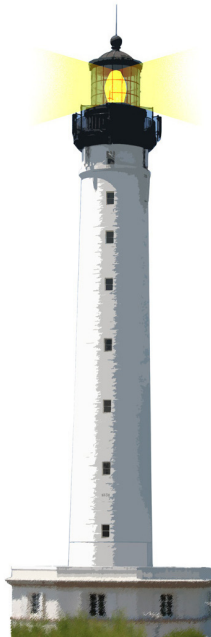
Advanced Object-Oriented Design

Composite

S. Ducasse



Pharo

http://www.pharo.org

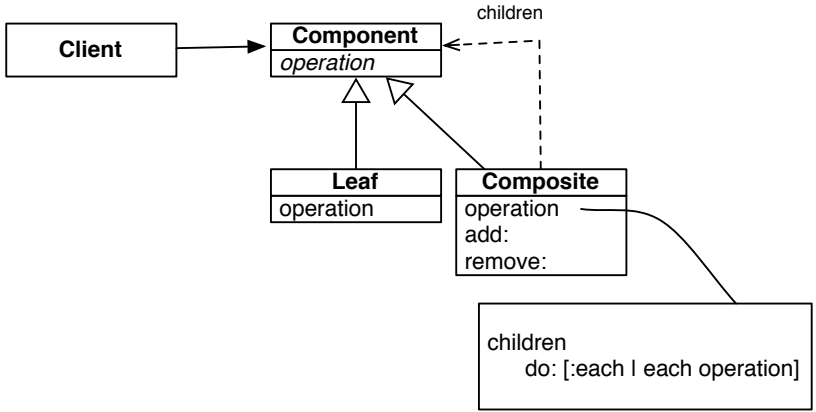
Outline

- Composite
- Composite discussions

Composite: Intent

- Compose objects into tree structures to represent part-whole hierarchies
- Composite lets clients treat individual objects and compositions of objects uniformly

Composite design essence



Composite motivation

A tree is a

- leave
- a node with trees as childen

Composite motivation

A book is composed of

- title
- table of contents
- chapters

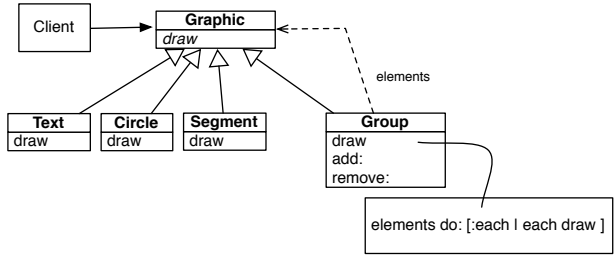
A chaper is composed of

- sections
- paragraph
- lists

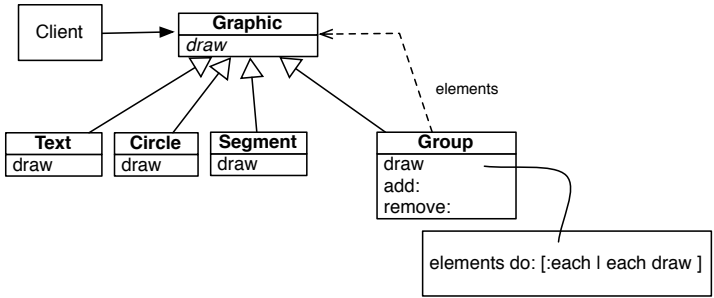
Composite motivation

A diagram is composed of elements
An element can be

- a circle
- a segment
- a text
- a group of elements

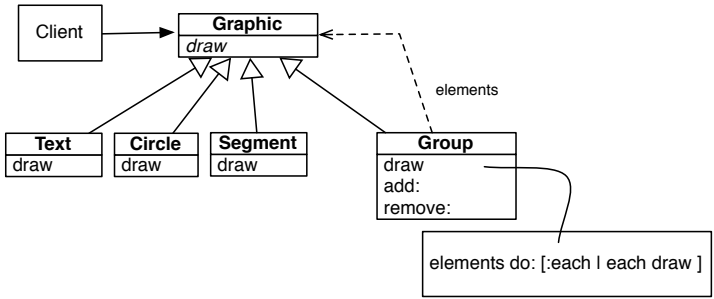


Composite participants: Client



Client manipulates objects in the composition through the **Component** interface

