InterBit
Training & Consulting Ltd.

Decorator Pattern

Chapter Content

InterBit
Training & Consulting Ltd.

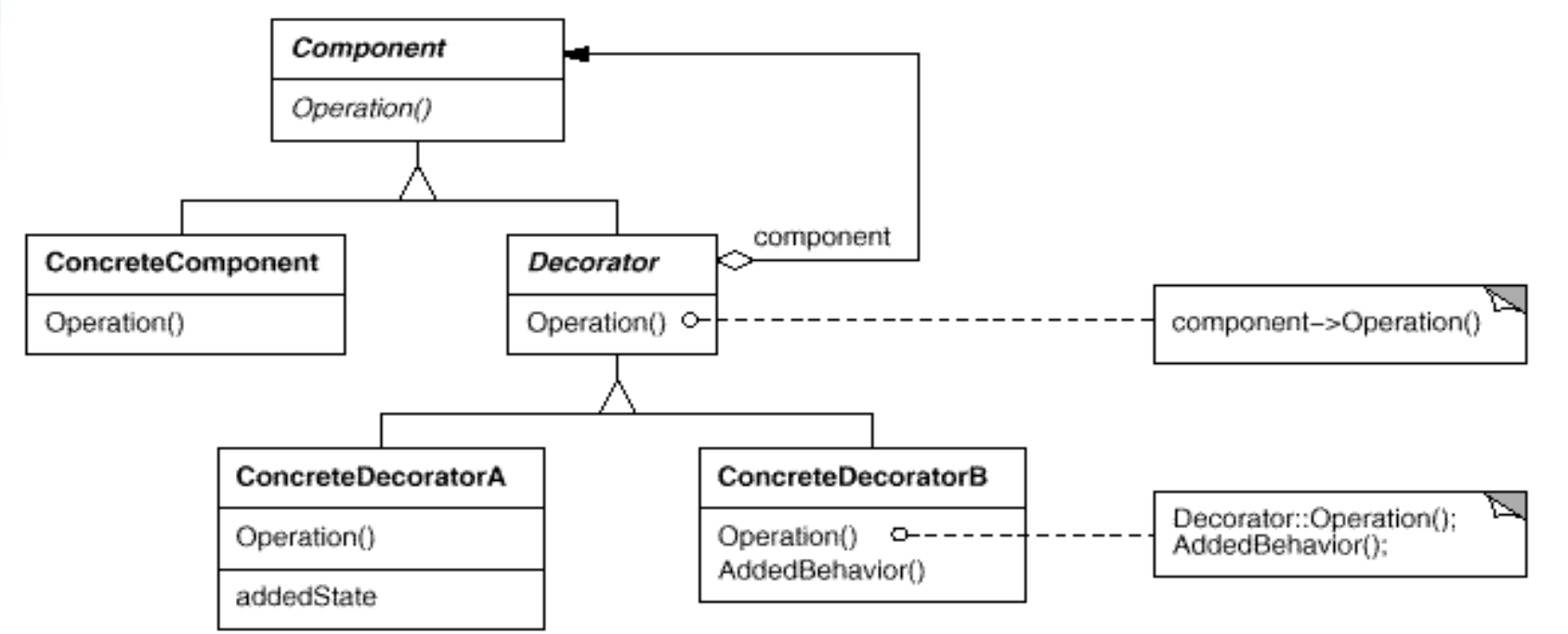
- Decorator Pattern Overview
- Decorator - UML Diagram
- Decorator – FilterStreams
- Implementation
- GUI Decorator
- GUI Decorator - UML Diagram
- Decorator Discussion

Decorator Pattern Overview

InterBit Training & Consulting Ltd.

- Decorators enhance the behavior of objects
  - Enhancing functionality of existing methods.
  - May also add new methods & attributes.
- **Decorators use composition rather than inheritance**. Thus:
  - You may select which individual objects to enhance (not necessarily the entire class).
  - Decorations can be added/remove dynamically.
  - Object may receive several decorations.

Decorator - UML Diagram



Wrap any Component with Decorator to enhance its behavior or functionality. Decorator both:

Contains a component (which facilitates polymorphism, and also allows you to enhance selected instances).

Extends Component (e.g. it can be used like any other component. Methods that are not enhanced will simply be passed on to the enclosed component).

Decorator: FilterStreams

- Example: `BufferedOutputStream`
  - **Implements** `OutputStream`
    - So it can write(), write(byte[]) ...
  - **Contains** an `OutputStream` and...
    - Decorates it with enhanced (buffered) behavior.
    - Additional attributes: buffer size.

```
// Using a buffered stream:

BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(“dest.jpg"));

Byte[] data = bos.write(data);
```

Decorator: FilterStreams (cont.)

