**Lab # 04**

**Lab Tasks:**

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

**Code:**

package dsa\_lab;

public class DSA\_Lab\_4 {

public static void main(String[] args){

int arr1[] = {3, 88, 25, 14};

int arr2[] = {4, 89, 21, 17};

System.out.print("Original Array1: ");

printArray(arr1);

System.out.print("Original Array2: ");

printArray(arr2);

// Swapping arrays

swap(arr1, arr2);

System.out.print("Swapped Array1: ");

printArray(arr1);

System.out.print("Swapped Array2: ");

printArray(arr2);

System.out.println();

}

public static void swap(int ar[], int ar1[]) {

int[] temp = new int[ar.length];

// Swap elements using temp array

for (int i = 0; i < ar.length; i++) {

temp[i] = ar[i];

ar[i] = ar1[i];

ar1[i] = temp[i];

}

}

public static void printArray(int[] array) {

for (int i : array) {

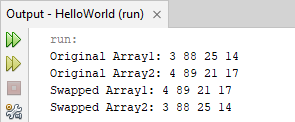
System.out.print(i + " ");

}System.out.println();

}

}

**Output:**



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

**Code:**

package dsa\_lab;

class DSA\_Lab {

private int[] existing;

public DSA\_Lab(int[] existingArray) {

this.existing = existingArray;

}

public int[] merge(int[] newArray) {

int[] mergedArray = new int[existing.length + newArray.length];

//from the existing array

for (int i = 0; i < existing.length; i++) {

mergedArray[i] = existing[i];

}

//from the new array

for (int i = 0; i < newArray.length; i++) {

mergedArray[existing.length + i] = newArray[i];

}

return mergedArray;

}

public static void main(String[] args) {

int[] fArray = {94, 100,110, 117};

int[] sArray = {123, 129, 132, 140};

DSA\_Lab merger = new DSA\_Lab(fArray);//Calling Existing via Constructor

int[] outputArray = merger.merge(sArray);//Called New via Method, stored in result named array

System.out.print("Existing Array is: ");

for (int elem: fArray) {

System.out.print(elem + " ");

}

System.out.print("\nNew Array is: ");

for (int elem: sArray) {

System.out.print(elem + " ");

}System.out.print("\nMerged Array is: ");

for (int elem: outputArray) {

System.out.print(elem + " ");

}

System.out.println();

}

}

**Output:**

A screenshot of a computer

Description automatically generated

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

**Code:**

package dsa\_lab;

public class DSA\_Lab\_4 {

public static void main(String[] args){

String arr[]={"Hussain","Mustafa","Amma"};

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

System.out.println(arr[i] + " is palindrome: " + isPal(arr[i]));

}

}

public static boolean isPal(String check) {

int f = 0;

int l = check.length() - 1;

check = check.toLowerCase();

while (f < l) {

if(f==l){

return true;

}

if (check.charAt(f) != check.charAt(l)){

return false;

}

f++;

l--;

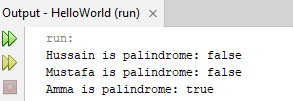
}

return true;

}

}

**Output:**



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

**Code:**

package dsa\_lab;

public class DSA\_Lab\_4 {

public static void main(String[] args){

int arr[]={7,8,0,967,1912,36,9041,1354,988};

evenOdd(arr);

}

public static void evenOdd(int[] ary){

int even=0;

int odd=0;

for(int e: ary){

if(e%2==0){

even=even+1;

}

else{

odd=odd+1;

}

}

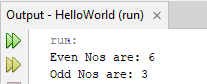
System.out.println("Even Nos are: "+even);

System.out.println("Odd Nos are: "+odd);

}

}

**Output:**



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

**Code:**

package dsa\_lab;

import java.util.\*;

public class DSA\_Lab\_4 {

public static void main(String[] args){

int arr1[]={9,8,0,987,1812,32,9031,1324,987};

int arr2[]={90,78,90,1812,320,31,1234,987};

merger(arr1,arr2);

}

public static void merger(int[] ar1,int[] ar2 ){

int temp[]=new int[ar1.length+ar2.length];

// Copy elements from ar1 into temp

for (int i = 0; i < ar1.length; i++) {

temp[i] = ar1[i];

}

// Copy elements from ar2 into temp, starting from where ar1 left off

for (int i = 0; i < ar2.length; i++) {

temp[ar1.length + i] = ar2[i];

}

// Use a LinkedHashSet to remove duplicates and maintain insertion order

Set<Integer> uniqueElements = new LinkedHashSet<>();

for (int e : temp) {

uniqueElements.add(e);

}

// Print merged array without duplicates

System.out.print("Merged Array : [ ");

for (int e : uniqueElements) {

System.out.print(e + " ");

}System.out.print("]");

System.out.println();

}

}

**Output:**

A screenshot of a computer

Description automatically generated

**Home Tasks:**

**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.**

**Code:**

public class HomeTaskLab\_4 {

public static void main(String[] args){

double array[] = new double[7];

