理解引用语义的自定义类型

确定继承关系用于模拟“is a”的关系
(https://www.boxueio.com/series/understand-ref-types/ebook/177)

永远不要重定义继承而来的默认参数
(https://www.boxueio.com/series/understand-ref-types/ebook/180)

确定对象的组合用于模拟“has a”的关系

Back to series (/series/understand-ref-types)

字号
字号
默认主题
金色主题
暗色主题

当一个类定义中包含其它类的属性时，这两个类的关系，就叫做composition，其实我们经常使用这个概念。例如，定义一个表示PC的类：

理解HAS-A关系

```
class PC {
    var cpu: CPU
    var mem: MEM
    var motherBoard: MotherBoard
}
```

这段代码很好理解，一台PC当然需要包含CPU、内存、主板等核心部件。因此，这时PC和它的三个属性之间的composition，就表达了 HAS A 的关系。通常，我们不会对这类关系有任何困惑。

但是，当我们面对"is a"和"is implemented by"这两种关系的时候，呃，有时，就又会神不知鬼不觉的分不清彼此。我们继续看下面的例子。

区分“is a”和“is implemented by”

假设，我们要实现一个先进先出的队列 FIFOQueue 。出于对 NSMutableArray 的理解，我们知道，绝大多数代码都是可以复用的，我们只需要封装一部分特殊的逻辑就可以。然后，为了实现复用，你可能会写出下面的代码：

```
class FIFOQueue: NSMutableArray {

}
```

在我们动手实现一些必要的方法之前，想一下为了复用选择继承真的合理么？在上一节我们提到过，如果两个类存在继承关系，那么需要基类对象的地方，一定是可以使用一个派生类对象的。但显然，NSMutableArray 是一个支持随机读取的类型，但是 FIFOQueue 并不行。因此，一个 FIFOQueue 并不是一个 NSMutableArray 。我们要表达的，仅仅是借助 NSMutableArray 的一些既有代码来实现 FIFOQueue ，这就是我们刚才说的"is implemented by"的关系：

```
class FIFOQueue {
    var storage: NSMutableArray

    init(_ storage: NSMutableArray) {
        self.storage = storage
    }
}
```

这样，我们就可以添加队列的核心方法了：

```
extension FIFOQueue {
    var count: Int {
        return storage.count
    }

    func enqueue(element: Any) {
        self.storage.insert(element, at: 0)
    }

    func dequeue() -> Any? {
        let last = storage.lastObject
        self.storage.removeLastObject()

        return last
    }
}
```

这样，我们就只是借用 NSMutableArray ，实现了一个最简单的队列。

What's next?

通过 PC 和 FIFOQueue 这两个例子，我们可以了解到，Composition作为面向对象的另一种捏合类型的手段，和继承一样，它有着自己明确的语义。一方面，当我们用composition去模拟一个现实中存在的事物时，它的语义就是 has a ；另一方面，当我们通过composition借用已有代码来完成某个类型的实现细节时，它的语义就是 is implemented by 。当我们要在inheritance和composition之间作出选择时，通常，前者容易辨别，而后者，区别往往不那么明显，我们要格外小心，再三考量。

了解了面向对象设计中两个最重要的设计手段之后，下一节，我们来看一个开发细节。当我们通过继承获得的方法带有默认参数时，不要改写它，否则，会带来意想不到的结果。

◀ 确定继承关系用于模拟“is a”的关系
(https://www.boxueio.com/series/understand-ref-types/ebook/177)

永远不要重定义继承而来的默认参数 ▶
(https://www.boxueio.com/series/understand-ref-types/ebook/180)