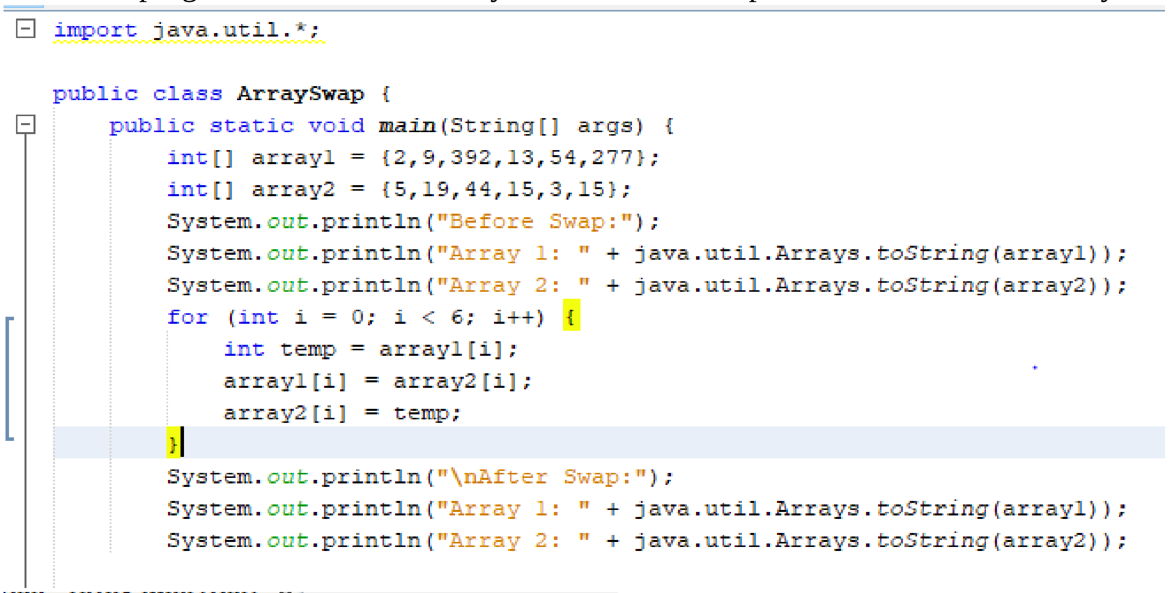
LAB # 04

ARRAYS IN JAVA

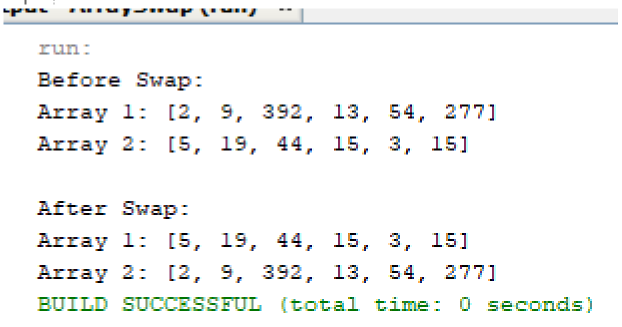
OBJECTIVE: To understand arrays and its memory allocation.

LAB TASKS

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

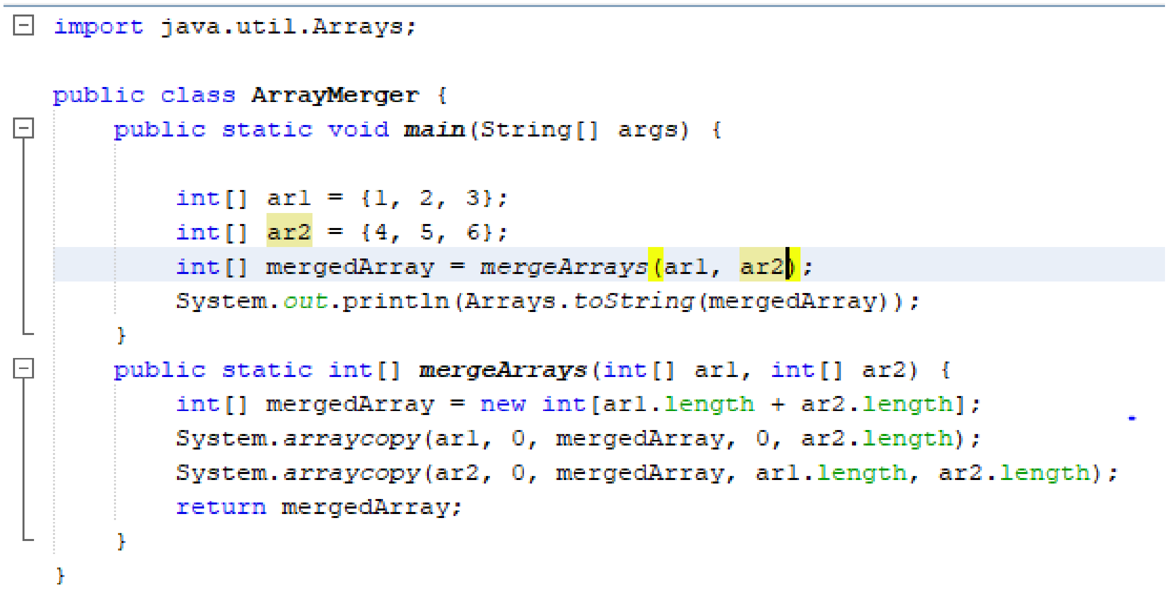


```java
import java.util.*;

public class ArraySwap {
    public static void main(String[] args) {
        int[] array1 = {2,9,392,13,54,277};
        int[] array2 = {5,19,44,15,3,15};
        System.out.println("Before Swap:");
        System.out.println("Array 1: " + java.util.Arrays.toString(array1));
        System.out.println("Array 2: " + java.util.Arrays.toString(array2));
        for (int i = 0; i < 6; i++) {
            int temp = array1[i];
            array1[i] = array2[i];
            array2[i] = temp;
        }
        System.out.println("\nAfter Swap:");
        System.out.println("Array 1: " + java.util.Arrays.toString(array1));
        System.out.println("Array 2: " + java.util.Arrays.toString(array2));
```

