

**What Mr. Spock would
possibly say about modern
unit testing: pragmatic and
emotional overview**

About me

Yaroslav Yermilov

Senior Software Engineer
EPAM Systems

Work on Big Data project
But responsible for test automation for
the last 2 years
Out of office is a big Groovy ecosystem fan
So the topic choice is obvious



Questions Mr. Spock has

What's about modern unit testing?

Why try Spock?

How to start with Spock?

Advanced and hidden features

What JUnit/TestNG/others offers instead?

Why use Spock?

Why do not use Spock?

When use Spock?



Acknowledgments and disclaimer



Peter Niederwieser
@pniederw



Luke Daley
@ldaley



Better to test before going to production

JUnit?



JUnit 5?



JUnit 3?



TestNG?



Unit testing framework on steroids?

Hamcrest

AssertJ

Google Truth

Mockito
PowerMock
JMockit



Something odd?



Cucumber
J8Spec
JGiven

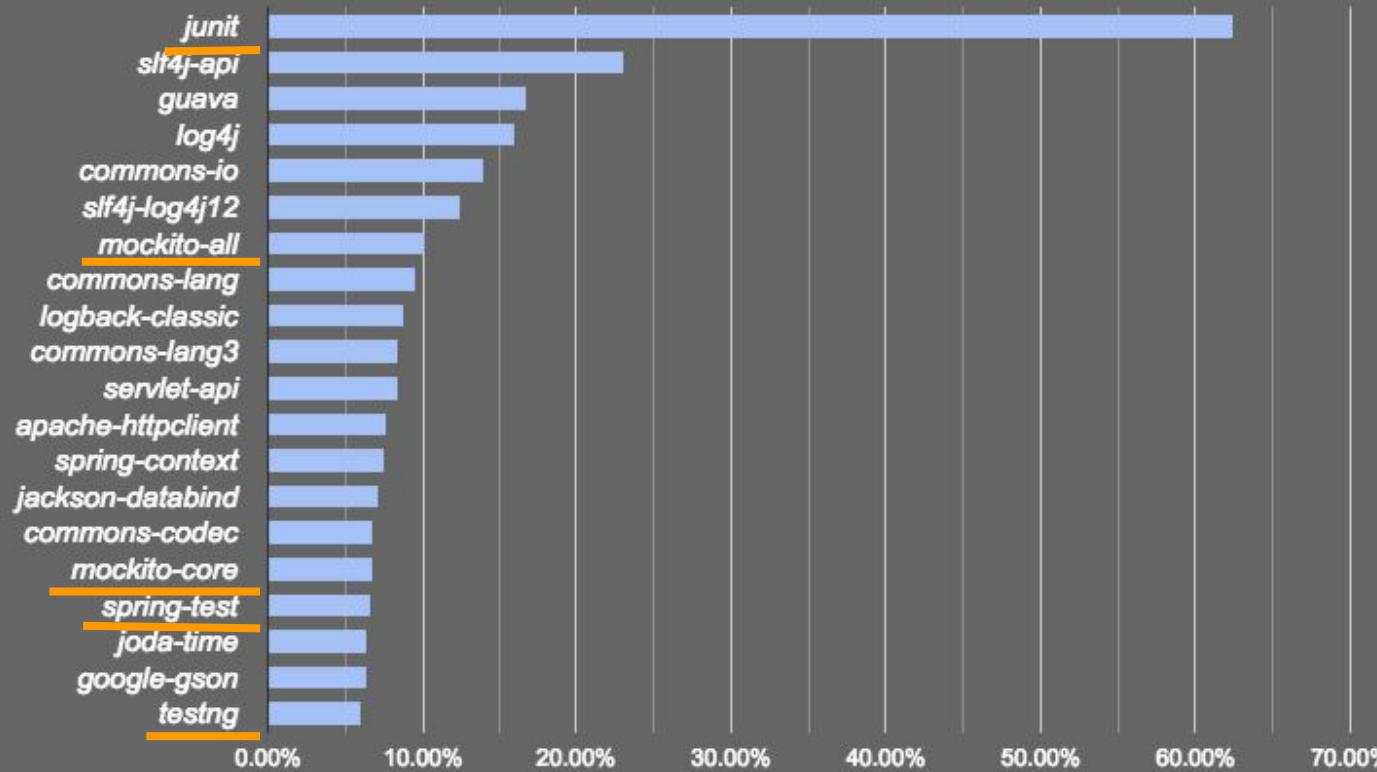
Spock?



Nothing?

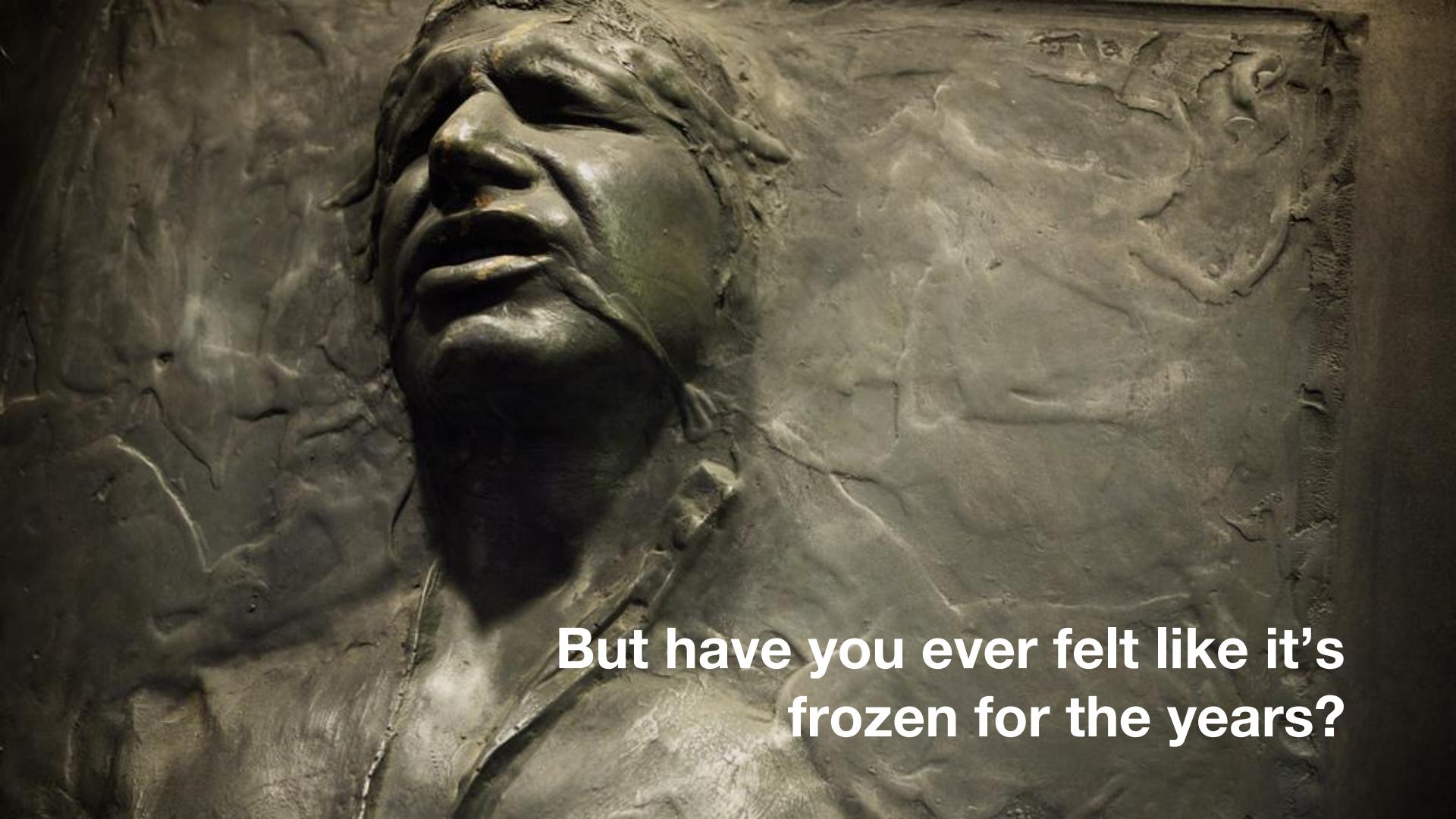


The Top 20 Libraries Used by Github's Most Popular Java Projects





**Automated testing
is a great concept**



**But have you ever felt like it's
frozen for the years?**

WRONG question for September 7, 2016

JUnit vs TestNG: Which Testing Framework Should You Choose?



By Alex Zhitnitsky - September 7, 2016

10 min read 3 Comments

● junit ● spock ● testing ● you are here



ACCEPTABLE question for September 7, 2016

JUnit 5 vs Spock: Which Testing Framework Should You Choose?



By Alex Zhitnitsky - September 7, 2016

10 min read 3 Comments

JUnit 5 vs Spock: Which Testing Framework Should You Choose?

