```
class Solution {

    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> largeK = new PriorityQueue<Integer>(k + 1);

                for(int el : nums) {
                    largeK.add(el);
                    if (largeK.size() > k) {
                        largeK.poll();
                    }
                }

                return largeK.poll();
    }


    public int findKthLargest(int[] nums, int k) {
        return helper(nums, nums.length - k, 0, nums.length-1);
    }

    private int helper(int[] nums, int k, int l, int r) {
        int pivot = nums[r];
        int index = l;
        for (int i = l; i < r; i++) {
            if (nums[i] < pivot) {
                swap(nums, i, index++);
            }
        }
        swap(nums, index, r);

        if (index == k) return nums[k];
        else if (index > k) {
            return helper(nums, k, l, index-1);
        } else {
            return helper(nums, k, index+1, r);
        }
    }

    private void swap(int[] nums, int i, int j) {
        int v = nums[i];
        nums[i] = nums[j];
        nums[j] = v;
    }
}
```

Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example 1:

```
Input: [3,2,1,5,6,4] and k = 2
```

Output: 5

Example 2:

Input: [3,2,3,1,2,4,5,5,6] and k = 4
Output: 4