LAB NO 4

ARRAYS IN JAVA

OBJECTIVE: To understand arrays and its memory allocation.

Question 1:

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

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3, 4};
        int[] array2 = {5, 6, 7, 8};
        System.out.println("Before Swap:");
        System.out.print("Array 1: ");
        printArray(array1);
        System.out.print("Array 2: ");
        printArray(array2);
        for (int i = 0; i < 4; i++) {
            int temp = array1[i];
            array1[i] = array2[i];
            array2[i] = temp;
        }
        System.out.println("\nAfter Swap:");
        System.out.print("Array 1: ");
        printArray(array1);
        System.out.print("Array 2: ");
        printArray(array2);
    }
    private static void printArray(int[] array) {
        for (int element : array) {
            System.out.print(element + " ");
        }
        System.out.println();
    }
}
```

OUTPUT:

```
Before Swap:
Array 1: 1 2 3 4
Array 2: 5 6 7 8

After Swap:
Array 1: 5 6 7 8
Array 2: 1 2 3 4
```

Question 2:

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

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3, 4};
        int[] array2 = {5, 6, 7, 8};
        System.out.println("Original Arrays:");
        printArray("Array 1", array1);
        printArray("Array 2", array2);
        int[] mergedArray = mergeArrays(array1, array2);
        printArray("Merged Array 1", mergedArray);
    }
    private static int[] mergeArrays(int[] array1, int[] array2) {
        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);
        return mergedArray;
    }
    private static void printArray(String label, int[] array) {
        System.out.print(label + ": ");
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

OUTPUT:

```
Original Arrays:
Array 1: 1 2 3 4
Array 2: 5 6 7 8
Merged Array 1: 1 2 3 4 5 6 7 8
```

Question 3:

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

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        String[] strings = {"apple", "hello", "radar", "ali", "level"};
        System.out.println("Palindrome Check Results:");
        for (String str : strings) {
            if (isPalindrome(str)) {
                System.out.println(str + " is a palindrome.");
            } else {
                System.out.println(str + " is not a palindrome.");
            }
        }
    }
    private static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;
        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }

        return true;
    }
}
```

OUTPUT:

```
apple is not a palindrome.
hello is not a palindrome.
radar is a palindrome.
ali is not a palindrome.
level is a palindrome.
```

Question 4:

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

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        int[] numbers = {12, 7, 34, 56, 89, 23, 14, 90, 15, 8};
        int evenCount = 0;
        int oddCount = 0;
        for (int num : numbers) {
            if (num % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }
        }
        System.out.println("Even numbers count: " + evenCount);
        System.out.println("Odd numbers count: " + oddCount);
    }
}
```

OUTPUT:

```
Even numbers count: 6
Odd numbers count: 4
----------------------------
```

Question 5:

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

INPUT:

```
import java.util.Arrays;
import java.util.HashSet;


public class Anas {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3, 4, 5};
        int[] array2 = {4, 5, 6, 7, 8};
        int[] mergedArray = mergeArrays(array1, array2);
        int[] resultArray = removeDuplicates(mergedArray);
        System.out.println("Merged Array without duplicates: " + Arrays.toString(resultArray));
    }
    private static int[] mergeArrays(int[] array1, int[] array2) {
        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);
        return mergedArray;
    }
    private static int[] removeDuplicates(int[] array) {
        HashSet<Integer> set = new HashSet<>();
        for (int num : array) {
            set.add(num);
        }
        int[] resultArray = new int[set.size()];
        int index = 0;
        for (int num : set) {
            resultArray[index++] = num;
        }

        return resultArray;
    }
}
```

OUTPUT:

```
--- exec:3.1.0:exec (default-cli) @ anas ---
Merged Array without duplicates: [1, 2, 3, 4, 5, 6, 7, 8]
```

HOME TASKS

Question 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.

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        double[] realNumbers = {1.5, 2.3, 3.7, 4.6, 5.1, 6.9, 7.2};
        double sum = calculateSum(realNumbers);
        double mean = calculateMean(sum, realNumbers.length);
        System.out.println("Array of Real Numbers: ");
        for (double num : realNumbers) {
            System.out.print(num + " ");
        }
        System.out.println("\nSum of the elements: " + sum);
        System.out.println("Mean of the elements: " + mean);
    }

    private static double calculateSum(double[] array) {
        double sum = 0.0;
        for (double num : array) {
            sum += num;
        }
        return sum;
    }
    private static double calculateMean(double sum, int length) {
        return sum / length;
    }
}
```

OUTPUT:

```
Array of Real Numbers:
1.5 2.3 3.7 4.6 5.1 6.9 7.2
Sum of the elements: 31.3
Mean of the elements: 4.4714285714285715
```

Question 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.

