```
 1 package DataStructures;
 2
 3 import Exceptions.*;
 4
 5 // QueueAr 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  * Array-based implementation of the queue.
20  */
21 public class QueueAr implements Queue
22 {
23     public QueueAr( )
24       { /* Figure 15.11 */ }
25
26     public boolean isEmpty( )
27       { return currentSize == 0; }
28     public void enqueue( Object x )
29       { /* Figure 15.12 */ }
30     public void makeEmpty( )
31       { /* Figure 15.15 */ }
32     public Object dequeue( ) throws Underflow
33       { /* Figure 15.14 */ }
34     public Object getFront( ) throws Underflow
35       { /* Figure 15.14 */ }
36
37     private Object [ ] theArray;
38     private int        currentSize;
39     private int        front;
40     private int        back;
41
42     static final int DEFAULT_CAPACITY = 10;
43
44     private int increment( int x )
45       { /* Figure 15.10 */ }
46     private void doubleQueue( )
47       { /* Figure 15.13 */ }
48 }
```

**Figure 15.9** QueueAr class skeleton