LAB # 04

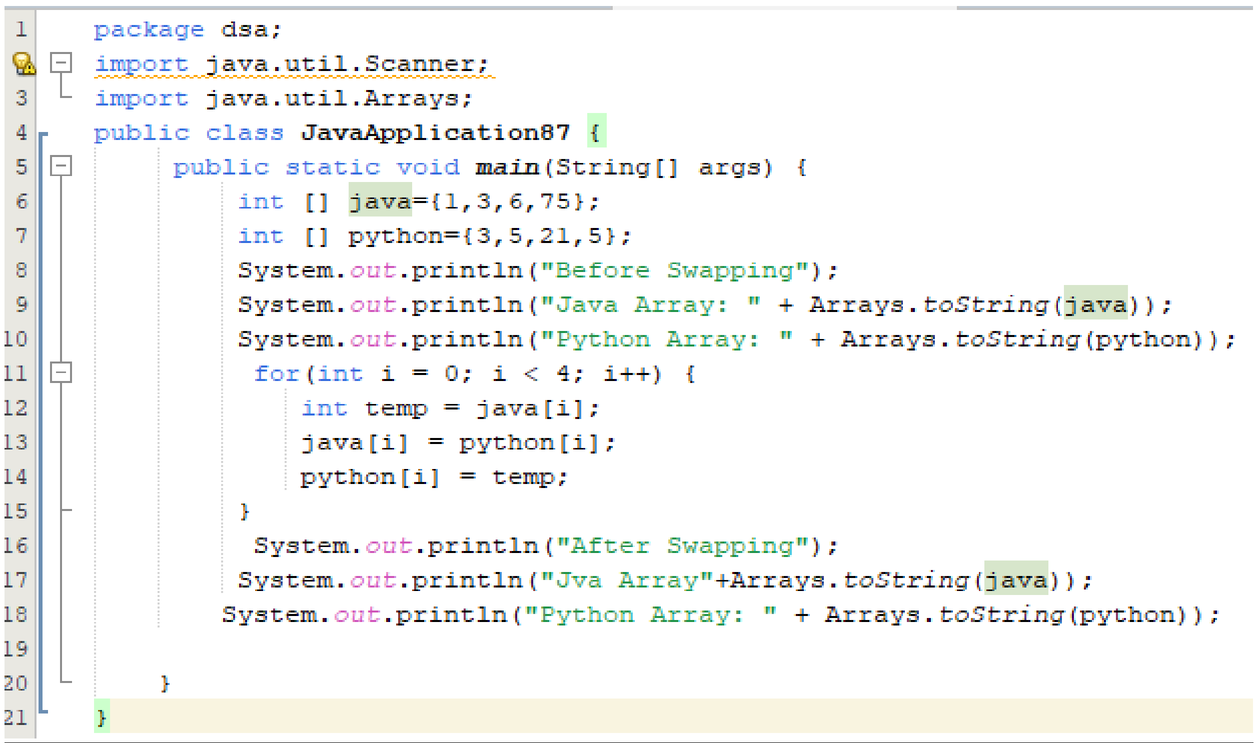
ARRAYS IN JAVA

OBJECTIVE: To understand arrays and its memory allocation.

LAB TASKS

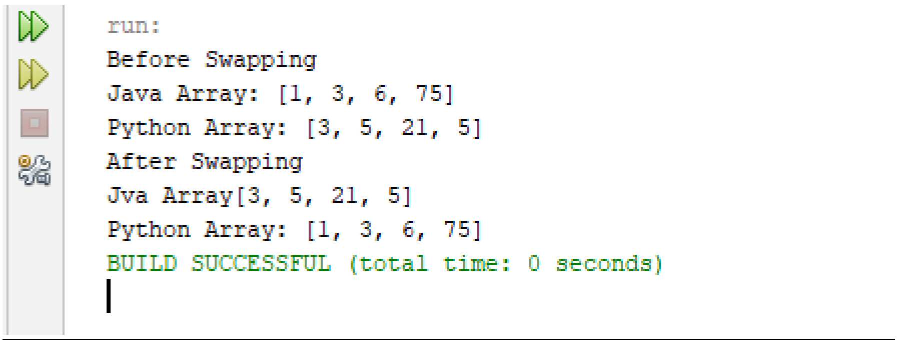
1. Write a program that takes two arrays of size 4 and swap the elements of those arrays

CODE:



```
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
    public static void main(String[] args) {
        int [] java={1,3,6,75};
        int [] python={3,5,21,5};
        System.out.println("Before Swapping");
        System.out.println("Java Array: " + Arrays.toString(java));
        System.out.println("Python Array: " + Arrays.toString(python));
         for(int i = 0; i < 4; i++) {
            int temp = java[i];
            java[i] = python[i];
            python[i] = temp;
        }
         System.out.println("After Swapping");
        System.out.println("Jva Array"+Arrays.toString(java));
       System.out.println("Python Array: " + Arrays.toString(python));

    }
}
```

