Figure 20.3 Sorting non-ints (continued).

To solve this, let's add a **compareTo** method to **Student**:

```
/**
 * compareTo: compares this.stunum to other.stunum
 *   – returns 0 if they are equal, or a -ve (+ve) int if this.stunum
 *     is lexicographically less than (greater than) other.stunum
 */

public int compareTo (Student other) {

    return stunum.compareTo(other.stunum);
}
```

Now we don't have to change line $\boxed{2}$ of **sort()** at all!

In general, to make a sort method for an arbitrary class,
copy the **int** version of **sort()** and

- in **sort()**, change **int** to the name of the class to be sorted
- in the class to be sorted, add a **compareTo()** method

However, we should avoid repetitive code!

There is a clear pattern now, but we need a new version of **sort()**
for every class to be sorted.

To avoid this, make all "sortable" classes subclasses of
a new **Sortable** class:

```
abstract class Sortable {
    abstract public int compareTo (Sortable other);
}
```

Recall that an abstract class

- cannot be instantiated, and
- subclasses of an abstract class that wish to be concrete (non-abstract)
  must override all inherited methods with implementations.