Shared Pools

Static sharing between hierarchies

S. Ducasse, L. Fabresse, G. Polito, and P. Tesone
Goals

- Revisit sharing
- Understand shared pools *(SharedPools)*

A question:

- Using shared variables, we can share values over multiple subclasses within the **same** hierarchy.
- How can we share objects between **different** hierarchies?
Remember: Sharing within a hierarchy

A shared variable can be accessed from

- Instance methods
- Class methods of the class defining it
- From its subclasses

Usually initialized from the class side of a root
privateBlue
"Private! Return the internal representation
of my blue component."

\^ rgb bitAnd: ComponentMask

Color
rgb
alpha
ColorRegistry
ComponentMask
privateBlue
...

initialize

Color class
instanceOf

initialize

ComponentMask := 1023.
HalfComponentMask := 512.
ComponentMax := 1023.0.
GreenShift := 10.
BlueShift := 0.
RandomStream := Random new.
self initializeIndexedColors.
self initializeColorRegistry.
self initializeGrayToIndexMap.
Need for sharing between different hierarchies

- Need to share values (generally constants) between **multiple** hierarchies:
  - For example LF, CR, ... between the hierarchies of String and Text
- Don’t want to repeat the shared variables and their initialization
A SharedPool is a **group of shared variables** contains

- the shared pools definition
- the initialization of shared variables

Users (classes) just declare that they use a shared pool to access its shared variables
A SharedPool definition

```
SharedPool << #ChronologyConstants
slots: {};
sharedVariables: { #NanosInSecond . #MonthNames . #SecondsInHour .
    #SecondsInDay . #DayNames . #DaysInMonth . #HoursInDay . #NanosInMillisecond .
    #SecondsInMinute . #SqueakEpoch . #MinutesInHour . #MicrosecondsInDay };
tag: 'Chronology';
package: 'Kernel'
```
A SharedPool initialization

ChronologyConstants class >> initialize

SqueakEpoch := 2415386. "Julian day number of 1 Jan 1901"
SecondsInDay := 86400.
MicrosecondsInDay := SecondsInDay * 1e6.
SecondsInHour := 3600.
SecondsInMinute := 60.
MinutesInHour := 60.
HoursInDay := 24.
NanosInSecond := 10 raisedTo: 9.
NanosInMillisecond := 10 raisedTo: 6.
DayNames := #(Sunday Monday Tuesday Wednesday Thursday Friday Saturday).
MonthNames := #(January February March April May June July August September October November December).
DaysInMonth := #(31 28 31 30 31 30 31 31 30 31 30 31).

Shared pools are initialized at class load time.
SharedPool users

Magnitude «<< #DateAndTime
  slots: { #seconds . #offset . #julianDayNumber . #nanos };
  sharedVariables: { #ClockProvider . #LocalTimeZoneCache };
  sharedPools: { ChronologyConstants };
  package: 'Kernel'

DateAndTime

- defines some shared variables
- uses the shared pool ChronologyConstants
SharedPool’s sharedVariable access

A shared variable defined in a shared pool is accessed as if defined in the class itself.

```smalltalk
DateAndTime >> secondsSinceMidnightLocalTime
^ self localSeconds \ SecondsInDay
```

```smalltalk
Duration class >> days: aNumber
^ self seconds: aNumber * SecondsInDay nanoSeconds: 0
```

SecondsInDay is just accessed directly both from instance and class side.
SharedPool users (2)

```plaintext
Timespan << #Week
slots: {};
sharedVariables: { #StartDay };  
sharedPools: { ChronologyConstants };  
package: 'Kernel-Chronology-Extras'

Week class >> indexOfDay: aSymbol
  ^ DayNames indexOf: aSymbol
```
Mixing shared variables and sharedPools

There is no problem mixing shared variables and shared pools

Timespan <= #Week
  sharedVariables: { #StartDay };
  sharedPools: { ChronologyConstants };
  package: 'Kernel−Chronology−Extras'

Week class >> startDay
  ^ StartDay ifNil: [ StartDay := DayNames first ]
Warning! Only for constants

- Should only store `constant` objects in shared pools
- Else you are creating global variables and you are breaking testability in isolation
Conclusion

Shared pools are:

- Handy to **share constants** between multiple classes (potentially in different inheritance trees)
- Handy to manage constants for bindings to C-libraries
- **Only** use them to share constants
Advanced Object-Oriented Design and Development with Pharo

A course by
S.Ducasse, L. Fabresse, G. Polito, and P. Tesone

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/