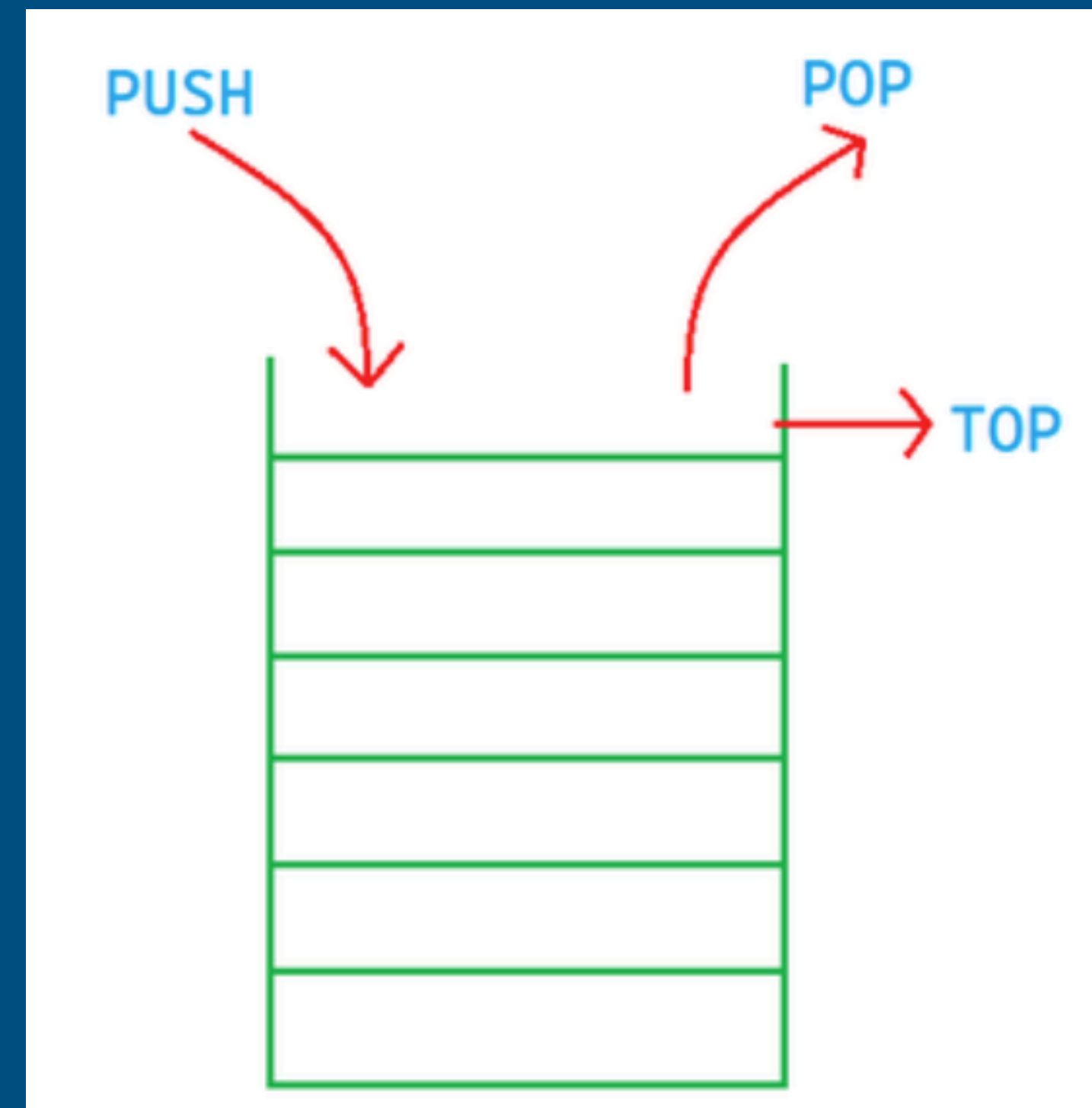


# TDD Stack Example

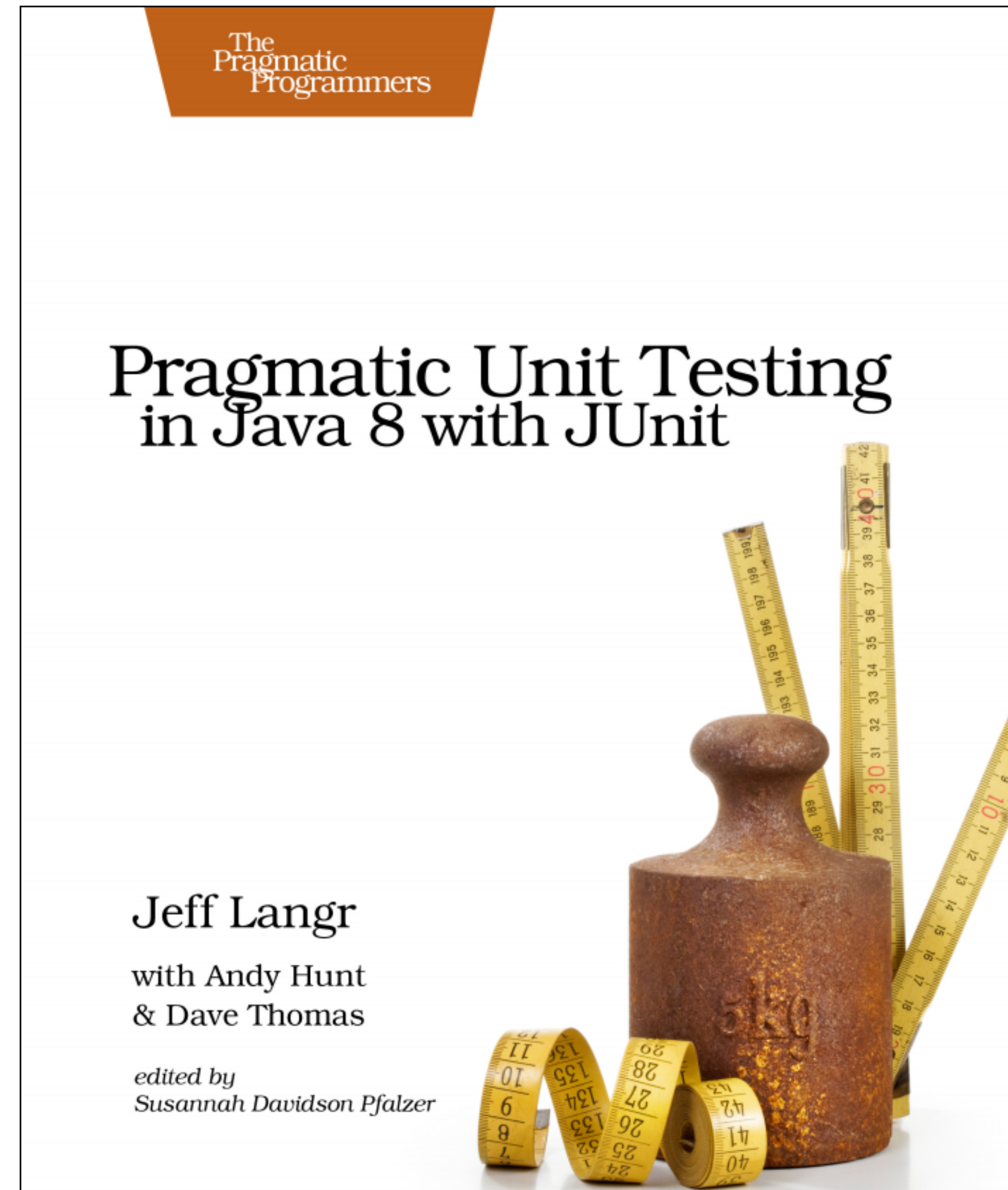


Full Stack Web Development

# Excellent Unit Testing Resource

---

- Published 2015.
- Some excellent “New School” testing approaches.
- Source code used in the book is available at:  
[https://pragprog.com/titles/utj2/source\\_code](https://pragprog.com/titles/utj2/source_code)



# Stack Interface

---

```
public interface Stack
{
    public String pop ();

    public void push (String item);

    public String top ();

    public boolean isEmpty ();
}
```

# ArrayStack

```
public class ArrayStack implements Stack
{
    private int    next_index;
    private String[] stack;

    public ArrayStack()
    {
        stack = new String[100];
        next_index = 0;
    }

    public String pop ()
    {
        if (next_index == 0)
        {
            throw new RuntimeException("empty stack");
        }
        return stack[--next_index];
    }

    public void delete (int n)
    {
        next_index -= n;
    }
}
```

the System Under Test  
(SUT)

```
    public void push (String aString)
    {
        stack[next_index++] = aString;
    }

    public String top ()
    {
        if (next_index == 0)
        {
            throw new
                RuntimeException ("empty stack");
        }
        return stack[next_index - 1];
    }

    public boolean isEmpty ()
    {
        return next_index == 0;
    }
}
```