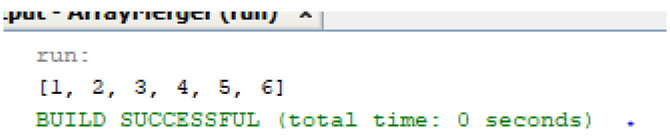


```
run:
Before Swap:
Array 1: [2, 9, 392, 13, 54, 277]
Array 2: [5, 19, 44, 15, 3, 15]

After Swap:
Array 1: [5, 19, 44, 15, 3, 15]
Array 2: [2, 9, 392, 13, 54, 277]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

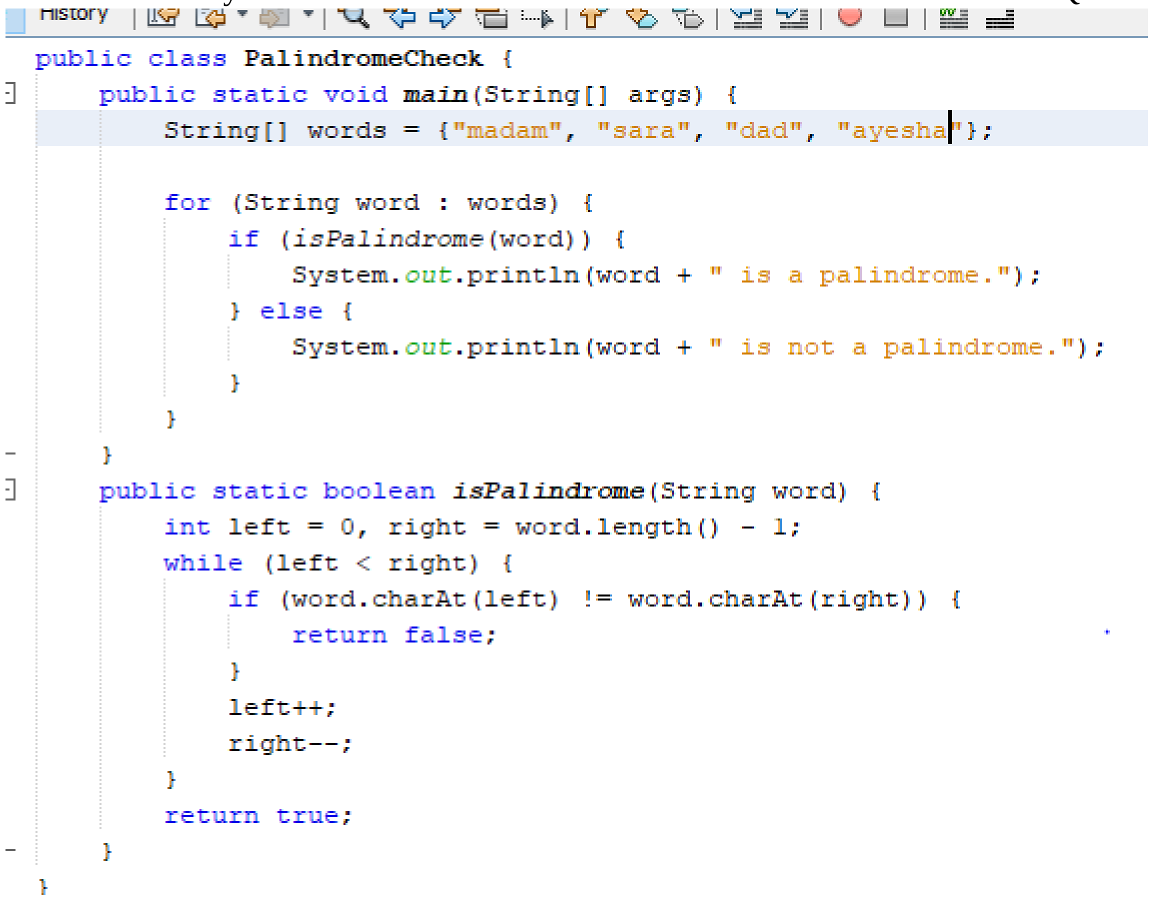


```
import java.util.Arrays;

public class ArrayMerger {
    public static void main(String[] args) {

        int[] ar1 = {1, 2, 3};
        int[] ar2 = {4, 5, 6};
        int[] mergedArray = mergeArrays(ar1, ar2);
        System.out.println(Arrays.toString(mergedArray));
    }
    public static int[] mergeArrays(int[] ar1, int[] ar2) {
        int[] mergedArray = new int[ar1.length + ar2.length];
        System.arraycopy(ar1, 0, mergedArray, 0, ar2.length);
        System.arraycopy(ar2, 0, mergedArray, ar1.length, ar2.length);
        return mergedArray;
    }
}
```

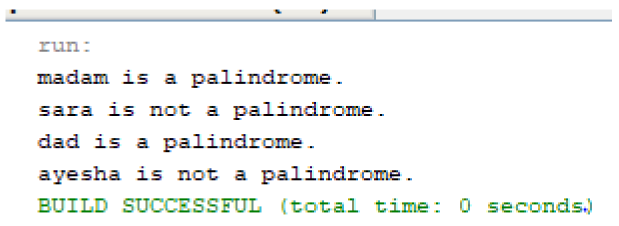


