THE OHIO STATE UNIVERSITY

PROJECT 6: DOUBLY-LINKED LIST IMPLEMENTATION OF LIST WITH RETREAT

Daniil Gofman

Ansh Pachauri

SW 2: Dev & Dsgn

Paolo Bucci

Yiyang Chen

Shivam Gupta

October 23, 2023

```
 1 import static org.junit.Assert.assertEquals;
 2
 3 import org.junit.Test;
 4
 5 import components.list.List;
 6
 7 /**
 8  * JUnit test fixture for {@code List<String>}'s constructor and kernel
   methods.
 9  *
10  * @author Daniil Gofman, Ansh Pachauri
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          * Call method under test
463          */
464         int i = list1.rightLength();
465         /*
466          * Assert that values of variables match expectations
467          */
468         assertEquals(0, i);
469         assertEquals(list2, list1);
470     }
471
472     @Test
473     public final void testRightLengthLeftNonEmptyRightNonEmpty() {
474         /*
475          * Set up variables
476          */
477         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
478                 "green", "purple");
479         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
480                 "green", "purple");
481         /*
```

```
482         * Call method under test
483         */
484        int i = list1.rightLength();
485        /*
486         * Assert that values of variables match expectations
487         */
488        assertEquals(2, i);
489        assertEquals(list2, list1);
490    }
491
492    @Test
493    public final void testLeftLengthLeftEmptyRightEmpty() {
494        /*
495         * Set up variables
496         */
497        List<String> list1 = this.createFromArgsTest(0);
498        List<String> list2 = this.createFromArgsRef(0);
499        /*
500         * Call method under test
501         */
502        int i = list1.leftLength();
503        /*
504         * Assert that values of variables match expectations
505         */
506        assertEquals(0, i);
507        assertEquals(list2, list1);
508    }
509
510    @Test
511    public final void testLeftLengthLeftEmptyRightNonEmpty() {
512        /*
513         * Set up variables
514         */
515        List<String> list1 = this.createFromArgsTest(0, "green", "red",
   "blue");
516        List<String> list2 = this.createFromArgsRef(0, "green", "red",
   "blue");
517        /*
518         * Call method under test
519         */
520        int i = list1.leftLength();
521        /*
522         * Assert that values of variables match expectations
523         */
524        assertEquals(0, i);
525        assertEquals(list2, list1);
526    }
527
528    @Test
529    public final void testLeftLengthLeftNonEmptyRightEmpty() {
```

```
530         /*
531          * Set up variables
532          */
533         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
534                 "purple");
535         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
536                 "purple");
537         /*
538          * Call method under test
539          */
540         int i = list1.leftLength();
541         /*
542          * Assert that values of variables match expectations
543          */
544         assertEquals(3, i);
545         assertEquals(list2, list1);
546     }
547 
548     @Test
549     public final void testLeftLengthLeftNonEmptyRightNonEmpty() {
550         /*
551          * Set up variables
552          */
553         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
554                 "green", "purple");
555         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
556                 "green", "purple");
557         /*
558          * Call method under test
559          */
560         int i = list1.leftLength();
561         /*
562          * Assert that values of variables match expectations
563          */
564         assertEquals(2, i);
565         assertEquals(list2, list1);
566     }
567 
568     /*
569      * Test cases for iterator.
570      */
571 
572     @Test
573     public final void testIteratorEmpty() {
574         /*
575          * Set up variables
576          */
577         List<String> list1 = this.createFromArgsTest(0);
578         List<String> list2 = this.createFromArgsRef(0);
579         List<String> list3 = this.createFromArgsRef(0);
```

```
580         /*
581          * Call method under test
582          */
583         for (String s : list1) {
584             list2.addRightFront(s);
585         }
586         /*
587          * Assert that values of variables match expectations
588          */
589         assertEquals(list3, list1);
590         assertEquals(list3, list2);
591     }
592 
593     @Test
594     public final void testIteratorOnlyRight() {
595         /*
596          * Set up variables
597          */
598         List<String> list1 = this.createFromArgsTest(0, "red", "blue");
599         List<String> list2 = this.createFromArgsRef(0);
600         List<String> list3 = this.createFromArgsRef(0, "red", "blue");
601         List<String> list4 = this.createFromArgsRef(0, "blue", "red");
602         /*
603          * Call method under test
604          */
605         for (String s : list1) {
606             list2.addRightFront(s);
607         }
608         /*
609          * Assert that values of variables match expectations
610          */
611         assertEquals(list3, list1);
612         assertEquals(list4, list2);
613     }
614 
615     @Test
616     public final void testIteratorOnlyLeft() {
617         /*
618          * Set up variables
619          */
620         List<String> list1 = this.createFromArgsTest(3, "red", "green",
    "blue");
621         List<String> list2 = this.createFromArgsRef(0);
622         List<String> list3 = this.createFromArgsRef(3, "red", "green",
    "blue");
623         List<String> list4 = this.createFromArgsRef(0, "blue", "green",
    "red");
624         /*
625          * Call method under test
626          */
```

```
627         for (String s : list1) {
628             list2.addRightFront(s);
629         }
630         /*
631          * Assert that values of variables match expectations
632          */
633         assertEquals(list3, list1);
634         assertEquals(list4, list2);
635     }
636 
637     @Test
638     public final void testIteratorLeftAndRight() {
639         /*
