

**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



<https://twitter.com/yermilov17>

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

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

# About Spock

## Spock

From Wikipedia, the free encyclopedia

This article is about the *Star Trek* character. For the pediatrician, see *Benjamin Spock*. For other uses, see *Spock (disambiguation)*.

**Spock**, commonly **Mr. Spock** (sometimes popularly referred to as: **Spock, son of Sarek**), is a fictional character in the *Star Trek* media franchise. Spock was first portrayed by Leonard Nimoy in the original *Star Trek* series, and also appears in the animated *Star Trek* series, a two-part episode of *Star Trek: The Next Generation*, eight of the *Star Trek* feature films, and numerous *Star Trek* novels, comics, and video games.<sup>[1][2]</sup> In addition to this, numerous actors portrayed the various stages of Spock's rapid growth, due to the effects of the Genesis Planet, in the 1984 *Star Trek* film *Star Trek III: The Search for Spock*. In the 2009 film *Star Trek*, Nimoy reprised his role with *Zachary Quinto*, who depicted a younger version of the character, existing within an alternate timeline. Both reprised their roles in the 2013 sequel *Star Trek Into Darkness* and Quinto will reprise his role again in 2016 *Star Trek Beyond*.<sup>[3]</sup>

Spock serves aboard the starship *Enterprise*, as science officer and *first officer*, and later as commanding officer of two iterations of the vessel. Spock's mixed *human-Vulcan* heritage serves as an important plot element in many of the character's appearances. Along with Captain James T. Kirk and Dr. Leonard "Bones" McCoy, he is one of the three central characters in the original *Star Trek* series and its films. After retiring from Starfleet, Spock serves as a Federation ambassador, contributing toward the easing of the strained relationship between the Federation and the *Klingon Empire*. In his later years, he serves as Federation ambassador to the *Romulan Star Empire* and becomes involved in the ill-fated attempt to save Romulus from a supernova.<sup>[3]</sup>

### Contents [hide]

<b>1 Appearances</b>
1.1 Original series television and films
1.2 <i>Star Trek: The Next Generation</i>
1.3 <i>Star Trek</i> (2009)
1.4 <i>Star Trek Into Darkness</i> (2013)
<b>2 Development</b>
2.1 Death in <i>The Wrath of Khan</i>
2.1.1 Reaction to Spock's death
2.2 Recasting
<b>3 Reception</b>
3.1 <i>Star Trek</i> (2009)
<b>4 Cultural impact</b>
4.1 Fan productions
4.2 "Spocking" Canadian \$5 notes
<b>5 See also</b>
<b>6 References</b>
6.1 Footnotes
6.2 Resources
<b>7 External links</b>

Spock



Leonard Nimoy as Spock (left) beside William Shatner as James T. Kirk

**Species** Half-Vulcan (paternal)  
Half-Human (maternal)

**Affiliation** Starfleet  
Vulcan Government

**Posting** USS *Enterprise*  
Second officer/Science officer  
Executive officer/Science officer  
commanding officer  
USS *Enterprise*-A  
Executive officer/Science officer  
Federation Ambassador-at-Large

**Rank** Lieutenant commander  
Commander

# Disclaimer



Peter Niederwieser  
@pniederw



Luke Daley  
@ldaley

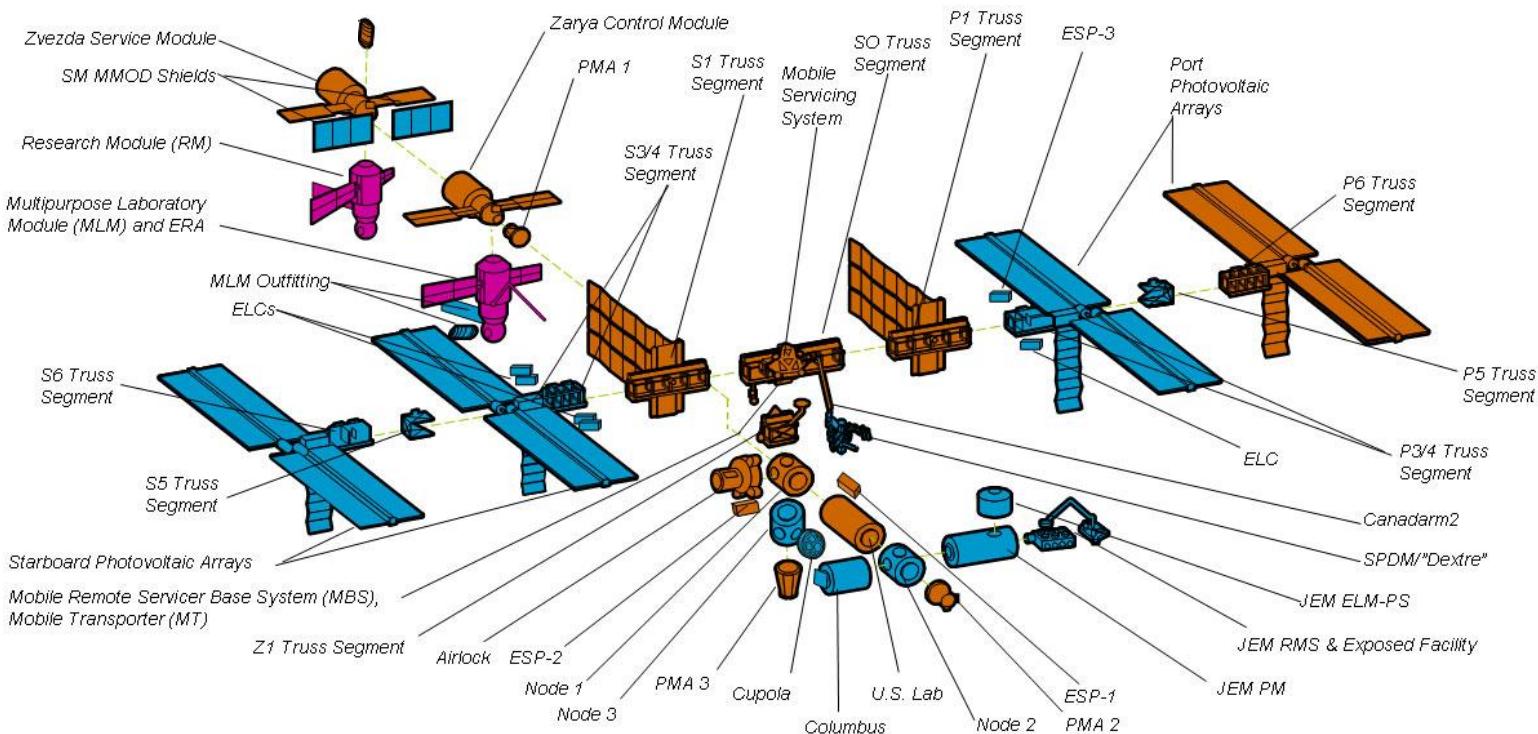
A close-up shot of the character Spock from the Star Trek franchise. He has his signature dark brown hair in a flat-top style and is wearing a light blue Starfleet uniform. His expression is one of intense anger or frustration, with his eyebrows raised and mouth open as if shouting. The background is a blurred red and grey.

What's the problem?



**Better to test before going to production**

# Firstly, to unit-test



**So what?**



# JUnit?



# JUnit 5?

May 21

junit-team / junit5

Code Issues 48 Pull requests (Track A)

Labels Milestones

3 Open ✓ 1 Closed

## 5.0 Backlog

No due date ⏲ Last updated 7 days ago

General backlog for issues that are being considered for inclusion in the next release.

## 5.0 M1

Due by May 31, 2016 ⏲ Last updated a few hours ago

First major milestone release for JUnit 5

### Epic:

- Stable, documented IDE-facing API
- Dynamic tests

Stream time (1st queue)

## 5.0 M2

Due by July 31, 2016 ⏲ Last updated about 17 hours ago

Track D  
Big Data/Data Processing  
400 seats

Track E  
Tools  
400 seats

Some coffee

Search and analyze your data with ElasticSearch  
Anton Udovychenko

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

Lambda Architecture with Apache Spark  
Taras Matyashovskyy

Implement your own profiler with blackjack and fun  
Vladimir Sitrnikov

Streams, flows and storms – how not to drown with your data?  
Maciek Próchniak

JUnit 5: Next Generation Testing on the JVM  
Nicolai Parlog



# JUnit 3?



# TestNG?



# Unit testing framework on steroids?

Hamcrest

AssertJ

Google Truth

Mockito  
PowerMock  
JMockit



# Spock?



Something odd?

