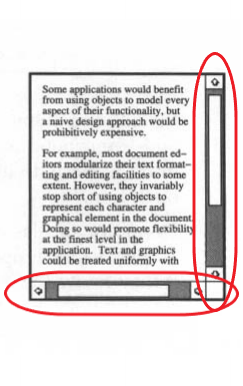
***APCS 3 - Group 4:***

* Nguyen Trung Hau - 18125129
* Nguyen Thao Ninh - 18125106
* Truong Thuy Quyen - 18125110

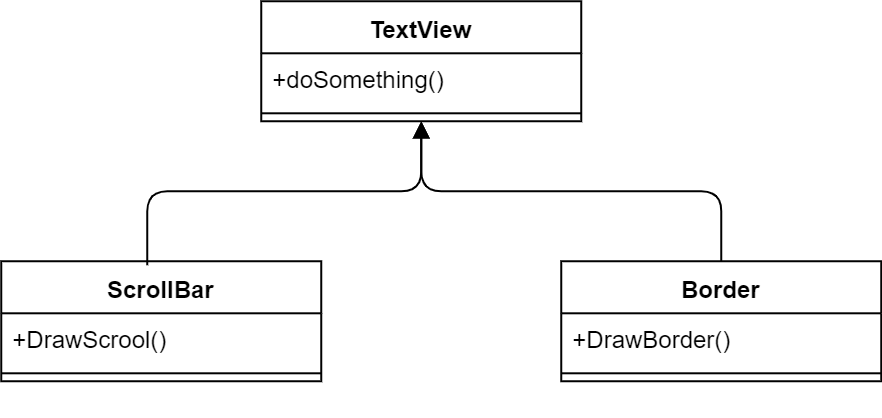
1. **Proposal**

Add properties like borders or behaviors like scrolling to any user interface component.



Our problem is to extend responsibilities for class.

1. **Solve the problem without using pattern (Inheritance)**



To solve problem with inheriting, we organize 2 classes ScrollBar (to create croll bar) and Border (to create thick black border if need) which inherit from class TextView (to print the content of subclass).

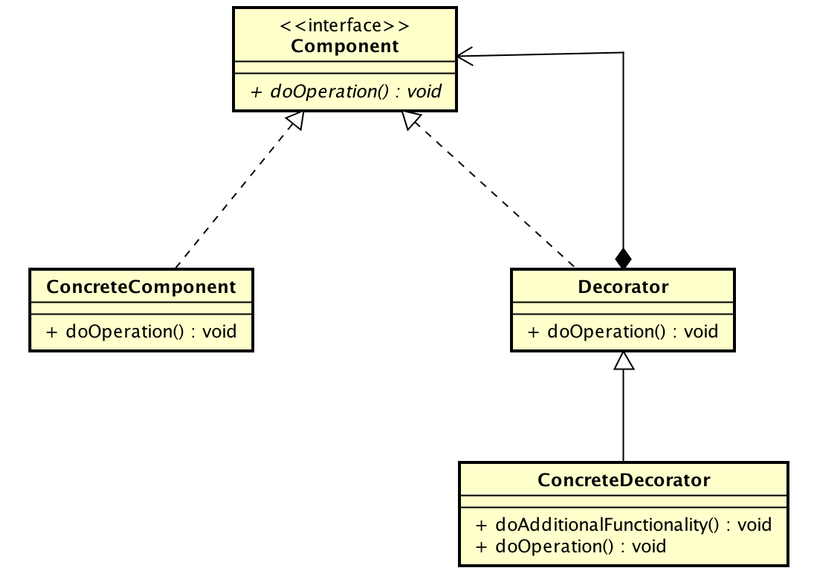
1. **Disadvantages of the method above**

In the inheriting method, A border and scroll are put around every subclass instance. But we sometimes do not need the border or scroll bar which means they are just options. So this way is inflexible when the choice of border and scroll are made statically.

