# 访问者模式简介

访问者模式包含访问者和被访问对象两个主要组成部门，这些被访问的元素具有不同的类型，且不同的访问者可以对其施加不同的访问操作。访问者模式使得用户可以在不修改现有系统的情况下扩展系统的功能，为这些不同类型的元素增加新的操作。

## 概述

1. 在医生开具处方单（药单）后，很多医院都存在以下处理流程：划价人员拿到处方单之后根据药品名称和数量计算总价，药房工作人员根据药品名称和数量准备药品。可以将处方单看成一个药品信息的集合，里面包含了一种或多种不同类型的药品信息，不同类型的工作人员（例如划价人员和药房工作人员）在操作同一个药品信息集合时将提供不同的处理方式，而且可能还会增加新类型的工作人员来操作处方单。在软件开发中有时候也需要处理像处方单这样的集合对象结构，在该对象结构中存储了多种不同类型的对象信息，而且对同一对象结构中的元素和操作方式并不唯一，可能需要提供多种不同的处理方式，还有可能增加新的处理方式。在设计模式中有一种设计模式可以满足上述要求，其模式动机就是以不同的方式操作复杂对象结构，该模式就是访问者模式。
2. 访问者模式包含访问者和被访问者元素两个主要组成部分，这些被访问的元素通常具有不同的类型，且不同的访问者可以对它们进行不同的访问操作访问者模式使得用户可以在不修改现有系统的情况下扩展系统的功能，为这些不同类型的元素增加新的操作。使用访问者模式时，被访问元素通常不是单独存在的，它们存储在一个集合中，这个集合被称为“对象结构”，访问者通过遍历对象结构实现对其中存储的元素逐个操作。
3. 访问者（Visitor Pattern）模式的定义如下：表示一个作用于某对象结构中的各个元素的操作。访问者让用户可以在不改变元素的类的前提下定义作用于这些元素的操作。
4. 访问者模式是一种对象行为型模式，它为操作存储不同类型元素的对象结构提供了一种解决方案，用户可以对不同类型的元素施加不同的操作。

## 结构

访问者模式结构图如下：



由图可知，访问者包含以下5个角色

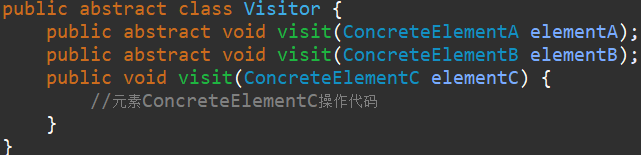
1. Visitor（抽象访问者）：抽象访问者为对象结构中的每一个具体元素类声明了一个访问操作，从这个操作的名称或参数类型可以清楚地知道需要访问的具体元素的类型，具体访问者需要实现这些操作方法，定义对这些元素的访问操作。
2. ConcreteVisitor（具体访问者）：具体访问者实现了每个由抽象访问者声明的操作，每一个操作用于访问对象结构中一种类型的元素。
3. Element（抽象元素）：抽象元素一般是抽象类或者接口，它声明了一个accept( )方法，用于接收访问者的访问操作，该方法通常以一个抽象访问者作为参数。
4. ConcreteElement（具体元素）：具体元素实现了抽象元素类中的accept( )方法，在accept( )方法中调用访问者的访问方法以便完成对一个元素的操作。
5. ObjectStructure（对象结构）：对象结构是一个元素的集合，用于存放元素对象，并且提供了遍历其内部元素的方法。对象结构可以结合组合模式来实现，也可以是一个简单的集合对象。

# 实现

## 实现原理

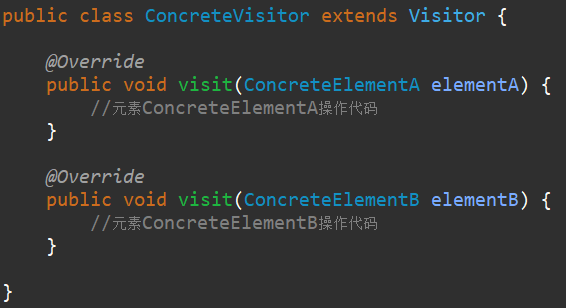
在访问者模式中，对象结构存储了不同类型的元素对象，以供不同访问者访问。访问者模式包括两个层次结构：一个是访问者层次结构，提供了抽象访问者和具体访问者；一个是元素层次结构，提供了抽象元素和具体元素。相同的访问者可以用不同的方式访问不同的元素，相同的元素可以接收不同访问者以不同的方式访问。在访问者模式中增加新的访问者无须修改原有系统，系统具有较好的可扩展性。