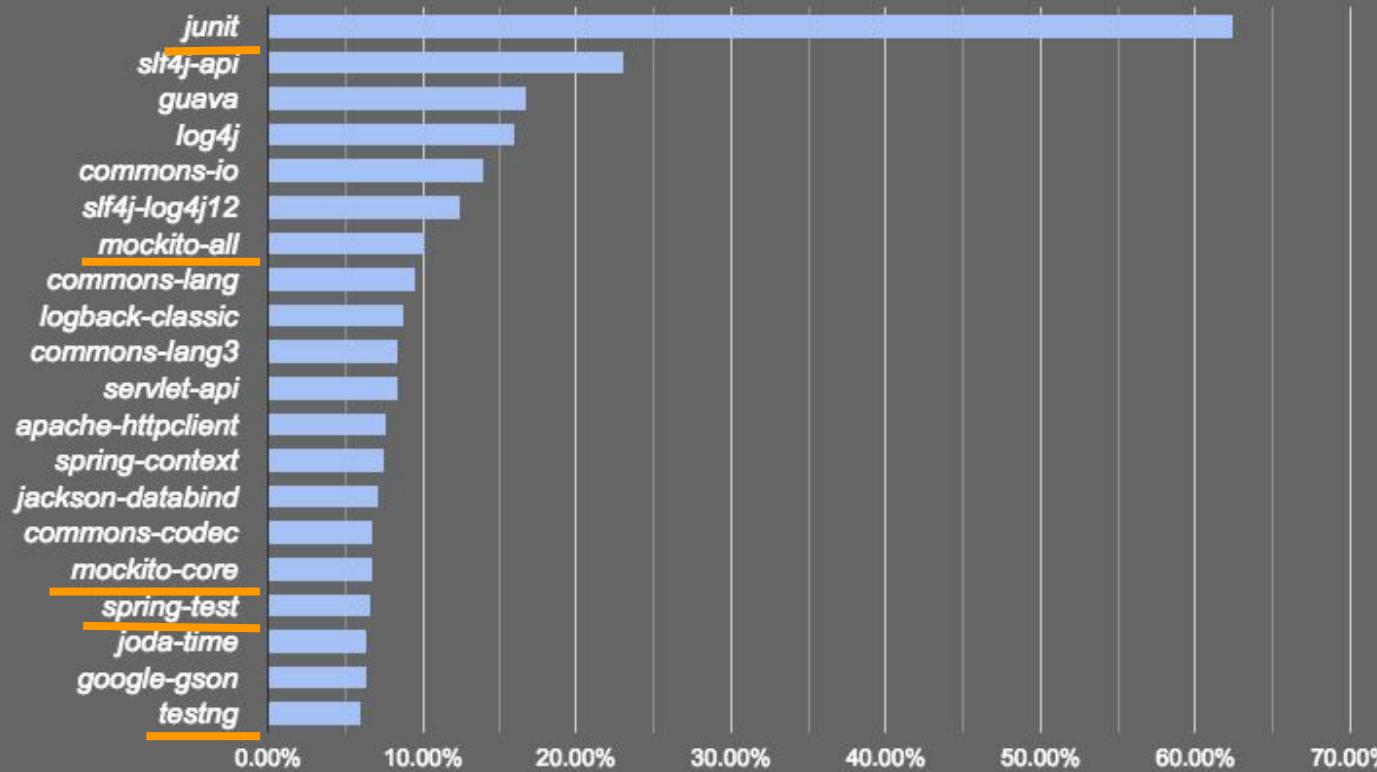


Cucumber  
J8Spec  
JGiven

# Nothing?

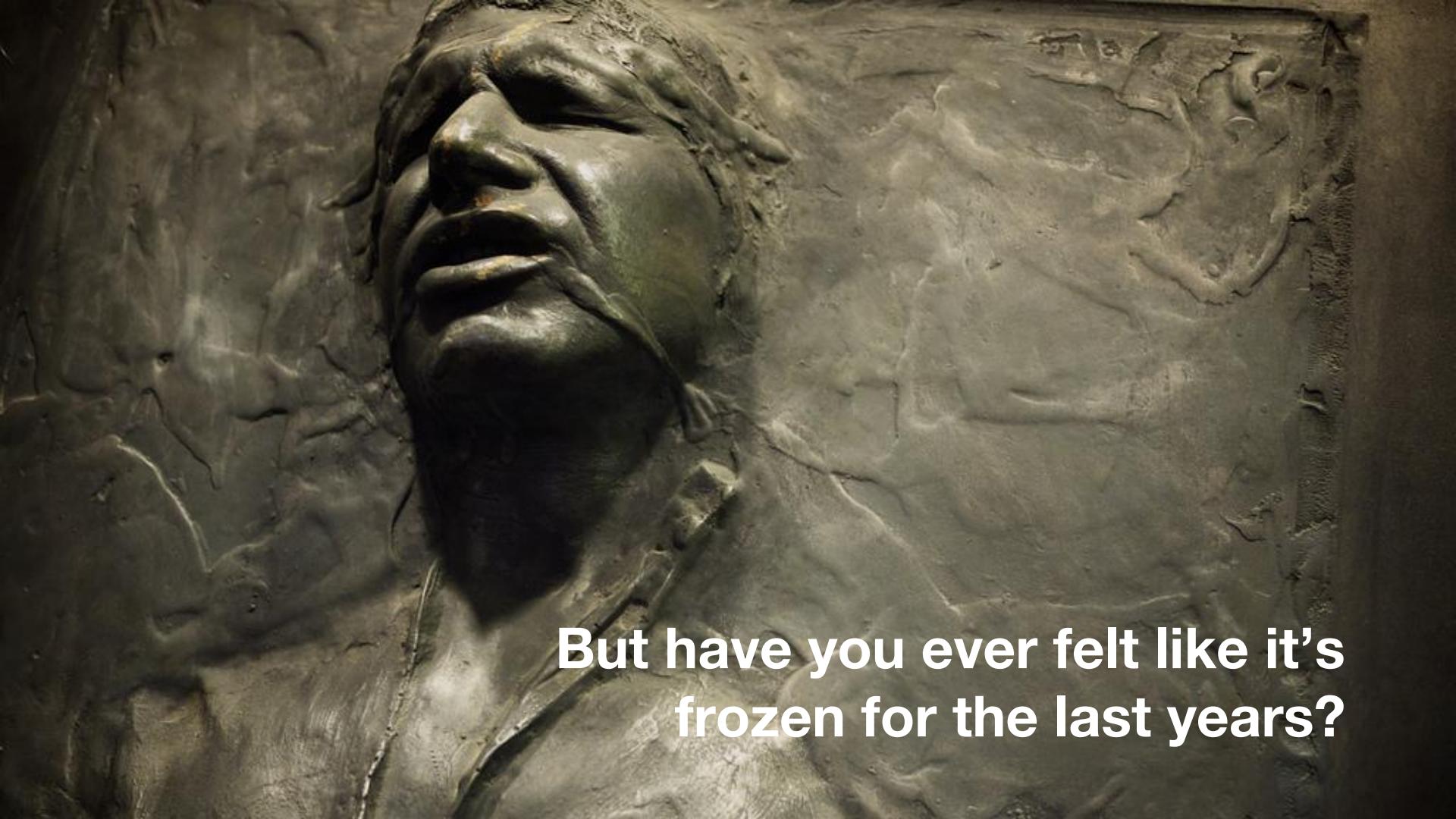


# The Top 20 Libraries Used by Github's Most Popular Java Projects



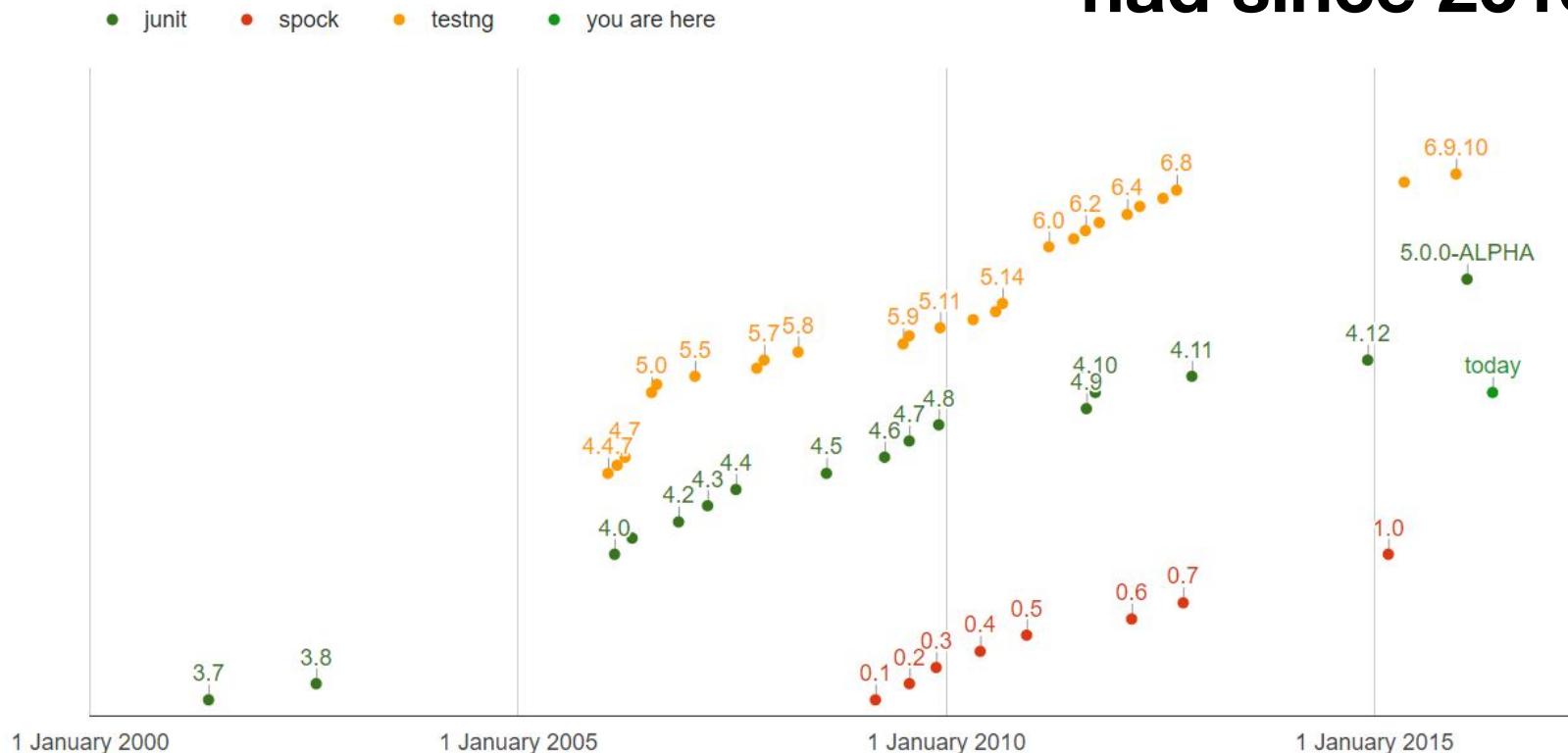


**Automated testing  
is a great concept**

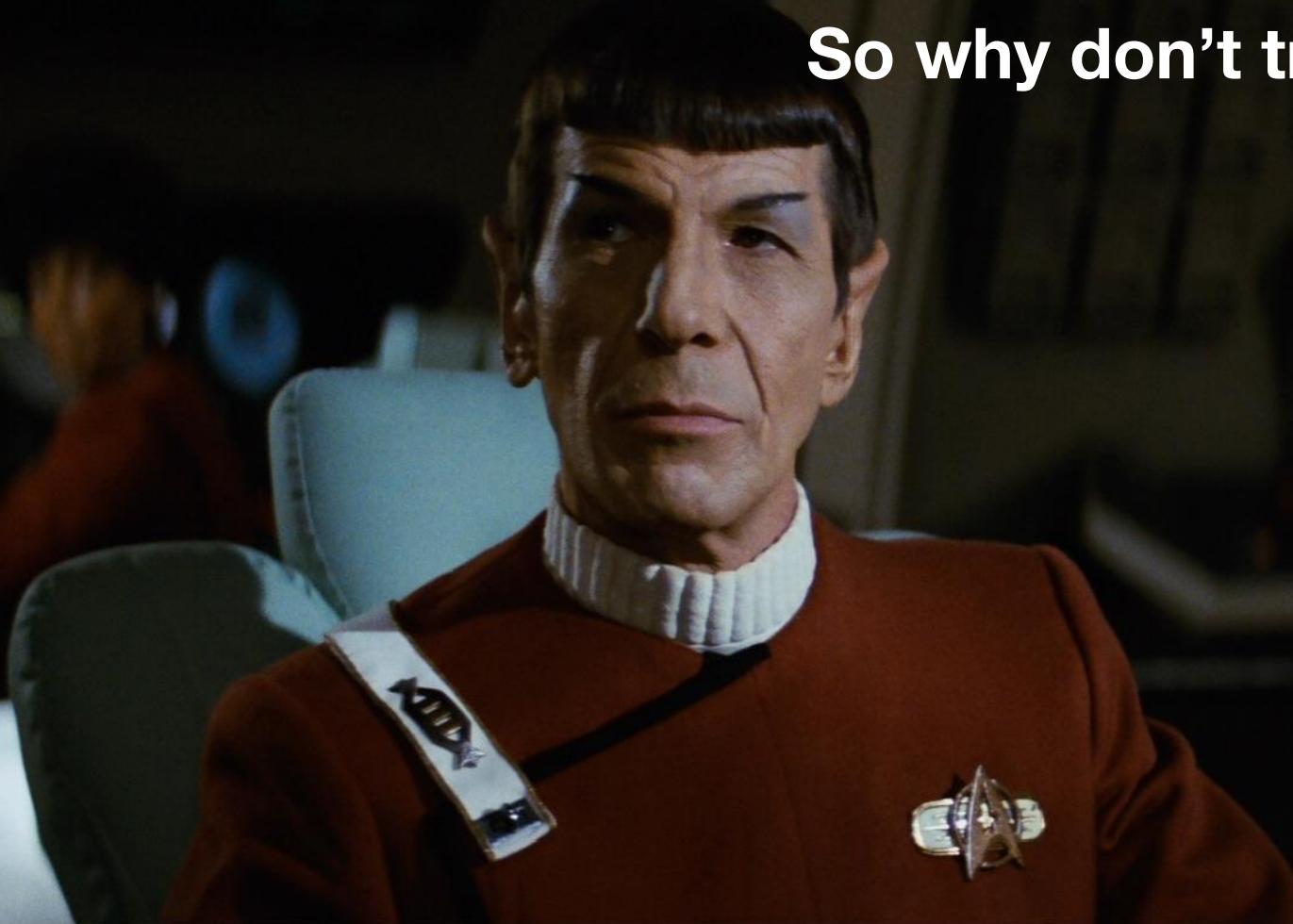


**But have you ever felt like it's  
frozen for the last years?**

# QUIZ: How many test framework releases we had since 2013?



So why don't try 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.

