Insertion Sort Implementation

```
/*Function to sort int array using Insertion Sort*/
void insertionSort(int arr[])
{
        int n = arr.length;
        for (int i = 1; i < n; ++i) {
            int key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
}
```

Bubble Sort Implementation

```
/*Function to sort int array using Bubble Sort*/
void bubbleSort(int arr[])
{
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {
                    //swap arr[j+1] and arr[j]
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
}
```

Merge Sort Implementation

```
/**The Merge class provides static methods for sorting an
 * array using a top-down, recursive version of Merge Sort.*/
public class Merge {

    //stably merge a[lo .. mid] with a[mid+1 ..hi] using aux[lo .. hi]
    private static void merge(Comparable[] a, Comparable[] aux, int
lo, int mid, int hi) {
        //copy to aux[]
        for (int k = lo; k <= hi; k++) {
            aux[k] = a[k];
        }
        //merge back to a[]
        int i = lo, j = mid+1;
        for (int k = lo; k <= hi; k++) {
            if      (i > mid)               a[k] = aux[j++];
            else if (j > hi)                a[k] = aux[i++];
            else if (less(aux[j], aux[i])) a[k] = aux[j++];
            else                            a[k] = aux[i++];
        }
    }

    //mergesort a[lo..hi] using auxiliary array aux[lo..hi]
    private static void sort(Comparable[] a, Comparable[] aux, int lo,
int hi) {
        if (hi <= lo) return;
        int mid = lo + (hi - lo) / 2;
        sort(a, aux, lo, mid);
        sort(a, aux, mid + 1, hi);
        merge(a, aux, lo, mid, hi);
    }

    //helper function. Is v < w?
    private static boolean less(Comparable v, Comparable w) {
        return v.compareTo(w) < 0;
    }
}
```