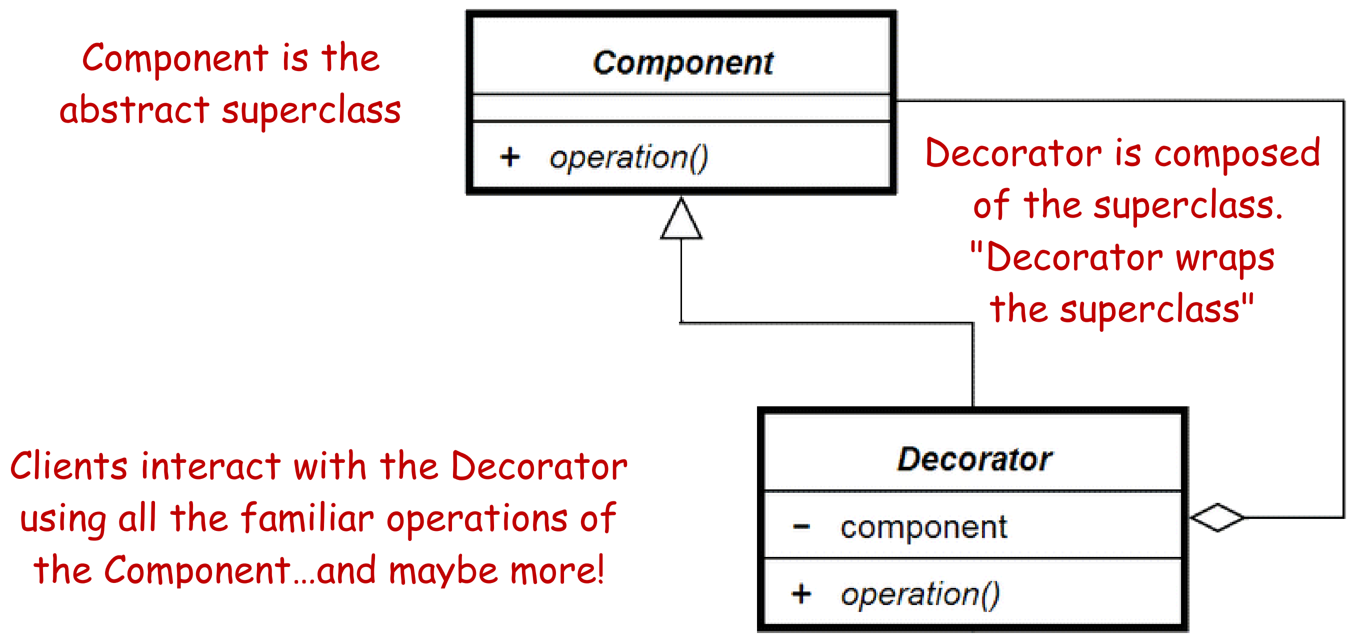
## The Decorator Pattern

The Decorator is known as a structural pattern, as it's used to form large object structures across many disparate objects. The definition of Decorator provided in the original Gang of Four book on Design Patterns[1] states:

*The Decorator Pattern:*

***Allows for the dynamic wrapping of objects in order to modify their existing responsibilities and behaviors***

Traditionally, you might consider subclassing to be the best way to approach this - but there will be cases that subclassing isn't possible, or is impractical. This leads us to the **Open/Closed Principle**: classes should be open for extension, but closed for modification. This is a good principle to keep in mind, as it keeps your class stable, but leaves it open for extension if someone wants to add behavior[2].


Component is the abstract superclass
Component
+ operation()
Decorator is composed of the superclass. "Decorator wraps the superclass"
Clients interact with the Decorator using all the familiar operations of the Component…and maybe more!
Decorator
- component
+ operation()

Decorators may be abstract when appropriate, or concrete. The Java IO API makes extensive use of concrete decorators like BufferedReaders.

