设计模式之访问者模式

刚刚学完设计模式的访问者模式(编译器模式),这里就对该模式进行了总结与分析.

一.产生原因

这里存在一个这样的问题:如果某系统已经完成了一个类层次并提供了满足需求的所有接口,现在要增加新的需求,我们需要怎么做?

可能你会采用增加该需求并把整个层次结构统统修改一遍.然而如果需求变动会不停的发生,而且需求的任何变动都会让整个结构统统修改一遍,此时你会怎么做呢?

所以,我们现在需要对这个系统结构进行重构,访问者模式也许就是你解决上面问题最好的选择.

那么什么是访问者模式呢?

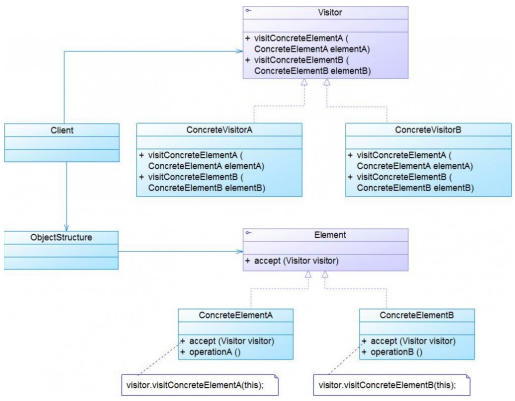
二.访问者模式的定义与结构

模式定义

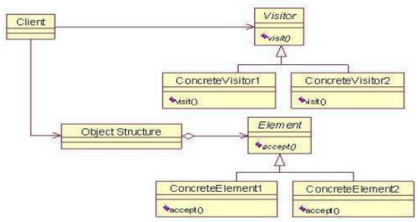
访问者模式(Visitor Pattern)：表示一个作用于某对象结构中的各元素的操作，它使我们可以在不改变各元素的类的前提下定义作用于这些元素的新操作。

访问者模式是一种对象行为型模式。顾名思义(通俗定义)

使用这个模式后就可以在不修改已有程序结构的前提下，通过添加额外的“访问者”完成对已有代码功能的提升。下面是一个很常见的访问者模式结构图:



把它简化后就是这样一张图:



由图可见,访问者模式主要是由以下5部分角色组成:

Vistor: 抽象访问者.定义接口，声明一个或多个访问操作.

ConcreteVisitor: 具体访问者.实现抽象访问者所声明的接口，也就是抽象访问者所声明的各个访问操作.

Element: 抽象元素.声明一个接受操作，接受一个访问者对象作为一个参数.

ConcreteElement:具体元素.实现抽象结点所规定的接受操作.

ObjectStructure:对象结构.可以遍历结构中的所有元素，提供一个接口让访问者对象都可以访问每一个元素.

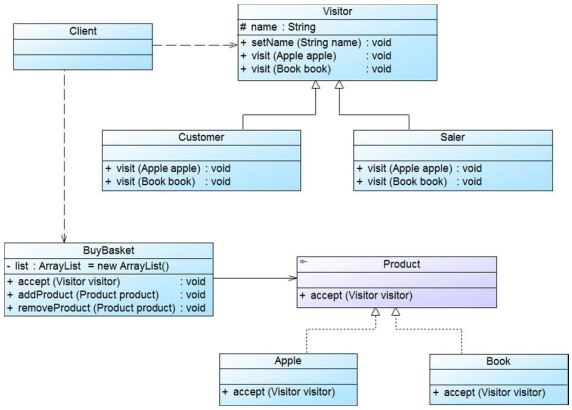
三.实例分析

通过下面的例子来分析该访问者模式的概念.这是清华大学课件上的例子,还有很多例子：男人与女人、欢乐谷等，但个人觉得这个例子更适合阐述清楚该模式。

购物车

顾客在超市中将选择的商品，如苹果、图书等放在购物车中，然后到收银员处付款。在购物过程中，顾客需要对这些商品进行访问，以便确认这些商品的质量，之后收银员计算价格时也需要访问购物车内顾客所选择的商品。此时，购物车作为一个ObjectStructure（对象结构）用于存储各种类型的商品，而顾客和收银员作为访问这些商品的访问者，他们需要对商品进行检查和计价。不同类型的商品其访问形式也可能不同，如苹果需要过秤之后再计价，而图书不需要。使用访问者模式来设计该购物过程。

它的访问者模式结构图为:



由图可以知:在该结构体系中Apple和Book都是抽象的元素，是Product的具体实现。而抽象访问者是Visitor，它的具体访问者是Customer(顾客)和Saler(收银员)，它们之间的关系通过BuyBasket(购物车)这个具体的对象结构联系,各自都有相关的方法.我尝试用C#实现了该模式，如下：

首先是元素代码部分:

```
//抽象元素:产品
public abstract class Product
{
    public abstract void accept(Visitor visitor);
}

//具体元素：苹果
public class Apple : Product
{
    public override void accept(Visitor visitor)
    {
        visitor.visit(this);
    }
}

//具体元素：书
public class Book : Product
{
    public override void accept(Visitor visitor)
    {
        visitor.visit(this);
    }
}
```

然后是访问者代码部分:

```
//抽象访问者:Visitor
public abstract class Visitor {
    public abstract void visit(Apple apple);
    public abstract void visit(Book book);
}

//具体访问者:顾客
public class Customer : Visitor {
    public override void visit(Apple apple) {
        Console.WriteLine("顾客购买苹果");
    }
    public override void visit(Book book)
    {
        Console.WriteLine("顾客购买书");
    }
}

//具体访问者: 售货员
public class saler : Visitor {
    public override void visit(Apple apple)
    {
        Console.WriteLine("售货员销售苹果");
    }
    public override void visit(Book book)
    {
        Console.WriteLine("售货员销售书");
    }
}
```