TAKIPI BLOG | Powered by OverOps

WHAT IS OVEROPS?

So why Spock?



The logo for Star Trek: The Next Generation. It features the words "STAR TREK" in a large, bold, blue, blocky font. Below it, "THE NEXT GENERATION" is written in a slightly smaller, blue, italicized, blocky font. The background is a dark space with numerous small white stars. From behind the text, several bright blue light rays radiate outwards, creating a sense of depth and motion.

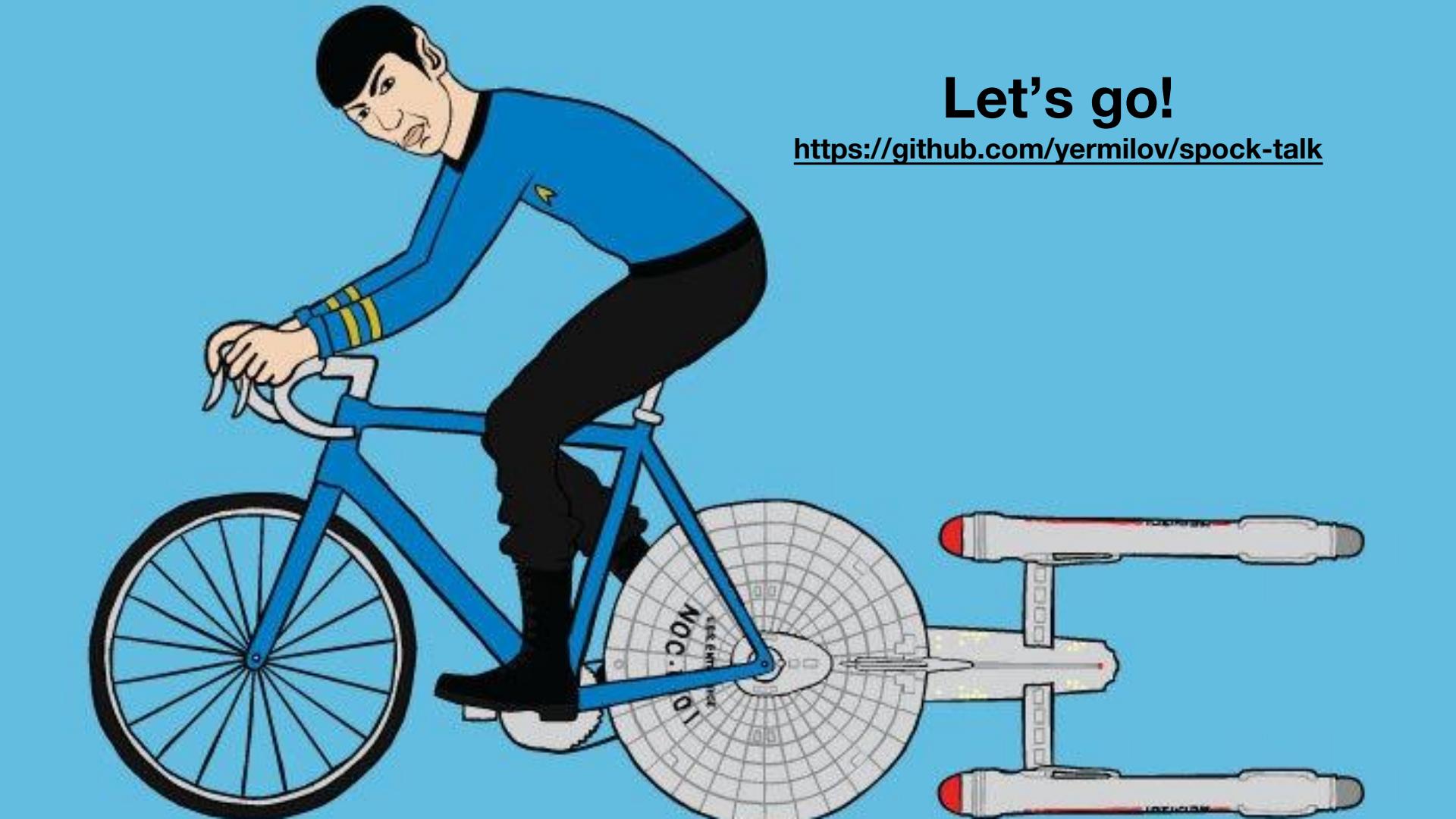
STAR TREK

THE NEXT GENERATION

Spock is the next generation testing framework

Spock is Enterprise ready





Let's go!

<https://github.com/yermilov/spock-talk>



Episode I: Every journey starts somewhere

Episode I: Every journey starts somewhere

```
@Test  
public void arrayList_length  
    ArrayList<String>  
    list.add("Test");  
    import org.junit.Test;  
    import static org.junit.Assert.assertEquals;  
    assertEquals(list.size(), 4);  
}
```

Episode I: Every journey starts somewhere

```
@Test
```

```
public void arrayList_length() {  
    ArrayList<String> list = new ArrayList<>();  
    list.add("we");  
    list.add("are");  
    list.add("here");  
    list.add("now");  
    import static org.junit.Assert.assertEquals;  
    assertEquals(list.size(), 4);  
}
```

Episode I: Every journey starts somewhere

```
@Test
```

```
public void arrayList_length_idm(`
```

```
    ArrayList<String> list
```

```
    list.add("we");
```

```
    list.add("-`
```

```
    lis+ import org.junit.jupiter.api.Test;
```

```
    lis+ import static org.junit.jupiter.api.Assertions.assertEquals;
```

```
    lis+     , `
```

```
    lis+     .size(4, list.size()));
```

```
}
```



Episode I: Every journey starts somewhere

```
public class N01J EasyStart {  
  
    @Test  
    public void arrayList_length_idm() {  
        // setup  
        ArrayList<String> list = new ArrayList<>();  
  
        // run  
        list.add("we");  
        list.add("all");  
        list.add("love");  
        list.add("junit");  
  
        // verify  
        assertThat(list, hasSize(4));  
    }  
}
```

Episode I: Every journey starts somewhere

```
class N04S_EasyStart extends Specification {  
  
    def arrayList_length() {  
        setup:  
            ArrayList<String> list = new ArrayList<>();  
  
        when:  
            list.add("we");  
            list.add("will");  
            list.add("love");  
            list.add("spock");  
  
        then:  
            assertThat(list, hasSize(4));  
    }  
}
```

Episode I: Every journey starts somewhere

```
class N04S EasyStart extends Specification {  
  
    def 'ArrayList.size() test, but much spockier'() {  
        setup:  
            ArrayList<String> list = new ArrayList<>()  
  
        when:  
            list.add('we')  
            list.add('will')  
            list.add('love')  
            list.add('spock')  
  
        then:  
            list.size() == 4  
    }  
}
```

Episode II: Assertions

Episode II: Assertions

then:

```
assertThat(list.  
        java.lang.Object[]  
    import static org.hamcrest.Matchers.hasSize  
    import static org.junit.Assert.assertThat  
    error:  
    collection with size <3>  
    but: collection size was <4>
```

then:

```
list.size() == 3
```

Condition not satisfied:

```
list.size() == 3  
| | |  
| 4   false  
[we, will, love, spock]
```

Episode II: Assertions

Condition not satisfied:

```
list.findAll({ it.length() < 5 }).groupBy({ it[0] }).find({ k, v -> v.size() > 1 }).value == list.findAll({ it.length() < 5 }).drop(1)
|   |
|   [we, will, love]           |                   |                           |   |   |   |   |
|   [we, will, love, spock]    |       w=[we, will]     [we, will, love]           [will, love]
[we, will, love, spock]      [w:[we, will], l:[love]]  |   |   [we, will, love, spock]
                           |   false
                           [we, will]
```

Episode II: Assertions

```
import org.assertj.core.api.Assertions  
then:  
assertThat(list).hasSize(3);
```

```
java.lang.AssertionError:  
Expected size:<3> but was:<4> in:  
<["we", "will", "love", "spock"]>
```

```
import com.google.common.truth.Truth  
then:  
assertThat(list).hasSize(3);
```

```
java.lang.AssertionError: Not true that  
<[we, will, love, spock]> has a size of <3>. It is <4>
```

Episode II: Assertions

then:

```
assertEquals(list.size(), 3);
```

java.lang.AssertionError:

Expected :4
Actual :3

then:

```
assertTrue(list.size() == 3);
```

java.lang.AssertionError

```
at org.junit.Assert.fail(Assert.java:86)
at org.junit.Assert.assertTrue(Assert.java:19)
at org.junit.Assert.assertTrue(Assert.java:19)
at N05S_Asserts.verifying array list size
```

Episode II: Assertions

then:

```
expect list, hasSize(3)
```

Condition not met:
Expected: a collection with size <3>

```
      hasSize(3)
```

```
      use [we, will, love, spock]
```

