Design And Analysis Of Algorithms

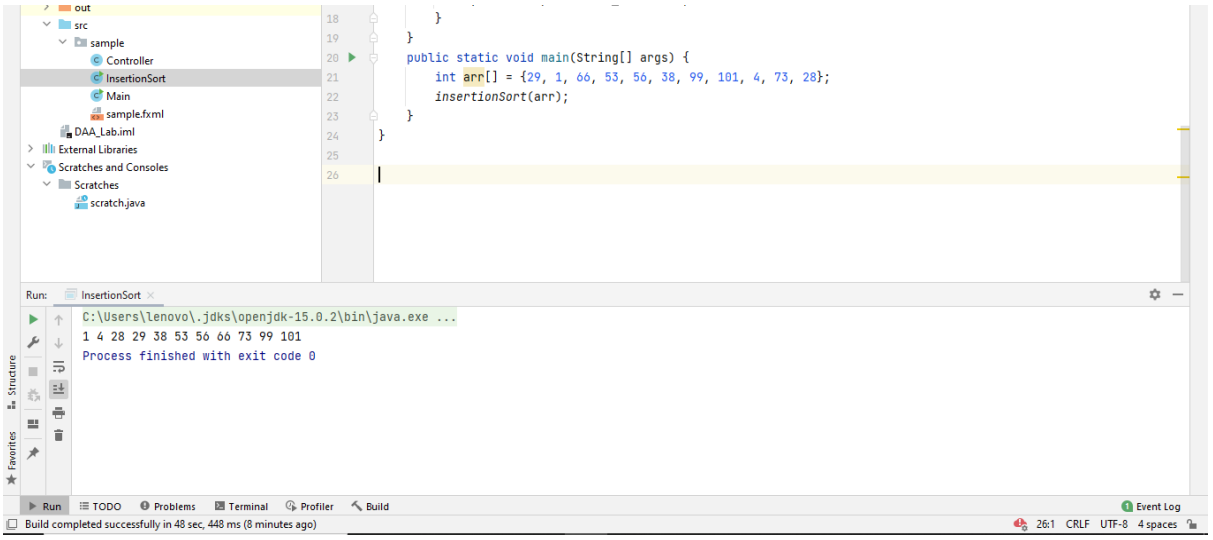
Practical- 1

Objective:-Implementation and analysis of Insertion Sort.

Code:-

```
package sample;
public class InsertionSort {
    public static void insertionSort(int[] arr){
        int temp;
        int n= arr.length;
        int j;
        for (int i = 1; i <n; i++) {
            temp=arr[i];
            j=i-1;
            while (j>=0 && arr[j]>temp){
                arr[j+1]=arr[j];
                j--;
            }
            arr[j+1]=temp;
        }
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] +  " ");
        }
    }
    public static void main(String[] args) {
        int arr[] = {29, 1, 66, 53, 56, 38, 99, 101, 4, 73, 28};
        insertionSort(arr);
    }
}
```

Output:-



Analysis:-

It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of insertion sort is **O(n)**.

When the array elements are in jumbled order that is not properly ascending and not properly descending. The average case and Worst Case time complexity of insertion sort is **$O(n^2)$**.

Practical- 1

Objective:-Implementation and analysis of **Bubble** Sort.

Code:-

```
package sample;
public class BubbleSort {
   public static void bubbleSort(int[] arr){
        int temp, j;
        int n=arr.length;
        for (int i = 0; i < n-1; i++) {
            for (j=0; j<n-1-i; j++){
                if (arr[j]>arr[j+1]){
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i]+ " ");
        }
    }
   public static void main(String[] args) {
        int arr[]={44,12,99,56,19,40,69,35,27};
        bubbleSort(arr);
    }
}
```

Output:-

Analysis:-

It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of Bubble sort is O(n).

Average and Worst case occurs when the array is reverse sorted.$O(n^2)$.