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

/**
 * JUnit test fixture for {@code Stack<String>}'s constructor and kernel
 * methods.
 *
 * @author Put your name here
 *
 */
public abstract class StackTest {

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

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

    /**
     *
     * Creates and returns a {@code Stack<String>} of the implementation under
     * test type with the given entries.
     *
     * @param args
     *            the entries for the stack
     * @return the constructed stack
     * @ensures createFromArgsTest = [entries in args]
     */
    private Stack<String> createFromArgsTest(String... args) {
        Stack<String> stack = this.constructorTest();
        for (String s : args) {
            stack.push(s);
        }
        stack.flip();
        return stack;
    }

    /**
     *
     * Creates and returns a {@code Stack<String>} of the reference
     * implementation type with the given entries.
     *
     * @param args
     *            the entries for the stack
     * @return the constructed stack
     * @ensures createFromArgsRef = [entries in args]
     */
    private Stack<String> createFromArgsRef(String... args) {
        Stack<String> stack = this.constructorRef();
        for (String s : args) {
            stack.push(s);
        }
        stack.flip();
        return stack;
    }

    // TODO - add test cases for constructor, push, pop, and length

    @Test
    public void pushTest1() {
        Stack<Object> test = new Stack2<>();
        Stack<Object> ref = new Stack2<>();

        test.push(4);
        test.push(5);
        test.push(6);
        ref.push(4);
        ref.push(5);
        ref.push(6);

        assertEquals(ref.top(), test.top());
    }

    public void pushTest2() {

    @Test
    public void pushTest3() {
        Stack<Object> test = new Stack2<>();
        Stack<Object> ref = new Stack2<>();

        test.push(9);
        ref.push(9);

        assertEquals(ref.top(), test.top());
    }

    public void popTest1() {

    @Test
    public void popTest2() {
        Stack<Object> test = new Stack2<>();
        Stack<Object> ref = new Stack2<>();

        test.push(4);
        test.push(5);
        test.push(6);
        test.push(4);
        test.push(5);
        test.push(6);
        ref.push(4);
        ref.push(5);
        ref.push(6);
        ref.push(4);
        ref.push(5);
        ref.push(6);

        assertEquals(ref.pop(), test.pop());
        assertEquals(ref.top(), test.top());

        assertEquals(ref.pop(), test.pop());
        assertEquals(ref.top(), test.top());

        assertEquals(ref.pop(), test.pop());
        assertEquals(ref.top(), test.top());

        assertEquals(ref.pop(), test.pop());
        assertEquals(ref.top(), test.top());

        assertEquals(ref.pop(), test.pop());
        assertEquals(ref.top(), test.top());

        assertEquals(ref.pop(), test.pop());
    }

    @Test
    public void pushPopTest1() {
        Stack<Object> test = new Stack2<>();
        Stack<Object> ref = new Stack2<>();

        test.push(1);
        ref.push(1);

        assertEquals(ref.top(), test.top());
        assertEquals(ref.pop(), test.pop());

        test.push(2);
        ref.push(2);
        test.push(3);
        ref.push(3);

        assertEquals(ref.top(), test.top());
        assertEquals(ref.pop(), test.pop());

        assertEquals(ref.top(), test.top());
        assertEquals(ref.pop(), test.pop());

        test.push(0);
        ref.push(0);

        assertEquals(ref.top(), test.top());
        assertEquals(ref.pop(), test.pop());
    }



    @Test
    public void lengthTest() {
        Stack<Object> test = new Stack2<>();
        Stack<Object> ref = new Stack2<>();

        test.push(4);
        test.push(5);
        test.push(6);
        ref.push(4);
        ref.push(5);
        ref.push(6);

        assertEquals(ref.length(), test.length());
    }
}

```