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

import org.junit.Test;

import components.map.Map;
import components.map.Map1L;
import components.queue.Queue;
import components.queue.Queue1L;
import components.set.Set;
import components.set.Set1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;

/**
 *
 * @author Joe Fong
 *
 */

public class GlossaryTest {

    /*
     * Tests for nextWordOrSeparator
     */

    // tests on separator
    @Test
    public void test_nextWordOrSeparator_1() {
        String text = "lion, tiger, and fish";
        int position = 4;
        Set<Character> separators = new Set1L<>();
        separators.add(',');
        separators.add(' ');
        String testWord = Glossary.nextWordOrSeparator(text, position,
                separators);
        String expectedWord = ", ";
        assertEquals(testWord, expectedWord);
    }

    // tests on whole word
    @Test
    public void test_nextWordOrSeparator_2() {
        String text = "lion, tiger, and fish";
        int position = 6;
        Set<Character> separators = new Set1L<>();
        separators.add(',');
        separators.add(' ');
        String testWord = Glossary.nextWordOrSeparator(text, position,
                separators);
        String expectedWord = "tiger";
        assertEquals(testWord, expectedWord);
    }

    // tests at end of string
    @Test
    public void test_nextWordOrSeparator_3() {
        String text = "lion, tiger, and fish";
        int position = 18;
        Set<Character> separators = new Set1L<>();
```

```
60          separators.add(',');
61          separators.add(' ');
62          String testWord = Glossary.nextWordOrSeparator(text, position,
63                  separators);
64          String expectedWord = "ish";
65          assertEquals(testWord, expectedWord);
66      }
67
68      // tests at start of string
69      @Test
70      public void test_nextWordOrSeparator_4() {
71          String text = "lion, tiger, and fish";
72          int position = 0;
73          Set<Character> separators = new Set1L<>();
74          separators.add(',');
75          separators.add(' ');
76          String testWord = Glossary.nextWordOrSeparator(text, position,
77                  separators);
78          String expectedWord = "lion";
79          assertEquals(testWord, expectedWord);
80      }
81
82      /*
83       * tests for generateElements
84       */
85
86      // tests repeating characters
87      @Test
88      public void test_generateElements_1() {
89          String text = "siuuuuu!";
90          Set<Character> testSet = new Set1L<>();
91          Set<Character> expectedSet = new Set1L<>();
92          expectedSet.add('s');
93          expectedSet.add('i');
94          expectedSet.add('u');
95          expectedSet.add('!');
96          Glossary.generateElements(text, testSet);
97          assertEquals(testSet, expectedSet);
98      }
99
100     // tests no repeating characters
101     @Test
102     public void test_generateElements_2() {
103         String text = "Ford";
104         Set<Character> testSet = new Set1L<>();
105         Set<Character> expectedSet = new Set1L<>();
106         expectedSet.add('F');
107         expectedSet.add('o');
108         expectedSet.add('r');
109         expectedSet.add('d');
110         Glossary.generateElements(text, testSet);
111         assertEquals(testSet, expectedSet);
112     }
113
114     // tests for nothing
115     @Test
116     public void test_generateElements_3() {
117         String text = "";
118         Set<Character> testSet = new Set1L<>();
```

```
119         Set<Character> expectedSet = new Set1L<>();
120         Glossary.generateElements(text, testSet);
121         assertEquals(testSet, expectedSet);
122     }
123 
124     /*
125      * tests for nextWordOrSeparator
126      */
127 
128     // tests for nothing
129     @Test
130     public void test_queueToSet_1() {
131         Queue<String> q = new Queue1L<>();
132         Set<String> testSet = new Set1L<>();
133         Set<String> expectedSet = new Set1L<>();
134         Glossary.queueToSet(q, testSet);
135         assertEquals(testSet, expectedSet);
136     }
137 
138     // tests for routine
139     @Test
140     public void test_queueToSet_2() {
141         Queue<String> q = new Queue1L<>();
142         q.enqueue("dog");
143         q.enqueue("cat");
144         Set<String> testSet = new Set1L<>();
145         Set<String> expectedSet = new Set1L<>();
146         expectedSet.add("dog");
147         expectedSet.add("cat");
148         Glossary.queueToSet(q, testSet);
149         assertEquals(testSet, expectedSet);
150     }
151 
152     // tests for repeat
153     @Test
154     public void test_queueToSet_3() {
155         Queue<String> q = new Queue1L<>();
156         q.enqueue("dog");
157         q.enqueue("cat");
158         q.enqueue("dog");
159         Set<String> testSet = new Set1L<>();
160         Set<String> expectedSet = new Set1L<>();
161         expectedSet.add("dog");
162         expectedSet.add("cat");
163         Glossary.queueToSet(q, testSet);
164         assertEquals(testSet, expectedSet);
165     }
166 
167     /*
168      * tests for collectTerms
169      */
170 
171     // test for multiple words
172     @Test
173     public void test_collectTerms_1() {
174         SimpleReader in = new SimpleReader1L("data/test.txt");
175         Queue<String> qTest = new Queue1L<>();
176         Queue<String> qExpected = new Queue1L<>();
177         qExpected.enqueue("word");
```

```
178         qExpected.enqueue("dog");
179         Map<String, String> mExpected = new Map1L<>();
180         mExpected.add("word", "apple friend");
181         mExpected.add("dog", "family tree");
182         Map<String, String> mTest = new Map1L<>();
183         qTest = Glossary.collectTerms(in, mTest);
184         assertEquals(qTest, qExpected);
185         assertEquals(mTest, mExpected);
186     }
187 
188     // test for spacing
189     @Test
190     public void test_collectTerms_2() {
191         SimpleReader in = new SimpleReader1L("data/test2.txt");
192         Queue<String> qTest = new Queue1L<>();
193         Queue<String> qExpected = new Queue1L<>();
194         qExpected.enqueue("alphabet");
195         qExpected.enqueue("continue");
196         Map<String, String> mExpected = new Map1L<>();
197         mExpected.add("alphabet", "a b c d");
198         mExpected.add("continue", "e f");
199         Map<String, String> mTest = new Map1L<>();
200         qTest = Glossary.collectTerms(in, mTest);
201         assertEquals(qTest, qExpected);
202         assertEquals(mTest, mExpected);
203     }
204 
205     // test for nothing
206     @Test
207     public void test_collectTerms_3() {
208         SimpleReader in = new SimpleReader1L("data/test3.txt");
209         Queue<String> qTest = new Queue1L<>();
210         Queue<String> qExpected = new Queue1L<>();
211         Map<String, String> mExpected = new Map1L<>();
212         Map<String, String> mTest = new Map1L<>();
213         qTest = Glossary.collectTerms(in, mTest);
214         assertEquals(qTest, qExpected);
215         assertEquals(mTest, mExpected);
216     }
217 
218 }
219 
```