JUnit

State of the Union

Görge Albrecht

Software Developer since 1989

Code Mentor - taking care of code

Need help writing simpler code? Contact me!

http://code-mentor.com

@g_o_rge

The Beginnings

SUnit ⇒ JUnit

- SUnit written by Kent Beck around 1992
- JUnit created by Kent Beck and Erich Gamma around 1997
- JUnit became de facto standard of testing Java code

> *Never in the field of software development was so much owed by so many to so few lines of code.*
>
> — Martin Fowler

JUnit 4

- steady progress up to JUnit 3.8.x
- big architectural change from framework style (subclassing) to DSL Style (annotations)
- JUnit 4.4 added dependency to Hamcrest
- JUnit 4.7 added Rules to setting the context of a test

JUnit ƛ

- 10 Years since release of Junit 4
- Java 8 introduced Lambdas
  - not supported nor utilized by JUnit
- JUnit has always been a pet project
- ⇒ Crowdfunding Campaign on Indiegogo

JUnit 5

- development started in Autumn 2015
- alpha release 2016-02-01
- three milestones in 2016
- M4 planned for 2017-04-01
- JUnit 5.0 GA expected Q3 2017

Vision

- Decouple test execution and reporting from test definition and provisioning
- Rethinking JUnit's extensibility story
- Making use of Java 8 features

Lessons learned from JUnit 4

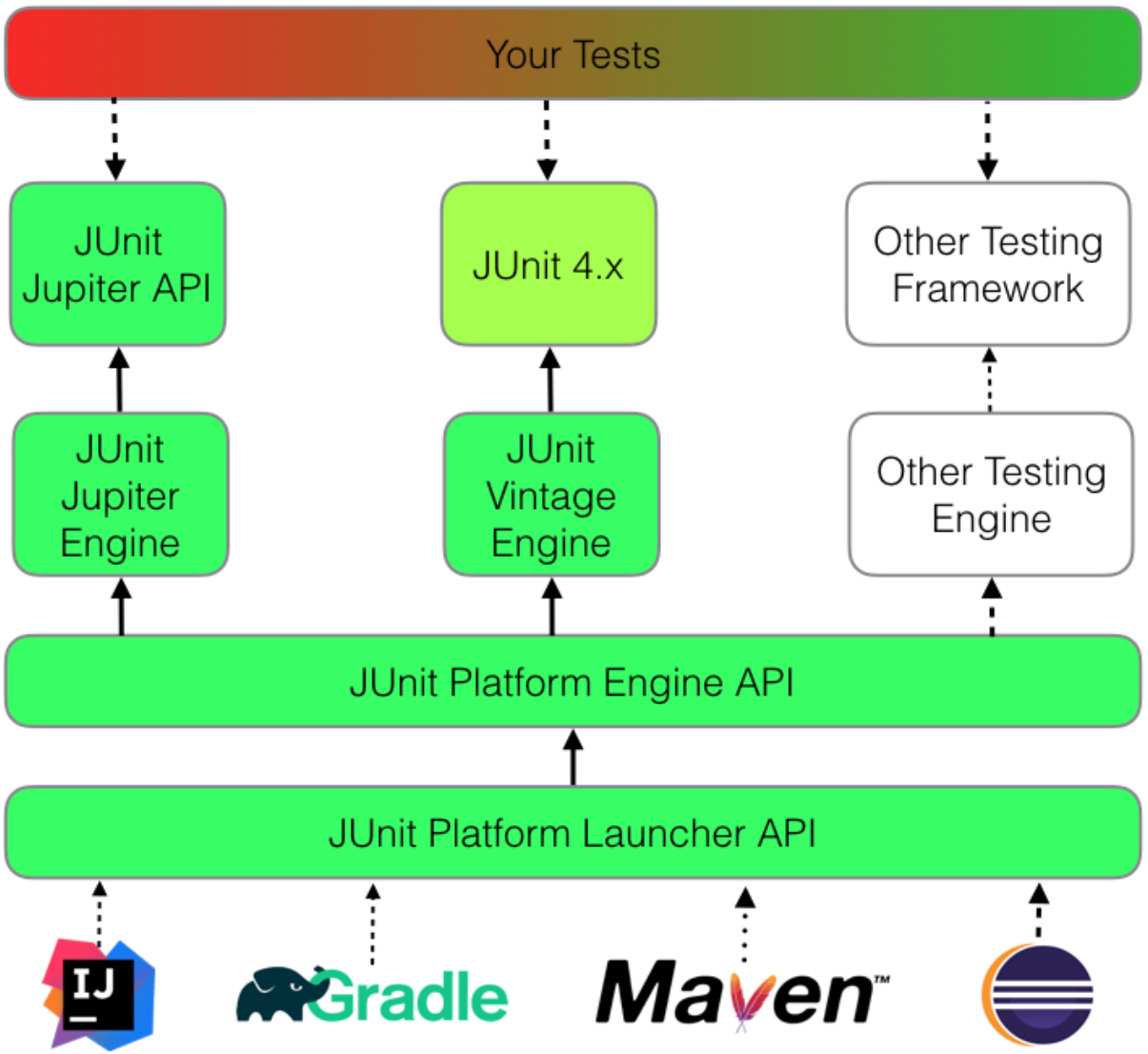
"If you release it, you're going to maintain it."

