```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code Program}'s constructor and kernel methods.
 *
 *
 * @author Robert Frenken
 * @author Bennett Palmer
 *
 */
public abstract class ProgramTest {

    /**
     * The name of a file containing a BL program.
     */
    private static final String FILE_NAME_1 = "data/program-sample.bl";

    private static final String FILE_NAME_2 = "data/program-sample2.bl";

    private static final String FILE_NAME_3 = "data/program-sample3.bl";

    /**
     * Invokes the {@code Program} constructor for the implementation under test
     * and returns the result.
     *
     * @return the new program
     * @ensures constructor = ("Unnamed", {}, compose((BLOCK, ?, ?), <>))
     */
    protected abstract Program constructorTest();

    /**
     * Invokes the {@code Program} constructor for the reference implementation
     * and returns the result.
     *
     * @return the new program
     * @ensures constructor = ("Unnamed", {}, compose((BLOCK, ?, ?), <>))
     */
    protected abstract Program constructorRef();

    /**
     *
     * Creates and returns a {@code Program}, of the type of the implementation
     * under test, from the file with the given name.
     *
     * @param filename
     *            the name of the file to be parsed to create the program
     * @return the constructed program
     * @ensures createFromFile = [the program as parsed from the file]
     */
    private Program createFromFileTest(String filename) {
        Program p = this.constructorTest();
        SimpleReader file = new SimpleReader1L(filename);
        p.parse(file);
        file.close();
        return p;
    }

```

```
67      /**
68       *
69       * Creates and returns a {@code Program}, of the reference implementation
70       * type, from the file with the given name.
71       *
72       * @param filename
73       *            the name of the file to be parsed to create the program
74       * @return the constructed program
75       * @ensures createFromFile = [the program as parsed from the file]
76       */
77      private Program createFromFileRef(String filename) {
78          Program p = this.constructorRef();
79          SimpleReader file = new SimpleReader1L(filename);
80          p.parse(file);
81          file.close();
82          return p;
83      }
84
85      /**
86       * Test constructor.
87       */
88      @Test
89      public final void testConstructor() {
90          /*
91           * Setup
92           */
93          Program pRef = this.constructorRef();
94
95          /*
96           * The call
97           */
98          Program pTest = this.constructorTest();
99
100         /*
101          * Evaluation
102          */
103         assertEquals(pRef, pTest);
104     }
105
106     // TEST FOR PROGRAM FILE 1
107     /**
108      * Test name.
109      */
110     @Test
111     public final void testName() {
112         /*
113          * Setup
114          */
115         Program pTest = this.createFromFileTest(FILE_NAME_1);
116         Program pRef = this.createFromFileRef(FILE_NAME_1);
117
118         /*
119          * The call
120          */
121         String result = pTest.name();
122
123         /*
```

```
124          * Evaluation
125          */
126         assertEquals(pRef, pTest);
127         assertEquals("Test", result);
128     }
129 
130     /**
131      * Test setName.
132      */
133     @Test
134     public final void testSetName() {
135         /*
136          * Setup
137          */
138         Program pTest = this.createFromFileTest(FILE_NAME_1);
139         Program pRef = this.createFromFileRef(FILE_NAME_1);
140         String newName = "Replacement";
141         pRef.setName(newName);
142 
143         /*
144          * The call
145          */
146         pTest.setName(newName);
147 
148         /*
149          * Evaluation
150          */
151         assertEquals(pRef, pTest);
152     }
153 
154     /**
155      * Test newContext.
156      */
157     @Test
158     public final void testNewContext() {
159         /*
160          * Setup
161          */
162         Program pTest = this.createFromFileTest(FILE_NAME_1);
163         Program pRef = this.createFromFileRef(FILE_NAME_1);
164         Map<String, Statement> cRef = pRef.newContext();
165 
166         /*
167          * The call
168          */
169         Map<String, Statement> cTest = pTest.newContext();
170 
171         /*
172          * Evaluation
173          */
174         assertEquals(pRef, pTest);
175         assertEquals(cRef, cTest);
176     }
177 
178     /**
179      * Test swapContext.
