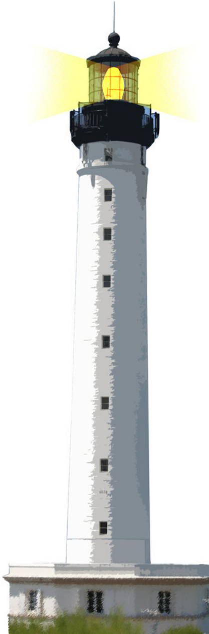
Advanced Object-Oriented Design

Shared Variables

S. Ducasse



http://www.pharo.org

Goal

- Revisit Shared Variables (e.g., ClassVariables in Smalltalk jargon)
- Think about scope of sharing

Instance variables are local to one object

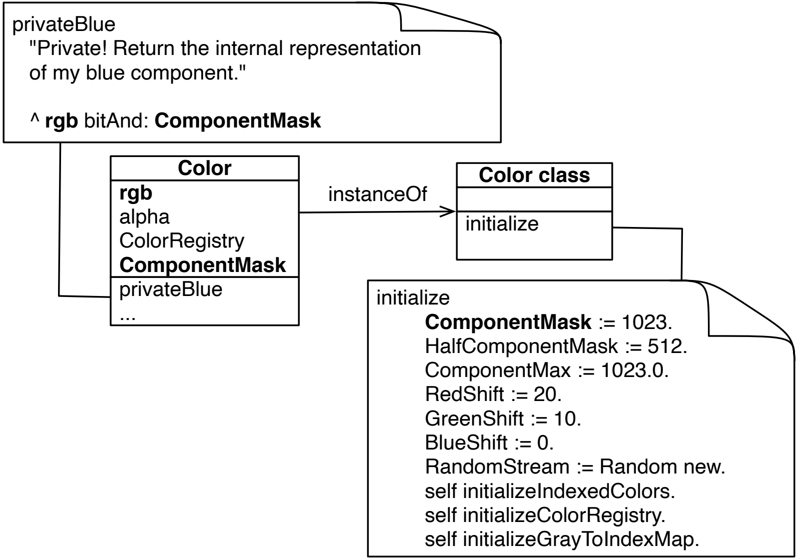
- An instance variable value is only accessible to the object
- If you modify an instance variable, you only modify that variable

No news!

Shared variables are shared by all the instances of a hierarchy

- All the instances of a class and its subclasses share the SAME shared variable
- If you modify a shared variables it impacts all the instances
- A shared variable is usually initialized at class load time (class initialize method)
- In Smalltalks, shared variables are called ClassVariables.

Color’s ComponentMask is a shared variable



```
privateBlue
   "Private! Return the internal representation
   of my blue component."

   ^ rgb bitAnd: ComponentMask
```

Color

rgb
alpha
ColorRegistry
ComponentMask

privateBlue
...

instanceOf

Color class

initialize

```
initialize
        ComponentMask := 1023.
        HalfComponentMask := 512.
        ComponentMax := 1023.0.
        RedShift := 20.
        GreenShift := 10.
        BlueShift := 0.
        RandomStream := Random new.
        self initializeIndexedColors.
        self initializeColorRegistry.
        self initializeGrayToIndexMap.
```

Color example

All the instances of Color and its subclasses share ComponentMask

```
Object << #Color
  slots: { #rgb . #cachedDepth . #cachedBitPattern . #alpha };
  sharedVariables: { #RedShift . #CachedColormaps . #IndexedColors .
  #ComponentMax . #ComponentMask . #ColorRegistry . #GreenShift . #BlueShift };
  package: 'Colors'
```

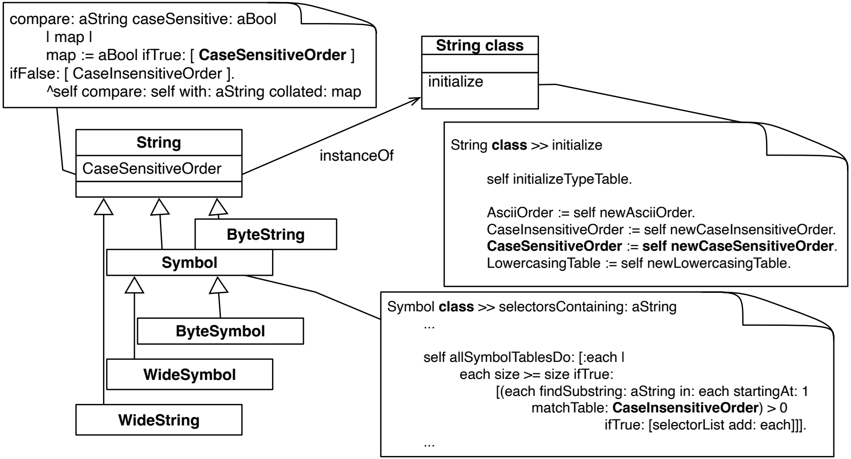
Shared variables are accessible from both instance and class methods

```
Color >> setRed: r green: g blue: b
  "Initialize this color's r, g, and b components to the given values in the range [0.0..1.0].
  Encoded in a single variable as 3 integers in [0..1023]."

  rgb == nil ifFalse: [ self attemptToMutateError ].
  rgb := (((r * ComponentMax) rounded bitAnd: ComponentMask) bitShift: RedShift)
     + (((g * ComponentMax) rounded bitAnd: ComponentMask) bitShift: GreenShift)
     + ((b * ComponentMax) rounded bitAnd: ComponentMask).
  cachedDepth := nil.
  cachedBitPattern := nil
```

