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

/**
 * Test class to test methods from Glossary utility class.
 *
 * @author Shyam Sai Bethina
 *
 */
public class GlossaryTest {

    /*
     * Tests of getTerms method
     */

    /**
     * Routine test case, three simple words.
     */
    @Test
    public void testGetTerms_3Words() {
        SimpleReader in = new SimpleReader1L("test/
testGetTerms1");
        Queue<String> words = Glossary.getTerms(in);
        Queue<String> expectedWords = new Queue1L<>();
        expectedWords.enqueue("hello");
        expectedWords.enqueue("bye");
        expectedWords.enqueue("goodbye");
        assertEquals(expectedWords, words);
        in.close();
    }

    /**
     * Boundary test case, singular letter.
     */
    @Test
    public void testGetTerms_singleLetter() {
        SimpleReader in = new SimpleReader1L("test/
testGetTerms2");
        Queue<String> words = Glossary.getTerms(in);
        Queue<String> expectedWords = new Queue1L<>();
        expectedWords.enqueue("l");
        assertEquals(expectedWords, words);
        in.close();
    }

```

```
54      /**
55       * Boundary test case, every line has space.
56       */
57      @Test
58      public void testGetTerms_lineSpace() {
59          SimpleReader in = new SimpleReader1L("test/
    testGetTerms3");
60          Queue<String> words = Glossary.getTerms(in);
61          Queue<String> expectedWords = new Queue1L<>();
62          assertEquals(expectedWords, words);
63          in.close();
64      }
65
66      /**
67       * Challenging test case, there are multiple empty lines
    between terms.
68       */
69      @Test
70      public void testGetTerms_multipleLines() {
71          SimpleReader in = new SimpleReader1L("test/
    testGetTerms4");
72          Queue<String> words = Glossary.getTerms(in);
73          Queue<String> expectedWords = new Queue1L<>();
74          expectedWords.enqueue("hello");
75          expectedWords.enqueue("bye");
76          expectedWords.enqueue("goodbye");
77          assertEquals(expectedWords, words);
78          in.close();
79      }
80
81      /*
82       * Tests of getDefinitions method
83       */
84
85      /**
86       * Routine test case, 3 definitions.
87       */
88      @Test
89      public void testGetDefinitions_3Definitions() {
90          SimpleReader in = new SimpleReader1L("test/testGetDef1");
91          Queue<String> definitions = Glossary.getDefinitions(in);
92          Queue<String> expectedDefinitions = new Queue1L<>();
93          expectedDefinitions.enqueue("A greeting");
94          expectedDefinitions.enqueue("A saying when leaving");
```

```
 95         expectedDefinitions.enqueue("A shorthand version of
    goodbye");
 96         assertEquals(expectedDefinitions, definitions);
 97         in.close();
 98     }
 99
100     /**
101      * Boundary test case, each definition line is only one
    character long.
102      */
103     @Test
104     public void testGetDefinitions_1Character() {
105         SimpleReader in = new SimpleReader1L("test/testGetDef2");
106         Queue<String> definitions = Glossary.getDefinitions(in);
107         Queue<String> expectedDefinitions = new Queue1L<>();
108         expectedDefinitions.enqueue(" l");
109         expectedDefinitions.enqueue(" m");
110         assertEquals(expectedDefinitions, definitions);
111         in.close();
112     }
113
114     /**
115      * Boundary test case, each line is part of one definition.
116      */
117     @Test
118     public void testGetDefinitions_1Definition() {
119         SimpleReader in = new SimpleReader1L("test/testGetDef3");
120         Queue<String> definitions = Glossary.getDefinitions(in);
121         Queue<String> expectedDefinitions = new Queue1L<>();
122         expectedDefinitions
123                 .enqueue("a greeting to say when meeting someone
    you know");
124         assertEquals(expectedDefinitions, definitions);
125         in.close();
126     }
127
128     /**
129      * Challenging test case, each definition as three or more
    lines.
130      */
131     @Test
132     public void testGetDefinitions_3Lines() {
133         SimpleReader in = new SimpleReader1L("test/testGetDef4");
134         Queue<String> definitions = Glossary.getDefinitions(in);
```