Expected: a collection with size <3>

but: collection size was <4>

Episode II: Assertions

```
@Test
void 'ArrayList.size()'() {
    // setup
    ArrayList<String> list =
        assumeThat(list,
            // run
            list +
                // failed
                assert list.findAll({ it.length() < 5 }) == list.drop(2)
                    | [we, will, love]
                    | [we, will, love, spock]
                    | [we, will, love, spock]
                    | false
        );
    list.findAll({ it.length() < 5 }) == list.drop(2)
}
```

Episode III: Test structure



NASA

National Aeronautics and
Space Administration

Marshall Space Flight Center

Huntsville, Alabama

Episode III: Test structure

```
static Sql sql  
  
@BeforeClass  
public static void createTable()  
    sql = Sql.newInstance("jdbc:h  
    sql.execute("create table  
}  
  
} }  
  
@AfterClass  
public static void dropTable()  
    sql.execute("drop table testing_tool")  
    sql.close()  
}  
  
import org.junit.After  
import org.junit.AfterClass  
import org.junit.Before  
import org.junit.BeforeClass  
@Before  
public void initData() {  
    String toolCount = "select count(*) from testing_tool"  
    String toolName = "select name from testing_tool"  
    String toolType = "select type from testing_tool"  
    String toolCountActual = sql.firstRow(toolCount).get(0)  
    String toolNameActual = sql.firstRow(toolName).get(0)  
    String toolTypeActual = sql.firstRow(toolType).get(0)  
    assertEquals("3", toolCountActual)  
    assertEquals("3L", toolNameActual)  
    assertEquals("3L", toolTypeActual)  
}  
  
@Test  
public void toolCount() {  
    // run  
    def actual = sql.firstRow("select count(*) from testing_tool")  
    // verify  
    assertThat(actual.toolCount, is(3L))  
}
```

Episode III: Test structure

```
static Sql sql
```

```
@BeforeClass
```

```
public static void createTable()  
    sql = Sql.newInstance("jdbc:h  
    sql.execute("create tabl
```

```
}
```

```
@AfterClass
```

```
public static vo  
    sql.execute("dete testing_tool")  
    sql.close()
```

```
}
```

```
@BeforeMethod
```

```
public ..
```

```
import org.testng.annotations.AfterClass  
import org.testng.annotations.AfterMethod  
import org.testng.annotations.BeforeClass  
import org.testng.annotations.BeforeMethod  
import org.testng.annotations.Test
```

```
tool values
```

```
@Test  
public void toolCount() {  
    // run  
    def actual = sql.firstRow("select co  
    // verify  
    assertEquals(actual.toolCount, 3L)  
}
```

Episode III: Test structure

```
static Sql sql
```

```
@BeforeAll
```

```
public static void createTable() {  
    sql = Sql.newInstance("jdbc:h2:  
    sql.execute("create table +  
}
```

```
@AfterAll
```

```
public static void u  
    sql.execute("drop  
    sql.close()  
}
```

```
@Before
```

```
rn
```

```
import org.junit.jupiter.api.AfterAll  
import org.junit.jupiter.api.AfterEach  
import org.junit.jupiter.api.BeforeAll  
import org.junit.jupiter.api.BeforeEach
```

```
_tool values
```

```
@Test
```

```
public void toolCount() {  
    // run  
    def actual = sql.firstRow("select count  
    // verify  
    assertEquals(3L, actual.toolCount)  
}
```

Episode III: Test structure

```
static Sql sql  
  
def setupSpec(){  
    sql = Sql.newInstance("jdbc:  
    sql.execute("create table te  
}  
  
def cleanupSpec(){  
    sql.execute("drop table testing_tool")  
    sql.close()  
}  
  
def setup(){  
    sql.execute("insert into testing_tool values  
}  
  
def cleanup(){  
    sql.execute("delete from tes  
}  
  
def 'tool count'(){  
    when: 'we count number of unit t  
    def actual = sql.firstRow("selec  
        then: 'it should be 3'  
        actual.toolCount == 3  
    }  
}
```

The diagram illustrates the flow of code execution. It starts with a call to `setupSpec()`, which initializes a `Sql` object and creates a table named `testing_tool`. This is followed by a call to `cleanupSpec()`, which drops the `testing_tool` table and closes the `Sql` connection. The next step is `setup()`, which inserts three rows into the `testing_tool` table. Finally, `cleanup()` is called to delete the rows from the table. Arrows indicate the sequence of method calls: `setupSpec()` points to `cleanupSpec()`, which points to `setup()`, which points to `cleanup()`.

Episode III: Test structure

```
trait DatabaseSpec {  
  
    static Sql sql  
  
    def setupSpec() {  
        sql = Sql.newInstance("jdbc:h2:mem:", "org.h2.Driver")  
        sql.execute("CREATE TABLE testing_tool (name, version)")  
    }  
    class N11S_SetupTeardown_AsYouLike extends Specification  
    def cleanup() {  
        sql.executeUpdate("DROP TABLE testing_tool")  
        sql.close()  
    }  
  
    def setup() {  
        sql.execute("insert into testing_tool values ('junit', '4.12')")  
    }  
}
```

Episode III: Test structure

```
@Shared @AutoCleanup sql
```

```
def 'JUnit5 is in game!'() {
    setup: 'add JUnit 5 to the list of unit testing tools'
        sql.execute("insert into testing_tool values (4, 'junit'

    when: 'we count number of JUnits'
        def actual = sql.firstRow("select count(*) as toolCount

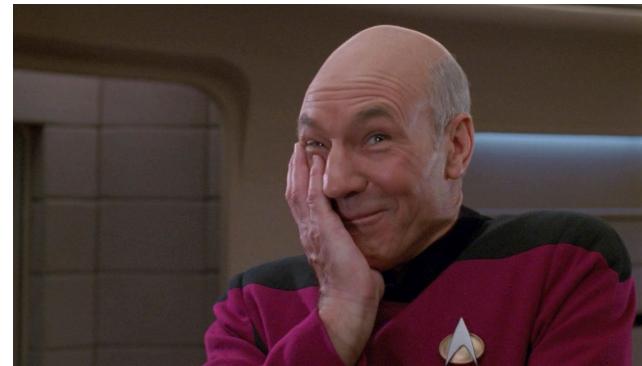
    then: 'it should be 2'
        actual.toolCount == 2

    cleanup: 'remove JUnit 5 from the list of unit testing t
        sql.execute("delete from testing_tool where id = 4")
}
```

Episode III: Test structure

```
// run
def actual = sql.firstRow("select count(*) as too

// verify
actual.toolCount == 2
} finally {
    try {
        // cleanup
        sql.execute("delete from testing_tool where i
    } finally {
        sql.close()
    }
}
```



Episode III: Test structure

```
@Title('ArrayList tests')
@Narrative('''
As Java developer
(that trust nothing)
I want to be sure ArrayList works
''')
class N07S_IdiomaticSpock extends Specification {
```

```
@Subject
ArrayList<String> list
```

```
@Issue('https://github.com/yermilov/spock-talk/issues/1')
def 'ArrayList.size()' {
    setup: 'new ArrayList instance'
    list = new ArrayList<>()
```

```
expect: 'that newly created ArrayList instance is empty'
list.empty
```

```
when: 'add value to list'
list.add 'we'
```

```
and: 'add one more value to list'
list.add 'will'
```

```
then: 'array list size should be 2'
list.size() == 2
```

Episode III: Test structure

ArrayList tests
As Java developer
(that trust nothing)
I want to be sure ArrayList works

Features:

- ArrayList.size()

ArrayList.size()

Issues:

- <https://github.com/yermilov/spock-talk/issues/1>

Given: new ArrayList instance

Expect: that newly created ArrayList instance is empty

When: add value to list

And: add one more value to list

Then: array list size should be 2

When: add two more values into list

A wide-angle shot of a massive, industrial-style hangar bay. The ceiling is high and dark, supported by thick metal beams and pipes. In the foreground and middle ground, thousands of Stormtroopers in white armor stand in precise, organized rows, filling the floor space. A massive, dark-colored Imperial starship hangs from the ceiling by a single support arm, its hull gleaming slightly. The background shows more of the hangar's interior and some distant structures under a hazy, orange-tinted sky.