然后是通过对象结构把元素和访问者串联在一起:

```
//通过ObjectStruct(对象结构)把元素和访问者串联
public class ObjectStruct {
    private ArrayList list = new ArrayList();

    //增加产品
    public void addProduct(Product product) {
        list.Add(product);
    }

    //移除产品
    public void removeProduct(Product product) {
        list.Remove(product);
    }

    //访问者访问
    public void Accept(Visitor visitor)
    {
        foreach (Product p in list)
            p.accept(visitor);
    }
}
```

最后是在Main中实现,实例化具体的方法并输出结果:

```
static void Main(string[] args)
{
    ObjectStruct os = new ObjectStruct();
    os.addProduct(new Apple());
    os.addProduct(new Book());

    Customer one = new Customer();
    saler two = new saler();
    os.Accept(one);
    os.Accept(two);

    Console.ReadLine();
}
```

```
file:///G:/software/Program software/C#项
顾客购买苹果
顾客购买书
售货员销售苹果
售货员销售书
```

由此可见,通过访问者能够实现具体的访问者顾客和售货员都访问到具体的元素苹果和书.

四.优点

现在又遇到一个问题:

要向原来的购物车类层次增加一个新的操作——工商局的检查超市产品的生产日期，怎么解决呢?

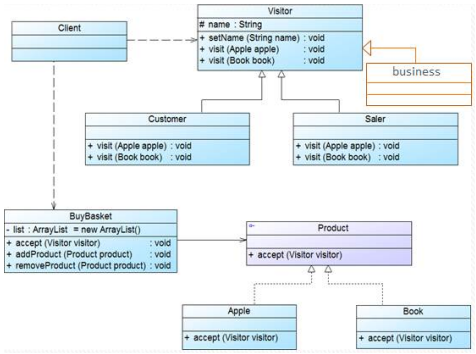
解决方法:

仅仅需要添加一个具体的访问者角色即可，而不必修改整个类层次，这样就符合“开闭原则”的需求。

而且每个访问者角色都对应一个相关的操作,当需要修改一个具体的访问者角色时仅仅修改该访问者角色,而不是修改整个

类层次.

如下图所示:添加红色部分具体的访问者business(工商局)即可实现该操作。



由此可见访问者模式的优点如下:

1.增加新的访问操作变得容易;

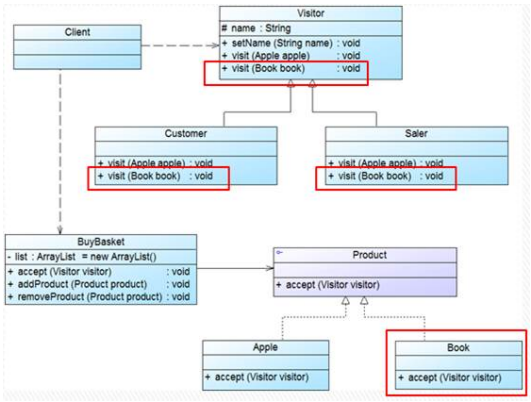
2.将有关元素对象的访问行为集中到一个访问者对象中,而不是分散到各个元素类中;

3.用户能在不修改类层次结构情况下定义该类层次结构的操作并修改.

五.缺点

现在又遇到一个问题:要向原来的购物车类层次增加一个新的元素“Pen”,或者“Book”元素需要改变，怎么解决呢?

解决方法：你必须要修改访问者角色和每一个具体访问者角色,对应的访问者每一个函数的参数都要相应的变化.如下图所示:修改"Book"时每个具体的访问者和抽象的访问者的方法都要修改，这就相当的麻烦了。



由此可见，访问者模式的缺点如下：

缺点一:访问者模式增加新的元素很困难,每增加一个新的元素或修改一个元素,就要对相应的访问者类进行修改,违背了“开闭原则”。

而且访问者角色要执行与角色相关的操作,就必须让元素的内部属性暴露出来,在java中就意味着对象可以访问，这就破坏了元素的封装性。

缺点二:破坏封装性。

缺点三:而且访问者之间能够传递的信息有限,这就往往会限制访问者模式的使用。

六.总结

这是一个巧妙而又复杂的模式,它的使用条件比较苛刻.

当系统存在固定的数据结构(类层次),又有不同的行为,那么访问者模式是个不错的选择.如果需要修改或增加元素时,就不应该使用访问者模式.

访问者模式要尽可能的将对象浏览逻辑存放在Visitor类中，而不是放在它的子类中；这样ConcreteVisitor类所访问的对象结构依赖较少，便于维护。

七.区别

由于该访问者模式(又称编译器模式)是金福生老师让我们与解释器模式一起学习的,所以老师课堂上提出一个这样的问题:

访问者模式(编译器模式)和解释器模式有啥区别?

通俗的说就是编译器模式(访问者模式)就是相当于把所有的都编译之后,能访问到所有东西的模式;而解释器模式是相当于编译一条语句执行一条语句的模式.具体有啥区别没有深入的分析,见谅.其中该博客的思想主要来自:自己的理　　解与分析清华大学的访问者模式课件

一些思想来自下面三个博客的阅读

http://blog.csdn.net/xiaoquanhuang/article/details/6311319

http://www.cnblogs.com/pursuedream/articles/795051.html

http://www.iteye.com/topic/345384

感谢上面3个文章的作者和清华大学的课件,这是我的一些理解与分析.仅供大家学习,有不足之处见谅.(By:Eastmount 2013-3)