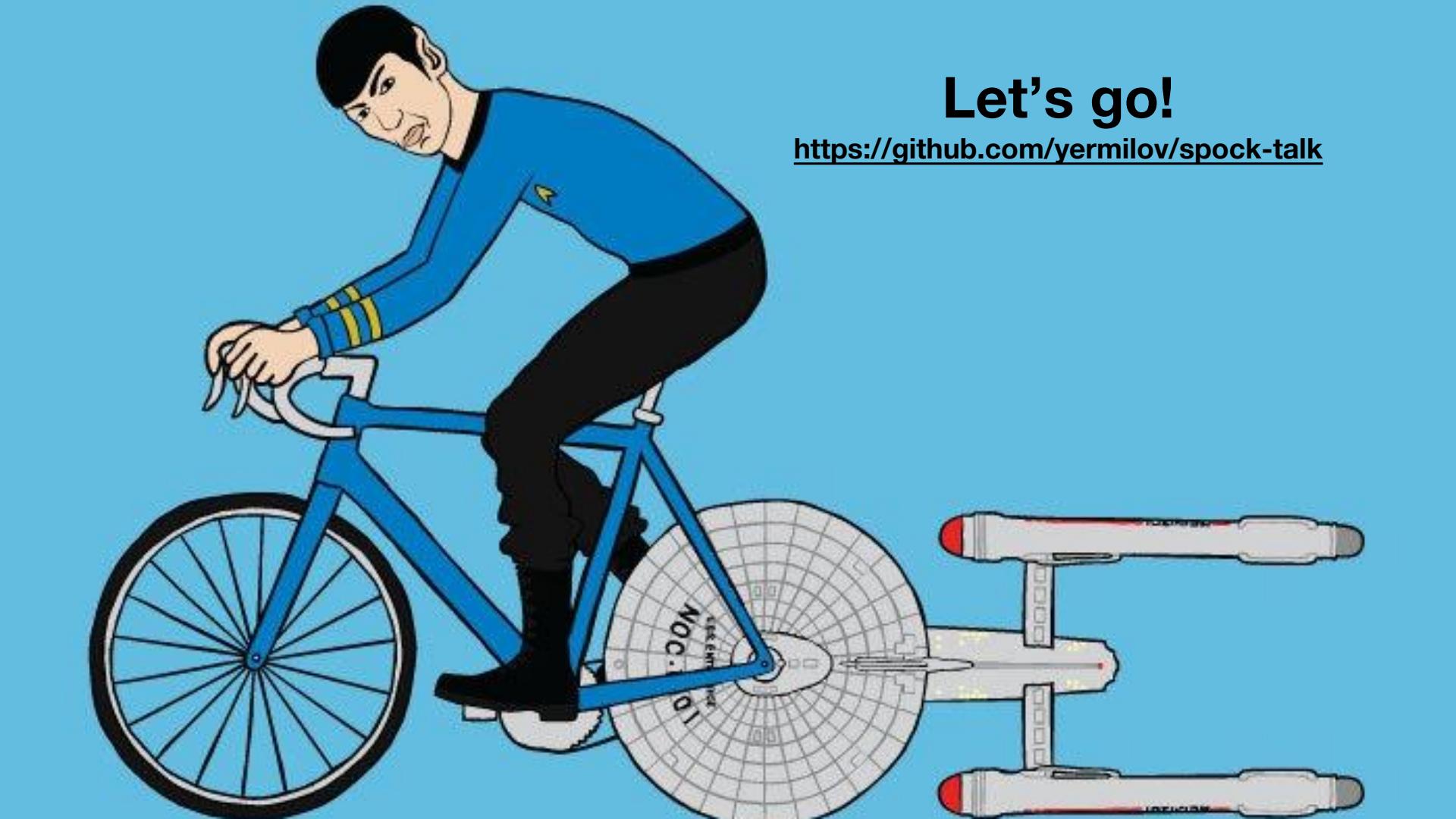
**STAR TREK**

**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_is()  
    ArrayList<String> list;  
    list.add("we");  
    list.add("is");  
    assertEquals(4, list.size());  
  
import org.junit.gen5.api.Test;  
import static org.junit.gen5.api.Assertions.assertEquals;
```



# 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

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.verifyng array list size
```

# 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:

```
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

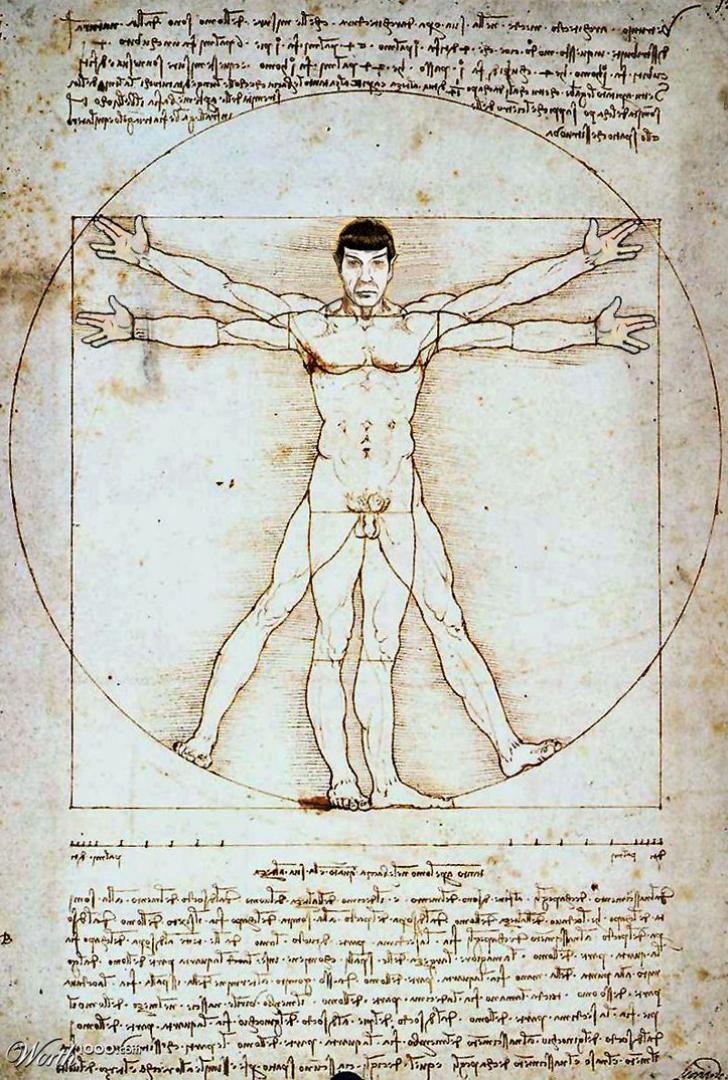
Condition not satisfied:

```
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

```
@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: Idiomatic Spock



# Episode III: Idiomatic Spock

