Object Composition Using Chain of Responsibility

Zoran Horvat

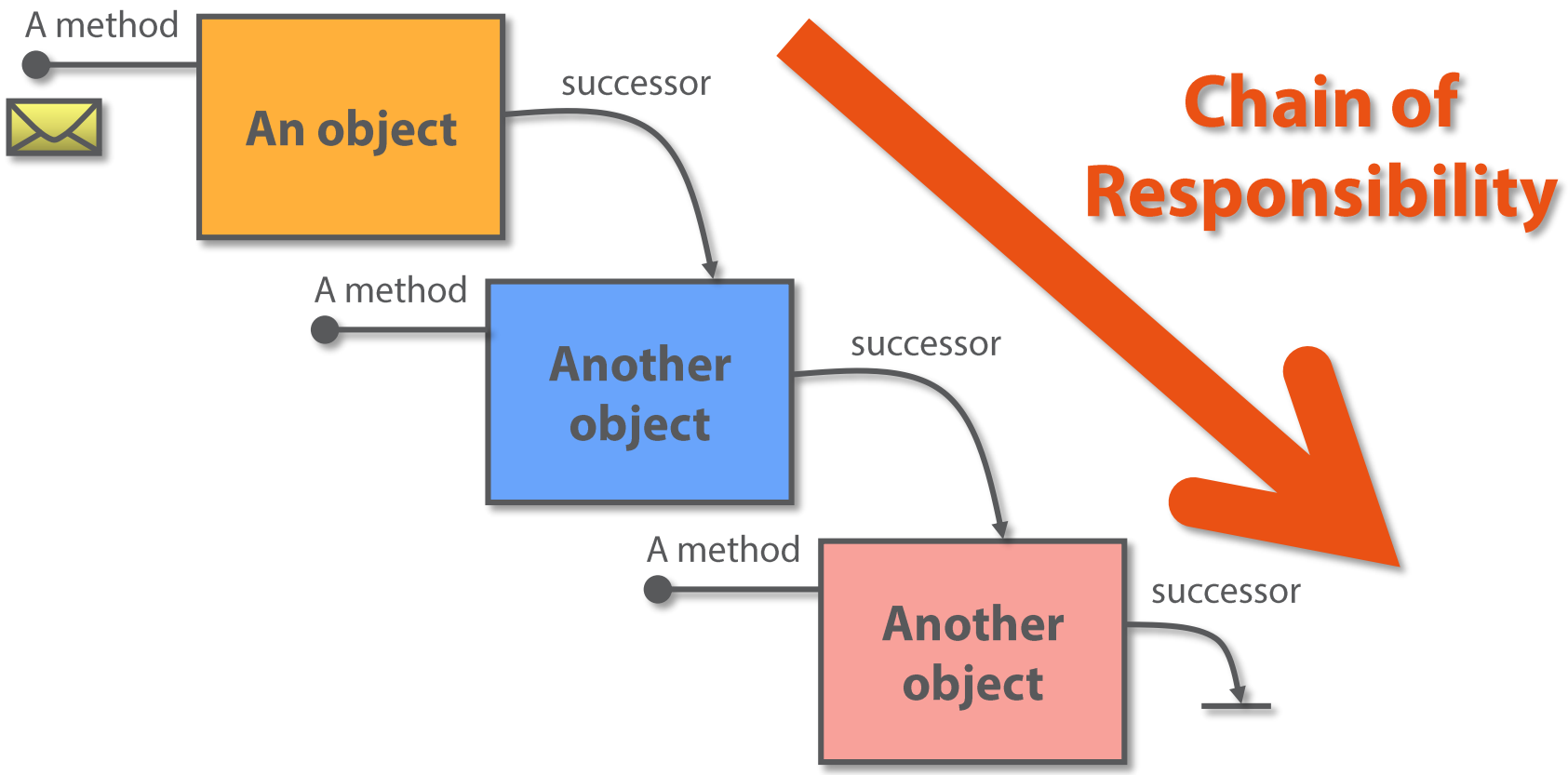
@zoranh75 | www.codinghelmet.com

Definition

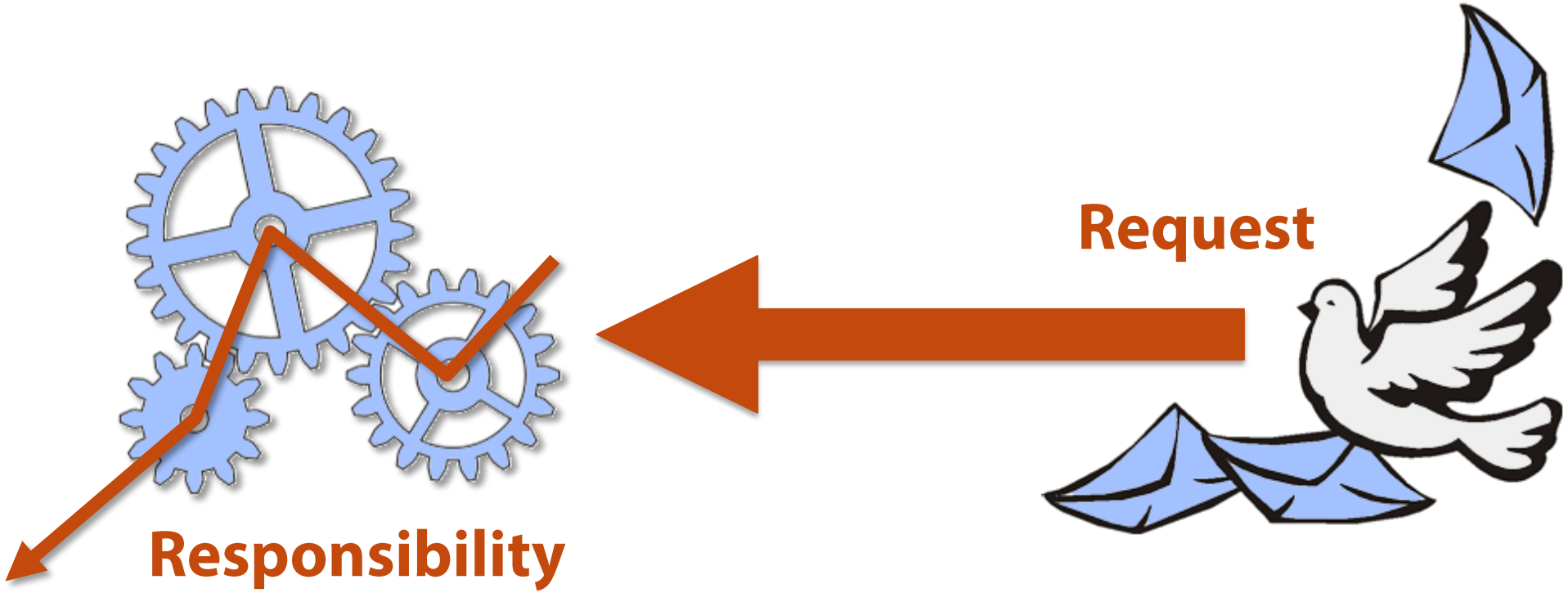
"Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."

