```
 1 import static org.junit.Assert.assertEquals;
 7
 8 /**
 9  * Sample JUnit test fixture for SequenceSmooth.
10  *
11  * @author Sam Espanioly
12  *
13  */
14 public final class SequenceSmoothTest {
15
16     /**
17      * Constructs and returns a sequence of the integers
    provided as arguments.
18      *
19      * @param args
20      *             0 or more integer arguments
21      * @return the sequence of the given arguments
22      * @ensures createFromArgs= [the sequence of integers
    in args]
23      */
24     private Sequence<Integer> createFromArgs(Integer...
    args) {
25         Sequence<Integer> s = new Sequence1L<Integer>();
26         for (Integer x : args) {
27             s.add(s.length(), x);
28         }
29         return s;
30     }
31
32     /**
33      * Test smooth with s1 = <2, 4, 6> and s2 = <-5, 12>.
34      */
35     @Test
36     public void test1() {
```

```
37          /*
38           * Set up variables and call method under test
39           */
40          Sequence<Integer> seq1 = this.createFromArgs(2, 4,
    6);
41          Sequence<Integer> expectedSeq1 =
    this.createFromArgs(2, 4, 6);
42          Sequence<Integer> seq2 = this.createFromArgs(-5,
    12);
43          Sequence<Integer> expectedSeq2 =
    this.createFromArgs(3, 5);
44          SequenceSmooth.smooth(seq1, seq2);
45          /*
46           * Assert that values of variables match
    expectations
47           */
48          assertEquals(expectedSeq1, seq1);
49          assertEquals(expectedSeq2, seq2);
50      }
51
52      /**
53       * Test smooth with s1 = <7> and s2 = <13, 17, 11>.
54       */
55      @Test
56      public void test2() {
57          /*
58           * Set up variables and call method under test
59           */
60          Sequence<Integer> seq1 = this.createFromArgs(7);
61          Sequence<Integer> expectedSeq1 =
    this.createFromArgs(7);
62          Sequence<Integer> seq2 = this.createFromArgs(13,
    17, 11);
63          Sequence<Integer> expectedSeq2 =
```

```
   this.createFromArgs();
64          SequenceSmooth.smooth(seq1, seq2);
65          /*
66           * Assert that values of variables match
   expectations
67           */
68          assertEquals(expectedSeq1, seq1);
69          assertEquals(expectedSeq2, seq2);
70      }
71
72      /**
73       * Test smooth with s1 = <3, 4, 5> and s2 = <15647,-5,
   -12, 496>.
74       */
75      @Test
76      public void test3() {
77          /*
78           * Set up variables and call method under test
79           */
80          Sequence<Integer> seq1 = this.createFromArgs(3, 4,
   5);
81          Sequence<Integer> expectedSeq1 =
   this.createFromArgs(3, 4, 5);
82          Sequence<Integer> seq2 =
   this.createFromArgs(15647, -5, -12, 496);
83          Sequence<Integer> expectedSeq2 =
   this.createFromArgs(1, 4);
84          SequenceSmooth.smooth(seq1, seq2);
85          /*
86           * Assert that values of variables match
   expectations
87           */
88          assertEquals(expectedSeq1, seq1);
89          assertEquals(expectedSeq2, seq2);
```

```
 90     }
 91
 92     /**
 93      * Test smooth with s1 = <-398496> and s2 = <164,
    968794, 6516>.
 94      */
 95     @Test
 96     public void test4() {
 97         /*
 98          * Set up variables and call method under test
 99          */
100         Sequence<Integer> seq1 =
    this.createFromArgs(-398496);
101         Sequence<Integer> expectedSeq1 =
    this.createFromArgs(-398496);
102         Sequence<Integer> seq2 = this.createFromArgs(164,
    968794, 6516);
103         Sequence<Integer> expectedSeq2 =
    this.createFromArgs(-398496 / 2);
104         SequenceSmooth.smooth(seq1, seq2);
105         /*
106          * Assert that values of variables match
    expectations
107          */
108         assertEquals(expectedSeq1, seq1);
109         assertEquals(expectedSeq2, seq2);
110     }
111
112     /**
113      * Test smooth with s1 = <456, 987, 321, 0> and s2 =
    <>.
114      */
115     @Test
116     public void test5() {
```

```
117         /*
118          * Set up variables and call method under test
119          */
120         Sequence<Integer> seq1 = this.createFromArgs(456,
    987, 321, 0);
121         Sequence<Integer> expectedSeq1 =
    this.createFromArgs(456, 987, 321, 0);
122         Sequence<Integer> seq2 = this.createFromArgs();
123         Sequence<Integer> expectedSeq2 =
    this.createFromArgs(721, 654, 160);
124         SequenceSmooth.smooth(seq1, seq2);
125         /*
126          * Assert that values of variables match
    expectations
127          */
128         assertEquals(expectedSeq1, seq1);
129         assertEquals(expectedSeq2, seq2);
130     }
131
132 }
133
```