# StackTest

- Test Fixture



```
public class StackTest
{
    private Stack testStack;

    @Before
    public void setUp()
    {
        testStack = new ArrayStack();
    }

    @After
    public void tearDown()
    {
        testStack = null;
    }
}
```

# Test Specifications

---

1. Starting with an empty stack, call `push()` to push a test string onto the stack. Verify that `top()` returns that string several times in a row, and that `isEmpty()` returns false.
2. For a brand-new stack, `isEmpty()` should be true, `top()` and `pop()` should throw exceptions.
3. Call `pop()` to remove the test string, and verify that it is the same string. `isEmpty()` should now be true. Call `pop()` again verify an exception is thrown.
4. Now do the same test again, but this time add multiple items to the stack - each of them strings which have the same value (say all "test"). Make sure you get the right ones back, in the right order (the most recent item added should be the one returned). In this case, `assertEquals()` isn't good enough; you need `assertSame()` to ensure it's the same object.
5. Push a null onto the stack and pop it; confirm you get a null back.
6. Ensure you can use the stack after it has thrown exceptions.

1.

---

Starting with an empty stack, call push() to push a test string onto the stack.

Verify that top() returns that string several times in a row.

Verify isEmpty() returns false.

```
@Test
public void top()
{
    testStack.push("Item 1");
    assertEquals("Item 1", testStack.top());
    assertEquals("Item 1", testStack.top());
    assertEquals("Item 1", testStack.top());
    assertFalse(testStack.isEmpty());
}
```

## 2.

---

- For a brand-new stack:
- isEmpty() should be true (case 1)
- top() and pop() should throw exceptions (case 2)

```
// Case 2
@Test
public void emptyStack()
{
    assertTrue(testStack.isEmpty());
}

// Case 2
@Test(expected = Exception.class)
public void testTopException()
{
    testStack.top();
}

// Case 2
@Test(expected = Exception.class)
public void testPopException()
{
    testStack.pop();
}
```



### 3.

- Call pop() to remove a test string, and verify that it is the same string.
- isEmpty() should now be true.
- Call pop() again verify an exception is thrown.

```
@Test
public void testPop() throws Exception
{
    String testStr = new String ("test");
    testStack.push(testStr);
    assertEquals(testStr, testStack.pop());
    assertTrue(testStack.isEmpty());
    try
    {
        testStack.pop();
        fail("Pop should throw exception");
    }
    catch (Exception e)
    {
        assertTrue(true);
    }
}
```

# 4.

- Now do the same test again, but this time add multiple items to the stack - each of them strings which have the same value (say all "test").
- Make sure you get the right ones back, in the right order (the most recent item added should be the one returned).
- In this case, assertEquals() isn't good enough; you need assertEquals() to ensure it's the same object

```
@Test
public void testPopDuplicate()
{
    String test1 = new String ("test");
    String test2 = new String ("test");
    String test3 = new String ("test");

    testStack.push(test1);
    testStack.push(test2);
    testStack.push(test3);

    assertEquals(test3, testStack.pop());
    assertEquals(test2, testStack.pop());
    assertEquals(test1, testStack.pop());
}
```

Note: two String objects can contain the same characters but be different objects (i.e. different memory locations). assertEquals() checks to see that two references are pointing to the same object, but the assertEquals() method checks the characters are the same.

5.

---

- Push a null onto the stack and pop it; confirm you get a null back.

```
@Test
public void testNull()
{
    testStack.push(null);
    assertEquals (null, testStack.pop());
}
```

6.

---

- Ensure you can use the stack after it has thrown exceptions

```
@Test
public void testException()
{
    try
    {
        testStack.pop();
        fail("Pop should throw exception");
    }
    catch (Exception e)
    {
        assertTrue(true);
    }
    testStack.push("test");
    assertEquals ("test", testStack.top());
    assertFalse (testStack.isEmpty());
}
```

Completely replace the  
System Under Test (SUT)

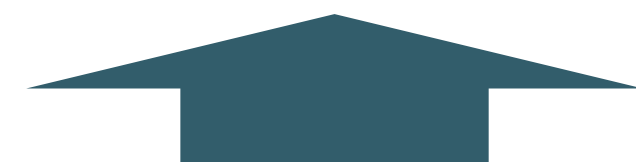


## CollectionStack

```
public class StackTest
{
    private Stack testStack;

    @Before
    public void setUp()
    {
        //testStack = new ArrayStack();
        testStack = new CollectionStack();
    }

    //as before...
```



