Java Assignment 3

Akshay Janrao

PRN: 250240520006

Array coding question:

1. Find the Largest and Smallest Element

○ Given an array, find the smallest and largest elements in it.

Answer

```
import java.util.*;

        public class Arr1{
        public static void main(String args[]){

        Scanner scn = new Scanner(System.in);

        System.out.println("Enter size of array");
        int n = scn.nextInt();

        int[] arr = new int[n];

System.out.println("Enter the elements");
        for (int i = 0; i < n ; i++){
        arr[i] = scn.nextInt();
}
        int largest = arr[0];
        int smallest = arr[0];

        for (int i = 1; i < n; i++){
         if(arr[i] > largest){
```

```
largest = arr[i];
}
if (arr[i] < smallest){
smallest = arr[i];
}
}
                System.out.println(" largest no "+largest);
                System.out.println("Smallest no "+smallest);
        }
}
```

2. Reverse an Array

○ Reverse the given array in place.

Answer

```
import java.util.*;

        public class Arr2{
                public static void main(String args[]){
                Scanner scn = new Scanner(System.in);
        System.out.println("Enter the size ");
        int j  = scn.nextInt();

        int[] arr = new int[j];

        System.out.println("Enter the element ");

        for (int i =0 ; i < j; i++){
        arr[i] = scn.nextInt();
}
        int start = 0, end = j-1;
        while (start < end){
        int temp = arr[start];
```

```
        arr[start] = arr[end];
        arr[end] = temp;

        start++;
        end--;

        }
        System.out.println("Reversed");
        for (int num : arr){
        System.out.println(num + " ");
        }
        }
        }
```

3. Find the Second Largest Element

○ Find the second-largest element in the given array.

Answer

```
public class Arr3 {
    public static void main(String[] args) {
        int[] array = {10, 82, 4, 45, 99};

        if (array.length < 2) {
            System.out.println("Array should have at least two elements");
            return;
        }

        int max1 = Integer.MIN_VALUE;
        int max2 = Integer.MIN_VALUE;

        for (int num : array) {
            if (num > max1) {
```

```
            max2 = max1;
            max1 = num;
        } else if (num > max2 && num < max1) {
            max2 = num;
        }
    }

    if (max2 == Integer.MIN_VALUE) {
        System.out.println("No second largest element (all elements might be equal)");
    } else {
        System.out.println("Second largest element is: " + max2);
    }
  }
}
```

4. Count Even and Odd Numbers

○ Count the number of even and odd numbers in an array.

```
public class Arr4{
  public static void main(String[] args) {
    int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int evenCount = 0;
    int oddCount = 0;

    for (int num : array) {
        if (num % 2 == 0) {
            evenCount++;
        } else {
            oddCount++;
        }
    }

    System.out.println("Even numbers: " + evenCount);
```

```
        System.out.println("Odd numbers: " + oddCount);
    }
}
```

5. Find Sum and Average

○ Compute the sum and average of all elements in the array.

```
public class Arr5{
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 54, 6, 87, 8, 78};
        int sum = 0;

        for (int num : array) {
            sum += num;
        }

        double average = (double) sum / array.length;

        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
    }
}
```

6. Remove Duplicates from a Sorted Array

○ Remove duplicate elements from a sorted array without using extra space.

```
public class Arr6{
    public static void main(String[] args) {
        int[] array = {1, 1, 2, 2, 3, 4, 4, 5}; // Sorted array
        int length = removeDuplicates(array);

        System.out.println("Array after removing duplicates:");
        for (int i = 0; i < length; i++) {
            System.out.print(array[i] + " ");
        }
```

```
    }

    public static int removeDuplicates(int[] array) {
        if (array.length == 0) return 0;

        int uniqueIndex = 0;

        for (int i = 1; i < array.length; i++) {
            if (array[i] != array[uniqueIndex]) {
                uniqueIndex++;
                array[uniqueIndex] = array[i];
            }
        }

        return uniqueIndex + 1; // Length of the unique elements
    }
}
```

7. Rotate an Array

○ Rotate the array to the right by k positions.

```
public class Arr7 {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5, 6, 7};
        int k = 3;
        rotate(array, k);

        System.out.println("Rotated array:");
        for (int num : array) {
            System.out.print(num + " ");
        }
    }
```

```
    public static void rotate(int[] array, int k) {
        int n = array.length;
        k = k % n;

        reverse(array, 0, n - 1);
        reverse(array, 0, k - 1);
        reverse(array, k, n - 1);
    }

    private static void reverse(int[] array, int start, int end) {
        while (start < end) {
            int temp = array[start];
            array[start] = array[end];
            array[end] = temp;
            start++;
            end--;
        }
    }
}
```