180      */
```

```
181     @Test
182     public final void testSwapContext() {
183         /*
184          * Setup
185          */
186         Program pTest = this.createFromFileTest(FILE_NAME_1);
187         Program pRef = this.createFromFileRef(FILE_NAME_1);
188         Map<String, Statement> contextRef = pRef.newContext();
189         Map<String, Statement> contextTest = pTest.newContext();
190         String oneName = "one";
191         pRef.swapContext(contextRef);
192         Pair<String, Statement> oneRef = contextRef.remove(oneName);
193         /* contextRef now has just "two" */
194         pRef.swapContext(contextRef);
195         /* pRef's context now has just "two" */
196         contextRef.add(oneRef.key(), oneRef.value());
197         /* contextRef now has just "one" */
198
199         /* Make the reference call, replacing, in pRef, "one" with "two": */
200         pRef.swapContext(contextRef);
201
202         pTest.swapContext(contextTest);
203         Pair<String, Statement> oneTest = contextTest.remove(oneName);
204         /* contextTest now has just "two" */
205         pTest.swapContext(contextTest);
206         /* pTest's context now has just "two" */
207         contextTest.add(oneTest.key(), oneTest.value());
208         /* contextTest now has just "one" */
209
210         /*
211          * The call
212          */
213         pTest.swapContext(contextTest);
214
215         /*
216          * Evaluation
217          */
218         assertEquals(pRef, pTest);
219         assertEquals(contextRef, contextTest);
220     }
221
222     /**
223      * Test newBody.
224      */
225     @Test
226     public final void testNewBody() {
227         /*
228          * Setup
229          */
230         Program pTest = this.createFromFileTest(FILE_NAME_1);
231         Program pRef = this.createFromFileRef(FILE_NAME_1);
232         Statement bRef = pRef.newBody();
233
234         /*
235          * The call
236          */
237         Statement bTest = pTest.newBody();
```

```
238
239         /*
240          * Evaluation
241          */
242         assertEquals(pRef, pTest);
243         assertEquals(bRef, bTest);
244     }
245
246     /**
247      * Test swapBody.
248      */
249     @Test
250     public final void testSwapBody() {
251         /*
252          * Setup
253          */
254         Program pTest = this.createFromFileTest(FILE_NAME_1);
255         Program pRef = this.createFromFileRef(FILE_NAME_1);
256         Statement bodyRef = pRef.newBody();
257         Statement bodyTest = pTest.newBody();
258         pRef.swapBody(bodyRef);
259         Statement firstRef = bodyRef.removeFromBlock(0);
260         /* bodyRef now lacks the first statement */
261         pRef.swapBody(bodyRef);
262         /* pRef's body now lacks the first statement */
263         bodyRef.addToBlock(0, firstRef);
264         /* bodyRef now has just the first statement */
265
266         /* Make the reference call, replacing, in pRef, remaining with first: */
267         pRef.swapBody(bodyRef);
268
269         pTest.swapBody(bodyTest);
270         Statement firstTest = bodyTest.removeFromBlock(0);
271         /* bodyTest now lacks the first statement */
272         pTest.swapBody(bodyTest);
273         /* pTest's body now lacks the first statement */
274         bodyTest.addToBlock(0, firstTest);
275         /* bodyTest now has just the first statement */
276
277         /*
278          * The call
279          */
280         pTest.swapBody(bodyTest);
281
282         /*
283          * Evaluation
284          */
285         assertEquals(pRef, pTest);
286         assertEquals(bodyRef, bodyTest);
287     }
288
289     // TEST FOR PROGRAM FILE 2
290     /**
291      * Test name.
292      */
293     @Test
294     public final void testName2() {
```

```
295         /*
296          * Setup
297          */
298         Program pTest = this.createFromFileTest(FILE_NAME_2);
299         Program pRef = this.createFromFileRef(FILE_NAME_2);
300 
301         /*
302          * The call
303          */
304         String result = pTest.name();
305 
306         /*
307          * Evaluation
308          */
309         assertEquals(pRef, pTest);
310         assertEquals("Test", result);
311     }
312 
313     /**
314      * Test setName.
315      */
316     @Test
317     public final void testSetName2() {
318         /*
319          * Setup
320          */
321         Program pTest = this.createFromFileTest(FILE_NAME_2);
322         Program pRef = this.createFromFileRef(FILE_NAME_2);
323         String newName = "Replacement";
324         pRef.setName(newName);
325 
326         /*
327          * The call
328          */
329         pTest.setName(newName);
330 
331         /*
332          * Evaluation
333          */
334         assertEquals(pRef, pTest);
335     }
336 
337     /**
338      * Test newContext.
339      */
340     @Test
341     public final void testNewContext2() {
342         /*
343          * Setup
344          */
345         Program pTest = this.createFromFileTest(FILE_NAME_2);
346         Program pRef = this.createFromFileRef(FILE_NAME_2);
347         Map<String, Statement> cRef = pRef.newContext();
348 
349         /*
350          * The call
351          */
```

```
352         Map<String, Statement> cTest = pTest.newContext();
353
354         /*
355          * Evaluation
356          */
357         assertEquals(pRef, pTest);
358         assertEquals(cRef, cTest);
359     }
360
361     /**
362      * Test swapContext.
