访问者模式(Visitor)

By: Zhi Li

1.什么是访问者模式?(What?)

访问者模式指的是一种将算法与对象结构分离的软件设计模式（概念很抽象，一般情况下你是会看不懂的，但是请继续往下看我的分析过程）

2.为什么使用访问者模式?（Why?）

对于相同的元素，可以定义不同的操作。这是访问者模式最重要的意义。
举个例子：
一个动物园具有不同的场馆，例如狮子馆，海洋馆，大象馆

狮子馆，海洋馆，大象馆他们都是相同的元素（因为本质上都是动物园，换句话说他们的父类是动物园）
但是对于这些相同的动物园元素之间，我们却有着不同爱好的游客。

假设有些游客来自于上海，他们观赏这三家场馆的方式不同。上海人最喜欢看狮子，那么他们在狮子馆中所观光的时间是最长的（2个小时以上），而在海洋馆和大象馆所观光的时间相对来说比较短（1个小时以下）。

另一批游客来自于杭州，他们更喜欢看大象。因此他们在大象馆中所花费的时间是最长的，而在另外两个馆所花费的时间较短。
看到这我估计你心里面大概对访问者模式的应用有个初步印象了，由于游客的种类不同，他们对于相同的元素（即不同的场馆之间）有着不同的操作。理解到这一层面以后咱们就可以继续往下了