```
@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: Idiomatic Spock

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

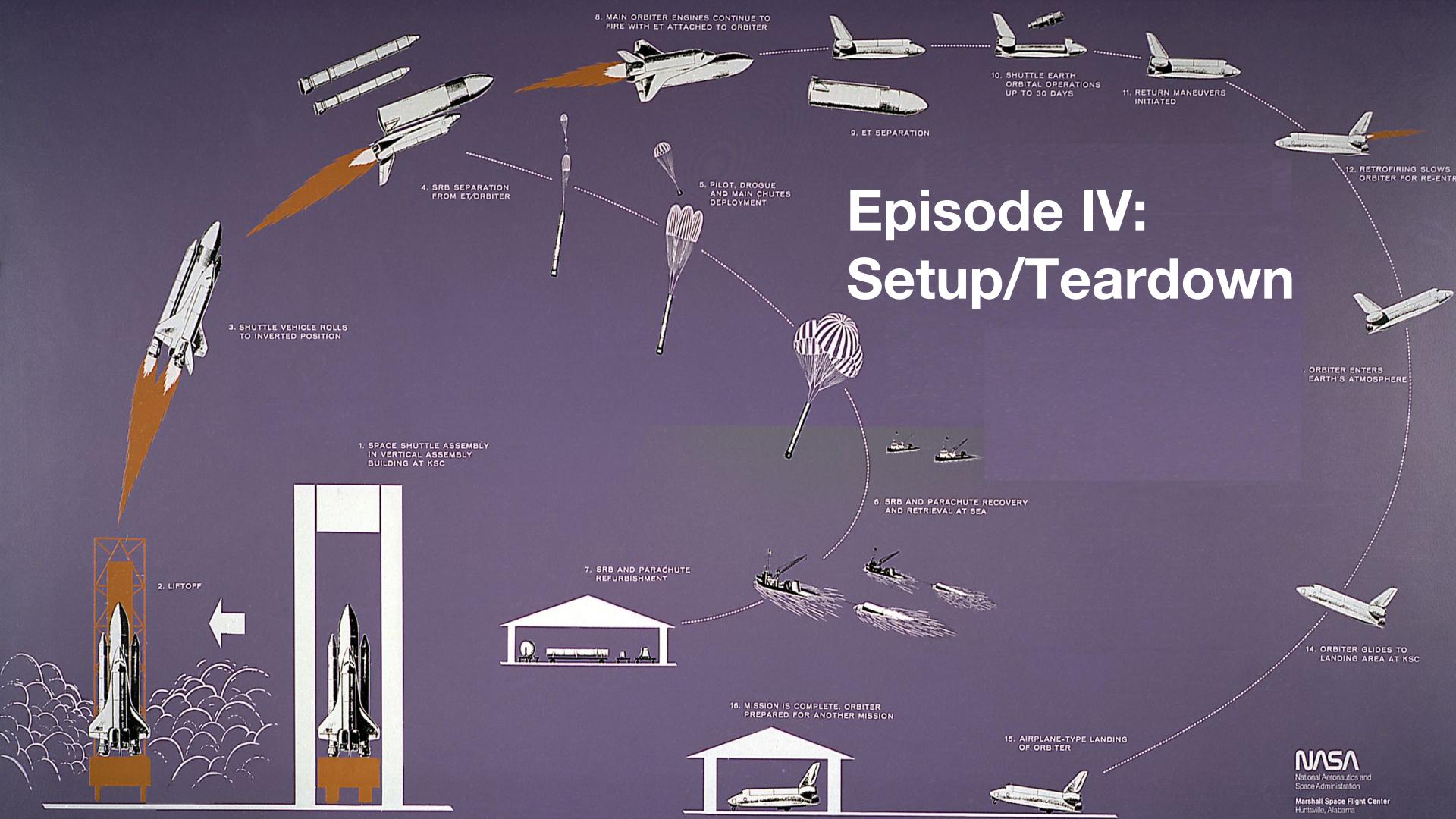
*When:* add value to list

*And:* add one more value to list

*Then:* array list size should be 2

*When:* add two more values into list

# Episode IV: Setup/Teardown



NASA

National Aeronautics and  
Space Administration

Marshall Space Flight Center

Huntsville, Alabama

# Episode IV: Setup/Teardown

```
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()
```

```
}
```

```
@Before
```

```
pub
```

```
icData() {
```

```
t into testing_tool val
```

```
import org.junit.After  
import org.junit.AfterClass  
import org.junit.Before  
import org.junit.BeforeClass  
import org.junit.Test
```

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

# Episode IV: Setup/Teardown

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

# Episode IV: Setup/Teardown

```
static Sql sql
```

```
@BeforeAll
```

```
public static void createTable() {  
    sql = Sql.newInstance("jdbc:mysql://localhost:3306/testin  
    sql.execute("create table tool (name, count)");  
}
```

```
@AfterAll
```

```
public static void dropTable() {  
    sql.execute("drop table testin  
    sql.close()
```

```
@BeforeEach
```

```
public "
```

```
import org.junit.gen5.api.AfterAll
```

```
import org.junit.gen5.api.AfterEach
```

```
{
```

```
string_tool values
```

```
@Test
```

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

# Episode IV: Setup/Tear down

```
static Sql sql
```

```
def setupSpec() {
```

```
    sql = Sql.newInstance("jdbc:  
    sql.execute("create table te
```

```
}
```

```
def setup() {
```

```
    sql.execute("insert into testing_tool values
```

```
}
```

```
def cleanup() {
```

```
    sql.execute("delete from tes
```

```
}
```

```
def cleanupSpec() {
```

```
    sql.execute("drop table testing_tool")  
    sql.close()
```

```
}
```

```
def 'tool count'() {
```

```
when: 'we count number of unit t
```

```
def actual = sql.firstRow("selec
```

```
then: 'it should be 3'
```

```
actual.toolCount == 3
```

```
}
```

# Episode IV: Setup/TearDown

```
@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

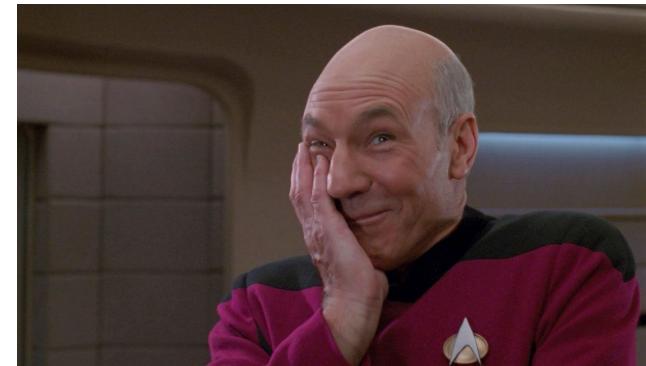
    then: 'it should be 2'
        actual.toolCount == 2

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

# Episode IV: Setup/Teardown

```
// 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 IV: @Shared or static?



A wide-angle shot from Star Wars: Episode V - The Empire Strikes Back. It depicts a massive hangar bay of the Death Star. In the foreground, thousands of Stormtroopers stand in formation on a dark platform. A massive Imperial transport ship, the Venator-class Star Destroyer, is docked at a long loading ramp that extends from the hangar floor up towards the massive cylindrical ceiling of the Death Star's hull. The sky outside shows a sunset or sunrise with orange and yellow hues. The overall atmosphere is one of immense scale and military might.

# Episode V: Data driven test

# Episode V: 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 V: 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 V: 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 V: 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 V: Data driven test

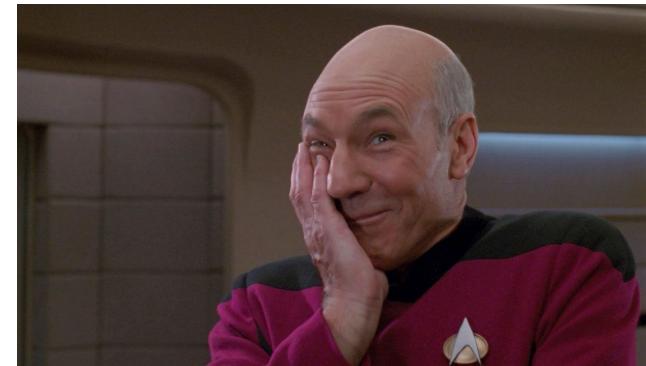
```
@RunWith(Parameterized.class)
public class N16J_DataTables {

    @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;
```



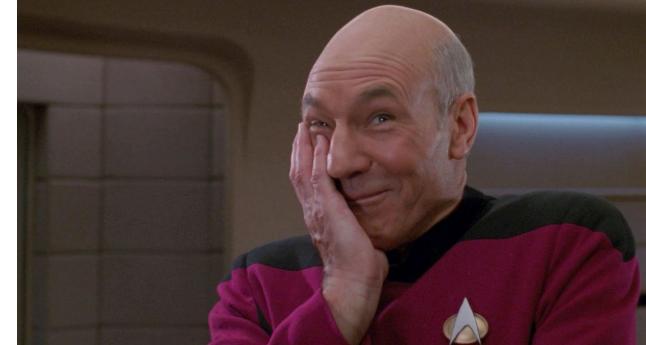
# Episode V: Data driven test

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 V: Data driven test

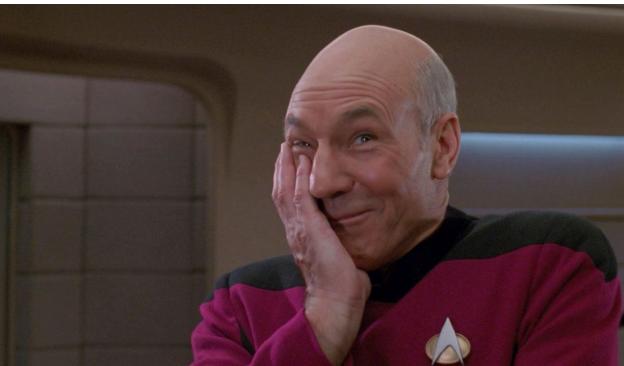
```
@DataProvider(name = "data")
public static Object[][] data() {
    return new Object[][] {
        { 0.0, 6.0, 0.0, 10.0, 60.0, 6.0 },
        { 5.0, 0.0, 3.0, 3.0, 17.0, 9.0 },
        { -50.0, 10.0, -1.0, 10.0, 0.0, 0.0 }
    };
}

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



# Episode V: 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



A Star Trek: Discovery promotional image. The starship Discovery, with its hull number NCC-1031 partially visible, is shown from a low angle, flying through a field of stars and nebulae. The ship's nacelles are prominent, and its hull is covered in various panels and equipment. The background features a large, hazy planet or star system.

# Episode VI: Exceptions

# Episode VI: Exceptions

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

# Episode VI: 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 VI: 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 VI: Exceptions

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

    // run
    arrayList.size();

    // verify no exception is thrown
}
```

# Episode VI: 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 VI: 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)  
}
```

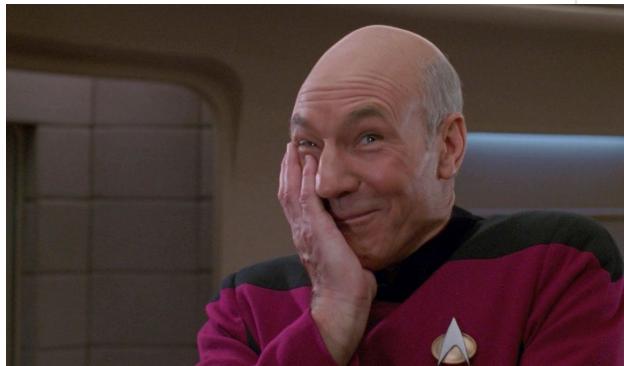
# Episode VI: Exceptions

```
@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 VI: Exceptions

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



# Episode VI: 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 with a black collar. All figures have a serious expression.

