Figure 18.1 Using an interface.

You don't extend an interface, you *implement* it:

```
class Student extends Person implements Sortable {
    // class body
}
```

- you can implement more than one interface
- your class must provide the behaviour for all interfaces it implements

Now, our sorting method looks like this:

```
// sort: sorts Sortables in "list" in increasing order
public static void sort (Sortable[] list) {
    Sortable[] sorted = new Sortable[list.length];
    // body of sort as before ...
}
```

However, there is still a problem. Since we wrote the **Sortable** interface, it can't be used to sort

- **int**s or any other primitive type
- any pre-existing class we didn't define, such as **String** (even though **String** has a **compareTo()** method)

Fortunately, Java defines a **Comparable** interface exactly like our **Sortable**, that is implemented by **String**, **Integer**, etc.

So, finally, our sorting method looks like this:

```
// sort: sorts Comparables in "list" in increasing order
public static void sort (Comparable[] list) {
    Comparable[] sorted = new Comparable[list.length];
    // body of sort as before ...
}
```

With this we can sort pretty much whatever we want!