The watchStock method ensures, through the data type of its first argument, that all registered objects implement the valueChanged method. It makes sense to use an interface data type here because it matters only that registrants implement a particular method. If StockMonitor had used a class name as the data type, that would artificially force a class relationship on its users. Because a class can have only one superclass, it would also limit what type of objects can use this service. By using an interface, the registered object's class could be anything—Applet or Thread, for instance—thus allowing any class anywhere in the class hierarchy to use this service.

Defining an Interface

Figure 69 shows that an interface definition has two components: the interface declaration and the interface body. The interface declaration declares various attributes about the interface, such as its name and whether it extends other interfaces. The interface body contains the constant and the method declarations for that interface.

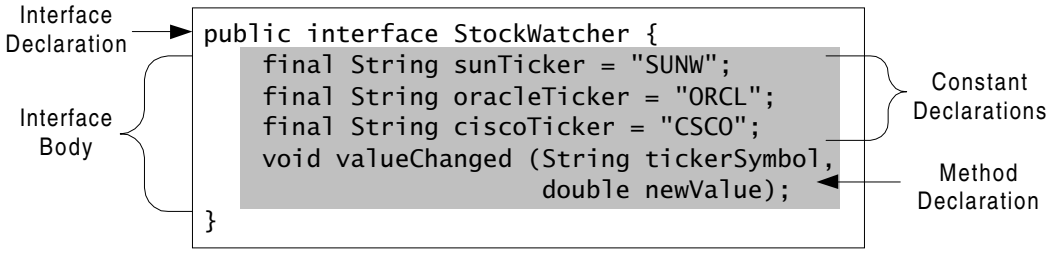

Figure 69 The StockWatcher interface and the structure of an interface definition.

```
public interface StockWatcher {
    final String sunTicker = "SUNW";
    final String oracleTicker = "ORCL";
    final String ciscoTicker = "CSCO";
    void valueChanged(String tickerSymbol, double newValue);
}
```

The interface shown in Figure 69 is the StockWatcher interface mentioned previously. This interface defines three constants, which are the ticker symbols of watchable stocks. This interface also declares, but does not implement, the valueChanged method. Classes that implement this interface provide the implementation for that method.