Episode IV: Data driven test

Episode IV: Data driven test

```
def 'is coin flip good enough for determine if integer is prime'() {  
    when: 'we flip a coin'  
    boolean coinFlip = new Random().nextBoolean()  
  
    then: 'it will be great if coin flip predict if number is prime'  
    (number % 2 == 0 ? false : coinFlip) == expectedAnswer  
  
    where: 'data is random'  
    number << (1..5).collect({ new Random().nextInt(1000) }).findAll({ it >= 2 }).sort()  
    expectedAnswer = Primes.isPrime(number)  
}
```

Episode IV: Data driven test

is coin flip good enough for determine if integer is prime

[Return](#)

When: we flip a coin

Then: it will be great if coin flip predict if number is prime

Where: data is random

	number	expectedAnswer		
Examples:	34	false	OK	
	511	false	FAIL	
	517	false	FAIL	
	898	false	OK	
	985	false	FAIL	2/5 passed

The following problems occurred:

- [511, false]
 - Condition not satisfied:

```
(number % 2 == 0 ? false : coinFlip) == expectedAnswer
|           |           |
511       1       false      true      |   |
                                         |   |
                                         false
```

- [517, false]

Episode IV: Data driven test

```
@Unroll
def `calculate runner speed and location after some time for #description`() {
    expect: 'that runner speed is equal to expected'
    initialSpeed + acceleration * time == expectedSpeed

    and: 'runner location is equal to expected'
    initialLocation + time * (initialSpeed + acceleration / 2 * time) == expectedLocation

    where: 'there are set of precalculated data for different situations'
```

initialLocation	initialSpeed	acceleration	time	expectedLocation	expectedSpeed	description
0	6	0	10	60	6	'steady run from starting point'
5	0	3	3	17	9	'starting from standing with acceleration'
-50	10	-1	10	0	0	'constant deceleration'

Episode IV: Data driven test

calculate runner speed and location after some time for steady run from starting point

[Return](#)

Expect: that runner speed is equal to expected

And: runner location is equal to expected

Where: there are set of precalculated data for different situations

calculate runner speed and location after some time for starting from standing with acceleration

[Return](#)

Expect: that runner speed is equal to expected

And: runner location is equal to expected

Where: there are set of precalculated data for different situations

The following problems occurred:

- Condition not satisfied:

```
initialLocation + time * (initialSpeed + acceleration / 2 * time) == expectedLocation
|           |   |           |   |           |   |   |
5           3   0           3   1.5  3   17
             18.5 13.5         4.5      false
```

calculate runner speed and location after some time for constant deceleration

[Return](#)

Expect: that runner speed is equal to expected

And: runner location is equal to expected

Where: there are set of precalculated data for different situations

Episode IV: Data driven test

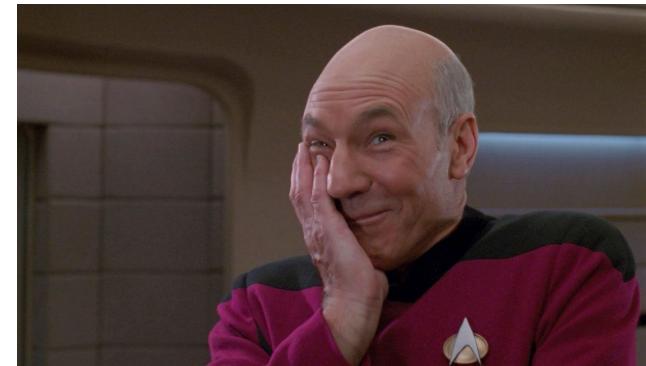
```
@RunWith(Parameterized.class)
public class N16J_DataTables {

    @Parameterized.Parameters(name = "calculate runner speed and location after
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][] {
            { 0.0,   6.0,  0.0,  10.0, 60.0, 6.0, "steady run from starting point" },
            { 5.0,   0.0,  3.0,  3.0,  17.0, 9.0, "starting from standing with a
                { -50.0, 10.0, -1.0, 10.0, 0.0,  0.0, "constant deceleration" }
        });
    }

    @Parameterized.Parameter(value = 0)
    public double initialLocation;

    @Parameterized.Parameter(value = 1)
    public double initialSpeed;

    @Parameterized.Parameter(value = 2)
    public double acceleration;
```



Episode IV: Data driven test

Test	Duration	Result
location[calculate runner speed and location after some time for constant deceleration]	0s	passed
location[calculate runner speed and location after some time for starting from standing with acceleration]	0s	failed
location[calculate runner speed and location after some time for steady run from starting point]	0s	passed
speed[calculate runner speed and location after some time for constant deceleration]	0s	passed
speed[calculate runner speed and location after some time for starting from standing with acceleration]	0s	passed
speed[calculate runner speed and location after some time for steady run from starting point]	0s	passed

Episode IV: Data driven test

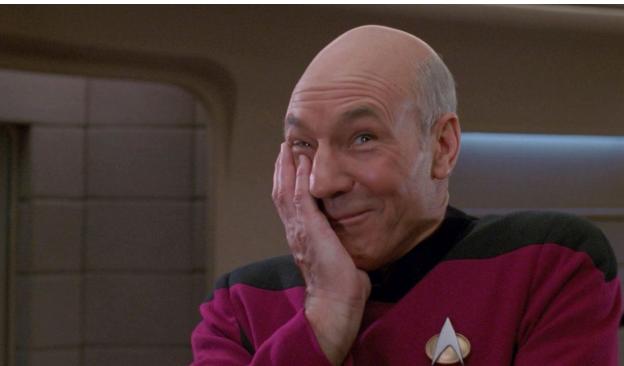


```
@DataProvider(name = "data")
public static Object[][][] data() {
    return new Object[][][] {
        { 0.0,   6.0,  0.0,  10.0, 60.0, 6.0 },
        { 5.0,   0.0,  3.0,  3.0,  17.0, 9.0 },
        { -50.0, 10.0, -1.0, 10.0, 0.0,  0.0 }
    };
}

@Test(dataProvider = "data")
public void speed(double initialLocation, double initialSpeed, double
    double speed = initialSpeed + acceleration * time;
    assertEquals(speed, expectedSpeed);
}
```

Episode IV: Data driven test

Test	Duration	Result
location[0](0.0, 6.0, 0.0, 10.0, 60.0, 6.0)	0s	passed
location[1](5.0, 0.0, 3.0, 3.0, 17.0, 9.0)	0.001s	failed
location[2](-50.0, 10.0, -1.0, 10.0, 0.0, 0.0)	0s	passed
speed[0](0.0, 6.0, 0.0, 10.0, 60.0, 6.0)	0s	passed
speed[1](5.0, 0.0, 3.0, 3.0, 17.0, 9.0)	0s	passed
speed[2](-50.0, 10.0, -1.0, 10.0, 0.0, 0.0)	0s	passed



Episode IV: Data driven test

```
private static final Object[][] DATA = new Object[][] {  
    { 0.0, 6.0, 0.0, 10.0, 60.0, 6.0, "steady run from starting point" },  
};  
@Tes  
publ  
    ! N18_J5_DynamicTests  
        speedTests()  
            steady run from starting point  
            starting from standing with acceleration  
            constant deceleration  
        locationTests()  
            steady run from starting point  
            ! starting from standing with acceleration  
            constant deceleration  
});  
}
```

A Star Trek: Discovery promotional image. The U.S.S. Discovery, a large, metallic starship, is shown from a low angle, angled upwards towards the viewer. The ship's hull features the letters "DISCOVERY" and the number "NCC-1010". In the background, a massive, dark planet with swirling clouds dominates the upper half of the frame. Several smaller, glowing starships are visible, some appearing to be in combat or maneuvering around the planet.

Episode V: Exceptions

Episode V: Exceptions

```
@Test(expected = IndexOutOfBoundsException.class)
public void exception_oldWay() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();
    // run
    arrayList.get(17);
}
```

Episode V: Exceptions

```
@Test
public void exceptionAndMessage_oldWay() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();

    // run
    try {
        arrayList.get(17);
        fail("Expected an IndexOutOfBoundsException to be thrown");
    } catch (IndexOutOfBoundsException exception) {
        // verify
        assertThat(exception.getMessage(), is("Index: 17, Size: 0"));
    }
}
```

Episode V: Exceptions

```
def 'empty ArrayList has no 17th element'() {
    given: 'empty array list'
    def arrayList = new ArrayList<Integer>()

    when: 'we try to retrieve element with index #17'
    arrayList.get(17)

    then: 'exception is thrown with expected message'
    IndexOutOfBoundsException exception = thrown()
    exception.message == 'Index: 17, Size: 0'
}
```

Episode V: Exceptions

```
@Test
public void noException_oldWay() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();

    // run
    arrayList.size();

    // verify no exception is thrown
}
```

Episode V: Exceptions

```
@Test
public void noException_uglyWay() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();

    // run
    try {
        arrayList.size();
    } catch (Exception exception) {
        // verify no exception is thrown
        fail("Expected no exception to be thrown");
    }
}
```

Episode V: Exceptions

```
def 'large ArrayList has 17th element'() {  
    given: 'list of 28 prime numbers'  
    def arrayList = new ArrayList<Integer>()  
    28.times { arrayList << Primes.nextPrime(arrayList.empty ? 1 : arrayList[-1]) }  
  
    when: 'we try to retrieve element with index #17'  
    arrayList.get(17)  
  
    then: 'no exception is thrown'  
    notThrown(IndexOutOfBoundsException)  
}
```

