BubbleSort

```java
public class BubbleSort {
    public static void bubbleSort(int[] arr){
        for(int i = 0; i < arr.length; i++){
            for(int j = 0; j < arr.length - i - 1; j++){
                if(arr[j] > arr[j + 1]){
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
    public static void main(String[] args) {
        int[] nums = {3, 2, 4, 1, 5};
        bubbleSort(nums);

        // printing Array after Soring
        for(int values: nums){
            System.out.print(values + " ");
        }
    }
}

```

InsertionSortProgram

```java
// Write a program to sort an Array using Insertion sort.

Codeium: Refactor | Explain
public class InsertionSortProgram {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void insertionSort(int[] arr){
        for(int i = 1; i < arr.length; i++){
            int current = arr[i];
            int j = i - 1;
            while(j >= 0 && current < arr[j]){
                arr[j + 1] = arr[j];
                j--;
            }
            //Placement
            arr[j + 1] = current;
        }
    }
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        int[] arr = {7, 8, 3, 2, 1};
        insertionSort(arr);
        System.out.println(Arrays.toString(arr));
    }
}

```

MergeSortAlogrithms

```java
public class MergeSortAlogrithms {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void merge(int[] arr, int start, int end){
        // Calculating mid element
        int mid = start + (end - start)/2;

        // creating a temp Array
        int[] temp = new int[end - start + 1];
        int i = start;
        int j = mid + 1;
        int k = 0;
        while(i <= mid && j <= end){
            if(arr[i] <= arr[j]){
                temp[k++] = arr[i++];
            }
            else{
                temp[k++] = arr[j++];
            }
        }
        while(i <= mid){
            temp[k++] = arr[i++];
        }
        while(j <= end){
            temp[k++] = arr[j++];
        }

        // Copying all Elements of temp into Original Array.
        for(int m = 0, n = start; m < temp.length; m++, n++){
            arr[n] = temp[m];
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void mergeSort(int[] arr, int start, int end){
        if(start >= end){
            return;
        }
        int mid = start + (end - start)/2;
        mergeSort(arr, start, mid);
        mergeSort(arr, mid + 1, end);
        merge(arr, start, end);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        int arr[] = { 41, 1, 32, 95, 7 };
        mergeSort(arr, 0, arr.length - 1);

        // Printing the array
        System.out.println(Arrays.toString(arr));
    }
}
```

QuickSortAlgorithms

```java
public class QuickSortAlgorithms {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static int partition(int[] arr, int low, int high){
        int pivot = arr[low];
        int i = low;
        int j = high;
        while(i < j){
            while(arr[i] <= pivot && i <= high - 1){
                i++;
            }
            while(arr[j] > pivot && j >= low){
                j--;
            }
            // Swapping ith index and jth index.
            if(i < j){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        // Swapping high with the pivot element.
        int temp = arr[low];
        arr[low] = arr[j];
        arr[j] = temp;

        return j;
    }
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void quicksort(int[] arr, int low, int high){
        if(low < high){
            int pivot = partition(arr, low, high);
            quicksort(arr, low, pivot - 1);
            quicksort(arr, pivot + 1, high);
        }
    }
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        int n = 8;
        int arr[] = { 4, 6, 2, 5, 7, 8, 1, 3 };
        System.out.println("Before Quick Sort: ");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
        quicksort(arr, 0, n - 1);
        System.out.println("After Quick Sort: ");
        for(int i = 0; i < n; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

BubbleSortUsingDepthOfRecursion

```java
public class BubbleSortUsingDepthOfRecursion {
    // This method is used to print the Array.
    public static void print(int[] arr){
        for(int i = 0; i < arr.length; i++){
            System.out.print(arr[i] + " ");
        }
    }

    // This is the inner recursion of the bubble sort. i.e. Recursion inside Recursion
    public static void bubbleSortDepth(int[] arr, int i, int j){
        if(j == arr.length - i - 1){
            return;
        }
        if(arr[j] > arr[j + 1]){
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
        bubbleSortDepth(arr, i, j + 1);
    }

    // This is the outer Recursion of Bubble Sort
    public static void bubbleSort(int[] arr, int i, int j){
        if(i == arr.length - 1){
            return;
        }
        bubbleSortDepth(arr, i, j);
        bubbleSort(arr, i + 1, 0);
    }

    // Main Method
    public static void main(String[] args) {
        int[] arr = {5, 3, 6, 1, 8, 2, 4};
        bubbleSort(arr, 0, 0);
        print(arr);
    }
}

```

RecursiveInsertionSort

```java
public class RecursiveInsertionSort {
    public static void insertionSort(int[] arr, int n, int i){
        if(n == 0){
            return;
        }
        int current = arr[i];
        int j = i - 1;
        while(j >= 0 && current < arr[j]){
            arr[j + 1] = arr[j];
            j--;
        }
        // placement
        arr[j + 1] = current;
        insertionSort(arr, n - 1, i + 1);
    }
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        int n = 8;
        int arr[] = { 4, 6, 2, 5, 7, 8, 1, 3 };
        System.out.println("Before Quick Sort: ");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
        insertionSort(arr, arr.length - 1, 1);
        System.out.println("After Quick Sort: ");
        for(int i = 0; i < n; i++){
            System.out.print(arr[i] + " ");
        }
    }
}
```

SelectionSort

```java
public class SelectionSort {
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void selectionSort(int[] arr){
        for(int i = 0; i < arr.length; i++){
            int smallest = i;
            for(int j = i + 1; j < arr.length; j++){
                if(arr[smallest] > arr[j]){
                    smallest = j;
                }
            }
            int temp = arr[i];
            arr[i] = arr[smallest];
            arr[smallest] = temp;
        }
    }
    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        int[] nums = {64, 25, 12, 22, 11};
        selectionSort(nums);

        // printing Array after Soring
        for(int values: nums){
            System.out.print(values + " ");
        }
    }
}

```