```
run:
[1, 2, 3, 4, 5, 6]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

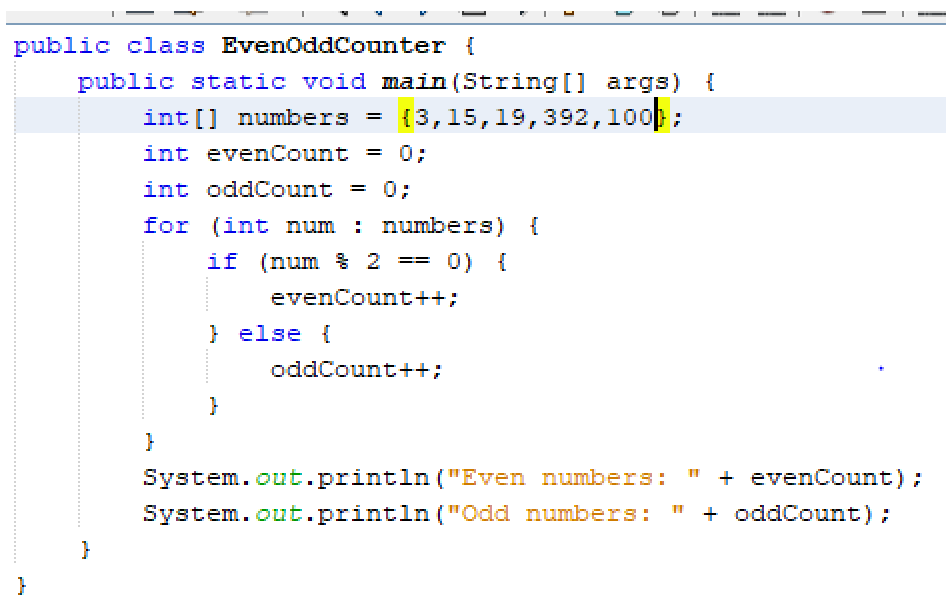


```java
public class PalindromeCheck {
    public static void main(String[] args) {
        String[] words = {"madam", "sara", "dad", "ayesha"};

        for (String word : words) {
            if (isPalindrome(word)) {
                System.out.println(word + " is a palindrome.");
            } else {
                System.out.println(word + " is not a palindrome.");
            }
        }
    }
    public static boolean isPalindrome(String word) {
        int left = 0, right = word.length() - 1;
        while (left < right) {
            if (word.charAt(left) != word.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

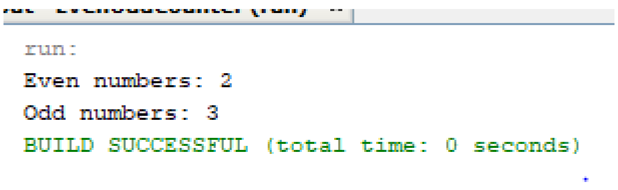


```
run:
madam is a palindrome.
sara is not a palindrome.
dad is a palindrome.
ayesha is not a palindrome.
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

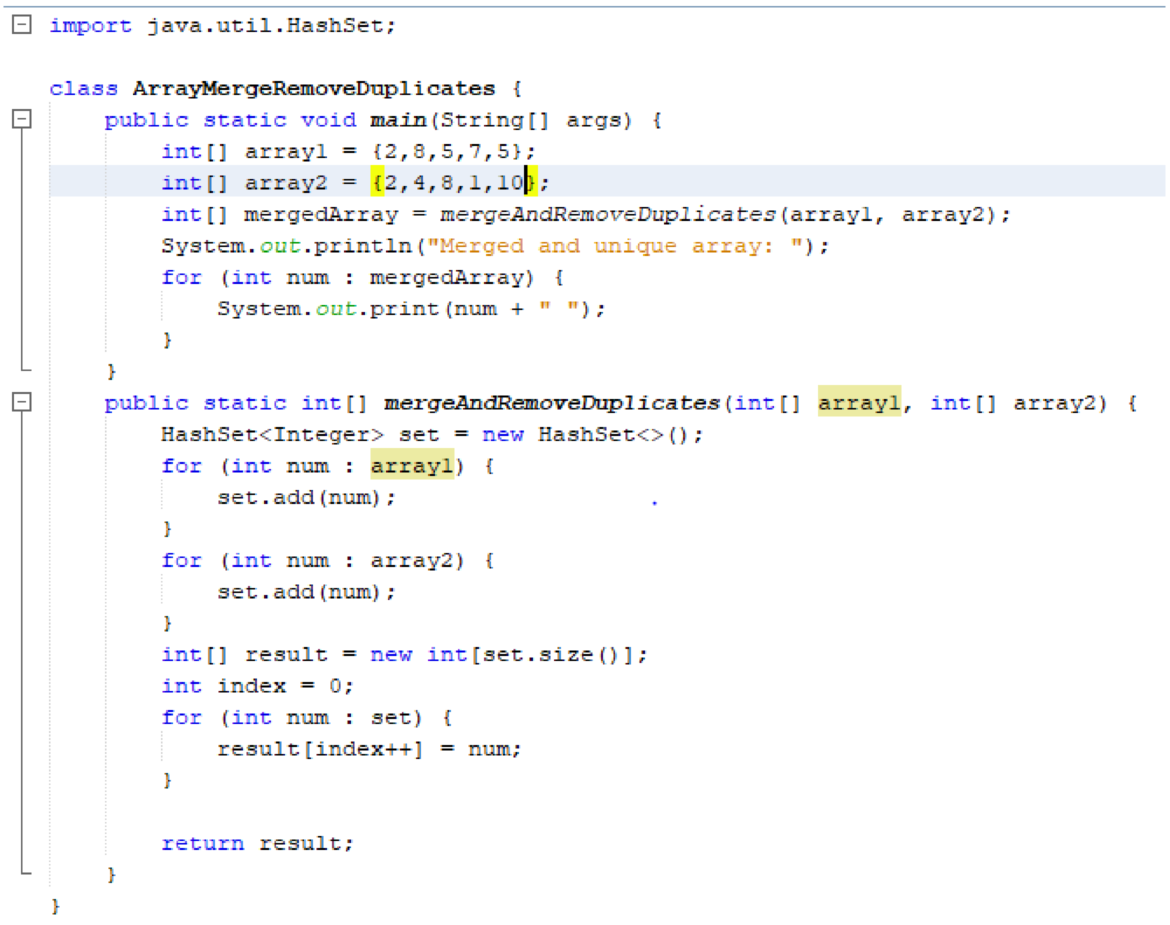


```java
public class EvenOddCounter {
    public static void main(String[] args) {
        int[] numbers = {3,15,19,392,100};
        int evenCount = 0;
        int oddCount = 0;
        for (int num : numbers) {
            if (num % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }
        }
        System.out.println("Even numbers: " + evenCount);
        System.out.println("Odd numbers: " + oddCount);
    }
}
```