640          * Set up variables
641          */
642         List<String> list1 = this.createFromArgsTest(2, "purple", "red",
643                 "green", "blue", "yellow");
644         List<String> list2 = this.createFromArgsRef(0);
645         List<String> list3 = this.createFromArgsRef(2, "purple", "red",
    "green",
646                 "blue", "yellow");
647         List<String> list4 = this.createFromArgsRef(0, "yellow", "blue",
648                 "green", "red", "purple");
649         /*
650          * Call method under test
651          */
652         for (String s : list1) {
653             list2.addRightFront(s);
654         }
655         /*
656          * Assert that values of variables match expectations
657          */
658         assertEquals(list3, list1);
659         assertEquals(list4, list2);
660     }
661 
662     /*
663      * Test cases for other methods: moveToFinish
664      */
665 
666     @Test
667     public final void testMoveToFinishLeftEmptyRightEmpty() {
668         /*
669          * Set up variables
670          */
671         List<String> list1 = this.createFromArgsTest(0);
672         List<String> list2 = this.createFromArgsRef(0);
673         /*
674          * Call method under test
675          */
```

```
676         list1.moveToFinish();
677         /*
678          * Assert that values of variables match expectations
679          */
680         assertEquals(list2, list1);
681     }
682 
683     @Test
684     public final void testMoveToFinishLeftEmptyRightNonEmpty() {
685         /*
686          * Set up variables
687          */
688         List<String> list1 = this.createFromArgsTest(0, "green", "red",
    "blue");
689         List<String> list2 = this.createFromArgsRef(3, "green", "red",
    "blue");
690         /*
691          * Call method under test
692          */
693         list1.moveToFinish();
694         /*
695          * Assert that values of variables match expectations
696          */
697         assertEquals(list2, list1);
698     }
699 
700     @Test
701     public final void testMoveToFinishLeftNonEmptyRightEmpty() {
702         /*
703          * Set up variables
704          */
705         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
706                 "purple");
707         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
708                 "purple");
709         /*
710          * Call method under test
711          */
712         list1.moveToFinish();
713         /*
714          * Assert that values of variables match expectations
715          */
716         assertEquals(list2, list1);
717     }
718 
719     @Test
720     public final void testMoveToFinishLeftNonEmptyRightNonEmpty() {
721         /*
722          * Set up variables
723          */
```

```
724         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
725                 "green", "purple");
726         List<String> list2 = this.createFromArgsRef(4, "yellow", "orange",
727                 "green", "purple");
728         /*
729          * Call method under test
730          */
731         list1.moveToFinish();
732         /*
733          * Assert that values of variables match expectations
734          */
735         assertEquals(list2, list1);
736     }
737
738     @Test
739     public final void testMoveToFinishShowBug() {
740         /*
741          * Set up variables
742          */
743         List<String> list1 = this.createFromArgsTest(0);
744         List<String> list2 = this.createFromArgsRef(0, "red");
745         /*
746          * Call method under test
747          */
748         list1.moveToFinish();
749         /*
750          * Evaluate the correctness of the result
751          */
752         list1.addRightFront("red");
753         assertEquals(list2, list1);
754     }
755
756     /*
757      * Test for Retreat Left empty
758      */
759     @Test
760     public final void testRetreatLeftEmpty() {
761         /*
762          * Set up variables
763          */
764         List<String> list1 = this.createFromArgsTest(1, "red", "yellow");
765         List<String> list2 = this.createFromArgsRef(0, "red", "yellow");
766         /*
767          * Call method under test
768          */
769         list1.retreat();
770         /*
771          * Evaluate the correctness of the result
772          */
773         assertEquals(list2, list1);
```

```
774     }
775
776     /*
777      * Test for Retreat Boundary
778      */
779     @Test
780     public final void testRetreatBoundary() {
781         /*
782          * Set up variables
783          */
784         List<String> list1 = this.createFromArgsTest(1, "red");
785         List<String> list2 = this.createFromArgsRef(0, "red");
786         /*
787          * Call method under test
788          */
789         list1.retreat();
790         /*
791          * Evaluate the correctness of the result
792          */
793         assertEquals(list2, list1);
794     }
795
796     /*
797      * Test for Retreat left non empty
798      */
799     @Test
800     public final void testRetreatLeftNonEmpty1() {
801         /*
802          * Set up variables
803          */
804         List<String> list1 = this.createFromArgsTest(3, "red", "yellow",
805                 "green");
806         List<String> list2 = this.createFromArgsRef(2, "red", "yellow",
807                 "green");
808         /*
809          * Call method under test
810          */
811         list1.retreat();
812         /*
813          * Evaluate the correctness of the result
814          */
815         assertEquals(list2, list1);
816     }
817
818     /*
819      * Test for Retreat left non empty
820      */
821     @Test
822     public final void testRetreatLeftNonEmpty2() {
823         /*
```

```
824          * Set up variables
825          */
826         List<String> list1 = this.createFromArgsTest(4, "red", "yellow",
827                 "green", "orange");
828         List<String> list2 = this.createFromArgsRef(3, "red", "yellow",
    "green",
829                 "orange");
830         /*
831          * Call method under test
832          */
833         list1.retreat();
834         /*
835          * Evaluate the correctness of the result
836          */
837         assertEquals(list2, list1);
838     }
839
840 }
841
```