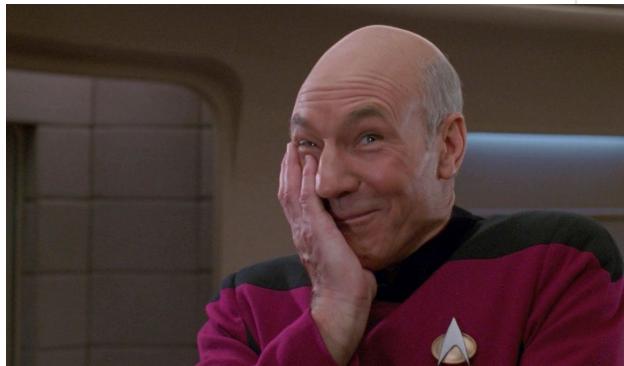
Episode V: Exceptions

```
@Rule  
public ExpectedException thrown = ExpectedException.none();
```

```
@Test  
public void exception_modernWay() {  
    // setup  
    ArrayList<Integer> arrayList = new ArrayList<>();  
  
    // expect  
    thrown.expect(IndexOutOfBoundsException.class);  
    thrown.expectMessage(is("Index: 17, Size: 0"));  
  
    // run  
    arrayList.get(17);  
}
```

Episode V: Exceptions

```
@Test(expectedExceptions = IndexOutOfBoundsException.class, expectedExceptionsMessageRegExp = "Index: 17, Size: 0")
public void exception() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();
    // run
    arrayList.get(17);
}
```



Episode V: Exceptions

```
@Test
public void exception() {
    // setup
    ArrayList<Integer> arrayList = new ArrayList<>();

    // run & verify
    Throwable thrown = expectThrows(IndexOutOfBoundsException.class, () -> { arrayList.get(17); });
    assertEquals("Index: 17, Size: 0", thrown.getMessage());
}
```

A photograph of three LEGO Star Trek minifigures standing in front of a blurred city skyline. The figure on the left has black hair and wears a blue uniform with a black collar and a 'K' insignia. The figure in the center has blonde hair and wears a yellow uniform with a black collar and a 'K' insignia, holding a grey phaser. The figure on the right has dark brown hair and wears a black uniform. All three figures have a serious, slightly annoyed expression.

Episode VI: Mocks, Stubs, Spies

Episode VI: Mocks, Stubs, Spies

```
Random random = mock(Random.class);
doReturn(0).when(random).nextInt(26);
```

```
Random random = Stub()
random.nextInt(26) >> 0
```

Episode VI: Mocks, Stubs, Spies

```
Random random = mock(Random.class);
doReturn(0).doReturn(1).doReturn(2).doReturn(3).doReturn(4).when(random).nextInt(26);
```

```
Random random = Stub()
random.nextInt(_) >>> [ 0, 1, 2, 3, 4 ]
```

Episode VI: Mocks, Stubs, Spies

```
Random random = mock(Random.class);
doThrow(new RuntimeException()).when(random).nextInt(26);
```

```
Random random = Stub()
random.nextInt(_) >> { throw new RuntimeException() }
```

Episode VI: Mocks, Stubs, Spies

```
Random random = mock(Random.class);
doAnswer(inv -> (int) inv.getArguments()[0] - 1).when(random).nextInt(26);
```

```
Random random = Stub()
random.nextInt(_) >> { int max -> max - 1 }
```

Episode VI: Mocks, Stubs, Spies

```
verify(random, times(5)).nextInt(anyInt());
```

```
5 * random.nextInt(_)
```

Episode VI: Mocks, Stubs, Spies

```
InOrder inOrder = inOrder(random);
inOrder.verify(random, times(1)).nextInt(10);
inOrder.verify(random, atLeast(8)).nextInt(26);
inOrder.verifyNoMoreInteractions();
```

```
    then: 'first random generator
          1 * random.nextInt(10)
```

```
    then: 'then random generator
          (8.._) * random.nextInt(_)
```

```
    then: 'random generator is ne
          0 * random._
```

Episode VI: Mocks, Stubs, Spies

```
1 * passwordGenerator.generate(!null)  
1 * passwordGenerator.generate(_ as Integer)
```

Episode VI: Mocks, Stubs, Spies

```
ArgumentCaptor<Integer> argument = ArgumentCaptor.forClass(Integer.class);
verify(passwordGenerator).generate(argument.capture());
assertThat(argument.getValue(), is(greaterThanOrEqualTo(8)));
assertThat(argument.getValue(), is(lessThan(18)));

1 * passwordGenerator.generate({ it >= 8 && it < 18 })
```

Episode VI: Mocks, Stubs, Spies

```
def 'static final mocking'() {  
    given: 'we can mock *static* method of *final* class java.lang.Math'  
    GroovySpy(Math, global: true)  
    1 * Math.abs(_) >> 28  
  
    expect: 'o_0'  
    Math.abs(17) == 28  
}
```

Episode VI: Mocks, Stubs, Spies

```
def 'syntax bomb'() {
    when: 'mocking goes to far'
    Mock(Random)

    then: '''that guy who ends up maintaining
            your code will be a violent psychopath
            who knows where you live'''
        (_..._) * _.(_) >> _

}
```



Episode VII: Time travel

Episode VII: Time travel

```
// run
def old = sql.firstRow("select count(*) as toolCount from testing_tool").toolCount
sql.execute("insert into testing_tool values (4, 'junit', '5')")
def actual = sql.firstRow("select count(*) as toolCount from testing_tool").toolCount

// verify
assertThat(actual, is(old + 1))
```

Episode VII: Time travel

```
sql.firstRow("select count(*) as toolCount from testing_tool").toolCount ==  
  old(sql.firstRow("select count(*) as toolCount from testing_tool").toolCount) + 1
```

Episode VII: Time travel

```
@Test(timeout = 2000)
public void infiniteLoop() {
    // setup
    ArrayList<String> arrayList = new ArrayList<>();

    // run
    while (true) { arrayList.add("junit forever!"); }

}

@Test(timeout = 2000)
public void infiniteLoop() {
    // setup
    ArrayList<String> arrayList = new ArrayList<>();

    // run
    while (true) { arrayList.add("testing forever!"); }

}
```

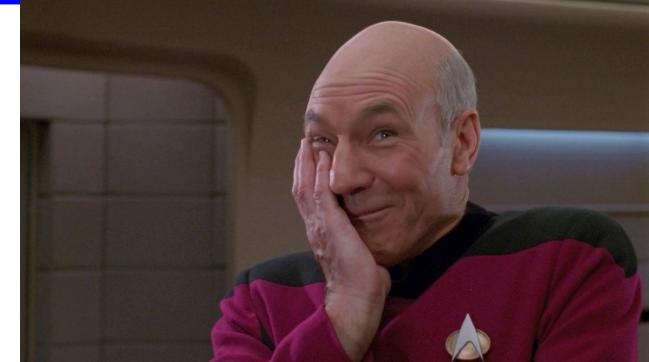
Episode VII: Time travel

```
@Timeout(value = 2, unit = TimeUnit.SECONDS)
def 'infinite loop'() {
    setup: 'array list'
    def arrayList = new ArrayList<String>()

    expect: 'we will add to it values forever'
    while (true) { arrayList.add('spock forever!') }
}
```

Episode VII: Time travel

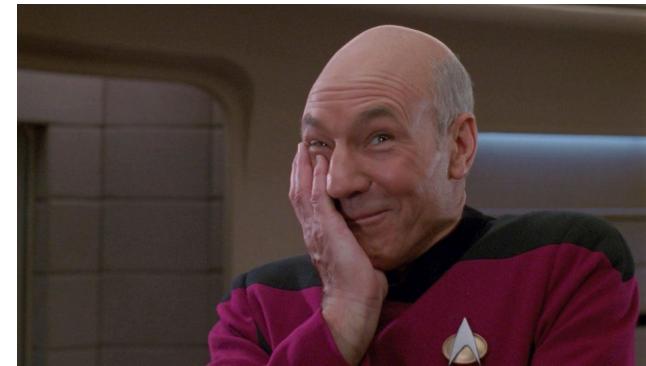
```
@Test  
void iWantToBelieve() {  
    // setup  
    Integer actual = null  
  
    // run  
    asyncNextPrime( 1728, { answer -> actual = answer } )  
  
    // verify  
    assert actual == 1733  
}
```



Episode VII: Time travel

```
@Test  
void hm() {  
    // setup  
    Integer actual = null  
  
    // run  
    asyncNextPrime( 1728, { answer -> actual = answer } )  
}
```

```
// hm  
Thread.sleep(3000)  
  
// verify  
assert actual == 1733  
}
```

