

LAB # 04

ARRAYS IN JAVA

OBJECTIVE:

To understand arrays and its memory allocation.

LAB TASKS:

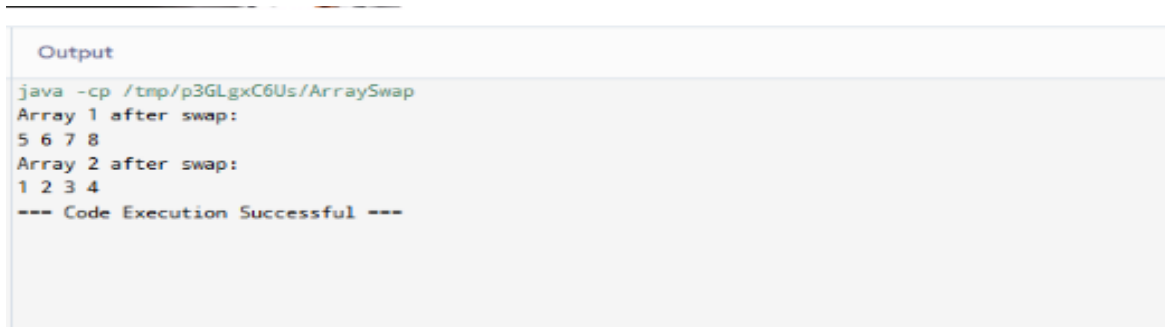
1. Write a program that takes two arrays of size 4 and swap the elements of those arrays.

CODE:



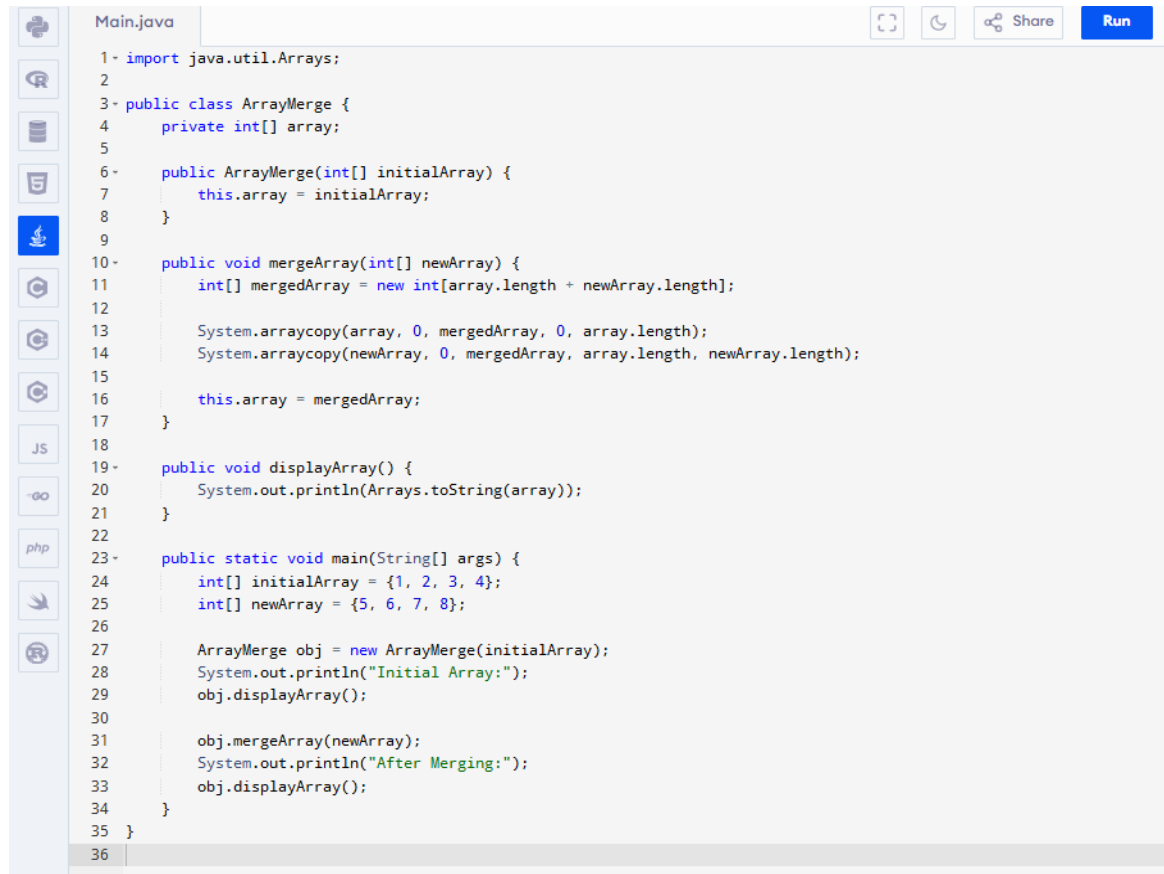
```
Main.java
1- public class ArraySwap {
2-     public static void main(String[] args) {
3         int[] array1 = {1, 2, 3, 4};
4         int[] array2 = {5, 6, 7, 8};
5
6         // Swap elements
7-         for (int i = 0; i < array1.length; i++) {
8             int temp = array1[i];
9             array1[i] = array2[i];
10            array2[i] = temp;
11        }
12
13        System.out.println("Array 1 after swap:");
14        for (int i : array1) System.out.print(i + " ");
15
16        System.out.println("\nArray 2 after swap:");
17        for (int i : array2) System.out.print(i + " ");
18    }
19 }
20
```

