```
public class MyStack {

    Queue<Integer> q1 = new LinkedList<>();
    Queue<Integer> q2 = new LinkedList<>();
    /** Initialize your data structure here. */
    public MyStack() {

    }

    /** Push element x onto stack. */
    public void push(int x) {
        q1.offer(x);

        /*
        if (q1.isEmpty()){
            q1.offer(x);
            while (!q2.isEmpty()){
                q1.offer(q2.poll());
            }
        } else {
            q2.offer(x);
            while (!q1.isEmpty()){
                q2.offer(q1.poll());
            }
        }
        */
    }

    /** Removes the element on top of the stack and returns that element. */
    public int pop() {
        while (q1.size() > 1){
            q2.offer(q1.poll());
        }
        int res = q1.poll();

        while (q2.size() > 0){
            q1.offer(q2.poll());
        }

        return res;
        /*
        if (q1.size() > 0) {
            return q1.poll();
        } else {
            return q2.poll();
        }
        */
    }

    /** Get the top element. */
    public int top() {
        while (q1.size() > 1){
            q2.offer(q1.poll());
        }
        int res = q1.peek();
        q2.offer(q1.poll());

        while (q2.size() > 0){
            q1.offer(q2.poll());
        }

        return res;

        /*
        if (q1.size() > 0) {
            return q1.peek();
        } else {
            return q2.peek();
        }
        */
    }

    /** Returns whether the stack is empty. */
    public boolean empty() {
        return q1.isEmpty() && q2.isEmpty();
    }
}

/**
 * Your MyStack object will be instantiated and called as such:
 * MyStack obj = new MyStack();
 * obj.push(x);
 * int param_2 = obj.pop();
 * int param_3 = obj.top();
 * boolean param_4 = obj.empty();
 */
```