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

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

        // Swap elements
        for (int i = 0; i < array1.length; i++) {
            int temp = array1[i];
            array1[i] = array2[i];
            array2[i] = temp;
        }

        System.out.println("Array 1 after swap:");
        for (int i : array1) System.out.print(i + " ");

        System.out.println("\nArray 2 after swap:");
        for (int i : array2) System.out.print(i + " ");
    }
}
```

OUTPUT:

Output

```
java -cp /tmp/p3GLgxC6Us/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:

Main.java

```java
import java.util.Arrays;

public class ArrayMerge {
    private int[] array;

    public ArrayMerge(int[] initialArray) {
        this.array = initialArray;
    }

    public void mergeArray(int[] newArray) {
        int[] mergedArray = new int[array.length + newArray.length];

        System.arraycopy(array, 0, mergedArray, 0, array.length);
        System.arraycopy(newArray, 0, mergedArray, array.length, newArray.length);

        this.array = mergedArray;
    }

    public void displayArray() {
        System.out.println(Arrays.toString(array));
    }

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

        ArrayMerge obj = new ArrayMerge(initialArray);
        System.out.println("Initial Array:");
        obj.displayArray();

        obj.mergeArray(newArray);
        System.out.println("After Merging:");
        obj.displayArray();
    }
}
```

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

```java
public class PalindromeCheck {
    public static boolean isPalindrome(String str) {
        int start = 0, end = str.length() - 1;
        while (start < end) {
            if (str.charAt(start) != str.charAt(end)) return false;
            start++;
            end--;
        }
        return true;
    }

    public static void main(String[] args) {
        String[] words = {"ZAINAB", "AHMED"};

        for (String word : words) {
            if (isPalindrome(word)) {
                System.out.println(word + " is a palindrome.");
            } else {
                System.out.println(word + " is not a palindrome.");
            }
        }
    }
}
```

OUTPUT:

Output

```
java -cp /tmp/s6BVUOPma1/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

```java
public class EvenOddCount {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8};
        int evenCount = 0, 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:

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:

Main.java

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

public class MergeAndRemoveDuplicates {
    public static int[] mergeAndRemoveDuplicates(int[] arr1, int[] arr2) {
        HashSet<Integer> set = new HashSet<>();

        for (int num : arr1) set.add(num);
        for (int num : arr2) set.add(num);

        int[] result = new int[set.size()];
        int i = 0;
        for (int num : set) {
            result[i++] = num;
        }

        return result;
    }

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

        int[] mergedArray = mergeAndRemoveDuplicates(array1, array2);
        System.out.println("Merged array without duplicates: " + Arrays.toString(mergedArray));
    }
}
```

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

```java
public class RealNumberArray {
    private double[] array;

    public RealNumberArray(double[] array) {
        this.array = array;
    }

    public void calculateSumAndMean() {
        double sum = 0;
        for (double num : array) {
            sum += num;
        }
        double mean = sum / array.length;

        System.out.println("Sum: " + sum);
        System.out.println("Mean: " + mean);
    }

    // Memory Management depiction
    public void displayMemoryAddresses() {
        System.out.println("Memory address of array object: " + System.identityHashCode(array));
        for (int i = 0; i < array.length; i++) {
            System.out.println("Memory address of element at index " + i + ": " + System.identityHashCode(array[i]
                ));
        }
    }

    public static void main(String[] args) {
        double[] numbers = {1.1, 2.2, 3.4, 4.1, 5.8, 6.9, 7.0};
        RealNumberArray obj = new RealNumberArray(numbers);

        obj.calculateSumAndMean();
        obj.displayMemoryAddresses();
    }
}
```

OUTPUT:

Output

```
java -cp /tmp/BMzoM3IgqD/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

```java
    public SplitArray(double[] array) {
        this.array = array;
    }

    public void splitAtKey(double key) {
        int index = -1;

        // Find the index of the key in the array
        for (int i = 0; i < array.length; i++) {
            if (array[i] == key) {
                index = i;
                break;
            }
        }

        if (index == -1) {
            System.out.println("Key not found in array.");
            return;
        }

        // Split the array into two parts
        double[] firstPart = new double[index + 1];
        double[] secondPart = new double[array.length - index - 1];

        System.arraycopy(array, 0, firstPart, 0, index + 1);
        System.arraycopy(array, index + 1, secondPart, 0, array.length - index - 1);

        System.out.println("First Part: " + java.util.Arrays.toString(firstPart));
        System.out.println("Second Part: " + java.util.Arrays.toString(secondPart));
    }

    public static void main(String[] args) {
        double[] numbers = {1.1, 2.8, 3.6, 4.1, 5.8, 6.2, 7.0};
        SplitArray obj = new SplitArray(numbers);

        obj.splitAtKey(4.1);
    }
}
```

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:

Main.java

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

public class UniqueCombinationSum {
    public static List<List<Integer>> findCombinations(int[] nums, int target) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(nums);
        backtrack(result, new ArrayList<>(), nums, target, 0);
        return result;
    }

    private static void backtrack(List<List<Integer>> result, List<Integer> temp, int[] nums, int target, int start)
        {
        if (target == 0) {
            result.add(new ArrayList<>(temp));
            return;
        }
        for (int i = start; i < nums.length; i++) {
            if (nums[i] > target) break;
            if (i > start && nums[i] == nums[i - 1]) continue; // Skip duplicates
            temp.add(nums[i]);
            backtrack(result, temp, nums, target - nums[i], i + 1);
            temp.remove(temp.size() - 1);
        }
    }

    public static void main(String[] args) {
        int[] nums = {10, 9, 2, 3, 6, 5};
        int target = 8;
        List<List<Integer>> combinations = findCombinations(nums, target);
        System.out.println("Combinations: " + combinations);
    }
}
```

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

```java
public class MissingNumber {
    public static int findMissingNumber(int[] nums) {
        int n = nums.length;
        int totalSum = n * (n + 1) / 2;
        int sum = 0;
        for (int num : nums) {
            sum += num;
        }
        return totalSum - sum;
    }

    public static void main(String[] args) {
        int[] nums = {0, 1, 3, 4, 5};
        System.out.println("Missing Number: " + findMissingNumber(nums));
    }
}
```

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

```java
import java.util.Arrays;

public class ZigzagPattern {
    public static void zigzagSort(int[] nums) {
        boolean flag = true; // If true, "less than" relation is expected

        for (int i = 0; i < nums.length - 1; i++) {
            if (flag) {
                // If current element is greater than the next element, swap them
                if (nums[i] > nums[i + 1]) {
                    int temp = nums[i];
                    nums[i] = nums[i + 1];
                    nums[i + 1] = temp;
                }
            } else {
                // If current element is less than the next element, swap them
                if (nums[i] < nums[i + 1]) {
                    int temp = nums[i];
                    nums[i] = nums[i + 1];
                    nums[i + 1] = temp;
                }
            }
            flag = !flag; // Flip flag for the next pair
        }
    }

    public static void main(String[] args) {
        int[] nums = {4, 3, 7, 9, 6, 2, 1};
        zigzagSort(nums);
        System.out.println("Zigzag Array: " + Arrays.toString(nums));
    }
}
```

OUTPUT:

Output

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

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