We may waste time for coding because of complication of inheritance

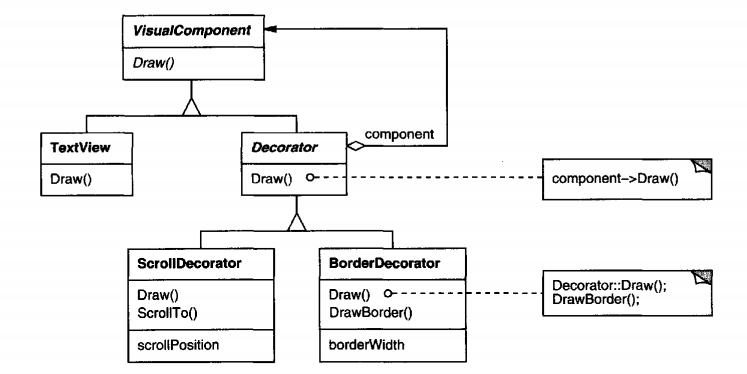
1. **Introduction about Decorator structural pattern**

Decorate, which is a one of structural patterns, is frequently used to changed the object’s functionality flexibly. Decorator does not change functionality of another objects.



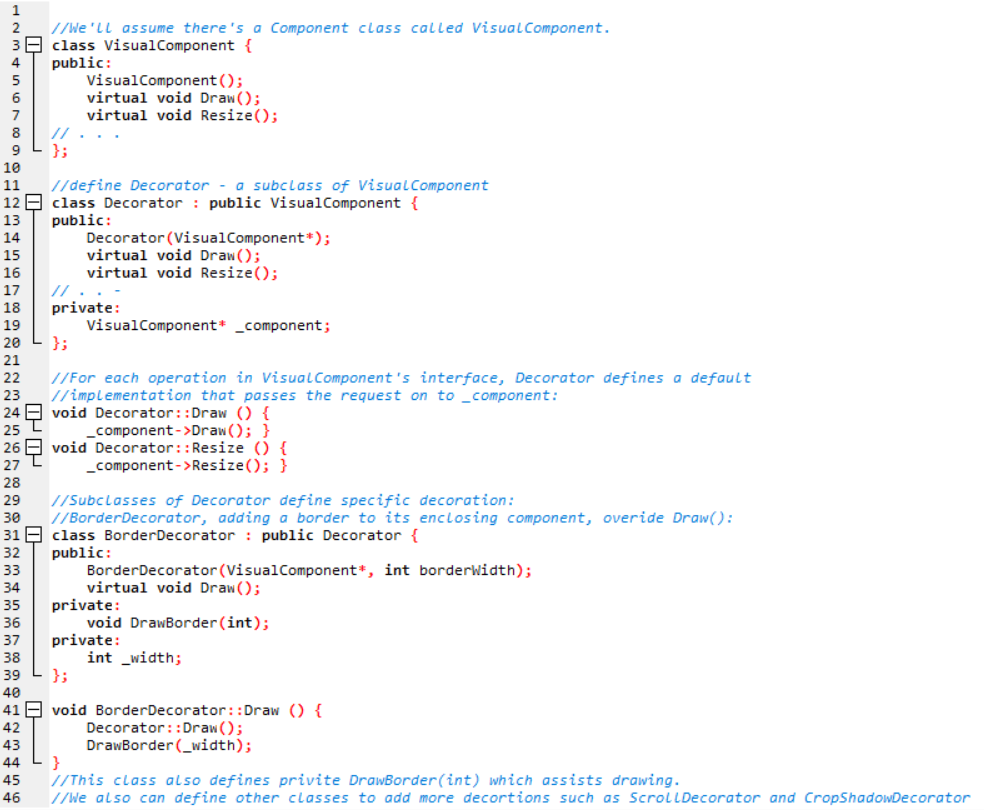
* Component is general interface used to add function to object at runtime.
* ConcreteComponent, which is a setting for Component interface, defines a instance needed adding responsibilities.
* Decorator, which is a abstract class, is used to maintain a reference of component object.
* ConcreteDecorator: is a setting of Decorator, which adds more components to component objects.