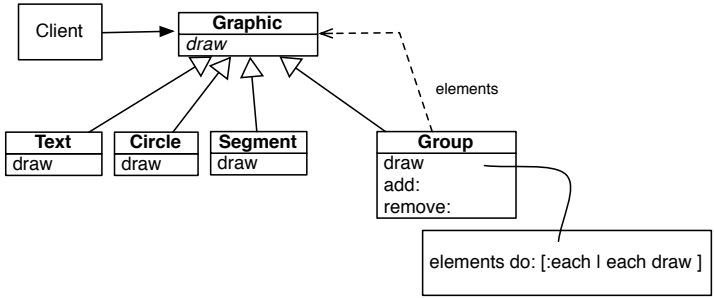
Composite participants: Component



Component (Graphic)

- declares the interface for objects in the composition
- may implement default behavior for common interfaces
- may declare an interface for accessing and managing its child components

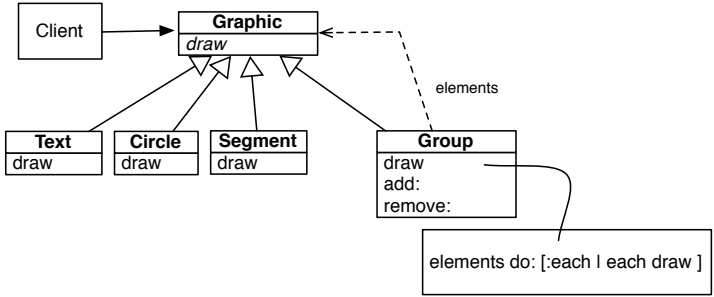
Composite participants: Leaf



Leaf (Circle, Segment, Text, ...)

- represents leaf objects in the composition. A leaf has no children
- defines behavior for primitive objects in the composition

Composite participants: Composite



Composite (Group)

- defines behaviour for components having children
- stores child components
- implements child-related operations (add/remove...)

Collaborations

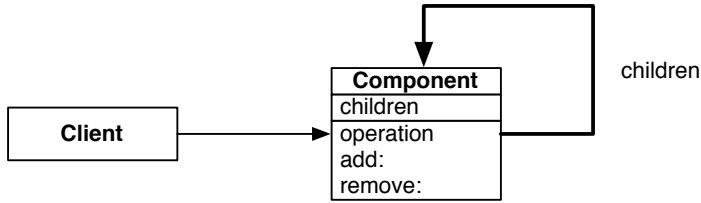
- Clients use the Component class interface to interact with objects in the composite structure
- Leaves handle requests directly
- Composites forward requests to its child components

Composite consequences

- Defines class hierarchies consisting of primitive and composite objects
- Makes the client simple. Composite and leaves objects are treated uniformly
- Eases the creation of new kinds of components
- Can make your design general

Alternate extreme implementation

- Remember a Design Pattern is a name + intent
- It can have multiple implementations



In a dynamically-typed language

- Composite not only groups leaves but can also contain composites
- In addition add:, remove: do not need to be declared into Component but only on Composite. This way we avoid to have to define dummy behavior for Leaf

In a dynamically-typed languages

- Can Composite contain any type of child? (domain issues)
- Is the Composites number of children limited?
- Forward/Delegation
  - Simple forward. Send the message to all the children and merge the results without performing any other behavior
  - Selective forward. Conditionally forward to some children
  - Extended forward. Extra behavior
  - Override. Instead of delegating

Working well with

Composite and Visitors: Visitors walk on structured recursive objects: composites
Composite and Factories Factories can create composite elements

Conclusion

- Composite is a natural way of composing
- Composite provide uniform API
- Basis for complex treatment expressed as Visitor

A course by

S. Ducasse, L. Fabresse, G. Polito, and Pablo Tesone