- `Parameterized`
- Hamcrest

JUnit 5 Core Principles

- Prefer extension points over features
  - an extension point should be good at one thing
- It should be hard to write tests that behave differently based on how they are run
- Tests should be easy to understand
- Minimize dependencies (especially third-party)

Components


Your Tests
JUnit Jupiter API
JUnit 4.x
Other Testing Framework
JUnit Jupiter Engine
JUnit Vintage Engine
Other Testing Engine
JUnit Platform Engine API
JUnit Platform Launcher API
IJ
Gradle
Maven

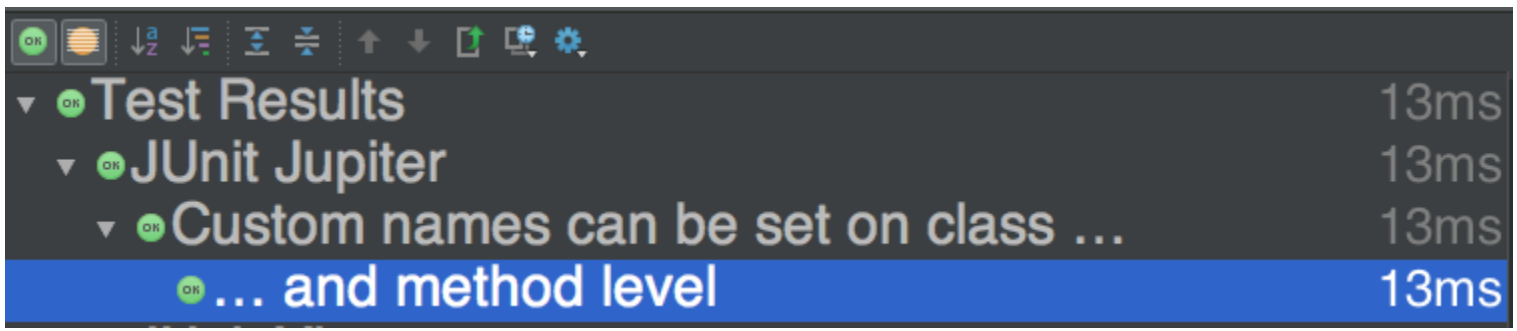
Syntactic sugar

no `public` anymore

```
package com.codementor.junit5;

import org.junit.jupiter.api.Test;

class NoPublicTest {

  @Test
  void name() {
  }
}
```

Display Names

```
@DisplayName("Custom names can be set on class …")
class DisplayNamesTest {

  @Test
  @DisplayName("… and method level")
  void name() {
  }
}
```



Contract Tests I

```
class ContractTest implements EqualsTester<String> {

  @Override
  public String value() {
    return "one";
  }

  @Override
  public String equalValue() {
    return "one";
  }

  @Override
  public String nonEqualValue() {
    return "two";
  }
}
```

