Design Patterns - Composite Pattern

设计模式-组合模式

Composite pattern is used where we need to treat a group of objects in similar way as a single object. Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchy. This type of design pattern comes under structural pattern as this pattern creates a tree structure of group of objects.

This pattern creates a class that contains group of its own objects. This class provides ways to modify its group of same objects.

We are demonstrating use of composite pattern via following example in which we will show employees hierarchy of an organization.

当我们需要以相似的方式对待一组的对象和一个对象的时候使用组合模式。组合模式按照树形结构来组织对象表示部分与整体之间的层次结构。这种类型的模式属于结构型模式的一种，它创建一组树形结构的对象。

该模式将创建一个包含一组自己对象的类。该类提供方法修改这组相同的对象。

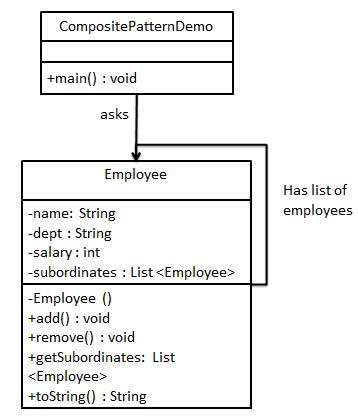
我们将通过下面的例子来展示一个组织的员工层次结构。

Implementation

实现

We have a class *Employee* which acts as composite pattern actor class.*CompositePatternDemo*, our demo class will use *Employee* class to add department level hierarchy and print all employees.

我们有一个Employee类，该类扮演组合的角色。*CompositePatternDemo*,我们的demo类将使用Employee类来 增加部门级别的层次结构，并且打印所有的员工。



Step 1

Create *Employee* class having list of *Employee* objects.

创建一个Employee类，该类持有一个Employee对象集合。

*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;

// constructor

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+" ]");

}

}

Step 2

Use the *Employee* class to create and print employee hierarchy.

使用这个Employee类，创建并且打印employee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);

//print all employees of the organization

System.out.println(CEO);

for (Employee headEmployee : CEO.getSubordinates()) {

System.out.println(headEmployee);

for (Employee employee : headEmployee.getSubordinates()) {

System.out.println(employee);

}

}

}

}

Step 3

Verify the output.

校验输出。

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 ]