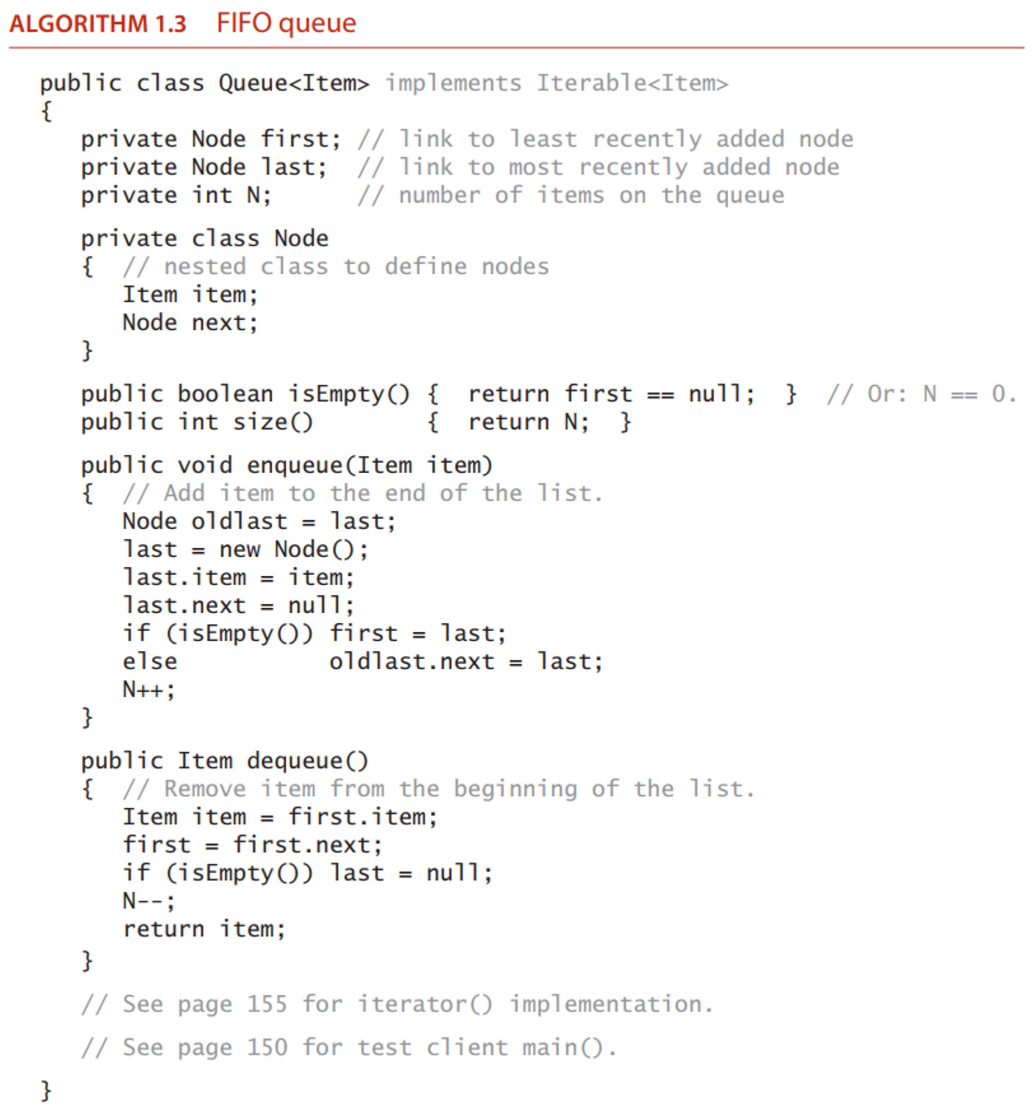
Queue implementation

Monday, September 12, 2022 1:17 PM

It mainains the queue as a linked list in order from least recently to most recently added items.



ALGORITHM 1.3 FIFO queue

```
public class Queue<Item> implements Iterable<Item>
{
   private Node first; // link to least recently added node
   private Node last;  // link to most recently added node
   private int N;      // number of items on the queue

   private class Node
   {  // nested class to define nodes
      Item item;
      Node next;
   }

   public boolean isEmpty() {  return first == null;  }  // Or: N == 0.
   public int size()        {  return N;  }

   public void enqueue(Item item)
   {  // Add item to the end of the list.
      Node oldlast = last;
      last = new Node();
      last.item = item;
      last.next = null;
      if (isEmpty()) first = last;
      else           oldlast.next = last;
      N++;
   }

   public Item dequeue()
   {  // Remove item from the beginning of the list.
      Item item = first.item;
      first = first.next;
      if (isEmpty()) last = null;
      N--;
      return item;
   }

   // See page 155 for iterator() implementation.

   // See page 150 for test client main().

}
```

```
public static void main(String[] args)
{  // Create a queue and enqueue/dequeue strings.

   Queue<String> q = new Queue<String>();

   while (!StdIn.isEmpty())
   {
      String item = StdIn.readString();
      if (!item.equals("-"))
           q.enqueue(item);
      else if (!q.isEmpty()) StdOut.print(q.dequeue() + " ");
   }

   StdOut.println("(" + q.size() + " left on queue)");
}
```

Test client for Queue