**Merge Sort Algorithm**

The Merge sort separate to items and working that more separate until last element when size = 1 and then start up with merge operation this operation called (Merge Sort Recursive)

**For Example**:

**O(n log n)**

**<-**  When size = 1 its meaning start do recursive, it start up and don sorting and then merge same like   
 (Binary Search) because it’s separate two items (divided/ 2 ) so complexity equal (log 2 base 2)

**1**

**2**

**4**

**4**

**8**

**8**

Each all result from (divided / 2) operation contain equal (16 elements) so this meaning you have more elements greater than 16 for all level (4 \* 16) so this meaning complexity equal (O (n log n))

**Merge sort has two methods:**   
1- The First method: Separate Operation to Left and Right side   
2- The Second method: Merge Operation

**1- The First method: Separate Operation:**

Separate (int a [])  
For (int i = 0 ; i < mid ; i++)  
 Left [i] = a[i]  
For (int j = mid ; j < a.lenght ; j++)  
 Right [j-mid] = a[j];

**2- The Second method: Merge Operation:**

Merge (int a[] , L[] , R[])  
While (i < L.lenght && j < R.lenght)  
 if (L[i] <= R[j])  
 int k = 0;  
 a[k] = R[j]  
 else a[k] = L[i]  
 j++;  
k++;

**Merge Sort Algorithm**

Public Class Separate (int a[]){

Int n = a.lenght;  
 if (n==1) return;  
 int mid = n / 2  
 int Left[] = new int [mid];  
 int Right = new int [n-mid];

**<-** It will working to separate until it‘s reach   
 the least element equal is number one

For (int i = 0 ; j < n ; i++ )  
 Left[i] = a[i];  
 For (int j = mid ; j < n ; j++)  
 Right [j-mid] = a[j]

Separate (Left);  
Separate (Right);  
Merge ( a , Left , Right);  
}

Public Class merge (int a[] , int Left [] , int Right[]){  
 int i = 0;  
 int j = 0;  
 int k=0;  
 while (i < Left.lenght && j < Right.lenght)  
 IF (Left[i] <= Right[j])  
 a[k] = Right[j];  
 i ++;  
 else a[k]= Left[i]:  
 j++;  
 k++;

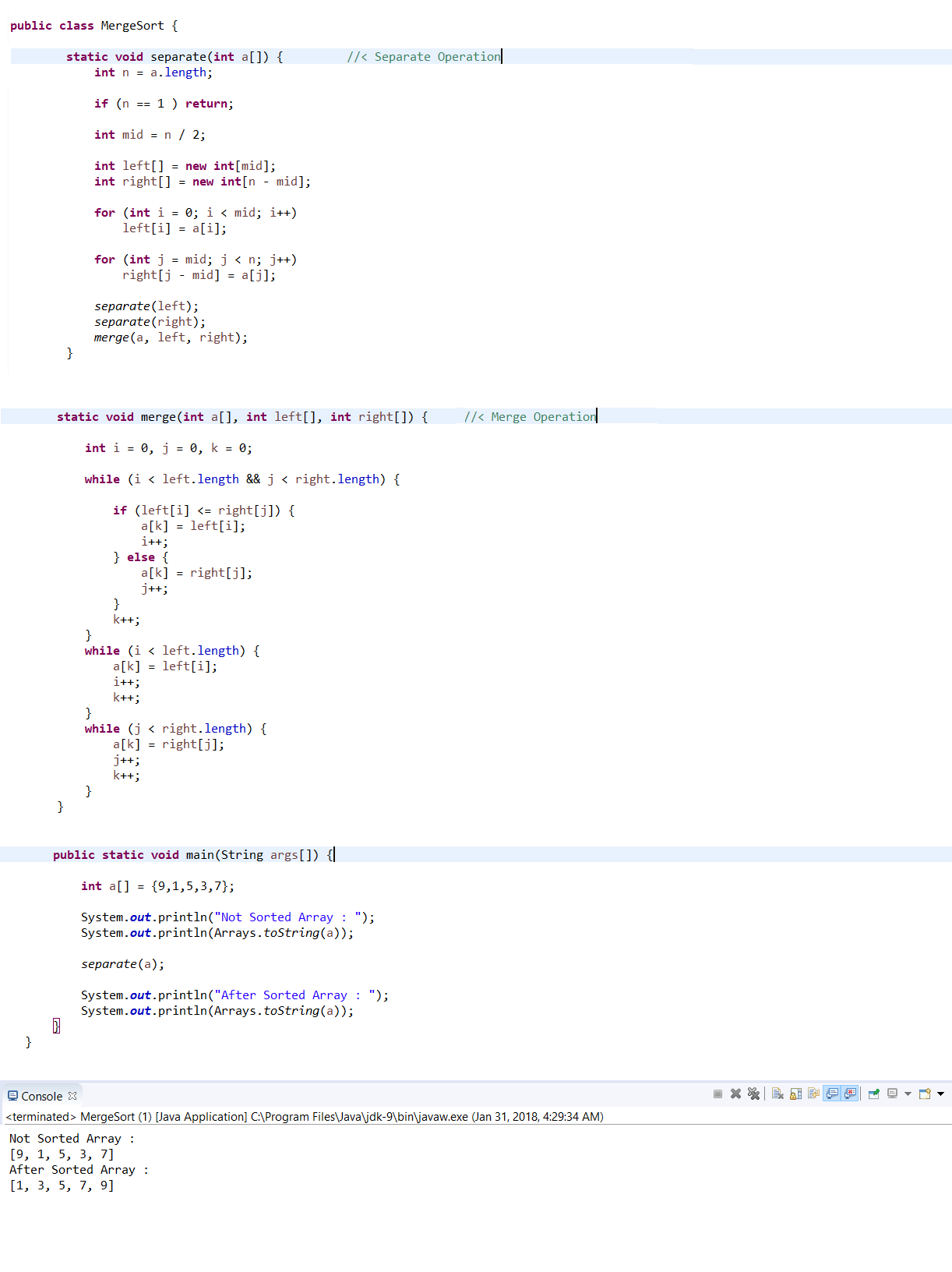
**<-** Compare Operation between the lower in   
 array right and another Left

But take care: any side (L or R) it’s must will   
 finish first for example (Left side) so reaming   
 number in (Right side) it’s will moving to (Array a)

while (i < Left.lenght)  
 a[k] = Left[i];  
 i ++;  
 k ++;  
 while (j < Right.lenght)   
 a[k] = Right[j];  
 j++:  
 k++;

**<-** If (side Left) is finished the first so will be   
 transfer of remaining number from (Right side)   
 to (Array a)

**<-** If (side Right) is finished the first so will be   
 transfer of remaining number from (Left side) to  
 (Array a)



**Implementation Code**