Using interface java.lang.Comparable

Suppose we have a class TimeOfDay and want to sort an array of objects of the class based on their times. Or, we could want to sort an array of class Movie based on our own ranking of these movies. In each case, it looks like we have to write a procedure to do the sorting. We show you that the use of interface Comparable makes this unnecessary.

To the right is iterator java.lang.Comparable. We have shortened the specification somewhat, removing some technical considerations. You can see the documentation for this interface here:

docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

```
public interface Comparable<T> {
    /** Return a negative int, zero, or a positive
      * int as this object is less than, equal to,
      * or greater than ob. */
    int compareTo(T ob);
}
```

Here is a class Time. It maintains the time of day. We omit all methods, except function compareTo, which is required because interface Comparable is implemented.

```
/** An instance maintains a time of day */
public class Time implements Comparable<Time> {
  private int hr; // hour of the day, in 0..23
  private int min; // minute of the hour, in 0..59

  public @Override int compareTo(Time ob) {
    return 60*hr + min -  (60*ob.hr + ob. min);
  }
}
```

Now suppose we create an array b, fill in its elements in some fashion, and are ready to sort the array:

```
Time[] b= new Time[50];
Store Time objects in all elements of b;
// Sort b
```

How do we sort? Let's look at the Java API documentation for class java.util.Arrays, which has this URL:

docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

We find this method sort (we have abbreviated the specification):

```
/** Sort c into ascending order, according to the natural ordering of its elements.
  * All elements in the array must implement interface Comparable. */
public static void sort(Object[] c)
```

The "natural ordering" is the ordering given by method compareTo. Objects of our class Time implement Comparable, so we can use this method to sort the array:

```
Time[] b= new Time[50];
Store Time objects in all elements of b;
// Sort b
java.util.Arrays.sort(b);
```

Summary

Without interfaces, we would have to write a different sort procedure for every different class —well, it would be essentially the same sort procedure except that the part that compares array elements would be different. But it would be an awkward mess. Using interface Comparable, we can use *one* sorting procedure.

Many classes that come with Java already implement interface Comparable, for example: all the wrapper classes, like Integer and Character as well as String. So you can sort arrays of these classes using method java.util.Arrays.sort.