

```
1 import static org.junit.Assert.assertEquals;
2
3 /**
4  * JUnit test fixture for {@code Set<String>}'s constructor
5  * and kernel methods.
6  *
7  * @author Zhuoyang Li + Xinci Ma
8  */
9 public abstract class SetTest {
10
11     /**
12      * Invokes the appropriate {@code Set} constructor for
13      * the implementation
14      * under test and returns the result.
15      *
16      * @return the new set
17      * @ensures constructorTest = {}
18      */
19     protected abstract Set<String> constructorTest();
20
21     /**
22      * Invokes the appropriate {@code Set} constructor for
23      * the reference
24      * implementation and returns the result.
25      *
26      * @return the new set
27      * @ensures constructorRef = {}
28      */
29     protected abstract Set<String> constructorRef();
30
31     /**
32      * Creates and returns a {@code Set<String>} of the
33      * implementation under
34      * test type with the given entries.
35      *
36      * @param args
37      *         the entries for the set
38      * @return the constructed set
39      * @requires [every entry in args is unique]
40      * @ensures createFromArgsTest = [entries in args]
41      */
42     private Set<String> createFromArgsTest(String... args) {
43         Set<String> set = this.constructorTest();
44         for (String s : args) {
45             assert !set.contains(
46                 s) : "Violation of: every entry in args";
47         }
48     }
49 }
```

```
        is unique";
50         set.add(s);
51     }
52     return set;
53 }
54
55 /**
56  * Creates and returns a {@code Set<String>} of the
57  * reference implementation
58  * type with the given entries.
59  *
60  * @param args the entries for the set
61  * @return the constructed set
62  * @requires [every entry in args is unique]
63  * @ensures createFromArgsRef = [entries in args]
64  */
65 private Set<String> createFromArgsRef(String... args) {
66     Set<String> set = this.constructorRef();
67     for (String s : args) {
68         assert !set.contains(
69             s) : "Violation of: every entry in args
70             is unique";
71         set.add(s);
72     }
73     return set;
74 }
75
76 /**
77  * Tests the default constructor by creating two sets
78  * using the tested
79  * constructor and the reference constructor, then
80  * asserts that both sets
81  * are equal.
82  */
83 @Test
84 public final void testConstructor() {
85     Set<String> s = this.constructorTest();
86     Set<String> sExpected = this.constructorRef();
87     assertEquals(sExpected, s);
88 }
89
90 @Test
91 public final void testAddToEmptySet() {
92     Set<String> s = this.createFromArgsTest();
93     Set<String> sExpected =
94     this.createFromArgsTest("apple");
95 }
```

```
91         s.add("apple");
92         assertEquals(s, sExpected);
93     }
94
95     @Test
96     public final void testAddToSetWithOneEntry() {
97         Set<String> s = this.createFromArgsTest("apple");
98         Set<String> sExpected =
99         this.createFromArgsTest("apple", "banana");
100        s.add("banana");
101        assertEquals(s, sExpected);
102    }
103
104    @Test
105    public final void testAddToSetWithThreeEntries() {
106        Set<String> s = this.createFromArgsTest("apple",
107        "banana", "orange");
108        Set<String> sExpected =
109        this.createFromArgsTest("apple", "banana",
110        "orange", "kiwi");
111        s.add("kiwi");
112        assertEquals(s, sExpected);
113    }
114
115    @Test
116    public final void testRemoveLastEntry() {
117        Set<String> s = this.createFromArgsTest("apple");
118        Set<String> sExpected = this.createFromArgsTest();
119        String removed = s.remove("apple");
120        assertEquals("apple", removed);
121        assertEquals(sExpected, s);
122    }
123
124    @Test
125    public final void testRemoveAnyFromSetWithOneEntry() {
126        Set<String> s = this.createFromArgsTest("apple");
127        Set<String> sExpected = this.createFromArgsRef();
128        String removed = s.removeAny();
129        assertEquals(sExpected, s);
130        assertEquals("apple", removed);
131    }
132
133    @Test
134    public final void
135    testRemoveAnyFromSetWithMultipleEntries() {
136        Set<String> s = this.createFromArgsTest("apple",
137        "banana", "orange");
```

```
133         Set<String> sExpected =
134             this.createFromArgsRef("apple", "banana",
135                                     "orange");
136         String removed = s.removeAny();
137         assertTrue(sExpected.contains(removed));
138         sExpected.remove(removed);
139         assertEquals(sExpected, s);
140     }
141     @Test
142     public final void testContainsWithOneEntry() {
143         Set<String> s = this.createFromArgsTest("apple");
144         assertTrue(s.contains("apple"));
145     }
146     @Test
147     public final void testDoesNotContainWithOneEntry() {
148         Set<String> s = this.createFromArgsTest("apple");
149         assertFalse(s.contains("banana"));
150     }
151     @Test
152     public final void testContainsWithThreeEntries() {
153         Set<String> s = this.createFromArgsTest("apple",
154             "banana", "orange");
155         assertTrue(s.contains("banana"));
156     }
157     @Test
158     public final void testDoesNotContainWithThreeEntries() {
159         Set<String> s = this.createFromArgsTest("apple",
160             "banana", "orange");
161         assertFalse(s.contains("kiwi"));
162     }
163     @Test
164     public final void testContainsWithEmptySet() {
165         Set<String> s = this.createFromArgsTest();
166         assertFalse(s.contains("apple"));
167     }
168     @Test
169     public final void testSizeWithNoEntries() {
170         Set<String> s = this.createFromArgsTest();
171         assertEquals(0, s.size());
172     }
173     }
```

```
177     @Test
178     public final void testSizeWithOneEntry() {
179         Set<String> s = this.createFromArgsTest("apple");
180         assertEquals(1, s.size());
181     }
182
183     @Test
184     public final void testSizeWithThreeEntries() {
185         Set<String> s = this.createFromArgsTest("apple",
186         "banana", "orange");
187         assertEquals(3, s.size());
188     }
189 }
```