Gama et al., Design Patterns: Elements of Reusable Object-Oriented Software

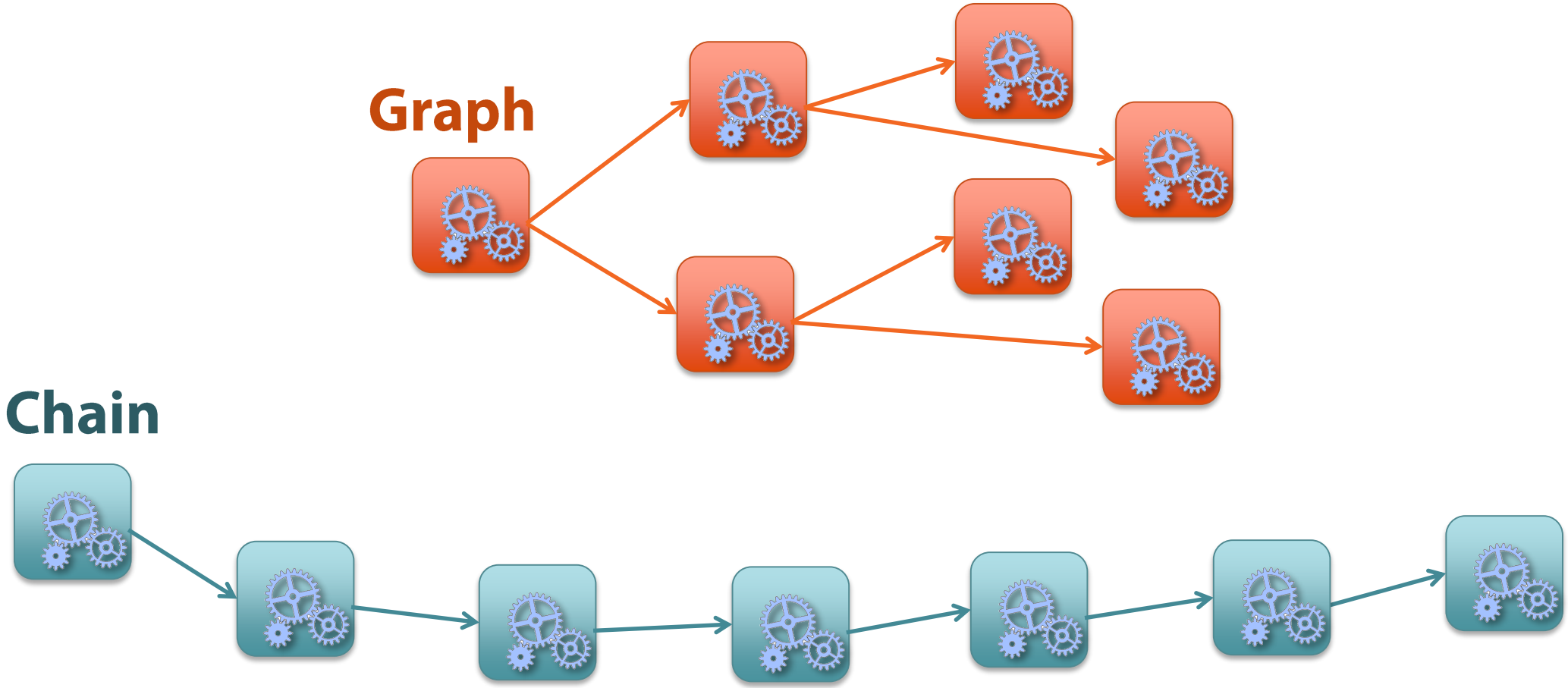
The Way It Works



Some Thoughts About the Pattern

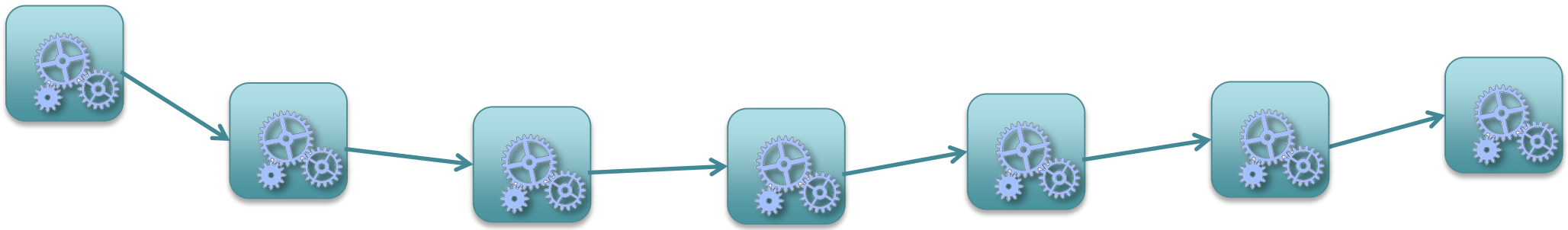


Some Thoughts About the Pattern



