山东大学 计算机科学与技术 学院

软件工程 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800130067 | 姓名：黄一鸣 | | 班级：二班 |
| 实验题目： 实验十 设计模块+编写程序（文档） | | | |
| 实验学时：2 | | 实验日期：2021.6.1 | |
| 实验目的：  1. 练习设计模式应用  2. 掌握编程的标准和过程。  3. 编程指导原则：讨论程序编写方式与规则：  4. 文档：内部文档、外部文档  5. 记录项目及小组的工作进度。 | | | |
| 硬件环境：  laptop | | | |
| 软件环境：  ubuntu | | | |
| 实验内容与设计：  . 1. 设计模式。  分工协作，继续编写涉及以下3种设计模式的例子（语言不限），并总结其特点。  （1）观察者模式  （2）组合模式  （3）访问者模式    2. 结合自己的项目，指出其中的数据管理、异常处理、UI方面的设计；它是否使用了框架？  3.上网查询“阿里 编程规范 （如：阿里巴巴JAVA开发手册）；华为 编程军规”等，对照自己的代码看有哪些不符合规范的地方，修改。  4. 参考软件文档编制案例，分工协作写出所负责系统的详细设计报告（包括界面、接口等）。  5. 检查系统的内部文档：对每个代码文件检查其格式和注释；外部文档：描述问题、描述算法、描述数据。  6. 讨论：对编程过程的不同实施方式、方法有什么思考和评价。  7. 记录项目及小组的最新进度及工作量。  记录项目及小组每个人最新的工作的进度、里程碑、工作量的跟踪图或表。每周更新。  每人向组长汇报自己工作。组长汇总进度里程碑，提交小组共同报告。  。 | | | |
| 结论分析与体会： 访问者模式模式的定义与特点 访问者（Visitor）模式的定义：将作用于某种数据结构中的各元素的操作分离出来封装成独立的类，使其在不改变数据结构的前提下可以添加作用于这些元素的新的操作，为数据结构中的每个元素提供多种访问方式。它将对数据的操作与数据结构进行分离，是行为类模式中最复杂的一种模式。 优点  访问者（Visitor）模式是一种对象行为型模式，其主要优点如下。   1. 扩展性好。能够在不修改对象结构中的元素的情况下，为对象结构中的元素添加新的功能。 2. 复用性好。可以通过访问者来定义整个对象结构通用的功能，从而提高系统的复用程度。 3. 灵活性好。访问者模式将数据结构与作用于结构上的操作解耦，使得操作集合可相对自由地演化而不影响系统的数据结构。 4. 符合单一职责原则。访问者模式把相关的行为封装在一起，构成一个访问者，使每一个访问者的功能都比较单一。   缺点 访问者（Visitor）模式的主要缺点如下。   1. 增加新的元素类很困难。在访问者模式中，每增加一个新的元素类，都要在每一个具体访问者类中增加相应的具体操作，这违背了“开闭原则”。 2. 破坏封装。访问者模式中具体元素对访问者公布细节，这破坏了对象的封装性。 3. 违反了依赖倒置原则。访问者模式依赖了具体类，而没有依赖抽象类。  模式的结构与实现 访问者（Visitor）模式实现的关键是如何将作用于元素的操作分离出来封装成独立的类，其基本结构与实现方法如下。  1. 模式的结构  访问者模式包含以下主要角色。  抽象访问者（Visitor）角色：定义一个访问具体元素的接口，为每个具体元素类对应一个访问操作 visit() ，该操作中的参数类型标识了被访问的具体元素。  具体访问者（ConcreteVisitor）角色：实现抽象访问者角色中声明的各个访问操作，确定访问者访问一个元素时该做什么。  抽象元素（Element）角色：声明一个包含接受操作 accept() 的接口，被接受的访问者对象作为 accept() 方法的参数。  具体元素（ConcreteElement）角色：实现抽象元素角色提供的 accept() 操作，其方法体通常都是 visitor.visit(this) ，另外具体元素中可能还包含本身业务逻辑的相关操作。  对象结构（Object Structure）角色：是一个包含元素角色的容器，提供让访问者对象遍历容器中的所有元素的方法，通常由 List、Set、Map 等聚合类实现。  其结构图如图 1 所示。  [访问者（Visitor）模式的结构图](http://c.biancheng.net/uploads/allimg/181119/3-1Q119101429D6.gif) 图1 访问者（Visitor）模式的结构图 观察者模式一、定义 观察者（Observer）模式的定义：指多个对象间存在一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。这种模式有时又称作发布-订阅模式、模型-视图模式，它是对象行为型模式。 