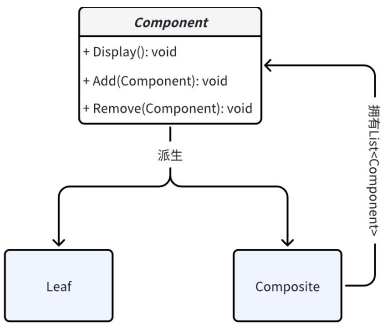
组合模式

概念

- 组合模式（Composite Pattern）：一种结构型设计模式，它允许将对象组合成树形结构来表示“部分-整体”的层次结构。组合模式使得客户端对单个对象和组合对象的使用具有一致性。
- 设计意图：
  - 客户端可以一致地处理单个对象和组合对象。
  - 通过递归组合的方式来构建复杂的对象结构。
- 组合模式适用于以下场景：
  - 需要表示对象的“部分-整体”层次结构时，例如文件系统、组织结构、UI组件树等。
  - 希望客户端可以忽略组合对象与单个对象的差异，统一地使用组合结构中的所有对象时。
- 组合模式主要包含以下角色：
  - Component（组件）：定义组合对象和叶子对象的接口。
  - Leaf（叶子）：表示组合中的叶子对象，没有子对象。
  - Composite（组合）：表示有子对象的组合对象，定义子对象的管理方法。



- 优点：
  - 简化客户端代码，使得客户端可以一致地处理组合对象和单个对象。
  - 更容易添加新的类型的组件。
- 缺点：
  - 使得设计变得更加复杂，特别是在需要管理大量对象时。
  - 可能会导致系统中的对象数量增加。

实例

- 抽象类：Component

```
abstract class Component
{
    protected string name;

    public Component(string name)
    {
        this.name = name;
    }

    public abstract void Add(Component c);
    public abstract void Remove(Component c);
    public abstract void Display(int depth);
}
```

- 叶子类：Leaf

```
class Leaf : Component
{
    public Leaf(string name) : base(name) { }

    public override void Add(Component c)
    {
        Console.WriteLine("Cannot add to a leaf");
    }

    public override void Remove(Component c)
    {
        Console.WriteLine("Cannot remove from a leaf");
    }

    public override void Display(int depth)
    {
        Console.WriteLine(new String('-', depth) + name);
    }
}
```

- 组合类：Composite

```
class Composite : Component
{
```

```
    private List<Component> children = new List<Component>();

    public Composite(string name) : base(name) { }

    public override void Add(Component component)
    {
        children.Add(component);
    }

    public override void Remove(Component component)
    {
        children.Remove(component);
    }

    public override void Display(int depth)
    {
        Console.WriteLine(new String('-', depth) + name);

        foreach (Component component in children)
        {
            component.Display(depth + 2);
        }
    }
}
```

- 实例：

```
Composite root = new Composite("root");
root.Add(new Leaf("Leaf A"));
root.Add(new Leaf("Leaf B"));

Composite comp = new Composite("Composite X");
comp.Add(new Leaf("Leaf XA"));
comp.Add(new Leaf("Leaf XB"));

root.Add(comp);
root.Add(new Leaf("Leaf C"));

Leaf leaf = new Leaf("Leaf D");
root.Add(leaf);
root.Remove(leaf);

root.Display(1);
```

- 打印结果

```
-root
--Leaf A
--Leaf B
--Composite X
----Leaf XA
----Leaf XB
--Leaf C
```