OUTPUT:

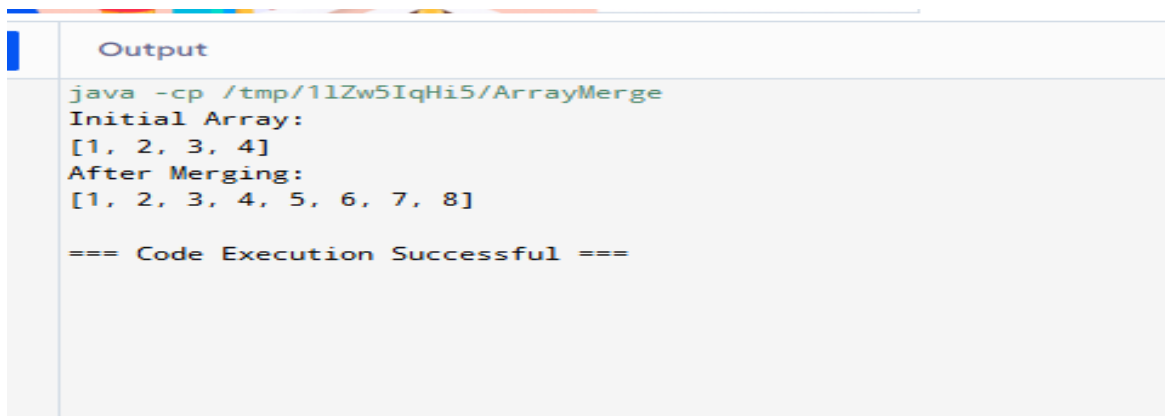


```
Output
java -cp /tmp/p3GLgx6Us/ArraySwap
Array 1 after swap:
5 6 7 8
Array 2 after swap:
1 2 3 4
--- Code Execution Successful ---
```

