```
 1 import static org.junit.Assert.assertEquals;
 6
 7 /**
 8  * JUnit test fixture for {@code List<String>}'s constructor and
   kernel methods.
 9  *
10  * @author Shyam Sai Bethina and Yihone Chu
11  *
12  */
13 public abstract class ListTest {
14
15     /**
16      * Invokes the appropriate {@code List} constructor for the
   implementation
17      * under test and returns the result.
18      *
19      * @return the new list
20      * @ensures constructorTest = (<>, <>)
21      */
22     protected abstract List<String> constructorTest();
23
24     /**
25      * Invokes the appropriate {@code List} constructor for the
   reference
26      * implementation and returns the result.
27      *
28      * @return the new list
29      * @ensures constructorRef = (<>, <>)
30      */
31     protected abstract List<String> constructorRef();
32
33     /**
34      * Constructs a {@code List<String>} with the entries in
   {@code args} and
35      * length of the left string equal to {@code leftLength}.
36      *
37      * @param list
38      *            the {@code List} to construct
39      * @param leftLength
40      *            the length of the left string in the constructed
   {@code List}
41      * @param args
42      *            the entries for the list
43      * @updates list
```

```
44      * @requires list = (<>, <>) and 0 <= leftLength <=
   args.length
45      * @ensures <pre>
46      * list = ([first leftLength entries in args], [remaining
   entries in args])
47      * </pre>
48      */
49     private void createFromArgsHelper(List<String> list, int
   leftLength,
50             String... args) {
51         for (String s : args) {
52             list.addRightFront(s);
53             list.advance();
54         }
55         list.moveToStart();
56         for (int i = 0; i < leftLength; i++) {
57             list.advance();
58         }
59     }
60
61     /**
62      * Creates and returns a {@code List<String>} of the
   implementation under
63      * test type with the given entries.
64      *
65      * @param leftLength
66      *            the length of the left string in the constructed
   {@code List}
67      * @param args
68      *            the entries for the list
69      * @return the constructed list
70      * @requires 0 <= leftLength <= args.length
71      * @ensures <pre>
72      * createFromArgs =
73      *   ([first leftLength entries in args], [remaining entries
   in args])
74      * </pre>
75      */
76     protected final List<String> createFromArgsTest(int
   leftLength,
77             String... args) {
78         assert 0 <= leftLength : "Violation of: 0 <= leftLength";
79         assert leftLength <= args.length : "Violation of:
   leftLength <= args.length";
```

```
 80         List<String> list = this.constructorTest();
 81         this.createFromArgsHelper(list, leftLength, args);
 82         return list;
 83     }
 84
 85     /**
 86      * Creates and returns a {@code List<String>} of the reference
 87      * implementation type with the given entries.
 88      *
 89      * @param leftLength
 90      *            the length of the left string in the constructed
    {@code List}
 91      * @param args
 92      *            the entries for the list
 93      * @return the constructed list
 94      * @requires 0 <= leftLength <= args.length
 95      * @ensures <pre>
 96      * createFromArgs =
 97      *   ([first leftLength entries in args], [remaining entries
    in args])
 98      * </pre>
 99      */
100     protected final List<String> createFromArgsRef(int leftLength,
101             String... args) {
102         assert 0 <= leftLength : "Violation of: 0 <= leftLength";
103         assert leftLength <= args.length : "Violation of:
    leftLength <= args.length";
104         List<String> list = this.constructorRef();
105         this.createFromArgsHelper(list, leftLength, args);
106         return list;
107     }
108
109     /*
110      * Test cases for constructor, addRightFront,
    removeRightFront, advance,
111      * moveToStart, leftLength, and rightLength.
