```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace _11.Sorting
 8  {
 9      class CAppQuickSort
10      {
11          #region Private Methods
12          private void Print<T>(ref T[] arr) where T : IComparable
13          {
14              foreach (T el in arr)
15                  Console.Write("{0}, ", el);
16
17              Console.WriteLine();
18          }
19
20          /// <summary>
21          /// Swap two elements
22          /// </summary>
23          /// <param name="arr"></param>
24          /// <param name="i"></param>
25          /// <param name="j"></param>
26          private void Swap<T>(ref T[] arr, int i, int j) where T : IComparable
27          {
28              Console.WriteLine("Swap {0} with {1}", i, j);
29
30              T tmp = arr[i];
31              arr[i] = arr[j];
32              arr[j] = tmp;
33          }
34
35          /// <summary>
36          ///
37          /// </summary>
38          /// <param name="a"></param>
39          /// <param name="b"></param>
40          /// <returns></returns>
41          private bool Less(IComparable a, IComparable b)
42          {
43              return a.CompareTo(b) == 1;
44          }
45
46          /// <summary>
47          ///
48          /// </summary>
49          /// <param name="a"></param>
50          /// <param name="lo"></param>
51          /// <param name="hi"></param>
52          /// <returns></returns>
```

```
53          private int Partition<T>(ref T[] a, int lo, int hi) where T : IComparable
54          {
55              // Partition into a[lo..i-1], a[i], a[i+1..hi].
56
57              // left and right scan indices
58              int iStart = lo, jEnd = hi + 1;
59
60              // partitioning item
61              IComparable v = a[lo];
62
63              while (true)
64              {
65                  // Scan right, scan left, check for scan complete, and exchange.
66                  while (Less(a[++iStart], v))
67                      if (iStart == hi) break;
68
69                  while (Less(v, a[--jEnd]))
70                      if (jEnd == lo) break;
71
72                  if (iStart >= jEnd)
73                      break;
74                  Swap(ref a, iStart, jEnd);
75              }
76
77              // Put v = a[j] into position
78              Swap(ref a, lo, jEnd);
79
80              // with a[lo..j-1] <= a[j] <= a[j+1..hi].
81              return jEnd;
82          }
83
84
85          /// <summary>
86          ///
87          /// </summary>
88          /// <param name="a"></param>
89          /// <param name="lo"></param>
90          /// <param name="hi"></param>
91          private void Sort<T>(ref T[] a, int lo, int hi) where T : IComparable
92          {
93              if (hi <= lo)
94                  return;
95              Console.WriteLine("Before part");
96              Print(ref a);
97
98              // Partition (see page 291).
99              int j = Partition(ref a, lo, hi);
100
101             Console.WriteLine("After part");
102             Console.WriteLine("J = {0}, low = {1}, high = {2}", j, lo, hi);
103             Print(ref a);
104             Console.WriteLine("*****************************");
```

```
105
106             // Sort left part a[lo .. j-1].
107             Sort(ref a, lo, j - 1);
108
109             // Sort right part a[j+1 .. hi].
110             Sort(ref a, j + 1, hi);
111         }
112         #endregion
113
114
115         #region Public Methods
116         /// <summary>
117         ///
118         /// </summary>
119         /// <param name="a"></param>
120         public void Sort<T>(ref T[] a) where T : IComparable
121         {
122             // Eliminate dependence on input.
123             //StdRandom.shuffle(a);
124
125             Sort(ref a, 0, a.Length - 1);
126         }
127         #endregion
128     }
129 }
130
```