```
135         Queue<String> expectedDefinitions = new Queue1L<>();
136         expectedDefinitions
137                 .enqueue("a greeting to say when meeting someone
    you know");
138         expectedDefinitions
139                 .enqueue("a saying to say when leaving someone you
    know");
140         expectedDefinitions.enqueue("an abbreviated version of
    saying goodbye");
141
142         assertEquals(expectedDefinitions, definitions);
143         in.close();
144     }
145
146     /*
147      * Tests for pageForWords method
148      */
149
150     /**
151      * Boundary test case, 1 word.
152      */
153     @Test
154     public void testPageForWords_1Word() {
155         Queue<String> words = new Queue1L<>();
156         words.enqueue("testForPage1");
157         Queue<String> expectedWords = new Queue1L<>();
158         expectedWords.enqueue("testForPage1");
159
160         Queue<String> definitions = new Queue1L<>();
161         definitions.enqueue("A test");
162         Queue<String> expectedDefinitions = new Queue1L<>();
163         expectedDefinitions.enqueue("A test");
164
165         String location = "test";
166
167         Glossary.pageForWords(words, definitions, location);
168
169         SimpleReader in = new SimpleReader1L("test/
    testForPage1.html");
170         String result = "";
171         while (!in.atEOS()) {
172             result += in.nextLine();
173         }
174         String expectedResult = "<html>" + "   <head>"
```

```
175                 + "       <title>testForPage1</title>" + "   </
    head>"
176                 + "   <body>" + "       <h2>"
177                 + "           <b><i><font
    color='red'>testForPage1</font></i></b>"
178                 + "       </h2>" + "       <blockquote>A test</
    blockquote>"
179                 + "       <hr>"
180                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
181                 + "   </body>" + "</html>";
182
183         assertEquals(expectedWords, words);
184         assertEquals(expectedDefinitions, definitions);
185         assertEquals("test", location);
186         assertEquals(expectedResult, result);
187         in.close();
188     }
189
190     /**
191      * Routine test case, 2 word.
192      */
193     @Test
194     public void testPageForWords_2Word() {
195         Queue<String> words = new Queue1L<>();
196         words.enqueue("testUp");
197         words.enqueue("testDown");
198         Queue<String> expectedWords = new Queue1L<>();
199         expectedWords.enqueue("testUp");
200         expectedWords.enqueue("testDown");
201
202         Queue<String> definitions = new Queue1L<>();
203         definitions.enqueue("Going up");
204         definitions.enqueue("Going down");
205         Queue<String> expectedDefinitions = new Queue1L<>();
206         expectedDefinitions.enqueue("Going up");
207         expectedDefinitions.enqueue("Going down");
208
209         String location = "test";
210
211         Glossary.pageForWords(words, definitions, location);
212
213         SimpleReader in = new SimpleReader1L("test/testUp.html");
214         String result = "";
```

```
215         while (!in.atEOS()) {
216             result += in.nextLine();
217         }
218         String expectedResult = "<html>" + "   <head>"
219                 + "       <title>testUp</title>" + "   </head>" +
    "   <body>"
220                 + "       <h2>"
221                 + "           <b><i><font color='red'>testUp</
    font></i></b>"
222                 + "       </h2>" + "       <blockquote>Going up</
    blockquote>"
223                 + "       <hr>"
224                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
225                 + "   </body>" + "</html>";
226 
227         SimpleReader in2 = new SimpleReader1L("test/
    testDown.html");
228         String result2 = "";
229         while (!in2.atEOS()) {
230             result2 += in2.nextLine();
231         }
232         String expectedResult2 = "<html>" + "   <head>"
233                 + "       <title>testDown</title>" + "   </head>"
    + "   <body>"
234                 + "       <h2>"
235                 + "           <b><i><font color='red'>testDown</
    font></i></b>"
236                 + "       </h2>" + "       <blockquote>Going
    down</blockquote>"
237                 + "       <hr>"
238                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
239                 + "   </body>" + "</html>";
240 
241         assertEquals(expectedWords, words);
242         assertEquals(expectedDefinitions, definitions);
243         assertEquals("test", location);
244         assertEquals(expectedResult, result);
245         assertEquals(expectedResult2, result2);
246         in.close();
247         in2.close();
248     }
249 
```

