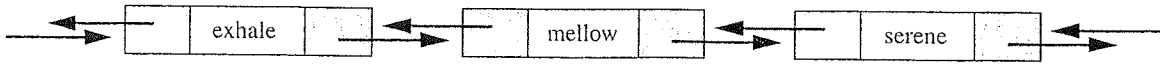
7.3 | DOUBLY LINKED LISTS

Suppose we want to insert "placid" in front of "serene" in the doubly linked list partially shown in Figure 7.2 and repeated here:



LinkedList into

First we need to get a reference to the Entry object that holds "serene"; that will take linear-in-n time, on average, where n is the size of the linked list. After that, as shown in Figure 7.8, the insertion entails constructing a new Entry object, storing "placid" as its element, and adjusting four links (the previous and next links for "placid", the next link for the predecessor of "serene", and the previous link for "serene"). In other words, once we have a reference to an Entry object, we can insert a new element in front of that Entry object in constant time.

The process for removal of an element in a doubly linked list is similar. For example, suppose we want to remove "mellow" from the partially shown linked list in Figure 7.8. First, we get a reference to the Entry object that houses "mellow", and this takes linear-in-n time, on average. Then, as shown in Figure 7.9, we adjust the

Figure 7.8 | The partially shown doubly linked list from Figure 7.2 after "placid" is inserted in front of "serene".

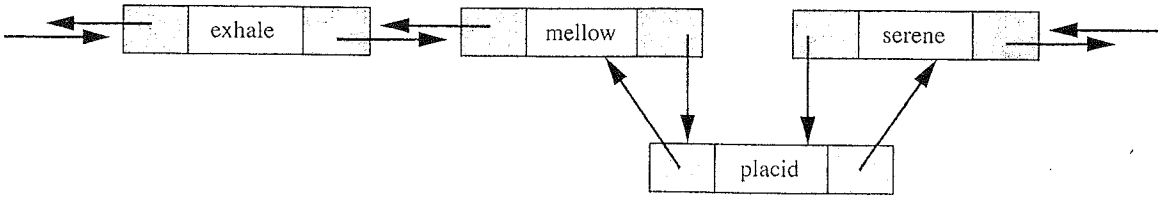
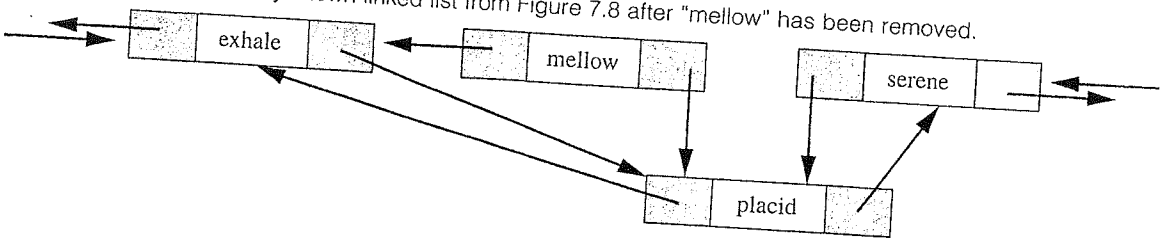


Figure 7.9 | The partially shown linked list from Figure 7.8 after "mellow" has been removed.



next link of the predecessor of "mellow" and the previous link of the successor of "mellow". Notice that there is no need to adjust either of the links in the Entry object that houses "mellow" because that object is not pointed to by any other Entry object's links.

The bottom line in the previous discussion is that it takes linear-in-n time to get a reference to an Entry object that houses an element, but once the reference is available, any number of insertions, removals, or retrievals can be accomplished in constant time for each one.

Now that you have a rough idea of what a doubly linked list looks like and can be manipulated, we are ready to study the LinkedList class, the Java Collections Framework's design and implementation of doubly linked lists. First as always, we start with a user's perspective, that is, with the method specifications.