The same tests can  
verify the behaviour of  
this new SUT

```
public class CollectionStack implements Stack
{
    private java.util.Stack<String> stack;

    public CollectionStack()
    {
        stack = new java.util.Stack<String>();
    }

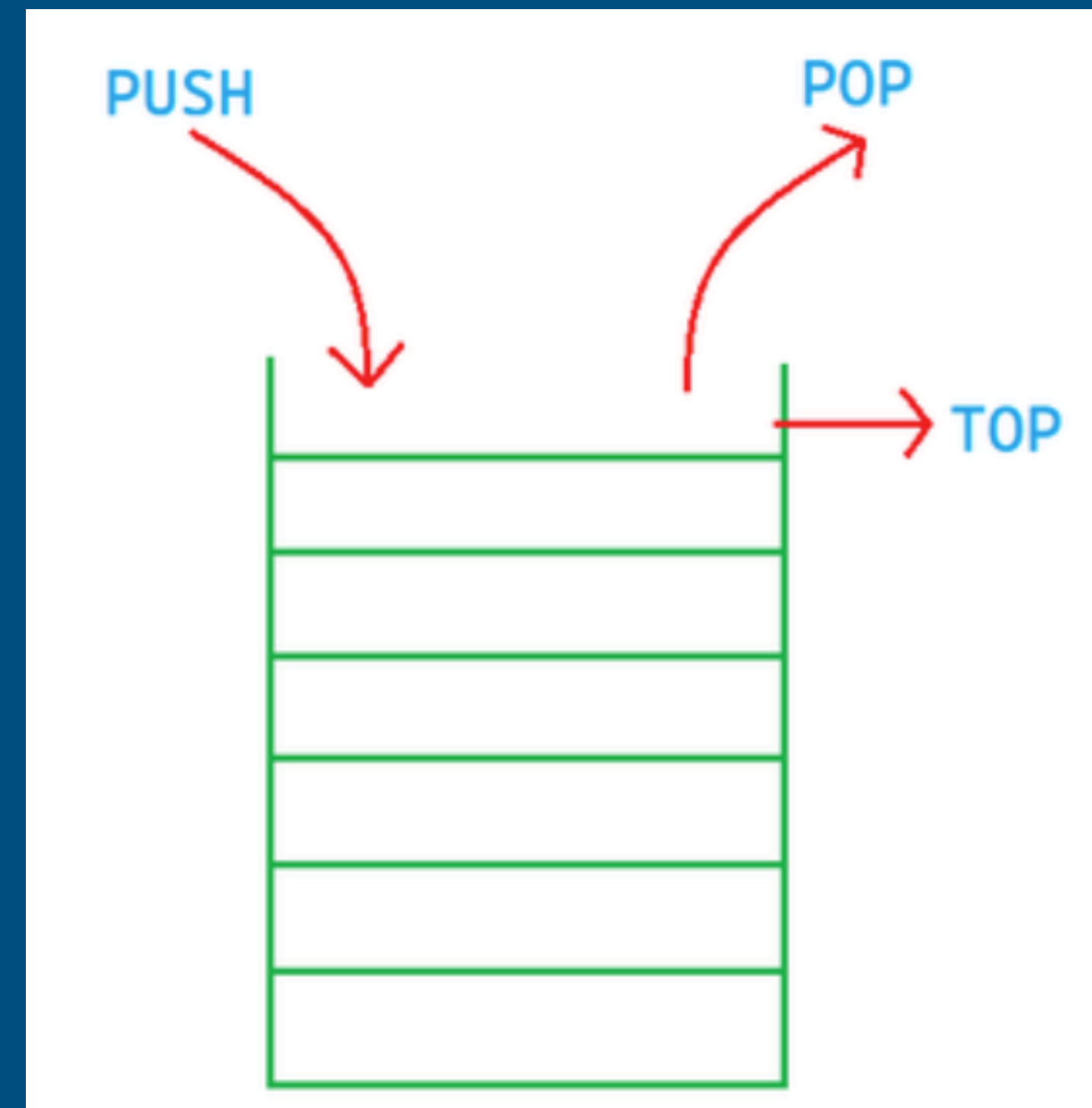
    public boolean isEmpty ()
    {
        return stack.isEmpty();
    }

    public String pop ()
    {
        return stack.pop();
    }

    public void push (String item)
    {
        stack.push(item);
    }

    public String top ()
    {
        return stack.peek();
    }
}
```

# TDD Stack Example



Full Stack Web Development