```
 1 import static org.junit.Assert.assertEquals;
 2
 3 import org.junit.Test;
 4
 5 import components.set.Set;
 6 import components.set.Set1L;
 7 import components.simplereader.SimpleReader;
 8 import components.simplereader.SimpleReader1L;
 9 import components.simplewriter.SimpleWriter;
10 import components.simplewriter.SimpleWriter1L;
11
12 public class StringReassemblyTest {
13
14     @Test
15     public void testCombination_Short2() {
16         String s1 = "Mi";
17         String s2 = "iami";
18         int overlap = StringReassembly.overlap(s1, s2);
19         String ans = StringReassembly.combination(s1, s2,
   overlap);
20         String ansExpected = "Miami";
21
22         assertEquals(ans, ansExpected);
23     }
24
25     @Test
26     public void testCombination_Long1() {
27         String s1 = "Four score, and seven years ago, our";
28         String s2 = "years ago, our fathers brought forth";
29         int overlap = StringReassembly.overlap(s1, s2);
30         String ans = StringReassembly.combination(s1, s2,
   overlap);
31         String ansExpected = "Four score, and seven years ago, our
   fathers brought forth";
32
33         assertEquals(ans, ansExpected);
34     }
35
36     @Test
37     public void testCombination_NoOverlap() {
38         String s1 = "Software ";
39         String s2 = "1";
40         int overlap = StringReassembly.overlap(s1, s2);
41         String ans = StringReassembly.combination(s1, s2,
```

```
   overlap);
42         String ansExpected = "Software 1";
43
44         assertEquals(ans, ansExpected);
45     }
46
47     @Test
48     public void testCombination_Short1() {
49         String s1 = "Hel";
50         String s2 = "ello";
51         int overlap = StringReassembly.overlap(s1, s2);
52         String ans = StringReassembly.combination(s1, s2,
   overlap);
53         String ansExpected = "Hello";
54
55         assertEquals(ans, ansExpected);
56     }
57
58     @Test
59     public void testAddToSetAvoidingSubstrings_AddingAString() {
60         Set<String> s = new Set1L<>();
61         s.add("Linear Algebra");
62         String str = "Foundations 1: Discrete Structures";
63
64         StringReassembly.addToSetAvoidingSubstrings(s, str);
65
66         Set<String> sExpected = new Set1L<>();
67         sExpected.add("Foundations 1: Discrete Structures");
68         sExpected.add("Linear Algebra");
69
70         assertEquals(s, sExpected);
71     }
72
73     @Test
74     public void testAddToSetAvoidingSubstrings_Superstring2() {
75         Set<String> s = new Set1L<>();
76         s.add(" Disc");
77         String str = "Foundations 1: Discrete Structures";
78
79         StringReassembly.addToSetAvoidingSubstrings(s, str);
80
81         Set<String> sExpected = new Set1L<>();
82         sExpected.add("Foundations 1: Discrete Structures");
83
```

```
 84         assertEquals(s, sExpected);
 85     }
 86
 87     @Test
 88     public void testAddToSetAvoidingSubstrings_Superstring1() {
 89         Set<String> s = new Set1L<>();
 90         s.add("Chinese 1102.51");
 91         s.add("Foundations");
 92         s.add("Calculus 3");
 93         String str = "Foundations 1: Discrete Structures";
 94
 95         StringReassembly.addToSetAvoidingSubstrings(s, str);
 96
 97         Set<String> sExpected = new Set1L<>();
 98         sExpected.add("Foundations 1: Discrete Structures");
 99         sExpected.add("Calculus 3");
100         sExpected.add("Chinese 1102.51");
101
102         assertEquals(s, sExpected);
103     }
104
105     @Test
106     public void testAddToSetAvoidingSubstrings_EqualStrings() {
107         Set<String> s = new Set1L<>();
108         s.add("Hello");
109         String str = "Hello";
110
111         StringReassembly.addToSetAvoidingSubstrings(s, str);
112
113         Set<String> sExpected = new Set1L<>();
114         sExpected.add("Hello");
115
116         assertEquals(s, sExpected);
117     }
118
119     @Test
120     public void testAddToSetAvoidingSubstrings_Substring2() {
121         Set<String> s = new Set1L<>();
122         s.add("Hello");
123         String str = "l";
124
125         StringReassembly.addToSetAvoidingSubstrings(s, str);
126
127         Set<String> sExpected = new Set1L<>();
```

```
128         sExpected.add("Hello");
129
130         assertEquals(s, sExpected);
131     }
132
133     @Test
134     public void testAddToSetAvoidingSubstrings_Substring1() {
135         Set<String> s = new Set1L<>();
136         s.add("Software 1 is fun!");
137         s.add("Calculus 3 is not.");
138         String str = "Software 1";
139
140         StringReassembly.addToSetAvoidingSubstrings(s, str);
141
142         Set<String> sExpected = new Set1L<>();
143         sExpected.add("Calculus 3 is not.");
144         sExpected.add("Software 1 is fun!");
145
146         assertEquals(s, sExpected);
147     }