Scanner sc = new Scanner(System.in);

for(int i = 0; i < array.length; i++){

System.out.print("Enter Real No: ");

array[i] = sc.nextDouble();

}

realFunc(array);

}

public static void realFunc(double arr[]){

double sum = 0;

double mean = 0;

int count = arr.length;

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

sum += arr[i];

}

mean = sum / count;

System.out.println("Sum is: " + sum);

System.out.println("Mean is: " + mean);

}

}

**Memory Management**

* **Array Storage**: The array array is allocated memory for 7 double values, and each element is stored in consecutive memory locations.
* **Stack Memory**: Local variables like sum, mean, and count are stored in the stack memory since they are method-specific.
* **Heap Memory**: The array object is stored in heap memory because it is dynamically allocated. The Scanner object sc is also allocated in the heap, managing input functionality.

**Output:**

A screenshot of a computer

Description automatically generated

**Task #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:**

class DSA\_Lab5 {

private int[] existing;

private int k;

public DSA\_Lab5(int[] existingArray, int k) {

this.existing = existingArray;

this.k = k;

}

public int[][] split() {

int index = -1;

// Find index of element 'k'

for (int i = 0; i < existing.length; i++) {

if(existing[i] == k) {

index = i;

break;

}

}

// If element is found, split the array

if (index != -1) {

int[] firstPart = Arrays.copyOfRange(existing, 0, index);

int[] secondPart = Arrays.copyOfRange(existing, index , existing.length);

return new int[][] { firstPart, secondPart };

} else {

System.out.println("Element " + k + " not found in the array.");

return new int[][] { existing, new int[0] }; // Return original array and empty array if 'k' isn’t found

}

}

public static void main(String[] args) {

int[] fArray = {94, 100, 110, 117,98,131,23,4,53,24,3};

int k = 110;

DSA\_Lab5 splitArray = new DSA\_Lab5(fArray, k);

int[][] outputArrays = splitArray.split();

System.out.print("Existing Array is: ");

for (int elem : fArray) {

System.out.print(elem + " ");

}

System.out.print("\nKey is: " + k);

System.out.print("\nSecond part of Array is: ");

for (int elem : outputArrays[1]) {

System.out.print(elem + " ");

}

System.out.println();

}

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**Task #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:**

public class HomeTaskLab\_4 {

public static void main(String[] args){

int[] nums = {2, 3, 6, 7,5,1,4};

int target = 8;

System.out.println("Combinations that sum to " + target + ":");

findCombinations(nums, target);

}

public static void findCombinations(int[] nums, int target) {

Arrays.sort(nums); // Optional sorting

int[] combination = new int[nums.length]; // Array to store current combination

findCombinations(nums, target, 0, combination, 0);

}

private static void findCombinations(int[] nums, int target, int start, int[] combination, int pos) {

if (target == 0) {

// Print the valid combination

for (int i = 0; i < pos; i++) {

System.out.print(combination[i] + " ");

}

System.out.println();

return;

}

for (int i = start; i < nums.length; i++) {

if (nums[i] > target) break; // Stop if number is greater than remaining target

combination[pos] = nums[i]; // Add number to combination array

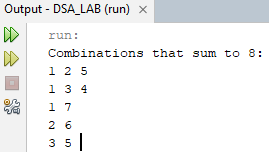
findCombinations(nums, target - nums[i], i + 1, combination, pos + 1); // Recursion when target reduced

}

}

}

**Output:**

****

**Task #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:**

public class HomeTaskLab\_4 {

public static void main(String[] args){

int[] given = {6,2,3,4,5,1,10}; // Example array

int missed = findM(given);

System.out.println("The missing number is: " + missed);

}

public static int findM(int[] no) {

int n = no.length;

//Expected sum from 0 to n

int expSum = n \* (n + 1) / 2;//

System.out.println("Expected Sum: "+expSum);

//Actual Sum

int actSum = 0;

for (int e : no) {

actSum += e;

}

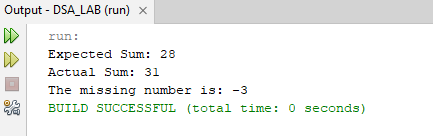
System.out.println("Actual Sum: "+actSum);

return expSum - actSum;

}

}

**Output:**

****

**Task #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:**

public class HomeTaskLab\_4 {

public static void main(String[] args){

int[] ary = {1,23,3,52,76};

zSort(ary);

// Print the zigzag pattern array

System.out.println("Zigzag pattern array:");

for (int num : ary) {

System.out.print(num + " ");

}

}

public static void zSort(int[] array) {

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

if (i % 2 == 0) {

if (array[i] > array[i + 1]) {

int store = array[i];

array[i] = array[i + 1];

array[i + 1] =store;

}

} else {

if (array[i] < array[i + 1]) {

int store = array[i];

array[i] = array[i + 1];

array[i + 1] = store;

}

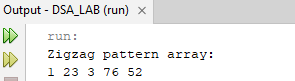
}

}

}

}

**Output:**

****