The Composite Pattern is a structural design pattern that allows you to compose objects into tree-like structures to represent part-whole hierarchies

* The **Component** interface declares the operation method.
* The **Leaf** class represents the leaf nodes, implementing the Component interface.
* The **Composite** class represents the composite nodes, managing their children and implementing the Component interface.
* The **Client** class demonstrates how the composite pattern can be used to create complex structures and perform operations on them.

*package* com.govtech.viswa.designpatterns.composite;  
  
*import* java.util.ArrayList;  
*import* java.util.*List*;  
  
*/\*\*  
 \** ***@author*** *Sarav on 07 May 2024  
 \** ***@project*** *govtech  
 \** ***@package*** *com.govtech.viswa.designpatterns.composite  
 \** ***@class*** *Composite  
 \*/  
  
// Component  
interface Component* {  
 *void* operation();  
}  
  
*// Leaf  
class* Leaf *implements Component* {  
 *private* String name;  
  
 *public* Leaf(String name) {  
 *this*.name = name;  
 }  
  
 @Override  
 *public void* operation() {  
 System.*out*.println("Leaf: " + name + " - Operation performed");  
 }  
}  
  
*// Composite  
public class* Composite *implements Component* {  
 *private List*<*Component*> children = *new* ArrayList<>();  
  
 *public void* add(*Component* component) {  
 children.add(component);  
 }  
  
 *public void* remove(*Component* component) {  
 children.remove(component);  
 }  
  
 @Override  
 *public void* operation() {  
 System.*out*.println("Composite - Operation performed");  
 *for* (*Component* component : children) {  
 component.operation();  
 }  
 }  
}  
  
*// Client  
class* Client {  
 *public static void* main(String[] args) {  
 *// Create leaf nodes* Leaf leaf1 = *new* Leaf("Leaf 1");  
 Leaf leaf2 = *new* Leaf("Leaf 2");  
 Leaf leaf3 = *new* Leaf("Leaf 3");  
  
 *// Create composite node and add leaf nodes* Composite composite = *new* Composite();  
 composite.add(leaf1);  
 composite.add(leaf2);  
  
 *// Create another composite node and add leaf and composite nodes* Composite composite2 = *new* Composite();  
 composite2.add(leaf3);  
 composite2.add(composite); *// Adding a composite node as a child  
  
 // Perform operation on composite node* composite2.operation();  
 }  
}

Composite - Operation performed

Leaf: Leaf 3 - Operation performed

Composite - Operation performed

Leaf: Leaf 1 - Operation performed

Leaf: Leaf 2 - Operation performed