```java
import components.stack.StackSecondary;

/**
 * {@code Stack} represented as a singly linked list, done "bare-handed", with
 * implementations of primary methods.
 *
 * <p>
 * Execution-time performance of all methods implemented in this class is O(1).
 *
 * @param <T>
 *            type of Stack entries
 * @convention <pre>
 * $this.length >= 0  and
 * if $this.length == 0 then
 *   [$this.top is null]
 * else
 *   [$this.top is not null]  and
 *   [$this.top points to the first node of a singly linked list
 *    containing $this.length nodes]  and
 *   [next in the last node of that list is null]
 * </pre>
 * @correspondence this = [data in $this.length nodes starting at $this.top]
 */
public class Stack2<T> extends StackSecondary<T> {

    /*
     * Private members
    ------------------------------------------------------------
     */

    /**
     * Node class for singly linked list nodes.
     */
    private final class Node {

        /**
         * Data in node.
         */
        private T data;

        /**
```

```
41          * Next node in singly linked list, or null.
42          */
43         private Node next;
44 
45     }
46 
47     /**
48      * Top node of singly linked list.
49      */
50     private Node top;
51 
52     /**
53      * Number of nodes in singly linked list, i.e., length = |
   this|.
54      */
55     private int length;
56 
57     /**
58      * Creator of initial representation.
59      */
60     private void createNewRep() {
61 
62         this.length = 0;
63         this.top = null;
64 
65     }
66 
67     /*
68      * Constructors
   -----------------------------------------------------------
69      */
70 
71     /**
72      * No-argument constructor.
73      */
74     public Stack2() {
75         this.createNewRep();
76     }
77 
78     /*
79      * Standard methods removed to reduce clutter...
80      */
81 
82     /*
```

```
 83     * Kernel methods
   ------------------------------------------------------------
 84     */
 85
 86    @Override
 87    public final void push(T x) {
 88        assert x != null : "Violation of: x is not null";
 89
 90        Node added = new Node();
 91        added.data = x;
 92        added.next = this.top;
 93        this.top = added;
 94        this.length++;
 95    }
 96
 97    @Override
 98    public final T pop() {
 99        assert this.length() > 0 : "Violation of: this /= <>";
100
101        T popped = this.top.data;
102        this.top = this.top.next;
103        this.length--;
104        return popped;
105    }
106
107    @Override
108    public final int length() {
109
110        return this.length;
111    }
112
113    /*
114     * Iterator code removed to reduce clutter...
115     */
116
117 }
```