363      */
364     @Test
365     public final void testSwapContext2() {
366         /*
367          * Setup
368          */
369         Program pTest = this.createFromFileTest(FILE_NAME_2);
370         Program pRef = this.createFromFileRef(FILE_NAME_2);
371         Map<String, Statement> contextRef = pRef.newContext();
372         Map<String, Statement> contextTest = pTest.newContext();
373         String oneName = "one";
374         pRef.swapContext(contextRef);
375         Pair<String, Statement> oneRef = contextRef.remove(oneName);
376         /* contextRef now has just "two" */
377         pRef.swapContext(contextRef);
378         /* pRef's context now has just "two" */
379         contextRef.add(oneRef.key(), oneRef.value());
380         /* contextRef now has just "one" */
381
382         /* Make the reference call, replacing, in pRef, "one" with "two": */
383         pRef.swapContext(contextRef);
384
385         pTest.swapContext(contextTest);
386         Pair<String, Statement> oneTest = contextTest.remove(oneName);
387         /* contextTest now has just "two" */
388         pTest.swapContext(contextTest);
389         /* pTest's context now has just "two" */
390         contextTest.add(oneTest.key(), oneTest.value());
391         /* contextTest now has just "one" */
392
393         /*
394          * The call
395          */
396         pTest.swapContext(contextTest);
397
398         /*
399          * Evaluation
400          */
401         assertEquals(pRef, pTest);
402         assertEquals(contextRef, contextTest);
403     }
404
405     /**
406      * Test newBody.
407      */
408     @Test
```

```
409     public final void testNewBody2() {
410         /*
411          * Setup
412          */
413         Program pTest = this.createFromFileTest(FILE_NAME_2);
414         Program pRef = this.createFromFileRef(FILE_NAME_2);
415         Statement bRef = pRef.newBody();
416 
417         /*
418          * The call
419          */
420         Statement bTest = pTest.newBody();
421 
422         /*
423          * Evaluation
424          */
425         assertEquals(pRef, pTest);
426         assertEquals(bRef, bTest);
427     }
428 
429     /**
430      * Test swapBody.
431      */
432     @Test
433     public final void testSwapBody2() {
434         /*
435          * Setup
436          */
437         Program pTest = this.createFromFileTest(FILE_NAME_2);
438         Program pRef = this.createFromFileRef(FILE_NAME_2);
439         Statement bodyRef = pRef.newBody();
440         Statement bodyTest = pTest.newBody();
441         pRef.swapBody(bodyRef);
442         Statement firstRef = bodyRef.removeFromBlock(0);
443         /* bodyRef now lacks the first statement */
444         pRef.swapBody(bodyRef);
445         /* pRef's body now lacks the first statement */
446         bodyRef.addToBlock(0, firstRef);
447         /* bodyRef now has just the first statement */
448 
449         /* Make the reference call, replacing, in pRef, remaining with first: */
450         pRef.swapBody(bodyRef);
451 
452         pTest.swapBody(bodyTest);
453         Statement firstTest = bodyTest.removeFromBlock(0);
454         /* bodyTest now lacks the first statement */
455         pTest.swapBody(bodyTest);
456         /* pTest's body now lacks the first statement */
457         bodyTest.addToBlock(0, firstTest);
458         /* bodyTest now has just the first statement */
459 
460         /*
461          * The call
462          */
463         pTest.swapBody(bodyTest);
464 
465         /*
```

```
466         * Evaluation
467         */
468        assertEquals(pRef, pTest);
469        assertEquals(bodyRef, bodyTest);
470    }
471
472    // TEST FOR PROGRAM FILE 3
473    /**
474     * Test name.
475     */
476    @Test
477    public final void testName3() {
478        /*
479         * Setup
480         */
481        Program pTest = this.createFromFileTest(FILE_NAME_3);
482        Program pRef = this.createFromFileRef(FILE_NAME_3);
483
484        /*
485         * The call
486         */
487        String result = pTest.name();
488
489        /*
490         * Evaluation
491         */
492        assertEquals(pRef, pTest);
493        assertEquals("Test", result);
494    }
495
496    /**
497     * Test setName.
