

# 组合模式

组合模式（**Composite Pattern**），又叫部分整体模式，是用于把一组相似的对象当作一个单一的对象。组合模式依据树形结构来组合对象，用来表示部分以及整体层次。这种类型的设计模式属于结构型模式，它创建了对象组的树形结构。

这种模式创建了一个包含自己对象组的类。该类提供了修改相同对象组的方式。

我们通过下面的实例来演示组合模式的用法。实例演示了一个组织中员工的层次结构。

## 介绍

意图：将对象组合成树形结构以表示"部分-整体"的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性。

主要解决：它在我们树型结构的问题中，模糊了简单元素和复杂元素的概念，客户程序可以向处理简单元素一样来处理复杂元素，从而使得客户程序与复杂元素的内部结构解耦。

何时使用： 1、您想表示对象的部分-整体层次结构（树形结构）。 2、您希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。

如何解决：树枝和叶子实现统一接口，树枝内部组合该接口。

关键代码：树枝内部组合该接口，并且含有内部属性 **List**，里面放 **Component**。

应用实例： 1、算术表达式包括操作数、操作符和另一个操作数，其中，另一个操作符也可以是操作数、操作符和另一个操作数。

2、在 **JAVA AWT** 和 **SWING** 中，对于 **Button** 和 **Checkbox** 是树叶，**Container** 是树枝。

优点： 1、高层模块调用简单。 2、节点自由增加。

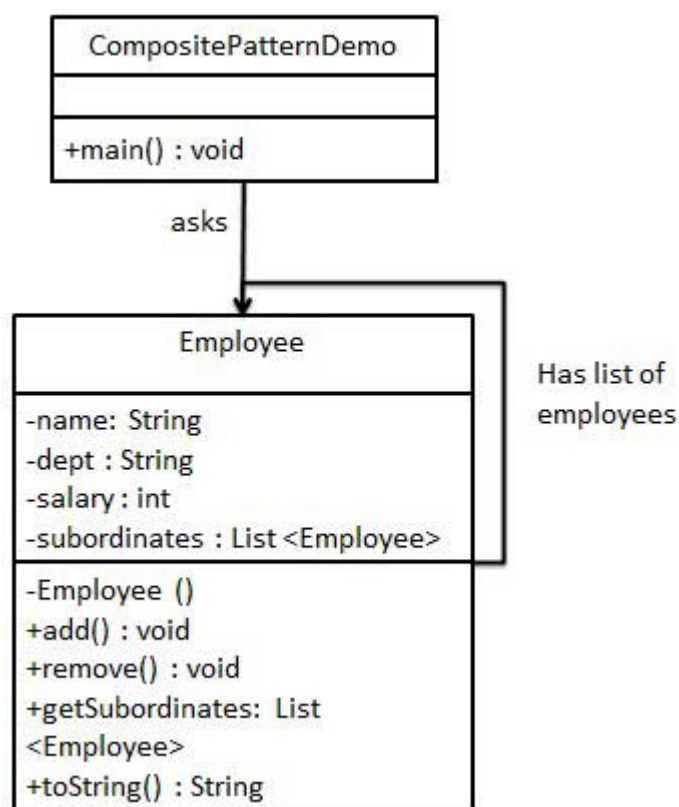
缺点：在使用组合模式时，其叶子和树枝的声明都是实现类，而不是接口，违反了依赖倒置原则。

使用场景：部分、整体场景，如树形菜单，文件、文件夹的管理。

注意事项：定义时为具体类。

## 实现

我们有一个类 **Employee**，该类被当作组合模型类。**CompositePatternDemo**，我们的演示类使用 **Employee** 类来添加部门层次结构，并打印所有员工。



## 步骤 1

创建 *Employee* 类，该类带有 *Employee* 对象的列表。

### Employee.java

```
import java.util.ArrayList;
import java.util.List;

public class Employee {
    private String name;
    private String dept;
    private int salary;
    private List<Employee> subordinates;

    //构造函数
    public Employee(String name,String dept, int sal) {
        this.name = name;
        this.dept = dept;
        this.salary = sal;
        subordinates = new ArrayList<Employee>();
    }

    public void add(Employee e) {
        subordinates.add(e);
    }

    public void remove(Employee e) {
        subordinates.remove(e);
    }

    public List<Employee> getSubordinates(){
        return subordinates;
    }
}
```

```

public String toString(){
    return ("Employee :[ Name : "+ name
    +", dept : "+ dept + ", salary : "
    + salary+" ]");
}
}

```

## 步骤 2

使用 *Employee* 类来创建和打印员工的层次结构。

### CompositePatternDemo.java

```

public class CompositePatternDemo {
    public static void main(String[] args) {
        Employee CEO = new Employee("John", "CEO", 30000);

        Employee headSales = new Employee("Robert", "Head Sales", 20000);

        Employee headMarketing = new Employee("Michel", "Head Marketing", 20000);

        Employee clerk1 = new Employee("Laura", "Marketing", 10000);
        Employee clerk2 = new Employee("Bob", "Marketing", 10000);

        Employee salesExecutive1 = new Employee("Richard", "Sales", 10000);
        Employee salesExecutive2 = new Employee("Rob", "Sales", 10000);

        CEO.add(headSales);
        CEO.add(headMarketing);

        headSales.add(salesExecutive1);
        headSales.add(salesExecutive2);

        headMarketing.add(clerk1);
        headMarketing.add(clerk2);

        //打印该组织的所有员工
        System.out.println(CEO);
        for (Employee headEmployee : CEO.getSubordinates()) {
            System.out.println(headEmployee);
            for (Employee employee : headEmployee.getSubordinates()) {
                System.out.println(employee);
            }
        }
    }
}

```

## 步骤 3

执行程序，输出结果为：

```

Employee :[ Name : John, dept : CEO, salary :30000 ]
Employee :[ Name : Robert, dept : Head Sales, salary :20000 ]
Employee :[ Name : Richard, dept : Sales, salary :10000 ]
Employee :[ Name : Rob, dept : Sales, salary :10000 ]
Employee :[ Name : Michel, dept : Head Marketing, salary :20000 ]

```

```
Employee :[ Name : Laura, dept : Marketing, salary :10000 ]  
Employee :[ Name : Bob, dept : Marketing, salary :10000 ]
```

- 1、组合模式，就是在一个对象中包含其他对象，这些被包含的对象可能是终点对象（不再包含别的对象），也有可能是非终点对象（其内部还包含其他对象，或叫组对象），我们将对象称为节点，即一个根节点包含许多子节点，这些子节点有的不再包含子节点，而有的仍然包含子节点，以此类推。
- 2、所谓组合模式，其实说的是对象包含对象的问题，通过组合的方式（在对象内部引用对象）来进行布局，我认为这种组合是区别于继承的，而另一层含义是指树形结构子节点的抽象（将叶子节点与数枝节点抽象为子节点），区别于普通的分别定义叶子节点与数枝节点的方式。