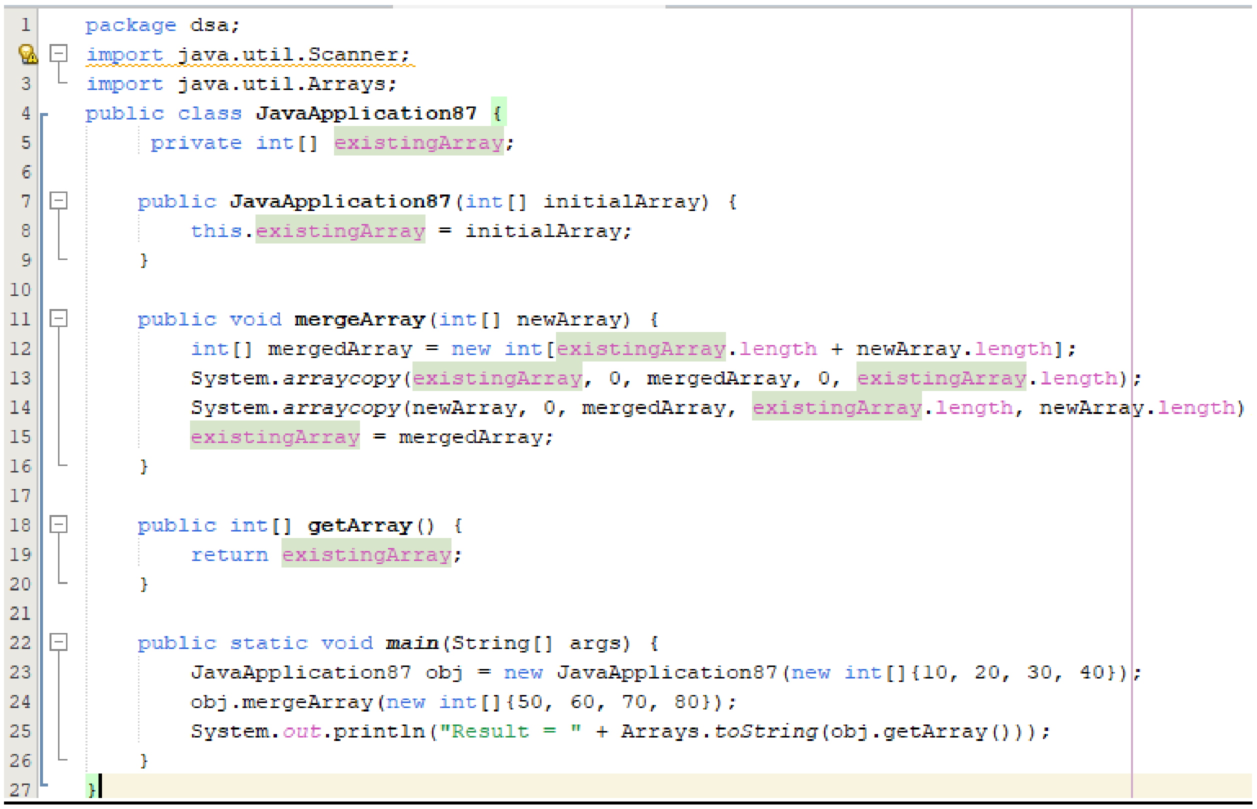
OUTPUT:



```
run:
Before Swapping
Java Array: [1, 3, 6, 75]
Python Array: [3, 5, 21, 5]
After Swapping
Jva Array[3, 5, 21, 5]
Python Array: [1, 3, 6, 75]
BUILD SUCCESSFUL (total time: 0 seconds)
```

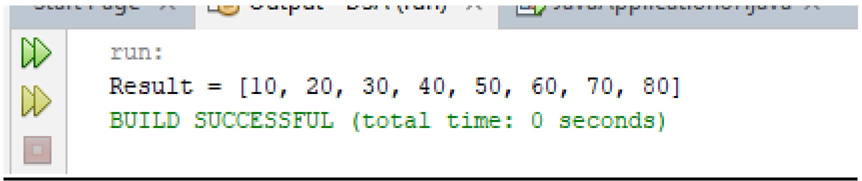
2. Add a method in the class that takes array and merge it with the existing one.

CODE



```
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
    private int[] existingArray;

    public JavaApplication87(int[] initialArray) {
        this.existingArray = initialArray;
    }

    public void mergeArray(int[] newArray) {
        int[] mergedArray = new int[existingArray.length + newArray.length];
        System.arraycopy(existingArray, 0, mergedArray, 0, existingArray.length);
        System.arraycopy(newArray, 0, mergedArray, existingArray.length, newArray.length)
        existingArray = mergedArray;
    }

    public int[] getArray() {
        return existingArray;
    }

    public static void main(String[] args) {
        JavaApplication87 obj = new JavaApplication87(new int[]{10, 20, 30, 40});
        obj.mergeArray(new int[]{50, 60, 70, 80});
        System.out.println("Result = " + Arrays.toString(obj.getArray()));
    }
}
```

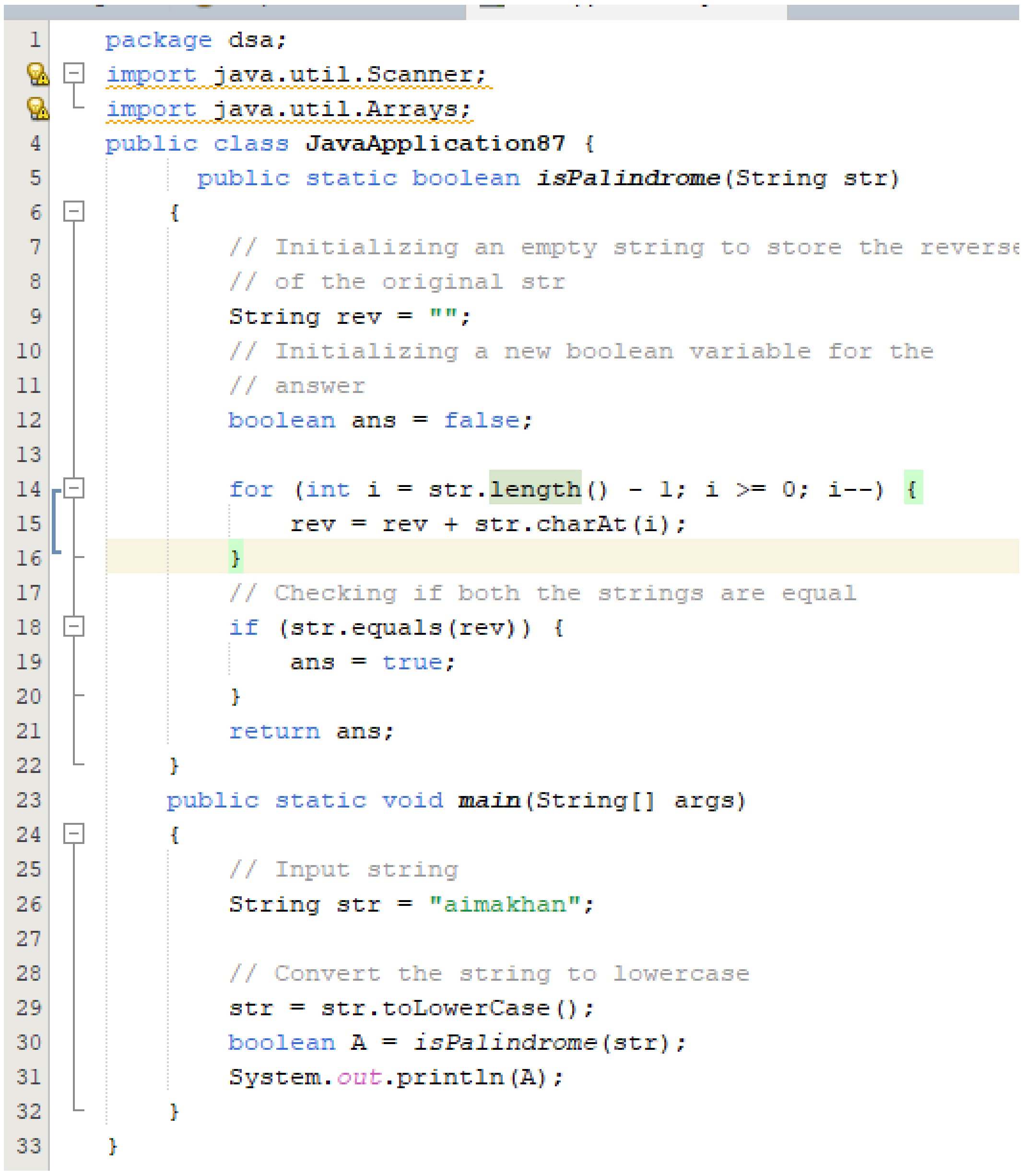
OUTPUT:



```
run:
Result = [10, 20, 30, 40, 50, 60, 70, 80]
BUILD SUCCESSFUL (total time: 0 seconds)
```

