## 组合（Composite）

### Intent

将对象组合成树形结构来表示“整体/部分”层次关系，允许用户以相同的方式处理单独对象和组合对象。

### Class Diagram

组件（Component）类是组合类（Composite）和叶子类（Leaf）的父类，可以把组合类看成是树的中间节点。

组合对象拥有一个或者多个组件对象，因此组合对象的操作可以委托给组件对象去处理，而组件对象可以是另一个组合对象或者叶子对象。

### Implementation

public abstract class Component {  
 protected String name;  
  
 public Component(String name) {  
 this.name = name;  
 }  
  
 public void print() {  
 print(0);  
 }  
  
 abstract void print(int level);  
  
 abstract public void add(Component component);  
  
 abstract public void remove(Component component);  
}

public class Composite extends Component {  
  
 private List<Component> child;  
  
 public Composite(String name) {  
 super(name);  
 child = new ArrayList<>();  
 }  
  
 @Override  
 void print(int level) {  
 for (int i = 0; i < level; i++) {  
 System.out.print("--");  
 }  
 System.out.println("Composite:" + name);  
 for (Component component : child) {  
 component.print(level + 1);  
 }  
 }  
  
 @Override  
 public void add(Component component) {  
 child.add(component);  
 }  
  
 @Override  
 public void remove(Component component) {  
 child.remove(component);  
 }  
}

public class Leaf extends Component {  
 public Leaf(String name) {  
 super(name);  
 }  
  
 @Override  
 void print(int level) {  
 for (int i = 0; i < level; i++) {  
 System.out.print("--");  
 }  
 System.out.println("left:" + name);  
 }  
  
 @Override  
 public void add(Component component) {  
 throw new UnsupportedOperationException(); // 牺牲透明性换取单一职责原则，这样就不用考虑是叶子节点还是组合节点  
 }  
  
 @Override  
 public void remove(Component component) {  
 throw new UnsupportedOperationException();  
 }  
}

public class Client {  
 public static void main(String[] args) {  
 Composite root = new Composite("root");  
 Component node1 = new Leaf("1");  
 Component node2 = new Composite("2");  
 Component node3 = new Leaf("3");  
 root.add(node1);  
 root.add(node2);  
 root.add(node3);  
 Component node21 = new Leaf("21");  
 Component node22 = new Composite("22");  
 node2.add(node21);  
 node2.add(node22);  
 Component node221 = new Leaf("221");  
 node22.add(node221);  
 root.print();  
 }  
}

Composite:root  
--left:1  
--Composite:2  
----left:21  
----Composite:22  
------left:221  
--left:3

### JDK

* javax.swing.JComponent#add(Component)
* java.awt.Container#add(Component)
* java.util.Map#putAll(Map)
* java.util.List#addAll(Collection)
* java.util.Set#addAll(Collection)