```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code Sequence<String>}'s constructor and kernel
 * methods.
 *
 * @author Gage Farmer
 *
 */
public abstract class SequenceTest {

    /**
     * Invokes the appropriate {@code Sequence} constructor for the
     * implementation under test and returns the result.
     *
     * @return the new sequence
     * @ensures constructorTest = <>
     */
    protected abstract Sequence<String> constructorTest();

    /**
     * Invokes the appropriate {@code Sequence} constructor for the reference
     * implementation and returns the result.
     *
     * @return the new sequence
     * @ensures constructorRef = <>
     */
    protected abstract Sequence<String> constructorRef();

    /**
     *
     * Creates and returns a {@code Sequence<String>} of the implementation
     * under test type with the given entries.
     *
     * @param args
     *            the entries for the sequence
     * @return the constructed sequence
     * @ensures createFromArgsTest = [entries in args]
     */
    private Sequence<String> createFromArgsTest(String... args) {
        Sequence<String> sequence = this.constructorTest();
        for (String s : args) {
            sequence.add(sequence.length(), s);
        }
        return sequence;
    }

    /**
     *
     * Creates and returns a {@code Sequence<String>} of the reference
     * implementation type with the given entries.
     *
     * @param args
     *            the entries for the sequence
     * @return the constructed sequence
     * @ensures createFromArgsRef = [entries in args]
     */
    private Sequence<String> createFromArgsRef(String... args) {
        Sequence<String> sequence = this.constructorRef();
```

```
64          for (String s : args) {
65              sequence.add(sequence.length(), s);
66          }
67          return sequence;
68      }
69
70      // TODO - add test cases for constructor, add, remove, and length
71
72      @Test
73      public void construcTest() {
74          Sequence<String> test = this.createFromArgsTest("1", "2", "3", "4");
75          Sequence<String> ref = this.createFromArgsRef("1", "2", "3", "4");
76
77          assertEquals(ref, test);
78      }
79
80      @Test
81      public void addTest() {
82          Sequence<String> test = this.createFromArgsTest("1", "2", "3");
83          Sequence<String> ref = this.createFromArgsRef("1", "2", "3", "4");
84
85          test.add(3, "4");
86
87          assertEquals(ref, test);
88      }
89
90      @Test
91      public void removeTest() {
92          Sequence<String> test = this.createFromArgsTest("1", "2", "3", "4");
93          Sequence<String> ref = this.createFromArgsRef("1", "2", "3");
94
95          test.remove(3);
96
97          assertEquals(ref, test);
98      }
99
100     @Test
101     public void lengthTest() {
102         Sequence<String> test = this.createFromArgsTest("1", "2", "3", "4");
103         Sequence<String> ref = this.createFromArgsRef("1", "2", "3", "4");
104
105         assertEquals(ref.length(), test.length());
106     }
107
108 }
109
```