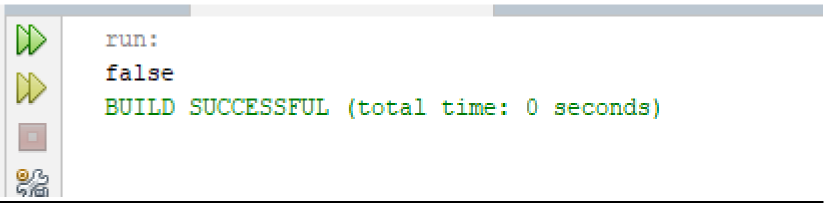
3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not

CODE



```java
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
        public static boolean isPalindrome(String str)
    {
        // Initializing an empty string to store the reverse
        // of the original str
        String rev = "";
        // Initializing a new boolean variable for the
        // answer
        boolean ans = false;

        for (int i = str.length() - 1; i >= 0; i--) {
            rev = rev + str.charAt(i);
        }
        // Checking if both the strings are equal
        if (str.equals(rev)) {
            ans = true;
        }
        return ans;
    }
    public static void main(String[] args)
    {
        // Input string
        String str = "aimakhan";

        // Convert the string to lowercase
        str = str.toLowerCase();
        boolean A = isPalindrome(str);
        System.out.println(A);
    }
}
```

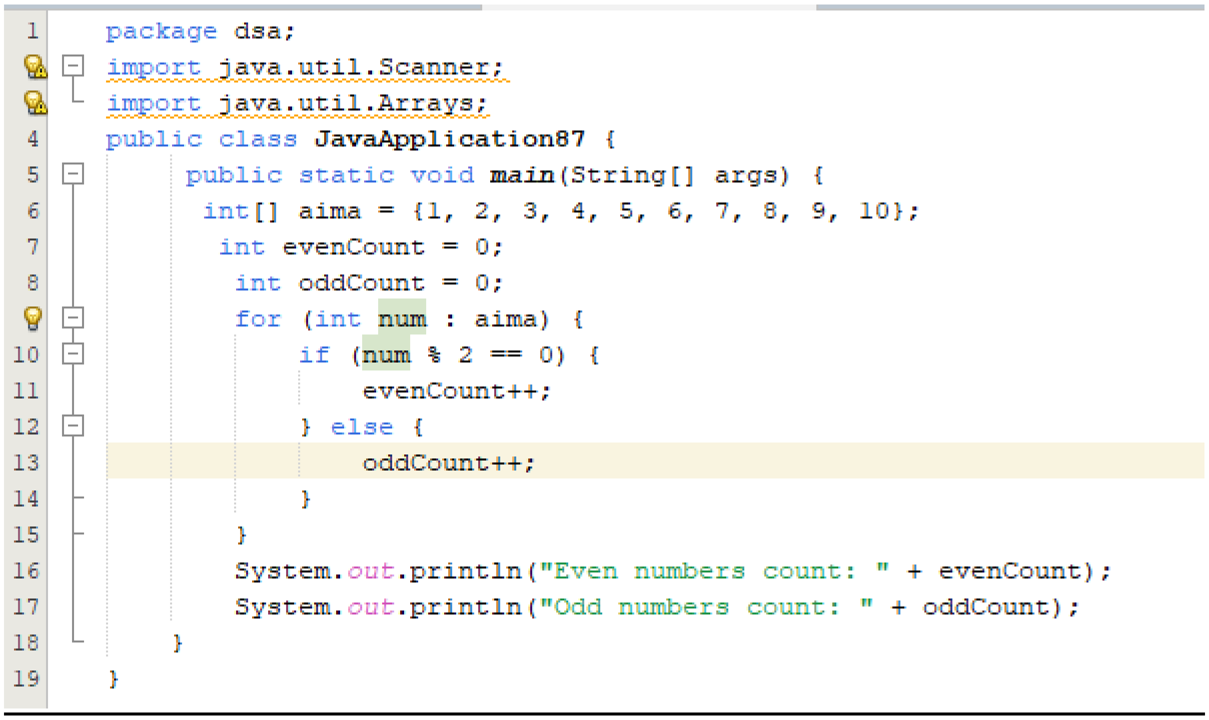
OUTPUT:



```
run:
false
BUILD SUCCESSFUL (total time: 0 seconds)
```

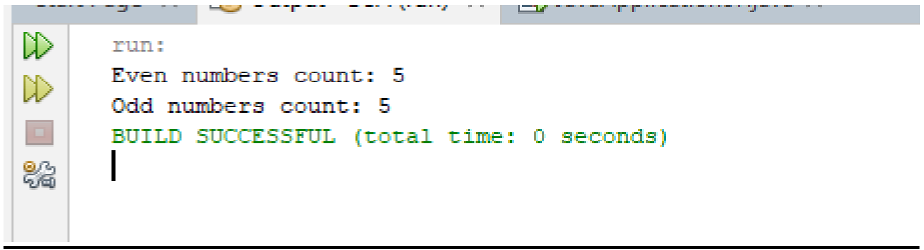
4. Given an array of integers, count how many numbers are even and how many are odd.

