**难度: 简单**

请你仅使用两个栈实现先入先出队列。队列应当支持一般队列的支持的所有操作（push、pop、peek、empty）：

实现 MyQueue 类：

void push(int x) 将元素 x 推到队列的末尾

int pop() 从队列的开头移除并返回元素

int peek() 返回队列开头的元素

boolean empty() 如果队列为空，返回 true ；否则，返回 false

说明：

你只能使用标准的栈操作 —— 也就是只有 push to top, peek/pop from top, size, 和 is empty 操作是合法的。

你所使用的语言也许不支持栈。你可以使用 list 或者 deque（双端队列）来模拟一个栈，只要是标准的栈操作即可。

进阶：

你能否实现每个操作均摊时间复杂度为 O(1) 的队列？换句话说，执行 n 个操作的总时间复杂度为 O(n) ，即使其中一个操作可能花费较长时间。

示例：

输入：

["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

输出：

[null, null, null, 1, 1, false]

解释：

MyQueue myQueue = new MyQueue();

myQueue.push(1); // queue is: [1]

myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)

myQueue.peek(); // return 1

myQueue.pop(); // return 1, queue is [2]

myQueue.empty(); // return false

提示：

1 <= x <= 9

最多调用 100 次 push、pop、peek 和 empty

假设所有操作都是有效的 （例如，一个空的队列不会调用 pop 或者 peek 操作）

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/implement-queue-using-stacks

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

思路(使用两个栈来模拟队列):

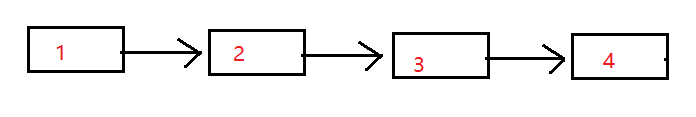
1). 第一个栈,正常进行push

2). 第二个栈,就是当进行pop的时候,把第一个栈的所以东西都依次pop进第二个栈,然后第二个栈pop出去之后,重写循环进第一个栈.

3). 当进行peek的时候也是相当于pop的,不过就自身查看,而不是抛出

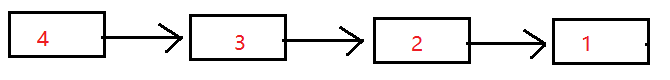
图解:

比如我们先依次push,1,2,3,4 (栈先进后出)



如果我们现在进行pop的时候是 4,3,2,1(栈先进后出)

此时我们先把这个栈的数据全部,pop到栈2,此时4是先进的,而1是最后进的



此时我们栈2pop的时候,就1了,相当于先进先出了

|  |
| --- |
| public class MyQueue {  private Deque<Integer> stack; //栈1  private Deque<Integer> stack2; //栈2  public MyQueue() {  stack = new ArrayDeque<>();  stack2 = new ArrayDeque<>();  }  public void push(int x) {  stack.push(x);  }  public int pop() {  stackSwap(stack,stack2);  int value = stack2.pop();  stackSwap(stack2,stack);  return value;  }  public int peek() {  stackSwap(stack,stack2);  int value = stack2.peek();  stackSwap(stack2,stack);  return value;  }  public boolean empty() {  return stack.isEmpty();  }  /\*\*  \*  \* @param s 要pop的栈  \* @param s2 接收pop的栈  \*/  private void stackSwap(Deque<Integer> s, Deque<Integer> s2) {  while(!s.isEmpty()) {  s2.push(s.pop());  }  }  } |