用两个栈实现队列_侯彦庆

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。

思想：
两个栈实现一个队列（先进先出）
删除一个元素步骤：
入队：将元素压入栈1；
出队：判断栈2是否为空：
如果栈2为空，将栈1中所有元素pop,并push压入栈2，栈2依次弹出即可；
如果栈2非空，则栈2中栈顶元素就是最先入队列的元素，直接弹出即可；
两个队列实现一个栈（后进先出）
入栈：将元素进队列1
出栈：判断队列1中元素个数是否为1
如果个数为1，则出队列；
如果个数大于1，将队列1中元素依次出队列放入队列2，直到队列1中元素个数为1，然后队列1中元素出队即可，再把队列2中元素出队列依次放入队列1中；

```
import java.util.Stack;
public class Solution {
    Stack<Integer> stack1 = new Stack<Integer>();
    Stack<Integer> stack2 = new Stack<Integer>();
    public void push(int node)    {
        stack1.push(node);
    }
    public int pop()    {
        int val;
        //栈2为空，将栈1push
        if(stack2.empty())     {
            while(!stack1.empty()) {
                //栈1不为空
                val=stack1.pop();
                stack2.push(val);
            }
        }
        //栈2不为空，直接弹出栈顶元素返回
        val=stack2.pop();
        return val;
    }
}
```

两个队列实现一个栈

```
//两个队列实现一个栈
import java.util.LinkedList;
import java.util.Queue;
public class QueueToStack {
    LinkedList<Integer> queue1=new LinkedList<>();
    LinkedList<Integer>queue2=new LinkedList<>();
```

```
    public void push(int val){
        queue1.addLast(val);
    }
    public int pop(){
        //出队列
        if((queue1.size()+queue2.size())!=0){
            //队列不为空
            if(!queue1.isEmpty()){
                //队列1不为空
                putN_1ToQueue2();
                //将队列1前N-1个元素进入队列2，再将剩下的一个元素出队列
                return queue1.removeFirst();
            } else{//否则队列1空，
                putN_1ToQueue2();//队列2不为空，则将队列2元素加入队列1尾部
                return queue2.removeFirst();
            }
        }else{
            System.out.println("队列空");
            return -1;
        }
    }
    public void putN_1ToQueue2(){
        //从非空中出队n-1个到另一个队列    因为队列总是一空一非空
        if(!queue1.isEmpty()){
            //队列1不为空，如果队列1长度>1,则将队列1元素加到队列2尾部，直到队列1只剩下一个元
素。
            while(queue1.size()>1){
                queue2.addLast(queue1.removeFirst());
            }
        }else if(!queue2.isEmpty()){
            //队列1为空（最后一个出队列了），队列2不为空，则将队列2元素循环加入到队列1尾部
            while(queue2.size()>1){
                queue1.addLast(queue2.removeFirst());
            }
        }
    }
    //主函数
    public static void main(String[] args) {
        QueueToStack stack=new QueueToStack();
        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
    }
}
```