```
250     /**
251      * Challenging test case, 3 words.
252      */
253     @Test
254     public void testPageForWords_3Words() {
255         Queue<String> words = new Queue1L<>();
256         words.enqueue("testHello");
257         words.enqueue("testGoodbye");
258         words.enqueue("testBye");
259         Queue<String> expectedWords = new Queue1L<>();
260         expectedWords.enqueue("testHello");
261         expectedWords.enqueue("testGoodbye");
262         expectedWords.enqueue("testBye");
263 
264         Queue<String> definitions = new Queue1L<>();
265         definitions.enqueue("a greeting to say when meeting
    someone you know");
266         definitions.enqueue("a saying to say when leaving someone
    you know");
267         definitions.enqueue("an abbreviated version of saying
    goodbye");
268         Queue<String> expectedDefinitions = new Queue1L<>();
269         expectedDefinitions
270                 .enqueue("a greeting to say when meeting someone
    you know");
271         expectedDefinitions
272                 .enqueue("a saying to say when leaving someone you
    know");
273         expectedDefinitions.enqueue("an abbreviated version of
    saying goodbye");
274 
275         String location = "test";
276 
277         Glossary.pageForWords(words, definitions, location);
278 
279         SimpleReader in1 = new SimpleReader1L("test/
    testHello.html");
280         String result1 = "";
281         while (!in1.atEOS()) {
282             result1 += in1.nextLine();
283         }
284         String expectedResult1 = "<html>" + "   <head>"
285                 + "       <title>testHello</title>" + "   </head>"
    + "   <body>"
```

```
286                 + "       <h2>"
287                 + "           <b><i><font color='red'>testHello</
    font></i></b>"
288                 + "       </h2>"
289                 + "       <blockquote>a greeting to say when
    meeting someone "
290                 + "you know</blockquote>" + "       <hr>"
291                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
292                 + "   </body>" + "</html>";
293
294         SimpleReader in2 = new SimpleReader1L("test/
    testGoodbye.html");
295         String result2 = "";
296         while (!in2.atEOS()) {
297             result2 += in2.nextLine();
298         }
299         String expectedResult2 = "<html>" + "   <head>"
300                 + "       <title>testGoodbye</title>" + "   </
    head>"
301                 + "   <body>" + "       <h2>"
302                 + "           <b><i><font
    color='red'>testGoodbye</font></i></b>"
303                 + "       </h2>"
304                 + "       <blockquote>a saying to say when leaving
    someone you "
305                 + "know</blockquote>" + "       <hr>"
306                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
307                 + "   </body>" + "</html>";
308
309         SimpleReader in3 = new SimpleReader1L("test/
    testBye.html");
310         String result3 = "";
311         while (!in3.atEOS()) {
312             result3 += in3.nextLine();
313         }
314         String expectedResult3 = "<html>" + "   <head>"
315                 + "       <title>testBye</title>" + "   </head>" +
    "   <body>"
316                 + "       <h2>"
317                 + "           <b><i><font color='red'>testBye</
    font></i></b>"
318                 + "       </h2>"
```

