组合模式实验

**【实验目的及要求】**

1. 掌握组合模式的结构与实现；
2. 学会使用组合模式开发应用程序；
3. 明白组合模式的扩展应用。

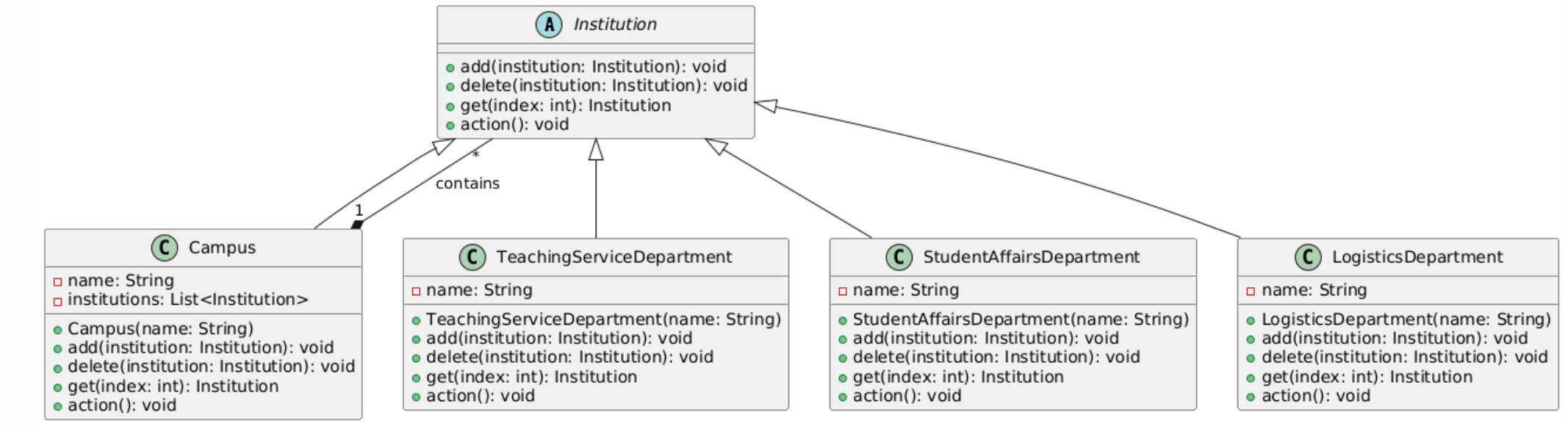
**【实验内容】**

**广东金融学院有多个校区：本部、肇庆校区和清远校区，校区都要教学处（部）、学工处（部）、后勤处（部），每个机构都有自己的职责工作。请使用组合模式完成学校、校区和部门之间的结构。**

**【实验步骤】8chao**

1. **类图**

**根据下面的代码画出类图。**

****

1. **实现代码**

**(1)抽象构件**

**Institution类**

**package** design.composite;

**public** **abstract** **class** Institution {

**public** **abstract** **void** add(Institution institution);

**public** **abstract** **void** delete(Institution institution);

**public** **abstract** Institution get(**int** index);

**public** **abstract** **void** action();//机构行使职责工作

}

**(2)容器构件**

**Campus类（校园类）**

**package** design.composite;

**import** java.util.ArrayList;

**import** java.util.List;

**public class Campus extends Institution {  
 private String name;  
 private List<Institution> institutions = new ArrayList<Institution>();  
  
 public Campus(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void add(Institution institution) {  
 institutions.add(institution);  
 }  
  
 @Override  
 public void delete(Institution institution) {  
 institutions.remove(institution);  
  
 }  
  
 @Override  
 public Institution get(int index) {  
 return (Institution) institutions.get(index);  
 }  
  
 @Override  
 public void action() {  
 System.*out*.println("----------"+name+"------------");  
 for (Institution institution : institutions) {  
 // 只对 Campus 类型的对象进行 action 调用  
 if (institution instanceof Campus) {  
 ((Campus) institution).action(); // 这里是安全的强制转换  
 } else {  
 institution.action(); // 对叶子节点直接调用 action  
 }  
 }  
  
 }  
}**

**(3)叶子构件**

**TeachingServiceDepartment类（教学处（部））**

**package** design.composite;

**public** **class** TeachingServiceDepartment **extends** Institution {

**private** String name;

**public** TeachingServiceDepartment(String name) {

**this**.name = name;

}

@Override  
public void add(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
}  
  
@Override  
public void delete(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
}  
  
@Override  
public Institution get(int index) {  
 System.*out*.println("对不起，不支持该方法");  
 return null;  
}  
  
@Override  
public void action() {  
 System.*out*.println("\*\*\*"+name+"教学处（部)的职责是负责教学管理工作");  
}

}

**StudentAffairsDepartment类（学工处（部））**

**public class StudentAffairsDepartment extends Institution {  
 private String name;  
  
 public StudentAffairsDepartment(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void add(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
 }  
  
 @Override  
 public void delete(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
 }  
  
 @Override  
 public Institution get(int index) {  
 System.*out*.println("对不起，不支持该方法");  
 return null;  
 }  
  
 @Override  
 public void action() {  
 System.*out*.println("\*\*\*"+name+"学工处（部)的职责是负责学工管理工作");  
 }  
  
}**

**LogisticsDepartment类（后勤处（部））**

**public class LogisticsDepartment extends Institution {  
 private String name;  
  
 public LogisticsDepartment(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void add(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
 }  
  
 @Override  
 public void delete(Institution institution) {  
 System.*out*.println("对不起，不支持该方法");  
 }  
  