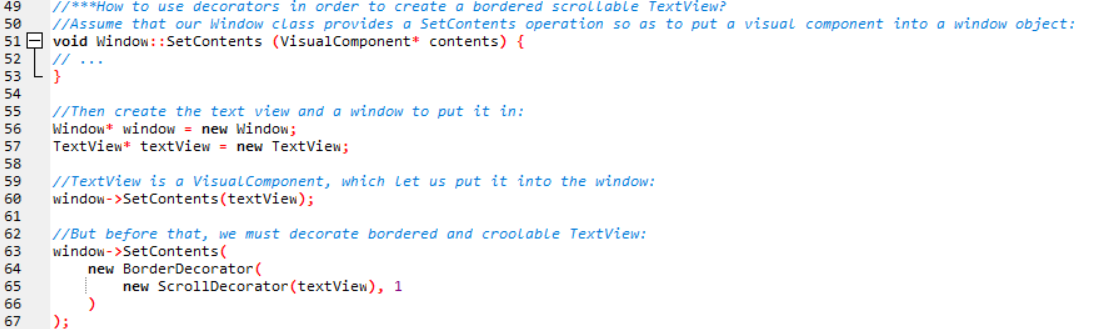
1. **Solving the problem with Decorator**



* ***Component (VisualComponent)*:** defines the interface for objects that can have responsibilities added to them dynamically.
* ***ConcreteComponent (TextView)***: defines an object to which additional responsibilities can be attached.
* ***Decorator*** : maintains a reference to a Component object and defines an interface that conforms to Component’s interface.
* ***ConcreteDecorator(BorderDecorator, ScrollDecorator)*** : add responsibilities to the component

1. **Sample Code**





1. **Some other problems of Decorator’s application**

Streams are a fundamental abstraction in most I/O facilities. A stream can provide an interface for converting objects into a sequence of bytes or characters. That

lets us transcribe an object to a file or to a string in memory for retrieval later. A

straightforward way to do this is to define an abstractStream class with subclasses

MemoryStream and FileStream.But suppose we also want to be able to do the

following:

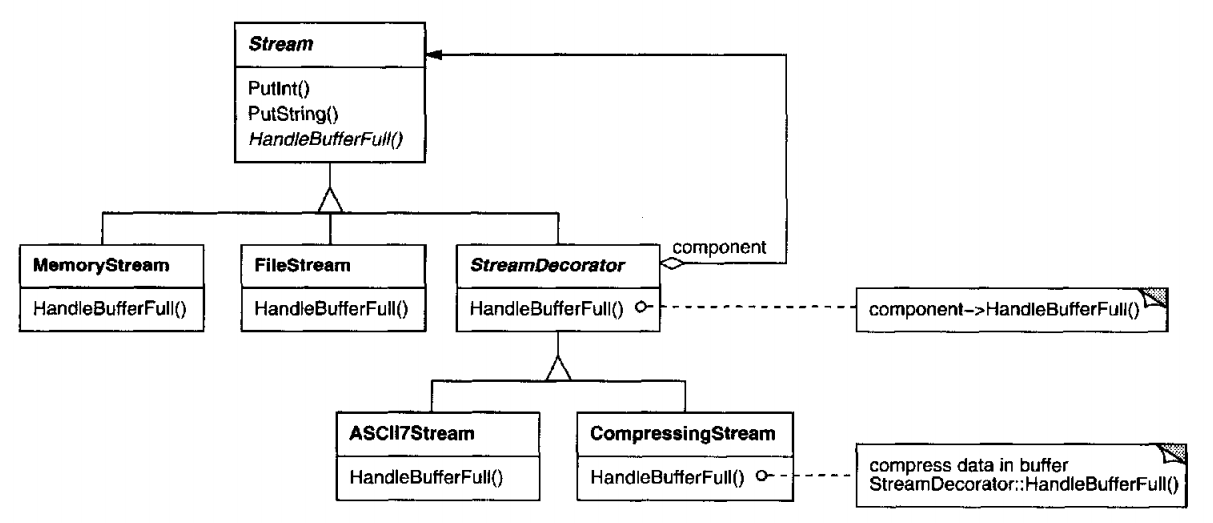
• Compress the stream data using different compression algorithms (runlength encoding, Lempel-Ziv, etc.).