112      */
113
114     @Test
115     public final void testConstructor() {
116         /*
117          * Set up variables and call method under test
118          */
119         List<String> list1 = this.constructorTest();
```

```
120         List<String> list2 = this.constructorRef();
121         /*
122          * Assert that values of variables match expectations
123          */
124         assertEquals(list2, list1);
125     }
126 
127     @Test
128     public final void testAddRightFrontLeftEmptyRightEmpty() {
129         /*
130          * Set up variables
131          */
132         List<String> list1 = this.createFromArgsTest(0);
133         List<String> list2 = this.createFromArgsRef(0, "red");
134         /*
135          * Call method under test
136          */
137         list1.addRightFront("red");
138         /*
139          * Assert that values of variables match expectations
140          */
141         assertEquals(list2, list1);
142     }
143 
144     @Test
145     public final void testAddRightFrontLeftEmptyRightNonEmpty() {
146         /*
147          * Set up variables
148          */
149         List<String> list1 = this.createFromArgsTest(0, "red",
    "blue");
150         List<String> list2 = this.createFromArgsRef(0, "green",
    "red", "blue");
151         /*
152          * Call method under test
153          */
154         list1.addRightFront("green");
155         /*
156          * Assert that values of variables match expectations
157          */
158         assertEquals(list2, list1);
159     }
160 
161     @Test
```

```
162     public final void testAddRightFrontLeftNonEmptyRightEmpty() {
163         /*
164          * Set up variables
165          */
166         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
167                 "purple");
168         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
169                 "purple", "red");
170         /*
171          * Call method under test
172          */
173         list1.addRightFront("red");
174         /*
175          * Assert that values of variables match expectations
176          */
177         assertEquals(list2, list1);
178     }
179 
180     @Test
181     public final void testAddRightFrontLeftNonEmptyRightNonEmpty()
    {
182         /*
183          * Set up variables
184          */
185         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
186                 "purple");
187         List<String> list2 = this.createFromArgsRef(2, "yellow",
    "orange",
188                 "green", "purple");
189         /*
190          * Call method under test
191          */
192         list1.addRightFront("green");
193         /*
194          * Assert that values of variables match expectations
195          */
196         assertEquals(list2, list1);
197     }
198 
199     @Test
200     public final void testRemoveRightFrontLeftEmptyRightOne() {
```

```
201         /*
202          * Set up variables
203          */
204         List<String> list1 = this.createFromArgsTest(0, "red");
205         List<String> list2 = this.createFromArgsRef(0);
206         /*
207          * Call method under test
208          */
209         String s = list1.removeRightFront();
210         /*
211          * Assert that values of variables match expectations
212          */
213         assertEquals("red", s);
214         assertEquals(list2, list1);
215     }
216 
217     @Test
218     public final void testRemoveRightFrontLeftEmptyRightNonEmpty()
    {
219         /*
220          * Set up variables
221          */
222         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
223         List<String> list2 = this.createFromArgsRef(0, "red",
    "blue");
224         /*
225          * Call method under test
226          */
227         String s = list1.removeRightFront();
228         /*
229          * Assert that values of variables match expectations
230          */
231         assertEquals("green", s);
232         assertEquals(list2, list1);
233     }
234 
235     @Test
236     public final void testRemoveRightFrontLeftNonEmptyRightOne() {
237         /*
238          * Set up variables
239          */
240         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
```

```
241                 "purple", "red");
242         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
243                 "purple");
244         /*
245          * Call method under test
246          */
247         String s = list1.removeRightFront();
248         /*
249          * Assert that values of variables match expectations
250          */
251         assertEquals("red", s);
252         assertEquals(list2, list1);
253     }
254 
255     @Test
256     public final void
    testRemoveRightFrontLeftNonEmptyRightNonEmpty() {
257         /*
258          * Set up variables
259          */
260         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
261                 "green", "purple");
262         List<String> list2 = this.createFromArgsRef(2, "yellow",
    "orange",
263                 "purple");
264         /*
265          * Call method under test
266          */
267         String s = list1.removeRightFront();
268         /*
269          * Assert that values of variables match expectations
270          */
271         assertEquals("green", s);
272         assertEquals(list2, list1);
273     }
274 
275     @Test
276     public final void testAdvanceLeftEmptyRightOne() {
277         /*
278          * Set up variables
279          */
280         List<String> list1 = this.createFromArgsTest(0, "red");
```

```
281         List<String> list2 = this.createFromArgsRef(1, "red");
282         /*
283          * Call method under test
284          */
285         list1.advance();
286         /*
287          * Assert that values of variables match expectations
288          */
289         assertEquals(list2, list1);
290     }
291 
292     @Test
