```
1     // Internal selection method that makes recursive calls.
2     // Places the kth smallest item in a[k-1].
3     // Initial call is quickSelect( a, 0, a.length - 1, k )
4
5     private static void
6     quickSelect( Comparable [ ] a, int low, int high, int k )
7     {
8         if( low + CUTOFF > high )
9             insertionSort( a, low, high );
10        else
11        {
12                // Sort low, middle, high
13            int middle = ( low + high ) / 2;
14            if( a[ middle ].lessThan( a[ low ] ) )
15                swapReferences( a, low, middle );
16            if( a[ high ].lessThan( a[ low ] ) )
17                swapReferences( a, low, high );
18            if( a[ high ].lessThan( a[ middle ] ) )
19                swapReferences( a, middle, high );
20
21                // Place pivot at position high - 1
22            swapReferences( a, middle, high - 1 );
23            Comparable pivot = a[ high - 1 ];
24
25                // Begin partitioning
26            int i, j;
27            for( i = low, j = high - 1; ; )
28            {
29                while( a[ ++i ].lessThan( pivot ) )
30                    ;
31                while( pivot.lessThan( a[ --j ] ) )
32                    ;
33                if( i < j )
34                    swapReferences( a, i, j );
35                else
36                    break;
37            }
38
39                // Restore pivot
40            swapReferences( a, i, high - 1 );
41
42                // Recurse; only this part changes
43            if( k - 1 < i )
44                quickSelect( a, low, i - 1, k );
45            else if( k - 1 > i )
46                quickSelect( a, i + 1, high, k );
47        }
48    }
```

**Figure 8.20** Quickselect with median-of-three partitioning and cutoff for small subarrays