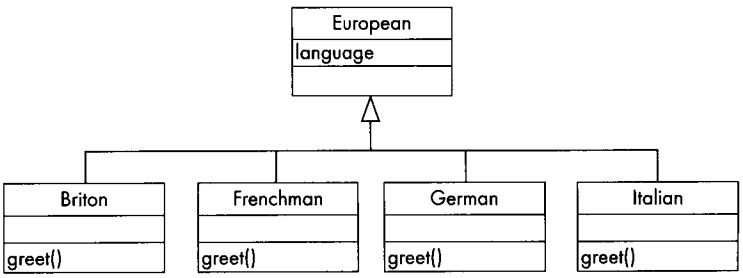

Figure 4.19 *Inheritance hierarchy for the European class*

```
1   public class Frenchman extends European
2   {
3   // 'String language' does not need to be declared as it is inherited
4
5       Frenchman()
6       {
7       language = "French";
8       }
9
10      void greet()
11      {
12      System.out.println(" Bonjour");
13      }
14  }
```

Figure 4.20 *The Java code for the Frenchman class*

Figure 4.19 shows a simple inheritance hierarchy for a European class, and Figure 4.20 contains the Java code for Frenchman, one of the subclasses in the hierarchy.

In Figure 4.20, line 10 is the operation part of the process greet(). Line 12 is the method.

Notice in the inheritance hierarchy in Figure 4.19 that, although it is inherited, the greet() operation appears in all the classes in the hierarchy. This is not because we have forgotten that inherited features can be omitted from the diagram, it is because greet() is redefined (i.e. is implemented by a different method) in all of the subclasses.

The significance of the distinction between operation and method is that, in an inheritance hierarchy, it becomes possible for one operation to be implemented by many methods. When a subclass inherits an operation, as long as it doesn't alter the interface, it can change the method to suit its own specialization. Altering a method is called *over-riding,* and an operation that is