1. 在访问者模式中，抽象访问者定义了访问元素对象的方法，通常为每一种类型的元素对象都提供一个访问方法，而具体访问者可以实现这些访问方法。这些访问方法的命名一般有两种方式：一种是直接在方法命名中标明待访问元素对象的具体类型，例如visitElementA(ElementA elementA)；另一种是统一命名为visit( )，通过参数类型的不同来定义一系列重载的visit( )方法。当然，如果所有的访问者对某一类型的元素的访问操作都相同，则可以将操作代码移到抽象访问者类中，其典型代码如下：

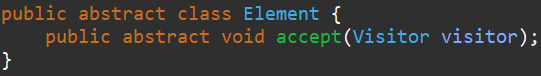


在这里使用了重载visit( )方法的方式来定义多个方法，用于操作不同类型的元素对象。

1. 在抽象访问者类Visitor的子类ConcreteVisitor中实现了抽象的访问方法，用于定义对不同类型元素对象的操作。具体访问者的典型代码如下：

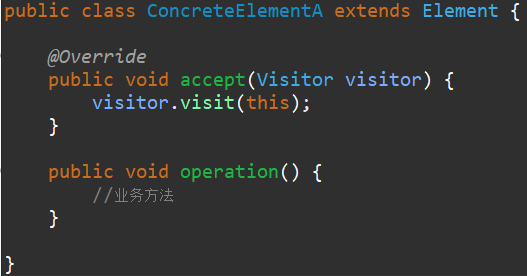


1. 对于元素类而言，在其中一般都定义一个accept( )方法，用于接收访问者的访问。典型的抽象元素类的代码如下：



需要注意的是，该方法传入一个抽象访问者Visitor类型的参数，即针对抽象访问者进行编程，而不是具体访问者，在程序运行期时再确定具体访问者的类型，并调用具体访问者对象的visit( )方法实现对元素对象的操作。

1. 在抽象元素类的子类中实现了accept( )方法，用于接收访问者的访问，在具体元素中还可以定义不同类型的元素所特有的业务方法。其典型代码如下：



在具体元素类ConcreteElementA的accept( )方法中，通过调用Visitor类的visit( )方法实现对元素的访问，并以当前对象作为visit( )方法的参数。其具体执行过程如下：

1. 调用具体元素类的accept(Visitor visitor)方法，并将Visitor子类对象作为其参数。
2. 在具体元素类accept(Visitor visitor)方法内部调用传入的Visitor对象的visit( )方法，例如visit(ConcreteElementA elementA)，将当前具体元素对象（this）作为参数，例如visitor.visit(this)。
3. 执行Visitor对象的visit( )方法，在其中还可以调用具体元素对象的业务方法。

这种机制也称为“双重分派”，正因为使用了双重分派机制，使得增加新的访问者无须修改现有类库代码，只需将新的访问者对象作为参数传入具体元素对象的accept( )方法，程序运行时将回调在新增访问者类中定义的visit( )方法，从而增加新的元素访问方式。

1. 在访问者模式中对象结构是一个集合，用于存储元素对象并接受访问者的访问。其典型代码如下：



在对象结构中可以使用迭代器对存储在集合中的元素对象进行遍历，并逐个调用每一个对象的accept( )方法，实现对元素对象的访问操作。

## 实例

实例说明：某OA系统中包含一个员工信息管理子系统，该公司员工包括正式员工和临时工，每周人力资源部和财务部等部门需要对员工数据进行汇总，汇总数据包括员工工作时间、员工工薪等。该公司的基本制度如下：

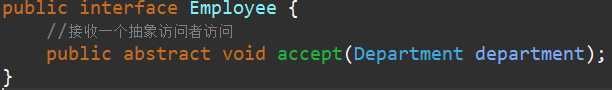
1. 正式员工每周工作时间为40小时，不同级别、不同部门的员工每周基本工资不同；如果超过40个小时，超出部分按照100元/小时作为加班费；如果少于40小时，所缺时间按照请假处理，请假所扣工资以80元/小时计算，知道基本工资扣除到零为止。除了记录实际工作时间外，人力资源部需要记录加班时长或请假时长，作为员工平时表现的一项依据。
2. 临时工每周工作时间不固定，基本工资按小时计算，不同岗位的临时工工资不同。人力资源部只需记录实际工作时间。

人力资源部和财务部工作人员可以根据需要对员工数据进行汇总处理，人力资源部负责汇总每周员工工作时间，而财务部负责计算每周员工工资。现使用访问者模式设计该系统，经过分析，本实例的结构图如下所示：

