Lab #04

ARRAYS IN JAVA

Objective: To understand arrays and its memory allocation.

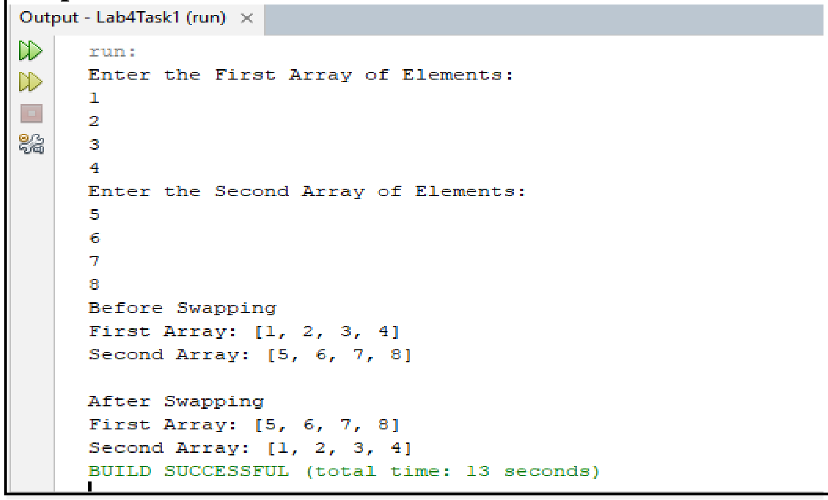
Lab Tasks:

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

Source Code:

```
import java.util.Arrays;
import java.util.Scanner;
public class Lab4Task1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int size = 4;
        int[] array1 = new int[size], array2 = new int[size];
        System.out.println("Enter the First Array of Elements: ");
        for (int i = 0; i < size; i++) array1[i] = sc.nextInt();
        System.out.println("Enter the Second Array of Elements: ");
        for (int i = 0; i < size; i++) array2[i] = sc.nextInt();
        System.out.println("Before Swapping");
        System.out.println("First Array: " + Arrays.toString(array1));
        System.out.println("Second Array: " + Arrays.toString(array2));
        for (int i = 0; i < size; i++) {
            int temp = array1[i];
            array1[i] = array2[i];
            array2[i] = temp;}
        System.out.println("\nAfter Swapping");
        System.out.println("First Array: " + Arrays.toString(array1));
        System.out.println("Second Array: " + Arrays.toString(array2));}}
```

Output:



Output - Lab4Task1 (run)

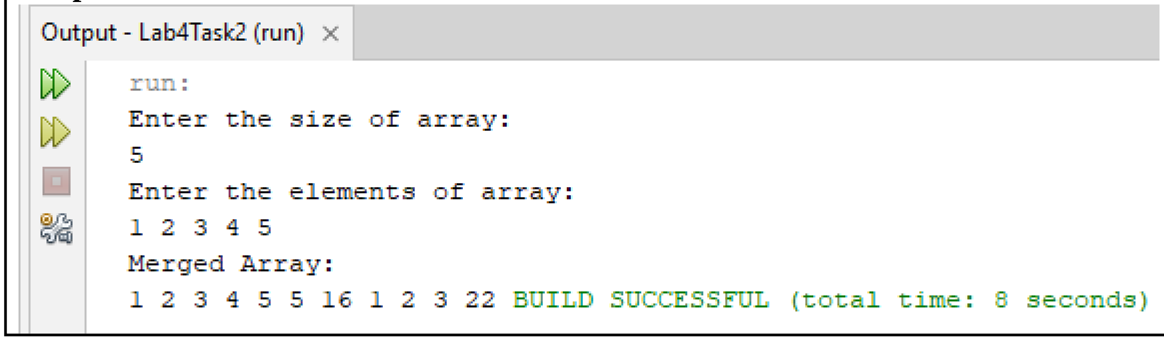
```
run:
Enter the First Array of Elements:
1
2
3
4
Enter the Second Array of Elements:
5
6
7
8
Before Swapping
First Array: [1, 2, 3, 4]
Second Array: [5, 6, 7, 8]

After Swapping
First Array: [5, 6, 7, 8]
Second Array: [1, 2, 3, 4]
BUILD SUCCESSFUL (total time: 13 seconds)
```