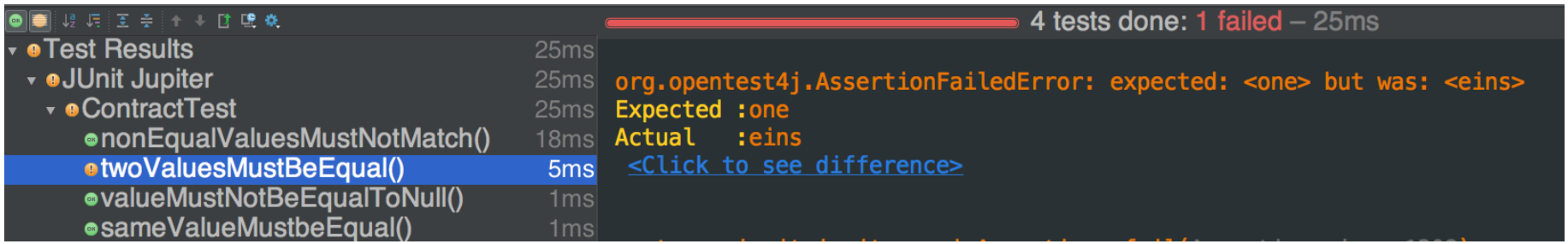
Contract Tests II

```
interface EqualsTester<T> {

  T value();

  T equalValue();

  T nonEqualValue();

  @Test
  default void twoValuesMustBeEqual() {
    assertEquals(value(), equalValue());
  }

  @Test
  default void nonEqualValuesMustNotMatch() {
    assertNotEquals(value(), nonEqualValue());
  }
```

Contract Tests III

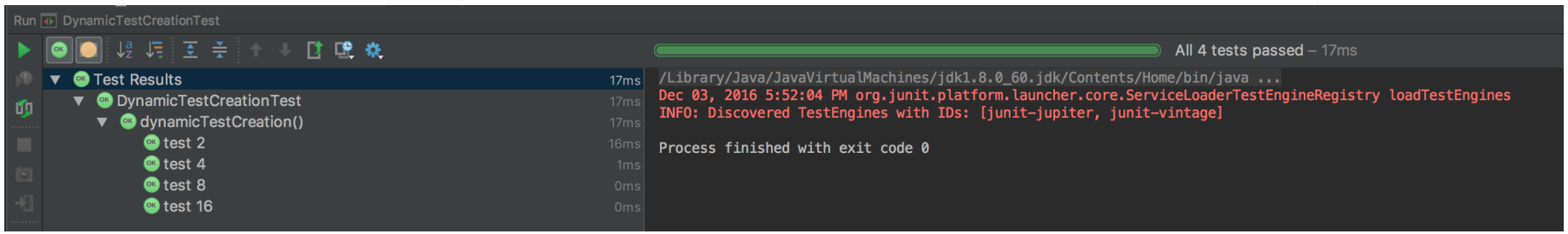
```
  @Test
  default void valueMustNotBeEqualToNull() {
    assertNotEquals(value(), null);
  }

  @Test
  default void sameValueMustbeEqual() {
    assertEquals(value(), value());
  }
}
```

