## Special Topic 15.2

### Static Inner Classes

You first saw the use of inner classes for event handlers. Inner classes are useful in that context, because their methods have the privilege of accessing private data members of outer-class objects. The same is true for the `LinkedListIterator` inner class in the sample code for this section. The iterator needs to access the `first` instance variable of its linked list.

However, the Node inner class has no need to access the outer class. In fact, it has no methods. Thus, there is no need to store a reference to the outer list class with each Node object. To suppress the outer-class reference, you can declare the inner class as static:

```
public class LinkedList
{
   . . .
   private static class Node
   {
      . . .
   }
}
```

The purpose of the reserved word static in this context is to indicate that the inner-class objects do not depend on the outer-class objects that generate them. In particular, the methods of a static inner class cannot access the outer-class instance variables. Declaring the inner class static is efficient, because its objects do not store an outer-class reference.

However, the LinkedListIterator class cannot be a static inner class. It frequently references the first element of the enclosing LinkedList.