|  |  |
| --- | --- |
| Student Name | Uzair Hussain |
| Roll Number | 21SW085 |
| Section # | 3rd or III |
| Lab # | 4th – Sorting in Arrays |

**Task#01**

**Code:**

import java.util.Arrays;

class Sorts{

    // Bubble sort of 1D Array

    public int[] Bubble\_Sort(int [] arr){

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

            for (int j=0; j< arr.length-1; j++){

                if (arr[j]>arr[j+1]){

                    int temp = arr[j];

                    arr[j] = arr[j+1];

                    arr[j+1] = temp;

                }

            }

        }

        return arr;

    }

    // Insertion Sort of 1D Array

    public int[] Insertion\_Sort(int [] arr){

        for (int i=1; i< arr.length; i++){

            int temp = arr[i];

            int j = i-1;

            while (j>=0 && arr[j]>temp){

                arr[j+1] = arr[j];

                j--;

            }

            arr[j+1] = temp;

        }

        return arr;

    }

    //Selction SOrt of 1D Array

    public int[] Selection\_Sort(int [] arr){

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

            int minIndex = i;

            for (int j=i+1; j< arr.length; j++){

                if (arr[j]<arr[minIndex]){

                    minIndex = j;

                }

            }

                int temp = arr[i];

                arr[i] = arr[minIndex];

                arr[minIndex] = temp;

        }

        return arr;

    }

    // Quick Sort

    public void quickSort(int[] array, int low, int high) {

        if (array == null || array.length == 0){

            return;

        }

        if (low >= high){

            return;

        }

        // pick the pivot

        int middle = low + (high - low) / 2;

        int pivot = array[middle];

        // make left < pivot and right > pivot

        int i = low, j = high;

        while (i <= j) {

            while (array[i] < pivot) {

                i++;

            }

            while (array[j] > pivot) {

                j--;

            }

            if (i <= j) {

                int temp = array[i];

                array[i] = array[j];

                array[j] = temp;

                i++;

                j--;

            }

        }

        // recursively sort two sub parts

        if (low < j)

            quickSort(array, low, j);

        if (high > i)

            quickSort(array, i, high);

    }

    public long TimeCalculator(long b, long a){

        return (a-b);

    }

}

public class Task1\_1D\_Sorting {

    public static void main(String[] args) {

        int [] array = {4, -6, 5, 7, 8, 9, 1, 10, -9};

        Sorts s = new Sorts();

        long before1 = System.nanoTime();

        System.out.println("Bubble Sort()  -->  "+Arrays.toString(s.Bubble\_Sort(array)));

        long after1 = System.nanoTime();

        long before2 = System.nanoTime();

        System.out.println("Insertion Sort()  -->   "+Arrays.toString(s.Insertion\_Sort(array)));

        long after2 = System.nanoTime();

        long before3 = System.nanoTime();

        System.out.println("Selection Sort()  -->   "+Arrays.toString(s.Selection\_Sort(array)));

        long after3 = System.nanoTime();

        long before4 = System.nanoTime();

        s.quickSort(array, 0, array.length-1);

        System.out.println("Quick Sort()  -->   "+Arrays.toString(array));

        long after4 = System.nanoTime();

        System.out.println("\n\nExecution time of Bubble Sort() method is -->   "+s.TimeCalculator(before1, after1));

        System.out.println("Execution time of Insertion Sort() method is -->   "+s.TimeCalculator(before2, after2));

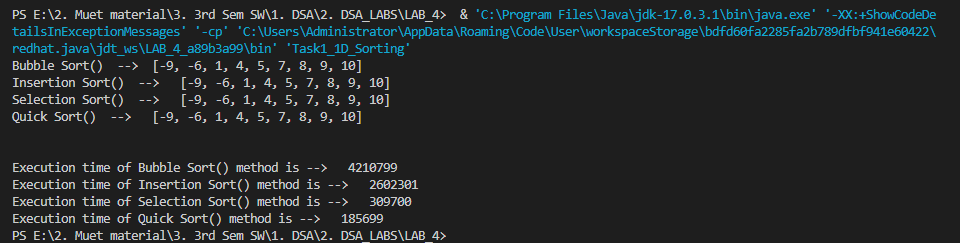
        System.out.println("Execution time of Selection Sort() method is -->   "+s.TimeCalculator(before3, after3));

        System.out.println("Execution time of Quick Sort() method is -->   "+s.TimeCalculator(before4, after4));