Some Thoughts About the Pattern

Chain

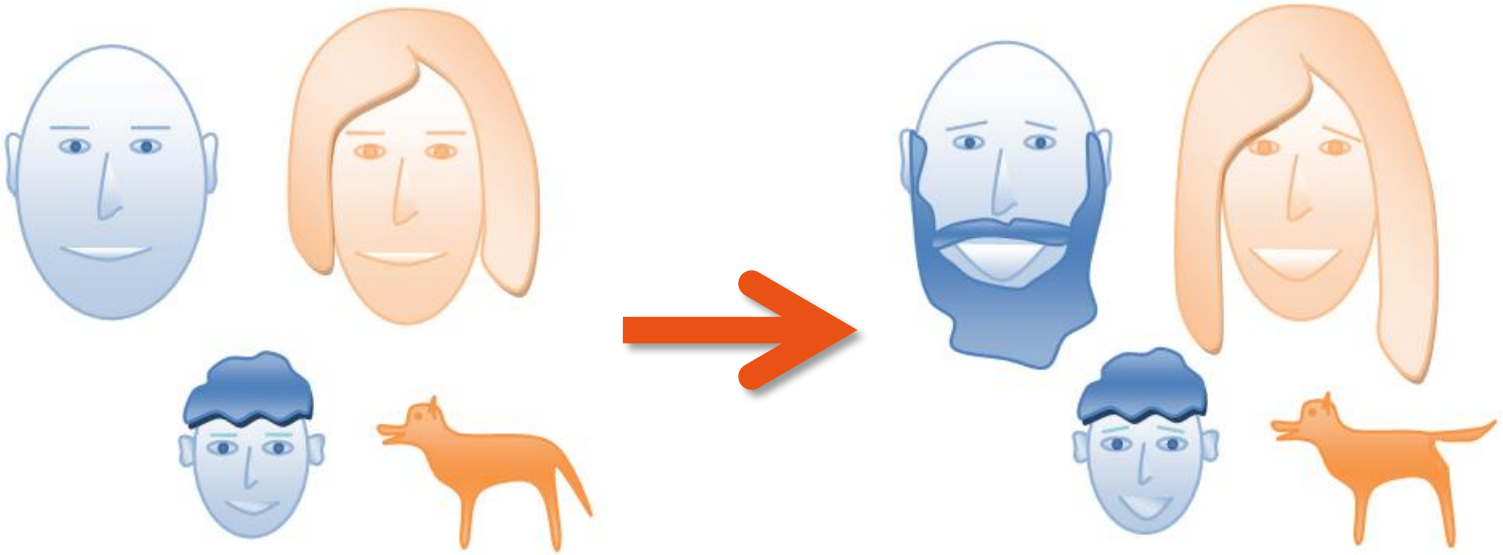


Request

a) Someone please execute DoSomething on **these data**

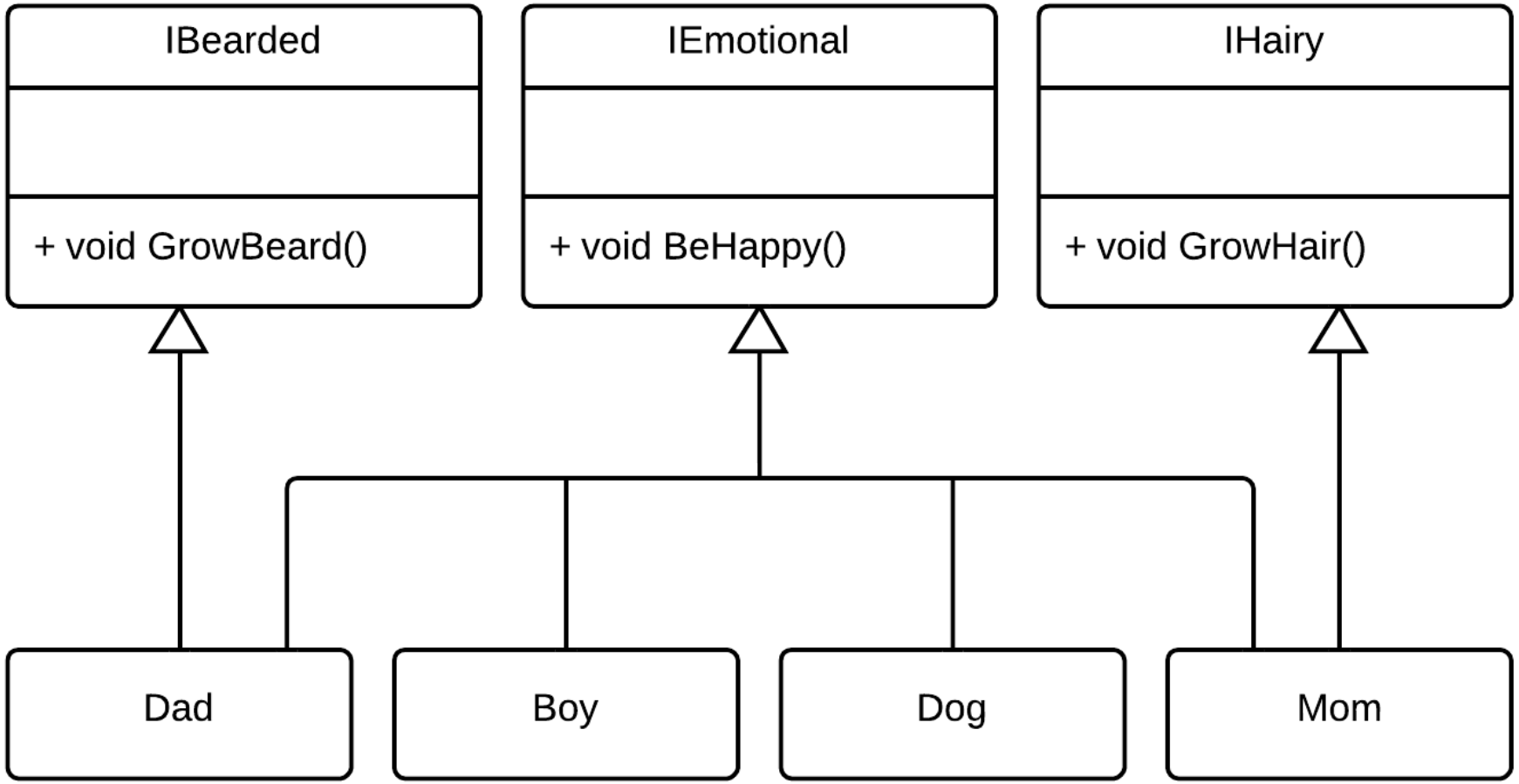
b) Whoever implements **this interface**, please answer (a.k.a. **dynamic downcast**)