Contract Tests IV



Dynamic Test Creation

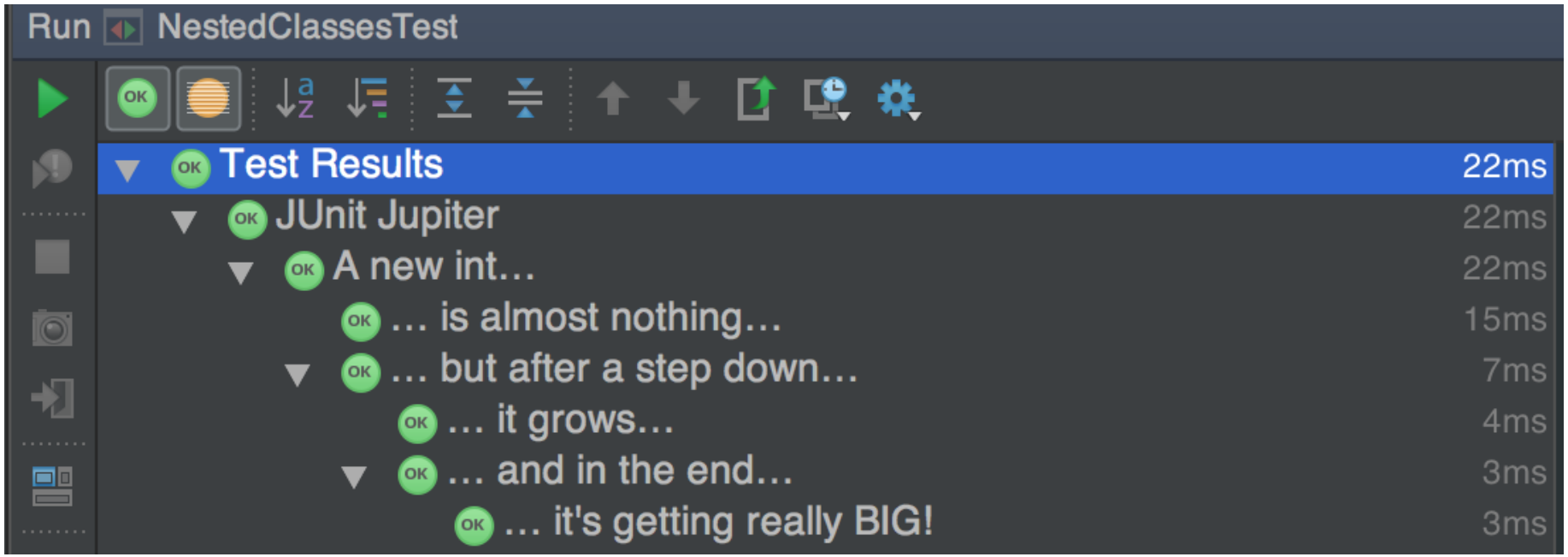
```
@TestFactory
Stream<DynamicTest> dynamicTestCreation() {
  return IntStream.iterate(2, n -> n * 2).limit(4).mapToObj(
    n -> dynamicTest("test " + n, () -> assertTrue(n % 2 == 0))
  );
}
```



Nested Test

```
@DisplayName("A new int…")
class NestedClassesTest {

  private int count = MIN_VALUE;

  @BeforeEach
  void setCountToZero() {
    count = 0;
  }

  @Test
  @DisplayName("… is almost nothing…")
  void countIsZero() {
    assertEquals(0, count);
  }

  @Nested
  @DisplayName("… but after a step down…")
```

Nested Test Output



Assertions

Message comes last, finally!

```
@Test
void messageComesLast() {
  org.junit.Assert.assertEquals(
    "JUnit <=4 message comes first", "aString", "aString");

  org.junit.jupiter.api.Assertions.assertEquals(
    "aString", "aString", "JUnit 5 message comes last");
}
```

Lambdas FTW

```
@Test
void assertWithBooleanClassic() {
  assertTrue(true);
}

@Test
void assertWithBooleanLambda() {
  assertTrue(() -> true);
}

@Test
void assertWithBooleanMethodReference() {
  assertTrue(TRUE::booleanValue);
}
```

Exceptions

```
@Test
void exception() {
  assertThrows(RuntimeException.class, this::illegal);
}

private void illegal() {
  throw new IllegalArgumentException("illegal");
}
```

Exception Message Assertion

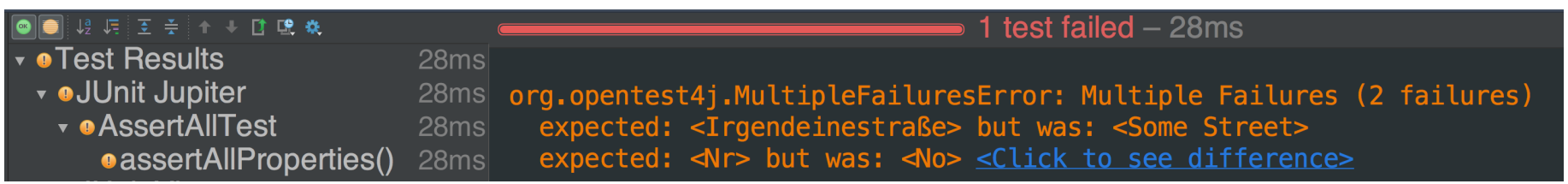
```
@Test
void exceptionMessageAssertion() {
  final RuntimeException exception =
    assertThrows(RuntimeException.class, this::illegal);
  assertEquals("illegal", exception.getMessage());
}

private void illegal() {
  throw new IllegalArgumentException("illegal");
}
```

