Public class MergeSorter

{

A = anArray;

} public void sort()

{

If(a.length <= 1) return;

Int[] first = new int[a.length/2];

Int[] second = new int[a.length – first.length];

System.arraycopy(a,0,first,first.length);

System.arraycopy(a, first.length, second, 0, second.length);

MergeSorter firstSorter = new MergeSorter(first);

MergeSorter secondSorter = new MergeSorter(second);

firstSorter.sort();

secondSorter.sort();

merge(first, second);

}

Private void merge(int[] first, int[] second)

{

Int iFirst = 0;

Int iSecond = 0;

Int j = 0;

While(iFirst < first.length && iSecond < second.length)

{

If(first[iFirst] < second[iSecond]

{

ArrayUtil.print(a);

A[j] = first[iFirst];

iFirst++;

}

Else

{

ArrayUtil.print(a);

A[j] = second[iSecond];

iSecond++;

}

}

Private int[] a;

Public class QuickSorter2

{

Public QuickSOrter2(int[] anArray)

{

A = anArray;

}

Public void sort()

{

Sort(0,a.length-1);

}

Public void sort(int start, int end)

{

Int l, r ,mid;

L = start;

R = end;

Mid = a[(l+r)/2)];

While(r>1)

{

While(a[l]<mid)

++l;

While (a[r]>mid)

--r;

If(r>l)

{

Swap(l,r);

ArayUtil.print(a);

++l;

--r;

}

If(r>start)

Sort(start,r);

If(l<end-1)

Sort(l,end);

}

}  
}

}