```
319                 + "       <blockquote>an abbreviated version of
    saying "
320                 + "goodbye</blockquote>" + "       <hr>"
321                 + "       <p>Return to <a
    href='index.html'>index</a>.</p>"
322                 + "   </body>" + "</html>";
323
324         assertEquals(expectedWords, words);
325         assertEquals(expectedDefinitions, definitions);
326         assertEquals("test", location);
327         assertEquals(expectedResult1, result1);
328         assertEquals(expectedResult2, result2);
329         assertEquals(expectedResult3, result3);
330         in1.close();
331         in2.close();
332         in3.close();
333     }
334
335     /*
336      * Test for outputHeader method - only had one test since
    there is only one
337      * possible solution which is to print out to the output file
    stream
338      */
339
340     /**
341      * Routine test case.
342      */
343     @Test
344     public void testOutputHeader() {
345         SimpleWriter out = new SimpleWriter1L("test/testHeader");
346         SimpleReader in = new SimpleReader1L("test/testHeader");
347         Glossary.outputHeader(out);
348         String result = "";
349         while (!in.atEOS()) {
350             result += in.nextLine();
351         }
352
353         String resultExpected = "<html>" + "   <head>"
354                 + "       <title>Glossary</title>" + "   </head>"
    + "   <body>"
355                 + "       <h2>Glossary</h2>" + "       <hr />"
356                 + "       <h3>Index</h3>" + "       <ul>";
357
```

```
358         assertEquals(resultExpected, result);
359         out.close();
360         in.close();
361     }
362 
363     /*
364      * Test for outputFooter method - only had one test since
    there is only one
365      * possible solution which is to print out to the output file
    stream
366      */
367 
368     /**
369      * Routine test case.
370      */
371     @Test
372     public void testOutputFooter() {
373         SimpleWriter out = new SimpleWriter1L("test/testFooter");
374         SimpleReader in = new SimpleReader1L("test/testFooter");
375         Glossary.outputFooter(out);
376         String result = "";
377         while (!in.atEOS()) {
378             result += in.nextLine();
379         }
380 
381         String resultExpected = "       </ul>" + "   </body>" +
    "</html>";
382 
383         assertEquals(resultExpected, result);
384         out.close();
385         in.close();
386     }
387 
388     /*
389      * Test for listForWords
390      */
391 
392     /**
393      * Routine test case, 2 words.
394      */
395     @Test
396     public void testListForWords() {
397         Queue<String> words = new Queue1L<>();
398         words.enqueue("hello");
```

```
399         words.enqueue("bye");
400         Queue<String> expectedWords = new Queue1L<>();
401         expectedWords.enqueue("bye");
402         expectedWords.enqueue("hello");
403 
404         Queue<String> definitions = new Queue1L<>();
405         definitions.enqueue("a greeting to say when meeting
    someone you know");
406         definitions.enqueue("a saying to say when leaving someone
    you know");
407         Queue<String> expectedDefinitions = new Queue1L<>();
408         expectedDefinitions
409                 .enqueue("a greeting to say when meeting someone
    you know");
410         expectedDefinitions
411                 .enqueue("a saying to say when leaving someone you
    know");
412 
413         SimpleWriter out = new SimpleWriter1L("test/testList1");
414         SimpleReader in = new SimpleReader1L("test/testList1");
415         Glossary.listForWords(out, words, definitions);
416 
417         String result = "";
418         while (!in.atEOS()) {
419             result += in.nextLine();
420         }
421 
422         String expectedResult = "           <li><a
    href=bye.html>bye</a></li>"
423                 + "           <li><a href=hello.html>hello</a></
    li>";
424 
425         assertEquals(expectedWords, words);
426         assertEquals(expectedDefinitions, definitions);
427         assertEquals(expectedResult, result);
428         out.close();
429         in.close();
430     }
431 
432     /**
433      * Boundary test case, 1 word.
434      */
435     @Test
436     public void testListForWords_1Word() {
```

