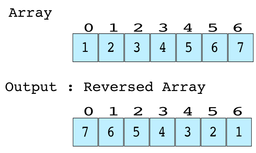
Invert the Position of Elements

Given an array, you must reverse the position of the elements in the array.



Implementing the Code

The following code explains the concept of reversing the elements of an array using recursion.

Experiment with the code by changing the `array` elements and `size` to see how the answer changes.

Recursive | Iterative

```
import java.io.*;

class ExampleClass {

  private static void invert(int[] array, int currentIndex) {
    if (currentIndex < array.length/2) {
      // swap array[currentIndex] and array[array.length-1-currentIndex]
      int temp = array[currentIndex];
      array[currentIndex] = array[array.length-1-currentIndex];
      array[array.length-1-currentIndex] = temp;

      invert(array, currentIndex+1);
    }
  }

  public static void main(String[] args) {
    System.out.println("Before: ");

    int[] array = {1,2,3,4,5,6,7};
    System.out.print("{ ");
    for (int i = 0; i < array.length; i++) {
      System.out.print(array[i] + " ");
    }
    System.out.println("} ");

    System.out.println("After: ");

    invert(array, 0);
```

Run | Save | Reset

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 between **lines 17** and **line 34**.

- In the driver code, we have an `array` that equals to 7.
- The method takes in 2 arguments: `array` and `0` (i.e.,the starting index value).
- The array gets completely reversed until the completion of the method's execution.
- The array is printed using a `for` loop.

Recursive Method

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

Base Case

The method terminates when the following condition is met:

- If the value of the `currentIndex` exceeds or is equal to half the size of the array, the method terminates. Otherwise, it continues executing the code that follows and eventually calls the recursive method.

Recursive Case

The recursive case is called on **line 12**.

- The method takes in two arguments. The first is the `array`. The second is the `currentIndex` which is the starting index of the array. The `currentIndex` is updated and incremented by 1 in each successive recursive call.
- Before the recursive call is made, we **swap** the values of the last index with the first index and continue modifying the value of the `currentIndex` in order to traverse through the array entirely.
- Initially, the value of `currentIndex` is 0, and the value at `array[0]` is swapped with the value at `array[7-1-0]`, meaning the size of the array is subtracted by 1 and the value of the `currentIndex`. The recursive method `invert` is then called.
- In each recursive call,
  - From **line 8 to line 10**, the values of the array are swapped using the process explained above.
  - The value of the `currentIndex` **is increased by 1** in order to update the value of the index and swap the values of the array at these indices in the next recursive call.
- The method is continually called through recursion until the base condition is met.

Understanding Through a Stack

The following illustration explains this concept:

1 of 6

Now that you have learned about recursion with arrays, let's check your knowledge by giving you a challenge in the next lesson.