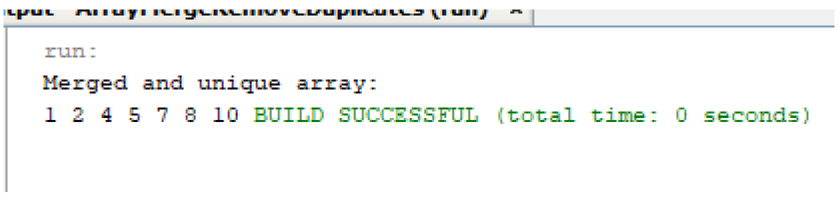


```
run:
Even numbers: 2
Odd numbers: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

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



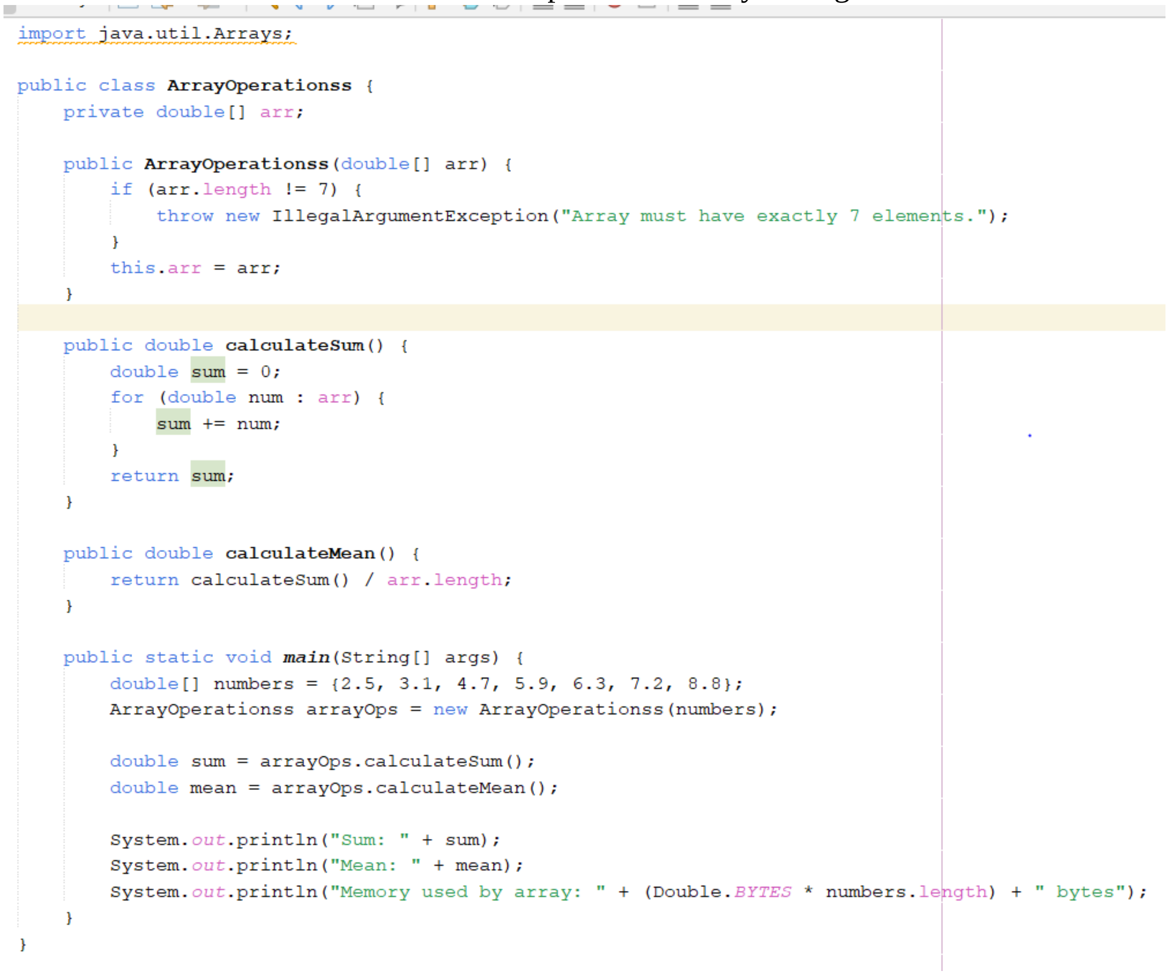
```java
import java.util.HashSet;

class ArrayMergeRemoveDuplicates {
    public static void main(String[] args) {
        int[] array1 = {2,8,5,7,5};
        int[] array2 = {2,4,8,1,10};
        int[] mergedArray = mergeAndRemoveDuplicates(array1, array2);
        System.out.println("Merged and unique array: ");
        for (int num : mergedArray) {
            System.out.print(num + " ");
        }
    }
    public static int[] mergeAndRemoveDuplicates(int[] array1, int[] array2) {
        HashSet<Integer> set = new HashSet<>();
        for (int num : array1) {
            set.add(num);
        }
        for (int num : array2) {
            set.add(num);
        }
        int[] result = new int[set.size()];
        int index = 0;
        for (int num : set) {
            result[index++] = num;
        }

        return result;
    }
}
```



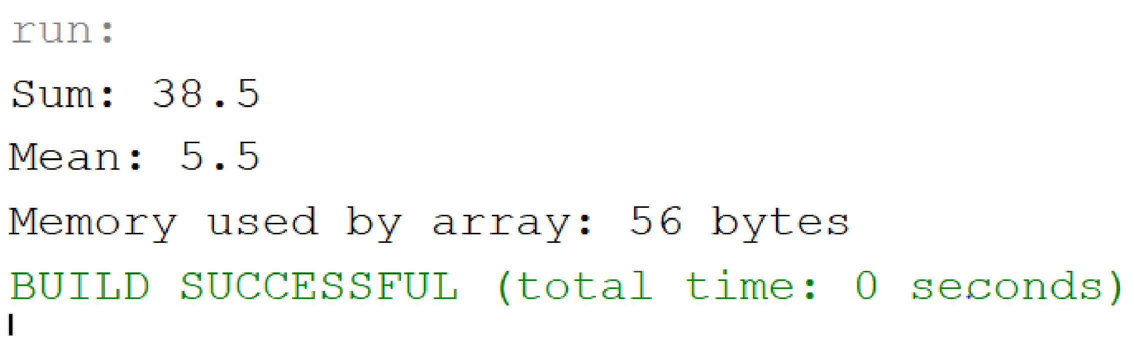
```
run:
Merged and unique array:
1 2 4 5 7 8 10 BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

