3-2 作业解析：用队列实现栈

作业解析：使用队列实现栈

在这一小节，我们先来看一下，如何使用队列实现栈。

在我们的 MyStack 中，有一个私有的成员变量 Queue q。我们在构造函数中，需要给这个 q 开空间。

class MyStack {

```
private Queue<Integer> q;

/** Initialize your data structure here. */
public MyStack() {
    q = new LinkedList<>();
}

// empty 的实现很简单，直接调用 q 的 isEmpty 就好了：）
/** Returns whether the stack is empty. */
public boolean empty() {
    return q.isEmpty();
}

// .... 其他方法
```

}

在这里大家注意两点：

1）Leetcode 的这个问题不需要使用泛型，我们封装的数据结构，默认接受整型数据。

2）在这里，我们直接使用 Java 内置的 Queue。Java 中的 Queue 是一个接口，具体实例化它，需要选择一个数据结构。在这里，我们选择使用 LinkedList。Link
表，关于链表，我们将在下一章进行详细的介绍。

下面，我们来看如何实现其他操作。

对于一个栈来说，关键是栈顶在哪里。

栈是一端入，同一端出；而队列是一端入，另一端出。如果只给出一个队列，我们先假设，入队的一端是栈顶。

一旦这样定义，那么，我们自己封装的这个 MyStack 的入栈操作就很简单：直接把元素放入队列就好了。

```
public void push(int x) {
q.add(x);
}
```

关键是，这样一来，我们如何实现 pop？换句话说，我们如何拿到队列尾的那个元素？

因为此时，我们只能取出队首的元素，所以，要想拿到队尾的元素，我们就必须先把现在队列中的 n - 1 个元素都取出来。剩下的那一个元素，就是队尾的元素。

可是，取出的 n - 1 个元素我们不能扔掉，问题又限制我们必须使用队列这种数据结构，所以，此时，我们可以使用另外一个队列 q2，来存储从 q 中取出的所有元
后，q 里只剩下一个元素，就是我们要拿出的“栈顶元素”。将这个元素删除后，q2 里的数据就是原始的数据，我们用 q2 覆盖 q 就好。

下面是我的参考代码：

public int pop() {

```
// 创建另外一个队列 q2
Queue<Integer> q2 = new LinkedList<>();

// 除了最后一个元素，将 q 中的所有元素放入 q2
while (q.size() > 1)
    q2.add(q.remove());

// q 中剩下的最后一个元素就是“栈顶”元素
int ret = q.remove();
```