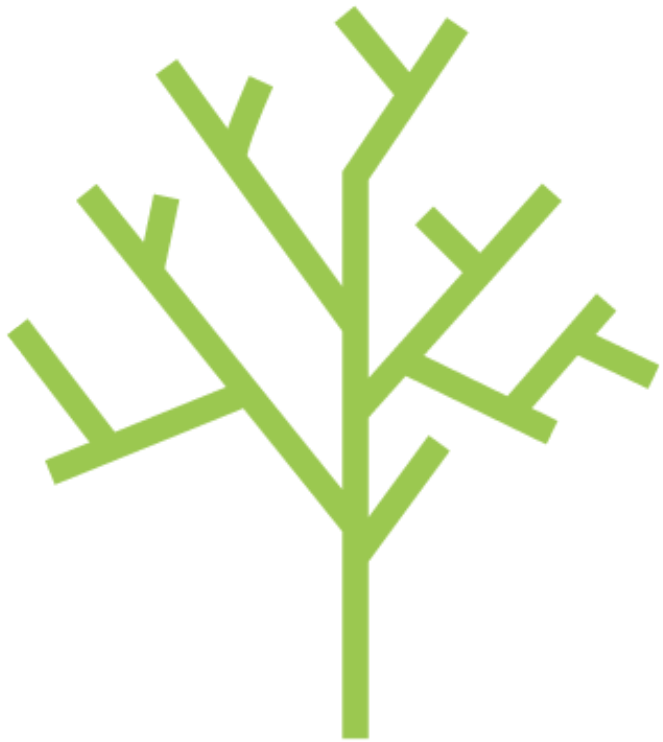
Climbing Trees with the Composite Pattern

Gerald Britton
IT SOLUTIONS DESIGNER

@GeraldBritton www.linkedin.com/in/geraldbritton



Part of a tree = mini tree

Part-whole hierarchies

A part resembles the whole

Composite Pattern handles hierarchies

Uniform code for the part or the whole

Motivation

Family trees

Parents and children

Want to find the oldest person

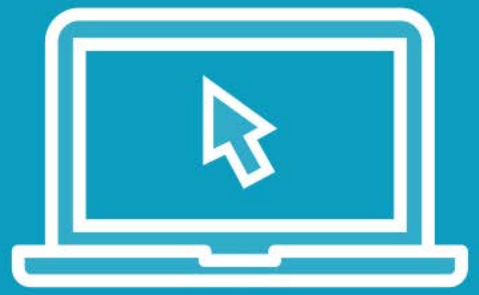
But some people not in families

Want to include them

What about married children?

Or grandparents?