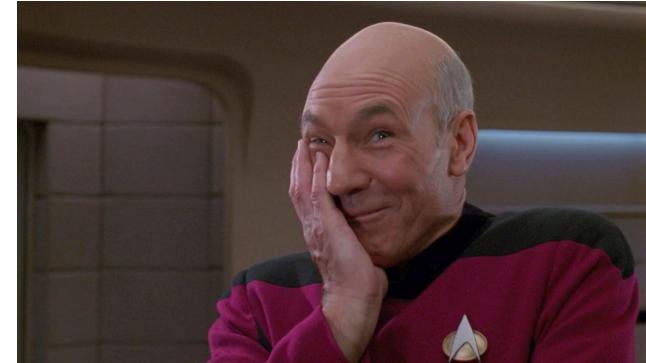


Episode VII: Time travel

```
@Test(timeout = 3000L)
void complicatedHm() {
    // setup
    Integer actual = null

    // run
    asyncNextPrime( 1728, { answer -> actual = answer } )

    // hmhm
    while (actual == null) {
        if (actual != null) {
            // verify
            assert actual == 1733
        }
    }
}
```



Episode VII: Time travel

```
def conditions = new PollingConditions(timeout: 30)

def 'find next prime after 1728 eventually'() {
    setup: 'holder for answer'
    Integer actual = null

    when: 'async start calculating next prime after 1728'
    asyncNextPrime( 1728, { answer -> actual = answer } )

    then: 'eventually answer will be found'
    conditions.eventually {
        assert actual == 1733
    }
}
```

Episode VII: Time travel

```
def conditions = new PollingConditions(timeout: 30)

def 'find next prime after 2817 within 3 seconds'() {
    setup: 'holder for answer'
    Integer actual = null

    when: 'async start calculating next prime after 1728'
    asyncNextPrime( 2817, { answer -> actual = answer } )

    then: 'within 3 seconds answer will be found'
    conditions.within(3) {
        assert actual == 2819
    }
}
```

Episode VII: Time travel

```
def conditions = new AsyncConditions()

def 'find next prime after 2817 within 3 seconds'() {
    when: 'async start calculating next prime after 1728'
        asyncNextPrime( 1728, { answer ->
            conditions.evaluate { assert answer == 1733 }
        } )
}

then: 'within 3 seconds answer will be found'
    conditions.await(3)
}
```



Episode VIII: Test control

Episode VIII: Test control

```
@Ignore("will fix it before commit")
@Test
public void alwaysIgnor    // TODO FIXME test
}                           @Test(enabled = false)
                            public void alwaysIgnored() {
                            @Disabled("will fix it before commit")
                            @Test
                            public void alwaysIgnored() {
                                // TODO FIXME test is failing
                                assertEquals(5, 2+2);
                            }
                            als(2+2, 5);
                            IXME test is failing
```

Episode VIII: Test control

```
@Ignore('will fix it before commit')
def 'this test is always ignored'() {
    // TODO FIXME test is failing
    expect: 'that 2+2=5'
    2 + 2 == 5
}
```

Episode VIII: Test control

```
@Test
public void ignoredOnWindows() {
    assumeThat(System.getProperty("os.name").toLowerCase(), is(not(containsString("windows"))));
```

Episode VIII: Test control

```
@Requires({ jvm.java8 && env['JAVA_HOME'] != null })  
def 'this test requires JAVA_HOME set and Java 8 installed'() {  
    expect: 'that we are on Java 8'  
    'java -version'.execute().errorStream.text.contains('java version "1.8.0_73"')  
}
```

Episode VIII: Test control

```
@IgnoreIf({ os.windows || sys['pretend.os'] == 'windows' })
def 'this test is ignored on Windows'() {
    expect: 'that we are not on Windows'
    !System.properties['os.name'].toString().toLowerCase().contains('windows')
}
```

Episode VIII: Test control

```
@IgnoreIf({ new Random().nextBoolean() })
def 'this test is SOMETIMES ignored'() {
    when: 'i hate my job'
    Integer.metaClass.plus = { Integer other ->
        return 5
    }

    then: 'i can make them pay'
    2 + 2 == 5
}
```

Episode VIII: Test control

```
@Requires({ N40S_ConditionalRuns_Part2.isGoogleSearchAvaiable() })  
def 'this test runs only if Google Search is avaible'() {  
    setup: 'http connection service'  
    def http = new HTTPBuilder('https://google.com')  
  
    when: 'we search for the best java unit testing framework'  
    def response = http.get(path : '/search', query : [q:'best java unit testing framework'])  
  
    then: 'answer mentions spock'  
    response.toString().toLowerCase().contains 'spock'  
}
```

Episode VIII: Test control

```
@IgnoreRest
```

```
def 'this test makes all other test ignored'() {  
    expect: 'a miracle'  
    2 + 2 == 5  
}
```

Episode VIII: Test control

```
@PendingFeature
```

```
def 'this test is passing'() {  
    expect: 'that 2*2=4'  
    2 * 2 == 4  
}
```



this test is passing



this test is failing

```
@PendingFeature
```

```
def 'this test is failing'() {  
    expect: 'that 2*2=5'  
    2 * 2 == 5  
}
```

12ms

3ms

Episode VIII: Test control

```
@Category(Fast.class)
```

```
@Test
```

```
public void passedFast() {  
    assertThat(2+2, is(4));  
}
```

```
@Category(Slow.class)
```

```
@Test
```

```
public void failingIn20Seconds() throws InterruptedException {  
    Thread.sleep(TimeUnit.SECONDS.toMillis(20));  
    assertThat(2+2, is(4));  
}
```

Episode VIII: Test control

```
!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
@Test(groups = {"fast", "slow"})
public void testSuite() {
    // ...
}

@Test(groups = {"slow"})
public void testSlowSuite() {
    // ...
}
```

Episode VIII: Test control

```
@Tag("fast")  
@Test  
public void passedFast() {  
    assertEquals(4, 2+2);  
}
```

```
@Tag("slow")  
@Test  
public void failingIn20Seconds() throws InterruptedException {  
    Thread.sleep(TimeUnit.SECONDS.toMillis(20));  
    assertEquals(5, 2+2);  
}
```

Episode VIII: Test control

```
@Fast  
def passedFast() {  
    expect: 'that everything is ok'  
    2 + 2 == 4  
}  
  
@Slow
```

```
def failingIn20Seconds() {  
    setup: 'some resource'  
    Thread.sleep(TimeUnit.SECONDS.toMillis(20))  
  
    expect: 'that everything is not ok'  
    2 + 2 == 5  
}
```

```
runner {  
    include Fast  
    exclude Slow  
  
    optimizeRunOrder true  
}
```

N44S_TestSuites		1m 0s 44ms
!	failingIn10Seconds	10s 718ms
!	failingIn20Seconds	20s 5ms
OK	passedFast	1ms
OK	passedIn10Seconds	10s 16ms
OK	passedIn20Seconds	20s 4ms



Episode IX: Encore

Episode IX: Encore

```
class ReportExtension extends AbstractAnnotationDrivenExtension<Report> {

    @Override
    void visitSpecAnnotation(Report annotation, SpecInfo spec) {
        spec.addListener(new AbstractRunListener() {

            @Override
            void afterFeature(FeatureInfo feature) {
                for (block in feature.blocks) {
                    for (text in block.texts) {
                        println "${block.kind.name().toLowerCase()} $text"
                    }
                }
            }
        })
    }
}
```

Episode IX: Encore

```
@Stepwise
class N41S_SystemProperties extends Specification {

    @RestoreSystemProperties
    def 'set spock version'() {
        expect: 'that spock.version is not set'
        System.getProperty('spock.version') == null

        when: 'spock version is set'
        System.setProperty('spock.version', '1.0')

        then: 'we can retrieve its value back'
        System.getProperty('spock.version') == '1.0'

    }

    def 'check spock version is not set'() {
        expect: 'that spock.version is not set'
        System.getProperty('spock.version') == null
    }
}
```

Episode IX: Encore

```
then: 'list is not empty, is of size 4, and contains all added values'  
with(list) {  
    empty == false  
    size() == 4  
    get(0) == 'we'  
    get(1) == 'will'  
    get(2) == 'love'  
    get(3) == 'spock'  
}
```

Episode IX: Encore

```
then: 'list is not empty, is of size 3, and contains all added values'  
verifyAll {  
    list.empty == false  
    list.size() == 3  
    list.get(0) == 'we'  
    list.get(1) == 'love'  
    list.get(2) == 'spock'  
}
```

Episode IX: Encore

```
@org.junit.Rule OutputCapture capture = new OutputCapture()

def "capture output print method"() {
    when: 'text is printed to console'
    print "2 + 2 = ${2+2}"

    then: 'it is printed as expected'
    capture.toString() == '2 + 2 = 4'
}
```

Episode IX: Encore

```
@ConfineMetaClassChanges([Integer])
def 'sometimes 2 + 2 = 5'() {
    setup: 'very special Integer + Integer operation'
    Integer.metaClass.plus = { Integer other ->
        return 5
    }

    expect: '2 + 2 == 5'
    2 + 2 == 5
}

def 'usually 2 + 2 == 4'() {
    expect: '2 + 2 == 4'
    2 + 2 == 4
}
```

Episode IX: Encore

```
@ContextConfiguration(classes = Config)
class N22S_Stubs extends Specification {

    @Autowired
    PasswordEncoder passwordGenerator
```