# Episode VII: Mocks, Stubs, Spies

# Episode VII: Mocks, Stubs, Spies

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

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

# Episode VII: 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 VII: Mocks, Stubs, Spies

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

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

# Episode VII: 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 VII: Mocks, Stubs, Spies

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

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

# Episode VII: 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 VII: Mocks, Stubs, Spies

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

# Episode VII: 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 VII: Mocks, Stubs, Spies

```
def 'something terrifying'() {
    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 VIII: Time travel

# Episode VIII: 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 VIII: 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 VIII: Time travel

The diagram illustrates two identical Java code snippets for testing infinite loops, connected by dashed lines.

```
@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("testng forever!"); }
}
```

# Episode VIII: 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 VIII: 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
    }
}
```



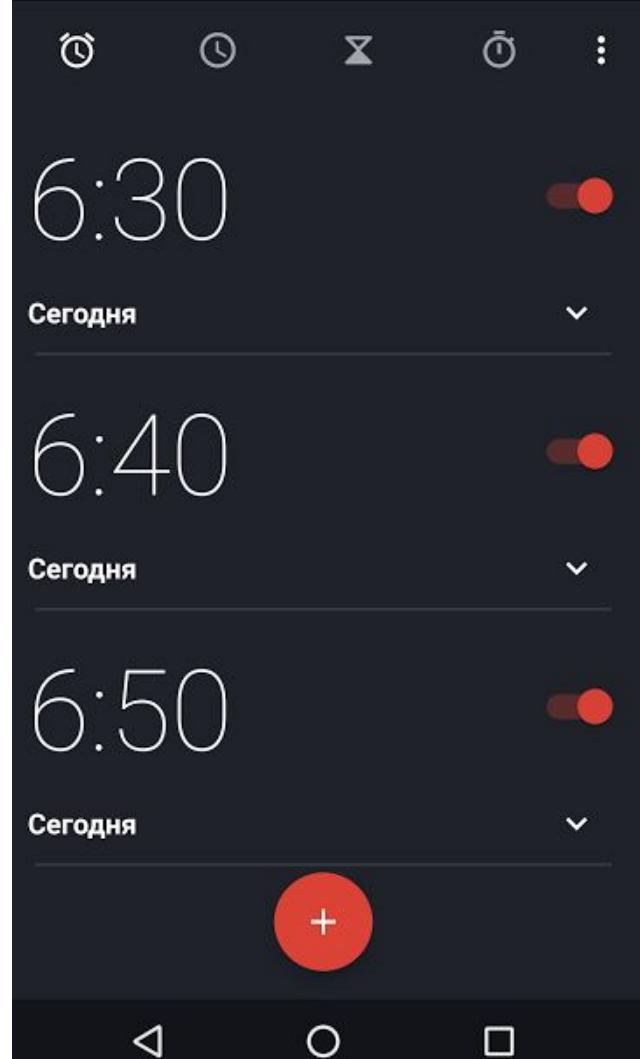
# Episode VIII: Time travel

```
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
    }
}
```

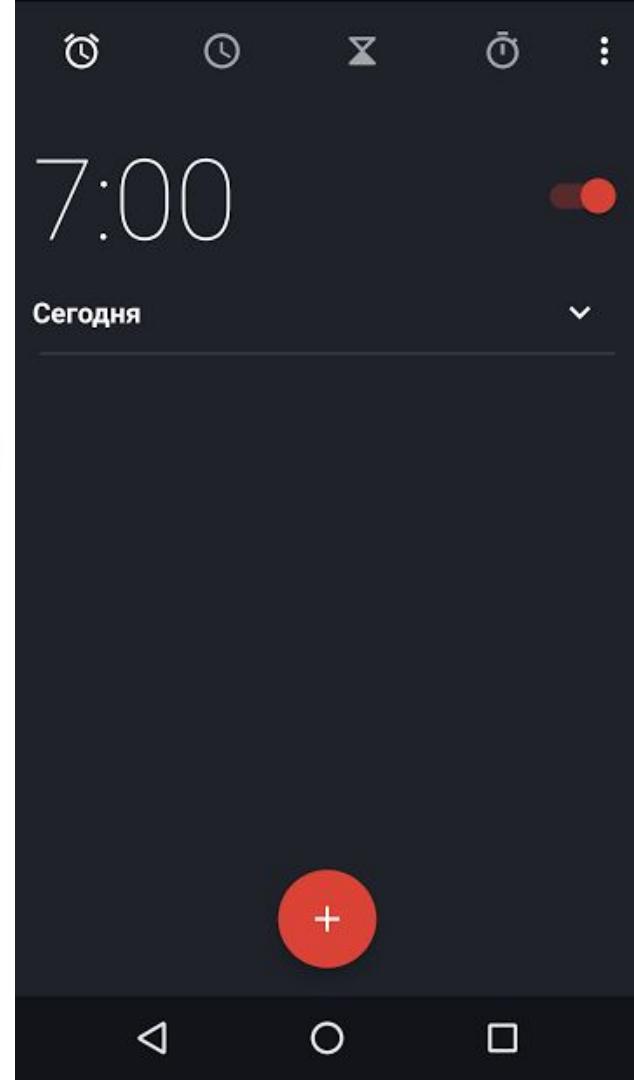


# Episode VIII: Time travel

```
def conditions = new AsyncConditions()

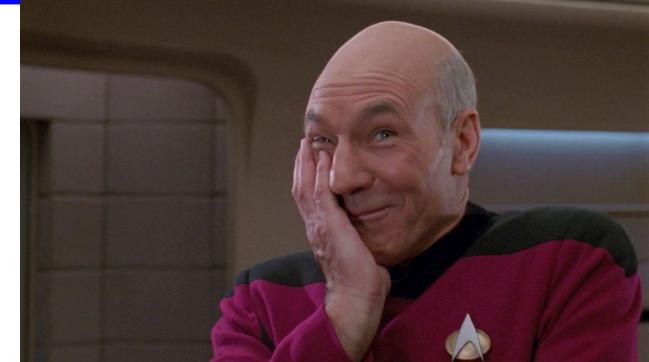
def 'find next prime after 2817 within 3 seconds'() {
    when: 'async start calculating next prime after 1728'
    asyncNextPrime( 1728, { answer ->
        conditions.evaluate { assert answer == 1733 }
    } )
}

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



# Episode VIII: Time travel

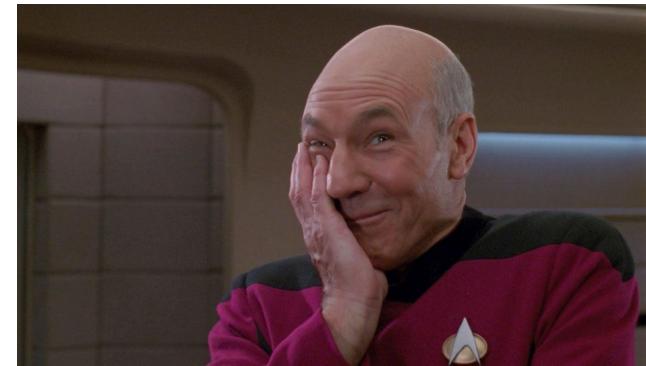
```
@Test  
void iWantToBelieve() {  
    // setup  
    Integer actual = null  
  
    // run  
    asyncNextPrime( 1728, { answer -> actual = answer } )  
  