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

Source Code:

```
import java.util.Scanner;
public class Lab4Task2 {
    public static int[] mergeArrays(int[] array1, int[] array2) {
        int[] mergedArray = new int[array1.length + array2.length];
        for (int i = 0; i < array1.length; i++) {
            mergedArray[i] = array1[i];}
        for (int j = 0; j < array2.length; j++) {
            mergedArray[array1.length + j] = array2[j];}
        return mergedArray;}
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of array: ");
        int size = sc.nextInt();
        int[] a = new int[size];
        System.out.println("Enter the elements of array: ");
        for (int i = 0; i < size; i++) {
            a[i] = sc.nextInt();}
        int[] b = {5, 16, 1, 2, 3, 22};
        int[] mergedArray = mergeArrays(a, b);
        System.out.println("Merged Array: ");
        for (int i = 0; i < mergedArray.length; i++) {
            System.out.print(mergedArray[i] + " ");}}}
```

Output:



Output - Lab4Task2 (run)

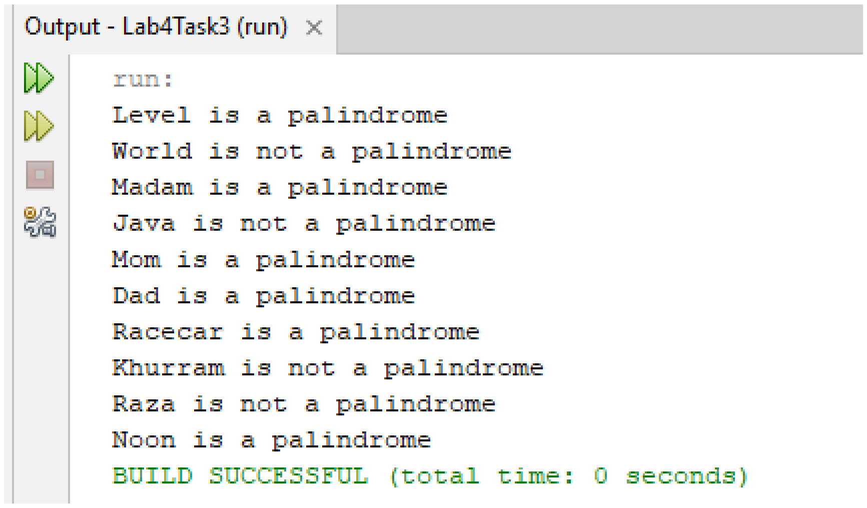
```
run:
Enter the size of array:
5
Enter the elements of array:
1 2 3 4 5
Merged Array:
1 2 3 4 5 5 16 1 2 3 22 BUILD SUCCESSFUL (total time: 8 seconds)
```

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

Source Code:

```
public class Lab4Task3 {
    public static void main(String[] args) {
        String[] words = {"Level", "World", "Madam", "Java", "Mom", "Dad", "Racecar", "Khurram", "Raza",
"Noon"};
        for (String word : words) {
            String reversed = new StringBuilder(word.toLowerCase()).reverse().toString();
            if (word.toLowerCase().equals(reversed)) {
                System.out.println(word + " is a palindrome");} else {
                System.out.println(word + " is not a palindrome");}}}}
```

Output:



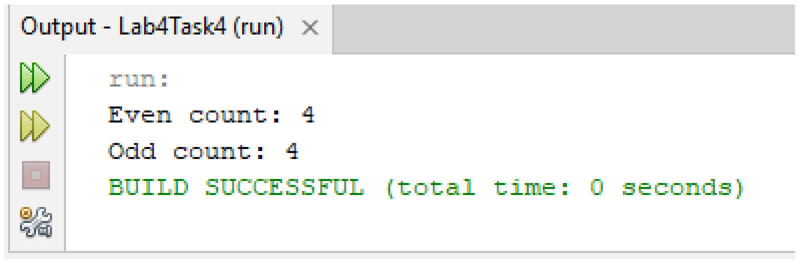
```
Output - Lab4Task3 (run)
run:
Level is a palindrome
World is not a palindrome
Madam is a palindrome
Java is not a palindrome
Mom is a palindrome
Dad is a palindrome
Racecar is a palindrome
Khurram is not a palindrome
Raza is not a palindrome
Noon is a palindrome
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Source Code:

```
public class Lab4Task4 {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8};
        int evens = 0, odds = 0;
        for (int num : numbers) {
            if (num % 2 == 0) evens++;
            else odds++;}
        System.out.println("Even count: " + evens);
        System.out.println("Odd count: " + odds);}}
```

Output:



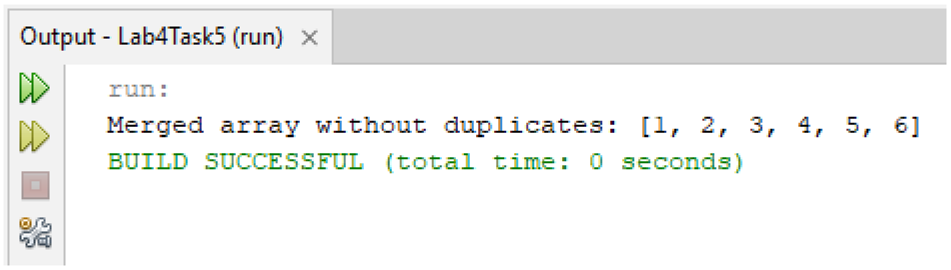
```
Output - Lab4Task4 (run)
run:
Even count: 4
Odd count: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Source Code:

```
import java.util.*;
public class Lab4Task5 {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4};
        int[] arr2 = {3, 4, 5, 6};
        Set<Integer> set = new HashSet<>();
        for (int num : arr1) set.add(num);
        for (int num : arr2) set.add(num);
        System.out.println("Merged array without duplicates: " + set);}}
```

Output:



```
run:
Merged array without duplicates: [1, 2, 3, 4, 5, 6]
BUILD SUCCESSFUL (total time: 0 seconds)
```

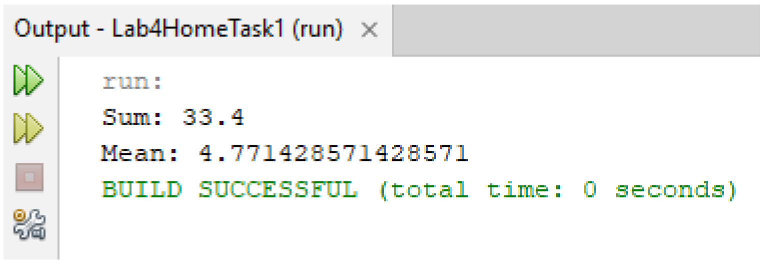
Home Tasks:

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.

Source Code:

```
public class Lab4HomeTask1 {
    public static void main(String[] args) {
        double[] arr = {2.5, 3.7, 4.1, 5.6, 6.3, 1.8, 9.4};
        double sum = 0;
        for (double num : arr) {
            sum += num;}
        double mean = sum / arr.length;
        System.out.println("Sum: " + sum);
        System.out.println("Mean: " + mean);}}
```

Output:



Output - Lab4HomeTask1 (run)

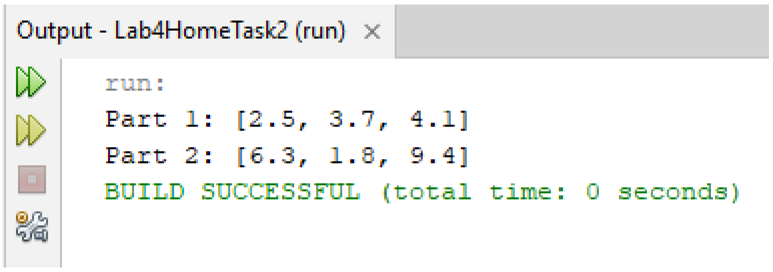
```
run:
Sum: 33.4
Mean: 4.771428571428571
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Source Code:

```
import java.util.Arrays;
public class Lab4HomeTask2 {
    public static void main(String[] args) {
        double[] arr = {2.5, 3.7, 4.1, 5.6, 6.3, 1.8, 9.4};
        double key = 5.6;
        splitArray(arr, key);}
    public static void splitArray(double[] arr, double key) {
        int index = Arrays.binarySearch(arr, key);
        if (index >= 0) {
            System.out.println("Part 1: " + Arrays.toString(Arrays.copyOfRange(arr, 0, index)));
            System.out.println("Part 2: " + Arrays.toString(Arrays.copyOfRange(arr, index + 1,
arr.length)));} else {
            System.out.println("Key not found!");}}}
```

Output:



Output - Lab4HomeTask2 (run)

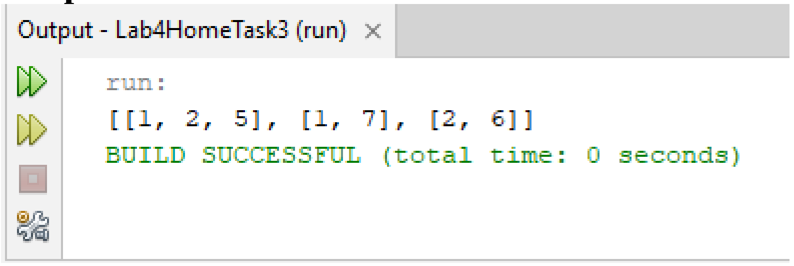
```
run:
Part 1: [2.5, 3.7, 4.1]
Part 2: [6.3, 1.8, 9.4]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Source Code:

```
import java.util.*;
public class Lab4HomeTask3 {
    public static void main(String[] args) {
        int[] arr = {10, 1, 2, 7, 6, 5};
        int target = 8;
        System.out.println(findCombinations(arr, target));}
    public static List<List<Integer>> findCombinations(int[] arr, int target) {
        Arrays.sort(arr);
        List<List<Integer>> result = new ArrayList<>();
        backtrack(arr, target, 0, new ArrayList<>(), result);
        return result;}
    private static void backtrack(int[] arr, int target, int start, List<Integer> temp,
List<List<Integer>> result) {
        if (target == 0) {
            result.add(new ArrayList<>(temp));
            return;}
        for (int i = start; i < arr.length && arr[i] <= target; i++) {
            if (i > start && arr[i] == arr[i - 1]) continue;
            temp.add(arr[i]);
            backtrack(arr, target - arr[i], i + 1, temp, result);
            temp.remove(temp.size() - 1);}}}
```

Output:



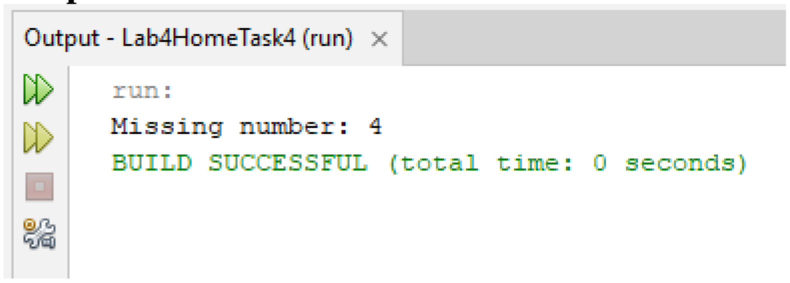
```
run:
[[1, 2, 5], [1, 7], [2, 6]]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Source Code:

```
public class Lab4HomeTask4 {
    public static void main(String[] args) {
        int[] arr = {0, 1, 2, 3, 5};
        System.out.println("Missing number: " + findMissingNumber(arr));}
    public static int findMissingNumber(int[] arr) {
        int n = arr.length;
        int expectedSum = n * (n + 1) / 2;
        int actualSum = 0;
        for (int num : arr) {
            actualSum += num;}
        return expectedSum - actualSum;}}
```

Output:



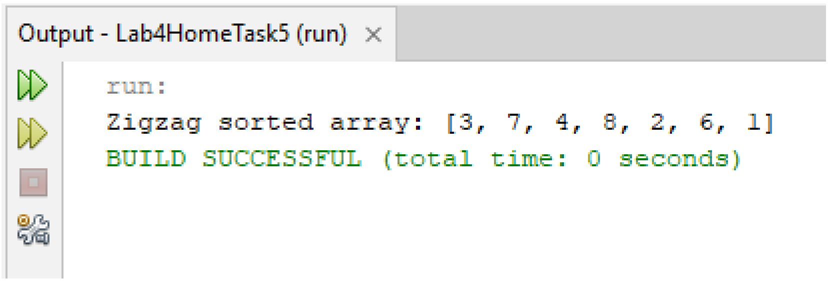
```
run:
Missing number: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Source Code:

```
import java.util.Arrays;

public class Lab4HomeTask5 {

    public static void main(String[] args) {

        int[] arr = {4, 3, 7, 8, 6, 2, 1};

        zigzagSort(arr);

        System.out.println("Zigzag sorted array: " + Arrays.toString(arr));}

    public static void zigzagSort(int[] arr) {

        for (int i = 0; i < arr.length - 1; i++) {

            if ((i % 2 == 0 && arr[i] > arr[i + 1]) || (i % 2 != 0 && arr[i] < arr[i + 1])) {

                // Swap if condition is met

                arr[i] ^= arr[i + 1];

                arr[i + 1] ^= arr[i];

                arr[i] ^= arr[i + 1];}}}}
```

Output:



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