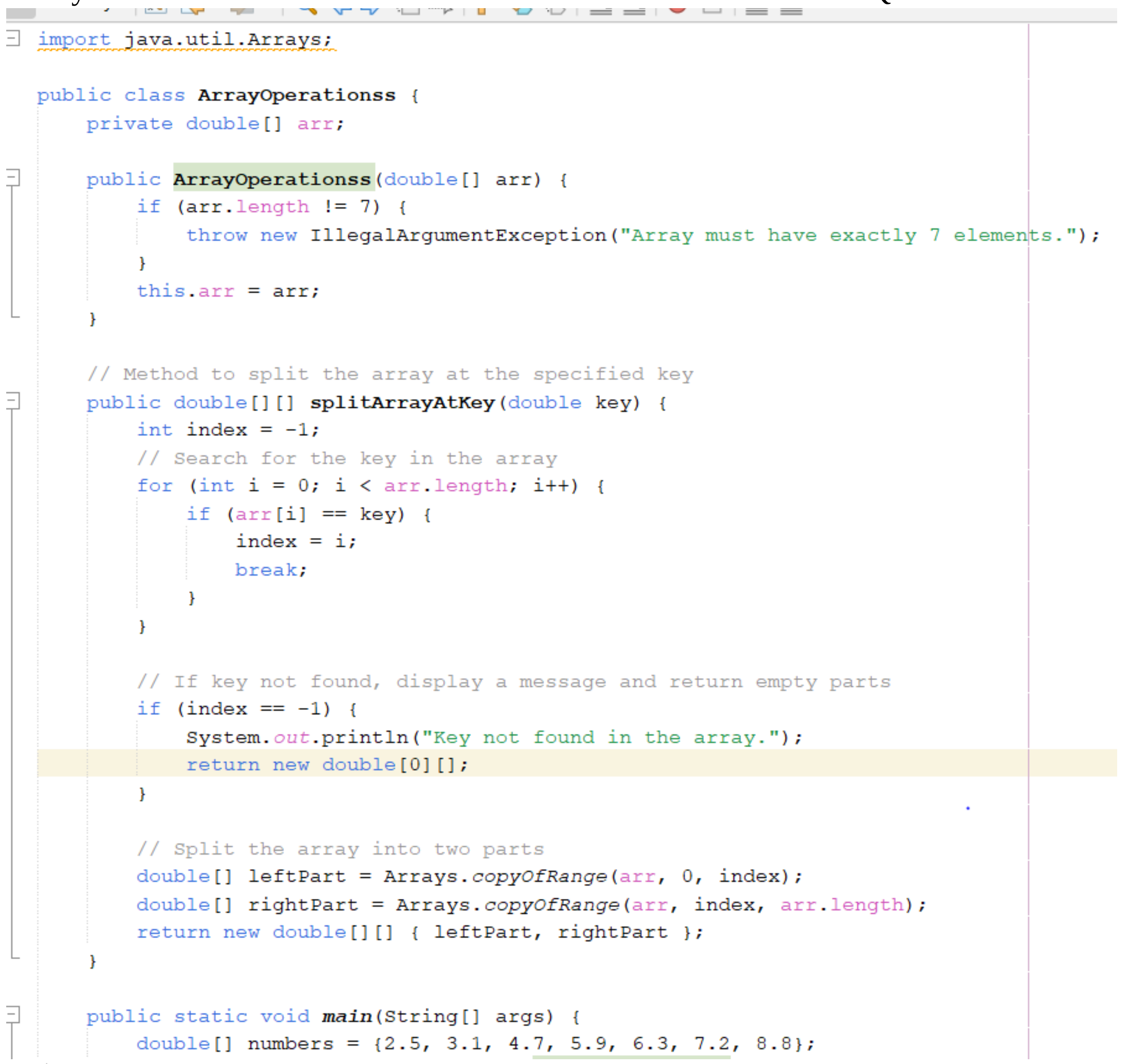


```
import java.util.Arrays;

public class ArrayOperationss {
    private double[] arr;

    public ArrayOperationss(double[] arr) {
        if (arr.length != 7) {
            throw new IllegalArgumentException("Array must have exactly 7 elements.");
        }
        this.arr = arr;
    }

    public double calculateSum() {
        double sum = 0;
        for (double num : arr) {
            sum += num;
        }
        return sum;
    }

    public double calculateMean() {
        return calculateSum() / arr.length;
    }

    public static void main(String[] args) {
        double[] numbers = {2.5, 3.1, 4.7, 5.9, 6.3, 7.2, 8.8};
        ArrayOperationss arrayOps = new ArrayOperationss(numbers);

        double sum = arrayOps.calculateSum();
        double mean = arrayOps.calculateMean();

        System.out.println("Sum: " + sum);
        System.out.println("Mean: " + mean);
        System.out.println("Memory used by array: " + (Double.BYTES * numbers.length) + " bytes");
    }
}
```

