迭代器模式(Iterator)

By:Zhi Li

1.什么是迭代器模式(What?)

迭代器模式提供一种方法顺序访问一个聚合对象中的各个元素，而又不暴露其内部的表示的行为型模式。

2.为什么使用迭代器模式(Why?)

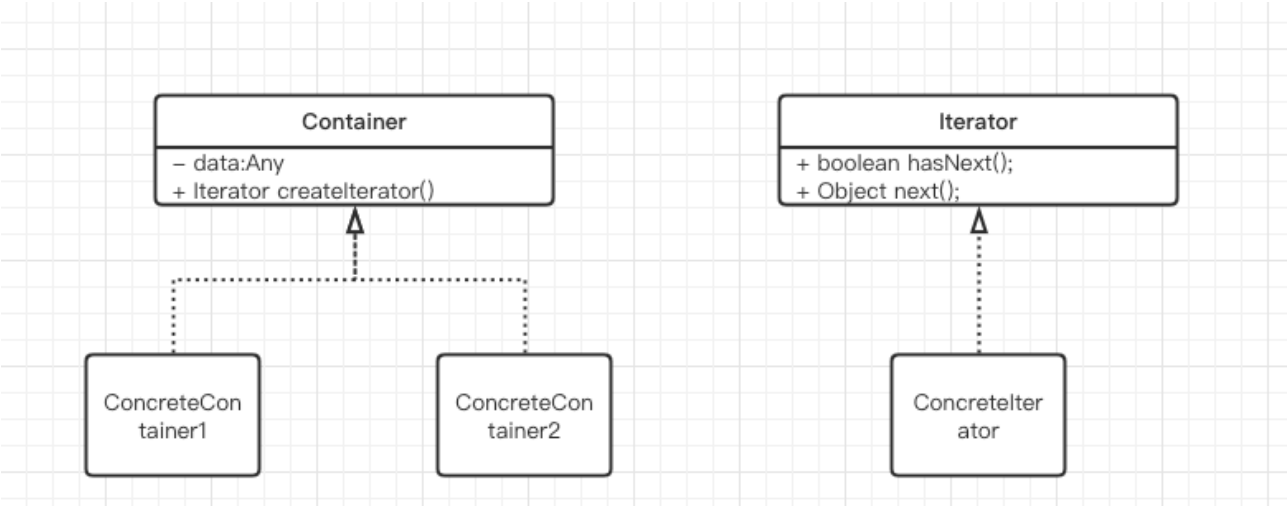
迭代器模式的使用动机就和它的定义相同，允许你在不暴露底层数据结构的前提之下（有些数据底层不一定全是数组，他们可能是树，也可能是链表或者堆等等），提供给客户端一种遍历元素的方式。因此，封装底层数据可以说是它唯一的价值。

3.怎么使用迭代器模式 (How?)

还是和之前的文档叙述方式相同，我在介绍一个模式的时候更喜欢从这个设计模式所具有的角色入手。迭代器模式所包含的角色只有两个：迭代器(Iterator)和数据容器(container)

迭代器定义的是遍历数据容器的方法，而数据容器则负责聚合和呈放你的数据结构。

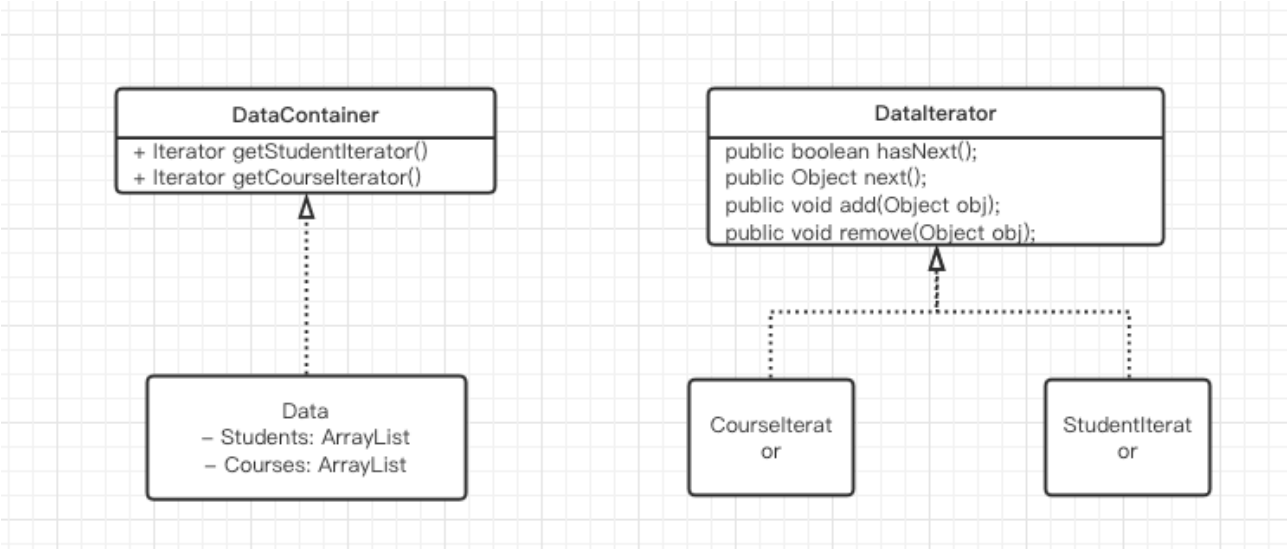
他们的类图如下所示:这里的演示仅仅只包含一些基本的函数，实际一个Iterator中包含的函数可以根据需求确定



4.在这个项目中如何使用?

将全体学生和课程公有化便是此次Iterator的用法：

具体见类图:



Data的底层数据结构为两个ArrayList, 分别是Students和Courses. 这两个ArrayList用来盛放该系统中的所有学生和课程（包括选修课和必修课）。这两个底层ArrayList全部被封装为private对象，并且不再设置get方法。唯一的访问方式就是通过Data类中的getStudentIterator或者getCourseIterator来创建课程迭代器或者学生迭代器的实例从而实现对学生或者课程的遍历。这两个Iterator都实现了add和remove方法，分别用于增加或者删除学生，增加或者删除课程。未来如果对于该迭代器还有任何额外的需求，都可以通过修改现有的DataIterator接口或者添加新的接口来完成。

由于引进了迭代器模式，因此有些许使用到了学生或者课程类的也因此受到了影响并发生了变化：

目前为止我仅仅只修改了Office相关的部分：包括Office,OfficeDecorator, StudentSupportOffice以及StudentAdmissionOffice。你会注意到这些办公室以前各自维护的学生列表也消失不见了。这是因为引入了迭代器模式后，我更希望的是所有的办公室通过一个共享的学生清单和课程清单来维护学生的信息。这个全局的清单就是Data对象, 它有且仅有一个，它在创建时会默认添加十个学生、十个选修课和十个必修课。因此，在今后的功能实现过程中，你如果需要访问某个学生的信息亦或者是课程的信息，那么请使用Data对象的getXXXIterator方法，然后使用该迭代器的Next方法遍历(具体遍历方式你可以参考我在Office里面写的遍历代码)。并且也请你不要再继续尝试实例化新的Data对象，因为它采取了单例模式设计，你new半天还是原来的那个实例。若需要使用，正确的做法是通过Data.getInstance()该静态方法获取。