Implement Stack using Queues

Implement the following operations of a stack using queues.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- empty() -- Return whether the stack is empty.

Notes:

- You must use *only* standard operations of a queue -- which means only `push to back`, `peek/pop from front`, `size`, and `is empty` operations are valid.
- Depending on your language, queue may not be supported natively. You may simulate a queue by using a list or deque (double-ended queue), as long as you use only standard operations of a queue.
- You may assume that all operations are valid (for example, no pop or top operations will be called on an empty stack).

Update (2015-06-11):
The class name of the **Java** function had been updated to **MyStack** instead of Stack.

Credits:
Special thanks to @jianchao.li.fighter for adding this problem and all test cases.

Solution 1

```
class Stack {
public:
    queue<int> que;
    // Push element x onto stack.
    void push(int x) {
        que.push(x);
        for(int i=0;i<que.size()-1;++i){
            que.push(que.front());
            que.pop();
        }
    }

    // Removes the element on top of the stack.
    void pop() {
        que.pop();
    }

    // Get the top element.
    int top() {
        return que.front();
    }

    // Return whether the stack is empty.
    bool empty() {
        return que.empty();
    }
};
```

written by sjtuldk original link here

Solution 2

```
class MyStack
{
    Queue<Integer> queue;

    public MyStack()
    {
        this.queue=new LinkedList<Integer>();
    }

    // Push element x onto stack.
    public void push(int x)
    {
       queue.add(x);
       for(int i=0;i<queue.size()-1;i++)
       {
           queue.add(queue.poll());
       }
    }

    // Removes the element on top of the stack.
    public void pop()
    {
        queue.poll();
    }

    // Get the top element.
    public int top()
    {
        return queue.peek();
    }

    // Return whether the stack is empty.
    public boolean empty()
    {
        return queue.isEmpty();
    }
}
```

written by YYZ90 original link here

Solution 3

```
class MyStack {
    Queue<Integer> q = new LinkedList<Integer>();

    // Push element x onto stack.
    public void push(int x) {
        q.add(x);
    }

    // Removes the element on top of the stack.
    public void pop() {
        int size = q.size();
        for(int i = 1; i < size; i++)
            q.add(q.remove());
        q.remove();
    }

    // Get the top element.
    public int top() {
        int size = q.size();
        for(int i = 1; i < size; i++)
            q.add(q.remove());
        int ret = q.remove();
        q.add(ret);
        return ret;
    }

    // Return whether the stack is empty.
    public boolean empty() {
        return q.isEmpty();
    }
}
```

written by leo_aly7 original link here