```
437         Queue<String> words = new Queue1L<>();
438         words.enqueue("hello");
439         Queue<String> expectedWords = new Queue1L<>();
440         expectedWords.enqueue("hello");
441 
442         Queue<String> definitions = new Queue1L<>();
443         definitions.enqueue("a greeting to say when meeting
    someone you know");
444         Queue<String> expectedDefinitions = new Queue1L<>();
445         expectedDefinitions
446                 .enqueue("a greeting to say when meeting someone
    you know");
447 
448         SimpleWriter out = new SimpleWriter1L("test/testList2");
449         SimpleReader in = new SimpleReader1L("test/testList2");
450         Glossary.listForWords(out, words, definitions);
451 
452         String result = "";
453         while (!in.atEOS()) {
454             result += in.nextLine();
455         }
456 
457         String expectedResult = "           <li><a
    href=hello.html>hello</a></li>";
458 
459         assertEquals(expectedWords, words);
460         assertEquals(expectedDefinitions, definitions);
461         assertEquals(expectedResult, result);
462         out.close();
463         in.close();
464     }
465 
466     /**
467      * Challenging test case, 3 word, 2 are the same.
468      */
469     @Test
470     public void testListForWords_3words() {
471         Queue<String> words = new Queue1L<>();
472         words.enqueue("hello");
473         words.enqueue("bye");
474         words.enqueue("bye");
475         Queue<String> expectedWords = new Queue1L<>();
476         expectedWords.enqueue("bye");
477         expectedWords.enqueue("bye");
```

```
478         expectedWords.enqueue("hello");
479
480         Queue<String> definitions = new Queue1L<>();
481         definitions.enqueue("a greeting to say when meeting
    someone you know");
482         definitions.enqueue("an abbreviated version of goodbye");
483         definitions.enqueue("an abbreviated version of goodbye");
484         Queue<String> expectedDefinitions = new Queue1L<>();
485         expectedDefinitions
486                 .enqueue("a greeting to say when meeting someone
    you know");
487         expectedDefinitions.enqueue("an abbreviated version of
    goodbye");
488         expectedDefinitions.enqueue("an abbreviated version of
    goodbye");
489
490         SimpleWriter out = new SimpleWriter1L("test/testList3");
491         SimpleReader in = new SimpleReader1L("test/testList3");
492         Glossary.listForWords(out, words, definitions);
493
494         String result = "";
495         while (!in.atEOS()) {
496             result += in.nextLine();
497         }
498
499         String expectedResult = "           <li><a
    href=bye.html>bye</a></li>"
500                 + "           <li><a href=bye.html>bye</a></li>"
501                 + "           <li><a href=hello.html>hello</a></
    li>";
502
503         assertEquals(expectedWords, words);
504         assertEquals(expectedDefinitions, definitions);
505         assertEquals(expectedResult, result);
506         out.close();
507         in.close();
508     }
509
510     /*
511      * Tests for linkWordsInDefinition method
512      */
513     /**
514      * Routine test case, two words.
515      */
```

```
516     @Test
517     public void testLinkWordsInDefinition_2Words() {
518         Queue<String> words = new Queue1L<>();
519         words.enqueue("hello");
520         words.enqueue("bye");
521 
522         Queue<String> expectedWords = new Queue1L<>();
523         expectedWords.enqueue("hello");
524         expectedWords.enqueue("bye");
525 
526         String definition = "hello my name is sai";
527         String result = Glossary.linkWordsinDefinition(words,
    definition);
528 
529         String expectedResult = "<a href=hello.html>hello</a> my
    name is sai";
530 
531         assertEquals(expectedWords, words);
532         assertEquals(expectedResult, result);
533     }
534 
535     /**
536      * Boundary test case, 1 word.
537      */
538     @Test
539     public void testLinkWordsInDefinition_1Word() {
540         Queue<String> words = new Queue1L<>();
541         words.enqueue("hello");
542 
543         Queue<String> expectedWords = new Queue1L<>();
544         expectedWords.enqueue("hello");
545 
546         String definition = "hello my name is sai";
547         String result = Glossary.linkWordsinDefinition(words,
    definition);
548 
549         String expectedResult = "<a href=hello.html>hello</a> my
    name is sai";
550 
551         assertEquals(expectedWords, words);
552         assertEquals(expectedResult, result);
553     }
554 
555     /**
```

