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

import org.junit.Test;

import components.stack.Stack;

/**
 * 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;
    }

    @Test
    public final void testDefaultConstructor() {
        Stack<String> s = this.constructorTest();
        Stack<String> sExpected = this.constructorRef();
        assertEquals(sExpected, s);
    }

    @Test
    public final void push1() {
        Stack<String> s = this.createFromArgsTest();
        Stack<String> sExpected = this.createFromArgsRef("Hello");

        s.push("Hello");

        assertEquals(sExpected, s);
    }

    @Test
    public final void push2() {
        Stack<String> s = this.createFromArgsTest();
        Stack<String> sExpected = this.createFromArgsRef("Hello", "Bye");

        s.push("Hello");
        s.push("Bye");

        assertEquals(sExpected, s);
    }

    @Test
    public final void push3() {
        Stack<String> s = this.createFromArgsTest();
        Stack<String> sExpected = this.createFromArgsRef("Hello", "Hello");

        s.push("Hello");
        s.push("Hello");

        assertEquals(sExpected, s);
    }

    @Test
    public final void pop1() {
        Stack<String> s = this.createFromArgsTest("Hello");
        Stack<String> sExpected = this.createFromArgsRef();

        String ans = s.pop();

        assertEquals(sExpected, s);
        assertEquals("Hello", ans);
    }

    @Test
    public final void pop2() {
        Stack<String> s = this.createFromArgsTest("Hello", "Bye");
        Stack<String> sExpected = this.createFromArgsRef();

        String ans = s.pop();

        assertEquals(sExpected, s);
        assertEquals("Hello", ans);
    }

    @Test
    public final void length() {
        Stack<String> s = this.createFromArgsTest("Hello");
        Stack<String> sExpected = this.createFromArgsRef("Hello");

        assertEquals(sExpected, s);
        assertEquals(sExpected.length(), s.length());
    }

}
```