    // verify  
    assert actual == 1733  
}
```



# Episode VIII: Time travel

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

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

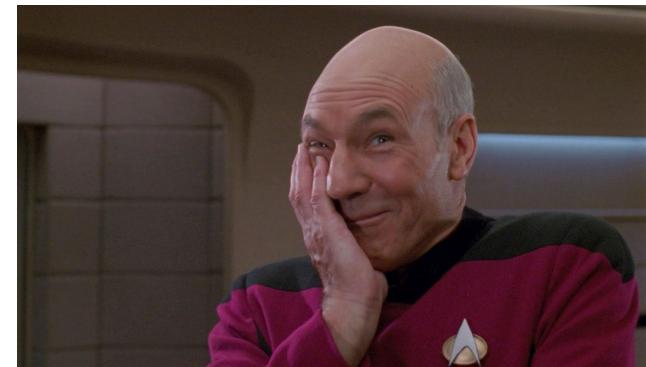


# Episode VIII: Time travel

```
@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
    }
}
```



A scene from Star Trek: The Next Generation showing the bridge crew in the control room. Captain Jean-Luc Picard is seated at the helm. Lieutenant Commander Geordi La Forge is seated in the center seat. Lieutenant Commander Data is seated in the seat behind the captain. Other crew members are standing or seated at various stations. The room is filled with glowing blue and green computer screens displaying complex data.

# Episode IX: Test control

# Episode IX: 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);  
    }  
    IXME test is failing  
    als(2+2, 5);  
}
```

# Episode IX: 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 IX: Test control

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

# Episode IX: 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 IX: 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 IX: 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 IX: 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 IX: Test control

```
@IgnoreRest
```