293     public final void testAdvanceLeftEmptyRightNonEmpty() {
294         /*
295          * Set up variables
296          */
297         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
298         List<String> list2 = this.createFromArgsRef(1, "green",
    "red", "blue");
299         /*
300          * Call method under test
301          */
302         list1.advance();
303         /*
304          * Assert that values of variables match expectations
305          */
306         assertEquals(list2, list1);
307     }
308 
309     @Test
310     public final void testAdvanceLeftNonEmptyRightOne() {
311         /*
312          * Set up variables
313          */
314         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
315                 "purple", "red");
316         List<String> list2 = this.createFromArgsRef(4, "yellow",
    "orange",
317                 "purple", "red");
318         /*
319          * Call method under test
320          */
```

```
321         list1.advance();
322         /*
323          * Assert that values of variables match expectations
324          */
325         assertEquals(list2, list1);
326     }
327
328     @Test
329     public final void testAdvanceLeftNonEmptyRightNonEmpty() {
330         /*
331          * Set up variables
332          */
333         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
334                 "green", "purple");
335         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
336                 "green", "purple");
337         /*
338          * Call method under test
339          */
340         list1.advance();
341         /*
342          * Assert that values of variables match expectations
343          */
344         assertEquals(list2, list1);
345     }
346
347     @Test
348     public final void testMoveToStartLeftEmptyRightEmpty() {
349         /*
350          * Set up variables
351          */
352         List<String> list1 = this.createFromArgsTest(0);
353         List<String> list2 = this.createFromArgsRef(0);
354         /*
355          * Call method under test
356          */
357         list1.moveToStart();
358         /*
359          * Assert that values of variables match expectations
360          */
361         assertEquals(list2, list1);
362     }
```

```
363
364     @Test
365     public final void testMoveToStartLeftEmptyRightNonEmpty() {
366         /*
367          * Set up variables
368          */
369         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
370         List<String> list2 = this.createFromArgsRef(0, "green",
    "red", "blue");
371         /*
372          * Call method under test
373          */
374         list1.moveToStart();
375         /*
376          * Assert that values of variables match expectations
377          */
378         assertEquals(list2, list1);
379     }
380
381     @Test
382     public final void testMoveToStartLeftNonEmptyRightEmpty() {
383         /*
384          * Set up variables
385          */
386         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
387                 "purple");
388         List<String> list2 = this.createFromArgsRef(0, "yellow",
    "orange",
389                 "purple");
390         /*
391          * Call method under test
392          */
393         list1.moveToStart();
394         /*
395          * Assert that values of variables match expectations
396          */
397         assertEquals(list2, list1);
398     }
399
400     @Test
401     public final void testMoveToStartLeftNonEmptyRightNonEmpty() {
402         /*
```

```
403          * Set up variables
404          */
405         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
406                 "green", "purple");
407         List<String> list2 = this.createFromArgsRef(0, "yellow",
    "orange",
408                 "green", "purple");
409         list1.moveToStart();
410         /*
411          * Assert that values of variables match expectations
412          */
413         assertEquals(list2, list1);
414     }
415 
416     @Test
417     public final void testRightLengthLeftEmptyRightEmpty() {
418         /*
419          * Set up variables
420          */
421         List<String> list1 = this.createFromArgsTest(0);
422         List<String> list2 = this.createFromArgsRef(0);
423         /*
424          * Call method under test
425          */
426         int i = list1.rightLength();
427         /*
428          * Assert that values of variables match expectations
429          */
430         assertEquals(0, i);
431         assertEquals(list2, list1);
432     }
433 
434     @Test
435     public final void testRightLengthLeftEmptyRightNonEmpty() {
436         /*
437          * Set up variables
438          */
439         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
440         List<String> list2 = this.createFromArgsRef(0, "green",
    "red", "blue");
441         /*
442          * Call method under test
```

```
443          */
444         int i = list1.rightLength();
445         /*
446          * Assert that values of variables match expectations
447          */
448         assertEquals(3, i);
449         assertEquals(list2, list1);
450     }
451 
452     @Test
453     public final void testRightLengthLeftNonEmptyRightEmpty() {
454         /*
455          * Set up variables
456          */
457         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
458                 "purple");
459         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
460                 "purple");
461 
462         /*
463          * Call method under test
464          */
465         int i = list1.rightLength();
466         /*
467          * Assert that values of variables match expectations
468          */
469         assertEquals(0, i);
470         assertEquals(list2, list1);
471     }
472 
473     @Test
474     public final void testRightLengthLeftNonEmptyRightNonEmpty() {
475         /*
476          * Set up variables
