Figure 21.3 Another way of sorting—*bubble* sort.

Scan through the list for adjacent items out of order, and swap them:

i	**list[i]**	**after comparing items**			
		0 and 1	**1 and 2**	**2 and 3**	**3 and 4**
0	Chan	Cassidy	Cassidy	Cassidy	Cassidy
1	Cassidy	Chan	Biggles	Biggles	Biggles
2	Biggles	Biggles	Chan	Chan	Chan
3	Clarke	Clarke	Clarke	Clarke	Adams
4	Adams	Adams	Adams	Adams	Clarke

Now we see that

- the list isn't sorted yet, though it's "more" sorted
- "Clarke" is in the right place, so the next scan can stop at item 3

For a list of size $\boxed{N}$, we need to repeat this scan-and-swap $\boxed{N-1}$ times.

This algorithm is called **bubble-sort**; here's it's Java code:

```
/**
 * sort: sorts Comparables in "list" in increasing order
 */

public static void sort (Comparable[] list) {

    final int N = list.length;

    // N - 1 scans
    for (int scan = 0; scan < N - 1; scan++) {

        // scan-and-swap pairs (0,1), (1,2), …, (N-2, N-1) on scan #0
        // pairs (0,1), (1,2), …, (N-2-scan, N-1-scan) on scan #scan
        for (int lower = 0; lower < N - 1 - scan; lower++) {

            if (list[lower].compareTo(list[lower + 1]) > 0) {
                Comparable temp = list[lower];    // swap
                list[lower] = list[lower + 1];
                list[lower + 1] = temp;
            }
        }    // ends "for lower"
    }    // ends "for scan"
}
```