232 用栈实现队列

Label：栈、队列

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

- 两个栈 实现 队列

```
class MyQueue {

    Stack<Integer> stack = null;
    Stack<Integer> stackQ = null;

    /** Initialize your data structure here. */
    public MyQueue() {
        stack = new Stack<>();
        stackQ = new Stack<>();
    }

    /** Push element x to the back of queue. */
    public void push(int x) {

        while (!stackQ.isEmpty()) {
            stack.add(stackQ.pop());
        }
        stackQ.push(x);
        while (!stack.isEmpty()) {
            stackQ.add(stack.pop());
        }
    }

    /** Removes the element from in front of queue and returns that element. */
    public int pop() {
        return stackQ.pop();
    }

    /** Get the front element. */
    public int peek() {
        return stackQ.peek();
    }

    /** Returns whether the queue is empty. */
    public boolean empty() {
        return stackQ.isEmpty();
    }
}
```

- 优化时间复杂度

```
class MyQueue {

    private Stack<Integer> a = null;  // 用来存储 临时加入队列的元素，b为空了，再将累计的元素倒进B
    private Stack<Integer> b = null;  // 用来存储 队列元素

    public MyQueue() {
         a = new Stack<Integer>();
         b = new Stack<Integer>();
    }

    public void push(int x) {
        a.push(x);
    }

    public int pop() {   // 这样就可以累积一定量之后再倒腾，优化时间复杂度
        //如果栈b不为空，直接pop即可
        if(!b.isEmpty()) {
            return b.pop();
        }
        //如果栈b为空，需要先将栈a中的数据倒腾到栈b中，再pop
        while(!a.isEmpty()) {
            b.push(a.pop());
        }
        return b.pop();
    }

    public int peek() {
        //如果栈b不为空，直接peek即可
        if(!b.isEmpty()) {
            return b.peek();
        }
        //如果栈b为空，需要先将栈a中的数据倒腾到栈b中，再peek
        while(!a.isEmpty()) {
            b.push(a.pop());
        }
        return b.peek();
    }

    public boolean empty() {
        //栈a和栈b需要同时判空
        return a.isEmpty() && b.isEmpty();
    }
}
```