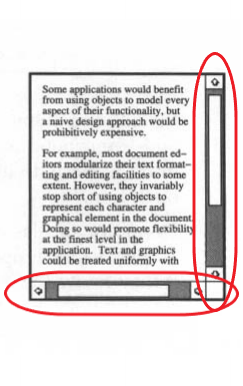
**DECORATOR - STRUCTURAL PATTERN**

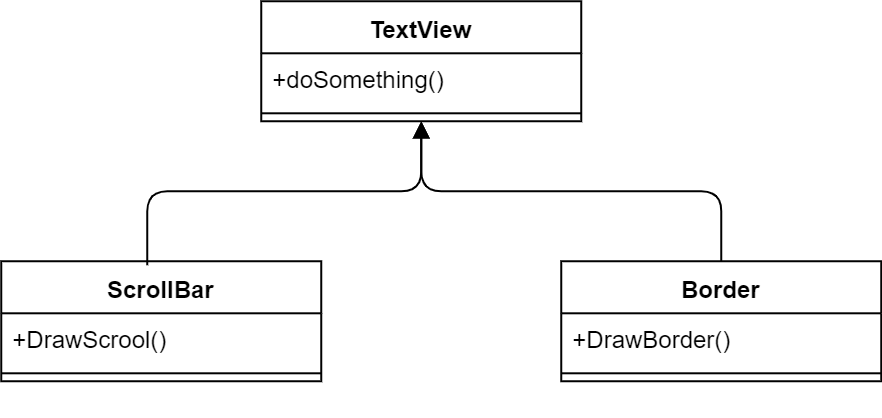
1. Proposal

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).

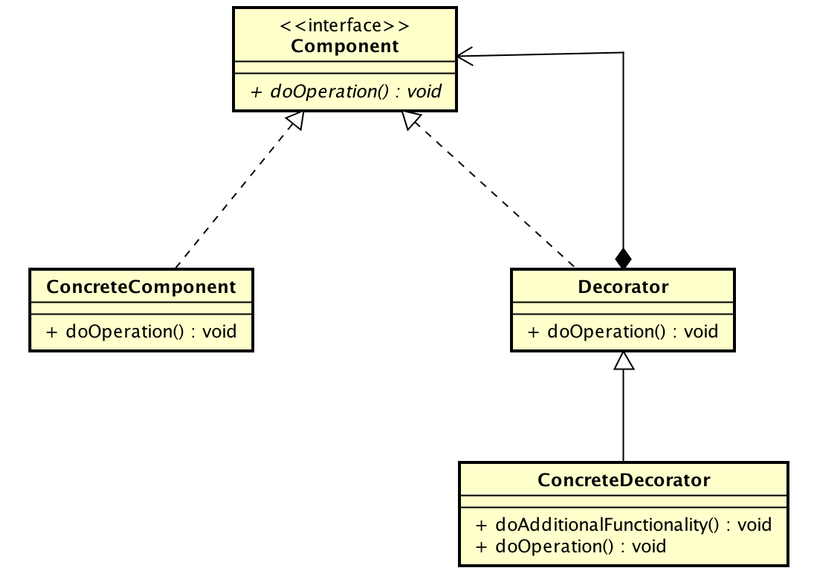
1. Disadvantages of the method above

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

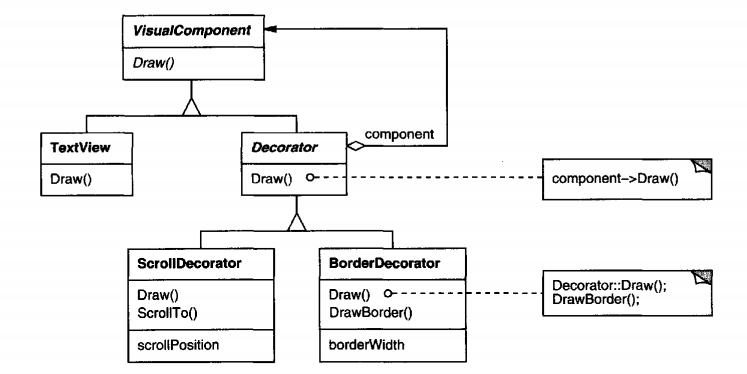
1. Introduction about Decorator structural pattern

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 interfacefor 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 tha

1. Some other problem of Decorator’s application
2. Ưu và nhược điểm của pattern.
3. Làm 5 câu hỏi trắc nghiệm trong slide