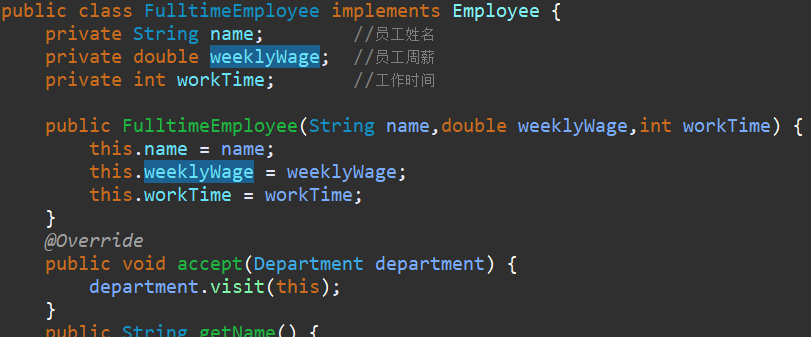


图中，FADepartment表示财务部，HRDepartment表示人力资源部，它们充当具体访问者角色，其抽象父类Department充当抽象访问者角色；EmployeeList充当对象结构，用于存储员工列表；FulltimeEmployee表示正式员工，ParttimeEmployee表示临时工，它们充当具体元素角色，其父接口Employee充当抽象元素角色。