2. Add a method in the class that takes array and merge it with the existing one.

CODE:

```
1- import java.util.Arrays;
2
3- public class ArrayMerge {
4     private int[] array;
5
6     public ArrayMerge(int[] initialArray) {
7         this.array = initialArray;
8     }
9
10    public void mergeArray(int[] newArray) {
11        int[] mergedArray = new int[array.length + newArray.length];
12
13        System.arraycopy(array, 0, mergedArray, 0, array.length);
14        System.arraycopy(newArray, 0, mergedArray, array.length, newArray.length);
15
16        this.array = mergedArray;
17    }
18
19    public void displayArray() {
20        System.out.println(Arrays.toString(array));
21    }
22
23    public static void main(String[] args) {
24        int[] initialArray = {1, 2, 3, 4};
25        int[] newArray = {5, 6, 7, 8};
26
27        ArrayMerge obj = new ArrayMerge(initialArray);
28        System.out.println("Initial Array:");
29        obj.displayArray();
30
31        obj.mergeArray(newArray);
32        System.out.println("After Merging:");
33        obj.displayArray();
34    }
35 }
36
```

OUTPUT:

```
Output
java -cp /tmp/1lZw5IqHi5/ArrayMerge
Initial Array:
[1, 2, 3, 4]
After Merging:
[1, 2, 3, 4, 5, 6, 7, 8]

=== Code Execution Successful ===
```

3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not.

CODE:

```
Main.java
1- public class PalindromeCheck {
2-     public static boolean isPalindrome(String str) {
3-         int start = 0, end = str.length() - 1;
4-         while (start < end) {
5-             if (str.charAt(start) != str.charAt(end)) return false;
6-             start++;
7-             end--;
8-         }
9-         return true;
10    }
11
12-    public static void main(String[] args) {
13-        String[] words = {"ZAINAB", "AHMED"};
14
15-        for (String word : words) {
16-            if (isPalindrome(word)) {
17-                System.out.println(word + " is a palindrome.");
18-            } else {
19-                System.out.println(word + " is not a palindrome.");
20-            }
21-        }
22    }
23 }
24
25
```

OUTPUT:

```
Output
java -cp /tmp/s68VUOPma1/PalindromeCheck
ZAINAB is not a palindrome.
AHMED is not a palindrome.

=== Code Execution Successful ===
```

4. Given an array of integers, count how many numbers are even and how many are odd.

CODE:

```
Main.java
1- public class EvenOddCount {
2-     public static void main(String[] args) {
3-         int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8};
4-         int evenCount = 0, oddCount = 0;
5-
6-         for (int num : numbers) {
7-             if (num % 2 == 0) {
8-                 evenCount++;
9-             } else {
10-                 oddCount++;
11-             }
12-         }
13-
14-         System.out.println("Even numbers count: " + evenCount);
15-         System.out.println("Odd numbers count: " + oddCount);
16-     }
17- }
18-
19-
```

