

A variation on sharing

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



Remember

We saw:

- Shared variables to share info between **all instances** of a class and instances of subclasses
- Mixing an instance variable and a shared variable: **sharing by default and instance-based** customization
- Flyweighth

Here is another variation on that theme taken from Bloc graphical framework



BElement

BElement is the basic graphical element

- It has many properties:
- background, border, clipChildren, elevation, geometry, compositingMode, effect, focusability, focused, mouseCursor, opacity, outskirts, visibility



BLElement property example

Properties are managed via an instance of BLElementVisuals

```
BLElement >> border  
^ visuals border
```

```
BLElement >> clipChildren  
^ visuals clipChildren
```

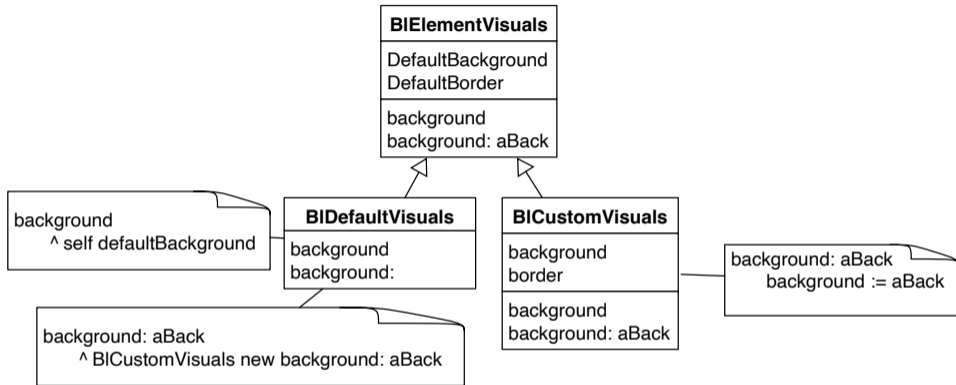


The objectives

- Default visuals are shared
- A property can be modified
- How to support property modification **without paying** an instance variable for all the shared default?



Overview



- Make sure that many default values are shared by default
- Modifications to these defaults is possible on a per instance level
- **But** without one instance variable per property

BlElementVisuals

BlElementVisuals defines API and default values

```
Object << #BlElementVisuals
  sharedVariables: { #DefaultBorder . #DefaultBackground . #DefaultGeometry .
    #DefaultVisibility };
  package: 'Bloc'
```

```
BlElementVisuals >> defaultBackground
  ^ DefaultBackground
```

```
BlElementVisuals >> background: aBlBackground
  ^ self subclassResponsibility
```



BlDefaultVisuals

- A subclass of BlElementVisuals
- BlDefaultVisuals a kind of singleton that holds many default values to be shared between multiples elements.

```
BlElement >> initialize
```

```
...
```

```
visuals := BlDefaultVisuals uniqueInstance.
```

```
...
```



BlDefaultVisuals getters

Getters access default shared values

```
BlDefaultVisuals >> background  
^ self defaultBackground
```

```
BlDefaultVisuals >> defaultBackground  
^ DefaultBackground
```



BlDefaultVisuals: setters are key

- BlDefaultVisuals is kind of read only, when setters are executed,
- they do not modify but create and return a **new** instance of BlCustomVisuals

BlDefaultVisuals >> background: aBlBackground

"Change the background and return new visuals to be used instead of previous one"

^ BlCustomVisuals new background: aBlBackground



BlCustomVisuals

Support for instance specific property modification

```
BlElementVisuals << #BlCustomVisuals  
  slots: { #geometry . #border . #background . #outskirts . #effect ... };  
  package: 'Bloc'
```

```
BlCustomVisuals >> background: aBlBackground  
  background := aBlBackground
```

- BlCustomVisuals stored in place of BlDefaultVisuals singleton to keep modifications
- BlCustomVisuals accumulates modifications because contrary to BlDefaultVisuals its setters modify the receiver



There is a catch - Property modification

Users should always store the result of the setters sent to a visuals

```
BlElement >> background: aBlBackground
```

```
"Change my background to a given one.
```

```
Raises BlElementBackgroundChangedEvent."
```

```
...
```

```
visuals := visuals background: aNewBackground.
```

```
...
```

- It is not really nice to hijack setter semantics this way



What is the difference with the TypeTable/typeTable

- Group different values in a single object
- Avoid to have one instance variable per customisation point
- But still we have instance-based and sharing



Analysis/Conclusion

- Is all the complexity needed?
 - Hijack default setter patterns
- Requires some memory analysis:
 - empty instance variables per instance that shared a default
- How many objects?



Produced as part of the course on <http://www.fun-mooc.fr>

Advanced Object-Oriented Design and Development with Pharo

A course by

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



Inria
LearningLab



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/>