Task#3

Sorting Algorithm

Bubble sort:

```
 */
public class Bubble_sort_algo {

    public static void main(String[] args) {
        int arr[]={6,4,5,1,8};
        int temp;
        int n=arr.length;
        System.out.print( s:"Array before sorting:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
        for (int i = 0; i < n-1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx]){
                    min_idx = j;
                }
            temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
        System.out.println( x:"");
        System.out.print( s:"Array after Bubble sort:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }

    }
}
```

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Bubble_sort_algo ---
Array before sorting:   6 4 5 1 8
Array after Bubble sort:   1 4 5 6 8
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.456 s
Finished at: 2023-07-11T15:07:54+05:00
------------------------------------------------------------------------
```

Selection sort:

```
*/
public class Selection sort Algo {

    public static void main(String[] args) {
        int arr[]={6,4,5,1,8};
        int temp;
        System.out.print( s:"Array before sorting:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
        for(int i=0; i<arr.length;i++){

            for(int j=0;j>i;j--){
                if(arr[i]>arr[j]){
                    temp=arr[j];
                    arr[j]=arr[i];
                    arr[i]=temp;
                }
            }
        //if(swape==false){
        //    break;
        //}
        }
        System.out.println( x:"");
        System.out.print( s:"Array after Selection sort:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }

    }
}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ selection_sort_Algo ---
Array before sorting:   6 4 5 1 8
Array after Selection sort:   6 4 5 1 8
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.211 s
Finished at: 2023-07-11T15:10:26+05:00
------------------------------------------------------------------------
```

Insertion sort:

```
*/
public class Insertion_sort_Algo {
    public static void sort_Algo(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j;
            for(j=i-1;j >= 0 && arr[j] > key;j--) {
                arr[j + 1] = arr[j];
            }
            arr[j + 1] = key;
        }
    }


    public static void main(String[] args) {
        int arr[]={6,4,5,1,8};
        System.out.print( s:"Array before sorting:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
        sort_Algo(arr);
        System.out.println( x:"");
        System.out.print( s:"Array after Insertion sort:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
    }
}
```

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Insertion_sort_Algo ---
Array before sorting:   6 4 5 1 8
Array after Insertion sort:   1 4 5 6 8
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.607 s
Finished at: 2023-07-11T15:12:23+05:00
------------------------------------------------------------------------
```

Heap sort:

```
public class Heap_sort_algo {
    public static void heap_tree(int arr[], int N, int i)
    {
        int largest = i;
        int l = 2 * i + 1;
        int r = 2 * i + 2;

        if (l < N && arr[l] > arr[largest])
            largest = l;

        if (r < N && arr[r] > arr[largest])
            largest = r;

        if (largest != i) {
            int swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;
            heap_tree(arr, N, i:largest);
        }

    }
    public static void heap_sort(int arr[])
    {
        int n = arr.length;

        for (int i = n / 2 - 1; i >= 0; i--)
            heap_tree(arr, N:n, i);

        for (int i = n - 1; i > 0; i--) {
            // Move current root to end
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;
```

```
    }
    public static void heap_sort(int arr[])
    {
        int n = arr.length;

        for (int i = n / 2 - 1; i >= 0; i--)
            heap_tree(arr, N:n, i);

        for (int i = n - 1; i > 0; i--) {
            // Move current root to end
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            heap_tree(arr, N:i, i:0);
        }
    }
    public static void main(String[] args) {
        int arr[]={6,4,5,1,8};
        System.out.print(s:"Array before sorting:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
        heap_sort(arr);
        System.out.println(x:"");
        System.out.print(s:"Array after Heap sort:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
    }
}
```

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ heap_sort_algo ---
Array before sorting:   6 4 5 1 8
Array after Heap sort:   1 4 5 6 8
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.113 s
Finished at: 2023-07-11T15:15:07+05:00
------------------------------------------------------------------------
```

Merge sort:

```java
public class Merge_sort_algo {
    public static void sort_array(int arr[], int l, int r){
        if (l < r) {
            int m = l + (r - l) / 2;
            sort_array(arr, l, r:m);
            sort_array(arr, m + 1, r);
            merge_array(arr, l, m, r);
        }
    }
    public static void merge_array(int arr[], int l, int m, int r)
    {
        // Find sizes of two subarrays to be merged
        int n1 = m - l + 1;
        int n2 = r - m;

        // Create temp arrays
        int Left[] = new int[n1];
        int Right[] = new int[n2];

        // Copy data to temp arrays
        for (int i = 0; i < n1; ++i)
            Left[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            Right[j] = arr[m + 1 + j];
        int i = 0, j = 0;
        int k = l;
        while (i < n1 && j < n2) {
            if (Left[i] <= Right[j]) {
                arr[k] = Left[i];
                i++;
            }
            else {
                arr[k] = Right[j];
                j++;
            }
```

```
            else {
                arr[k] = Right[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = Left[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = Right[j];
            j++;
            k++;
        }
    }

    public static void main(String[] args) {
        int arr[]={6,4,5,1,8};
        System.out.print( s:"Array before sorting:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
        sort_array(arr, l:0,arr.length-1);
        System.out.println( x:"");
        System.out.print( s:"Array after Heap sort:  ");
        for(int i=0;i<arr.length;i++){
            System.out.print(" "+arr[i]);
        }
}
}
```

```
-------------------< com.mycompany:Merge_sort_algo >-------------------
Building Merge_sort_algo 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Merge_sort_algo ---
Array before sorting:  6 4 5 1 8
Array after Heap sort:  1 4 5 6 8
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.485 s
Finished at: 2023-07-11T15:17:37+05:00
------------------------------------------------------------------------
```