3.怎么使用访问者模式?（How）

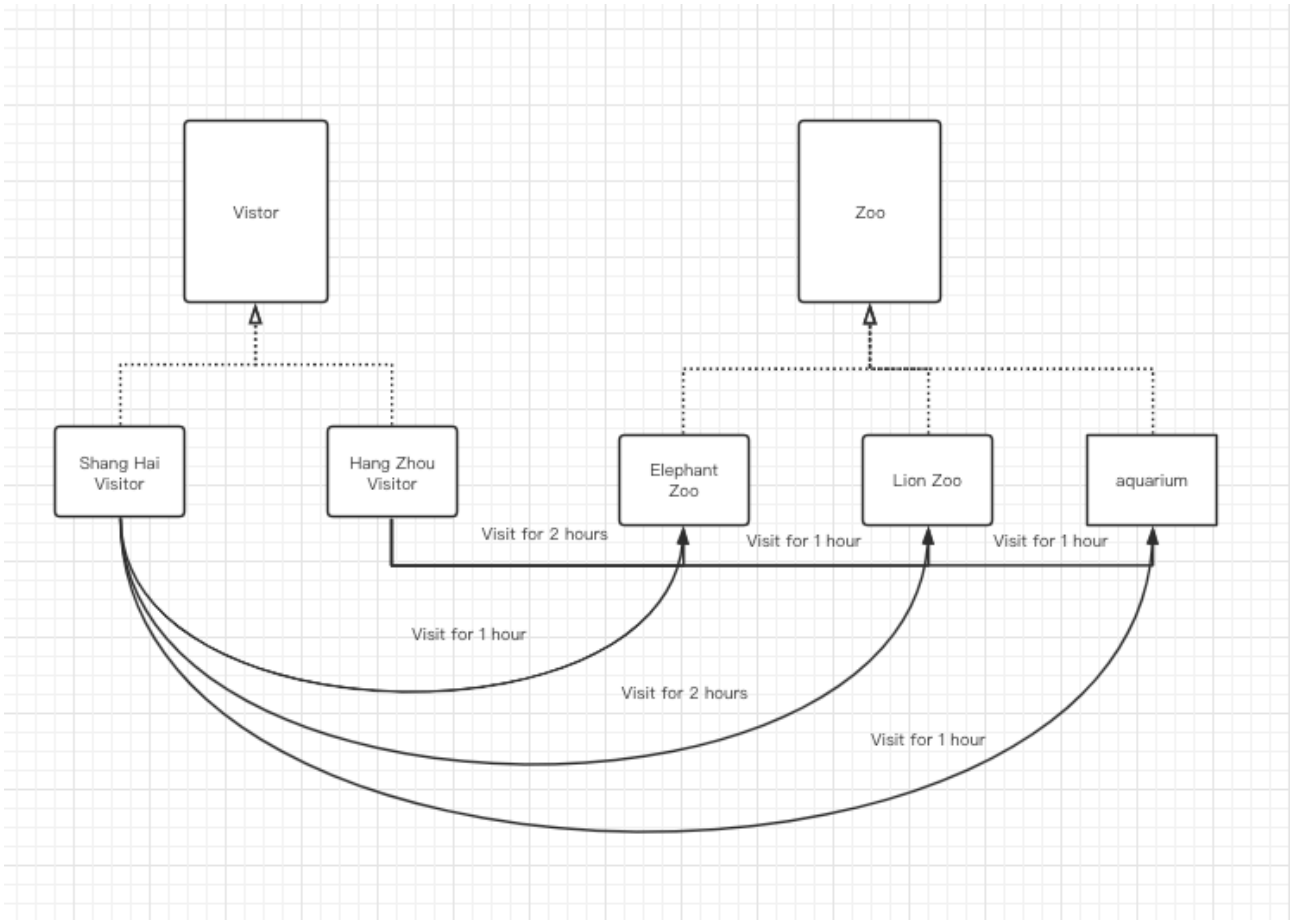
首先要理解清楚访问者模式的基本角色和概念

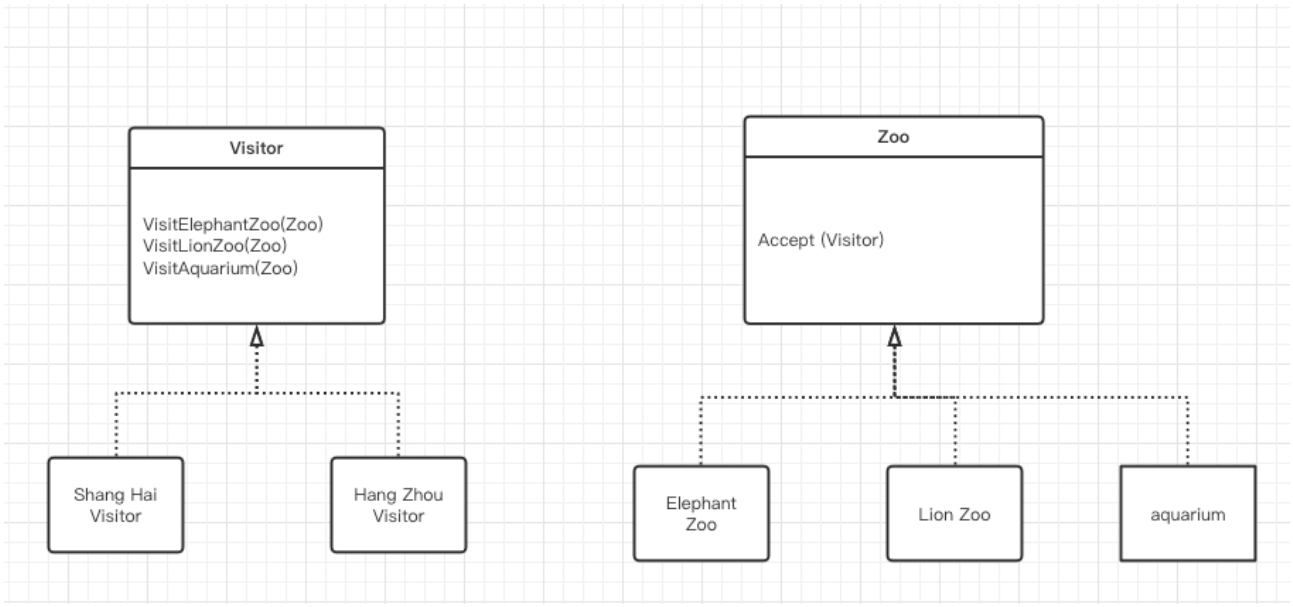
访问者模式的角色仅仅只包含两种：

Vistor (访问者)：对元素发起操作的人

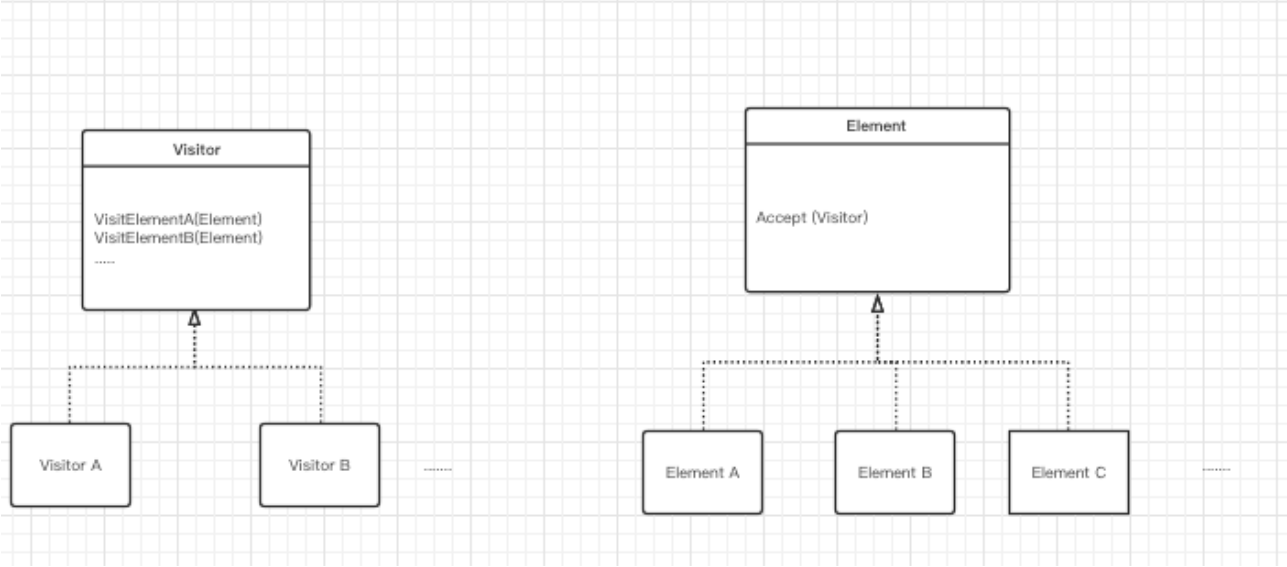
Element (元素，即被访问者)：被访问者所操作的对象

但是在实际开发场景之中，元素和访问者对应的是一个抽象的概念，换句话说他们对应的只是接口，而不是实例。
因此上述的动物园例子放在这里，对应的就是
interface Zoo (动物园类,对应的是角色Element)
Interface Visitor (观光者类，对应的是角色Vistor)
然后狮子馆大象馆和海洋馆才是角色Element的真正实例，上海人和杭州人才是Vistor的真正实例。
从这个例子出发，我们的访问者模式的一个类图模型就如下图所示


Vistor
Zoo
Shang Hai Visitor
Hang Zhou Visitor
Elephant Zoo
Lion Zoo
