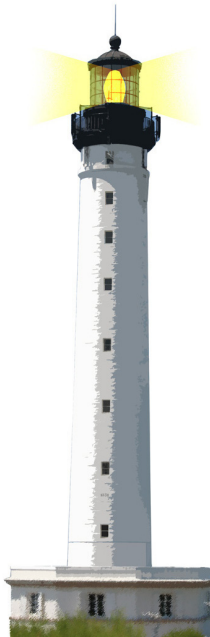
Learning Object-Oriented
Programming and Design with TDD

Advanced Points on Class

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



Pharo
http://www.pharo.org

W7S01

Roadmap

- Sharing state between instances of a class
- Instance variables of classes
- Class initialization

Roadmap

- **Sharing state between instances of a class**
- Instance variables of classes
- Class initialization

Sharing State?

How do you share state between instances of a class?

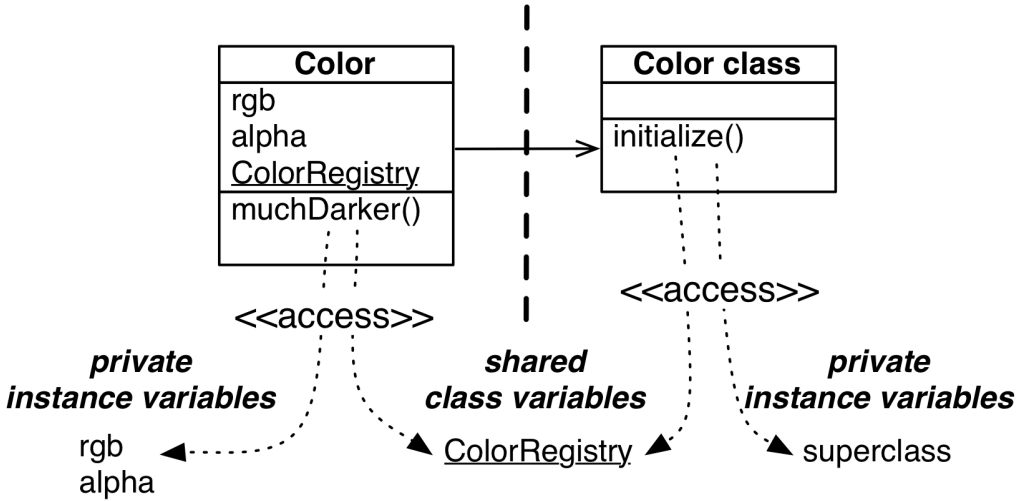
- in Java, an "instance" variable can be static
- in Pharo, we use **class variables**

```
Object subclass: #Color
  instanceVariableNames: 'rgb cachedDepth...'
  classVariableNames: 'ColorRegistry ComponentMask...'
  package: 'Graphics−Primitives'
```

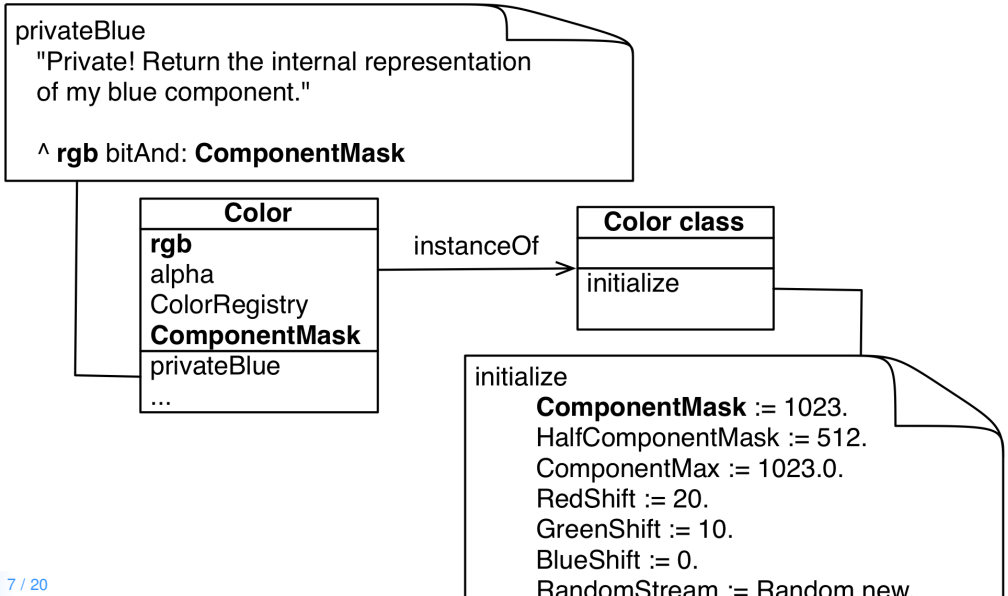
Class Variables

- shared by the instances of the class and subclasses
- accessible from instance and class methods
- start with an uppercase letter