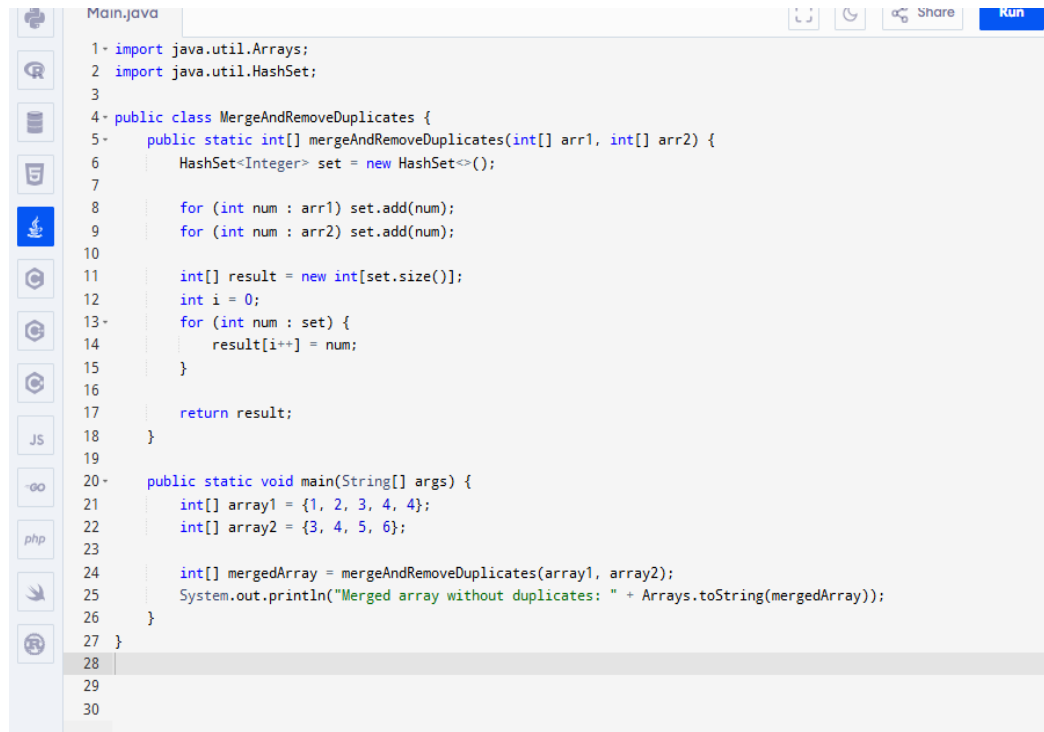
OUTPUT:

```
Output
java -cp /tmp/G6eWswiljQ/EvenOddCount
Even numbers count: 4
Odd numbers count: 4

=== Code Execution Successful ===
```

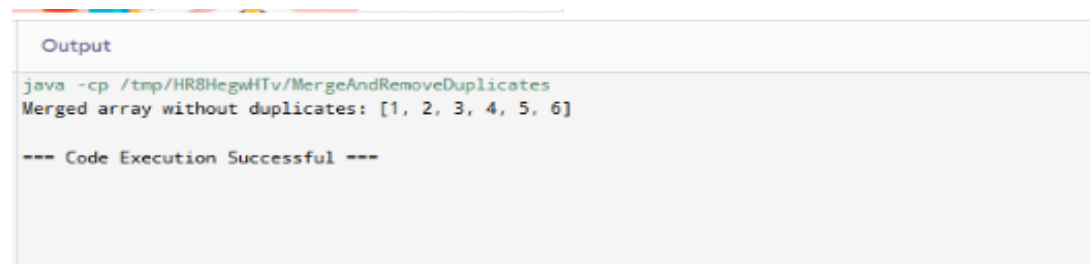
5. Given two integer arrays, merge them and remove any duplicate values from the resulting array

CODE:



```
1- import java.util.Arrays;
2 import java.util.HashSet;
3
4- public class MergeAndRemoveDuplicates {
5-     public static int[] mergeAndRemoveDuplicates(int[] arr1, int[] arr2) {
6         HashSet<Integer> set = new HashSet<>();
7
8         for (int num : arr1) set.add(num);
9         for (int num : arr2) set.add(num);
10
11         int[] result = new int[set.size()];
12         int i = 0;
13         for (int num : set) {
14             result[i++] = num;
15         }
16
17         return result;
18     }
19
20-     public static void main(String[] args) {
21         int[] array1 = {1, 2, 3, 4, 4};
22         int[] array2 = {3, 4, 5, 6};
23
24         int[] mergedArray = mergeAndRemoveDuplicates(array1, array2);
25         System.out.println("Merged array without duplicates: " + Arrays.toString(mergedArray));
26     }
27 }
28
29
30
```

OUTPUT:



```
Output
java -cp /tmp/HR8HegwHTv/MergeAndRemoveDuplicates
Merged array without duplicates: [1, 2, 3, 4, 5, 6]

--- Code Execution Successful ---
```

HOME TASKS

1. Write a program that takes an array of Real numbers having size 7 and calculate the sum and mean of all the elements. Also depict the memory management of this task.