二、优点 观察者模式是一种对象行为型模式，其主要优点如下  1、降低了目标与观察者之间的耦合关系，两者之间是抽象耦合关系。符合依赖倒置原则。  2、目标与观察者之间建立了一套触发机制。 三、缺点  它的主要缺点如下。  1、目标与观察者之间的依赖关系并没有完全解除，而且有可能出现循环引用。  2、当观察者对象很多时，通知的发布会花费很多时间，影响程序的效率。 四、模式的结构与实现 实现观察者模式时要注意具体目标对象和具体观察者对象之间不能直接调用，否则将使两者之间紧密耦合起来，这违反了面向对象的设计原则。  模式的结构  观察者模式的主要角色如下。   1. 抽象主题（Subject）角色：也叫抽象目标类，它提供了一个用于保存观察者对象的聚集类和增加、删除观察者对象的方法，以及通知所有观察者的抽象方法。   2、具体主题（Concrete Subject）角色：也叫具体目标类，它实现抽象目标中的通知方法，当具体主题的内部状态发生改变时，通知所有注册过的观察者对象。  3、抽象观察者（Observer）角色：它是一个抽象类或接口，它包含了一个更新自己的抽象方法，当接到具体主题的更改通知时被调用。   1. 具体观察者（Concrete Observer）角色：实现抽象观察者中定义的抽象方法，以便在得到目标的更改通知时更新自身的状态。  五、使用场景 关联行为场景  事件多级触发场景  跨系统的消息变换场景，如消息队列的处理机制 组合模式一、定义 组合（Composite Pattern）模式的定义：有时又叫作整体-部分（Part-Whole）模式，它是一种将对象组合成树状的层次结构的模式，用来表示“整体-部分”的关系，使用户对单个对象和组合对象具有一致的访问性，属于结构型[设计模式](http://c.biancheng.net/design_pattern/" \t "_blank)。  组合模式一般用来描述整体与部分的关系，它将对象组织到树形结构中，顶层的节点被称为根节点，根节点下面可以包含树枝节点和叶子节点，树枝节点下面又可以包含树枝节点和叶子节点，树形结构图如下。  组合模式树形结构图  由上图可以看出，其实根节点和树枝节点本质上属于同一种数据类型，可以作为容器使用；而叶子节点与树枝节点在语义上不属于用一种类型。但是在组合模式中，会把树枝节点和叶子节点看作属于同一种数据类型（用统一接口定义），让它们具备一致行为。  这样，在组合模式中，整个树形结构中的对象都属于同一种类型，带来的好处就是用户不需要辨别是树枝节点还是叶子节点，可以直接进行操作，给用户的使用带来极大的便利。  二、优点  组合模式的主要优点有：   1. 组合模式使得客户端代码可以一致地处理单个对象和组合对象，无须关心自己处理的是单个对象，还是组合对象，这简化了客户端代码； 2. 更容易在组合体内加入新的对象，客户端不会因为加入了新的对象而更改源代码，满足“开闭原则”；   其主要缺点是：   1. 设计较复杂，客户端需要花更多时间理清类之间的层次关系； 2. 不容易限制容器中的构件； 3. 不容易用继承的方法来增加构件的新功能；  三、实现 组合模式的结构与实现  组合模式的结构不是很复杂，下面对它的结构和实现进行分析。  1. 模式的结构  组合模式包含以下主要角色。   1. 抽象构件（Component）角色：它的主要作用是为树叶构件和树枝构件声明公共接口，并实现它们的默认行为。在透明式的组合模式中抽象构件还声明访问和管理子类的接口；在安全式的组合模式中不声明访问和管理子类的接口，管理工作由树枝构件完成。（总的抽象类或接口，定义一些通用的方法，比如新增、删除） 2. 树叶构件（Leaf）角色：是组合中的叶节点对象，它没有子节点，用于继承或实现抽象构件。 3. 树枝构件（Composite）角色 / 中间构件：是组合中的分支节点对象，它有子节点，用于继承和实现抽象构件。它的主要作用是存储和管理子部件，通常包含 Add()、Remove()、GetChild() 等方法。   组合模式分为透明式的组合模式和安全式的组合模式。  (1) 透明方式  在该方式中，由于抽象构件声明了所有子类中的全部方法，所以客户端无须区别树叶对象和树枝对象，对客户端来说是透明的。但其缺点是：树叶构件本来没有 Add()、Remove() 及 GetChild() 方法，却要实现它们（空实现或抛异常），这样会带来一些安全性问题。其结构图如图 1 所示。  透明式的组合模式的结构图  图1 透明式的组合模式的结构图    (2) 安全方式  在该方式中，将管理子构件的方法移到树枝构件中，抽象构件和树叶构件没有对子对象的管理方法，这样就避免了上一种方式的安全性问题，但由于叶子和分支有不同的接口，客户端在调用时要知道树叶对象和树枝对象的存在，所以失去了透明性。其结构图如图 2 所示。  安全式的组合模式的结构图 图2 安全式的组合模式的结构图 | | | |