```java
import java.util.Arrays;

public class CommonSortApp {

    public void bubbleSort(int[] array) {
        if (array.length <= 1)    return;

        for (int i = 0; i < array.length; i++) {
            boolean flag = false;
            for (int j = 0; j < array.length-1 - i; j++) {
                if (array[j] > array[j+1]) {
                    int temp = array[j];
                    array[j] = array[j+1];
                    array[j+1] = temp;
                    flag = true;
                }
            }
            if (!flag)    break;
        }
    }

    public void insertionSort(int[] array) {
        if (array.length <= 1)    return;

        for (int i = 1; i < array.length; i++) {
            int value = array[i];
            int j = i - 1;
            for (; j >= 0; j--) {
                if (array[j] > value) {
                    array[j+1] = array[j];
                } else {
                    break;
                }
            }
            array[j+1] = value;
        }
    }

    public void recursionMerge(int[] array, int low, int high) {
        if (low >= high)    return;

        int mid = low + (high - low)/2;
        recursionMerge(array, low, mid);
        recursionMerge(array, mid+1, high);
        merge(array, low, mid, high);
    }

    public void merge(int[] array, int low, int mid, int high) {
```

```
49          int i = low;
50          int j = mid + 1;
51          int k = 0;
52          int[] temp = new int[high - low + 1];
53          while (i <= mid && j <= high) {
54              if (array[i] < array[j]) {
55                  temp[k++] = array[i];
56              } else {
57                  temp[k++] = array[j];
58              }
59          }
60
61          int start = i;
62          int end = mid;
63          if (j <= high) {
64              start = j;
65              end = high;
66          }
67
68          while (start <= high) {
69              temp[k++] = array[start++];
70          }
71
72          for (int z = 0; z <= high - low; z++) {
73              array[low + z] = temp[z];
74          }
75      }
76
77
78      public void quickSortRecursion(int[] array, int low,
    int high) {
79          if (low >= high)    return;
80
81          int keyPoint = partition(array, low, high);
82          quickSortRecursion(array, low, keyPoint-1);
83          quickSortRecursion(array, keyPoint+1, high);
84      }
85
86      public int partition(int[] array, int low, int high) {
87          int i = low;
88          int pivot = array[high];
89          for (int j = low; j < high; j++) {
90              if (array[j] > pivot) {
91                  int temp = array[i];
92                  array[i] = array[j];
93                  array[j] = temp;
94                  i++;
95              }
96          }
97
```

```
 98          int tempValue = array[i];
 99          array[i] = array[high];
100          array[high] = tempValue;
101          System.out.print(i + "");
102          return i;
103      }
104
105      private int pivotCompared(int array[]) {
106          int high = array.length-1;
107          int[] pivotArray = {array[0], array[(int)(high/2)]
    , array[high]};
108          Arrays.sort(pivotArray);
109          return pivotArray[1];
110      }
111 }
```