JAVA Implementation:

Bubble sort:

```
public static void BubbleSort( int [ ] num )
{
    int j;
    boolean flag = true;   // set flag to true to begin first pass
    int temp;   //holding variable

    while ( flag )
    {
        flag= false;    //set flag to false awaiting a possible swap
        for( j=0;  j < num.length -1;  j++ )
        {
            if ( num[ j ] < num[j+1] )   // change to > for ascending sort
            {
                temp = num[ j ];               //swap elements
                num[ j ] = num[ j+1 ];
                num[ j+1 ] = temp;
                flag = true;              //shows a swap occurred
            }
        }
    }
}
```

Quick sort:

JAVA Implementation:

```
int partition(int arr[], int left, int right)
{
    int i = left, j = right;
    int tmp;
```

```
    int pivot = arr[(left + right) / 2];
    while (i <= j) {
        while (arr[i] < pivot)
            i++;
        while (arr[j] > pivot)
            j--;
        if (i <= j) {
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i++;
            j--;
        }
    };
    return i;
}

void quickSort(int arr[], int left, int right) {
    int index = partition(arr, left, right);
    if (left < index - 1)
        quickSort(arr, left, index - 1);
    if (index < right)
    quickSort(arr, index, right);
}
```

Merge sort:

```
void doMergeSort(int lowerIndex, int higherIndex) {

        if (lowerIndex < higherIndex) {
            int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
            // Below step sorts the left side of the array
            doMergeSort(lowerIndex, middle);
            // Below step sorts the right side of the array
            doMergeSort(middle + 1, higherIndex);
            // Now merge both sides
            mergeParts(lowerIndex, middle, higherIndex);
        }
    }

     mergeParts(int lowerIndex, int middle, int higherIndex) {

        for (int i = lowerIndex; i <= higherIndex; i++) {
            tempMergArr[i] = array[i];
        }
        int i = lowerIndex;
        int j = middle + 1;
        int k = lowerIndex;
        while (i <= middle && j <= higherIndex) {
            if (tempMergArr[i] <= tempMergArr[j]) {
                array[k] = tempMergArr[i];
                i++;
            } else {
                array[k] = tempMergArr[j];
                j++;
            }
            k++;
        }
        while (i <= middle) {
            array[k] = tempMergArr[i];
            k++;
            i++;
        }
```