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

/**
 * JUnit test fixture for {@code Set<String>}'s constructor and kernel methods.
 *
 * @author Shyam Sai Bethina and Yihone Chu
 *
 */
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;
    }

    /**
     * Test the no argument constructor.
     */
    @Test
    public void testConstructor() {
        Set<String> test = this.constructorTest();
        Set<String> expected = this.constructorRef();

        assertEquals(expected, test);
    }

    /**
     * Test add using an edge case.
     */
    @Test
    public void testAdd() {
        Set<String> test = this.createFromArgsTest();
        test.add("hello");

        Set<String> expected = this.createFromArgsRef();
        expected.add("hello");

        assertEquals(expected, test);
    }

    /**
     * Test add using an routine case.
     */
    @Test
    public void testAdd2() {
        Set<String> test = this.createFromArgsTest();
        test.add("hello");
        test.add("there");

        Set<String> expected = this.createFromArgsRef();
        expected.add("hello");
        expected.add("there");

        assertEquals(expected, test);
    }

    /**
     * Test add using an routine case.
     */
    @Test
    public void testAdd3() {
        Set<String> test = this.createFromArgsTest();
        test.add("hello");
        test.add("there");
        test.add("my");

        Set<String> expected = this.createFromArgsRef();
        expected.add("hello");
        expected.add("there");
        expected.add("my");

        assertEquals(expected, test);
```

```java
    }

    /**
     * Test add using an challenging case.
     */
    @Test
    public void testAdd4() {
        Set<String> test = this.createFromArgsTest();
        test.add("");

        Set<String> expected = this.createFromArgsRef();
        expected.add("");

        assertEquals(expected, test);
    }

    /**
     * Test remove using an edge case.
     */
    @Test
    public void testRemove1() {
        Set<String> test = this.createFromArgsTest("hello");
        String testRemoved = test.remove("hello");

        Set<String> expected = this.createFromArgsRef("hello");
        String expectedRemoved = test.remove("hello");

        assertEquals(expected, test);
        assertEquals(expectedRemoved, testRemoved);
    }

    /**
     * Test remove using an routine case.
     */
    @Test
    public void testRemove2() {
        Set<String> test = this.createFromArgsTest("hello",
"there");
        String testRemoved = test.remove("there");

        Set<String> expected = this.createFromArgsRef("hello",
"there");
        String expectedRemoved = test.remove("there");

        assertEquals(expected, test);
        assertEquals(expectedRemoved, testRemoved);
    }

    /**
     * Test remove using an routine case.
     */
    @Test
    public void testRemove3() {
        Set<String> test = this.createFromArgsTest("hello",
"there", "general");
        String testRemoved = test.remove("general");

        Set<String> expected = this.createFromArgsRef("hello",
"there",
                "general");
        String expectedRemoved = test.remove("general");

        assertEquals(expected, test);
        assertEquals(expectedRemoved, testRemoved);
    }

    /**
     * Test remove using an challenging case.
     */
    @Test
    public void testRemove4() {
        Set<String> test = this.createFromArgsTest("");
        String testRemoved = test.remove("");

        Set<String> expected = this.createFromArgsRef("");
        String expectedRemoved = test.remove("");

        assertEquals(expected, test);
        assertEquals(expectedRemoved, testRemoved);
    }

    /**
     * Test contains using an edge case
     */
    @Test
    public void testContains1() {
        Set<String> test = this.createFromArgsTest("hello");
        Set<String> expected = this.createFromArgsRef("hello");
```

```java

        boolean testBoolean = test.contains("hello");

        boolean expectedBoolean = expected.contains("hello");

        assertEquals(expected, test);
        assertEquals(expectedBoolean, testBoolean);

    }

    /**
     * Test contains using an routine case
     */
    @Test
    public void testContains2() {
        Set<String> test = this.createFromArgsTest("hello",
"there");
        Set<String> expected = this.createFromArgsRef("hello",
"there");

        boolean testBoolean = test.contains("there");

        boolean expectedBoolean = expected.contains("there");

        assertEquals(expected, test);
        assertEquals(expectedBoolean, testBoolean);