    }

}

**Output 1:**

****

**Code 2:**

import java.lang.reflect.Array;

import java.text.CollationElementIterator;

import java.util.Arrays;

import java.util.ArrayList;

import java.util.Collections;

public class Task2\_2D\_Sorting {

    public static int[][] insertionSort(int [][] matrix){

        int [] arr = conversionIn1D(matrix);

        for (int i=1; i< arr.length; i++){

            int temp = arr[i];

            int j = i-1;

            while (j>=0 && arr[j]>temp){

                arr[j+1] = arr[j];

                j--;

            }

            arr[j+1] = temp;

        }

        return conversionIn2D(arr, matrix.length, matrix[0].length);

    }        // end of insertionSort() method

    public static int[][] selectionSort(int [][] matrix){

        int [] arr = conversionIn1D(matrix);

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

            int min = i;

            for (int j=i+1; j< arr.length; j++){

                if (arr[j]<arr[min]){

                    min = j;

                }

            }

            if (i!=min){

                int temp = arr[i];

                arr[i] = arr[min];

                arr[min] = temp;

            }

        }

        return conversionIn2D(arr, matrix.length, matrix[0].length);

    }     // end of selectionSort() method

    public static int[] conversionIn1D(int [][] matrix){

        ArrayList<Integer> arrayList = new ArrayList<>();

        int k=0;

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

            for (int j=0; j< matrix[i].length; j++) {

                arrayList.add(k, matrix[i][j]);

                k++;

            }

        }

        int [] array = new int[arrayList.size()];

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

            array[i] = arrayList.get(i);

        }

        return array;

    }    // end of conversionIn1D() method

    public static int[][] conversionIn2D(int [] array, int rows, int cols){

        int[][] matrix = new int[rows][cols];

        int k = 0;

        for (int i=0; i< rows; i++){

            for (int j=0; j< cols; j++){

                matrix[i][j] = array[k];

                k++;

            }

        }

        return matrix;

    }   // end of conversionIn2D() method

    public static int[][] bubbleSort(int [][] matrix){

        int [] array = conversionIn1D(matrix);

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

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

                if (array[j]<array[i]){

                    int temp = array[i];

                    array[i] = array[j];

                    array[j] = temp;

                }

            }

        }

        return conversionIn2D(array, matrix.length, matrix[0].length);

    }

    public static int partition(int [] a, int lowerbound, int upperbound){

        int pivot = a[upperbound];

        int i = lowerbound-1;

        for (int j=lowerbound; j<upperbound; j++){

            if(a[j]<=pivot){

                i++;

                int temp = a[i];

                a[i] = a[j];

                a[j] = temp;

            }

        }

        int temp = a[i+1];

        a[i+1] = a[upperbound];

        a[upperbound] = temp;

        return (i+1);

    }

    public static void quickSort(int [] a, int lowerbound, int upperbound){

        if(lowerbound<upperbound){

            int location = partition(a, lowerbound, upperbound);

            quickSort(a, lowerbound, location-1);

            quickSort(a, location+1, upperbound);

        }

    }

    public static void main(String[] args) {

        int [][] matrix = {{2, 10, 15}, {5, 1, 3}, {6, 9, 4}};

        System.out.println("Original Matrix:    "+Arrays.deepToString(matrix));

        System.out.println("\n\tSorted Matrices.\n");

        System.out.println("Bubble Sort:    "+Arrays.deepToString(bubbleSort(matrix)));

        System.out.println("Selection Sort:    "+Arrays.deepToString(selectionSort(matrix)));

        System.out.println("Insertion Sort:    "+Arrays.deepToString(insertionSort(matrix)));

        int [] array = conversionIn1D(matrix);

        quickSort(array, 0, array.length-1);

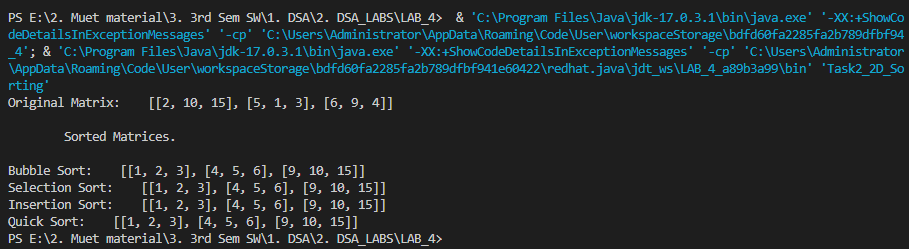
        matrix  = conversionIn2D(array, matrix.length, matrix[0].length);

        System.out.println("Quick Sort:    "+Arrays.deepToString(matrix));

    }   //  end of main() method

}      //  end of program

**Output 2:**

****

**The End!**