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



# Episode X: Encore

# Episode X: 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'

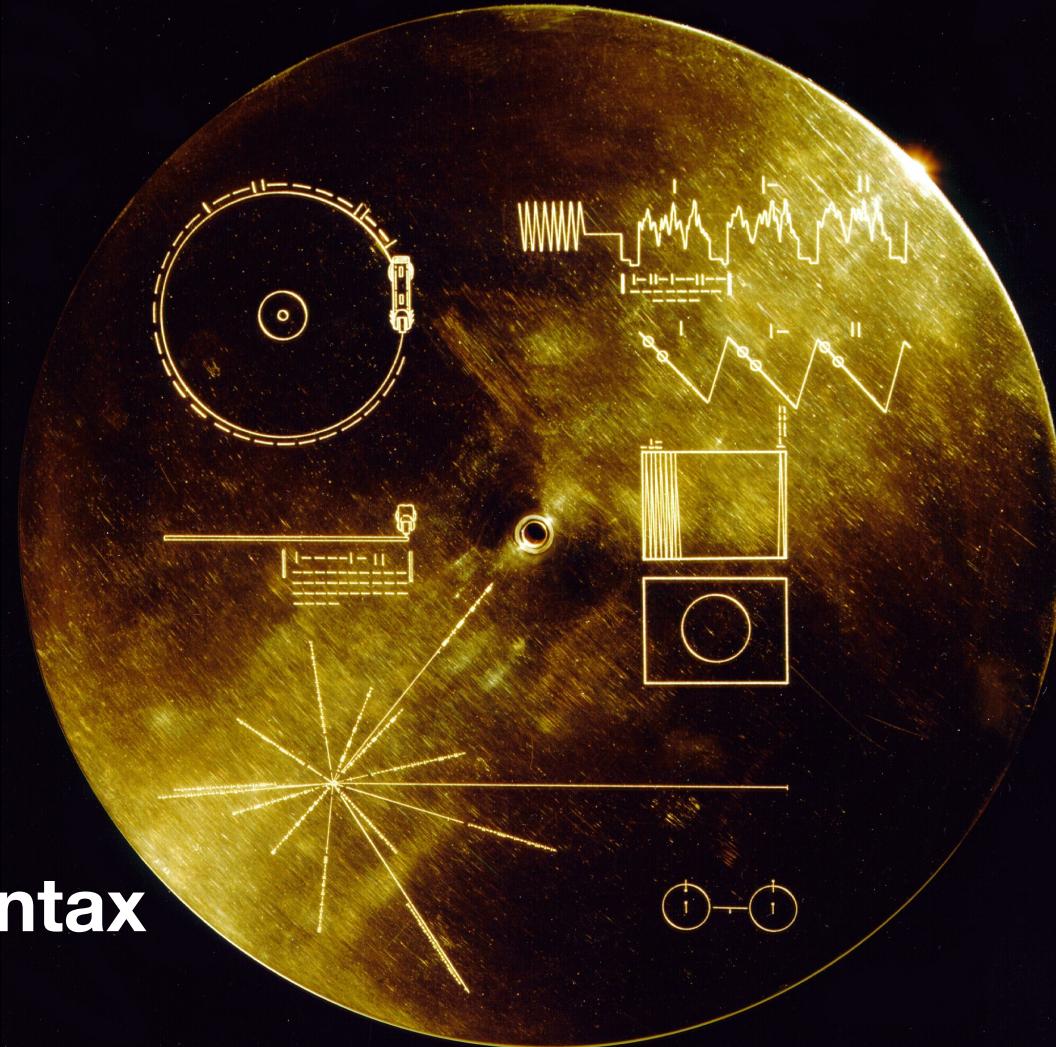
    }

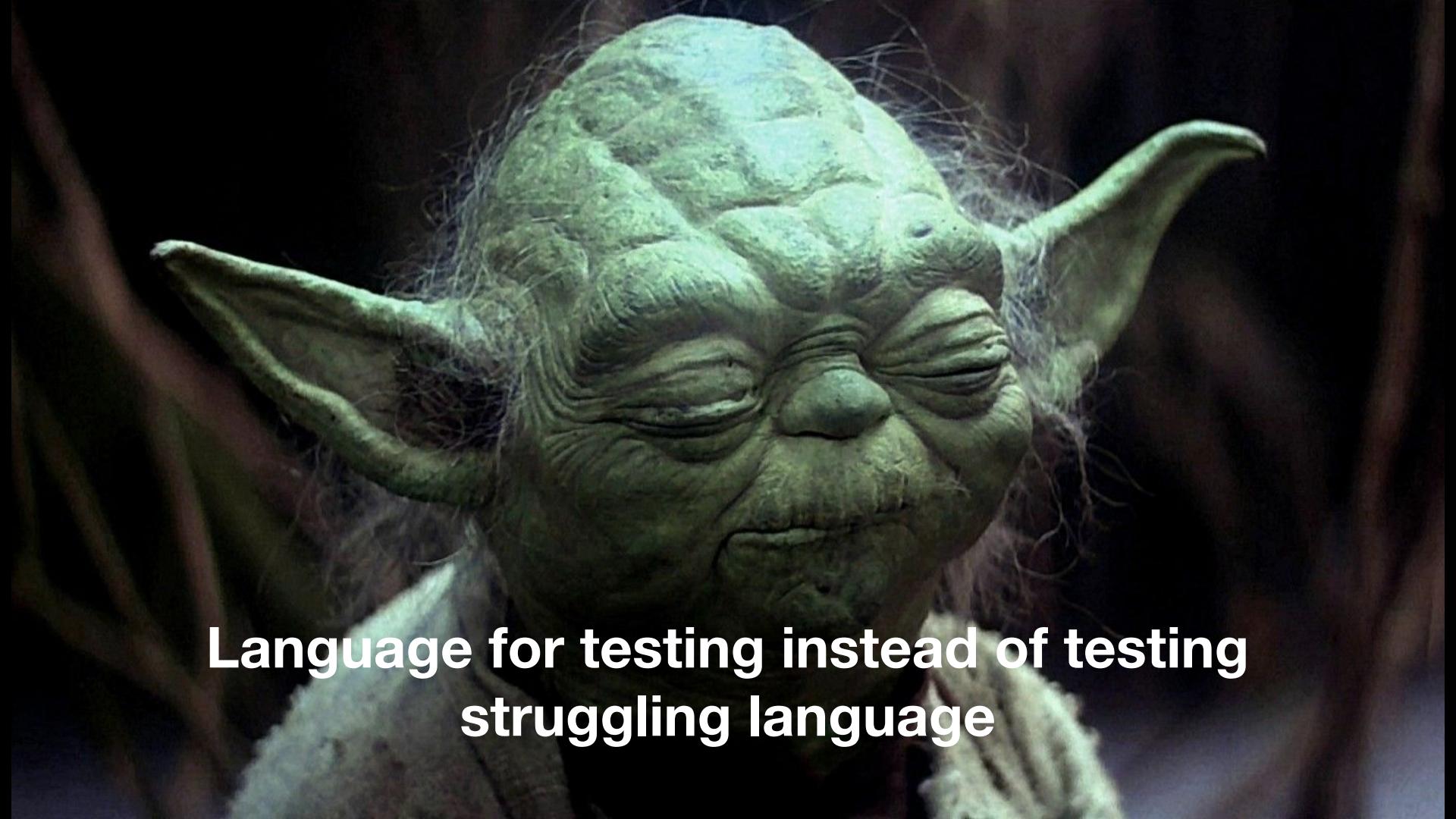
    def 'check spock version is not set'() {
        expect: 'that spock.version is not set'
        System.getProperty('spock.version') == null
    }
}
```

# Why one should use Spock?



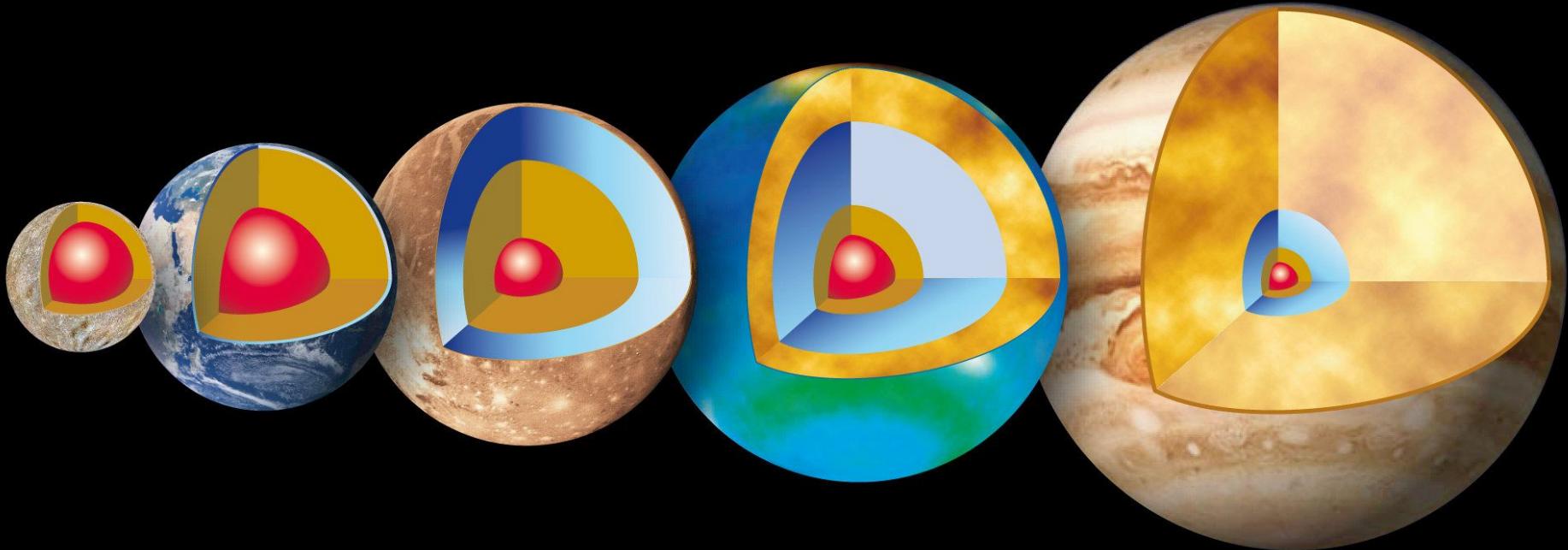
# Concise syntax





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

# Clear test structure





Code and reports readable for everyone

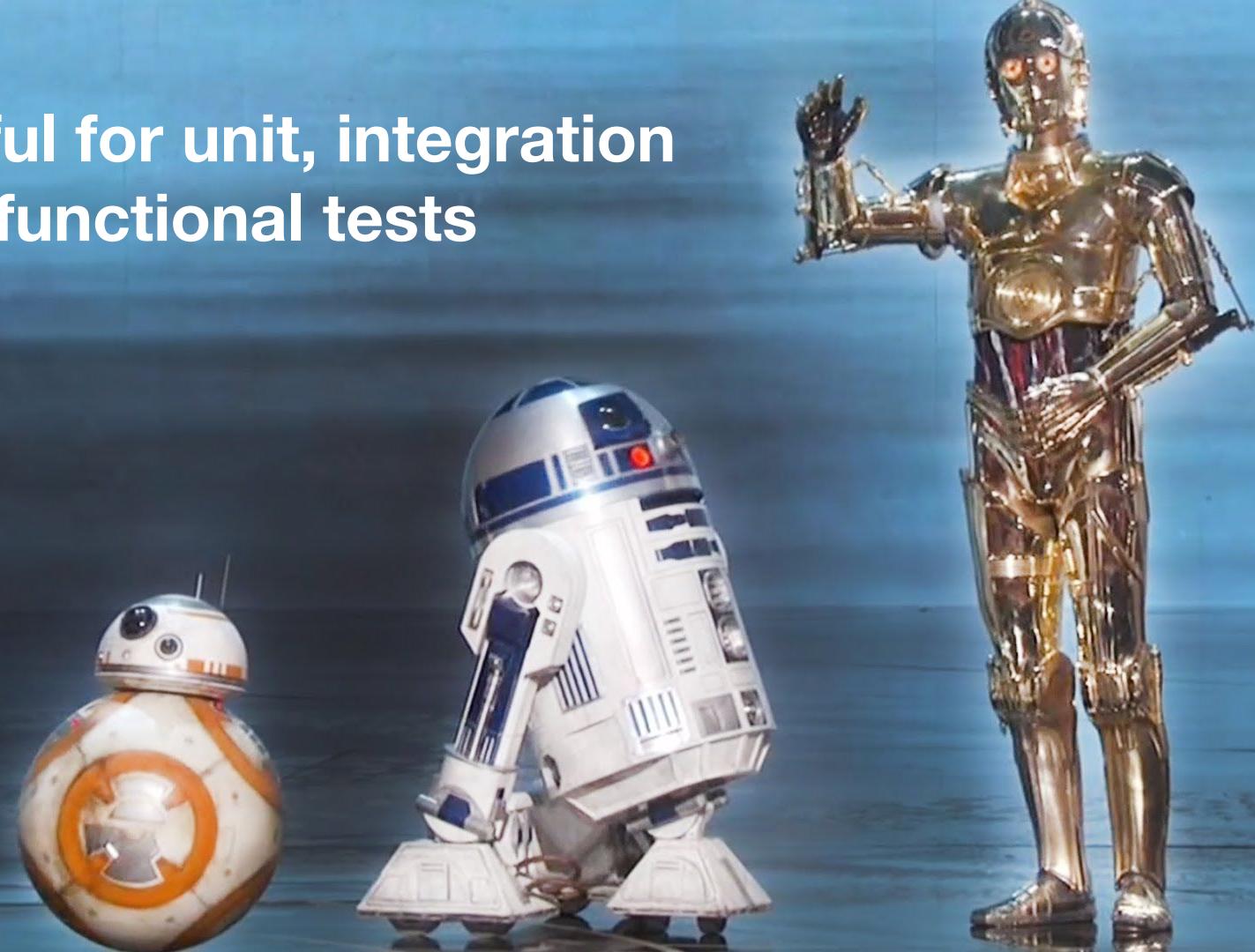
# Powerful built-in features



# Extensions



**Useful for unit, integration  
and functional tests**



# Behaves as one more JUnit runner



# Works for both Groovy and Java





**Why one should not use Spock?**

# JUnit/TestNG has 90% similar/imitable features



# Spock lost its lead developer

spockframework / spock

Watch 111 Star 1,057 Fork 194

Code Issues 163 Pull requests 37 Pulse Graphs

Contributors Commits Code frequency Punch card Network Members

Feb 15, 2009 – May 10, 2016

Contributions to master, excluding merge commits

**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)

# 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?

<http://spockframework.github.io/spock/docs/1.0/index.html>

<http://meetspock.appspot.com/>

<https://github.com/spockframework>

**Just use it in your projects and enjoy!**

A portrait of Mr. Spock from Star Trek. He is wearing his iconic blue Starfleet uniform with gold piping on the collar and cuffs, and a gold Starfleet insignia on his left chest. He has his characteristic Vulcan ponytail hairstyle and is making his famous Vulcan salute with his right hand. His expression is neutral to slightly stern.

Thanks!