aquarium
Visit for 2 hours
Visit for 1 hour
Visit for 1 hour
Visit for 1 hour
Visit for 2 hours
Visit for 1 hour


Visitor
VisitElephantZoo(Zoo)
VisitLionZoo(Zoo)
VisitAquarium(Zoo)
Shang Hai Visitor
Hang Zhou Visitor
Zoo
Accept (Visitor)
Elephant Zoo
Lion Zoo
aquarium

对这个类图模型再进行进一步的抽象，我们就得到了Vistor模式的一个通用类图模板，即

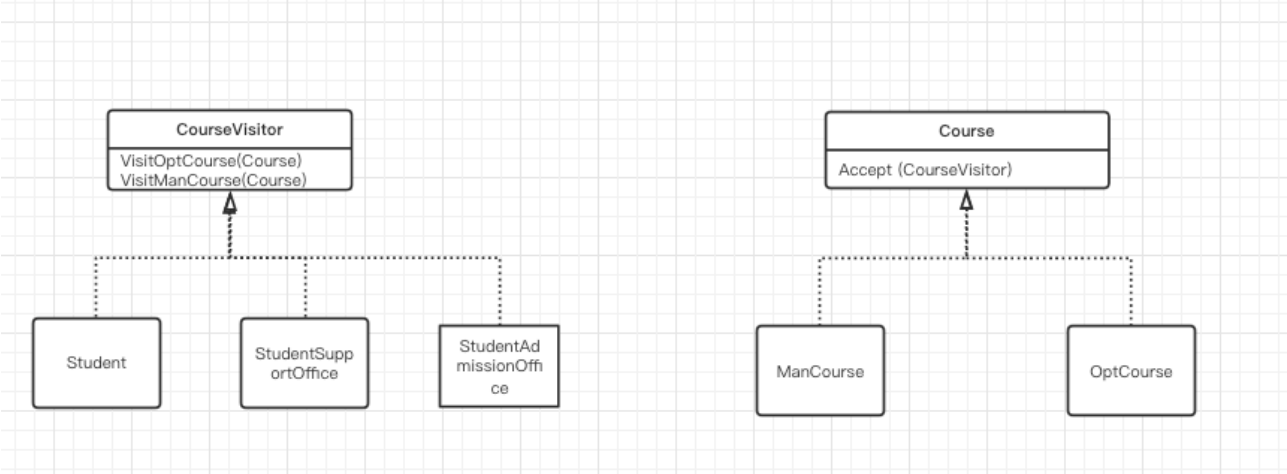


根据通用类图模板我们可知，在一个项目中若想使用访问者模式，那么关键就是在于实现Visitor和Element角色以及其下属的Visit方法和Accept方法。在动物园的例子中，我们的访问时间长短的逻辑就可以写在VisitElement中，而Accept方法中我们要做的事情就是调用VisitElement方法。

4.该项目中如何使用访问者模式?

