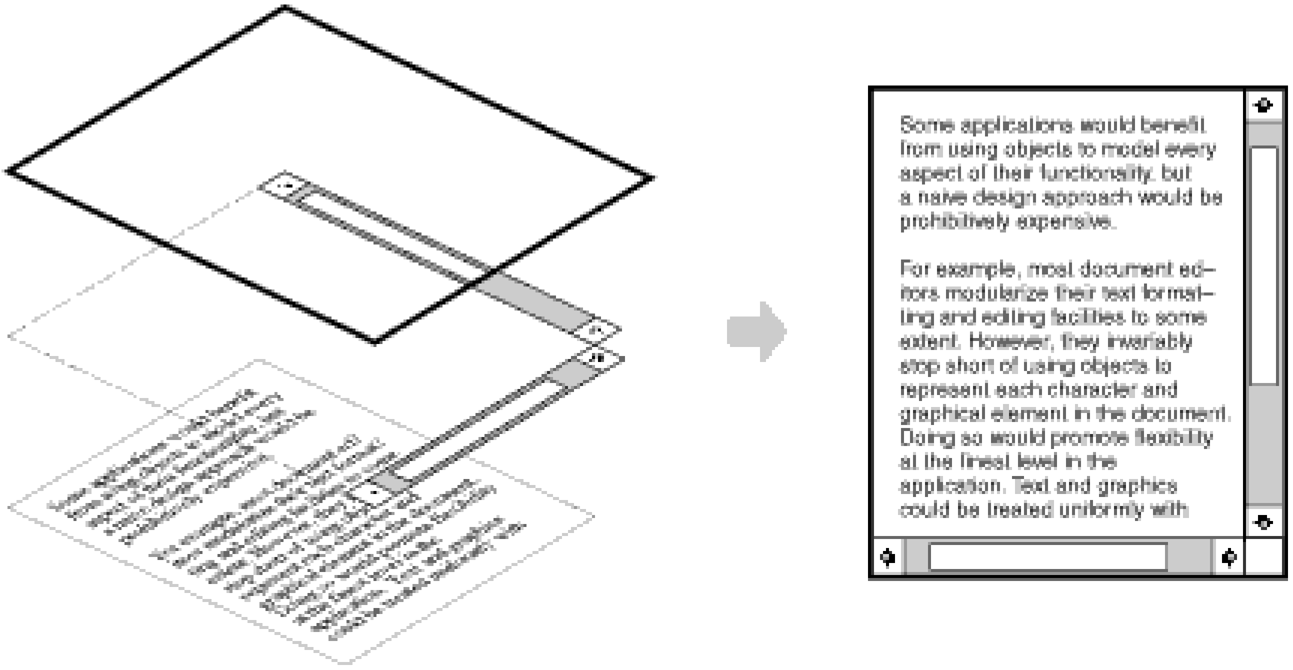
Design Issue #5

Embellishing the user interface

- Two ornaments (embellishments):
  - a border around the text editing area
  - scroll bars



Some applications would benefit
from using objects to model every
aspect of their functionality, but
a naive design approach would be
prohibitively expensive.

For example, most document ed–
itors modularize their text format–
ting and editing facilities to some
extent. However, they invariably
stop short of using objects to
represent each character and
graphical element in the document.
Doing so would promote flexibility
at the finest level in the
application. Text and graphics
could be treated uniformly with

Design Issue #5

Embellishing the user interface

Team exercise:

- One approach: using inheritance
- Second approach: using object composition

Design Issue #5

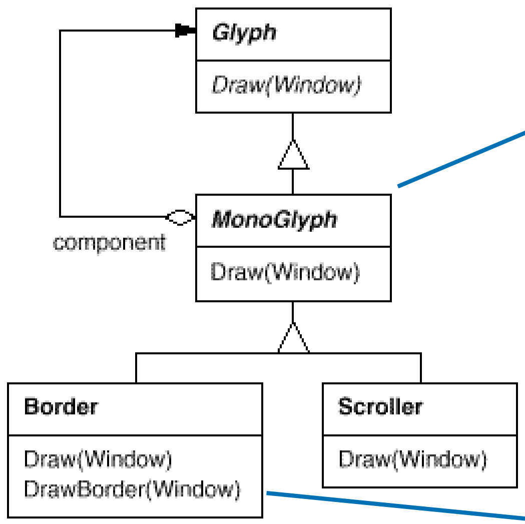
Embellishing the User Interface

- Basically, we want to extend the code to provide a Transparent Enclosure
  - Transparent in that the page itself does not know anything about the changes – it behaves the same
- How should we do this?
  - We could use Inheritance, how would that look?
  - We have a Composition class…
    - To add a Border we add a BorderedComposition class
    - To add a Scroll bar we add a ScrollableComposition class
    - What about both? BorderedScrollableComposition class?
- How could we do it with object composition instead?
  - What object “has” what object?
  - How do we make it extensible?

Design Issue #5

Embellishing the User Interface

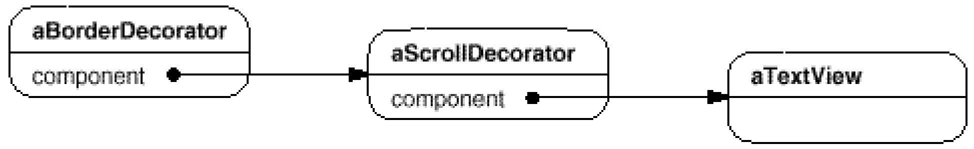
- Meet the **Decorator** pattern



```
// Delegate it
void MonoGlyph::Draw (Window* w)
{
    _component->Draw(w);
}

// Do it
void Border::Draw (Window* w)
{
   MonoGlyph::Draw(w);
   DrawBorder(w);
}
```

Multiple decorations (ornaments)….



Design Issue #5

Embellishing the User Interface

- To conclude Design Issue #5:
  - Adding additional functionalities (embellishments, ornaments) to a class can be either achieved either via inheritance (soon you will end-up with an unmanageable hierarchy) or, better, using object composition with the help of patterns like Decorator or **Chain of Responsibility** (for Chain of Responsibility see also Design Issue #8)