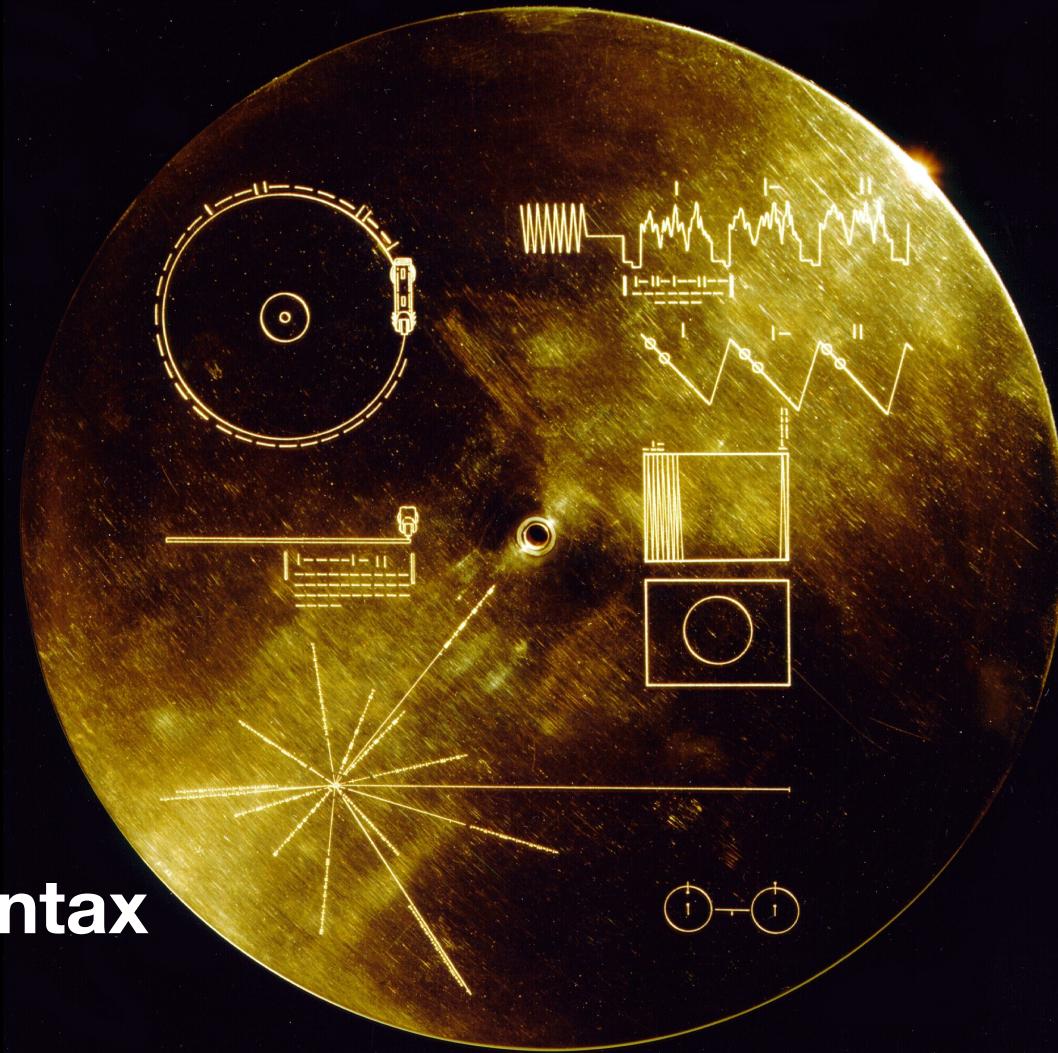
Episode IX: Encore

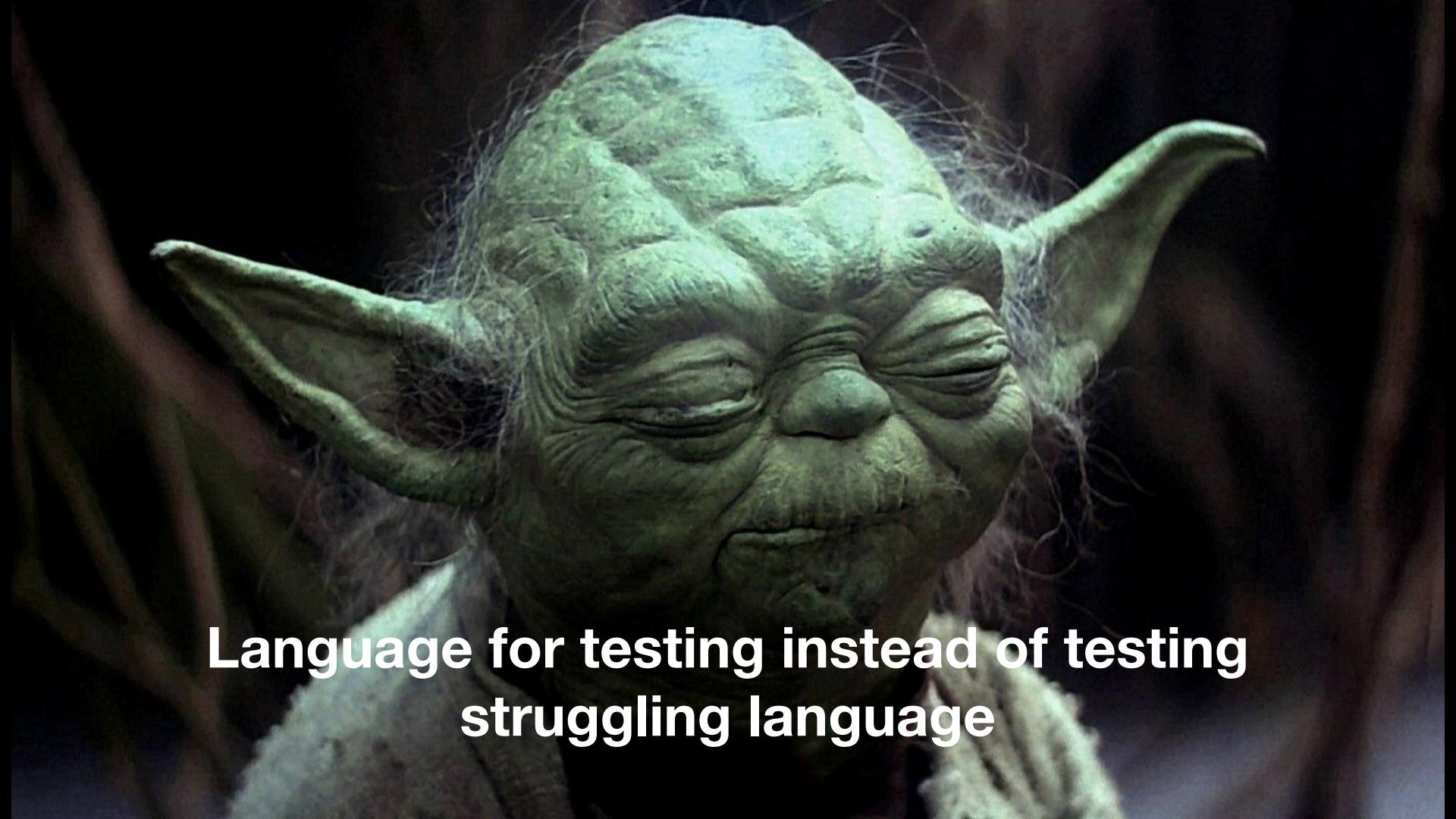
```
class N49S_Geb extends GebSpec {  
  
    def 'search for wikipedia in google'() {  
        when:  
            go 'http://www.google.com'  
  
        then:  
            title == 'Google'  
  
        when:  
            $('input', name: 'q').value('wikipedia')  
            $('input', name: 'btnG').click()  
  
        then:  
            waitFor { title == 'wikipedia - Поиск в Google' }  
  
        when:  
            def result = $('*', 0)  
  
        then:  
            result.text().contains('ru.wikipedia.org')  
    }  
}
```

Why one should use Spock?



