## 访问者模式

1. 定义与使用场景

定义：封装一些作用于某种数据结构中的各元素的操作，它可以在不改变这个数据结构的前提下定义作用于这些元素的操作

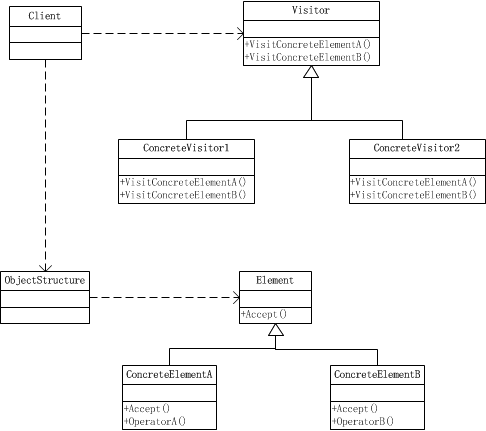
使用场景：

1）、对象结构比较稳定，但经常需要在此对象结构上定义新的特性

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，而需要避免这些操作“污染”这些类，也不希望在增加新操作时修改这些类

访问者模式是一种将数据操作与数据结构分离的设计模式，是23中设计模式中最复杂的一个，但它的使用频率不高。

1. UML类图



Vistor：接口或者抽象类，它定义了每一个元素Element访问的行为，它的参数就是可以访问的的元素，它的方法个数理论上讲与元素个数是一样，因此，访问者模式要求元素的类族稳定

ConcreteVisitorA，ConcreteVistorB：具体的访问者，它需要给出对每个元素类访问时所产生的具体行为

Element：元素接口或者抽象类，它定义了接受访问者accept的方法，其意义是指每一个元素都要可以被访问者访问

ElementA、ElementB：具体的元素，它提供接受访问方法的具体实现，而这个具体的实现，通常情况下是使用访问者提供的访问该元素类的方法

ObjectStructure：定义当中所提到的对象结构，对象结构是一个抽象描述，它内部管理了元素结合，并且可以迭代这些元素提供访问者访问

1. 简单示例

访问者模式最大的优点就是增加访问者非常容易

1. 源码实现

注解：1）、运行时注解，；2）、编译时注解

运行时注解由于性能问题被一些人诟病；

编译时注解的核心原理依赖APT（Annotation Processing Tools）实现，比较著名的ButterKnife、Dagger、Retrofit等开源库都是基于APT。

编译时Annotation解析的基本原理是，在某些代码元素上（类型、函数、字段等）添加注解，在编译时编译器会检查AbstractProcessor的子类，并且调用该类型的process函数，然后将添加了注解的所有元素都传递到process函数中，使得开发人员可以在编译器进行相应的处理。例如：根据注解生成新的Java类，这也就是ButterKnife等开源库的基本原理。

编写注解处理器的核心是AnnotationProcessFactory和AnnotationProcessor两个接口，后者表示的是注解处理器，而前者则是为某些注解类型创建注解处理器的工厂。

对于编译器来说，代码中的元素结构是基本不变的，例如，组成代码的基本元素有包、类、函数、字段、类型参数、变量。JDK中为这些元素定义了一个基类，也就是Element类，它有如下几个子类：

1）、PackageElement：包元素，包含了某个包下的信息，可以获取到包名等；

2）、TypeElement：类型元素，如某个字段属于某种类型

3）、ExecutableElement：可执行元素，代表了函数类型的元素

4）、VariableElement：变量元素

5）、TypeParameterElement：类型参数元素

1. 实战总结