8. Merge Two Sorted Arrays

○ Merge two sorted arrays into a single sorted array without using extra space.

```
public class Arr8{
    public static void main(String[] args) {
        int[] array1 = {1, 3, 5, 7}; // First sorted array
        int[] array2 = {2, 4, 6, 8}; // Second sorted array
        merge(array1, array2);
```

```
    System.out.print("Merged array1: ");
    for (int num : array1) {
        System.out.print(num + " ");
    }

    System.out.print("\nMerged array2: ");
    for (int num : array2) {
        System.out.print(num + " ");
    }
}

public static void merge(int[] array1, int[] array2) {
    int m = array1.length;
    int n = array2.length;

    for (int i = 0; i < m; i++) {
        if (array1[i] > array2[0]) {
            int temp = array1[i];
            array1[i] = array2[0];
            array2[0] = temp;



            int first = array2[0];
            int k;
            for (k = 1; k < n && array2[k] < first; k++) {
                array2[k - 1] = array2[k];
            }
            array2[k - 1] = first;
        }
    }
}}
```

9. Find Missing Number in an Array

○ Given an array of size n-1 containing numbers from 1 to n, find the missing number.

```
public class Arr9{
   public static void main(String[] args) {
      int[] array = {1, 2, 4, 5, 6};
      int n = array.length + 1;)

      int totalSum = n * (n + 1) / 2;
      int arraySum = 0;

      for (int num : array) {
         arraySum += num;
    }

      int missingNumber = totalSum - arraySum;
      System.out.println("The missing number is: " + missingNumber);
    }
}
```

10. Find Intersection and Union of Two Arrays

Find the intersection and union of two unsorted arrays.

```
import java.util.*;


public class Arr10 {

   public static int[] findIntersection(int[] arr1, int[] arr2) {
      Set<Integer> set1 = new HashSet<>();
      Set<Integer> intersectionSet = new HashSet<>();
```

```
    for (int num : arr1) {
        set1.add(num);
    }
    for (int num : arr2) {
        if (set1.contains(num)) {
            intersectionSet.add(num);
        }
    }
    int[] intersectionArray = new int[intersectionSet.size()];
    int index = 0;
    for (int num : intersectionSet) {
        intersectionArray[index++] = num;
    }

    return intersectionArray;
}

public static int[] findUnion(int[] arr1, int[] arr2) {
    Set<Integer> unionSet = new HashSet<>();

    for (int num : arr1) {
        unionSet.add(num);
    }


    for (int num : arr2) {
        unionSet.add(num);
    }

    int[] unionArray = new int[unionSet.size()];
    int index = 0;
```

```
        for (int num : unionSet) {
            unionArray[index++] = num;
        }

        return unionArray;
    }

    public static void main(String[] args) {
        int[] arr1 = {7, 1, 5, 2, 3, 6};
        int[] arr2 = {3, 8, 6, 20, 7};

        int[] intersection = findIntersection(arr1, arr2);
        int[] union = findUnion(arr1, arr2);

        System.out.println("Intersection: " + Arrays.toString(intersection));
        System.out.println("Union: " + Arrays.toString(union));
    }
}
```

11. Find a Subarray with Given Sum

○ Given an array of integers, find the subarray that sums to a given value S

```
import java.util.*;

public class SubarrayWithGivenSum {
    public static int[] findSubarrayWithSum(int[] arr, int S) {
        int start = 0, currentSum = 0;

        for (int end = 0; end < arr.length; end++) {
            currentSum += arr[end];
```

```
        while (currentSum > S && start <= end) {
            currentSum -= arr[start];
            start++;
        }

        if (currentSum == S) {
            return Arrays.copyOfRange(arr, start, end + 1);
        }
    }

    return null;
  }

  public static void main(String[] args) {
    int[] arr = {1, 2, 3, 7, 5};
    int S = 12;

    int[] result = findSubarrayWithSum(arr, S);
    if (result != null) {
      System.out.println("Subarray with the given sum: " + Arrays.toString(result));
    } else {
      System.out.println("No subarray with the given sum was found.");
    }
  }
}
```

12. Write a program to accept 20 integer numbers in a single Dimensional Array. Find and Display the following:

○ Number of even numbers.