CODE:

```
Main.java
1- public class RealNumberArray {
2-     private double[] array;
3-
4-     public RealNumberArray(double[] array) {
5-         this.array = array;
6-     }
7-
8-     public void calculateSumAndMean() {
9-         double sum = 0;
10-         for (double num : array) {
11-             sum += num;
12-         }
13-         double mean = sum / array.length;
14-
15-         System.out.println("Sum: " + sum);
16-         System.out.println("Mean: " + mean);
17-     }
18-
19-     // Memory Management depiction
20-     public void displayMemoryAddresses() {
21-         System.out.println("Memory address of array object: " + System.identityHashCode(array));
22-         for (int i = 0; i < array.length; i++) {
23-             System.out.println("Memory address of element at index " + i + ": " + System.identityHashCode(array[i]));
24-         }
25-     }
26-
27-     public static void main(String[] args) {
28-         double[] numbers = {1.1, 2.2, 3.4, 4.1, 5.8, 6.9, 7.0};
29-         RealNumberArray obj = new RealNumberArray(numbers);
30-
31-         obj.calculateSumAndMean();
32-         obj.displayMemoryAddresses();
33-     }
34- }
35-
36-
```

OUTPUT:

```
Output
java -cp /tmp/BMzoM3IggQ/RealNumberArray
Sum: 30.5
Mean: 4.357142857142857
Memory address of array object: 705265961
Memory address of element at index 0: 1464642111
Memory address of element at index 1: 1190524793
Memory address of element at index 2: 472654579
Memory address of element at index 3: 26117480
Memory address of element at index 4: 870698190
Memory address of element at index 5: 1514322932
Memory address of element at index 6: 654582261

--- Code Execution Successful ---
```

2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key.

CODE:

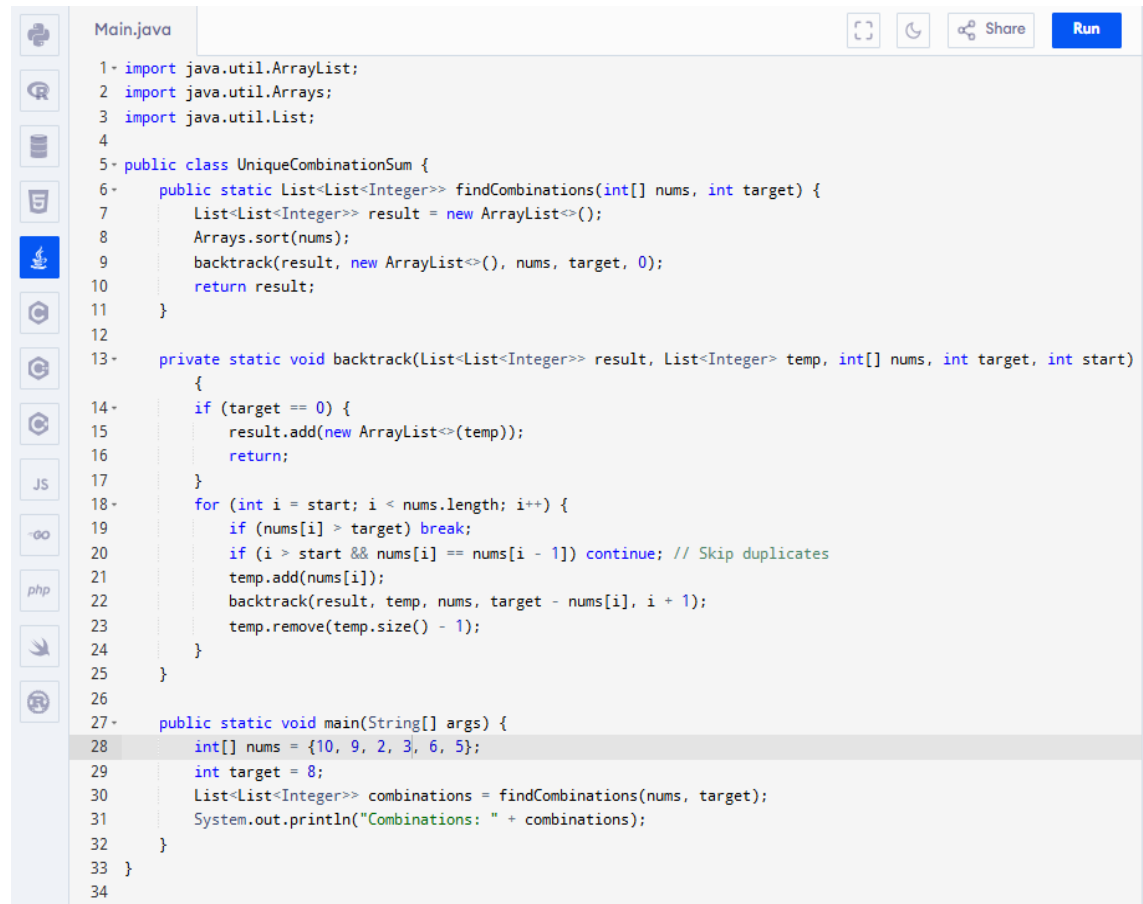
```
Main.java
4- public SplitArray(double[] array) {
5-     this.array = array;
6- }
7-
8- public void splitAtKey(double key) {
9-     int index = -1;
10-
11-     // Find the index of the key in the array
12-     for (int i = 0; i < array.length; i++) {
13-         if (array[i] == key) {
14-             index = i;
15-             break;
16-         }
17-     }
18-
19-     if (index == -1) {
20-         System.out.println("Key not found in array.");
21-         return;
22-     }
23-
24-     // Split the array into two parts
25-     double[] firstPart = new double[index + 1];
26-     double[] secondPart = new double[array.length - index - 1];
27-
28-     System.arraycopy(array, 0, firstPart, 0, index + 1);
29-     System.arraycopy(array, index + 1, secondPart, 0, array.length - index - 1);
30-
31-     System.out.println("First Part: " + java.util.Arrays.toString(firstPart));
32-     System.out.println("Second Part: " + java.util.Arrays.toString(secondPart));
33- }
34-
35- public static void main(String[] args) {
36-     double[] numbers = {1.1, 2.8, 3.6, 4.1, 5.8, 6.2, 7.0};
37-     SplitArray obj = new SplitArray(numbers);
38-
39-     obj.splitAtKey(4.1);
40- }
41- }
42- }
```

OUTPUT:

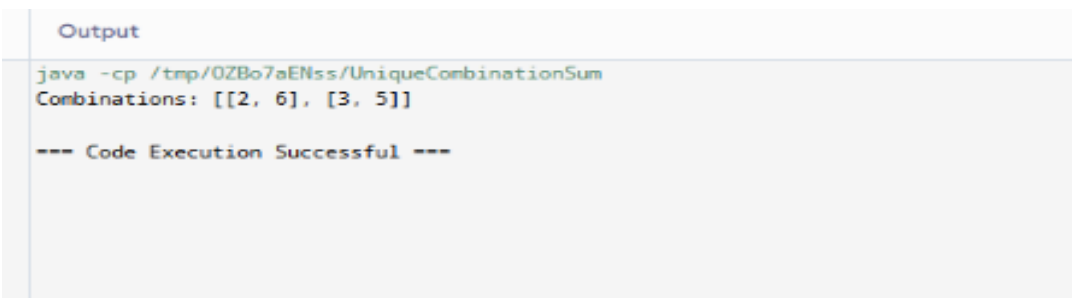
```
Output
~ java -cp /tmp/b5Ub4N8vRJ/SplitArray
First Part: [1.1, 2.8, 3.6, 4.1]
Second Part: [5.8, 6.2, 7.0]

=== Code Execution Successful ===
```

3. Given an array of distinct integers and a target integer, return all unique combinations of numbers that add up to the target. Each number can be used only once in the combination.

