```java
import java.util.Iterator;
import java.util.NoSuchElementException;

import components.stack.Stack;
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;

        /**
         * Next node in singly linked list, or null.
         */
        private Node next;

    }

    /**
     * Top node of singly linked list.
     */
    private Node top;

    /**
     * Number of nodes in singly linked list, i.e., length = |this|.
     */
    private int length;

    /**
     * Creator of initial representation.
     */
    private void createNewRep() {

        // TODO - fill in body
        this.top = null;
        this.length = 0;

```

```
70      }
71
72      /*
73       * Constructors ------------------------------------------------------------
74       */
75
76      /**
77       * No-argument constructor.
78       */
79      public Stack2() {
80          this.createNewRep();
81      }
82
83      /*
84       * Standard methods removed to reduce clutter...
85       */
86
87      /*
88       * Kernel methods ----------------------------------------------------------
89       */
90
91      @Override
92      public final void push(T x) {
93          assert x != null : "Violation of: x is not null";
94
95          // TODO - fill in body
96          Node bob = new Node;
97          bob.next = null;
98          bob.data = x;
99          this.top = bob;
100         length()++;
101
102     }
103
104     @Override
105     public final T pop() {
106         assert this.length() > 0 : "Violation of: this /= <>";
107
108         // TODO - fill in body
109         Node bob = this.top;
110         T result = bob.data;
111         this.top = bob.next;
112         length()--;
113
114     }
115
116     @Override
117     public final int length() {
118
119         return this.length();
120
121     }
122
123     /*
124      * Iterator code removed to reduce clutter...
125      */
126
127 }
```