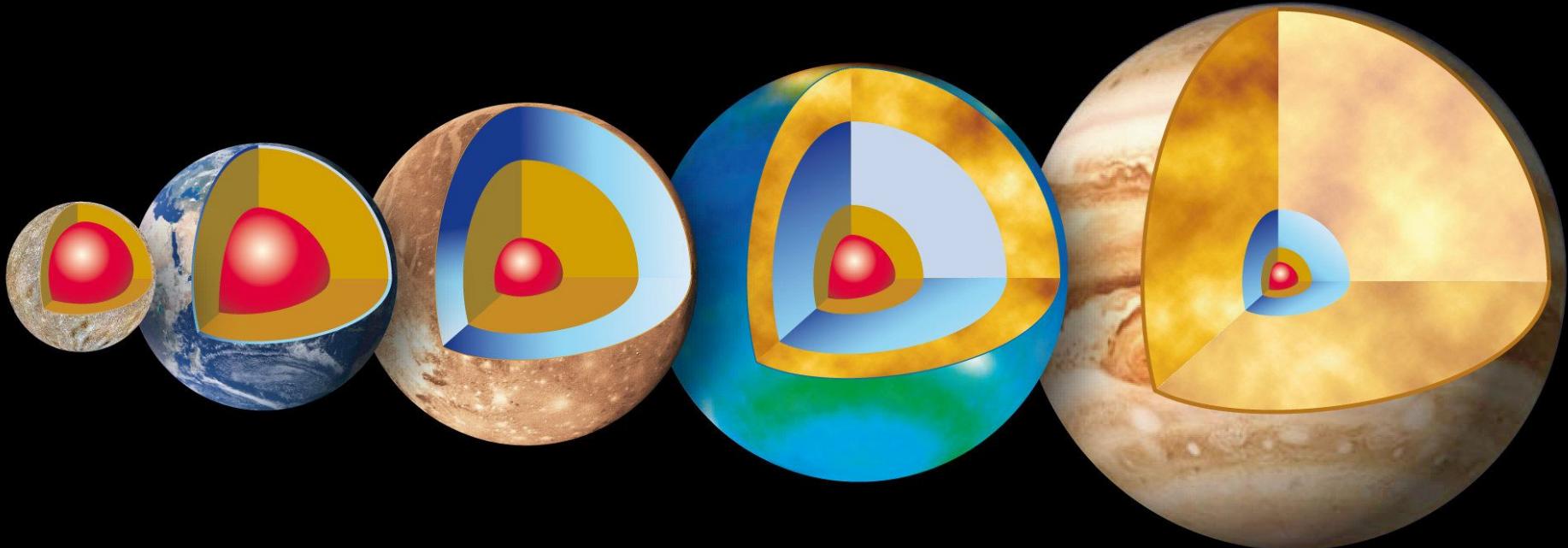
Concise syntax





**Language for testing instead of testing
struggling language**

Clear test structure





Code and reports readable for everyone

Powerful built-in features



Extensions



One framework for all
kinds of testing



Behaves as one more JUnit runner

```
@RunWith(Sputnik.class)
public abstract class Specification extends MockingApi {
```

Works for both Groovy and Java





Why one should not use Spock?

JUnit/TestNG has 90% similar/imitable features



Spock lost its lead developer (long time ago)

spockframework / spock

Watch 119 Star 1,234 Fork 236

Code Issues Pull requests Pulse Graphs

Contributors Commits Code frequency Punch card Network Members

Feb 15, 2009 – Sep 10, 2016

Contributions to master, excluding merge commits

Contribution chart showing spikes of activity between 2010 and 2015.

User	Commits	Contributor ID
pniederw	1,364	#1
alkemist	176	#2
leonard84	34	#3
robfletcher	7	#4

Peter Niederwieser
Software Engineer at Apple
Австрия | Программное обеспечение

Сейчас Apple
Ранее Gradleware, Smarter Ecommerce GmbH, Siemens AG
Образование Johannes Kepler Universität Linz

Отправить InMail участнику Peter

155 контактов

<https://at.linkedin.com/in/pniederwru>

Контактные сведения

Профессиональные достижения

Опыт работы

Software Engineer
Apple
март 2015 – настоящее время (1 год 3 месяца)

Principal Engineer
Gradleware
январь 2011 – декабрь 2014 (4 года)

Apple

Gradle

When one can use Spock?





Your team is
ready to make
one step
forward

OTV

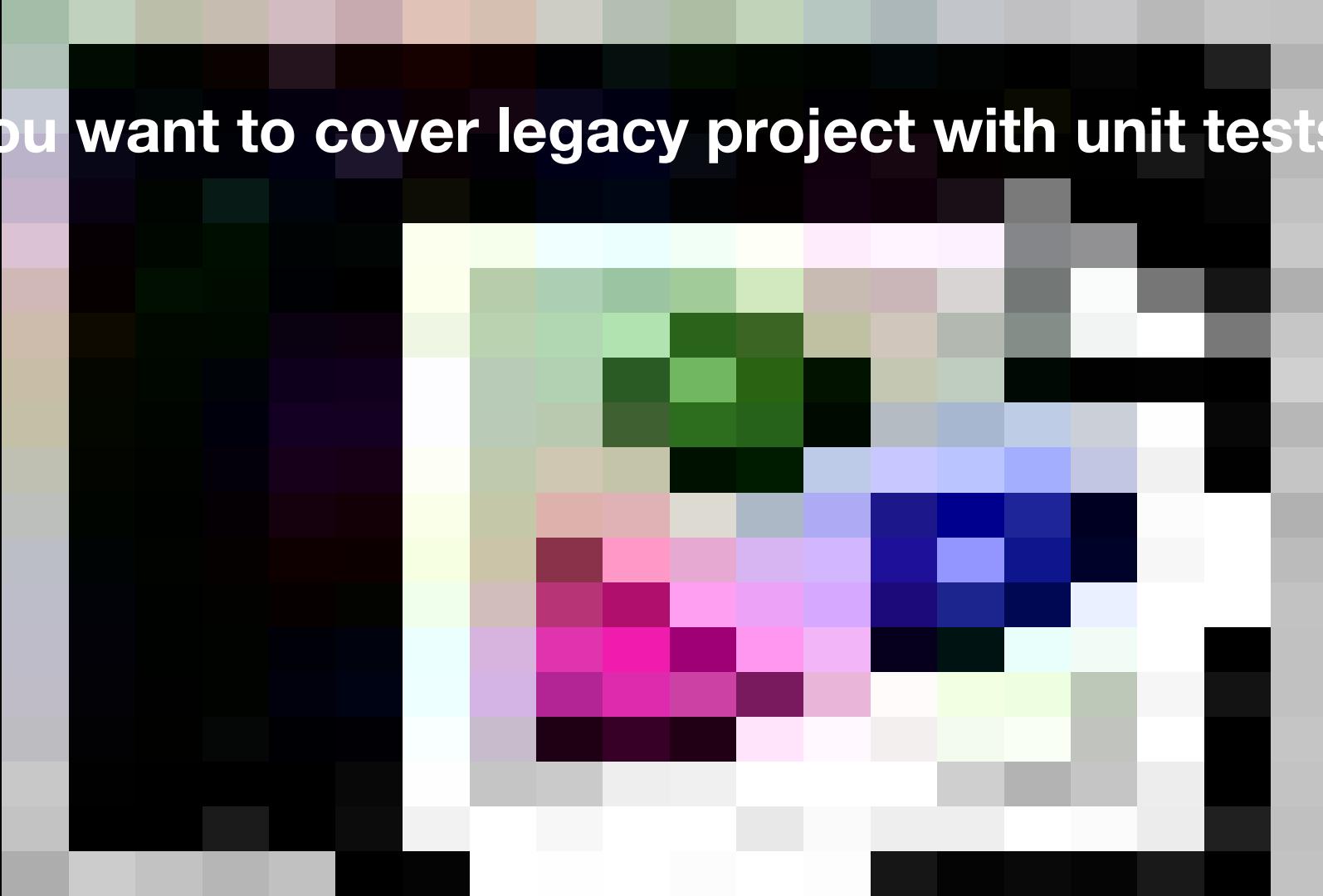
You are about to start new project

Episode IV

A NEW HOPE

*It is a period of civil war.
Rebel spaceships, striking
from a hidden base, have won
their first victory against
the evil Galactic Empire.*

*During the battle, Rebel
spies managed to steal secret
plans to the Empire's
ultimate weapon, the DEATH
STAR, an armored space*



You want to cover legacy project with unit tests

You want to introduce functional testing in your project



What's next?

[Source code from this talk](#)

[Article based on this talk](#)

<https://twitter.com/yermilov17>

<https://www.facebook.com/yaroslav.yermilov>

<https://ua.linkedin.com/pub/yaroslav-yermilov/58/682/506>

What's next?

[Spock documentation](#)

[Spock source repositories](#)

[Spock examples project](#)

[Spock smoke tests](#)

[Spock Web Console](#)

[Spock reports](#)

What's next?

[mrhaki about Spock](#)

[Spock vs JUnit by Kostis Kapelonis and Athens Greece](#)

[Smarter testing Java code with Spock Framework by Marcin Zajęczkowski](#)

[Idiomatic Spock by Rob Fletcher](#)

[Spock: A Logical Framework for Enterprise Testing by Ken Kousen](#)

What's next?

[Java Testing with Spock by Konstantinos Kapelonis](#)

[Geb documentation](#)

[Geb for browser automation by Jacob Aae Mikkelsen](#)

A portrait of Mr. Spock from Star Trek, wearing his iconic blue uniform with gold piping and a Starfleet insignia. He has his characteristic Vulcan ponytail and is making a hand gesture with his right hand, palm facing forward. His expression is neutral to slightly stern.

Thanks!