Problem Statement

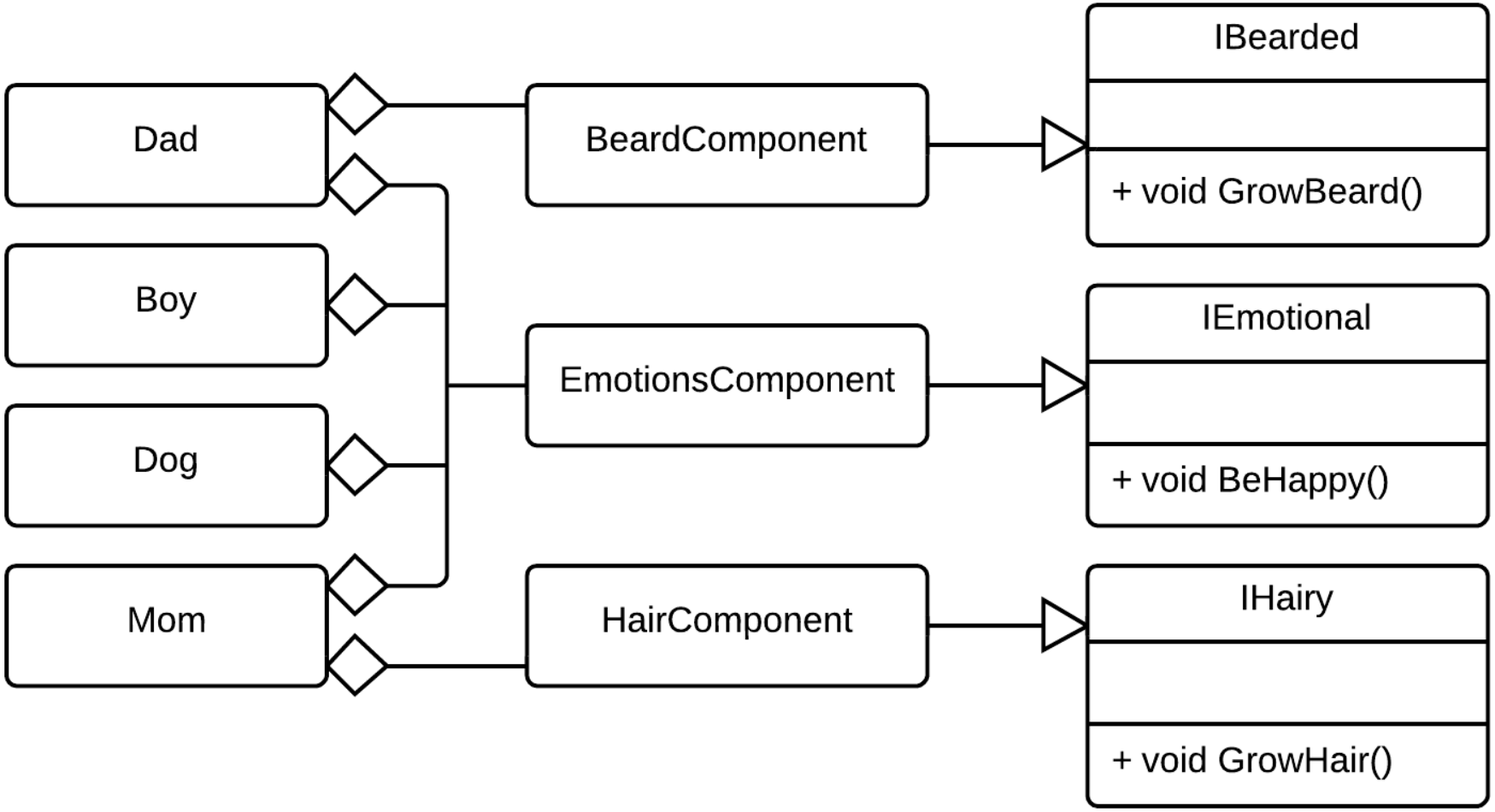


- Behaviors
  - Grow beard - Only dad can grow beard
  - Be happy - People laugh, the dog waves its tail
  - Grow hair - Only mom grows hair