Class Variable Access



Class Variable Access



Roadmap

- Sharing state between instances of a class
- **Instance variables of classes**
- Class initialization

Class Instance Variables

A class can have instance variables like any object

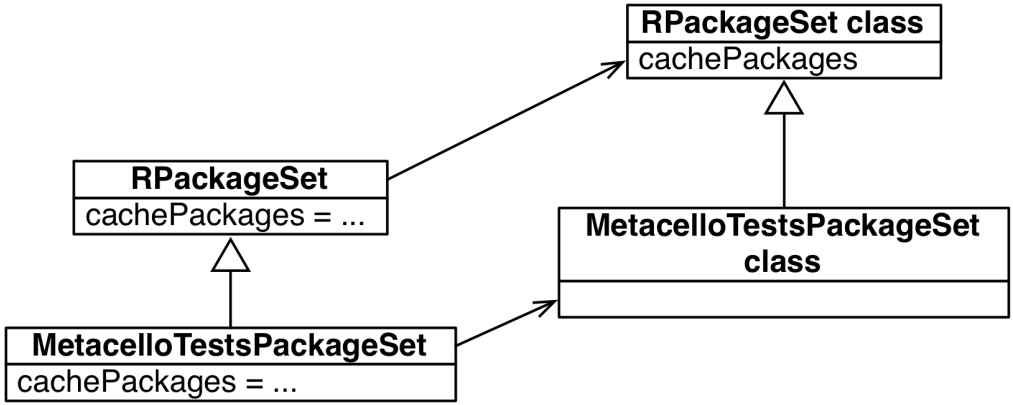
- a class is an instance of a class (its metaclass)
- a metaclass can specify **class instance variables**

```
RPackageSet class
  instanceVariableNames: 'cachePackages'
```

- accessible from class methods only
- start with a lowercase letter

No Sharing with Class Instance Variables

Each instance has a different value for cachePackages



Singleton Design Pattern

- Intent: Enforce that a class has only one instance
- A solution: Keep the instance in a variable of the class

Singleton with a Class Instance Variable

```
WebServer class
  instanceVariableNames: 'uniqueInstance'

WebServer class >> new
  self error: 'Can''t create a new instance'

WebServer class >> uniqueInstance
  ^ uniqueInstance
    ifNil: [ uniqueInstance := super new ]
```

Consequence:

- each subclass has its own value for uniqueInstance
  - each subclass of WebServer has its own singleton

Singleton with a Class Variable

```
Object subclass: #WebServer
  instanceVariableNames: ''
  classVariableNames: 'UniqueInstance'
  package: 'Web'

WebServer class >> new
  self error: 'Can''t create a new instance'

WebServer class >> uniqueInstance
  ^ UniqueInstance
    ifNil: [ UniqueInstance := super new ]
```

Consequence:

- only one singleton for the complete class hierarchy
  - class variable values are shared

Roadmap

- Sharing state between instances of a class
- Instance variables of classes
- **Class initialization**

Class Initialization

- Everything is an object
- An object is initialized at creation time
- Classes are objects too

How and when are classes initialized?

Class Initialization

A class is initialized

- at load time after its methods are loaded
- or explicitly by the programmer:

```
Color initialize
```

Color Initialization

```
Color class >> initialize
  "Externally, the red, green, and blue components of color are floats in the range
      [0.0..1.0]. Internally, they are represented as integers in the range [0..
      ComponentMask] packing into a small integer to save space and to allow fast
      hashing and equality testing."
  ComponentMask := 1023.
  HalfComponentMask := 512. "used to round up in integer calculations"
  ComponentMax := 1023.0. "used to normalize components"
  RedShift := 20.
  GreenShift := 10.
  BlueShift := 0.
  self initializeIndexedColors.
  self initializeColorRegistry.
  self initializeGrayToIndexMap.
```

Warning

- don't write super initialize in a class initialize method
  - this will initialize superclasses that are already initialized

What You Should Know

- state is shared between instances through class variables
- a class can store values in class instance variables
- a class is initialized through the class method initialize

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