CODE



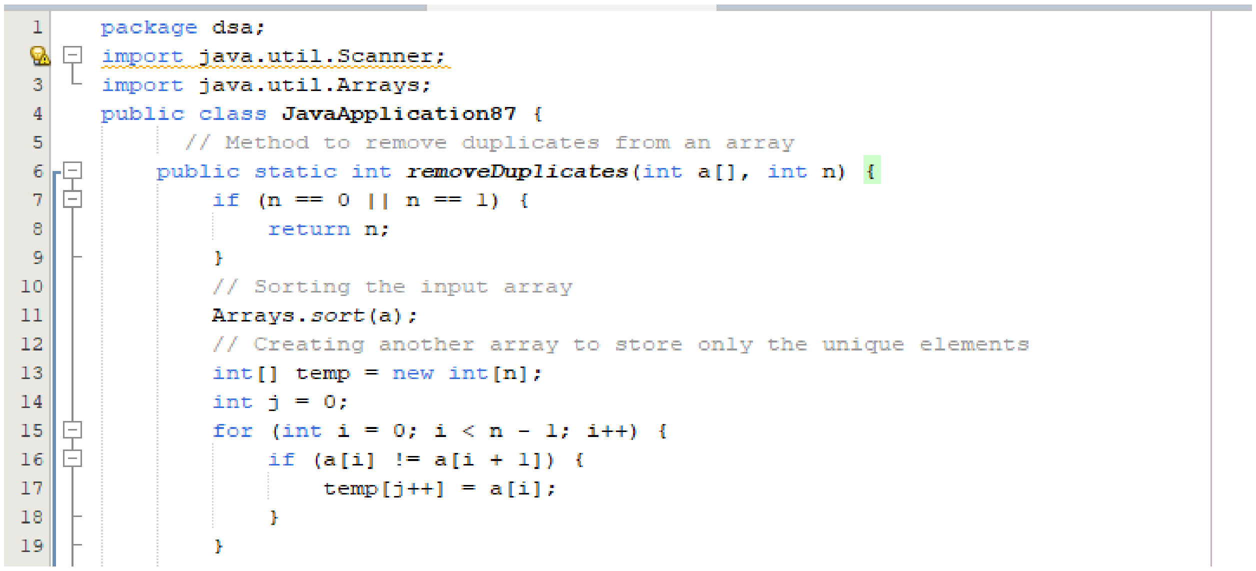
```java
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
        public static void main(String[] args) {
         int[] aima = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
          int evenCount = 0;
           int oddCount = 0;
           for (int num : aima) {
               if (num % 2 == 0) {
                   evenCount++;
               } else {
                   oddCount++;
               }
           }
           System.out.println("Even numbers count: " + evenCount);
           System.out.println("Odd numbers count: " + oddCount);
       }
}
```

OUTPUT:

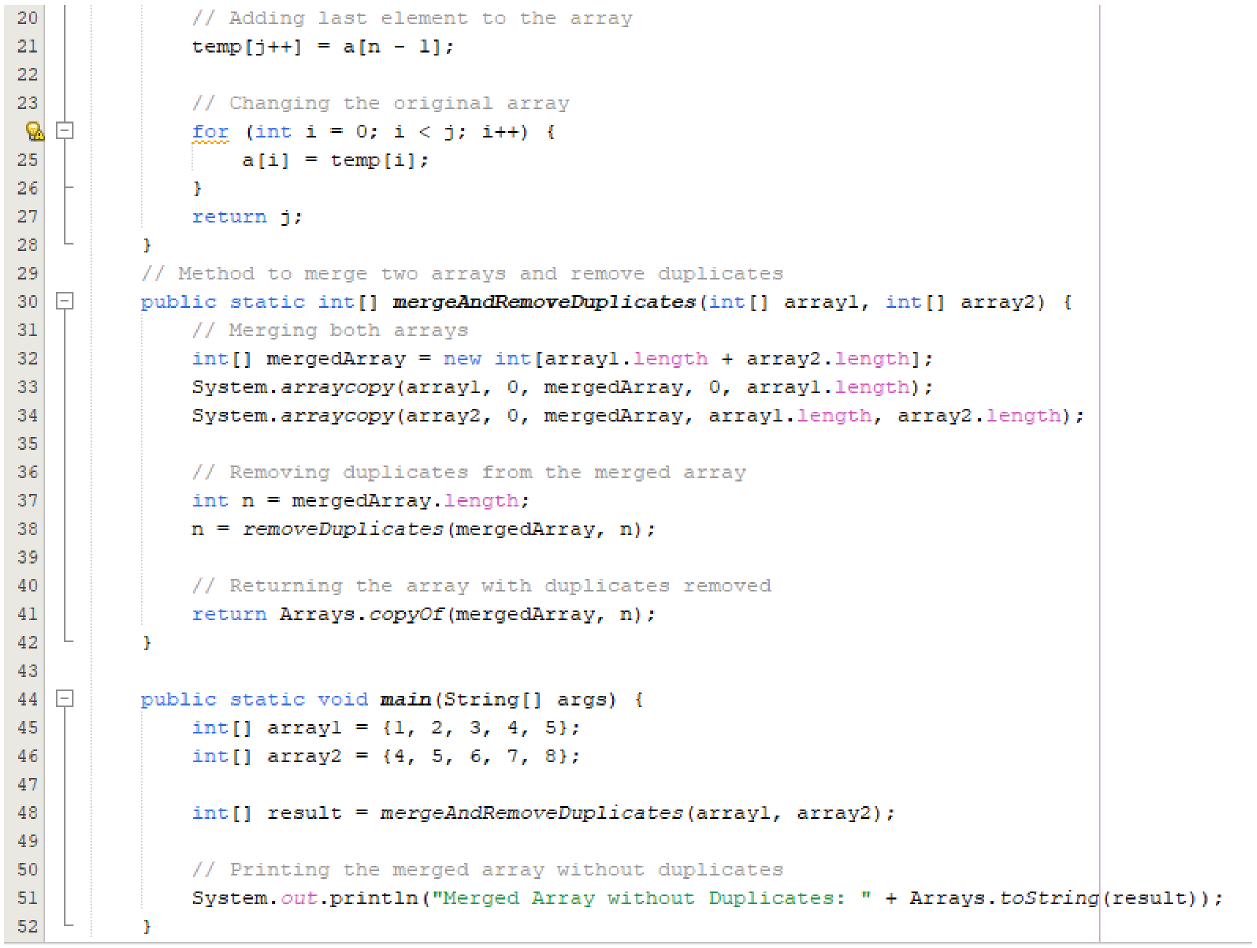


```
run:
Even numbers count: 5
Odd numbers count: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

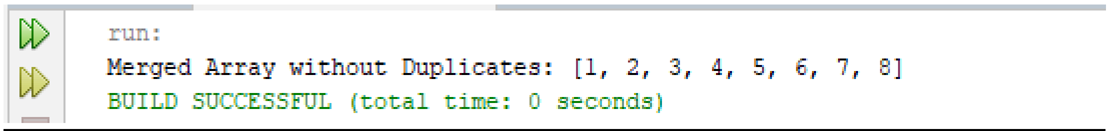


