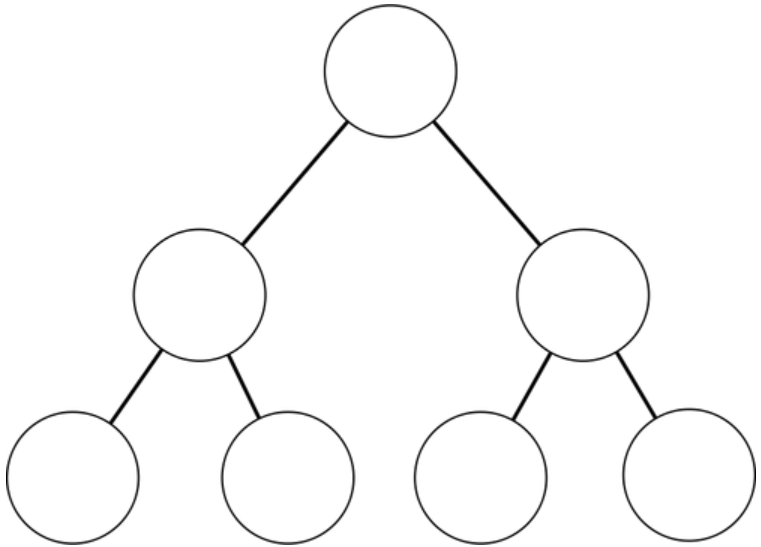
Composite Design Pattern

Presentation by: Artur, Mike, Evan



Definition

The Composite Pattern is a structural design pattern that allows you to compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly. It allows you to have a tree structure and ask each node in the tree structure to perform a task.

These structures are made up of 4 things:

Component

Leaf

Container, and

Clients

Component

Component is an interface which describes operations that are in common to both simple and complex elements of the tree.

Leaf

A Leaf is a basic element of the tree that has no sub-elements

Container

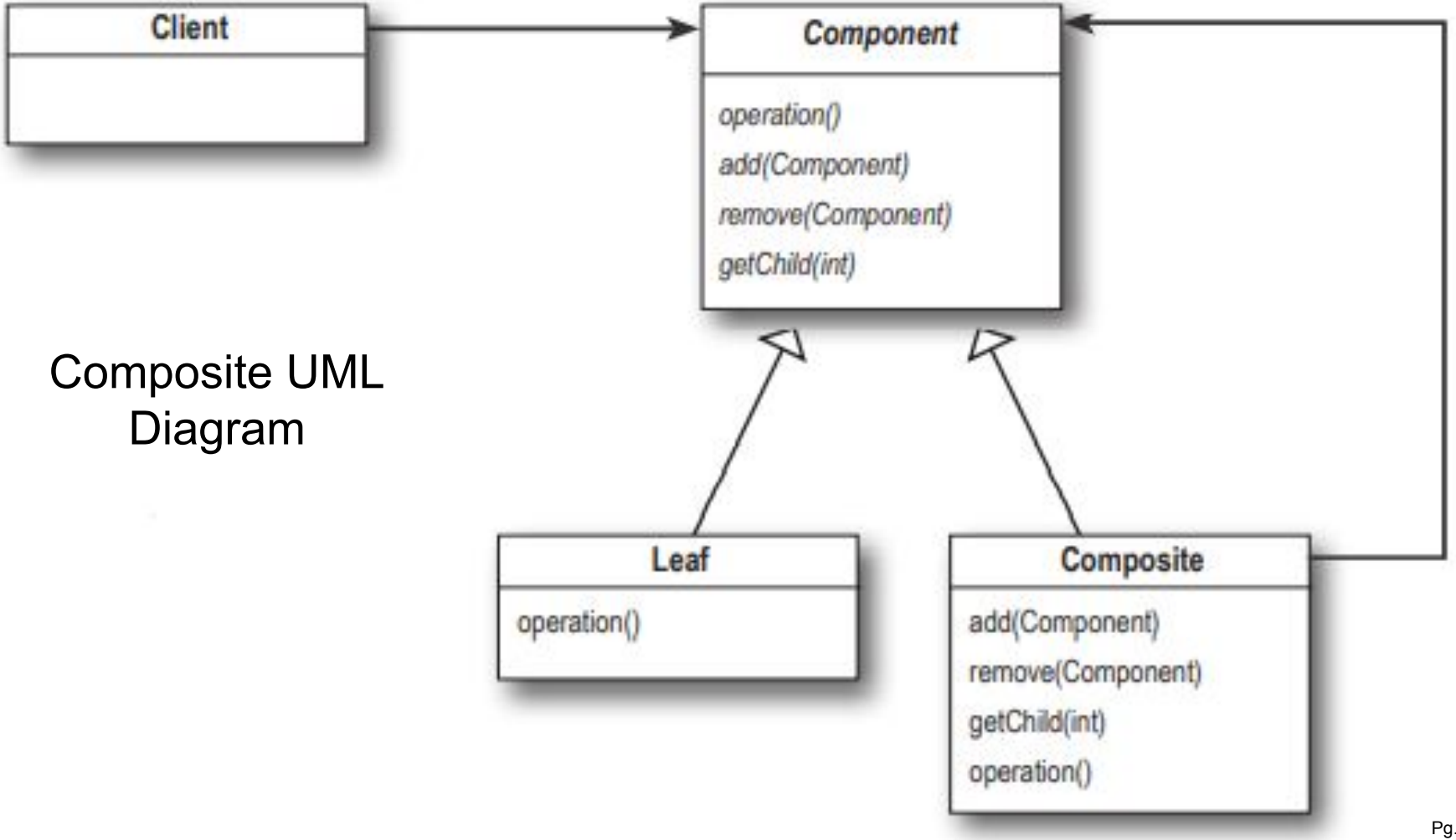
Also known as the composite, a container is an element with sub-elements. These other elements can either be other containers or leaves.

