题目 用两个栈实现队列

考点 栈和队列 热点指数 94419 通过率 35.74%

具体题目

用两个栈来实现一个队列，完成队列的Push和Pop操作。 队列中的元素为int类型。

栈是只能进出栈都在栈顶，而队列是队尾进队，队头出队。所以很明显，直接用栈是不行的。
　然后进队列是在队尾，而栈进栈也可以看做是栈尾。所以很明显，进队列操作和进栈操作是一样的，所以只需要使用一个栈就行了，这里我们假设使用stack1。
所以接下来我们只要考虑如何使用两个栈使得出队操作是输出栈头部元素。输出头部元素，我们必须将除了第一个进栈的元素外的其他元素全部出栈，比如进栈操作12345，必须将2345出栈。我们很容易想到将2345存在另一个栈中(stack2.push(stack1.pop()))，此时stack2顺序是5432。然后将1出栈赋值给一个变量(result = stack1.pop())，这个变量就是return的值。然后再讲stack2全部出栈赋值给stack1，此时stack1中元素就为2345(stack1.push(stack2.pop()))。
但是这里最容易出错的地方：其实是循环，一开始我使用的是for(int i = 0;i < stack1.size();i++)，发现答案是错误的。因为stack1.pop()之后，satck1.size()也发生了变化。导致执行了2次循环后就满足条件了。因此，在stack1出栈操作之前，使用一个变量存储stack1的初始长度。

```
    public void push(int node) {
        stack1.push(node);
    }

    public int pop() {
        if(stack2.empty()){
            while(!stack1.empty()){
                stack2.push(stack1.pop());
            }
        }
        return stack2.pop();
    }
```

题目描述 用两个栈来实现一个队列，完成队列的Push和Pop操作。 队列中的元素为int类型。 解题思路 队列是先进先出，栈是先进后出，如何用两个栈来实现这种先进先出呢？ 其实很简单，我们假设用stack1专门来装元素，那么直接stack1.pop肯定是不行的，这个时候stack2就要发挥作用了。 我们的规则是：只要stack2中有元素就pop，如果stack2为空，则将stack1中所有元素倒进satck2中，就是说，新元素只进stack1，元素出来只从stack2出来。 这样子，就能保证每次从stack2中pop出来的元素就是最老的元素了。 我的答案

```
import java.util.Stack;
public class Solution{
    //负责装元素
    Stack<Integer> stack1 = new Stack<Integer>();
    //负责出元素
    Stack<Integer> stack2 = new Stack<Integer>();
    public void push(int node) {
        stack1.push(node);
    }
    //主要思想是：stack2有元素就pop，没有元素就将stack1中所有元素倒进来再pop
    public int pop() throws Exception{
        if(!stack2.isEmpty()){
            int node = stack2.pop();
            return node;
        }else{
            if(stack1.isEmpty()){
                throw new Exception("no valid element");
            }
            while(!stack1.isEmpty()){
                stack2.push(stack1.pop());
```

```
        }
        return stack2.pop();
    }
}
```