148
149     @Test
150     public void testAddToSetAvoidingSubstrings_EmptySet() {
151         Set<String> s = new Set1L<>();
152         String str = "Software 1";
153
154         StringReassembly.addToSetAvoidingSubstrings(s, str);
155
156         Set<String> sExpected = new Set1L<>();
157         sExpected.add(str);
158
159         assertEquals(s, sExpected);
160     }
161
162     @Test
163     public void testLinesFromInput_NoNewLines() {
164         Set<String> lines = new Set1L<>();
165         SimpleWriter out = new SimpleWriter1L("testLines1.txt");
166         out.print("HelloellolloMy");
167         SimpleReader in = new SimpleReader1L("testLines1.txt");
168         lines = StringReassembly.linesFromInput(in);
169
170         Set<String> linesExpected = new Set1L<>();
171         linesExpected.add("HelloellolloMy");
```

```
172
173         assertEquals(lines, linesExpected);
174     }
175
176     @Test
177     public void testLinesFromInput_SeveralRepeats() {
178         Set<String> lines = new Set1L<>();
179         SimpleWriter out = new SimpleWriter1L("testLines1.txt");
180         out.print("Hello\nello\nllo\nMy");
181         SimpleReader in = new SimpleReader1L("testLines1.txt");
182         lines = StringReassembly.linesFromInput(in);
183
184         Set<String> linesExpected = new Set1L<>();
185         linesExpected.add("Hello");
186         linesExpected.add("My");
187
188         assertEquals(linesExpected, lines);
189     }
190
191     @Test
192     public void testLinesFromInput_NoRepeats() {
193         Set<String> lines = new Set1L<>();
194         SimpleWriter out = new SimpleWriter1L("testLines1.txt");
195         out.print("Hello\nMy\nName");
196         SimpleReader in = new SimpleReader1L("testLines1.txt");
197         lines = StringReassembly.linesFromInput(in);
198
199         Set<String> linesExpected = new Set1L<>();
200         linesExpected.add("Hello");
201         linesExpected.add("My");
202         linesExpected.add("Name");
203
204         assertEquals(lines, linesExpected);
205     }
206
207     @Test
208     public void testPrintWithLineSeparators_NoSquigglies() {
209         String text = "Hello, my name is Isaac";
210         SimpleWriter out = new SimpleWriter1L("testPrint1.txt");
211
212         StringReassembly.printWithLineSeparators(text, out);
213         SimpleReader in = new SimpleReader1L("testPrint1.txt");
214
215         assertEquals(text, in.nextLine());
```

```
216
217         in.close();
218         out.close();
219     }
220
221     @Test
222     public void testPrintWithLineSeparators_MiddleSquiggly() {
223         String text = "Hello!~My name is Isaac.";
224         SimpleWriter out = new SimpleWriter1L("testPrint1.txt");
225
226         StringReassembly.printWithLineSeparators(text, out);
227         SimpleReader in = new SimpleReader1L("testPrint1.txt");
228
229         String l1 = "Hello!";
230         String l2 = "My name is Isaac.";
231
232         assertEquals(l1, in.nextLine());
233         assertEquals(l2, in.nextLine());
234
235         in.close();
236         out.close();
237     }
238
239     @Test
240     public void testPrintWithLineSeparators_BeginningSquiggly() {
241         String text = "~Ni Hao";
242         SimpleWriter out = new SimpleWriter1L("testPrint1.txt");
243
244         StringReassembly.printWithLineSeparators(text, out);
245         SimpleReader in = new SimpleReader1L("testPrint1.txt");
246
247         String l1 = "";
248         String l2 = "Ni Hao";
249
250         assertEquals(l1, in.nextLine());
251         assertEquals(l2, in.nextLine());
252
253         in.close();
254         out.close();
255     }
256
257     @Test
258     public void testPrintWithLineSeparators_EndingSquiggly() {
259         String text = "Hola~";
```

```
260         SimpleWriter out = new SimpleWriter1L("testPrint1.txt");
261
262         StringReassembly.printWithLineSeparators(text, out);
263         SimpleReader in = new SimpleReader1L("testPrint1.txt");
264
265         String l1 = "Hola";
266
267         assertEquals(l1, in.nextLine());
268
269         in.close();
270         out.close();
271     }
272
273     @Test
274     public void testPrintWithLineSeparators_SeveralSquigglies() {
275         String text = "S~W~~1";
276         SimpleWriter out = new SimpleWriter1L("testPrint1.txt");
277
278         StringReassembly.printWithLineSeparators(text, out);
279         SimpleReader in = new SimpleReader1L("testPrint1.txt");
280
281         String l1 = "S";
282         String l2 = "W";
283         String l3 = "";
284         String l4 = "1";
285
286         assertEquals(l1, in.nextLine());
287         assertEquals(l2, in.nextLine());
288         assertEquals(l3, in.nextLine());
289         assertEquals(l4, in.nextLine());
290
291         in.close();
292         out.close();
293     }
294
295 }
296
```