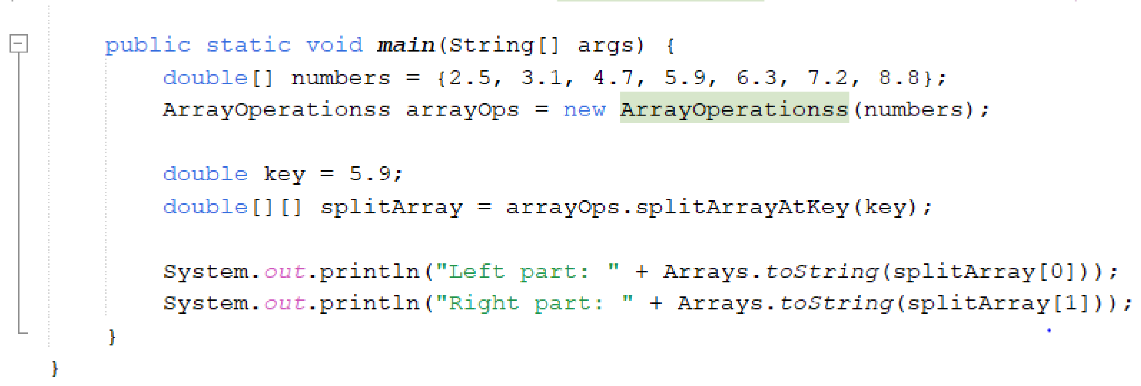


```
run:
Sum: 38.5
Mean: 5.5
Memory used by array: 56 bytes
BUILD SUCCESSFUL (total time: 0 seconds)
```

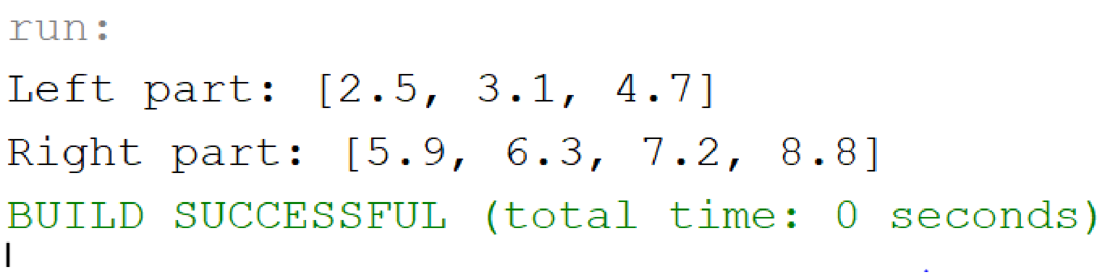
2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key.



```java
import java.util.Arrays;

public class ArrayOperationss {
    private double[] arr;

    public ArrayOperationss(double[] arr) {
        if (arr.length != 7) {
            throw new IllegalArgumentException("Array must have exactly 7 elements.");
        }
        this.arr = arr;
    }

    // Method to split the array at the specified key
    public double[][] splitArrayAtKey(double key) {
        int index = -1;
        // Search for the key in the array
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                index = i;
                break;
            }
        }

        // If key not found, display a message and return empty parts
        if (index == -1) {
            System.out.println("Key not found in the array.");
            return new double[0][];
        }

        // Split the array into two parts
        double[] leftPart = Arrays.copyOfRange(arr, 0, index);
        double[] rightPart = Arrays.copyOfRange(arr, index, arr.length);
        return new double[][] { leftPart, rightPart };
    }

    public static void main(String[] args) {
        double[] numbers = {2.5, 3.1, 4.7, 5.9, 6.3, 7.2, 8.8};
```

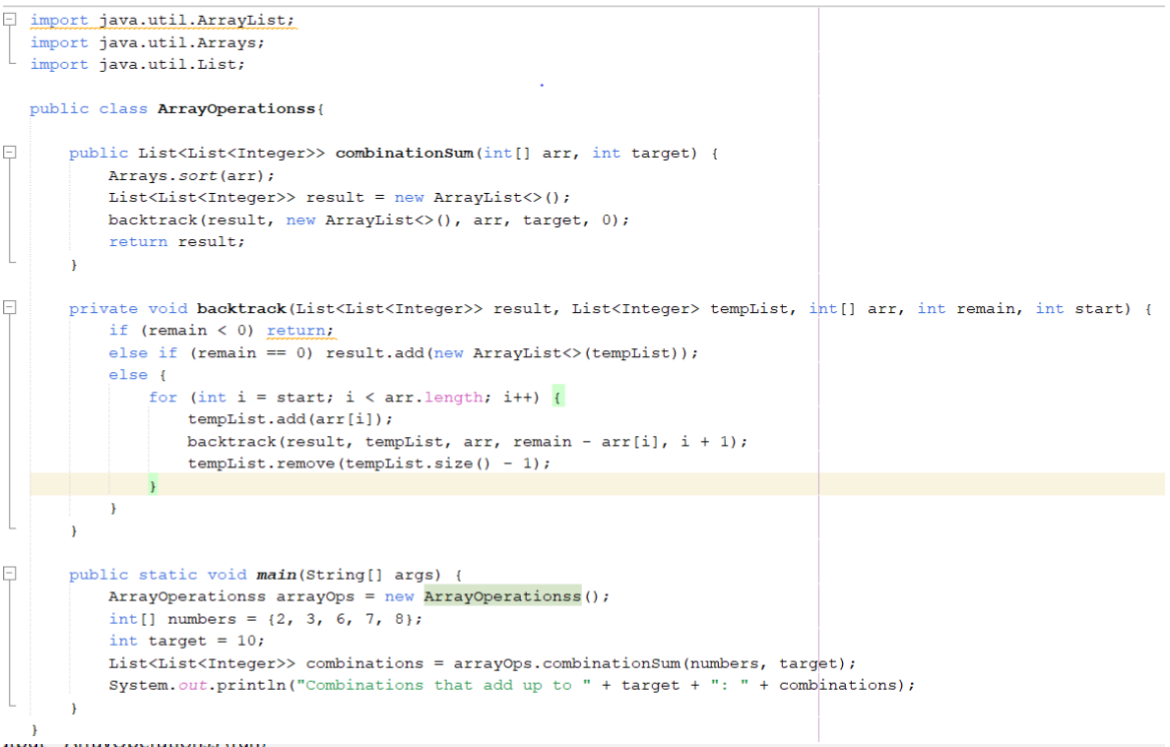


```java
    public static void main(String[] args) {
        double[] numbers = {2.5, 3.1, 4.7, 5.9, 6.3, 7.2, 8.8};
        ArrayOperationss arrayOps = new ArrayOperationss(numbers);

        double key = 5.9;
        double[][] splitArray = arrayOps.splitArrayAtKey(key);

        System.out.println("Left part: " + Arrays.toString(splitArray[0]));
        System.out.println("Right part: " + Arrays.toString(splitArray[1]));
    }
}
```