终于到了介绍访问者模式最重要的一个环节，那就是在该项目中如何使用该模式？理论学习完以后最难的一点就是在于应用，一般情况下对于一个正常人来说如果一个知识点他不反复理解三五次以及使用三五次以上是不可能达到这个层次的，这一点对于我来说同样也不例外。因此对于这个问题来说，相对于直接应用该模式，我选择了一种更加投机取巧的方式：对现有代码中的类进行类比，以此来判断是否能将该模式的模板直接套用在我们的项目中。**“课程”和”办公室、学生”就是满足这个模板的几个类。课程(Course) 作为“Element”，具有“选修课(OptCourse)”和“必修课(ManCourse) ”之分，而“办公室、学生”则作为访问该元素的访问者(Visitor)而存在。**

对于学生来说，如果他是一个已注册的学生，那么他具有能够选择选修课和删除选修课的能力，如果他是一个未注册的学生，那么它不具备任何选择课程的能力。而对于办公室来说，只有学生帮助办公室(Student Support Office)才具有删除课程和添加课程的能力，学生行政办公室(Student Admission Office则不具备该能力). 因此，依据这个想法为指导，访问者在该模式中的应用被实现如下:



三个Visitor: Student, StudentSupportOffice, StudentAdmissionOffice
两个Element(Course): ManCourse和OptCourse;

每个Visitor都要实现VisitOptCourse和VisitManCourse函数，在这两个函数中分别执行该角色对传入课程的逻辑。例如学生在已注册状态下可以修改选修课但无法操作必修课，学生行政办公室无法访问任何学生课程，学生帮助办公室则可以操作所有学生的所有必修课程和选修课程。

对于两个Course，只需要负责实现Accept方法。在Accept方法中调用Visitor的VisitXXXCourse方法(例如ManCourse的Accept方法中就只会调用Visitor.VisitManCourse)。accept方法的意义在于将自己这个实例传回给Visitor,然后Visitor根据实例方法判断调用哪个VisitXXX函数。这也就是Visitor模式的意义：对于相同的元素，不同的角色所访问的方法和逻辑不同。
这种模式相当于是将下列的If else 逻辑语句给解藕成了多个对象的形式来分别处理

If (this course is accessed by student): do something
Else if (this course is accessed by StudentSupportOffice): do something else.
…
..
到此为止，Visitor模式设计结束。我个人认为我的设计文档已经很详尽了，如果说你还有任何问题，请你再次阅读一遍该文档。若你还是看不懂，那我真没办法了。

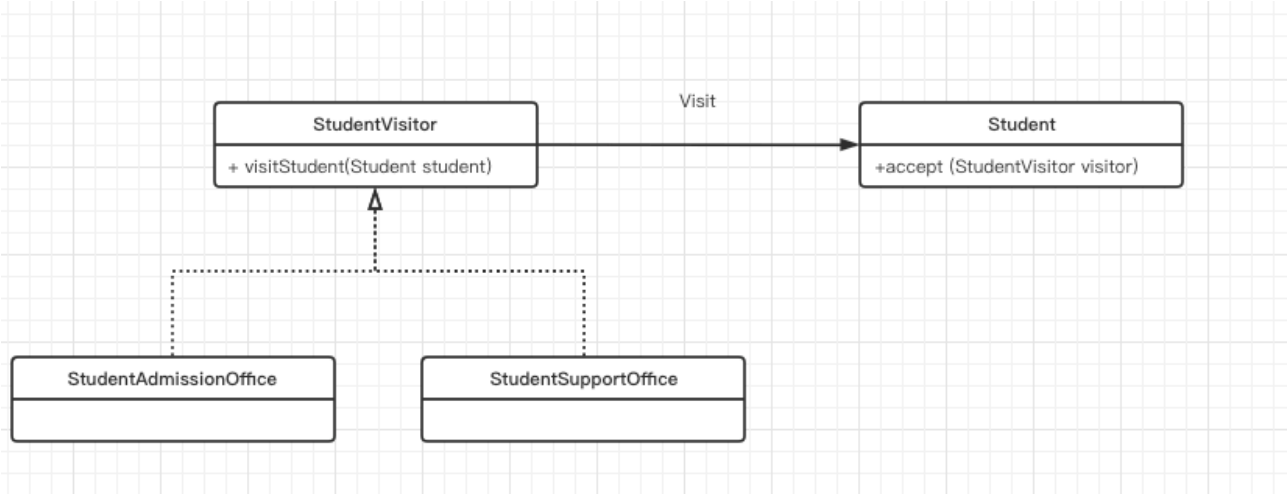
改动:

(After commit f436e):

Course的Visitor从原来的三个角色Student,StudentSupportOffice和StudentAdmissionOffice到现在仅仅变成了只有一个Student。这是因为我们认为课程这个元素不应当是办公室所访问的对象，而应该是学生访问的对象。

从逻辑上来说，办公室若要想增删课程，那么它应该访问的对象首先是学生，其次通过学生获取它的课表，进而实现删除和增加课程的功能。

因此按照这个实现的思想作为样本，新增了一个Visitor模式，大体如下:



现在，学生既是被访问者(被办公室访问)，也是访问者(访问课程)