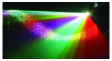
Supplement to
The Design and Implementation of Multimedia Software

The Decorator Pattern

Prof. David Bernstein

James Madison University

users.cs.jmu.edu/bernstdh



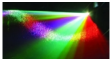
Motivation

- Adding Capabilities to a Class:

  Specialization

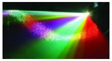
- Adding Capabilities to an Object:

  Needs to be done "dynamically" (i.e., at run time)

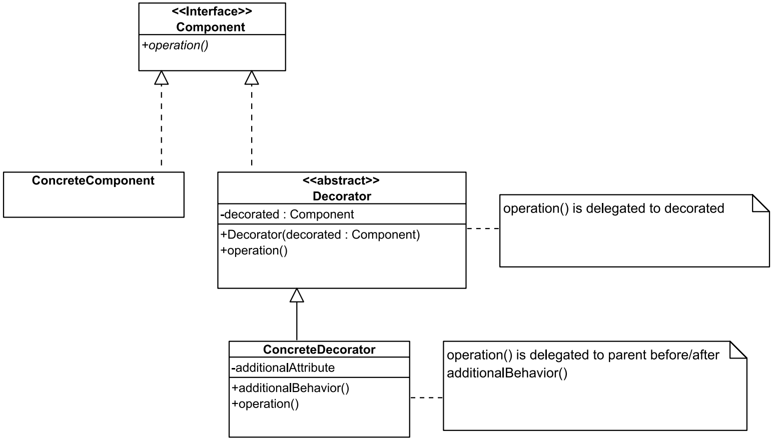


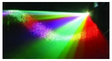
Examples of Use

- I/O Streams in Java
- Borders, etc.. on GUI components

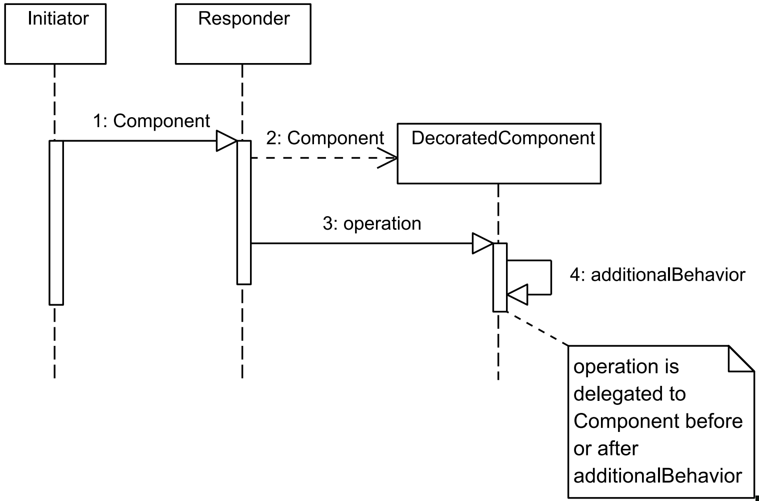


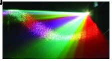
The Decorator Pattern





A Generic Application





An Example

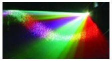
- Common Examples:

  Many examples of the decorator pattern involve Graphical User Interface (GUI) widgets

  While such examples are instructive, they tend to make people think that the decorator pattern is only used in GUIs

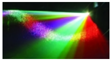
- A Non-GUI Example:

  Consider an example that is related to “printing”



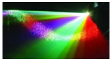
An Example

```
public interface Printer
{
    public abstract void print(String text);
}
```



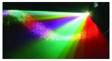
An Example (cont.)

```
public class ConsolePrinter implements Printer
{
    public void print(String text)
    {
       System.out.print(text);
    }


}
```



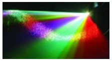
An Example (cont.)

```
public abstract class PrinterDecorator implements Printer
{
    protected Printer     decorated;

    public PrinterDecorator(Printer decorated)
    {
       this.decorated = decorated;
    }


    public abstract void print(String text);

}
```



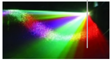
An Example (cont.)

```
public class UppercasePrinter extends PrinterDecorator
{
    public UppercasePrinter(Printer decorated)
    {
       super(decorated);
    }


    public void print(String text)
    {
       decorated.print(text.toUpperCase());
    }
}
```



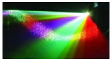
An Example (cont.)

```
import java.util.*;

public class WrappingPrinter extends PrinterDecorator
{
    protected int         width;


    public WrappingPrinter(Printer decorated, int width)
    {
       super(decorated);
       this.width = width;
    }


    public void print(String text)
    {
       int                    required, used;
       String                 token;
       StringTokenizer        st;

       st   = new StringTokenizer(text);
       used = 0;

       while (st.hasMoreTokens())
       {
          token    = st.nextToken();
          required = token.length();
```



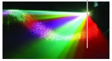
An Example (cont.) (cont.)

```
          if ((required + used + 1) > width)
          {
             decorated.print("\n");
             decorated.print(token);
             used = required + 1;
          }
          else
          {
             if (used != 0)
             {
                decorated.print(" ");
                ++used;
             }

             decorated.print(token);
             used += required;
          }
       }
    }
}
```



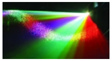
An Example (cont.)

```
public class Driver
{
    public static void main(String[] args)
    {
       Printer           printer;
       String            text;


       text = "This is the text that we will use " +
              "to demonstrate the capabilities "   +
              "of different Printer objects.";


       printer = new ConsolePrinter();
       printer.print(text);

       System.out.print("\n\n");

       printer = new UppercasePrinter(new ConsolePrinter());
       printer.print(text);

       System.out.print("\n\n");

       printer = new WrappingPrinter(new ConsolePrinter(), 20);
       printer.print(text);

       System.out.print("\n\n");
```



An Example (cont.) (cont.)

```
        printer = new WrappingPrinter(
                       new UppercasePrinter(
                           new ConsolePrinter()), 20);
        printer.print(text);
    }

}
```



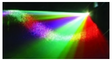
An Alternative

- The Issue:

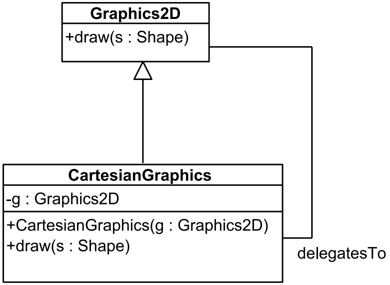
  One wants to decorate an object that does not implement an explicit interface

- A Common Solution:

  Have the `Decorator` to specialize the class to be decorated



Another Example - Decorating a `Graphics2D` Object


Graphics2D
+draw(s : Shape)
CartesianGraphics
-g : Graphics2D
+CartesianGraphics(g : Graphics2D)
+draw(s : Shape)
delegatesTo