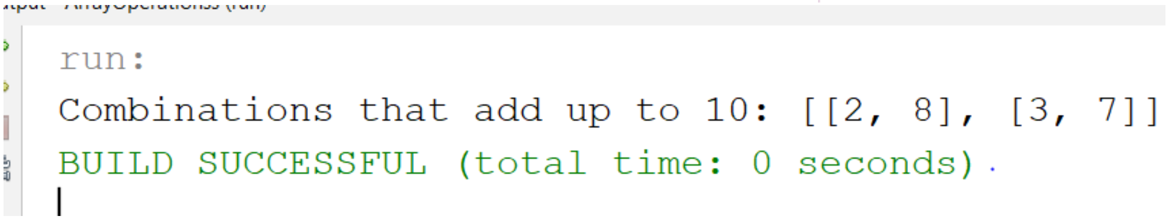


```
run:
Left part: [2.5, 3.1, 4.7]
Right part: [5.9, 6.3, 7.2, 8.8]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

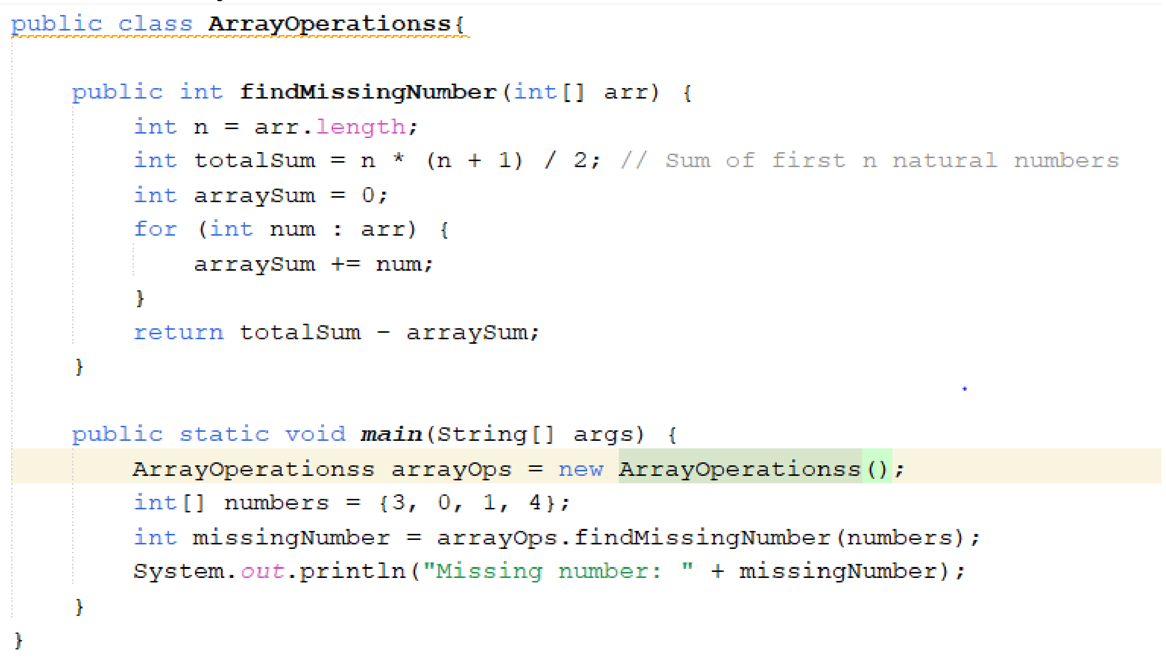


```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ArrayOperationss{

    public List<List<Integer>> combinationSum(int[] arr, int target) {
        Arrays.sort(arr);
        List<List<Integer>> result = new ArrayList<>();
        backtrack(result, new ArrayList<>(), arr, target, 0);
        return result;
    }

    private void backtrack(List<List<Integer>> result, List<Integer> tempList, int[] arr, int remain, int start) {
        if (remain < 0) return;
        else if (remain == 0) result.add(new ArrayList<>(tempList));
        else {
            for (int i = start; i < arr.length; i++) {
                tempList.add(arr[i]);
                backtrack(result, tempList, arr, remain - arr[i], i + 1);
                tempList.remove(tempList.size() - 1);
            }
        }
    }

    public static void main(String[] args) {
        ArrayOperationss arrayOps = new ArrayOperationss();
        int[] numbers = {2, 3, 6, 7, 8};
        int target = 10;
        List<List<Integer>> combinations = arrayOps.combinationSum(numbers, target);
        System.out.println("Combinations that add up to " + target + ": " + combinations);
    }
}
```

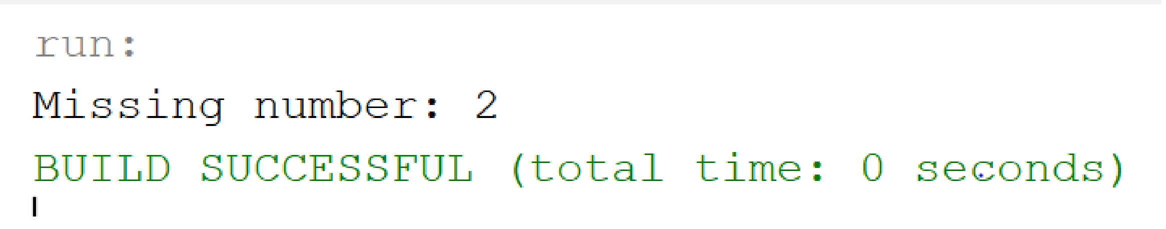


```
run:
Combinations that add up to 10: [[2, 8], [3, 7]]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