Demo



Simple family

Plus some unattached singles

Look at one way to find the oldest

See what kind of trouble I get into

Composite

Classification: Structural

Compose objects into tree structures

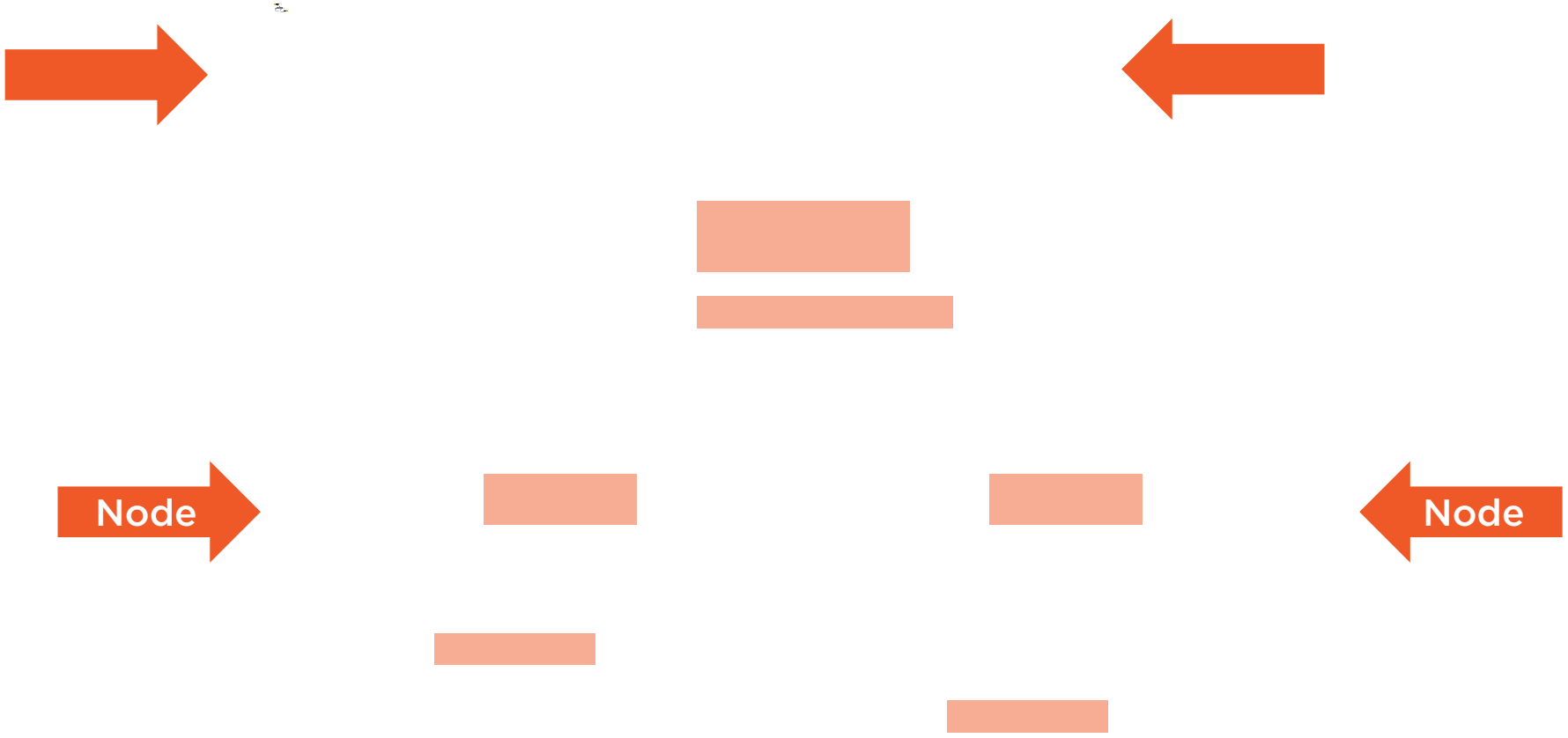
Represent part-whole hierarchies

Clients can handle individual objects

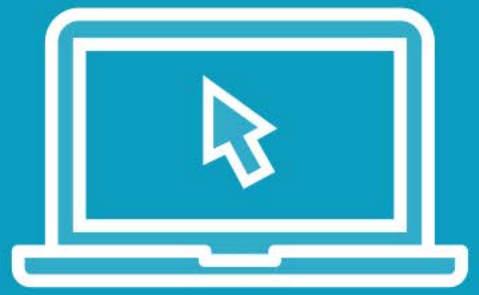
...and collections of objects

Using the same code

Composite Pattern Structure



Demo



Implement the Composite Pattern

Create a tree holding families and singles

Family and Person -> AbsComposite

Simpler client code

Consequences

- Single interface to tree structure
- Uniform access to subtrees and leaf nodes
- Simplified client code
- No need to do run time type checking
- Easy to add new kinds of components
- ...without changing client code
- Follows the Open/Closed principle
- Violates the Single Responsibility Principle

Summary

When to use Composite Pattern?

When your data fits a tree-like structure

Client code can treat data uniformly

Children can maintain parent references

Possible to share components

Can make your design too general