```java
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
       // Method to remove duplicates from an array
    public static int removeDuplicates(int a[], int n) {
        if (n == 0 || n == 1) {
            return n;
        }
        // Sorting the input array
        Arrays.sort(a);
        // Creating another array to store only the unique elements
        int[] temp = new int[n];
        int j = 0;
        for (int i = 0; i < n - 1; i++) {
            if (a[i] != a[i + 1]) {
                temp[j++] = a[i];
            }
        }
```



```java
        // Adding last element to the array
        temp[j++] = a[n - 1];

        // Changing the original array
        for (int i = 0; i < j; i++) {
            a[i] = temp[i];
        }
        return j;
    }
    // Method to merge two arrays and remove duplicates
    public static int[] mergeAndRemoveDuplicates(int[] array1, int[] array2) {
        // Merging both arrays
        int[] mergedArray = new int[array1.length + array2.length];
        System.arraycopy(array1, 0, mergedArray, 0, array1.length);
        System.arraycopy(array2, 0, mergedArray, array1.length, array2.length);

        // Removing duplicates from the merged array
        int n = mergedArray.length;
        n = removeDuplicates(mergedArray, n);

        // Returning the array with duplicates removed
        return Arrays.copyOf(mergedArray, n);
    }

    public static void main(String[] args) {
        int[] array1 = {1, 2, 3, 4, 5};
        int[] array2 = {4, 5, 6, 7, 8};

        int[] result = mergeAndRemoveDuplicates(array1, array2);

        // Printing the merged array without duplicates
        System.out.println("Merged Array without Duplicates: " + Arrays.toString(result));
    }
```

OUTPUT:

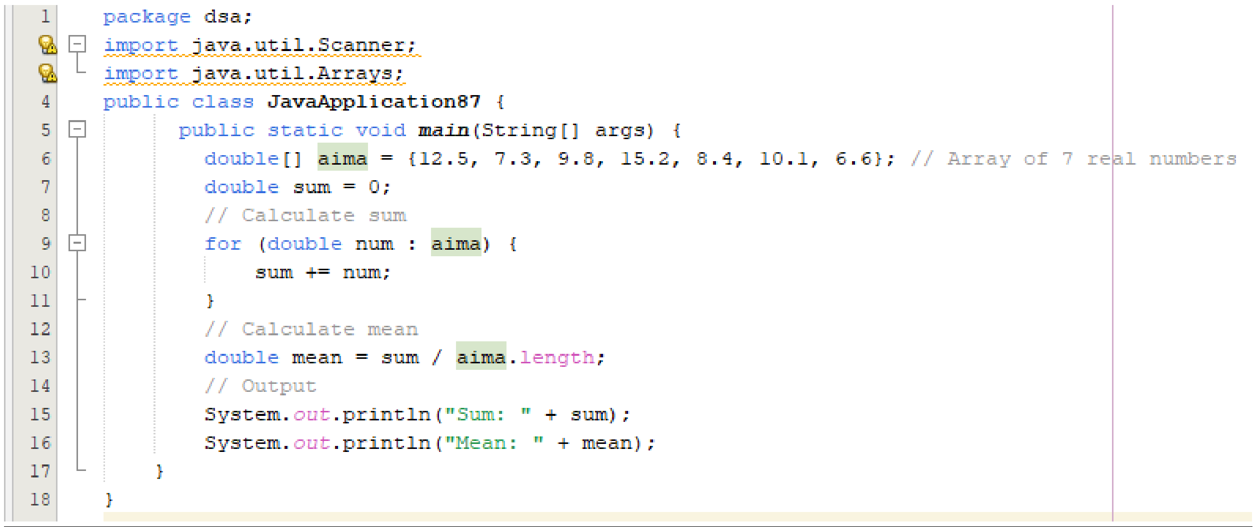


```
run:
Merged Array without Duplicates: [1, 2, 3, 4, 5, 6, 7, 8]
BUILD SUCCESSFUL (total time: 0 seconds)
```

HOME TASKS

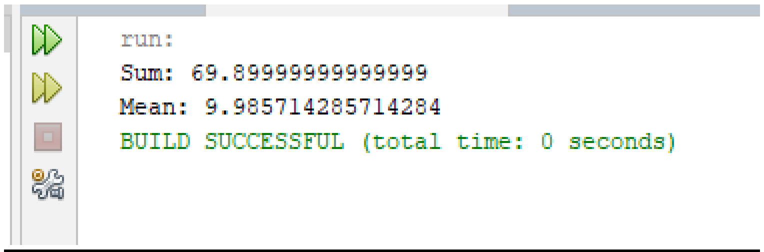
1. Write a program that takes an array of Real numbers having size 7 and calculate the sum and mean of all the elements. Also depict the memory management of this task.

   CODE:



```java
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
    public static void main(String[] args) {
      double[] aima = {12.5, 7.3, 9.8, 15.2, 8.4, 10.1, 6.6}; // Array of 7 real numbers
      double sum = 0;
      // Calculate sum
      for (double num : aima) {
          sum += num;
      }
      // Calculate mean
      double mean = sum / aima.length;
      // Output
      System.out.println("Sum: " + sum);
      System.out.println("Mean: " + mean);
    }
}
```

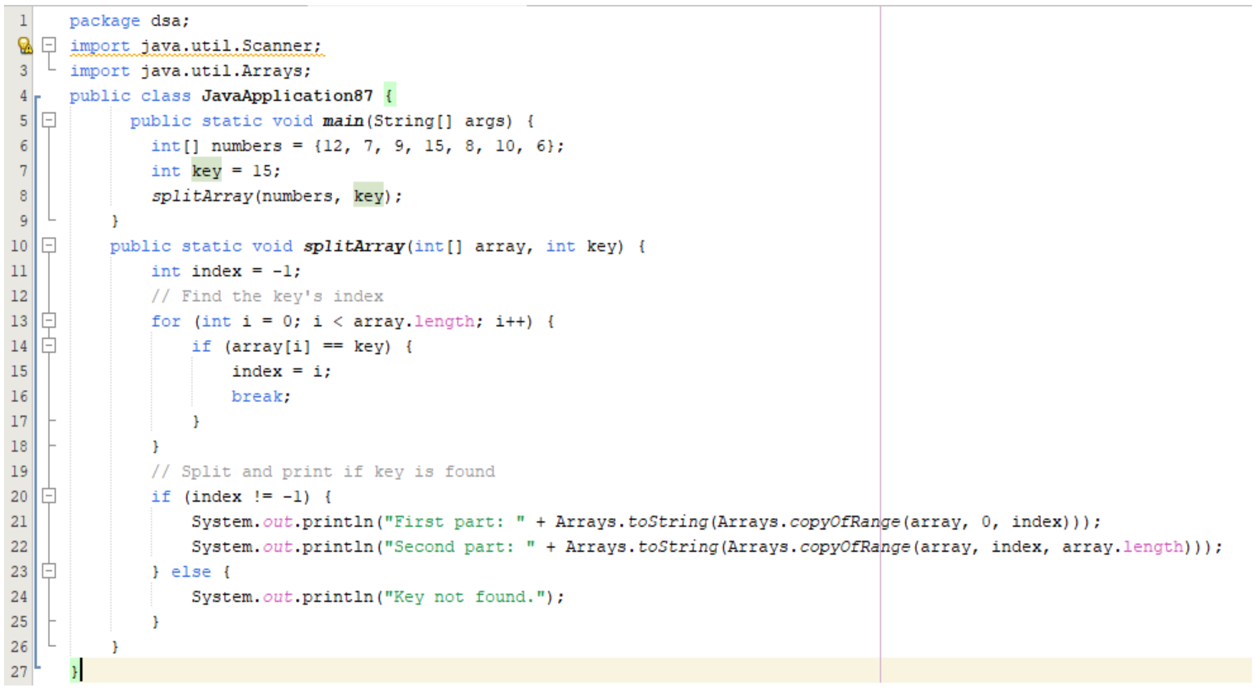
OUTPUT:



```
run:
Sum: 69.89999999999999
Mean: 9.985714285714284
BUILD SUCCESSFUL (total time: 0 seconds)
```

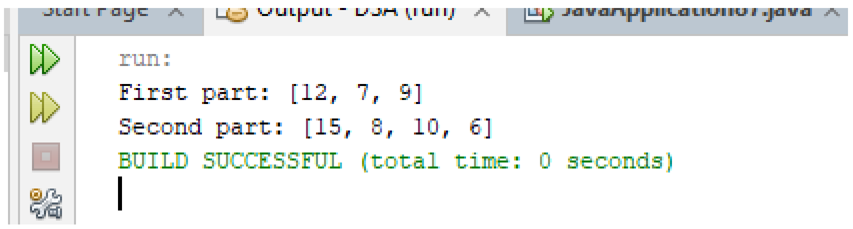
2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key

CODE



```java
package dsa;
import java.util.Scanner;
import java.util.Arrays;
public class JavaApplication87 {
    public static void main(String[] args) {
      int[] numbers = {12, 7, 9, 15, 8, 10, 6};
      int key = 15;
      splitArray(numbers, key);
    }
    public static void splitArray(int[] array, int key) {
      int index = -1;
      // Find the key's index
      for (int i = 0; i < array.length; i++) {
          if (array[i] == key) {
              index = i;
              break;
          }
      }
      // Split and print if key is found
      if (index != -1) {
          System.out.println("First part: " + Arrays.toString(Arrays.copyOfRange(array, 0, index)));
          System.out.println("Second part: " + Arrays.toString(Arrays.copyOfRange(array, index, array.length)));
      } else {
          System.out.println("Key not found.");
      }
    }
}
```

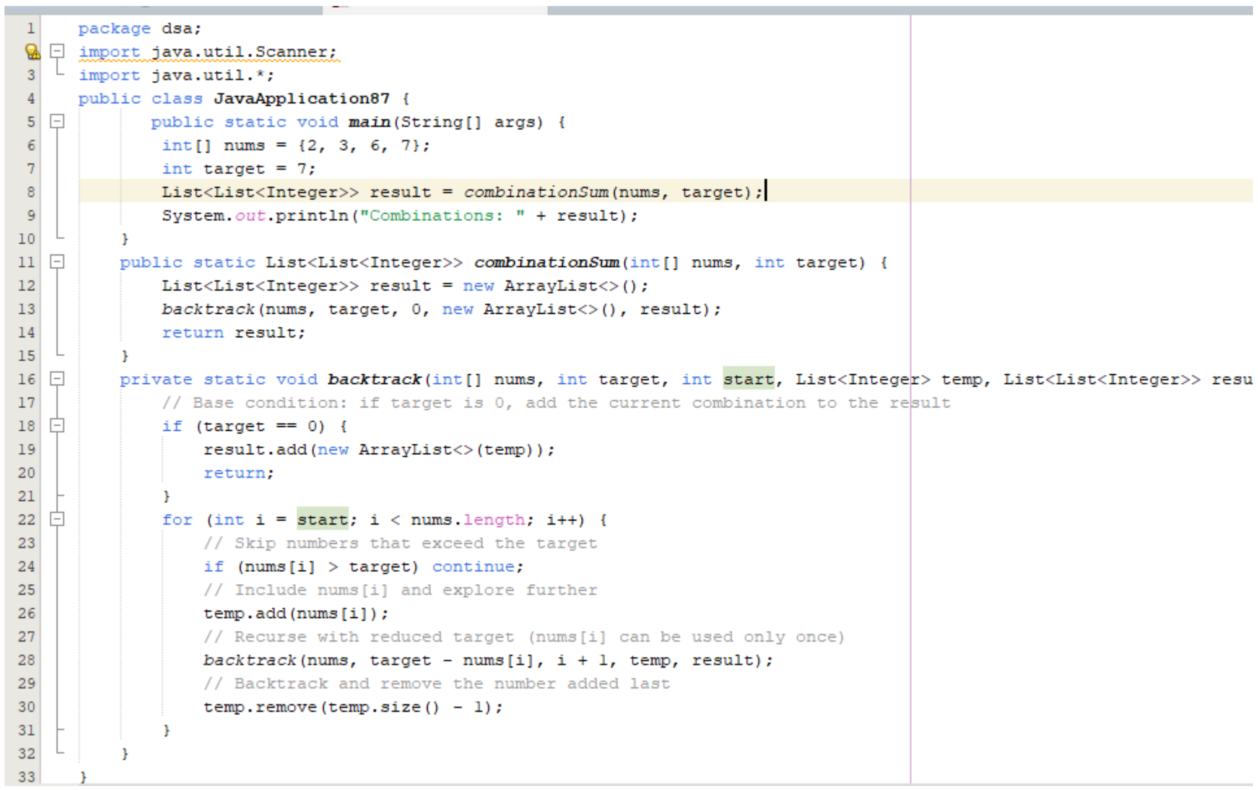
OUTPUT:



```
run:
First part: [12, 7, 9]
Second part: [15, 8, 10, 6]
BUILD SUCCESSFUL (total time: 0 seconds)
```

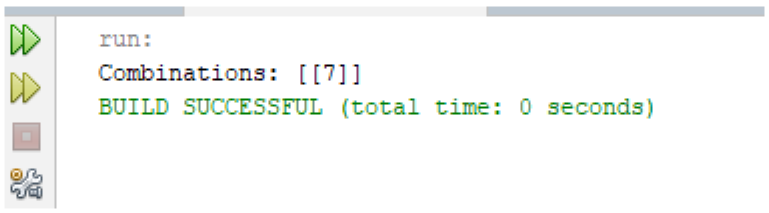
3. Given an array of distinct integers and a target integer, return all unique combinations of numbers that add up to the target. Each number can be used only once in the combination.

CODE:



```java
package dsa;
import java.util.Scanner;
import java.util.*;
public class JavaApplication87 {
        public static void main(String[] args) {
        int[] nums = {2, 3, 6, 7};
        int target = 7;
        List<List<Integer>> result = combinationSum(nums, target);
        System.out.println("Combinations: " + result);
    }
    public static List<List<Integer>> combinationSum(int[] nums, int target) {
        List<List<Integer>> result = new ArrayList<>();
        backtrack(nums, target, 0, new ArrayList<>(), result);
        return result;
    }
    private static void backtrack(int[] nums, int target, int start, List<Integer> temp, List<List<Integer>> resu
        // Base condition: if target is 0, add the current combination to the result
        if (target == 0) {
            result.add(new ArrayList<>(temp));
            return;
        }
        for (int i = start; i < nums.length; i++) {
            // Skip numbers that exceed the target
            if (nums[i] > target) continue;
            // Include nums[i] and explore further
            temp.add(nums[i]);
            // Recurse with reduced target (nums[i] can be used only once)
            backtrack(nums, target - nums[i], i + 1, temp, result);
            // Backtrack and remove the number added last
            temp.remove(temp.size() - 1);
        }
    }
}
```

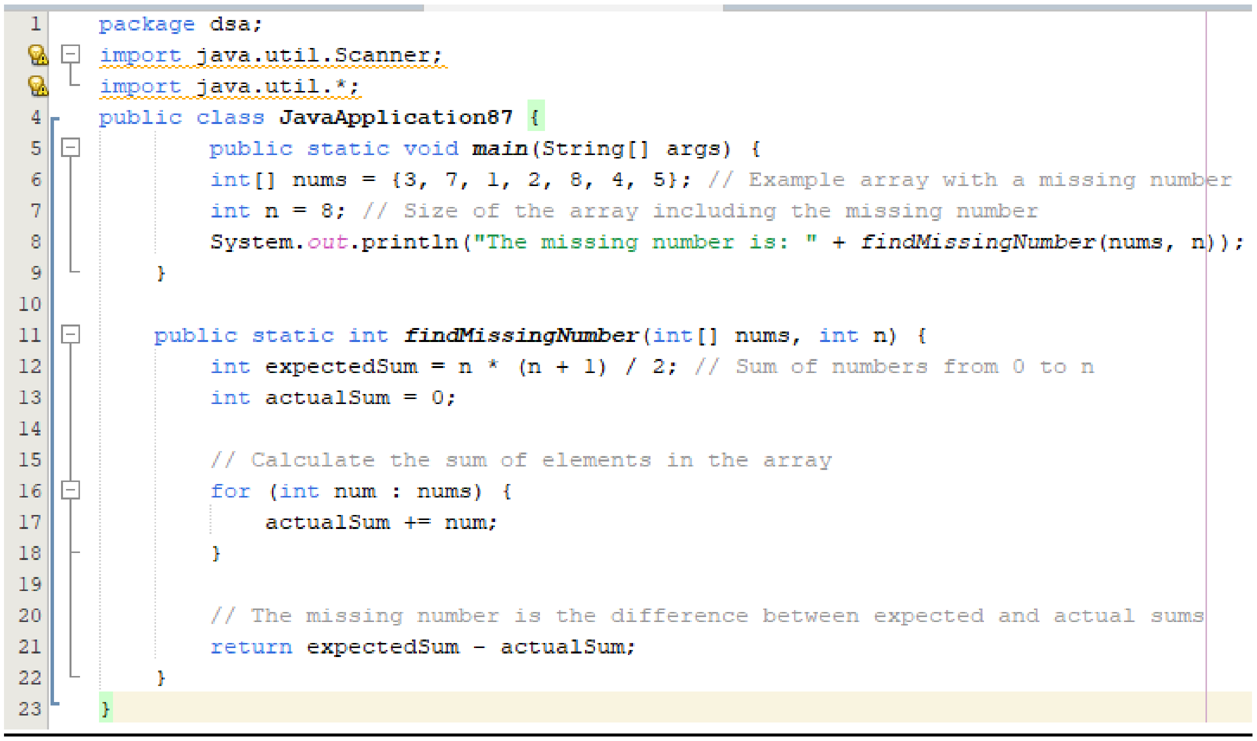
OUTPUT:



```
run:
Combinations: [[7]]
BUILD SUCCESSFUL (total time: 0 seconds)
```

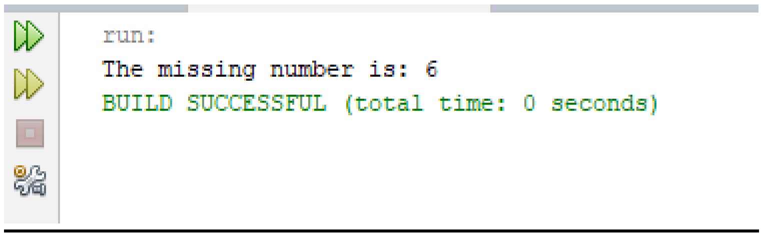
4. You are given an array containing n distinct numbers taken from 0, 1, 2, ..., n. Write a program to find the one number that is missing from the array.

CODE:



```java
package dsa;
import java.util.Scanner;
import java.util.*;
public class JavaApplication87 {
        public static void main(String[] args) {
        int[] nums = {3, 7, 1, 2, 8, 4, 5}; // Example array with a missing number
        int n = 8; // Size of the array including the missing number
        System.out.println("The missing number is: " + findMissingNumber(nums, n));
    }

    public static int findMissingNumber(int[] nums, int n) {
        int expectedSum = n * (n + 1) / 2; // Sum of numbers from 0 to n
        int actualSum = 0;

        // Calculate the sum of elements in the array
        for (int num : nums) {
            actualSum += num;
        }