○ Number of odd numbers.

○ Number of multiples of 3

```
import java.util.Scanner;

public class Arr12 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] numbers = new int[20];
        int evenCount = 0, oddCount = 0, multipleOf3Count = 0;

        System.out.println("Enter 20 integers:");

        for (int i = 0; i < numbers.length; i++) {
            numbers[i] = scanner.nextInt();


            if (numbers[i] % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }

            if (numbers[i] % 3 == 0) {
                multipleOf3Count++;
            }
        }

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

```
        System.out.println("Number of odd numbers: " + oddCount);
        System.out.println("Number of multiples of 3: " + multipleOf3Count);
    }
}
```

13. Write a program to accept the marks in Physics, Chemistry and Maths secured by 20 class students in a single Dimensional Array. Find and display the following:

○ Number of students securing 75% and above in aggregate.

○ Number of students securing 40% and below in aggregate.

```
import java.util.Scanner;

public class Arr13 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] marks = new int[20][3];
        int above75Count = 0;
        int below40Count = 0;

        System.out.println("Enter marks of 20 students in Physics, Chemistry, and Maths:");


        for (int i = 0; i < 20; i++) {
            System.out.println("Enter marks for student " + (i + 1) + " (Physics, Chemistry, Maths):");
            for (int j = 0; j < 3; j++) {
                marks[i][j] = scanner.nextInt();
            }
        }


        for (int i = 0; i < 20; i++) {
            int total = marks[i][0] + marks[i][1] + marks[i][2];
```

```
            double aggregatePercentage = (total / 300.0) * 100;

            if (aggregatePercentage >= 75) {
                above75Count++;
            } else if (aggregatePercentage <= 40) {
                below40Count++;
            }
        }

        System.out.println("Number of students securing 75% and above in aggregate: " +
above75Count);
        System.out.println("Number of students securing 40% and below in aggregate: " +
below40Count);
    }
}
```

14. Write a program in Java to accept 20 numbers in a single dimensional array arr[20]. Transfer and store all the even numbers in an array even [ ] and all the odd numbers in another array odd []. Finally, print the elements of the even & the odd array.

```
import java.util.*;

public class Arr14 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] arr = new int[20];
        ArrayList<Integer> even = new ArrayList<>();
        ArrayList<Integer> odd = new ArrayList<>();

        System.out.println("Enter 20 integers:");

        for (int i = 0; i < arr.length; i++) {
            arr[i] = scanner.nextInt();
            if (arr[i] % 2 == 0) {
```

```
            even.add(arr[i]);
        } else {
            odd.add(arr[i]);
        }
    }

    Integer[] evenArray = even.toArray(new Integer[0]);
    Integer[] oddArray = odd.toArray(new Integer[0]);

    System.out.println("Even numbers:");
    for (int num : evenArray) {
        System.out.print(num + " ");
    }
    System.out.println();

    System.out.println("Odd numbers:");
    for (int num : oddArray) {
        System.out.print(num + " ");
    }
    System.out.println();

    scanner.close();
  }
}
```

15. Write a Java program to print all sub-arrays with 0 sum present in a given array of integers. array with Given Sum

Example

Input:

nums1 = {1, 3, -7, 3, 2, 3, 1, -3, -2, -2}

nums2 = {1, 2, -3, 4, 5, 6}

nums3= {1, 2, -2, 3, 4, 5, 6}

Output:

Sub-arrays with 0 sum: [1, 3, -7, 3]

Sub-arrays with 0 sum: [3, -7, 3, 2, 3, 1, -3, -2]

Sub-arrays with 0 sum: [1, 2, -3]

Sub-arrays with 0 sum: [2, -2]

```
import java.util.*;

public class Arr15 {
   public static void findZeroSumSubarrays(int[] nums) {

      HashMap<Integer, List<Integer>> sumMap = new HashMap<>();
      List<int[]> result = new ArrayList<>();
      int cumulativeSum = 0;

      sumMap.put(0, new ArrayList<>());
      sumMap.get(0).add(-1);

      for (int i = 0; i < nums.length; i++) {
         cumulativeSum += nums[i];



         if (sumMap.containsKey(cumulativeSum)) {
            for (int startIndex : sumMap.get(cumulativeSum)) {
               result.add(new int[]{startIndex + 1, i});
            }
         }

         sumMap.putIfAbsent(cumulativeSum, new ArrayList<>());
         sumMap.get(cumulativeSum).add(i);
      }
```

