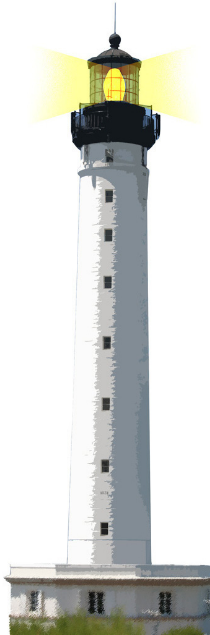
Learning Object-Oriented
Programming and Design with TDD

Use vs. Inheritance

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

http://www.pharo.org

W6S06



Outline

- An exercise
- Some criteria
- Solutions
- Comparing solutions

Exercise Set Up

Imagine the TextEditor class and several algorithms

- formatWithTex (t TextEditor)
- formatFastColoring (t TextEditor)
- formatSlowButPreciseColoring (t TextEditor)

How can we create an editor that will format differently texts?

Next step

- Propose one solution
- Propose different solutions
- Define some criteria
- Compare the approaches using such criteria

Solution 1: Inheritance

```
TextEditor subclass: #SlowFormatingTextEditor

SlowFormatingTextEditor >> format
    self formatSlowButPreciseColoring: text
```

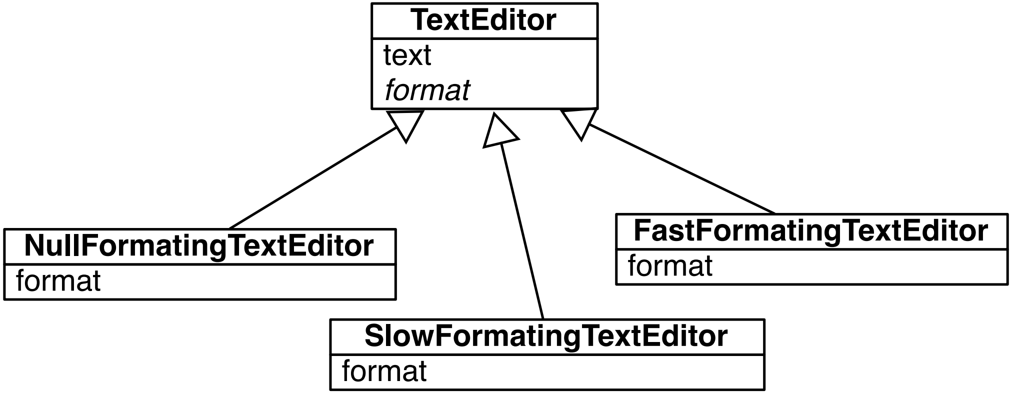
```
TextEditor subclass: #FastFormatingTextEditor

SlowFormatingTextEditor >> format
    self formatFastColoring: text
```

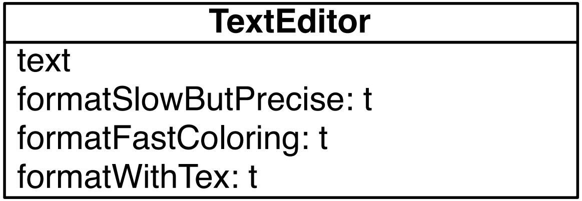
```
TextEditor subclass: #NullFormatingTextEditor

NullFormatingTextEditor >> format
   ^ self "do nothing"
```

Inheritance



Solution 2: With conditionals



```
TextEditor >> format
    currentSelection = #slow
      ifTrue: [ self formatSlowButPreciseColoring: text]
      ifFalse: [ currentSelection = #fast
        ifTrue: [self formatFastColoring: text]
        ....]
```

Alternate with Registry and Meta Programming

```
Object subclass: #TextEditor
    currentSelection formatters text
```

```
TextEditor class >> initialize
    self formatters
      at: #slow put: #slowFormat: ;
      at: #fast put: #fastFormat: ;
      at: #null put: #nullFormat: ;
      at: #tex put: #texFormat:
```

```
TextEditor >> format
    self perform: (formatters at: currentSelection) with: text
```

What are your criterias to compare these and other solutions?