Shared variables are accessible from both instance and class methods

```
Color class >> initialize

  ComponentMask := 1023.
  HalfComponentMask := 512.  "used to round up in integer calculations"
  ComponentMax := 1023.0. "a Float used to normalize components"
  RedShift := 20.
  GreenShift := 10.
  BlueShift := 0.
  self initializeIndexedColors.
  self initializeColorRegistry.
  self initializeGrayToIndexMap.
```

Shared variable's example: String



```
compare: aString caseSensitive: aBool
        | map |
        map := aBool ifTrue: [ CaseSensitiveOrder ]
ifFalse: [ CaseInsensitiveOrder ].
        ^self compare: self with: aString collated: map
```

```
String class >> initialize

        self initializeTypeTable.

        AsciiOrder := self newAsciiOrder.
        CaseInsensitiveOrder := self newCaseInsensitiveOrder.
        CaseSensitiveOrder := self newCaseSensitiveOrder.
        LowercasingTable := self newLowercasingTable.
```

```
Symbol class >> selectorsContaining: aString
        ...

        self allSymbolTablesDo: [:each |
                each size >= size ifTrue:
                        [(each findSubstring: aString in: each startingAt: 1
                                matchTable: CaseInsensitiveOrder) > 0
                                                ifTrue: [selectorList add: each]]].
        ...
```

Shared Variables of String

```
ArrayedCollection << #String
  slots: {};
  sharedVariables: { #CaseSensitiveOrder . #CSSeparators . #CSNonSeparators .
      #UppercasingTable .
  #CSLineEnders . #LowercasingTable . #CaseInsensitiveOrder . #TypeTable . #Tokenish .
       #AsciiOrder };
  package: 'Collections–Strings'
```

Shared variable CaseSensitiveOrder accessed in subclass method

```
ByteSymbol >> beginsWith: prefix
  "Answer whether the receiver begins with the given prefix string.
  The comparison is case—sensitive."

  "(#pharo beginsWith: #pharoProject) >>> false"
  "(#pharo beginsWith: #phuro) >>> false"
  "(#pharo beginsWith: #pha) >>> true"

  prefix class isBytes ifFalse: [^super beginsWith: prefix].

  self size < prefix size ifTrue: [^ false].
  ^ (self findSubstring: prefix in: self startingAt: 1
      matchTable: CaseSensitiveOrder) = 1
```

Shared variable accessed in subclass class method

```
Symbol class >> selectorsContaining: aString
  "Answer a list of selectors that contain aString within them. Case—insensitive.
  Does return symbols that begin with a capital letter."


  ...


  self allSymbolTablesDo: [:each |
    each size >= size ifTrue:
      [(each findSubstring: aString in: each startingAt: 1
        matchTable: CaseInsensitiveOrder) > 0
            ifTrue: [selectorList add: each]]].
  ...
```

Investigating...

```
Smalltalk globals allClasses
  select: [:each | each classVariablesString isEmpty not
        and: [ each hasSubclasses ]]
```

Implications

- There is a difference between Shared variables and instance variable of the metaclass
- There is a difference between:

```
Object << #BorderStyle
  sharedVariables: { #Default };
  package: 'Morphic–Core'
```

and

```
BorderStyle class
  slots: {#default};
  package: 'Morphic–Core'
```

Implications: One for all

```
Object << #BorderStyle
  sharedVariables: { #Default };
  package: 'Morphic–Core'
```

There is only one instance of BorderStyle for all the subclasses: SimpleBorderStyle BottomBorderStyle ComplexBorderStyle ...

Implications: One for each

```
BorderStyle class
  slots: {#default};
  package: 'Morphic−Core'
```

There is one instance for EACH of all the subclasses (potentially the same depending on the creation logic)

Conclusion

- Pay attention modifying shared variables potentially impacts many objects.
- Can be used to support different sharing optimization (see other lectures)

A course by

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