INPUT:

```
import java.util.Arrays;
public class Anas {
    public static void main(String[] args) {
        double[] realNumbers = {1.5, 2.3, 3.7, 4.6, 5.1, 6.9, 7.2};
        double sum = calculateSum(realNumbers);
        double mean = calculateMean(sum, realNumbers.length);
        System.out.println("Array of Real Numbers: ");
        printArray(realNumbers);
        System.out.println("Sum of the elements: " + sum);
        System.out.println("Mean of the elements: " + mean);
        double key = 4.6;
        System.out.println("\nSplitting array at key: " + key);
        splitArray(realNumbers, key);
    }
    private static double calculateSum(double[] array) {
        double sum = 0.0;
        for (double num : array) {
            sum += num;
        }
        return sum;
    }
    private static double calculateMean(double sum, int length) {
        return sum / length;
    }
    private static void splitArray(double[] array, double key) {
        int index = -1;
        for (int i = 0; i < array.length; i++) {
            if (array[i] == key) {
                index = i;
                break;
            }
        }
        if (index != -1) {
            double[] part1 = Arrays.copyOfRange(array, 0, index);
            double[] part2 = Arrays.copyOfRange(array, index, array.length);
            System.out.println("First part of the array (before the key): " + Arrays.toString(part1));
            System.out.println("Second part of the array (from the key onwards): " + Arrays.toString(part2));
        } else {
            System.out.println("Key " + key + " not found in the array.");
        }
    }
    private static void printArray(double[] array) {
        for (double num : array) {
            System.out.print(num + " ");
        }
```

OUTPUT:

```
Array of Real Numbers:
1.5 2.3 3.7 4.6 5.1 6.9 7.2
Sum of the elements: 31.3
Mean of the elements: 4.4714285714285715

Splitting array at key: 4.6
First part of the array (before the key): [1.5, 2.3, 3.7]
Second part of the array (from the key onwards): [4.6, 5.1, 6.9, 7.2]
```

Question 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.

INPUT:

```
public class Anas {
    public static void main(String[] args) {
        int[] candidates = {2, 3, 6, 7};
        int target = 7;
        List<List<Integer>> result = combinationSum(candidates, target);
        System.out.println("Unique combinations that sum to " + target + ":");
        for (List<Integer> combination : result) {
            System.out.println(combination);
        }
    }
    public static List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(candidates);
        backtrack(result, new ArrayList<>(), candidates, target, 0);
        return result;
    }
    private static void backtrack(List<List<Integer>> result, List<Integer> tempList,
                                  int[] candidates, int target, int start) {
        if (target == 0) {
            result.add(new ArrayList<>(tempList));
            return;
        }
        for (int i = start; i < candidates.length; i++) {
            if (candidates[i] > target) {
                break;
            }
            tempList.add(candidates[i]);
            backtrack(result, tempList, candidates, target - candidates[i], i + 1);
            tempList.remove(tempList.size() - 1);
        }
    }
```

OUTPUT:

```
Unique combinations that sum to 7:
[7]
```

Question 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.

Input:

```
public class Anas {
    public static void main(String[] args) {
        int[] nums = {3, 7, 1, 2, 8, 4, 5};
        int n = nums.length + 1;
        int missingNumber = findMissingNumber(nums, n);
        System.out.println("The missing number is: " + missingNumber);
    }
    public static int findMissingNumber(int[] nums, int n) {
        int expectedSum = n * (n + 1) / 2;
        int actualSum = 0;
        for (int num : nums) {
            actualSum += num;
        }
        return expectedSum - actualSum;
    }
}
```

Output:

```
The missing number is: 6
```

Question 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.

Input:

```
public class Anas {
    public static void main(String[] args) {
        int[] arr = {4, 3, 7, 8, 6, 2, 1};
        System.out.println("Original array: " + Arrays.toString(arr));
        zigzagSort(arr);
        System.out.println("Zigzag sorted array: " + Arrays.toString(arr));
    }
    public static void zigzagSort(int[] arr) {
        Arrays.sort(arr);
        for (int i = 1; i < arr.length; i += 2) {
            if (i + 1 < arr.length && arr[i] < arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
            if (i - 1 >= 0 && arr[i - 1] > arr[i]) {
                int temp = arr[i - 1];
                arr[i - 1] = arr[i];
                arr[i] = temp;
            }
        }
    }
}
```

Output:

```
Original array: [4, 3, 7, 8, 6, 2, 1]
Zigzag sorted array: [1, 3, 2, 6, 4, 8, 7]
```