498     */
499    @Test
500    public final void testSetName3() {
501        /*
502         * Setup
503         */
504        Program pTest = this.createFromFileTest(FILE_NAME_3);
505        Program pRef = this.createFromFileRef(FILE_NAME_3);
506        String newName = "Replacement";
507        pRef.setName(newName);
508
509        /*
510         * The call
511         */
512        pTest.setName(newName);
513
514        /*
515         * Evaluation
516         */
517        assertEquals(pRef, pTest);
518    }
519
520    /**
521     * Test newContext.
522     */
```

```
523     @Test
524     public final void testNewContext3() {
525         /*
526          * Setup
527          */
528         Program pTest = this.createFromFileTest(FILE_NAME_3);
529         Program pRef = this.createFromFileRef(FILE_NAME_3);
530         Map<String, Statement> cRef = pRef.newContext();
531 
532         /*
533          * The call
534          */
535         Map<String, Statement> cTest = pTest.newContext();
536 
537         /*
538          * Evaluation
539          */
540         assertEquals(pRef, pTest);
541         assertEquals(cRef, cTest);
542     }
543 
544     /**
545      * Test swapContext.
546      */
547     @Test
548     public final void testSwapContext3() {
549         /*
550          * Setup
551          */
552         Program pTest = this.createFromFileTest(FILE_NAME_3);
553         Program pRef = this.createFromFileRef(FILE_NAME_3);
554         Map<String, Statement> contextRef = pRef.newContext();
555         Map<String, Statement> contextTest = pTest.newContext();
556         String oneName = "FindObstacle";
557         pRef.swapContext(contextRef);
558         Pair<String, Statement> oneRef = contextRef.remove(oneName);
559         /* contextRef now has just "two" */
560         pRef.swapContext(contextRef);
561         /* pRef's context now has just "two" */
562         contextRef.add(oneRef.key(), oneRef.value());
563         /* contextRef now has just "one" */
564 
565         /* Make the reference call, replacing, in pRef, "one" with "two": */
566         pRef.swapContext(contextRef);
567 
568         pTest.swapContext(contextTest);
569         Pair<String, Statement> oneTest = contextTest.remove(oneName);
570         /* contextTest now has just "two" */
571         pTest.swapContext(contextTest);
572         /* pTest's context now has just "two" */
573         contextTest.add(oneTest.key(), oneTest.value());
574         /* contextTest now has just "one" */
575 
576         /*
577          * The call
578          */
579         pTest.swapContext(contextTest);
```

```
580
581         /*
582          * Evaluation
583          */
584         assertEquals(pRef, pTest);
585         assertEquals(contextRef, contextTest);
586     }
587
588     /**
589      * Test newBody.
590      */
591     @Test
592     public final void testNewBody3() {
593         /*
594          * Setup
595          */
596         Program pTest = this.createFromFileTest(FILE_NAME_3);
597         Program pRef = this.createFromFileRef(FILE_NAME_3);
598         Statement bRef = pRef.newBody();
599
600         /*
601          * The call
602          */
603         Statement bTest = pTest.newBody();
604
605         /*
606          * Evaluation
607          */
608         assertEquals(pRef, pTest);
609         assertEquals(bRef, bTest);
610     }
611
612     /**
613      * Test swapBody.
614      */
615     @Test
616     public final void testSwapBody3() {
617         /*
618          * Setup
619          */
620         Program pTest = this.createFromFileTest(FILE_NAME_3);
621         Program pRef = this.createFromFileRef(FILE_NAME_3);
622         Statement bodyRef = pRef.newBody();
623         Statement bodyTest = pTest.newBody();
624         pRef.swapBody(bodyRef);
625         Statement firstRef = bodyRef.removeFromBlock(0);
626         /* bodyRef now lacks the first statement */
627         pRef.swapBody(bodyRef);
628         /* pRef's body now lacks the first statement */
629         bodyRef.addToBlock(0, firstRef);
630         /* bodyRef now has just the first statement */
631
632         /* Make the reference call, replacing, in pRef, remaining with first: */
633         pRef.swapBody(bodyRef);
634
635         pTest.swapBody(bodyTest);
636         Statement firstTest = bodyTest.removeFromBlock(0);
```

```
637         /* bodyTest now lacks the first statement */
638         pTest.swapBody(bodyTest);
639         /* pTest's body now lacks the first statement */
640         bodyTest.addToBlock(0, firstTest);
641         /* bodyTest now has just the first statement */
642 
643         /*
644          * The call
645          */
646         pTest.swapBody(bodyTest);
647 
648         /*
649          * Evaluation
650          */
651         assertEquals(pRef, pTest);
652         assertEquals(bodyRef, bodyTest);
653     }
654 
655 }
656 
```