 @Override  
 public Institution get(int index) {  
 System.*out*.println("对不起，不支持该方法");  
 return null;  
 }  
  
 @Override  
 public void action() {  
 System.*out*.println("\*\*\*"+name+"后勤处（部)的职责是负责后勤管理工作");  
 }  
  
}**

**(5)测试类**

**Test**

**package** design.composite;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Institution gduf;//学校

Institution gzCampus,zqCampus,qyCampus;//校区

//各部门

Institution gzTeachinService,gzStuAffDepart,gzLogistics;

Institution zqTeachinService,zqStuAffDepart,zqLogistics;

Institution qyTeachinService,qyStuAffDepart,qyLogistics;

gduf = **new** Campus("广东金融学院");

gzCampus = **new** Campus("本部");

zqCampus = **new** Campus("肇庆校区");

qyCampus = **new** Campus("清远校区");

gzTeachinService = **new** TeachingServiceDepartment("教务处（部）");

gzStuAffDepart = **new** StudentAffairsDepartment("学生（处）部");

gzLogistics = **new** LogisticsDepartment("后勤处（部）");

zqTeachinService = **new** TeachingServiceDepartment("教务处（部）");

zqStuAffDepart = **new** StudentAffairsDepartment("学生（处）部");

zqLogistics = **new** LogisticsDepartment("后勤处（部）");

qyTeachinService = **new** TeachingServiceDepartment("教务处（部）");

qyStuAffDepart = **new** StudentAffairsDepartment("学生（处）部");

qyLogistics = **new** LogisticsDepartment("后勤处（部）");

//对象之间建立关联

gzCampus.add(gzTeachinService);

gzCampus.add(gzStuAffDepart);

gzCampus.add(gzLogistics);

zqCampus.add(zqTeachinService);

zqCampus.add(zqStuAffDepart);

zqCampus.add(zqLogistics);

qyCampus.add(qyTeachinService);

qyCampus.add(qyStuAffDepart);

qyCampus.add(qyLogistics);

gduf.add(gzCampus);

gduf.add(zqCampus);

gduf.add(qyCampus);

gduf.action();

}

}

**【问题】**

1. 修改上面代码，分别转换成透明组合模式与安全组合模式。

组合模式分为透明组合模式和安全组合模式，由于上面代码就是透明组合模式，下面只给出安全组合模式的代码。

// 抽象构件

public abstract class Institution {

public abstract void action(); // 机构行使职责工作

}

// 容器构件：Campus 类

public class Campus extends Institution {

private String name;

private List<Institution> institutions = new ArrayList<>();

public Campus(String name) {

this.name = name;

}

public void add(Institution institution) {

institutions.add(institution);

}

public void delete(Institution institution) {

institutions.remove(institution);

}

public Institution get(int index) {

return institutions.get(index);

}

@Override

public void action() {

System.out.println("----------" + name + "------------");

for (Institution institution : institutions) {

institution.action();

}

}

}

// 叶子构件：教学处（部）

public class TeachingServiceDepartment extends Institution {

private String name;

public TeachingServiceDepartment(String name) {

this.name = name;

}

@Override

public void action() {

System.out.println("\*\*\*" + name + "教学处（部）负责教学管理工作");

}

}

// 叶子构件：学工处（部）

public class StudentAffairsDepartment extends Institution {

private String name;

public StudentAffairsDepartment(String name) {

this.name = name;

}

@Override

public void action() {

System.out.println("\*\*\*" + name + "学工处（部）负责学工管理工作");

}

}

// 叶子构件：后勤处（部）

public class LogisticsDepartment extends Institution {

private String name;

public LogisticsDepartment(String name) {

this.name = name;

}

@Override

public void action() {

System.out.println("\*\*\*" + name + "后勤处（部）负责后勤管理工作");

}

}

// 测试类

public class Test {

public static void main(String[] args) {

Institution gduf = new Campus("广东金融学院");

Institution gzCampus = new Campus("本部");

Institution zqCampus = new Campus("肇庆校区");

Institution qyCampus = new Campus("清远校区");

Institution gzTeachinService = new TeachingServiceDepartment("教务处（部）");

Institution gzStuAffDepart = new StudentAffairsDepartment("学工处（部）");

Institution gzLogistics = new LogisticsDepartment("后勤处（部）");

gzCampus.add(gzTeachinService);

gzCampus.add(gzStuAffDepart);

gzCampus.add(gzLogistics);

gduf.add(gzCampus);

gduf.add(zqCampus);

gduf.add(qyCampus);

gduf.action();

}

}

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | **实验总结：**  本次实验使用了**组合模式**来模拟广东金融学院的校区和部门结构。组合模式是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-整体层次结构。本实验中，Institution 类作为抽象构件，Campus 类作为容器构件，分别实现了部门（如教学处、学工处、后勤处）与校区（如本部、肇庆校区等）之间的层次关系。  通过组合模式，我们能够很方便地将不同的部门对象（叶子构件）组织到各个校区（容器构件）中，再通过调用统一的 action 方法，使每个对象按其职责行使工作。容器类 Campus 可以包含多个部门或子校区，并且提供了 add、delete 和 get 等操作方法，而叶子类（如各个部门）则只需要实现 action 方法来执行具体的工作。该模式的优点在于使得客户端代码处理机构层次结构时，不需要区分容器类和叶子类，统一而简洁地进行管理。  实验通过实现透明组合模式，使得客户端可以通过同一个接口统一操作所有机构对象。通过本实验，我更深入理解了组合模式的实现方式及其应用场景，并掌握了如何使用该模式简化复杂结构的管理。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |