ArrayStack2Lab.java

```java
/**
    A class of stacks whose entries are stored in an array.
    @author Frank M. Carrano
    @version 3.0
*/
import java.util.Arrays;
public class ArrayStack2Lab<T> implements StackInterface<T>
{
    public static void main(String[] args)  {
    // Add you lab tests here

        // TEST for display method
        System.out.println("Running tests for the 'display' method:\n");

        // Test Case 1
        ArrayStack2Lab<String> test1stack = new ArrayStack2Lab<String>();
        test1stack.push("dog");
        test1stack.push("cat");
        test1stack.push("fish");
        test1stack.push("crocodile");
        System.out.println("test 1: \nThe expected output is: \ncrocodile\nfish\ncat\ndog\nThe actual output is:");
        test1stack.display();

        // Test Case 2
        ArrayStack2Lab<String> test2stack = new ArrayStack2Lab<String>();
        System.out.println("test 2: \nThe expected output is: \nThe stack is empty\nThe acutal output is:");
        test2stack.display();

        // TEST for remove method
        System.out.println("\nRunning tests for the 'remove' method:\n");

        // Test Case 3
        ArrayStack2Lab<String> test3stack = new ArrayStack2Lab<String>();
        test3stack.push("apple");
        test3stack.push("orange");
        test3stack.push("pear");
        System.out.println("test 3: \nExpected output is: 2");
        System.out.println("Actual output is: " + test3stack.remove(2));

        // Test Case 4
        ArrayStack2Lab<String> test4stack = new ArrayStack2Lab<String>();
        test4stack.push("apple");
        test4stack.push("orange");
        test4stack.push("pear");
        System.out.println("test 4: \nExpected output is: 3");
        System.out.println("Actual output is: " + test4stack.remove(8));
  }

    // Problem 1
    public void display() {
        // check if the stack is empty
        if (topIndex == -1)
```

```
53              // print appropriate message
54              System.out.println("The stack is empty");
55
56          // iterate through the stack starting from top
57          for (int i = topIndex; i >= 0; i--) {
58              System.out.println(stack[i]);
59          }
60      }
61
62      // Problem 2
63      public int remove(int n) {
64          /* the method removes the n top most entries for a stack. If the stack
    contains less than n items, the stack becomes empty. The method returns the number of
    items removed. */
65
66          // assign a counter
67          int numRemoved = 0;
68
69          // handle cases where n is larger than contents in stack
70          if (n >= topIndex + 1) {
71              numRemoved = topIndex + 1;
72              clear();
73              return numRemoved;
74          }
75
76          // pop n times and update counter accordingly
77          for (int i = 0; i < n; i++) {
78              if (!isEmpty()) {
79                  pop();
80                  numRemoved++;
81              }
82          }
83
84          // return counter
85          return numRemoved;
86      }
87
88
89     private T[] stack;    // array of stack entries
90     private int topIndex; // index of top entry
91     private static final int DEFAULT_INITIAL_CAPACITY = 50;
92
93     public ArrayStack2Lab()
94     {
95        this(DEFAULT_INITIAL_CAPACITY);
96     } // end default constructor
97
98     public ArrayStack2Lab(int initialCapacity)
99     {
100       // the cast is safe because the new array contains null entries
101       @SuppressWarnings("unchecked")
102       T[] tempStack = (T[])new Object[initialCapacity];
103       stack = tempStack;
104       topIndex = -1;
105    } // end constructor
106
107    public void push(T newEntry)
```

```
108     {
109         ensureCapacity();
110         topIndex++;
111         stack[topIndex] = newEntry;
112     } // end push
113 
114     private void ensureCapacity()
115     {       if (topIndex == stack.length - 1) // if array is full, double size of array
116       stack = Arrays.copyOf(stack, 2 * stack.length);
117     } // end ensureCapacity
118 
119     public T peek()
120     {
121         T top = null;
122         if (!isEmpty())
123       top = stack[topIndex];
124         return top;
125     } // end peek
126 
127     public T pop()
128     {
129         T top = null;
130 
131         if (!isEmpty()) {
132       top = stack[topIndex];
133       stack[topIndex] = null;
134       topIndex--;
135         } // end if
136 
137         return top;
138     } // end pop
139 
140     public boolean isEmpty()
141     {
142         return topIndex < 0;
143     } // end isEmpty
144 
145     public void clear()
146     {
147         for(int i = 0; i <= topIndex; ++i)
148       stack[i] = null;
149         topIndex = -1;
150     }
151 
152 } // end ArrayStack2Lab
153 
```