    }

    /**
     * Test contains using an routine case
     */
    @Test
    public void testContains3() {
        Set<String> test = this.createFromArgsTest("hello",
"there", "general");
        Set<String> expected = this.createFromArgsRef("hello",
"there",
                "general");

        boolean testBoolean = test.contains("kenobi");

        boolean expectedBoolean = expected.contains("kenobi");

        assertEquals(expected, test);
        assertEquals(expectedBoolean, testBoolean);

    }

    /**
     * Test contains using an challenging case
     */
    @Test
    public void testContains4() {
        Set<String> test = this.createFromArgsTest();
        Set<String> expected = this.createFromArgsRef();

        boolean testBoolean = test.contains("");

        boolean expectedBoolean = expected.contains("");

        assertEquals(expected, test);
        assertEquals(expectedBoolean, testBoolean);

    }

    /**
     * Test size using an edge case
     */
    @Test
    public void testSize1() {
        Set<String> test = this.createFromArgsTest();
        Set<String> expected = this.createFromArgsRef();

        int testReturn = test.size();
        int expectedReturn = expected.size();

        assertEquals(expected, test);
        assertEquals(expectedReturn, testReturn);

    }

    /**
     * Test size using an routine case
     */
    @Test
    public void testSize2() {
        Set<String> test = this.createFromArgsTest("hello",
"there");
        Set<String> expected = this.createFromArgsRef("hello",
"there");

        int testReturn = test.size();
        int expectedReturn = expected.size();

        assertEquals(expected, test);
        assertEquals(expectedReturn, testReturn);

    }

    /**
     * Test size using an routine case
     */
    @Test
    public void testSize3() {
        Set<String> test = this.createFromArgsTest("hello",
"there", "general");
        Set<String> expected = this.createFromArgsRef("hello",
"there",
                "general");

        int testReturn = test.size();
        int expectedReturn = expected.size();

        assertEquals(expected, test);
        assertEquals(expectedReturn, testReturn);

    }

    /**
     * Test size using an challenging case
     */
    @Test
    public void testSize4() {
        Set<String> test = this.createFromArgsTest("");
        Set<String> expected = this.createFromArgsRef("");

        int testReturn = test.size();
        int expectedReturn = expected.size();

        assertEquals(expected, test);
        assertEquals(expectedReturn, testReturn);
```

```java

    }

    /**
     * Test removeAny using an edge case
     */
    @Test
    public void testRemoveAny1() {

        /*
         * Set up variables and call method under test
         */
        //Setup
        Set<String> test = this.createFromArgsTest("hello");
        Set<String> expected = this.createFromArgsRef("hello");

        //Call
        String capture = test.removeAny();

        //Evaluation
        assertEquals(true, expected.contains(capture));
        expected.remove(capture);
        assertEquals(expected, test);

    }

    /**
     * Test removeAny using an routine case
     */
    @Test
    public void testRemoveAny2() {

        /*
         * Set up variables and call method under test
         */
        //Setup
        Set<String> test = this.createFromArgsTest("hello",
"there");
        Set<String> expected = this.createFromArgsRef("hello",
"there");

        //Call
        String capture = test.removeAny();

        //Evaluation
        assertEquals(true, expected.contains(capture));
        expected.remove(capture);
        assertEquals(expected, test);

    }

    /**
     * Test removeAny using an routine case
     */
    @Test
    public void testRemoveAny3() {

        /*
         * Set up variables and call method under test
         */
        //Setup
        Set<String> test = this.createFromArgsTest("hello",
"there", "general");
        Set<String> expected = this.createFromArgsRef("hello",
"there",
                "general");

        //Call
        String capture = test.removeAny();

        //Evaluation
        assertEquals(true, expected.contains(capture));

        expected.remove(capture);

        assertEquals(expected, test);

    }

    /**
     * Test removeAny using an challenging case
     */
    @Test
    public void testRemoveAny4() {

        /*
         * Set up variables and call method under test
         */
        //Setup
        Set<String> test = this.createFromArgsTest("");
        Set<String> expected = this.createFromArgsRef("");

        //Call
        String capture = test.removeAny();

        //Evaluation
        assertEquals(true, expected.contains(capture));
        expected.remove(capture);
        assertEquals(expected, test);

    }

    // TODO - add test cases for constructor, add, remove, removeAny, contains, and size

}

```