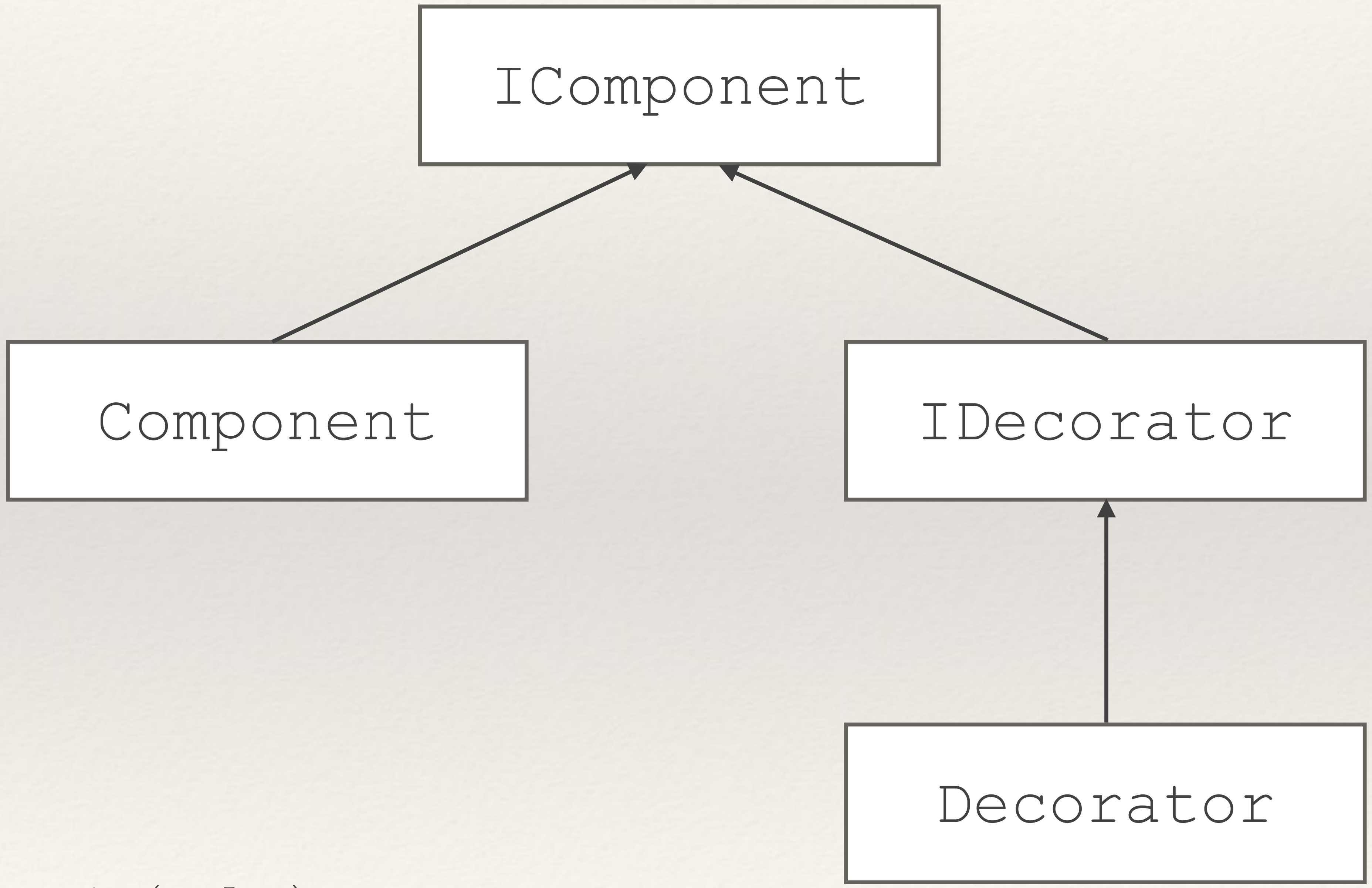
DigInG Class Fall 2015

Decorator Pattern
AOP

Decorator Pattern

Adding additional functionality to an existing object without modifying its structure.

Decorator Pattern



Statically typed language version (e.g. Java)
little different in Python or Ruby

```
class Pony implements IPony:
    def ride():
        print “riding the pony”

class SattledPony implements IPony, IDecoratedPony:
    var mypony;
    def init(pony):
        mypony = pony

    def ride():
        print “saddle the pony”
        mypony.ride()

class Unicorn implements IPony, IDecoratedPony:
    var mypony;
    def init(pony);
        mypony = pony

    def ride():
        print “put a horn on the pony”
        mypony.ride()
```

Aspect-oriented programming

"Aspect-Oriented Programming (AOP) complements OO programming by allowing the developer to dynamically modify the static OO model to create a system that can grow to meet new requirements."

http://www.onjava.com/pub/a/onjava/2004/01/14/aop.html

Terminology

- Cross-Cutting Concerns
- Advice
- Point-cut
- Aspect

Concrete

- https://github.com/diging/quadriga/blob/develop/Quadriga/src/main/java/edu/asu/spring/quadriga/aspects/AccessAspect.java
- https://github.com/diging/vogon-web/blob/38e18bc38ea178d67c4873b2fe02b68444c21292/concepts/signals.py
- http://blog.arkency.com/2013/07/ruby-and-aop-decouple-your-code-even-more/