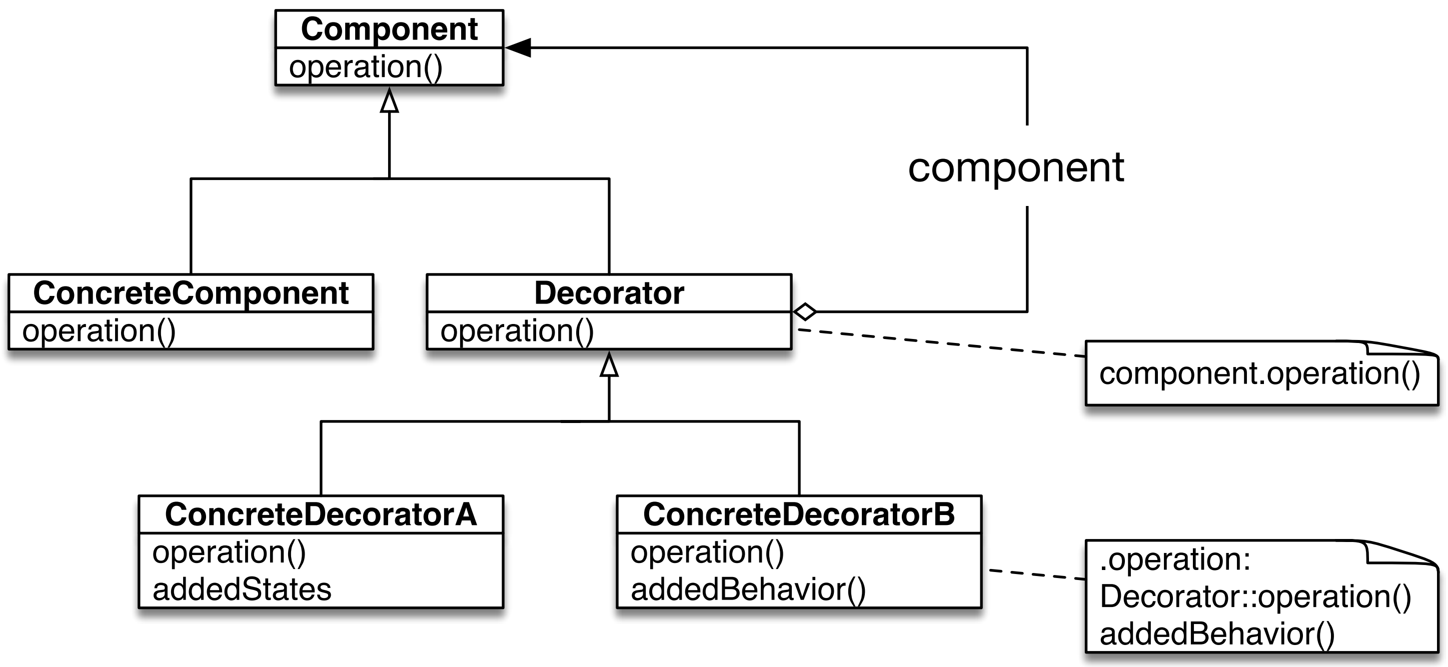
Decorator / Wrapper

Attach additional responsibilities to an object dynamically.
Provide a flexible alternative to subclassing for extending functionality.
Client doesn't have to know if it's a decorated component, or a concrete component. (Decorator should conform to the interface of its component) (think of graphical embellishments like scrollbar)



Decorator offers more flexibility than static inheritance, avoids feature-laden classes higher up in the hierarchy. Keep in mind that a decorator and its component have different object identity; and decorator pattern may introduce lots of little objects.
Data-wise, Component class should be lightweight, so that Decorator class can be lightweight. (if they are not lightweight, or if you mean to change the guts of the Component rather than the skin, consider using Strategy pattern)

Looking at the class diagram, decorator can be thought of a composite pattern whose composite child class has one and only one leaf component, that cannot be added or removed dynamically (given at ctor). (and have no pointer to parent, and meant to add responsibilities to the Component)
Decorator interface may not be needed, if you only need one extra behavior class.