477          */
478         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
479                 "green", "purple");
480         List<String> list2 = this.createFromArgsRef(2, "yellow",
    "orange",
481                 "green", "purple");
482         /*
```

```
483          * Call method under test
484          */
485         int i = list1.rightLength();
486         /*
487          * Assert that values of variables match expectations
488          */
489         assertEquals(2, i);
490         assertEquals(list2, list1);
491     }
492 
493     @Test
494     public final void testLeftLengthLeftEmptyRightEmpty() {
495         /*
496          * Set up variables
497          */
498         List<String> list1 = this.createFromArgsTest(0);
499         List<String> list2 = this.createFromArgsRef(0);
500         /*
501          * Call method under test
502          */
503         int i = list1.leftLength();
504         /*
505          * Assert that values of variables match expectations
506          */
507         assertEquals(0, i);
508         assertEquals(list2, list1);
509     }
510 
511     @Test
512     public final void testLeftLengthLeftEmptyRightNonEmpty() {
513         /*
514          * Set up variables
515          */
516         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
517         List<String> list2 = this.createFromArgsRef(0, "green",
    "red", "blue");
518         /*
519          * Call method under test
520          */
521         int i = list1.leftLength();
522         /*
523          * Assert that values of variables match expectations
524          */
```

```
525         assertEquals(0, i);
526         assertEquals(list2, list1);
527     }
528 
529     @Test
530     public final void testLeftLengthLeftNonEmptyRightEmpty() {
531         /*
532          * Set up variables
533          */
534         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
535                 "purple");
536         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
537                 "purple");
538         /*
539          * Call method under test
540          */
541         int i = list1.leftLength();
542         /*
543          * Assert that values of variables match expectations
544          */
545         assertEquals(3, i);
546         assertEquals(list2, list1);
547     }
548 
549     @Test
550     public final void testLeftLengthLeftNonEmptyRightNonEmpty() {
551         /*
552          * Set up variables
553          */
554         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
555                 "green", "purple");
556         List<String> list2 = this.createFromArgsRef(2, "yellow",
    "orange",
557                 "green", "purple");
558         /*
559          * Call method under test
560          */
561         int i = list1.leftLength();
562         /*
563          * Assert that values of variables match expectations
564          */
```

```
565         assertEquals(2, i);
566         assertEquals(list2, list1);
567     }
568 
569     /*
570      * Test cases for iterator.
571      */
572 
573     @Test
574     public final void testIteratorEmpty() {
575         /*
576          * Set up variables
577          */
578         List<String> list1 = this.createFromArgsTest(0);
579         List<String> list2 = this.createFromArgsRef(0);
580         List<String> list3 = this.createFromArgsRef(0);
581         /*
582          * Call method under test
583          */
584         for (String s : list1) {
585             list2.addRightFront(s);
586         }
587         /*
588          * Assert that values of variables match expectations
589          */
590         assertEquals(list3, list1);
591         assertEquals(list3, list2);
592     }
593 
594     @Test
595     public final void testIteratorOnlyRight() {
596         /*
597          * Set up variables
598          */
599         List<String> list1 = this.createFromArgsTest(0, "red",
    "blue");
600         List<String> list2 = this.createFromArgsRef(0);
601         List<String> list3 = this.createFromArgsRef(0, "red",
    "blue");
602         List<String> list4 = this.createFromArgsRef(0, "blue",
    "red");
603         /*
604          * Call method under test
605          */
```

```
606         for (String s : list1) {
607             list2.addRightFront(s);
608         }
609         /*
610          * Assert that values of variables match expectations
611          */
612         assertEquals(list3, list1);
613         assertEquals(list4, list2);
614     }
615 
616     @Test
617     public final void testIteratorOnlyLeft() {
618         /*
619          * Set up variables
620          */
621         List<String> list1 = this.createFromArgsTest(3, "red",
    "green", "blue");
622         List<String> list2 = this.createFromArgsRef(0);
623         List<String> list3 = this.createFromArgsRef(3, "red",
    "green", "blue");
624         List<String> list4 = this.createFromArgsRef(0, "blue",
    "green", "red");
625         /*
626          * Call method under test
627          */
628         for (String s : list1) {
629             list2.addRightFront(s);
630         }
631         /*
632          * Assert that values of variables match expectations
633          */
634         assertEquals(list3, list1);
635         assertEquals(list4, list2);
636     }
637 
638     @Test
639     public final void testIteratorLeftAndRight() {
640         /*
641          * Set up variables
642          */
643         List<String> list1 = this.createFromArgsTest(2, "purple",
    "red",
644                 "green", "blue", "yellow");
645         List<String> list2 = this.createFromArgsRef(0);
```

```
646         List<String> list3 = this.createFromArgsRef(2, "purple",
    "red", "green",
647                 "blue", "yellow");
648         List<String> list4 = this.createFromArgsRef(0, "yellow",
    "blue",
649                 "green", "red", "purple");
