

All of the algorithms discussed in this chapter work not only for whole arrays but also for subarrays. We shall generally state the problem to be solved in terms of a (sub)array $a[\textit{left} \dots \textit{right}]$.

Java, like most programming languages, has no special notation for subarrays. This causes no difficulties in practice: a method that must process the subarray $a[\textit{left} \dots \textit{right}]$ can be coded with `a`, `left`, and `right` as separate parameters.

3.1.2 Sorted arrays

An array or subarray is *sorted* if the values of its components are in ascending order, i.e., the value of each component of the (sub)array is less than or equal to the value of that component's right neighbor. Many of the algorithms discussed in this chapter assume that a (sub)array is sorted.

Figure 3.4 illustrates this definition with sorted and unsorted arrays of words and of integers.

The meaning of ' x is *less* than y ' (or equivalently ' y is *greater* than x ') must be defined separately for each data type. For numbers it means that x is a smaller number than y . For strings, it conventionally means that x lexicographically precedes y . For dates, it naturally means that x is earlier than y .

The meanings of *least* and *greatest* are derived in the obvious way from the meanings of *less* and *greater*. In a sorted array, the leftmost component contains the least value, and the rightmost component the greatest value.

Java 2's `java.lang.Comparable` interface (shown in Program 3.5) captures the notions of *less* and *greater*. This interface requires a `compareTo` method to compare two objects, such that `x.compareTo(y)` returns a negative integer if x is *less* than y , zero if x is *equal* to y , or a positive integer if x is *greater* than y . `Comparable` is implemented by several Java 2 library classes including `Integer` and `String`. Other

Sorted:	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: center; padding: 0 5px;">0</td> <td style="text-align: center; padding: 0 5px;">1</td> <td style="text-align: center; padding: 0 5px;">2</td> <td style="text-align: center; padding: 0 5px;">3</td> </tr> <tr> <td style="padding: 0 5px;">George</td> <td style="padding: 0 5px;">John</td> <td style="padding: 0 5px;">Paul</td> <td style="padding: 0 5px;">Ringo</td> </tr> </table>	0	1	2	3	George	John	Paul	Ringo	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: center; padding: 0 5px;">0</td> <td style="text-align: center; padding: 0 5px;">1</td> <td style="text-align: center; padding: 0 5px;">2</td> <td style="text-align: center; padding: 0 5px;">3</td> <td style="text-align: center; padding: 0 5px;">4</td> </tr> <tr> <td style="padding: 0 5px;">5</td> <td style="padding: 0 5px;">5</td> <td style="padding: 0 5px;">12</td> <td style="padding: 0 5px;">17</td> <td style="padding: 0 5px;">23</td> </tr> </table>	0	1	2	3	4	5	5	12	17	23
0	1	2	3																	
George	John	Paul	Ringo																	
0	1	2	3	4																
5	5	12	17	23																
Unsorted:	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: center; padding: 0 5px;">0</td> <td style="text-align: center; padding: 0 5px;">1</td> <td style="text-align: center; padding: 0 5px;">2</td> <td style="text-align: center; padding: 0 5px;">3</td> </tr> <tr> <td style="padding: 0 5px;">John</td> <td style="padding: 0 5px;">Paul</td> <td style="padding: 0 5px;">George</td> <td style="padding: 0 5px;">Ringo</td> </tr> </table>	0	1	2	3	John	Paul	George	Ringo	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="text-align: center; padding: 0 5px;">0</td> <td style="text-align: center; padding: 0 5px;">1</td> <td style="text-align: center; padding: 0 5px;">2</td> <td style="text-align: center; padding: 0 5px;">3</td> <td style="text-align: center; padding: 0 5px;">4</td> </tr> <tr> <td style="padding: 0 5px;">5</td> <td style="padding: 0 5px;">23</td> <td style="padding: 0 5px;">12</td> <td style="padding: 0 5px;">17</td> <td style="padding: 0 5px;">5</td> </tr> </table>	0	1	2	3	4	5	23	12	17	5
0	1	2	3																	
John	Paul	George	Ringo																	
0	1	2	3	4																
5	23	12	17	5																

Figure 3.4 Sorted and unsorted arrays.

```
public interface Comparable {
    public int compareTo (Object that);
    // Return a negative integer if this object is less than that,
    // or zero if this object is equal to that,
    // or a positive integer if this object is greater than that.
}
```

Program 3.5 The Java `Comparable` interface.