One Class per Family Member



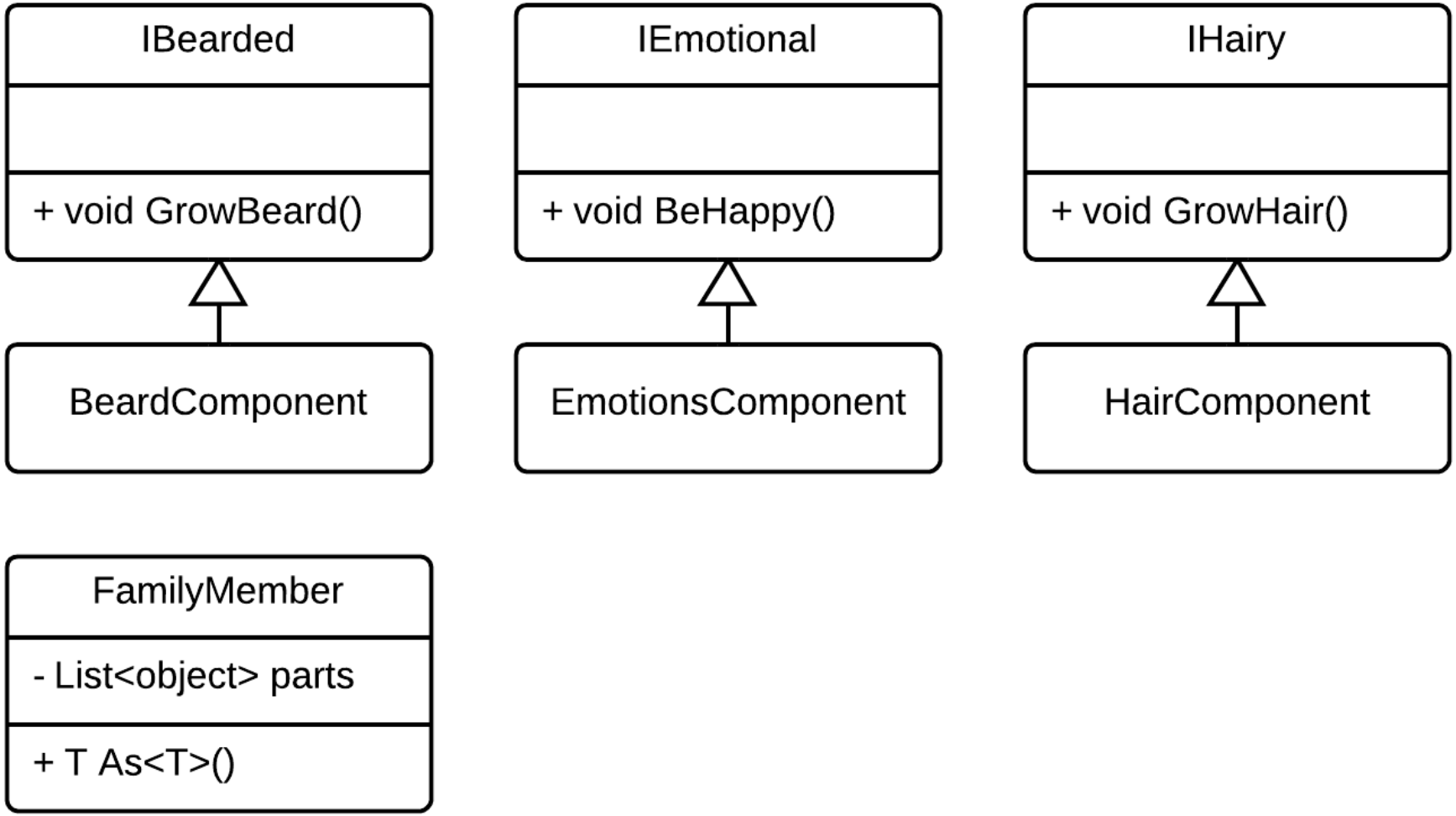
Object Composition Instead of Inheritance



