composite模式：递归算法的实现

场景：

（1）父部门->子部门->子部门

（2）我们要对部门树，删除一个父部门以及其下所有的子部门

public abstract **class Component** {

   public abstract void operate();

   public void addChild(Component child) {

  }

   public void removeChild(Component child) {

  }

   public Component getChildren(int index) {

  }

}

【非叶子节点】

public class **Composite extends Component** {

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

   public void addChild(Component child) {

       children.add(child);

  }

   public abstract void removeChild(Component child) {

       children.remove(child);

  }

   public Component getChild(int index) {

       children.get(index);

  }

   public void operate() {

       for(Component component : children) {

           component.operate();

      }

  }

}

【叶子节点】

public class **Leaf extends Component** {

   public void operate() {

       System.out.println("执行具体功能");

  }

}

组合模式非常适用于有较多父子层级关系的那种场景，比如部门树的递归删除，或者是文件树的递归删除之类的场景。这个还是比较好模拟的，我们会在电商系统的权限那块给大家演示，比如删除一个父权限，就要级联递归删除所有的子权限，这个时候可以对组合模式做一点变种，自动从根权限开始，递归查询所有层级的子权限，然后一直到叶子节点的权限，从叶子节点开始删除

853769620