```
556      * Boundary test case, 1 word, no links.
557      */
558     @Test
559     public void testLinkWordsInDefinition_1WordNoLinks() {
560         Queue<String> words = new Queue1L<>();
561         words.enqueue("bye");
562 
563         Queue<String> expectedWords = new Queue1L<>();
564         expectedWords.enqueue("bye");
565 
566         String definition = "hello my name is sai";
567         String result = Glossary.linkWordsinDefinition(words,
    definition);
568 
569         String expectedResult = "hello my name is sai";
570 
571         assertEquals(expectedWords, words);
572         assertEquals(expectedResult, result);
573     }
574 
575     /**
576      * Challenging test case, 3 words, all linked.
577      */
578     @Test
579     public void testLinkWordsInDefinition_3WordsAllLinked() {
580         Queue<String> words = new Queue1L<>();
581         words.enqueue("bye");
582         words.enqueue("hello");
583         words.enqueue("goodbye");
584 
585         Queue<String> expectedWords = new Queue1L<>();
586         expectedWords.enqueue("bye");
587         expectedWords.enqueue("hello");
588         expectedWords.enqueue("goodbye");
589 
590         String definition1 = "hello my name is sai";
591         String result1 = Glossary.linkWordsinDefinition(words,
    definition1);
592 
593         String definition2 = "goodbye see you later";
594         String result2 = Glossary.linkWordsinDefinition(words,
    definition2);
595 
596         String definition3 = "bye!";
```

```
597         String result3 = Glossary.linkWordsinDefinition(words,
    definition3);
598 
599         String expectedResult1 = "<a href=hello.html>hello</a> my
    name is sai";
600         String expectedResult2 = "<a href=goodbye.html>goodbye</a>
    see you later";
601         String expectedResult3 = "<a href=bye.html>bye</a>!";
602 
603         assertEquals(expectedWords, words);
604         assertEquals(expectedResult1, result1);
605         assertEquals(expectedResult2, result2);
606         assertEquals(expectedResult3, result3);
607 
608     }
609 
610     /*
611      * Tests for generateElements method
612      */
613 
614     /**
615      * Routine test case, string with length 3 and all different
    characters.
616      */
617     @Test
618     public void testGenerateElements() {
619         Set<Character> charSet = new Set1L<>();
620         String test = "bye";
621         Glossary.generateElements(test, charSet);
622 
623         Set<Character> expectedCharSet = new Set1L<>();
624         expectedCharSet.add('b');
625         expectedCharSet.add('y');
626         expectedCharSet.add('e');
627 
628         String expectedTest = "bye";
629 
630         assertEquals(expectedCharSet, charSet);
631         assertEquals(expectedTest, test);
632     }
633 
634     /**
635      * Boundary test case, string with length 1.
636      */
```

```
637     @Test
638     public void testGenerateElements_1Letter() {
639         Set<Character> charSet = new Set1L<>();
640         String test = "b";
641         Glossary.generateElements(test, charSet);
642 
643         Set<Character> expectedCharSet = new Set1L<>();
644         expectedCharSet.add('b');
645 
646         String expectedTest = "b";
647 
648         assertEquals(expectedCharSet, charSet);
649         assertEquals(expectedTest, test);
650     }
651 
652     /**
653      * Challenging test case, string that is a sentence and has
    some repetitive
654      * characters, also has the same letter but uppercase and
    lowercase.
655      */
656     @Test
657     public void testGenerateElements_sentence() {
658         Set<Character> charSet = new Set1L<>();
659         String test = "Whatsup my name is sai, I like to win";
660         Glossary.generateElements(test, charSet);
661 
662         Set<Character> expectedCharSet = new Set1L<>();
663         expectedCharSet.add('W');
664         expectedCharSet.add('h');
665         expectedCharSet.add('a');
666         expectedCharSet.add('t');
667         expectedCharSet.add('s');
668         expectedCharSet.add('u');
669         expectedCharSet.add('p');
670         expectedCharSet.add(' ');
671         expectedCharSet.add('m');
672         expectedCharSet.add('y');
673         expectedCharSet.add('n');
674         expectedCharSet.add('e');
675         expectedCharSet.add('i');
676         expectedCharSet.add(',');
677         expectedCharSet.add('I');
678         expectedCharSet.add('l');
```

