Min Stack

Solution

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Example 1:

```
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

Constraints:

- Methods `pop`, `top` and `getMin` operations will always be called on **non-empty** stacks.

Show Hint #1

Java

```java
// using 2 stacks
// stack 1: 3,5,1,3,6
// stack 2: 3,3,1,1,1
class MinStack {

    Stack<Integer> stack;
    Stack<Integer> minStack;

    /** initialize your data structure here. */
    public MinStack() {
        stack = new Stack<>();
        minStack = new Stack<>();
    }

    public void push(int x) {
        stack.push(x);
        minStack.push(Math.min(x, isEmpty() ? x : getMin()));
    }

    public void pop() {
        stack.pop();
        minStack.pop();
    }

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

    public int getMin() {
        return minStack.peek();
    }
```

```
32
33      private boolean isEmpty() {
34          return minStack.isEmpty();
35      }
36  }
37
38  /**
39   * Your MinStack object will be instantiated and called as such:
40   * MinStack obj = new MinStack();
41   * obj.push(x);
42   * obj.pop();
43   * int param_3 = obj.top();
44   * int param_4 = obj.getMin();
45   */
```

Custom Testcase (Contribute !)

Run Code Submit