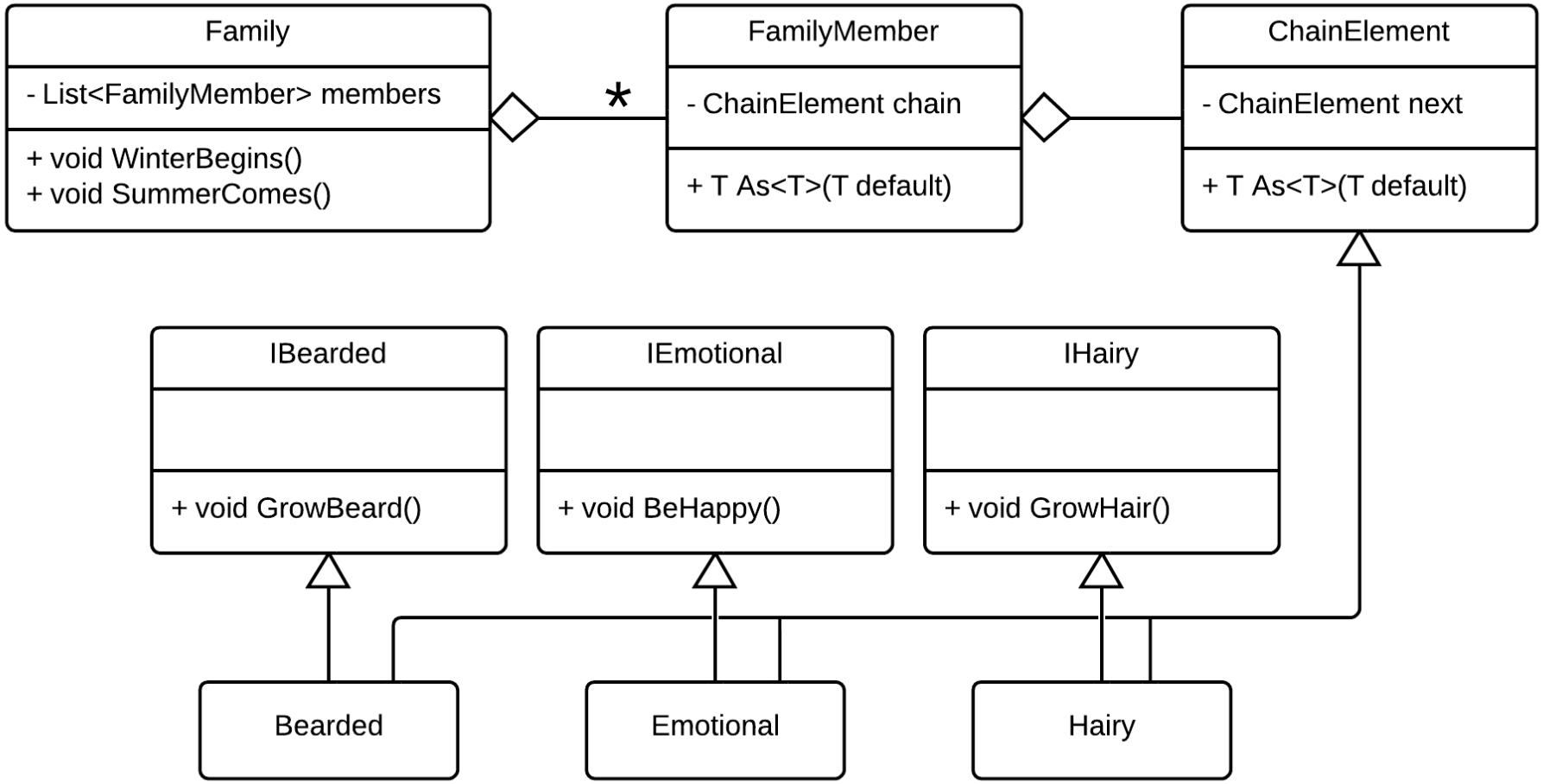
Discussing the Composition Idea

- Every concrete class must still implement interfaces
  - Implementation would just delegate calls to contained components
- Client must still perform conditional downcast to desired interface
- Fundamental problem is accessing the contained functionalities
  - Interface of a concrete class must look like union of its components
  - That might open room for the Chain of Responsibility design pattern
  - The responsibility would be to expose contained objects