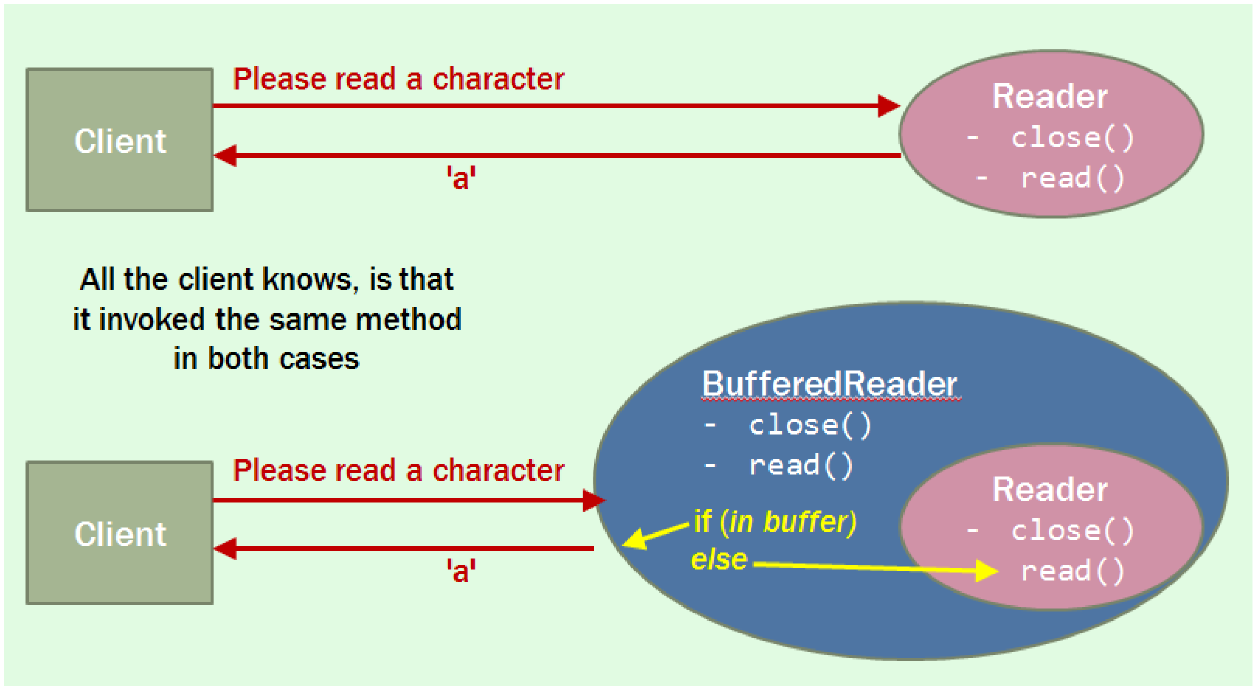
[1] Gamma, Erich. *Design patterns: elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley, 1995.

[2] "Design Patterns Uncovered: The Decorator Pattern." *Javalobby*. N.p., n.d. Web. 10 Apr. 2014. <http://java.dzone.com/articles/design-patterns-decorator>.

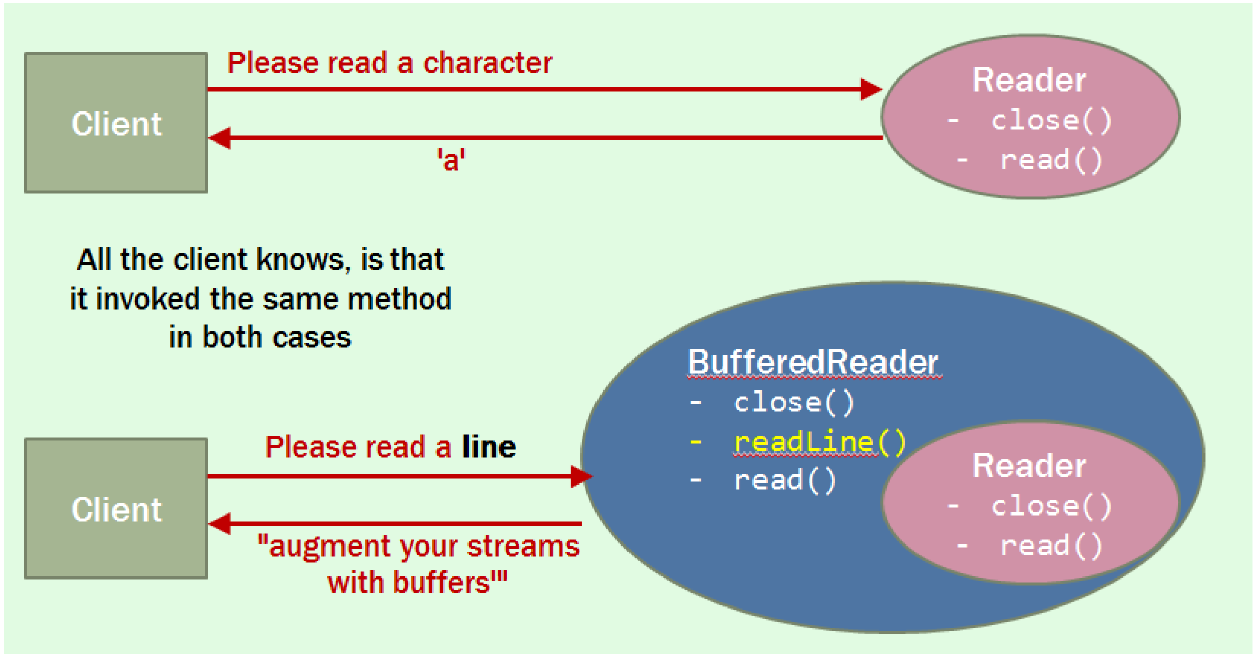
# Java IO examples

A Decorator may provide improved functionality over the Component it decorates, as seen in Figure 1 when a client invokes the read() method.  Additionally, a Decorator can add new functionality as seein in Figure 2.

## Figure 1:   Improved functionality of read()



## Figure 2:  Additional Decorator functionality – the addition of readLine()



Client
Please read a character
'a'
Reader
- close()
- read()
All the client knows, is that it invoked the same method in both cases
Client
Please read a line
"augment your streams with buffers"
BufferedReader
- close()
- readLine()
- read()
Reader
- close()
- read()