A container doesn’t know the concrete classes of its children, it only work with the sub-elements via the component interface.

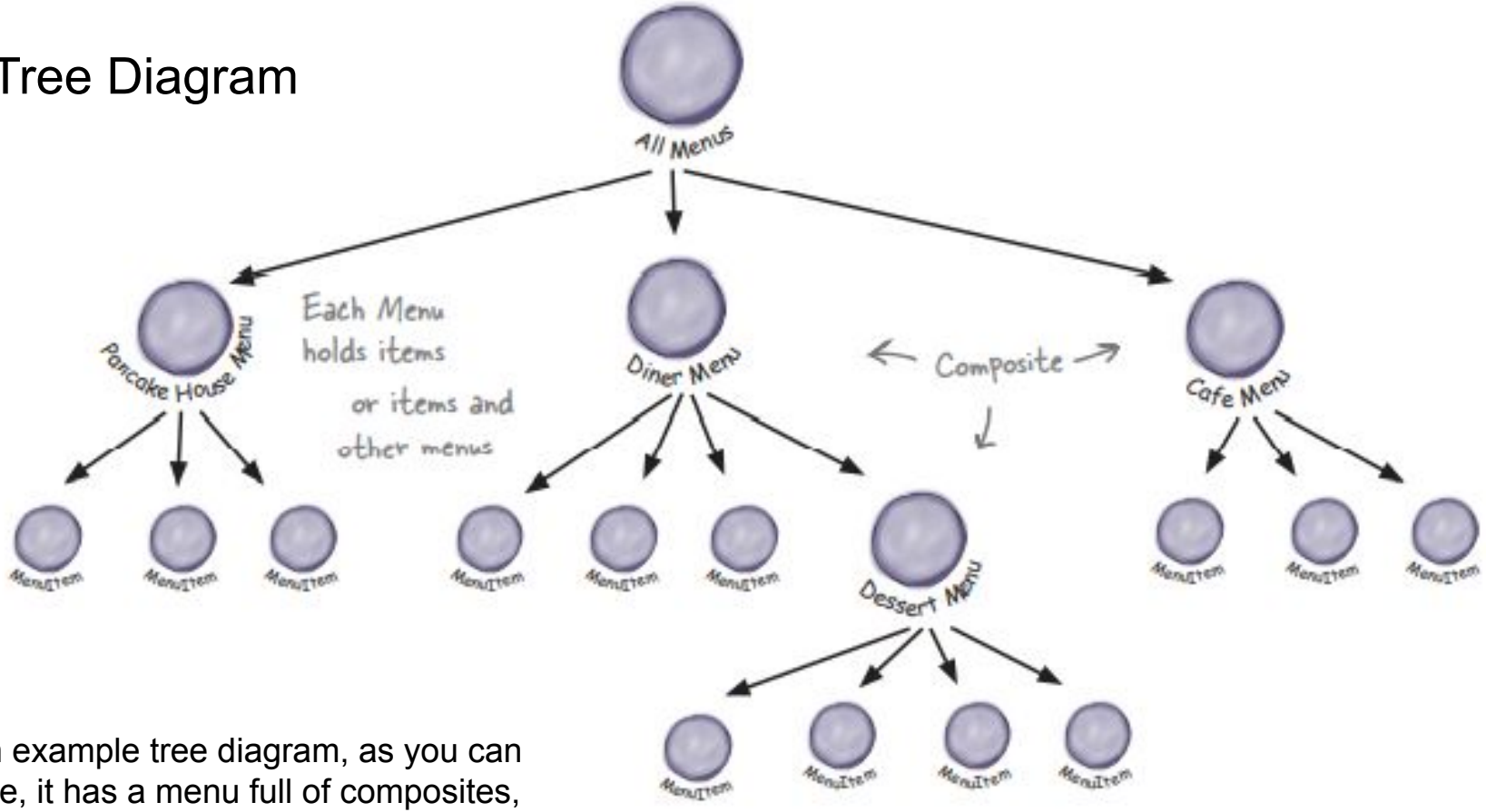
Client

A Client works with all elements through the component interface, and because of this can work in the same way with both simple and complex elements of a tree.

