provide at least those listed in the interface. The interface is implemented at lines 23 to 30. Notice that we must implement the *exact method* specified in the interface. Thus these methods take `Comparable` as a parameter, instead of `MyInteger`.

A class that implements an interface can be extended if it is not final. Thus, if `MyInteger` was not final, it could be extended.

A class that implements an interface may still extend one other class. For instance, we could in principle have written

```
public class MyInteger extends Integer implements Comparable
```

This is illegal only because `Integer` is a final class and cannot be extended.

```
 1 package Supporting;
 2
 3 final public class MyInteger implements Comparable
 4 {
 5         // Constructor
 6     public MyInteger( int x )
 7       { value = x; }
 8
 9         // Some methods
10     public String toString( )
11       { return Integer.toString( value ); }
12
13     public int intValue( )
14       { return value; }
15
16     public boolean equals( Object rhs )
17     {
18         return rhs instanceof MyInteger &&
19                value == ((MyInteger)rhs).value;
20     }
21
22         // Implement the interface
23     public int compares( Comparable rhs )
24     {
25         return value < ((MyInteger)rhs).value ? -1 :
26                value == ((MyInteger)rhs).value ? 0 : 1;
27     }
28
29     public boolean lessThan( Comparable rhs )
30       { return value < ((MyInteger)rhs).value; }
31
32         // Data field
33     private int value;
34 }
```

Figure 4.13 The `MyInteger` class (preliminary version), which implements the `Comparable` interface