CODE:

```
1- import java.util.ArrayList;
2- import java.util.Arrays;
3- import java.util.List;
4-
5- public class UniqueCombinationSum {
6-     public static List<List<Integer>> findCombinations(int[] nums, int target) {
7-         List<List<Integer>> result = new ArrayList<>();
8-         Arrays.sort(nums);
9-         backtrack(result, new ArrayList<>(), nums, target, 0);
10        return result;
11    }
12
13-    private static void backtrack(List<List<Integer>> result, List<Integer> temp, int[] nums, int target, int start)
14    {
15        if (target == 0) {
16            result.add(new ArrayList<>(temp));
17            return;
18        }
19        for (int i = start; i < nums.length; i++) {
20            if (nums[i] > target) break;
21            if (i > start && nums[i] == nums[i - 1]) continue; // Skip duplicates
22            temp.add(nums[i]);
23            backtrack(result, temp, nums, target - nums[i], i + 1);
24            temp.remove(temp.size() - 1);
25        }
26    }
27
28-    public static void main(String[] args) {
29        int[] nums = {10, 9, 2, 3, 6, 5};
30        int target = 8;
31        List<List<Integer>> combinations = findCombinations(nums, target);
32        System.out.println("Combinations: " + combinations);
33    }
34}
```

OUTPUT:

```
Output
java -cp /tmp/OZBo7aENss/UniqueCombinationSum
Combinations: [[2, 6], [3, 5]]

--- Code Execution Successful ---
```


4. You are given an array containing n distinct numbers taken from 0, 1, 2, ..., n. Write a program to find the one number that is missing from the array.

CODE:

```
Main.java
1+ public class MissingNumber {
2+     public static int findMissingNumber(int[] nums) {
3         int n = nums.length;
4         int totalSum = n * (n + 1) / 2;
5         int sum = 0;
6+         for (int num : nums) {
7             sum += num;
8         }
9         return totalSum - sum;
10    }
11
12+    public static void main(String[] args) {
13        int[] nums = {0, 1, 3, 4, 5};
14        System.out.println("Missing Number: " + findMissingNumber(nums));
15    }
16 }
17
18
```

OUTPUT:

```
Output
java -cp /tmp/RxQKpndfAu/MissingNumber
Missing Number: 2

=== Code Execution Successful ===
```

5. You are given an array of integers. Write a program to sort the array such that it follows a zigzag pattern: the first element is less than the second, the second is greater than the third, and so on.

CODE:

```
Main.java
1+ import java.util.Arrays;
2
3+ public class ZigzagPattern {
4+     public static void zigzagSort(int[] nums) {
5         boolean flag = true; // If true, "less than" relation is expected
6
7+         for (int i = 0; i < nums.length - 1; i++) {
8+             if (flag) {
9                 // If current element is greater than the next element, swap them
10+                 if (nums[i] > nums[i + 1]) {
11                     int temp = nums[i];
12                     nums[i] = nums[i + 1];
13                     nums[i + 1] = temp;
14                 }
15+             } else {
16                 // If current element is less than the next element, swap them
17+                 if (nums[i] < nums[i + 1]) {
18                     int temp = nums[i];
19                     nums[i] = nums[i + 1];
20                     nums[i + 1] = temp;
21                 }
22             }
23             flag = !flag; // Flip flag for the next pair
24         }
25     }
26
27+     public static void main(String[] args) {
28         int[] nums = {4, 3, 7, 9, 6, 2, 1};
29         zigzagSort(nums);
30         System.out.println("Zigzag Array: " + Arrays.toString(nums));
31     }
32 }
33
```

OUTPUT:

```
Output
java -cp /tmp/s0810GJoUE/ZigzagPattern
Zigzag Array: [3, 7, 4, 9, 2, 6, 1]

=== Code Execution Successful ===
```