Sorting with the Comparable and Comparator Interfaces

Example

I want to write a program that will read a sequence of names, sort it, and print it back in alphabetical order. Typical names might be

bob

Fred Flintstone

John Frederick Oberlin

Charles Philip Arthur George Windsor

Naturally, we want to sort by last name.

What class structures will we use for this?

I like this structure:

First, we'll have a Name class to hold the names. This has two String class variables: first and last (which might more properly be called givenNames, familyName). We will make a constructor that takes a string holding the full name and splits it up into those fields.

Then we need to store a whole sequence of these. This seems like another good use of ArrayLists, so we'll make an ArrayList<Name> object to hold the list.

At this point the program is much like our SimpleList program. The one difference is that we need to sort the list.

There are two ways to get Java to handle the sorting. One way is to have the Name class implement the *Comparable*<Name> interface, which just means that we give it a compareTo( ) method. This takes a Name *other* as an argument and returns -1, 0, or 1 if this < other, this ==other, or this > other.

The *Collections* class has a static method sort( ) that can be applied to any list whose base type implements the Comparable interface. So we add method compareTo( ) to Name, we note that Name implements the Comparable interface, and then we call Collections.sort(L), where L is the list of names to be sorted.

The other way is to make a class that implements the *Comparator*<Name> interface. This class needs a method Compare(n1, n2) where n1 and n2 have the base type of the list being sorted. Compare returns -1, 0 or 1 if n1<n2, n1==n2 or n1 > n2.

The Collections class has a method sort(L, comp) where comp is an object of the class that implements the Comparator interface. This time we don't need to make any changes to the Name class; we add the comparator class as a nested class inside our application program.

See the NameListComparable and NameListComparator programs.