Lazy message evaluation

```
@Test
void assertWithLambdaMessage() {
  assertTrue(
    true, () -> "JUnit 5" + " message are evaluated " + "lazily");
}
```

Assert all properties at once

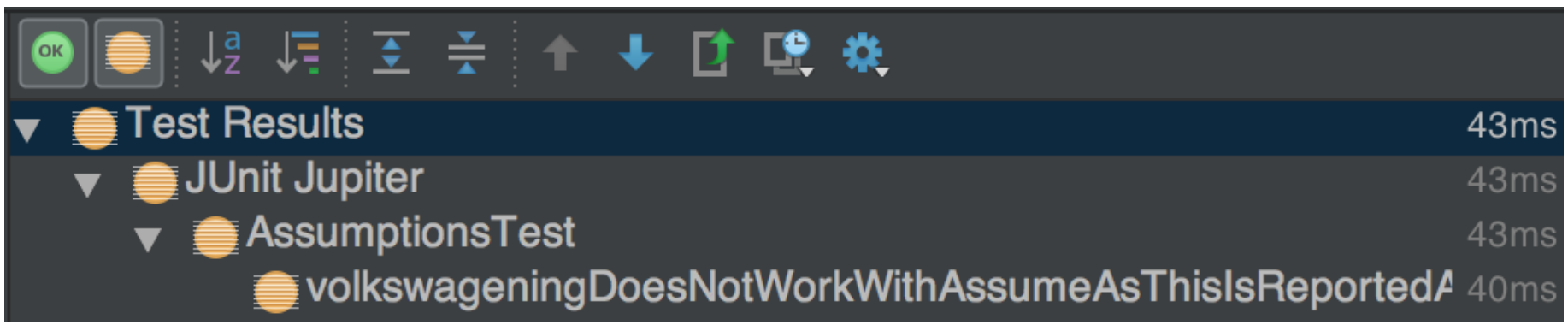
```java
@Test
void assertAllProperties() {
  Address address = new Address("New City", "Some Street", "No");

  assertAll(
    () -> assertEquals("New City", address.city),
    () -> assertEquals("Irgendeinestraße", address.street),
    () -> assertEquals("Nr", address.number)
  );
}
```



Assumptions

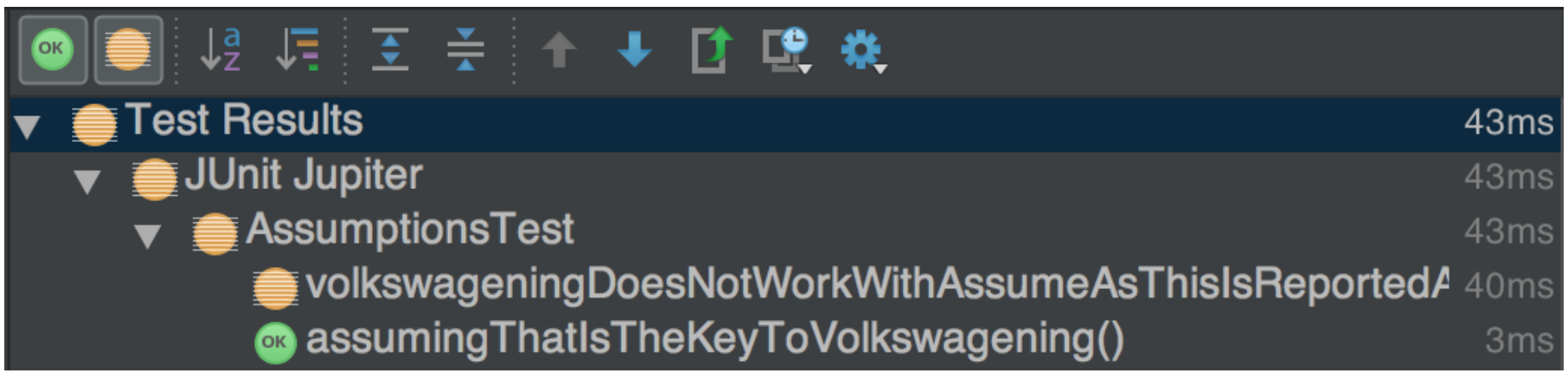
Volkswagening

```
@Test
void volkswageningDoesNotWorkWithAssumeAsThisIsReportedAsAborted() {
  assumeTrue(notOnTestStation());
  assertTrue(abgasWerteUnterVorgabe());
}
```



real Volkswagening

```
@Test
void assumingThatIsTheKeyToVolkswagening() {
  assumingThat(notOnTestStation(),
    () -> assertTrue(abgasWerteUnterVorgabe()));
}
```