Composite UML Diagram



Tree Diagram



An example tree diagram, as you can see, it has a menu full of composites, and those are full of leaves and/or other composites.

Real world Examples

Think about a company. A company may have the CEO or head of a company. Under them could be general managers, and under them employee workers or developers.

An army works well too. An army consists of several divisions, each division is a set of brigades, each brigade consists of platoons, which can be broken down into squad of soldiers. Orders from the top are passed down until every soldier knows what needs to be done.

Benefits of the Composite Design Pattern

Tree data structure

Polymorphism

Recursion

Open principle

Memory usage

Iterator traversal

Drawbacks of the Composite Design Pattern

Class similarity has to be close

Potentially over-generalized

Use Case / Intent

The intent of the Composite pattern is to let you compose objects into tree structures and then work with these structures as if they were individual objects.

Use the pattern when you want the client code to treat both simple and complex elements uniformly, or when clients need to ignore the differences between compositions of objects and individual objects. So if you find you are using multiple objects in the same way and have near identical code to handle them, then composite is a good choice.

Demo

Relations with other patterns

Iterator pattern can be used to traverse Composite trees

Composite and Decorator have similar structures since they both rely on recursive composition for organization

- Decorator adds additional responsibilities to the wrapped object, while composite sums up its childrens results
- Decorator can thus extend the behavior of a specific object in the Composite tree

Sources

“Composite Design Pattern.” *GeeksforGeeks*, 1 Sept. 2021, https://www.geeksforgeeks.org/composite-design-pattern/.

“Composite Pattern.” *Refactoring.Guru*, https://refactoring.guru/design-patterns/composite#:~:text=The%20Composite%20pattern%20lets%20you,product%20or%20a%20sophisticated%20box.

Freeman, Eric, and Elisabeth Freeman. “Chapter 9.” *Head First Design Patterns*, O'Reilly Media, Sebastopol, CA, 2005.

Multiple Choice Questions

In the composite design pattern, what does the container/composite do?

A. It is an element with sub-elements
B. It works with all elements through the component interface
C. It is the basic element of a tree without sub-elements
D. It is an interface which describes operations that are in common to both simple and complex elements

Question 2

What data structure is used with the Composite Design Pattern?

A. Linked List
B. Tree
C. Array
D. Hashmap

Question 3

A Container/Composite holds:

A. Only Leaves
B. Only Containers/Composites
C. Both Leaves and Containers/Composites
D. The Client
E. water

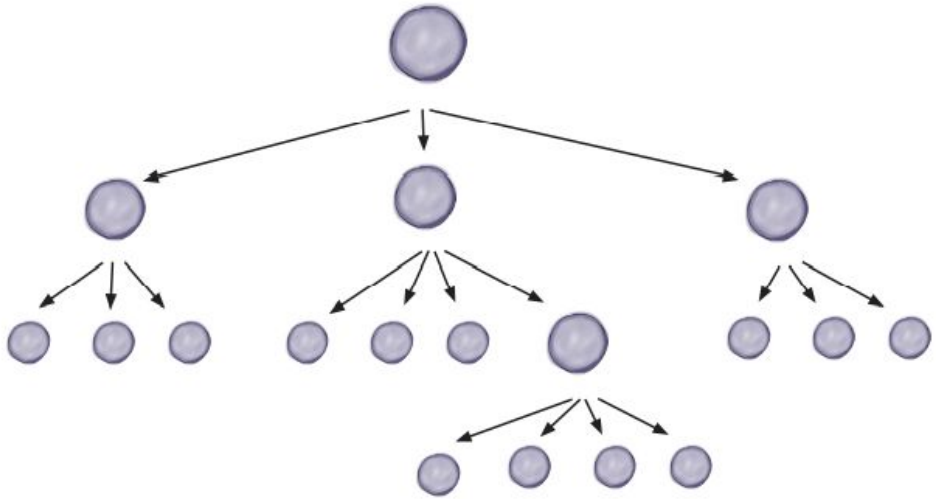
Question 4

Which of these are benefits of the composite design pattern?

A. Data Tree Structure
B. Open Principle
C. Memory Usage
D. All of the above
E. None of the above

Question 5

How many composites are in this diagram?



A. 1 B. 4 C. 5 D. 13