**难度: 简单**

实现一个MyQueue类，该类用两个栈来实现一个队列。

示例：

MyQueue queue = new MyQueue();

queue.push(1);

queue.push(2);

queue.peek(); // 返回 1

queue.pop(); // 返回 1

queue.empty(); // 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。

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-lcci

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。

**思路全部注释在代码里面(使用java自带的栈创建双栈):**

|  |
| --- |
| public class MyQueue {  private Stack<Integer> stack1;  private Stack<Integer> stack2;  /\*\* Initialize your data structure here. \*/  //在此初始化您的数据结构。  public MyQueue() {  //第一个栈用来存储  stack1 = new Stack<>();  //第二个栈用来辅助  stack2 = new Stack<>();  }  /\*\* Push element x to the back of queue. \*/  //将元素x推到队列的后面。  public void push(int x) {  //使用第一个栈进行存储  stack1.push(x);  }  /\*\* Removes the element from in front of queue and returns that element. \*/  //从队列前面删除该元素并返回该元素。  public int pop() {  //判断栈1是否为空 这里如果不为空是 false, 所以用! 取反  while (!stack1.empty()) {  //把栈1的值,全部放到栈2  //由于栈是先进后出, 所以会把循序进行颠倒  //例如: 存储 1 2 3  //那么用于先进后出 就把 3 2 1 存储进 栈2了  stack2.push(stack1.pop());  }  //这里抛出栈2的值, 由于先进后出的原则 3 2 1 此时抛出 1  int result = stack2.pop();  //判断栈2是否为空 这里如果不为空是 false, 所以用! 取反  while (!stack2.empty()) {  //此时把刚刚颠倒的顺序还原而已  stack1.push(stack2.pop());  }  return result;  }  /\*\* Get the front element. \*/  //获取前元素。  public int peek() {  //判断栈1是否为空 这里如果不为空是 false, 所以用! 取反  while (!stack1.empty()) {  //由于栈是先进后出, 所以会把循序进行颠倒  stack2.push(stack1.pop());  }  //偷窥值  int result = stack2.peek();  while (!stack2.empty()) {  //此时把刚刚颠倒的顺序还原  stack1.push(stack2.pop());  }  return result;  }  /\*\* Returns whether the queue is empty. \*/  //返回队列是否为空。  public boolean empty() {  return stack1.empty();  }  } |