class name; cr
```

Get the butter and the money

- Make sure that you can plug another stream to your stream

```
MicAbstractBlock >> logStream: aStream
  logStream := aStream
```

- Now you can pass a Transcript and get the same as before but better.
- Bonus: You can write tests in isolation

About Globals

Pros:

- You do not have to add an instance variable to you domain
- You do not have to initialize such global on your specific case

Cons:

- You have only one (e.g., if an entity belongs to one global model, you cannot have two models)
- Testing requires care and is sometimes not possible because of side effect
- You cannot initialize it for your context (there is only one)

About parametrization

Sometimes you cannot add an instance variable to your objects

- Too many of them
- Fixed size inherited from old design

At least factor the global usage to ease future changes

In general: Avoid Globals

- Avoid Singleton
- Avoid Globals
- They make your code less modular, less **
- Related to Singleton and Disguised Singleton Lecture

A course by

S. Ducasse, L. Fabresse, G. Polito, and Pablo Tesone