Composite Design Pattern

The Story So Far - Creational Design Patterns

Abstract Factory	Creates an instance of several families of classes
Factory Method	Creates an instance of several derived classes
Builder	Separates object construction from its representation
Singleton	A class of which only a single instance can exist
Prototype	A fully initialized instance to be copied or cloned

Structural Design Patterns

Structural patterns are concerned with how classes and objects are composed to form larger structures.

Adapter	structural *class* pattern
Composite	structural *object* pattern
Decorator	structural *object* pattern

Behavioral Design Patterns

Behavioral patterns are concerned with the assignment of responsibilities. They describe the patterns of communication between objects.

Observer	maintains a dependency between objects
State	Object changes its behavior as the state
Chain of Responsibility	Routes requests through a chain of candidate objects

Structural Design Patterns

▸▸ Adapter	structural *class* pattern
▸▸ Composite	structural *object* pattern
Decorator	structural *object* pattern

Composite Pattern

Intent

Compose objects to represent part-whole hierarchies.

Let clients treat individual objects and compositions of objects uniformly.

Composite Pattern

Motivation

Recursive composition/containment

Graphics elements in Google Drawing tool.

Files and directories in UNIX file systems.

Non-recursive, dynamic composition

Chrome Web browser - tabs and frame windows.

Composite Pattern

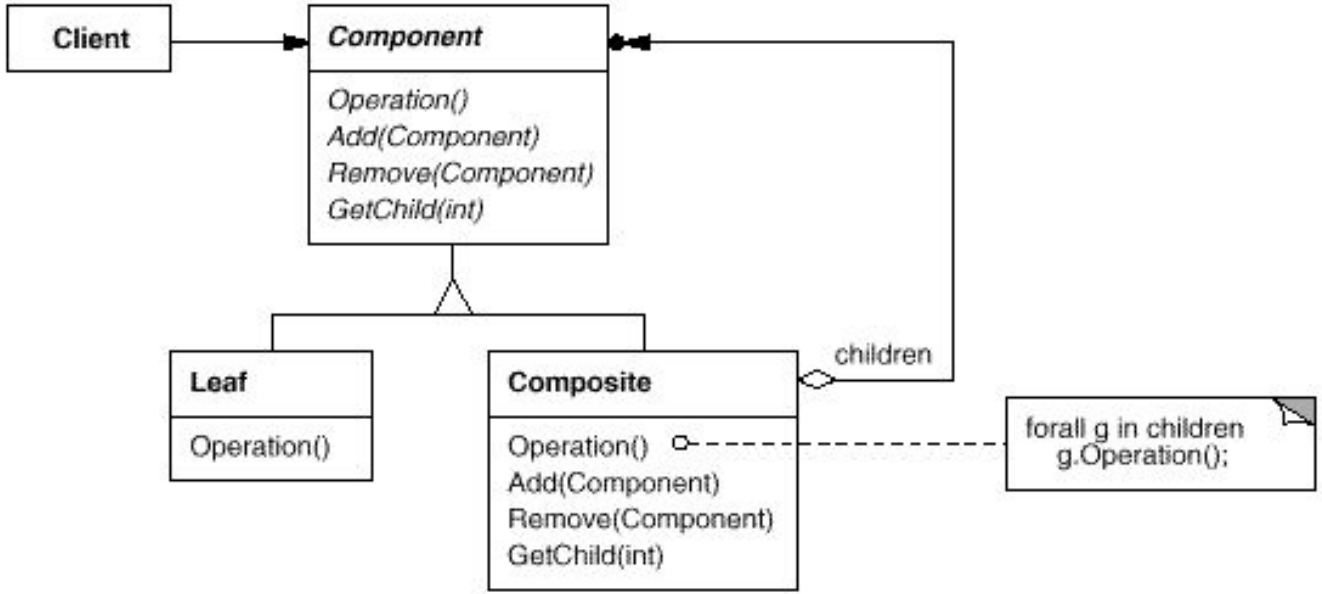
Applicability

Use the Composite pattern when

- you want to represent part-whole hierarchies of objects.
- you want clients to be able to ignore the difference between compositions of objects and individual objects.
- Clients need to treat all objects in the composite structure uniformly.

Composite Pattern

Structure



Composite Pattern

Participants

Component (Graphic/File/Window)

- declares the interface for objects in the composition.
- implements default behavior for the interface common to all classes, as appropriate.
- declares an interface for accessing and managing its child components.
- (optional) defines an interface for accessing a component's parent in the recursive structure.

Composite Pattern

Participants

Leaf (Rectangle, Line, Text/RegularFile, DeviceFile/Tab)

represents leaf objects in the composition. A leaf has no children.

defines behavior for primitive objects in the composition.

Composite (Picture/Directory/Frame)

defines behavior for components having children.

stores child components.

implements child-related operations in the Component interface.

Composite Pattern

Participants

Client

Manipulates objects in the composition using the Component interface.

Collaboration

Clients use the Component interface for interaction.

recipient is a Leaf - handle the request directly.

recipient is a Composite - forward requests to child components.

Composite may also perform operations before/after forwarding.

Composite Pattern

Consequences

defines class hierarchies of primitive objects and composite objects.

simplifies the Client.

clients can treat composite structures and individual objects uniformly.

makes it easier to add new kinds of components.

Composite Pattern

Implementation

Explicit parent references.

Sharing components.

Maximizing the Component interface.

What's the best data structure for storing components?

Child ordering.

Who should delete components?

Composite Pattern

Sample Code

Check the code in the *src* directory.

Known Uses