Possible Inputs…

1 - Push the element x into the stack.

2 - Delete the element present at the top of the stack.

3 - Print the maximum element in the stack.

**When size of stack goes beyond 100000000. Intensive CPU is needed to get the maximum element using collection APIs.**

**Example:**

**Stack stack = new LinkedList();**

**Collections.max(stack);//This method will give max value. But below code will give max at o(1). Below code stores max value for each element in stack.**

**package** com.sample.basics;

**import** java.util.Scanner;

**import** java.util.Stack;

**class** Dummy {

**private** **static** **class** StackNode {

**int** val;

**int** curMax;

**public** StackNode(**int** val, **int** curMax) {

**this**.val = val;

**this**.curMax = curMax;

}

**public** String toString() {

**return** val + " [" + curMax + "]";

}

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**int** n = sc.nextInt();

**int** max = Integer.***MIN\_VALUE***;

Stack<StackNode> stack = **new** Stack<StackNode>();

**while** (n > 0) {

**int** choice = sc.nextInt();

**if**(choice == 1) {

**int** val = sc.nextInt();

max = Math.*max*(val, max);

stack.add(**new** StackNode(val, max));

} **else** **if**(choice == 2) {

**if**(!stack.isEmpty())

stack.pop();

// reset max

**if**(stack.isEmpty())

max = Integer.***MIN\_VALUE***;

**else**

max = stack.peek().curMax;

} **else** **if**(choice == 3) {

**if**(!stack.isEmpty()) {

System.***out***.println(stack.peek().curMax);

}

}

n--;

}

sc.close();

}

}