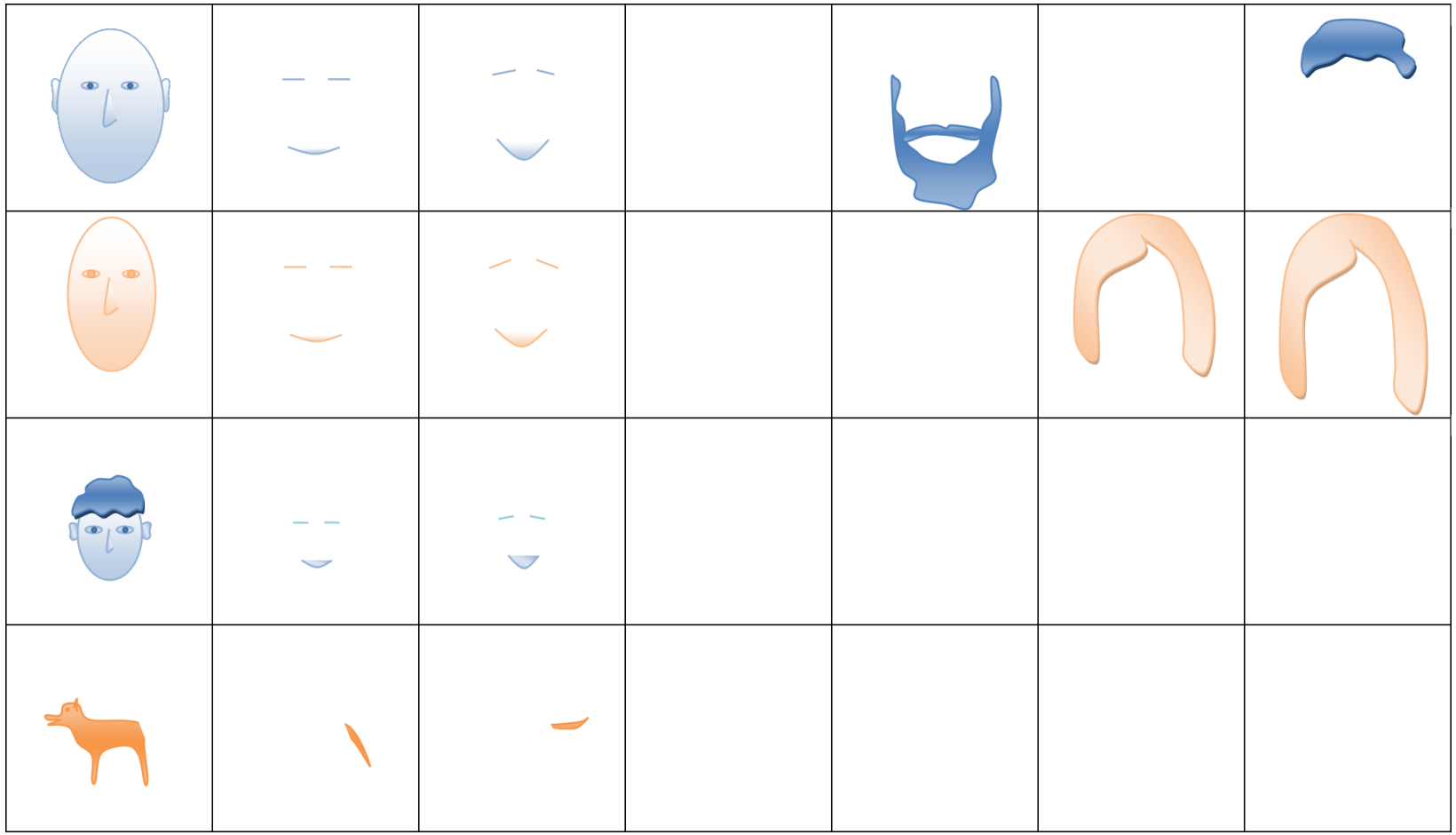
Generalized Object Composition



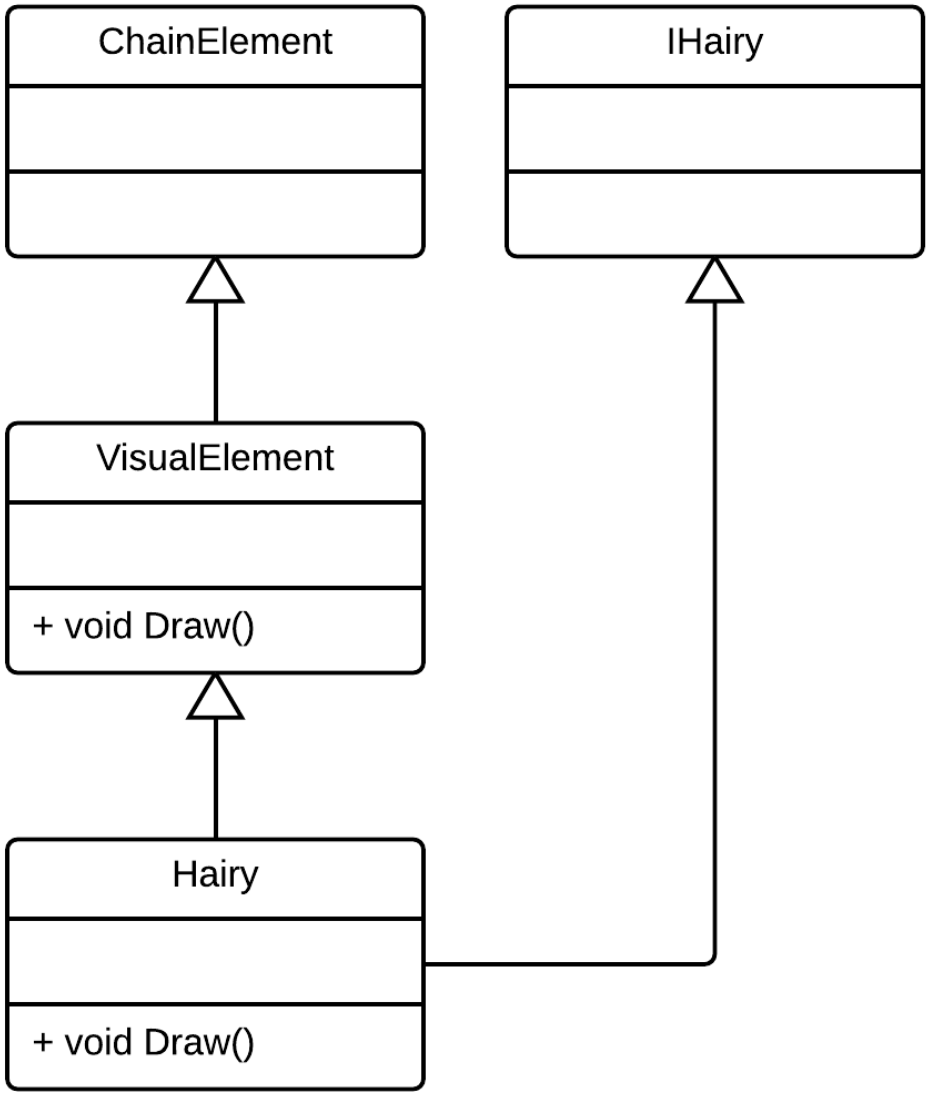
Object Composition Using Chain of Responsibility



Family Drawing Elements



Chain of Responsibility Organization



Summary

- Chain of Responsibility can be used to compose objects
  - Responsibility which is chained was tuned to application's needs
  - Null Object design pattern was used to support the chain
  - Singleton design pattern was used to support Null Objects
- Lessons learned
  - Design patterns are not just added to the design
    - Instead, they emerge gradually
  - Design patterns do not emerge in isolation
    - Often more than one design pattern emerges together

Summary

- We do not want to just use design patterns
  - We want to solve design problems
  - That is where design patterns help us
- The next module
  - Visitor design pattern
    - Provides a different means of moving responsibilities to other classes