```
        for (int[] subarray : result) {
            System.out.print("Sub-array with 0 sum: [");
            for (int k = subarray[0]; k <= subarray[1]; k++) {
                System.out.print(nums[k] + (k < subarray[1] ? ", " : ""));
            }
            System.out.println("]");
        }
    }

    public static void main(String[] args) {
        int[] nums1 = {1, 3, -7, 3, 2, 3, 1, -3, -2, -2};
        int[] nums2 = {1, 2, -3, 4, 5, 6};
        int[] nums3 = {1, 2, -2, 3, 4, 5, 6};

        System.out.println("Input Array 1:");
        findZeroSumSubarrays(nums1);

        System.out.println("Input Array 2:");
        findZeroSumSubarrays(nums2);

        System.out.println("Input Array 3:");
        findZeroSumSubarrays(nums3);
    }
}
```

16. Given two sorted arrays A and B of size p and q, write a Java program to merge elements of A with B by maintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements.

Example:

Input :

int[] A = { 1, 5, 6, 7, 8, 10 }

int[] B = { 2, 4, 9 }

Output:

Sorted Arrays:

A: [1, 2, 4, 5, 6, 7]

B: [8, 9, 10]

```
import java.util.Arrays;

public class Arr16 {
    public static void mergeArrays(int[] A, int[] B) {
        int p = A.length;
        int q = B.length;
        int[] combined = new int[p + q];
        System.arraycopy(A, 0, combined, 0, p);
        System.arraycopy(B, 0, combined, p, q);

        Arrays.sort(combined);

        for (int i = 0; i < p; i++) {
            A[i] = combined[i];
        }

        for (int i = 0; i < q; i++) {
            B[i] = combined[p + i];
        }
    }

    public static void main(String[] args) {
        int[] A = {1, 5, 6, 7, 8, 10};
        int[] B = {2, 4, 9};

        System.out.println("Original A: " + Arrays.toString(A));
```

```
        System.out.println("Original B: " + Arrays.toString(B));

        mergeArrays(A, B);

        System.out.println("Sorted Arrays:");
        System.out.println("A: " + Arrays.toString(A));
        System.out.println("B: " + Arrays.toString(B));
    }
}
```

17. Write a Java program to find the maximum product of two integers in a given array of integers.

Example:

Input :

nums = { 2, 3, 5, 7, -7, 5, 8, -5 }

Output:
Pair is (7,8) Maximum product:56

```
public class Arr17 {
    public static void main(String[] args) {
        int[] nums = {2, 3, 5, 7, -7, 5, 8, -5};

        // Find the maximum product and the pair
        int maxProduct = Integer.MIN_VALUE;
        int num1 = 0, num2 = 0; // To store the pair of numbers

        for (int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                int product = nums[i] * nums[j];
                if (product > maxProduct) {
                    maxProduct = product;
                    num1 = nums[i];
                    num2 = nums[j];
```

```
            }
        }
    }

    // Output the result
    System.out.println("Pair is (" + num1 + ", " + num2 + ")");
    System.out.println("Maximum product: " + maxProduct);
  }
}
```

18. Print a Matrix

○ Given an m x n matrix, print all its elements row-wise.

```
public class Arr18 {
  public static void printMatrixRowWise(int[][] matrix) {
    int rows = matrix.length;
    int columns = matrix[0].length;

    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < columns; j++) {
        System.out.print(matrix[i][j] + " ");
      }
      System.out.println();
    }
  }

  public static void main(String[] args) {
    int[][] matrix = {
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
    };
```

```
        System.out.println("Matrix elements row-wise:");
        printMatrixRowWise(matrix);
    }
}
```

19. Transpose of a Matrix

○ Given a matrix, return its transpose (swap rows and columns).

```
public classArr19 {
    public static int[][] transposeMatrix(int[][] matrix) {
        int rows = matrix.length;
        int columns = matrix[0].length;

        int[][] transpose = new int[columns][rows];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }

        return transpose;
    }

    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int elem : row) {
                System.out.print(elem + " ");
            }
            System.out.println();
        }
```

```
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("Original Matrix:");
        printMatrix(matrix);

        int[][] transposed = transposeMatrix(matrix);

        System.out.println("Transposed Matrix:");
        printMatrix(transposed);
    }
}
```

20. Sum of Two Matrices

○ Given two matrices of the same size, compute their sum.

