LAIBA SHAHID
2023F-BSE-247

LAB 04

ARRAYS IN JAVA

OBJECTIVE: To understand arrays and their memory allocation.

LAB TASKS

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

INPUT:

```
package lab04;
import java.util.Arrays;
public class Lab04 {
    public static void main(String[] args) {
        int[] A1 = {2,4,6,8};
        int[] A2 = {1,3,5,7};

        System.out.println(x: "Original arrays:");
        System.out.println("Array 1: " + Arrays.toString(a: A1));
        System.out.println("Array 2: " + Arrays.toString(a: A2));

        for (int i = 0; i < 4; i++) {
            int temp = A1[i];
            A1[i] = A2[i];
            A2[i] = temp;
        }
        System.out.println(x: "Swapped arrays:");
        System.out.println("Array 1: " + Arrays.toString(a: A1));
        System.out.println("Array 2: " + Arrays.toString(a: A2));
    }
}
```

OUTPUT:

```
Original arrays:
Array 1: [2, 4, 6, 8]
Array 2: [1, 3, 5, 7]
Swapped arrays:
Array 1: [1, 3, 5, 7]
Array 2: [2, 4, 6, 8]
```

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

INPUT:

```
import java.util.Arrays;
public class Lab04 {
    public static void main(String[] args) {

        int[] A1 = {56,97,34,89};
        int[] A2 = {45,78,23,556};
        int[] mergedArray = mergeArrays(A1, A2);
        System.out.println("Merged array: " + Arrays.toString(a: mergedArray));
    }

    public static int[] mergeArrays(int[] A1, int[] A2) {
        int[] mergedArray = new int[A1.length + A2.length];

        for (int i = 0; i < A1.length; i++) {
            mergedArray[i] = A1[i];
        }
        for (int i = 0; i < A2.length; i++) {
            mergedArray[A1.length + i] = A2[i];
        }

        return mergedArray;
    }
}
```

OUTPUT:

```
run:
Merged array: [56, 97, 34, 89, 45, 78, 23, 556]
```

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

INPUT:

```
import java.util.Scanner;
public class Lab04 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of strings: ");
        int size = scanner.nextInt();
        scanner.nextLine();
        String[] words = new String[size];
        System.out.println("Enter the strings:");
        for (int i = 0; i < size; i++) {
            words[i] = scanner.nextLine();
        }
        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 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:

```
Enter the number of strings: 2
Enter the strings:
level
spain
level is a palindrome.
spain is not a palindrome.
```

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

INPUT:

```
public class Lab04 {
    public static void main(String[] args) {
        int[] no = {3,6,5,7,5};
        int even = 0;
        int odd = 0;

        for (int num : no) {
            if (num % 2 == 0) {
                even++;
            } else {
                odd++;
            }
        }

        System.out.println("Number of even numbers: " + even);
        System.out.println("Number of odd numbers: " + odd);

    }
}
```

OUTPUT:

```
Number of even numbers: 1
Number of odd numbers: 4
```

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

INPUT:

```
import java.util.Arrays;
public class Lab04 {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {4, 5, 6, 7, 8};

        int[] mergedArray = mergeRemoveDuplicate(arr1, arr2);
        System.out.println(x: Arrays.toString(a: mergedArray));
    }
    public static int[] mergeRemoveDuplicate(int[] arr1, int[] arr2) {
        int[] mergedArray = new int[arr1.length + arr2.length];

        System.arraycopy(src: arr1, srcPos: 0, dest:mergedArray, destPos: 0, length: arr1.length);
        System.arraycopy(src: arr2, srcPos: 0, dest:mergedArray, destPos: arr1.length, length: arr2.length);

        int[] uniqueArray = new int[mergedArray.length];
        int uniqueCount = 0;

        for (int i = 0; i < mergedArray.length; i++) {
            boolean isDuplicate = false;
            for (int j = 0; j < uniqueCount; j++) {
                if (mergedArray[i] == uniqueArray[j]) {
                    isDuplicate = true;
                    break;
                }
            }
            if (!isDuplicate) {
                uniqueArray[uniqueCount++] = mergedArray[i];
            }
        }
        return Arrays.copyOf(original: uniqueArray, newLength:uniqueCount);
    }
}
```

OUTPUT:

```
run:
[1, 2, 3, 4, 5, 6, 7, 8]
```

HOME TASK

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 Lab04 {
    public static void main(String[] args) {
        double[] numbers = {1.2, 2.5, 3.8, 4.1, 5.6, 6.3, 7.0};
        double sum = 0;

        for (double num : numbers) {
            sum += num;
        }

        double mean = sum / numbers.length;
        System.out.println("Sum: " + sum);
        System.out.println("Mean: " + mean);
    }
}
```

OUTPUT:

```
Sum: 30.5
Mean: 4.357142857142857
```

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 Lab04 {
    public static void main(String[] args) {
        int[] Arr = {1, 2, 3, 4, 5, 6};
        int key = 3;
        splitArray(Arr, key);
    }
    public static void splitArray(int[] Arr, int key) {
        int index = -1;
            for (int i = 0; i < Arr.length; i++) {
            if (Arr[i] == key) {
                index = i;
                break;
            }
        }

        if (index != -1) {
            System.out.println("First Part: " + Arrays.toString(a: Arrays.copyOfRange(original: Arr, from:0, to: index)));
            System.out.println("Second Part: " + Arrays.toString(a: Arrays.copyOfRange(original: Arr, from:index, to: Arr.length)));
        } else {
            System.out.println(x: "Key not found.");
        }
    }
}
```

OUTPUT:

```
First Part: [1, 2]
Second Part: [3, 4, 5, 6]
```

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 Lab04 {
    public static void main(String[] args) {
        int[] arr = {10, 1, 2, 7, 6, 5};
        int target = 8;
        System.out.println(x: findComb(arr, target));
    }

    public static List<List<Integer>> findComb(int[] arr, int target) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(a: arr);
        rightcomb(result, new ArrayList<>(), arr, remain: target, start: 0);
        return result;
    }
    private static void rightcomb(List<List<Integer>> result, List<Integer> tempList, int[] arr, int remain, int start) {
        if (remain == 0) {
            result.add(new ArrayList<>(c: tempList));
        } else if (remain > 0) {
            for (int i = start; i < arr.length; i++) {
                if (i > start && arr[i] == arr[i - 1]) continue;
                tempList.add(arr[i]);
                rightcomb(result, tempList, arr, remain - arr[i], i + 1);
                tempList.remove(tempList.size() - 1);
            }
        }
    }
}
```

OUTPUT:

```
[[1, 2, 5], [1, 7], [2, 6]]
```

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 Lab04 {
    public static void main(String[] args) {
        int[] arr = {3, 0, 1};
        System.out.println("The missing number is: " + findNumber(arr));
    }
    public static int findNumber(int[] arr) {
        int n = arr.length;
        int sumOfArray = 0;
        int sumOfRange = 0;

        for (int num : arr) {
            sumOfArray += num;
        }
        for (int i = 0; i <= n; i++) {
            sumOfRange += i;
        }
        return sumOfRange - sumOfArray;
    }
}
```

OUTPUT:

```
The missing number is: 2
```

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:

```
import java.util.Arrays;

public class Lab04 {
    public static void ZZsort(int[] arr) {
        Arrays.sort(a: arr);
        for (int i = 1; i < arr.length - 1; i += 2) {
            int temp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = temp;
        }
    }

    public static void main(String[] args) {
        int[] arr = {4, 3, 7, 8, 6, 2, 1};
        ZZsort(arr);
        System.out.println("Zigzag Array: " + Arrays.toString(a: arr));
    }
}
```

OUTPUT:

```
Zigzag Array: [1, 3, 2, 6, 4, 8, 7]
```