

# First Occurrence of a Number#

When given an array, find the first occurrence of a given number in that array and return the index of that number.

The following illustration explains how to approach this problem.

Array

0	1	2	3	4	5	6
2	3	4	1	7	8	3

First index of 3

0	1	2	3	4	5	6
2	3	4	1	7	8	3

Output : 1

## Implementing the Code#

The following code explains how to find the first occurrence of a number in an array.

Experiment with the code by changing the values of `array` and `num` to see how it works!

```
1 class ArrayClass {
2
3     private static int firstOccurrence(int[] a, int n, int currentIndex) {
4         if (a.length == currentIndex) {
5             return -1;
6         }
7         else if (a[currentIndex] == n) {
8             return currentIndex;
9         }
10        else {
11            return firstOccurrence(a, n, currentIndex+1);
12        }
13    }
14
15    public static void main(String[] args) {
16        System.out.print("{");
17
18        int[] array = {2,3,4,1,7,8,3};
19        for (int i = 0; i < array.length; i++) {
20            System.out.print(array[i] + " ");
21        }
22        System.out.println("}");
23
24        int num = 3;
25
26        int result = firstOccurrence(array, num, 0);
27        System.out.println("The first occurrence of the number " + num + " is at index: " + result);
28    }
}
```

## Understanding the Code#

The code snippet above can be broken down into **two parts**. The **recursive method** and the **main** where the method is called.

### Driver Method#

The driver code is found from **line 16 to line 27**.

- From **line 18 to line 24** we have an `array` equal to of 7 and the number `num` whose first occurrence is to be found respectively.
- On **line 26** the method `firstOccurrence` is defined and takes in 3 arguments i.e., `array`, `num` and `0` (i.e. the first index). When the method is called, it returns the index of the first occurrence `num`.

### Recursive Method#

In the recursive method, we define the **base case** and the **recursive case**.

#### Base Case#

The method terminates when one of the following conditions is met:

- The first base case is defined from **line 4 to line 6**. If the whole array has been traversed and the element is not found,the method returns -1.
- The second base case is defined from **line 7 to line 9**. If the first occurrence of the number is found, the index of that number is returned.

#### Recursive Case#

The recursive case is called on **line 11**.

- The method takes in three arguments. The first argument is the array `a`. The second is the number `n` whose first occurrence is to be located. The third and argument is the `currentIndex` which is incremented in successive recursive calls.
- Initially, the value of the `currentIndex` is 0. Each time the recursive method is invoked, the value at the `currentIndex` of the array is compared with the number `n` whose first occurrence is to be located. If the result of the comparison is true, the method terminates and returns the index of the `n`.If the result of the comparison is false, the method **increments the `currentIndex`** in order to compare `n` with the value stored at the next index of the array. It then keeps comparing until it reaches the base case.

## Understanding Through a Stack#

The following illustration explains this concept:

Find first occurrence of 3

0	1	2	3	4	5	6
2	3	5	7	9	1	3

Stack is empty

1 of 6

0	1	2	3	4	5	6
2	3	5	7	9	1	3

↑

array, 3, 0 ← top

if array[0]==3 => False

2 of 6

0	1	2	3	4	5	6
2	3	5	7	9	1	3

↑

array, 3,1 ← top

array, 3,0

if array[1]==3 => True  
//base case  
stack starts popping at this point

3 of 6

array, 3,1 ← top

array, 3,0

4 of 6

array, 3,0 ← top

5 of 6

Stack is empty

Function returns index of 3

Output:1

0	1	2	3	4	5	6
2	3	5	7	1	9	3

6 of 6