```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 Put your name here
 *
 */
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]
     */
    private Set<String> createFromArgsTest(String... args) {
        Set<String> set = this.constructorTest();
        for (String s : args) {
            assert !set.contains(
                    s) : "Violation of: every entry in args is unique";
            set.add(s);
        }
        return set;
    }

    /**
     * Creates and returns a {@code Set<String>} of the reference implementation
     * type with the given entries.
     *
     * @param args
     *            the entries for the set
     * @return the constructed set
     * @requires [every entry in args is unique]
     * @ensures createFromArgsRef = [entries in args]
     */
    private Set<String> createFromArgsRef(String... args) {
        Set<String> set = this.constructorRef();
        for (String s : args) {
            assert !set.contains(
                    s) : "Violation of: every entry in args is unique";
            set.add(s);
        }
        return set;
    }

    /**
     * Tests adding a new element to an empty set.
     */
    @Test
    public void testAddToEmptySet() {
        Set<String> set = this.createFromArgsTest();
        Set<String> expectedSet = this.createFromArgsRef("Alice");
        set.add("Alice");
        assertEquals(set, expectedSet);
    }

    /**
     * Tests adding a new element to a non-empty set.
     */
    @Test
    public void testAddElementToNonEmptySet() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob");
        Set<String> expectedSet = this.createFromArgsRef("Alice", "Bob",
                "Charlie");
        set.add("Charlie");
        assertEquals(set, expectedSet);
    }

    /**
     * Tests removing the only element in the set.
     */
    @Test
    public void testRemoveElementLeavingEmptySet() {
        Set<String> set = this.createFromArgsTest("David");
        Set<String> expectedSet = this.createFromArgsRef();
        set.remove("David");
        assertEquals(set, expectedSet);
    }

    /**
     * Tests removing an element from a non-empty set.
     */
    @Test
    public void testRemoveElementFromNonEmptySet() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob", "David");
        Set<String> expectedSet = this.createFromArgsRef("Alice", "Bob");
        set.remove("David");
        assertEquals(set, expectedSet);
    }

    /**
     * Tests removing any element from a set with multiple elements.
     */
    @Test
    public void testRemoveAnyElementFromMultiple() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob", "David");
        Set<String> expectedSet = this.createFromArgsRef("Alice", "Bob",
                "David");
        String removedElement = set.removeAny();
        assertTrue(expectedSet.contains(removedElement));
        expectedSet.remove(removedElement);
        assertEquals(set, expectedSet);
    }

    /**
     * Tests removing any element from a set with a single element.
     */
    @Test
    public void testRemoveAnyElementFromSingle() {
        Set<String> set = this.createFromArgsTest("Alice");
        Set<String> expectedSet = this.createFromArgsRef("Alice");
        String removedElement = set.removeAny();
        assertTrue(expectedSet.contains(removedElement));
        expectedSet.remove(removedElement);
        assertEquals(set, expectedSet);
    }

    /**
     * Tests checking the presence of an element in a single-element set.
     */
    @Test
    public void testContainsSingleElement() {
        Set<String> set = this.createFromArgsTest("Alice");
        assertTrue(set.contains("Alice"));
    }

    /**
     * Tests checking the presence of elements in a multiple-element set.
     */
    @Test
    public void testContainsMultipleElements() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob", "Charlie");
        assertTrue(set.contains("Alice"));
        assertTrue(set.contains("Bob"));
        assertTrue(set.contains("Charlie"));
    }

    /**
     * Tests checking the absence of an element in a set.
     */
    @Test
    public void testContainsNotFound() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob", "Charlie");
        assertFalse(set.contains("David"));
    }

    /**
     * Tests the size of an empty set.
     */
    @Test
    public void testSizeOfEmptySet() {
        Set<String> set = this.createFromArgsTest();
        int size = set.size();
        int expectedSize = 0;
        assertEquals(size, expectedSize);
    }

    /**
     * Tests the size of a set with a single element.
     */
    @Test
    public void testSizeOfSingleElementSet() {
        Set<String> set = this.createFromArgsTest("Alice");
        int size = set.size();
        int expectedSize = 1;
        assertEquals(size, expectedSize);
    }

    /**
     * Tests the size of a set with multiple elements.
     */
    @Test
    public void testSizeOfMultipleElementsSet() {
        Set<String> set = this.createFromArgsTest("Alice", "Bob", "Charlie");
        int size = set.size();
        int expectedSize = 3;
        assertEquals(size, expectedSize);
    }
}

```