

```

1 package proj5;
2
3 import org.junit.Rule;
4 import org.junit.Test;
5 import org.junit.rules.Timeout;
6 import static org.junit.Assert.*;
7
8 /**
9  * test the functionality of GenericContainableBag.
10  java
11  * author: Son Nguyen (Kyrie)
12  * version: 6/4/2020
13  */
14 public class GenericBagTest {
15     @Rule // a test will fail if it takes longer than
16           1/10 of a second to run
17     public Timeout timeout = Timeout.millis(100);
18     private GenericContainableBag<Synonym>
19     makeDefaultBag() {
20         GenericContainableBag<Synonym> sample = new
21         GenericContainableBag<>(10);
22         return sample;
23     }
24     private GenericContainableBag<Synonym>
25     makeSpecificBag(String text) {
26         GenericContainableBag<Synonym> sample = new
27         GenericContainableBag<>(10);
28         String[] parts = text.split(" ");
29         for (String part: parts) {
30             sample.add(new Synonym(part));
31         }
32         return sample;
33     }
34     @Test // string representation
35     public void testToString() {
36         GenericContainableBag<Synonym> empty =
37         makeDefaultBag();
38         assertEquals("{ }", empty.toString());
39         assertEquals(0, empty.size());
40     }
41 }

```

```
38         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
39         assertEquals("{Hello, CSC, 151}", nonEmpty.
toString());
40         assertEquals(3, nonEmpty.size());
41     }
42
43     @Test // capacity of a bag
44     public void testCapacity() {
45         GenericContainableBag<Synonym> empty =
makeDefaultBag();
46         assertEquals(10, empty.capacity());
47
48         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
49         assertEquals(10, nonEmpty.capacity());
50     }
51
52     @Test // size of a bag
53     public void testSize() {
54         GenericContainableBag<Synonym> empty =
makeDefaultBag();
55         assertEquals(0, empty.size());
56
57         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
58         assertEquals(3, nonEmpty.size());
59     }
60
61     @Test // empty state of a bag
62     public void testEmpty() {
63         GenericContainableBag<Synonym> empty =
makeDefaultBag();
64         assertTrue(empty.isEmpty());
65
66         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
67         assertFalse(nonEmpty.isEmpty());
68     }
69
70     @Test // adding an item to the bag
71     public void testAdd() {
72         GenericContainableBag<Synonym> empty =
makeDefaultBag();
```

```

73         empty.add(new Synonym("Hi"));
74         assertFalse(empty.isEmpty());
75         assertEquals(1, empty.size());
76
77         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC");
78         nonEmpty.add(new Synonym("151"));
79         assertEquals(3, nonEmpty.size());
80     }
81
82     @Test // check if a bag contains an item
83     public void testContains() {
84         GenericContainableBag<Synonym> empty =
makeDefaultBag();
85         assertFalse(empty.contains(new Synonym("
Anything"))));
86
87         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
88         assertTrue(nonEmpty.contains(new Synonym("
Hello"))));
89         assertFalse(nonEmpty.contains(new Synonym("
Something"))));
90     }
91
92     @Test // remove an item from the bag
93     public void testRemove() {
94         GenericContainableBag<Synonym> empty =
makeDefaultBag();
95         empty.remove(new Synonym("Anything"));
96         assertTrue(empty.isEmpty());
97
98         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151 151");
99         nonEmpty.remove(new Synonym("151"));
100        assertEquals(3, nonEmpty.size());
101    }
102
103    @Test // remove a random item from the bag
104    public void testRemoveRandom() {
105        GenericContainableBag<Synonym> empty =
makeDefaultBag();
106        assertNull(empty.removeRandom());
107        assertTrue(empty.isEmpty());

```

```
108
109         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
110         assertNotNull(nonEmpty.removeRandom());
111         assertEquals(2, nonEmpty.size());
112     }
113
114     @Test // grab a random item from the bag
115     public void testGrabRandom() {
116         GenericContainableBag<Synonym> empty =
makeDefaultBag();
117         assertNull(empty.grabRandom());
118         assertTrue(empty.isEmpty());
119
120         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
121         assertNotNull(nonEmpty.grabRandom());
122         assertEquals(3, nonEmpty.size());
123     }
124
125     @Test // trim a bag to its size
126     public void testTrim() {
127         GenericContainableBag<Synonym> empty =
makeDefaultBag();
128         empty.trimToSize();
129         assertEquals(empty.size(), empty.capacity
());
130
131         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
132         nonEmpty.trimToSize();
133         assertEquals(nonEmpty.size(), nonEmpty.
capacity());
134     }
135
136     @Test // clear the bag
137     public void testClear() {
138         GenericContainableBag<Synonym> empty =
makeDefaultBag();
139         empty.clear();
140         assertEquals(0, empty.size());
141
142         GenericContainableBag<Synonym> nonEmpty =
makeSpecificBag("Hello CSC 151");
```

```
143         nonEmpty.clear();
144         assertEquals(0, nonEmpty.size());
145     }
146
147     @Test // clone a bag
148     public void testClone() {
149         GenericContainableBag<Synonym> original =
makeSpecificBag("Hello CSC 151");
150         GenericContainableBag<Synonym> copy =
makeSpecificBag("Hello CSC 151");
151         assertTrue(original.equals(copy));
152     }
153
154     @Test // check if two bags is equal
155     public void testEquals() {
156         GenericContainableBag<Synonym> original =
makeSpecificBag("Hello CSC 151");
157         GenericContainableBag<Synonym> copy =
makeSpecificBag("Hello CSC 151");
158         assertTrue(original.equals(copy));
159         copy.add(new Synonym("Something"));
160         assertFalse(original.equals(copy));
161     }
162
163     @Test // merge two bag
164     public void testUnion() {
165         GenericContainableBag<Synonym> original =
makeSpecificBag("Hello 20SP CSC 151.");
166         GenericContainableBag<Synonym> toAdd =
makeSpecificBag("This is Kyrie Nguyen from Hanoi,
Vietnam.");
167         original = original.union(toAdd);
168         int expectedSize = 11;
169         int expectedCapacity = 20;
170         assertEquals(expectedSize, original.size());
171         assertEquals(expectedCapacity, original.
capacity());
172     }
173 }
174
```