• Reduce the stream data to 7-bit ASCII characterssothat it canbe transmitted

over an ASCII communication channel.

The Decorator pattern gives us an elegant way to add these responsibilities to

streams. The diagram below shows one solution to the problem:



1. **Advantages and disadvantages of Decorator pattern**

* *Strong point:*

Decoration is more convenient for adding functionalities to objects instead of entire classes at runtime. With decoration it is also possible to remove the added functionalities dynamically.

* *Weak point:*

Decoration adds functionality to objects at runtime which would make debugging system functionality harder.

1. ***Five Multiple-choice-questions***

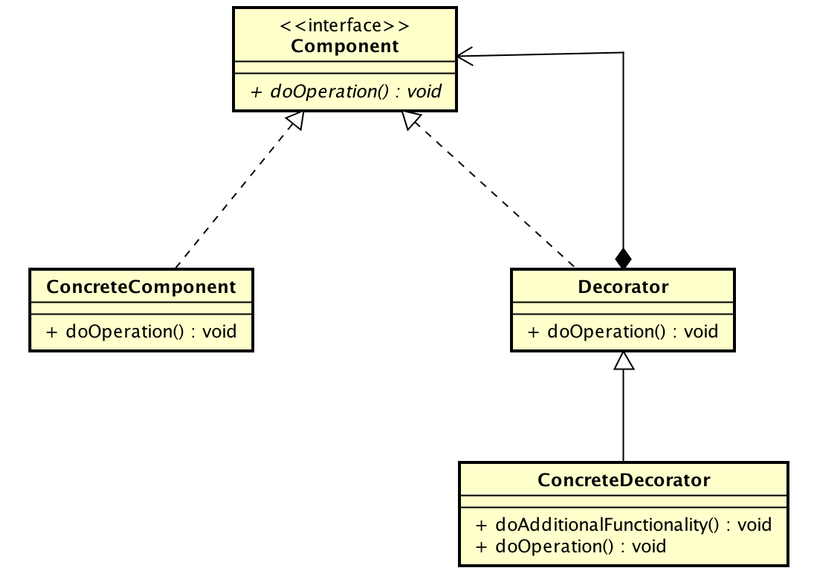
***Question 1: How can we add more responsibilities into (all/some/an) object(s) in Object-Oriented? (Choose the best answer.)***

1. Using inheritance.
2. Using Decorator Design Pattern.
3. Both of 2 first choices.
4. Can not add.

***Question 2: Which is the way that allows us to add more responsibilities into objects but not of entire class?***

1. Using inheritance.
2. Using Decorator Pattern.
3. Using Adapter Pattern.
4. Can not add.

***Question 3: In the diagram of Decorator below, what is ConcreteComponent class used for ?***



1. Interface for objects that can have responsibilities added to them dynamically.
2. Defines an object to which additional responsibilities can be added.
3. Maintains a reference to a Component object and defines an interface that conforms to Component's interface.
4. Concrete Decorators extend the functionality of the component by adding state or adding behavior.

***Question 4:In the diagram of Decorator below, what is ConcreteDecorator class used for ?***

1. Interface for objects that can have responsibilities added to them dynamically.
2. Defines an object to which additional responsibilities can be added.
3. Maintains a reference to a Component object and defines an interface that conforms to Component's interface.
4. Concrete Decorators extend the functionality of the component by adding state or adding behavior.

***Question 5: When does Decoration add functionalities to objects ?***

1. At Compiletime.
2. At Runtime.
3. At the time codes were written.
4. At midnight.
5. **Reference**

Design Patterns Elements Of Reusable Object-Oriented Software [Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides]

Decorator Pattern <https://www.oodesign.com/decorator-pattern.html>