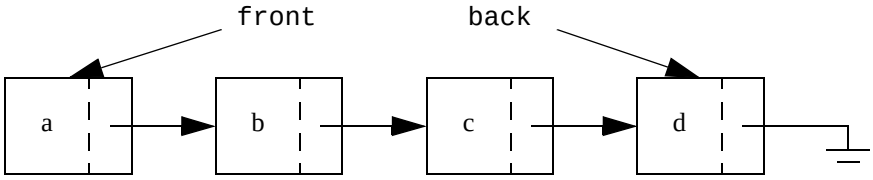

Figure 15.20 Linked-list implementation of the queue

```
 1 package DataStructures;
 2
 3 import Exceptions.*;
 4
 5 // QueueLi class
 6 //
 7 // CONSTRUCTION: with no initializer
 8 //
 9 // ******************PUBLIC OPERATIONS*********************
10 // void enqueue( x )    --> Insert x
11 // Object getFront( )   --> Return least recently inserted item
12 // Object dequeue( )    --> Return and remove least recent item
13 // boolean isEmpty( )   --> Return true if empty; else false
14 // void makeEmpty( )    --> Remove all items
15 // ******************ERRORS********************************
16 // getFront or dequeue on empty queue
17
18 /**
19  * List-based implementation of the queue.
20  */
21 public class QueueLi implements Queue
22 {
23     public QueueLi( )
24       { makeEmpty( ); }
25
26     public boolean isEmpty( )
27       { return front == null; }
28     public void makeEmpty( )
29       { front = back = null; }
30     public Object dequeue( ) throws Underflow
31       { /* Figure 15.22 */ }
32     public Object getFront( ) throws Underflow
33       { /* Figure 15.23 */ }
34     public void enqueue( Object x )
35       { /* Figure 15.25 */ }
36
37     private ListNode front;
38     private ListNode back;
39 }
```

Figure 15.21 Skeleton for the linked-list-based `Queue` class