        // The missing number is the difference between expected and actual sums
        return expectedSum - actualSum;
    }
}
```

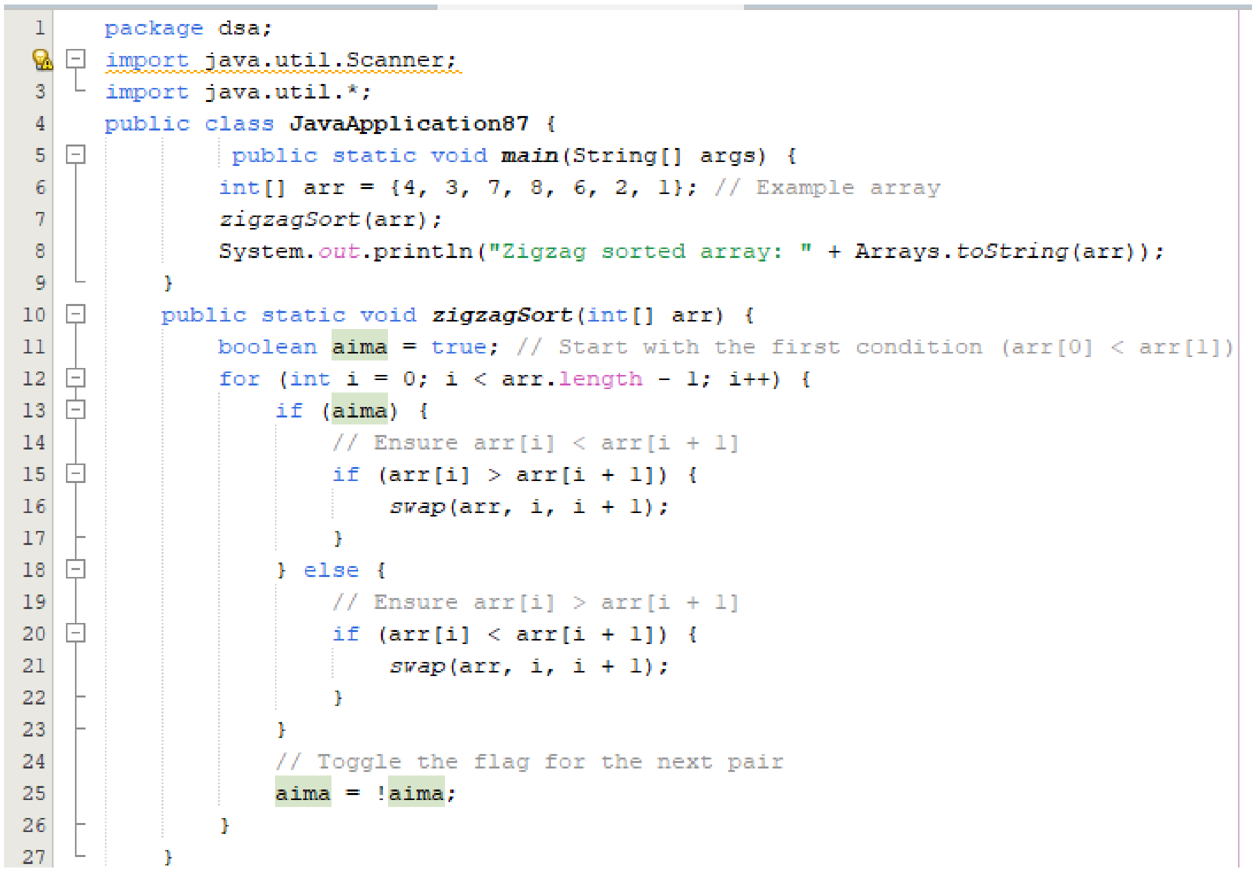
OUTPUT:



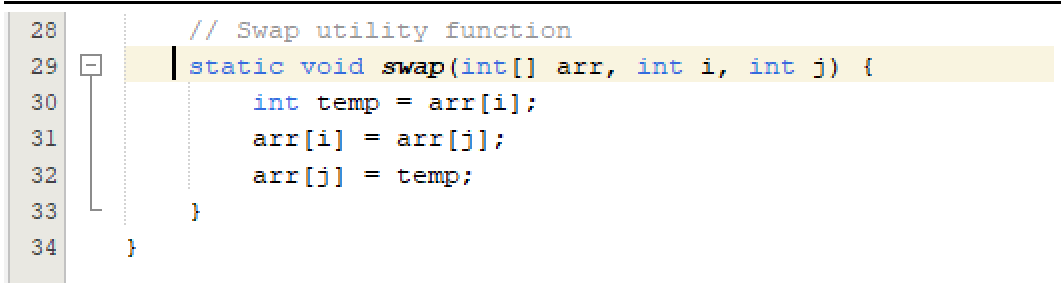
```
run:
The missing number is: 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

CODE:

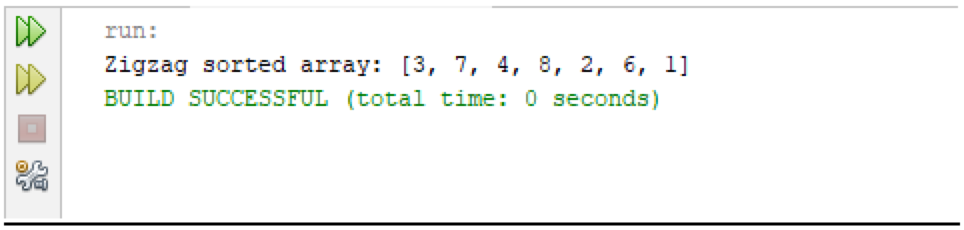


```
package dsa;
import java.util.Scanner;
import java.util.*;
public class JavaApplication87 {
          public static void main(String[] args) {
        int[] arr = {4, 3, 7, 8, 6, 2, 1}; // Example array
        zigzagSort(arr);
        System.out.println("Zigzag sorted array: " + Arrays.toString(arr));
    }
    public static void zigzagSort(int[] arr) {
        boolean aima = true; // Start with the first condition (arr[0] < arr[1])
        for (int i = 0; i < arr.length - 1; i++) {
            if (aima) {
                // Ensure arr[i] < arr[i + 1]
                if (arr[i] > arr[i + 1]) {
                    swap(arr, i, i + 1);
                }
            } else {
                // Ensure arr[i] > arr[i + 1]
                if (arr[i] < arr[i + 1]) {
                    swap(arr, i, i + 1);
                }
            }
            // Toggle the flag for the next pair
            aima = !aima;
        }
    }
```



```
    // Swap utility function
    static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

OUTPUT:



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