Given an array of integers, every element appears *twice* except for one. Find that single one.

**Note:**  
Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**import** java.util.\*;

**public** **class** Solution {

**public** **int** singleNumber(**int**[] nums) {

**if**(nums.length == 1)

**return** nums[0];

ArrayList<Integer> num = **new** ArrayList<Integer>();

**for**(**int** i=0; i<nums.length ; i++) {

num.add(i,nums[i]);

}

**for**(**int** i=0 ; i<nums.length ; i++) {

**int** j = num.get(i);

**if**(num.indexOf(j)==num.lastIndexOf(j)) **return** j;

}

**return** 0;

}

}

import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.Iterator;  
  
public class Solution {  
  
public int singleNumber1(int[] nums) {  
if(nums.length == 1)   
return nums[0];  
ArrayList<Integer> num = new ArrayList<Integer>();  
for(int i=0; i<nums.length ; i++)  
num.add(i,nums[i]);  
  
for(int i=0 ; i<nums.length ; i++) {  
int j = num.get(i);  
if(num.indexOf(j) == num.lastIndexOf(j))  
return j;  
}  
return 0;  
}  
  
  
public int singleNumber2(int[] nums) {  
HashSet<Integer> h=new HashSet<Integer>();  
for(int i=0; i<nums.length ; i++)  
if(h.contains(nums[i]))  
h.remove(nums[i]);  
else  
h.add(nums[i]);  
  
Iterator<Integer> iterator = h.iterator();  
return (int) iterator.next();  
}  
  
  
public int singleNumber3(int[] nums) {  
    int res = 0;  
    for (int i = 0; i < nums.length; i++)  
        res = res^nums[i];  
    return res;  
}  
  
  
public static void main(String args[]) {  
int [] a={1,2,1,2,3};  
System.out.println(new Solution().singleNumber3(a));  
}  
}