Elements of Design - Inheritance/Composition

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

A Formating Text Editor

With several possible algorithms

formatWithTex
formatFastColoring
formatSlowButPreciseColoring

Code Smells

```
Composition>>repair
    formatting == #Simple
        ifTrue: [ self formatWithSimpleAlgo]
        ifFalse: [ formatting == #Tex
                    ifTrue: [self formatWithTex]

                ....]
```

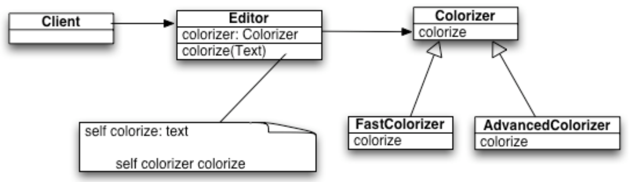
Inheritance?

May not be the solution since:
- you have to create objects of the right class
- it is difficult to change the policy at run-time
- you can get an explosion of classes bloated with the use of a functionality and the functionalities.
- no clear identification of responsibility

Inheritance vs. Composition

- Inheritance is not a panacea
  - Require class definition
  - Require method definition
  - Extension should be prepared in advance
  - No run-time changes
- Ex: editor with spell-checkerS, colorizerS, mail-readerS….
  - No clear responsibility
  - Code bloated
  - Cannot load a new colorizers

Delegating to other Objects



- myEditor setColorizer: FastColorizer new.
- myEditor setColorizer: AdvancedColorizer new.
- Strategy design pattern

Composition Analysis

- Pros
  - Possibility to change at run-time
  - Clear responsibility
  - No blob
  - Clear interaction protocol
- Cons
  - New class
  - Delegation
  - New classes