## 第26章 访问者模式

## 25.1

熊小猫：这个月就要发年终奖了，辛苦了一年，准备怎么奖励一下自己？

兔小白：是呀，这也是我工作以来第一次拿奖金，想想还有点兴奋。不过我的工资基数低，年终奖不会太多，我计划过年回家给父母买点礼物，剩下的先存起来。你打算怎么花呢？看起来已经计划好了吧？

熊小猫：我也没多少，先存起来吧，生活中花钱的地方太多了。

兔小白：你可别谦虚了！你工作了这么多年，薪资基数大。而且我听说普通员工是按照年薪的12%发奖金，但管理人员是按15%发奖金。你的奖金一定少不了。

熊小猫：管理人员的薪酬构成中，奖金的比例更大，这是为了刺激管理层协助公司取得更高的业绩。但是万一公司业绩不达标，导致奖金被打了折扣，管理人员的损失也会更大。

兔小白：今年公司业绩貌似还不错，应该可以全额拿到奖金。好羡慕你呀！

熊小猫：只要你脚踏实地，学好技术，早晚会超过我的。经过这么久的设计模式学习，马上就要学完了。现在只差最后一个模式！把设计模式全部攻克后，咱们就可以继续学习新知识了。

兔小白：咱们今天就把最后一种设计模式讲完吧！我也期待早日升职加薪呢。

熊小猫：好呀！最后一个设计模式叫做访问者模式。我刚刚想到，正好可以用计算薪资这个场景来学习。

## 25.2

熊小猫：普通员工和管理者的奖金计算存在差异，我们假设员工拿到全额奖金，普通员工的奖金是年薪的12%，管理者的奖金是年薪的15%。你来想想程序怎么实现。

兔小白：这太简单了，等我5分钟。

5分钟后，兔小白写完了代码。

兔小白：定义抽象员工类Employee，含有姓名和月薪属性。calculateBonus为计算奖金的抽象方法。

//代码

Employee有两个子类，分别为普通员工和管理者。两个子类对calculateBonus方法有不同实现。先看普通员工OrdinaryEmployee类，calculateBonus方法中按照年薪12%计算奖金。

//代码

管理者Manager类按照年薪15%计算奖金。

//代码

客户端中构建了一个Employee类型对象的列表，迭代每个Employee对象计算奖金。

//代码

程序输出如下。

//代码

熊小猫：你的程序设计是比较常见的设计方式，通过继承，子类重写父类的方法，实现自己独特的逻辑。但这并不代表就是最优的实现方式。现在我要加新的需求了。

兔小白：没问题，你说吧，我对自己的程序有信心。

熊小猫：新的需求是，增加新的薪资项计算——电话费津贴。普通员工是每月100元，管理者是每月300元。

兔小白：你不按套路出牌呀，我想的是你要增加一种员工类型，有新的奖金计算方式。这样的话，我增加Employee的实现子类就可以了。程序符合开闭原则。

熊小猫：这就没意思啦，都是你会的东西。我这回增加的是新操作。

兔小白：这样的话，只能先给Employee增加计算电话津贴的抽象方法，就叫做getTelephoneAllowance吧。然后它的两个子类分别实现该方法。

熊小猫：Employee继承体系中所有的类都需要修改，严重违反了开闭原则。

兔小白：这没有办法呀，要增加类的行为，只能修改。

熊小猫：我有办法不修改类，就可以增加它的行为。

## 25.3

熊小猫：你的设计角度是从行为发出者入手。所以增加行为，必然修改行为的发出者。我们换个角度，从行为出发。行为针对不同的发出者，将会有不同表现。你来试试将calculateBonus的行为抽象为对象，根据计算的对象不同，执行不同的逻辑。

兔小白：我觉得可以这样修改 ，定一个BonusCalculator类，封装calculateBonus行为。针对不同的计算对象类型，提供两个重载的calculateBonus方法。

//代码

熊小猫：你完成了改造的第一步，现在Calculator支持扩展，也就是说做到了操作可扩展，接下来需要将Employee和Calculator关联起来。在Employee中定义一个接受Calculator类型对象的方法，这样Employee可以通过使用传入不同的Caculator子类对象实现不同的操作。

先来看Employee抽象类的修改，增加accept抽象方法。

//代码

OrdinaryEmployee和Manager对accept的实现几乎一样，都是调用BonusCalculator的calculateBonus方法，将自己作为参数传入。由于自身类型不同，分别调用了两个重载的calculateBonus方法。