InterBit Training & Consulting Ltd.

- Example: `DataOutputStream`
  - **Implements** `OutputStream`.
  - **Contains** an `OutputStream`.
  - **Additional capabilities**:
    write primitive types (double, int…) in binary format.

```
//Using a Data stream:
DataOutputStream dos = new DataOutputStream(
     new FileOutputStream(“dest.jpg"));
Byte[] data = dos.write(data);
dos.write(89.7F);
```

Implementation

InterBit Training & Consulting Ltd.

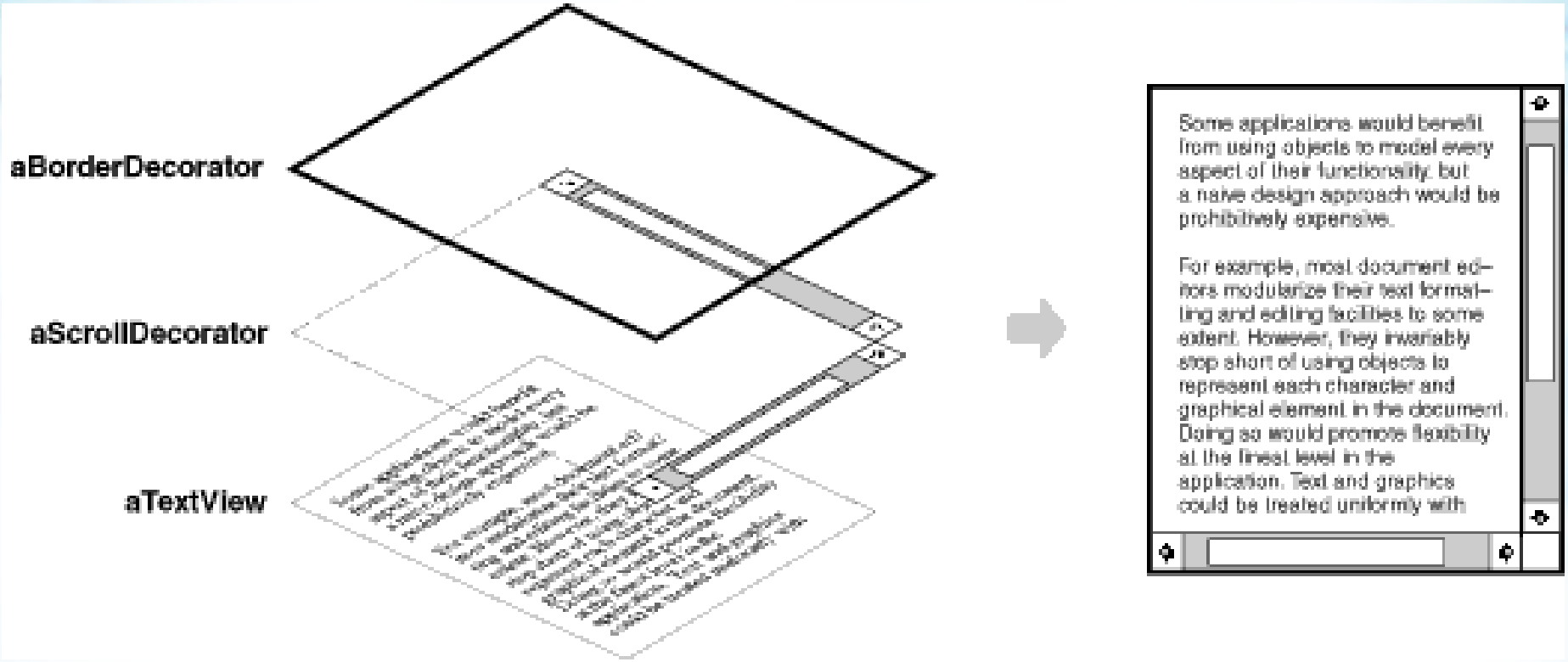
```
// Outlines a DataOutputStream implementation.
// Note: code has been simplified (standard java.io.DataOutputStreams
// inherit from FilterOutputStream, and also do additional bookkeeping)
class DataOutputStream extends OutputStream {

    private OutputStream out; // wrapee

    public void write(byte[] bytes) { out.write(bytes); }

    // The bitwise & operator performs a bitwise AND operation.
    // The unsigned right shift operator ">>>" shifts a zero into the
    // leftmost position. Shift left multiplies by 2; shift right
    // divides by 2
    public void writeShort (short s){
        out.write((s >>> 8) & 0xFF);
        out.write((s >>> 0) & 0xFF);
    }
    ...
}
```