Criteria

- Yes what are they?

Criteria

- **Adding a new formatting algo**
  - what is the cost to define a new formatting algorithm
- **Dynamically use a formatter**
  - can I switch dynamically to a new formatting algorithm
- **Packaging**
  - can I deploy a new formatting algorithm separately from others

Analysing Solution 1: Inheritance?

Adding a new formatting algo:

- we can add a new formatter

Packaging:

- we can package a new formatter

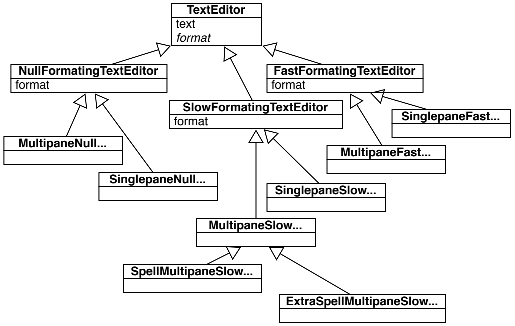
Not the best solution since:

- you have to create objects of the right class
- it is difficult to change the policy at run-time.
    - we do not want to have and reopen the texteditor

Analysing Solution 1: Inheritance?

You can get an explosion of classes bloated with functionalities

- we do not want a hierarchy for each text editor features to be **multiplied** with previous ones
- TextEditor API can get large: no clear identification of responsibility



Analysing Solution 2: Conditionals

Dynamic use: we can use a different formatter dynamically
Adding a new formatting algo:

- adding a version requires to edit and **recompile** the conditionals

Packaging:

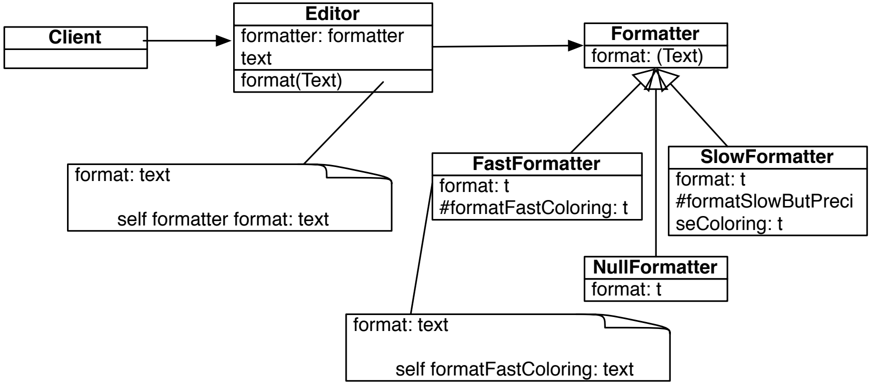
- we cannot package a new algorithm separately

Another solution...

Delegating to a formatter

- Sketch the solution

Delegating to a formatter



```
myEditor formatter: FastFormatter new.
myEditor format.
myEditor formatter: SlowFormatter new.
```

Delegating to a formatter

Dynamic use:

- we can use a different formatter dynamically. Just create a new instance and set it.

Adding a new formatting algo:

- adding a version is just adding a new class

Packaging:

- we package a new algorithm separately

Strategy Design Pattern

- Uniformize the communication (API) between the Editor and the Formatter
  - all formatters should understand `format:`
- Modular
- Incremental

But there is nothing like a free lunch

- The formatter should access the state (i.e. the text, positions... contained in the text editor)
- Information should flow between the textEditor and the formatter
- API of textEditor should be opened to support it

Conclusion

Inheritance

- is about incremental static definition
- It can lead of static design
- It can help
  - build dynamic solutions
  - structure abstractions
- It supports late binding

Delegation (Use)

- can bring runtime flexibility
- can be combined with inheritance

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