OrdinaryEmployee代码修改如下。

//代码

Manager代码修改如下。

//代码

现在的程序结构使得Employee可以使用accept方法接收不同的Calculator来实现不同操作。我们可以为计算电话津贴定义一个新的Calculator子类TelephoneAllowanceCalculator。

//代码

客户端代码中，在调用Employee的accept方法时，通过改变入参Calculator类型对象，来切换Employee的行为。

//代码

程序输出符合预期。

兔小白：Employee定义的accept方法很关键，这是一个“万能”方法！accept方法可以接收任何Calculator子类对象，然后用该对象完成特定的操作。

熊小猫：这就是访问者模式。下面我们来详细分析访问者模式。

## 26.4

熊小猫：我们先来看看访问者模式的结构图。

//图

访问者模式包含两套继承体系。一套是Visitor，对应练习中的Calculator。另外一套是Element，对应练习中的Employee。

Visitor是访问者类，定义了多个针对不同Element子类的操作方法。ConcreteVisitor是具体的访问者类，负责实现一种具体操作。ConcreteVisitor将对不同Element子类的操作逻辑，分散在一系列重载方法中实现。访问者模式的名字便来自于这种操作实现方式，可以理解为Visitor根据自己访问的Element类型，来提供相应的处理。

Element是元素类，主要目的是定义accept方法。ConcreteElement是具体元素类，实现accept方法。accept方法接收一个Visitor类型参数。accept方执行的操作由传入的Visitor子类类型所决定。这样就做到了元素和操作的分离。不修改元素，也可以为元素添加操作。

//图双重决定画清楚

ObjectStructure是一个存储Element对象的结构类，可以是树状结构，也可以是列表等集合。它提供一个能够访问每个Element的方法。它对应练习中的Employee列表。

此外，由于Visitor提供的方法可以接受任何类型的参数，它可以访问不具有相同父类的对象。[[1]](#footnote-1)1因此，ObjectStructure中存储的Element对象不一定具有相同的父类。

访问者模式分离了元素和作用于元素的操作。这使得不改变元素，也可以定义作用于元素的新操作。它具备下面这些优点：

1. 对元素的操作可扩展。这是访问者模式的核心优势，增加新的操作，只需要增加新的访问者。
2. 集中同类型操作。将同类型的操作集中在一个访问者类中，而不是分散在各个元素类中。这样有利于程序的维护。
3. 符合单一职责原则。元素类负责数据维护，一个访问者类负责对元素的一种操作，职责明确且单一。

兔小白：访问者模式的优点确实很多，但是我觉得它对元素类的扩展支持并不好。

熊小猫：你说的没错，鱼和熊掌不能兼得，访问者模式获得了操作的扩展性，但是牺牲了元素的扩展性。增加新的元素类，需要为每一个访问者子类增加匹配的方法，供新的元素类调用。访问者类执行操作时，常常需要了解元素类的内部属性，这在一定程度上破坏了元素类的封装。

为了扬长避短，适合使用访问者模式的场景，应具备如下特点。

1. 元素类只负责数据的维护，不提供操作。
2. 元素类的种类固定。由于增加或减少元素类，会影响所有已经存在的访问者类。因此需要稳定的元素类种类。
3. 对元素的操作经常发生变化。元素操作类型不稳定的场景下，访问者模式才能发挥它易于扩展操作的优势。

兔小白：你常说软件设计要贴近现实世界。我觉得访问者模式更适合Element对象天然就只是封装数据，而不应刻意塑造。如果某个场景中，想要抽象为Element的对象并不单纯封装数据，那么强行将它的行为分离成Visitor类，并不可取。

熊小猫：设计模式的运用要自然，不要刻意。如果问题还没彻底分析清楚，就先选定一种设计模式，那么后面的问题分析和程序设计就会被框定在这种设计模式中，刻意往上靠。最终写出程序不但难以理解，而且也无法发挥出设计模式的优势。

到今天为止，我们学完了所有23种设计模式。但这只是刚刚上路，学懂不等于会用。今后的工作中多思考、多实践，才能融会贯通，运用自如。加油吧，兔小白！

1. 1 《设计模式》书中的访问者模式结构图没有体现这一点，但在对访问者模式的效果部分讲解了这一特点。 [↑](#footnote-ref-1)