```
public class Arr20{
    public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
        int rows = matrix1.length;
        int columns = matrix1[0].length;
```

```
    int[][] result = new int[rows][columns];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }

    return result;
}

public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int elem : row) {
            System.out.print(elem + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] matrix1 = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int[][] matrix2 = {
        {9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}
```

```
        };

        System.out.println("Matrix 1:");
        printMatrix(matrix1);

        System.out.println("Matrix 2:");
        printMatrix(matrix2);

        int[][] result = addMatrices(matrix1, matrix2);

        System.out.println("Sum of the matrices:");
        printMatrix(result);
    }
}
```

21. Row-wise and Column-wise Sum

○ Find the sum of each row and each column of a given matrix.

```
public class Arr21 {
    public static void findRowAndColumnSums(int[][] matrix) {
        int rows = matrix.length;
        int columns = matrix[0].length;

        System.out.println("Row-wise sums:");
        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < columns; j++) {
                rowSum += matrix[i][j];
            }
```

```
            System.out.println("Row " + (i + 1) + ": " + rowSum);
        }

        System.out.println("Column-wise sums:");
        for (int j = 0; j < columns; j++) {
            int columnSum = 0;
            for (int i = 0; i < rows; i++) {
                columnSum += matrix[i][j];
            }
            System.out.println("Column " + (j + 1) + ": " + columnSum);
        }
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        findRowAndColumnSums(matrix);
    }
}
```

22. Find the Maximum Element in a Matrix

○ Find the largest element in a given matrix.

```
public class Arr22{
    public static int findMaximumElement(int [][] matrix) {
        int max = Integer.MIN_VALUE;

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

```
            for (int j = 0; j < matrix[i].length; j++) {
                if (matrix[i][j] > max) {
                    max = matrix[i][j];
                }
            }
        }

        return max;
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("The maximum element in the matrix is: " + findMaximumElement(matrix));
    }
}
```

23. Matrix Multiplication

○ Multiply two matrices and return the resultant matrix.

```
public class Arr23 {
    public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {
        int m = matrix1.length;
        int n = matrix1[0].length;
        int p = matrix2[0].length;



        int[][] result = new int[m][p];
```

```
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            for (int k = 0; k < n; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }

    return result;
}

public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int elem : row) {
            System.out.print(elem + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] matrix1 = {
        {1, 2, 3},
        {4, 5, 6}
    };

    int[][] matrix2 = {
        {7, 8},
        {9, 10},
```

```
            {11, 12}
        };

        System.out.println("Matrix 1:");
        printMatrix(matrix1);

        System.out.println("Matrix 2:");
        printMatrix(matrix2);

        int[][] result = multiplyMatrices(matrix1, matrix2);

        System.out.println("Resultant Matrix:");
        printMatrix(result);
    }
}
```

24. Rotate a Matrix by 90 Degrees

Rotate a given N x N matrix by 90 degrees clockwise.

```
public class Arr24{
    public static void rotate90Clockwise(int[][] matrix) {
        int n = matrix.length;



        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n / 2; j++) {
            int temp = matrix[i][j];
            matrix[i][j] = matrix[i][n - 1 - j];
            matrix[i][n - 1 - j] = temp;
        }
    }
}

public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int elem : row) {
            System.out.print(elem + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] matrix = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    System.out.println("Original Matrix:");
    printMatrix(matrix);

    rotate90Clockwise(matrix);
```

```
        System.out.println("Matrix after clockwise rotation:");
        printMatrix(matrix);
    }
}
```

25. Find the Diagonal Sum

Compute the sum of both diagonals in a square matrix.

```
public class Arr25 {
    public static int findDiagonalSum(int[][] matrix) {
        int n = matrix.length;
        int sum = 0;

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

            sum += matrix[i][i];

            if (i != n - 1 - i) {
                sum += matrix[i][n - 1 - i];
            }
        }

        return sum;
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 56, 6},
            {7, 8, 9}
        };
```

```
        System.out.println("Sum of diagonals: " + findDiagonalSum(matrix));
    }
}
```