650         /*
651          * Call method under test
652          */
653         for (String s : list1) {
654             list2.addRightFront(s);
655         }
656         /*
657          * Assert that values of variables match expectations
658          */
659         assertEquals(list3, list1);
660         assertEquals(list4, list2);
661     }
662 
663     /*
664      * Test cases for other methods: moveToFinish
665      */
666 
667     @Test
668     public final void testMoveToFinishLeftEmptyRightEmpty() {
669         /*
670          * Set up variables
671          */
672         List<String> list1 = this.createFromArgsTest(0);
673         List<String> list2 = this.createFromArgsRef(0);
674         /*
675          * Call method under test
676          */
677         list1.moveToFinish();
678         /*
679          * Assert that values of variables match expectations
680          */
681         assertEquals(list2, list1);
682     }
683 
684     @Test
685     public final void testMoveToFinishLeftEmptyRightNonEmpty() {
686         /*
687          * Set up variables
```

```
688          */
689         List<String> list1 = this.createFromArgsTest(0, "green",
    "red", "blue");
690         List<String> list2 = this.createFromArgsRef(3, "green",
    "red", "blue");
691         /*
692          * Call method under test
693          */
694         list1.moveToFinish();
695         /*
696          * Assert that values of variables match expectations
697          */
698         assertEquals(list2, list1);
699     }
700 
701     @Test
702     public final void testMoveToFinishLeftNonEmptyRightEmpty() {
703         /*
704          * Set up variables
705          */
706         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
707                 "purple");
708         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
709                 "purple");
710         /*
711          * Call method under test
712          */
713         list1.moveToFinish();
714         /*
715          * Assert that values of variables match expectations
716          */
717         assertEquals(list2, list1);
718     }
719 
720     @Test
721     public final void testMoveToFinishLeftNonEmptyRightNonEmpty()
    {
722         /*
723          * Set up variables
724          */
725         List<String> list1 = this.createFromArgsTest(2, "yellow",
    "orange",
```

```
726                 "green", "purple");
727         List<String> list2 = this.createFromArgsRef(4, "yellow",
    "orange",
728                 "green", "purple");
729         /*
730          * Call method under test
731          */
732         list1.moveToFinish();
733         /*
734          * Assert that values of variables match expectations
735          */
736         assertEquals(list2, list1);
737     }
738
739     @Test
740     public final void testMoveToFinishShowBug() {
741         /*
742          * Set up variables
743          */
744         List<String> list1 = this.createFromArgsTest(0);
745         List<String> list2 = this.createFromArgsRef(0, "red");
746         /*
747          * Call method under test
748          */
749         list1.moveToFinish();
750         /*
751          * Evaluate the correctness of the result
752          */
753         list1.addRightFront("red");
754         assertEquals(list2, list1);
755     }
756
757     /*
758      * Edge case
759      */
760     @Test
761     public final void testRetreatLeftOneRightEmpty() {
762         /*
763          * Set up variables
764          */
765         List<String> list1 = this.createFromArgsTest(1, "red");
766         List<String> list2 = this.createFromArgsRef(0, "red");
767         /*
768          * Call method under test
```

```
769          */
770         list1.retreat();
771         /*
772          * Assert that values of variables match expectations
773          */
774         assertEquals(list2, list1);
775     }
776 
777     /*
778      * Challenging case
779      */
780     @Test
781     public final void testRetreatLeftNonEmptyRightEmpty() {
782         /*
783          * Set up variables
784          */
785         List<String> list1 = this.createFromArgsTest(1, "green",
    "red", "blue");
786         List<String> list2 = this.createFromArgsRef(0, "green",
    "red", "blue");
787         /*
788          * Call method under test
789          */
790         list1.retreat();
791         /*
792          * Assert that values of variables match expectations
793          */
794         assertEquals(list2, list1);
795     }
796 
797     /*
798      * Routine case
799      */
800     @Test
801     public final void testRetreatLeftNonEmptyRightOne() {
802         /*
803          * Set up variables
804          */
805         List<String> list1 = this.createFromArgsTest(4, "yellow",
    "orange",
806                 "purple", "red");
807         List<String> list2 = this.createFromArgsRef(3, "yellow",
    "orange",
808                 "purple", "red");
```

```
809         /*
810          * Call method under test
811          */
812         list1.retreat();
813         /*
814          * Assert that values of variables match expectations
815          */
816         assertEquals(list2, list1);
817     }
818 
819     /*
820      * Routine case
821      */
822     @Test
823     public final void testRetreatLeftNonEmptyRightNonEmpty() {
824         /*
825          * Set up variables
826          */
827         List<String> list1 = this.createFromArgsTest(3, "yellow",
    "orange",
828                 "green", "purple");
829         List<String> list2 = this.createFromArgsRef(2, "yellow",
    "orange",
830                 "green", "purple");
831         /*
832          * Call method under test
833          */
834         list1.retreat();
835         /*
836          * Assert that values of variables match expectations
837          */
838         assertEquals(list2, list1);
839     }
840 
841 }
842 
```