```
679         expectedCharSet.add('k');
680         expectedCharSet.add('o');
681         expectedCharSet.add('w');
682 
683         String expectedTest = "Whatsup my name is sai, I like to
    win";
684 
685         assertEquals(expectedCharSet, charSet);
686         assertEquals(expectedTest, test);
687     }
688 
689     /*
690      * Tests for nextWordOrSeparator
691      */
692 
693     /**
694      * Routine test case, simple sentence.
695      */
696     @Test
697     public void testNextWordOrSeparator_simple() {
698         final String separatorStr = " \t,!.?(){}[];:'";
699         Set<Character> separatorSet = new Set1L<>();
700         Glossary.generateElements(separatorStr, separatorSet);
701 
702         String test = "Bye, have a good day!";
703 
704         Queue<String> result = Glossary.nextWordOrSeparator(test,
    separatorSet);
705 
706         Queue<String> expectedResult = new Queue1L<>();
707         expectedResult.enqueue("Bye");
708         expectedResult.enqueue(", ");
709         expectedResult.enqueue("have");
710         expectedResult.enqueue(" ");
711         expectedResult.enqueue("a");
712         expectedResult.enqueue(" ");
713         expectedResult.enqueue("good");
714         expectedResult.enqueue(" ");
715         expectedResult.enqueue("day");
716         expectedResult.enqueue("!");
717 
718         String expectedTest = "Bye, have a good day!";
719 
720         assertEquals(expectedResult, result);
```

```
721         assertEquals(expectedTest, test);
722     }
723 
724     /**
725      * Boundary test case, only separators.
726      */
727     @Test
728     public void testNextWordOrSeparator_separators() {
729         final String separatorStr = " \t,!.?(){}[];:'";
730         Set<Character> separatorSet = new Set1L<>();
731         Glossary.generateElements(separatorStr, separatorSet);
732 
733         String test = "[]{}!.?";
734 
735         Queue<String> result = Glossary.nextWordOrSeparator(test,
    separatorSet);
736 
737         Queue<String> expectedResult = new Queue1L<>();
738         expectedResult.enqueue("[]{}!.?");
739 
740         String expectedTest = "[]{}!.?";
741 
742         assertEquals(expectedResult, result);
743         assertEquals(expectedTest, test);
744     }
745 
746     /**
747      * Boundary test case, only letters.
748      */
749     @Test
750     public void testNextWordOrSeparator_letters() {
751         final String separatorStr = " \t,!.?(){}[];:'";
752         Set<Character> separatorSet = new Set1L<>();
753         Glossary.generateElements(separatorStr, separatorSet);
754 
755         String test = "abcdefg";
756 
757         Queue<String> result = Glossary.nextWordOrSeparator(test,
    separatorSet);
758 
759         Queue<String> expectedResult = new Queue1L<>();
760         expectedResult.enqueue("abcdefg");
761 
762         String expectedTest = "abcdefg";
```

```
763
764         assertEquals(expectedResult, result);
765         assertEquals(expectedTest, test);
766     }
767
768     /**
769      * Challenging test case, only separators inside letters.
770      */
771     @Test
772     public void testNextWordOrSeparator_sepInsideLetters() {
773         final String separatorStr = " \t,!.?(){}[];:'";
774         Set<Character> separatorSet = new Set1L<>();
775         Glossary.generateElements(separatorStr, separatorSet);
776
777         String test = "he!?.llo m!y na:me";
778
779         Queue<String> result = Glossary.nextWordOrSeparator(test,
    separatorSet);
780
781         Queue<String> expectedResult = new Queue1L<>();
782         expectedResult.enqueue("he");
783         expectedResult.enqueue("!?.");
784         expectedResult.enqueue("llo");
785         expectedResult.enqueue(" ");
786         expectedResult.enqueue("m");
787         expectedResult.enqueue("!");
788         expectedResult.enqueue("y");
789         expectedResult.enqueue(" ");
790         expectedResult.enqueue("na");
791         expectedResult.enqueue(":");
792         expectedResult.enqueue("me");
793
794         String expectedTest = "he!?.llo m!y na:me";
795
796         assertEquals(expectedResult, result);
797         assertEquals(expectedTest, test);
798     }
799
800 }
801
```