```
class MaxStack {

    Stack<Integer> s1;
    Stack<Integer> s2;
    public MaxStack() {
        s1 = new Stack<>();
        s2 = new Stack<>();
    }

    public void push(int x) {
        pushHelper(x);
    }

    private void pushHelper(int x) {
        int tmp = s2.isEmpty() ? Integer.MIN_VALUE : s2.peek();
        tmp = Math.max(tmp, x);
        s1.push(x);
        s2.push(tmp);
    }

    public int pop() {
        s2.pop();
        return s1.pop();
    }

    public int top() {
        return s1.peek();
    }

    public int peekMax() {
        return s2.peek();
    }

    public int popMax() {
        int max = s2.peek();
        Stack<Integer> tmpStack = new Stack<>();
        while (s1.peek() != max) {
            tmpStack.push(s1.pop());
            s2.pop();
        }
        s1.pop();
        s2.pop();
        while (!tmpStack.isEmpty()) {
            int x = tmpStack.pop();
            pushHelper(x);
        }
        return max;
    }
}
```

1. push(x) -- Push element x onto stack.
2. pop() -- Remove the element on top of the stack and return it.

3. top() -- Get the element on the top.
4. peekMax() -- Retrieve the maximum element in the stack.
5. popMax() -- Retrieve the maximum element in the stack, and remove it. If you find more than one maximum elements, only remove the top-most one.