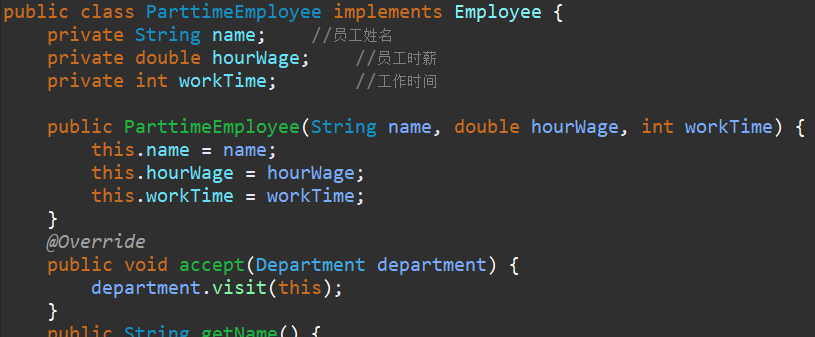
1. Employee：员工类，充当抽象元素类



1. FulltimeEmployee：全职员工类，充当具体元素类



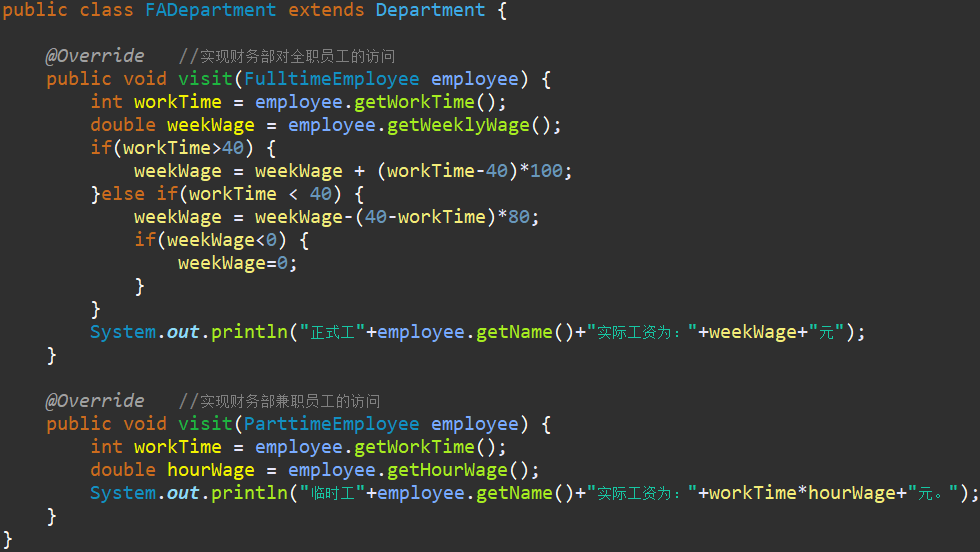
1. ParttimeEmployee：兼职员工类，充当具体元素类



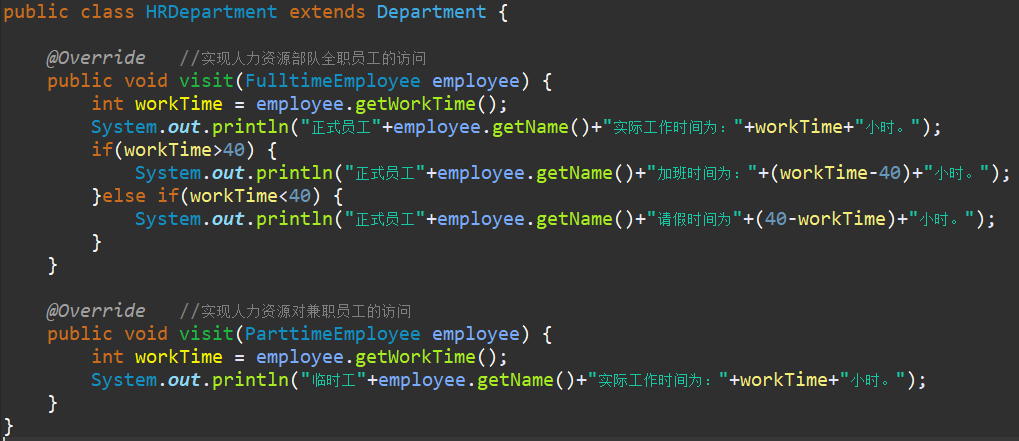
1. Department：部门类，充当抽象访问者类



1. FADepartment：财务部类，充当具体访问者类



1. HRDepartment：人力资源类，充当具体访问者



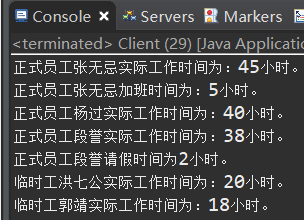
1. EmployeeList：员工列表类，充当对象结构



1. Client：客户端测试类



运行结果如下：



如果要在系统中增加一种新的访问者，无须修改源代码，只要增加一个新的具体访问者类即可，在该具体访问者中封装了新的操作元素对象的方法。从增加新的访问者的角度看，访问者模式符合开闭原则。如果要在系统中增加一个新的具体元素，例如增加一个新的员工类型为“退休人员”，由于原有系统并未提供相应的访问接口（在抽象访问者中没有声明任何访问“退休人员”的方法），因此必须对原有系统进行修改，在原有的抽象访问者类和具体访问者类中增加相应的访问方法。从增加新的元素角度来看，访问者模式违背了开闭原则。综上所述，访问者模式与抽象工厂模式类似，对开闭原则的支持具有倾斜性，可以很方便的添加新的访问者，但是添加新的元素较为麻烦。

## 访问者模式与组合模式的联用

在访问者模式中包含一个用于存储元素集合的对象结构，通常可以使用迭代器来遍历对象结构，同时具体元素之间可以存在整体和部门关系，有些元素作为容器对象，有些元素作为成员对象，可以使用组合模式来组合元素。引入组合模式后的访问者模式结构图如下：



需要注意的是，上图结构中，由于叶子元素的遍历操作已经在容器元素中完成，因此要防止单独将已增加到容器元素中的叶子元素再次加入对象结构中，在对象结构中只需保存容器元素和孤立的叶子元素。

# 优缺点及适用环境

由于访问者模式的使用条件较为苛刻，本身结构也较为复杂，因此在实际应用中使用频率不是特别高。当系统中存在一个较为复杂的对象结构，且不同访问者对其所采取的操作也不相同时，可以考虑使用访问者模式进行设计。在XML文档解析、编译器的设计、复杂集合对象的处理等领域中访问者模式得到了一定的应用。

## 优点

访问者模式的优点主要如下：

1. 在访问者模式中增加新的访问操作很方便。使用访问者模式，增加新的访问操作就意味着增加一个新的具体访问者类，实现简单，无须修改源代码，符合开闭原则。
2. 访问者模式将有关元素对象的访问行为集中到一个访问者对象中，而不是分散在一个个元素类中。类的职责更加清晰，有利于对象结构中元素对象的复用，相同的对象结构可以供多个不同的访问者访问。
3. 访问者模式让用户能够在不修改现有元素类层次结构的情况下定义作用于该层次结构的操作。

## 缺点

访问者模式的缺点主要如下：

1. 在访问者模式中增加新的元素类困难。在访问者模式中，每增加一个新的元素类都意味着要在抽象访问者角色中增加一个新的抽象操作，并在每一个具体访问者类中增加相应的具体操作（覆写），这违背了开闭原则的要求。
2. 访问者模式破坏了对象的封装性。访问者模式要求访问者对象访问并调用一个元素对象的操作，这意味着元素对象有时候必须暴露一些自己的内部操作和内部状态，否则无法提供访问者访问。

## 适用环境

在以下情况下可以考虑适用访问者模式：

1. 一个对象结构包含多个类型的对象，希望对这些对象实施一些依赖其具体类型的操作。在访问者中针对每一种具体的类型都提供一个访问操作，不同类型的对象可以有不同的访问操作。
2. 需要对一个对象结构中的对象进行很多不同的并且不相关的操作，而需要避免让这些操作“污染”这些对象的类，也不希望在增加新操作时修改这些类。访问者模式使得用户可以将相关的访问操作集中起来定义在访问者类中，对象结构可以被多个不同的访问者类所使用，将对象本身与对象的访问操作分离。
3. 对象结构中对象对应的类很少变化，但经常需要在此对象结构上定义新的操作。