```
1 import static org.junit.Assert.assertEquals;
3import org.junit.Test;
5 import components.set.Set;
6 import components.set.Set1L;
7 import components.simplereader.SimpleReader;
8 import components.simplereader.SimpleReader1L;
9import components.simplewriter.SimpleWriter;
10 import components.simplewriter.SimpleWriter1L;
12 public class StringReassemblyTest {
13
14
      @Test
15
      public void testOverlap_stream() {
          String str1 = "broadcast";
16
17
          String str2 = "casting";
18
          int overlap = StringReassembly.overlap(str1, str2);
19
          assertEquals(4, overlap);
20
      }
21
22
      @Test
23
      public void testOverlap_prefixsuffix() {
          String str1 = "prefixsuffix";
24
          String str2 = "ixsuffixplus";
25
26
          int overlap = StringReassembly.overlap(str1, str2);
27
          assertEquals(8, overlap);
28
      }
29
30
      @Test
31
      public void testOverlap_nointersect() {
          String str1 = "nointersect";
32
33
          String str2 = "completely";
34
          int overlap = StringReassembly.overlap(str1, str2);
35
          assertEquals(0, overlap);
36
      }
37
38
      @Test
39
      public void testCombination_stream() {
40
          String str1 = "stre";
41
          String str2 = "ream";
          int overlap = 2;
42
43
          String combine = StringReassembly.combination(str1, str2, overlap);
44
          assertEquals("stream", combine);
45
      }
46
47
      @Test
48
      public void testCombination_prefixsuffix() {
49
          String str1 = "overlap";
50
          String str2 = "lapping";
51
          int overlap = 3;
52
          String combine = StringReassembly.combination(str1, str2, overlap);
53
          assertEquals("overlapping", combine);
54
      }
55
56
      @Test
57
      public void testAddToSetAvoidingSubstrings_1() {
```

```
58
           Set<String> test = new Set1L<>();
 59
           test.add("abc");
 60
           test.add("def");
           test.add("jkl");
 61
 62
           String str = "jklmn";
 63
           Set<String> expected = new Set1L<>();
 64
           expected.add("abc");
           expected.add("def");
 65
           expected.add("jklmn");
 66
 67
           StringReassembly.addToSetAvoidingSubstrings(test, str);
 68
           assertEquals(expected, test);
 69
       }
 70
 71
       @Test
 72
       public void testAddToSetAvoidingSubstrings_2() {
 73
           Set<String> test = new Set1L<>();
           test.add("123");
 74
           test.add("456");
 75
 76
           String str = "456789";
 77
           Set<String> expected = new Set1L<>();
 78
           expected.add("123");
 79
           expected.add("456789");
 80
           StringReassembly.addToSetAvoidingSubstrings(test, str);
 81
           assertEquals(expected, test);
 82
       }
 83
 84
       @Test
 85
       public void testAssemble_1() {
 86
           Set<String> test = new Set1L<>();
 87
           test.add("part1 ");
 88
           test.add("1 part2");
 89
           test.add("part2 part3");
 90
           Set<String> expected = new Set1L<>();
 91
           expected.add("part1 part2 part3");
 92
           StringReassembly.assemble(test);
 93
           assertEquals(expected, test);
 94
       }
95
 96
       @Test
 97
       public void testAssemble_2() {
 98
           Set<String> test = new Set1L<>();
           test.add("123");
99
           test.add("234");
100
           test.add("345");
101
102
           test.add("456");
103
           Set<String> expected = new Set1L<>();
104
           expected.add("123456");
105
           StringReassembly.assemble(test);
106
           assertEquals(expected, test);
107
       }
108
109
       @Test
110
       public void testPrintWithLineSeparators_1() {
111
           SimpleWriter out = new SimpleWriter1L("test.txt");
112
           SimpleReader in = new SimpleReader1L("test.txt");
113
           String text = "ABC~DEF~GHI";
114
           String expected = "ABC\nDEF\nGHI";
```

```
115
           StringReassembly.printWithLineSeparators(text, out);
116
           String test1 = in.nextLine();
117
           String test2 = in.nextLine();
118
           String test3 = in.nextLine();
119
           in.close();
120
           out.close();
           assertEquals(expected, test1 + "\n" + test2 + "\n" + test3);
121
122
       }
123
124
       @Test
125
       public void testPrintWithLineSeparators 2() {
126
           SimpleWriter out = new SimpleWriter1L("test.txt");
127
           SimpleReader in = new SimpleReader1L("test.txt");
           String text = "123~456~789";
128
129
           String expected = "123\n456\n789";
           StringReassembly.printWithLineSeparators(text, out);
130
131
           String test1 = in.nextLine();
132
           String test2 = in.nextLine();
133
           String test3 = in.nextLine();
134
           in.close();
135
           out.close();
           assertEquals(expected, test1 + "\n" + test2 + "\n" + test3);
136
137
       }
138
139
       @Test
140
       public void testLinesFromInput 1() {
141
           SimpleReader in = new SimpleReader1L("xd.txt");
142
           Set<String> test = new Set1L<>();
143
           Set<String> expected = new Set1L<>();
144
           expected.add("Bucks -- Beat");
145
           expected.add("Go Bucks");
146
           expected.add("o Bucks -- B");
           expected.add("Beat Mich");
147
           expected.add("Michigan~");
148
149
           test = StringReassembly.linesFromInput(in);
150
           assertEquals(test, expected);
151
       }
152 }
153
```