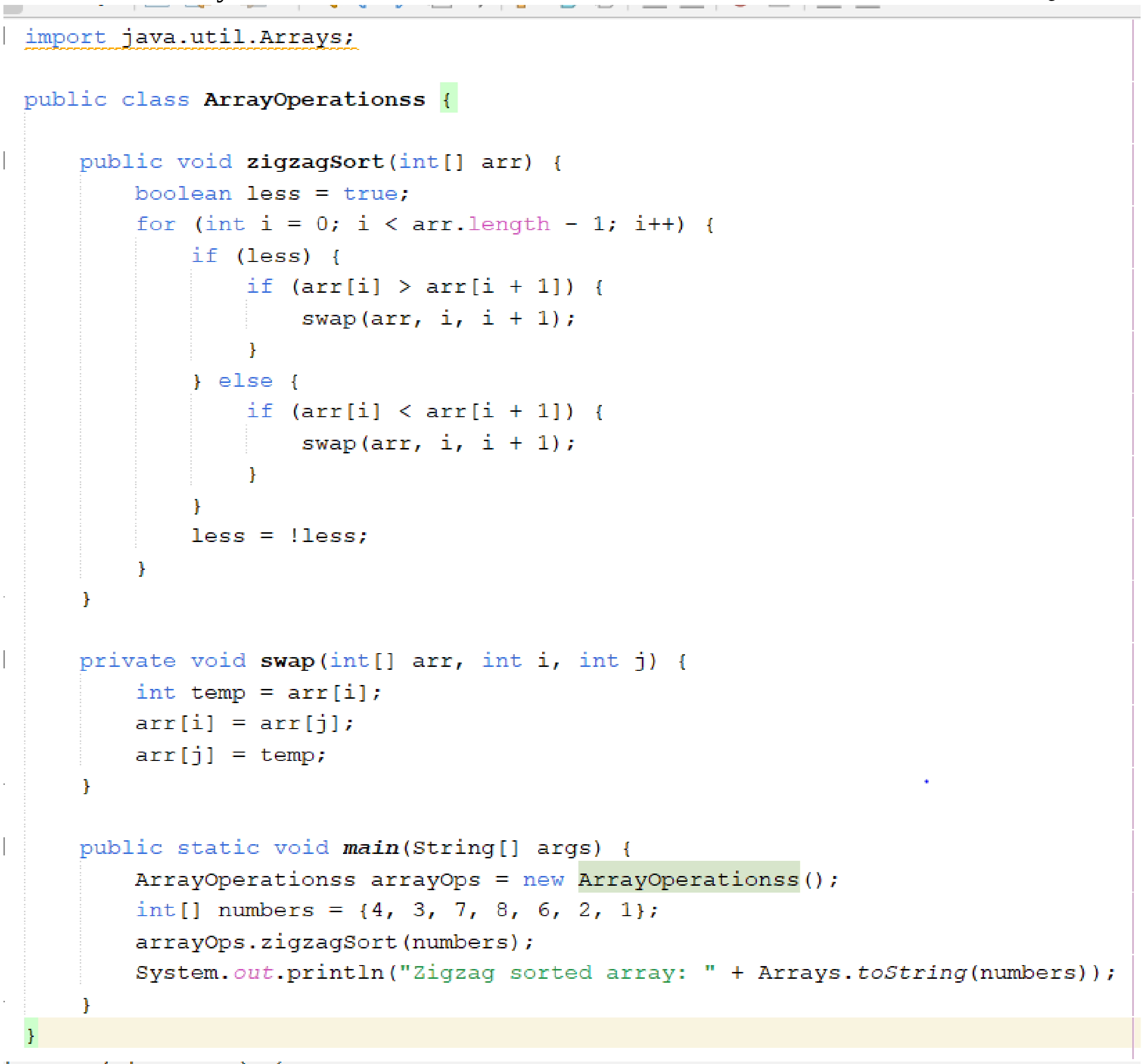


```java
public class ArrayOperationss{

    public int findMissingNumber(int[] arr) {
        int n = arr.length;
        int totalSum = n * (n + 1) / 2; // Sum of first n natural numbers
        int arraySum = 0;
        for (int num : arr) {
            arraySum += num;
        }
        return totalSum - arraySum;
    }

    public static void main(String[] args) {
        ArrayOperationss arrayOps = new ArrayOperationss();
        int[] numbers = {3, 0, 1, 4};
        int missingNumber = arrayOps.findMissingNumber(numbers);
        System.out.println("Missing number: " + missingNumber);
    }
}
```

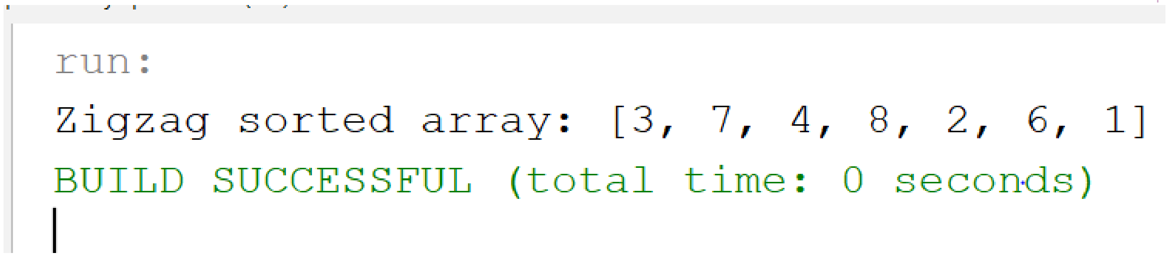


```
run:
Missing number: 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.



```java
import java.util.Arrays;

public class ArrayOperationss {

    public void zigzagSort(int[] arr) {
        boolean less = true;
        for (int i = 0; i < arr.length - 1; i++) {
            if (less) {
                if (arr[i] > arr[i + 1]) {
                    swap(arr, i, i + 1);
                }
            } else {
                if (arr[i] < arr[i + 1]) {
                    swap(arr, i, i + 1);
                }
            }
            less = !less;
        }
    }

    private void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void main(String[] args) {
        ArrayOperationss arrayOps = new ArrayOperationss();
        int[] numbers = {4, 3, 7, 8, 6, 2, 1};
        arrayOps.zigzagSort(numbers);
        System.out.println("Zigzag sorted array: " + Arrays.toString(numbers));
    }
}
```



```
run:
Zigzag sorted array: [3, 7, 4, 8, 2, 6, 1]
BUILD SUCCESSFUL (total time: 0 seconds)
```