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

import org.junit.Test;

import components.set.Set;

/**
 * JUnit test fixture for {@code Set<String>}'s constructor and kernel methods.
 *
 * @author Zhuoyang Li + Xinci Ma
 *
 */
public abstract class SetTest {

    /**
     * Invokes the appropriate {@code Set} constructor for the implementation
     * under test and returns the result.
     *
     * @return the new set
     * @ensures constructorTest = {}
     */
    protected abstract Set<String> constructorTest();

    /**
     * Invokes the appropriate {@code Set} constructor for the reference
     * implementation and returns the result.
     *
     * @return the new set
     * @ensures constructorRef = {}
     */
    protected abstract Set<String> constructorRef();

    /**
     * Creates and returns a {@code Set<String>} of the implementation under
     * test type with the given entries.
     *
     * @param args
     *            the entries for the set
     * @return the constructed set
     * @requires [every entry in args is unique]
     * @ensures createFromArgsTest = [entries in args]
```

```
44      */
45     private Set<String> createFromArgsTest(String... args) {
46         Set<String> set = this.constructorTest();
47         for (String s : args) {
48             assert !set.contains(
49                     s) : "Violation of: every entry in args
   is unique";
50             set.add(s);
51         }
52         return set;
53     }
54
55     /**
56      * Creates and returns a {@code Set<String>} of the
   reference implementation
57      * type with the given entries.
58      *
59      * @param args
60      *            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
   is unique";
70             set.add(s);
71         }
72         return set;
73     }
74
75     /**
76      * Tests the default constructor by creating two sets
   using the tested
77      * constructor and the reference constructor, then
   asserts that both sets
78      * are equal.
79      */
80     @Test
81     public final void testConstructor() {
82         Set<String> s = this.constructorTest();
83         Set<String> sExpected = this.constructorRef();
84         assertEquals(sExpected, s);
85     }
```

```
 86
 87      @Test
 88      public final void testAddToEmptySet() {
 89          Set<String> s = this.createFromArgsTest();
 90          Set<String> sExpected =
    this.createFromArgsTest("apple");
 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 =
    this.createFromArgsTest("apple", "banana");
 99          s.add("banana");
100          assertEquals(s, sExpected);
101      }
102
103      @Test
104      public final void testAddToSetWithThreeEntries() {
105          Set<String> s = this.createFromArgsTest("apple",
    "banana", "orange");
106          Set<String> sExpected =
    this.createFromArgsTest("apple", "banana",
107                  "orange", "kiwi");
108          s.add("kiwi");
109          assertEquals(s, sExpected);
110      }
111
112      @Test
113      public final void testRemoveLastEntry() {
114          Set<String> s = this.createFromArgsTest("apple");
115          Set<String> sExpected = this.createFromArgsTest();
116          String removed = s.remove("apple");
117          assertEquals("apple", removed);
118          assertEquals(sExpected, s);
119      }
120
121      @Test
122      public final void testRemoveAnyFromSetWithOneEntry() {
123          Set<String> s = this.createFromArgsTest("apple");
124          Set<String> sExpected = this.createFromArgsRef();
125          String removed = s.removeAny();
126          assertEquals(sExpected, s);
127          assertEquals("apple", removed);
128      }
```

```
129
130     @Test
131     public final void
    testRemoveAnyFromSetWithMultipleEntries() {
132         Set<String> s = this.createFromArgsTest("apple",
    "banana", "orange");
133         Set<String> sExpected =
    this.createFromArgsRef("apple", "banana",
134                 "orange");
135         String removed = s.removeAny();
136         assertTrue(sExpected.contains(removed));
137         sExpected.remove(removed);
138         assertEquals(sExpected, s);
139     }
140
141     @Test
142     public final void testContainsWithOneEntry() {
143         Set<String> s = this.createFromArgsTest("apple");
144         assertTrue(s.contains("apple"));
145     }
146
147     @Test
148     public final void testDoesNotContainWithOneEntry() {
149         Set<String> s = this.createFromArgsTest("apple");
150         assertFalse(s.contains("banana"));
151     }
152
153     @Test
154     public final void testContainsWithThreeEntries() {
155         Set<String> s = this.createFromArgsTest("apple",
    "banana", "orange");
156         assertTrue(s.contains("banana"));
157     }
158
159     @Test
160     public final void testDoesNotContainWithThreeEntries() {
161         Set<String> s = this.createFromArgsTest("apple",
    "banana", "orange");
162         assertFalse(s.contains("kiwi"));
163     }
164
165     @Test
166     public final void testSizeWithNoEntries() {
167         Set<String> s = this.createFromArgsTest();
168         assertEquals(0, s.size());
169     }
170
```

```
171     @Test
172     public final void testSizeWithOneEntry() {
173         Set<String> s = this.createFromArgsTest("apple");
174         assertEquals(1, s.size());
175     }
176 
177     @Test
178     public final void testSizeWithThreeEntries() {
179         Set<String> s = this.createFromArgsTest("apple",
    "banana", "orange");
180         assertEquals(3, s.size());
181     }
182 }
183 
```