GUI Decorators

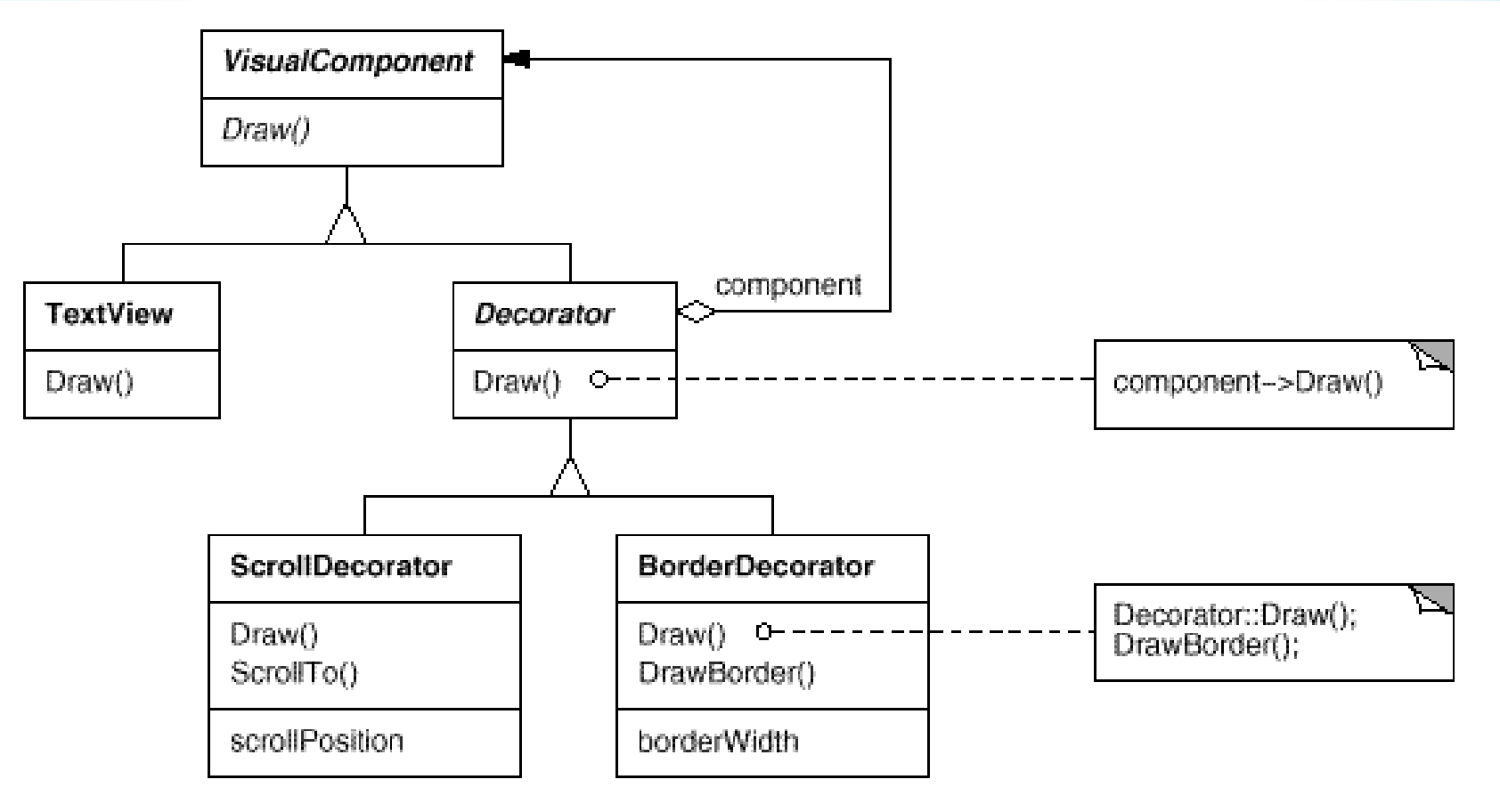


Define a BorderDecorator and a ScrollDecorator that may wrap any GUI component, allowing you to dynamically add borders and / or scrolls to selected components (note order is important !)

Note: this is not identical to the existing java.swing approach (swing takes a similar approach with scrolls, but not with borders).

GUI Decorator - UML Diagram

InterBit Training & Consulting Ltd.



Decorator Discussion

InterBit Training & Consulting Ltd.

- **Advantages over inheritance**:
  - You may choose to enhance individual objects.
  - Modular composition: ***every*** stream can be decorated with buffering ***and/or*** zipping; every component can have borders/scrollbars.
- **Disadvantages:**
  - Type-checks: Decorator is not the exact same type as the object it decorates.