Extensions & Dependency Injection

Extensions

- no `Runner` and `@Rule` / `@ClassRule` anymore
- replaced by Extensions
- registered via `@ExtendWith` on class, method or annotation
- multiple extensions per element possible
- registered extensions are inherited

Extensions example

```
@ExtendWith(MockitoExtension.class)
@ExtendWith(SpringExtension.class)
class MyTest {

  @Test void test1() {}

  @ExtendWith(TimingExtension.class)
  @Test void test2() {}
}
```

Extension Types

- Conditional Test Execution
  - `ContainerExecutionCondition`
  - `TestExecutionCondition`
- Test Instance Post-processing
  - `TestInstancePostProcessor` (`MockitoExtension`, `SpringExtension`)
- Parameter Resolution
  - `ParameterResolver`

Test Lifecycle Callbacks

- `BeforeAllCallback`
  - `BeforeEachCallback`
    - `BeforeTestExecutionCallback`
    - `AfterTestExecutionCallback`
  - `AfterEachCallback`
- `AfterAllCallback`

↯ `TestExecutionExceptionHandler`

Dependency Injection

- JUnit 4.x: DI via Runner setting method parameters
- JUnit 4.7: DI via `@Rule` settings fields in test class
- JUnit 5: `ParameterResolver` FTW
  - multiple `ParameterResolver` per test method possible
  - each `ParameterResolver` can handle different parameter types

DI example

```
class MyTest {

  @ExtendWith(MockitoExtension.class)
  @ExtendWith(VertxExtension.class)
  @Test void test(@Mock MyService service, TestContext context) {}
}
```

Built-in Parameter Resolver

- TestInfoParameterResolver
  - provides test with context (name, tags, etc.) via `TestInfo` parameter
  - works also on lifecycle callbacks
- `TestReporterParameterResolver`
  - injects `TestReporter` into test
  - test can provide runtime information via `TestReporter` to execution env

Migration from JUnit 4

Remarks

- JUnit 5 is a complete rewrite
- neither compile nor binary compatibility
- bridges provided down- and upwards for smooth migration
- limited JUnit 4 rule support
- ⇒ no need to update all existing tests

Execution

- run JUnit 5 based tests with JUnit 4 ?
  - JUnit 4 Runner: `JUnitPlatform`
- run JUnit 4 based tests with JUnit 5 ?
  - JUnit Platform engine: `junit-vintage-engine`

Packages

JUnit < 4	`junit`
JUnit 4.x	`org.junit`
JUnit 5	`org.junit.jupiter.api`

Assertions & Assumptions

JUnit 5	JUnit 4.x
`o.j.j.a.Assertions`	`o.j.Assert`
`o.j.j.a.Assumptions`	`o.j.Assume`

Annotations

JUnit 5	JUnit 4.x
@Test (new package)	@Test
@TestFactory	-
@Disabled	@Ignore
@DisplayName	-
@Tag	@Category

Lifecycle Annotations

JUnit 5	JUnit 4.x
@BeforeEach	@Before
@AfterEach	@After
@BeforeAll	@BeforeClass
@AfterAll	@AfterClass
@ExtendWith	~ @RunWith, @Rule, @ClassRule

Switch to JUnit 5 ?

well ...

- no big improvements regarding Assertions API
  - use AssertJ or Hamcrest
- IDEs not there yet
  - eclipse: beta branch
  - IntellIJ: steady progress released
  - use maven/gradle in the meantime

but ...

- new extension points will bring big opportunities 👍
- launcher and engine API
  - enables IDEs execute tests independent of the actual testing framework 👍

Start today!

- analyze your code base and identify open ends
- convert your `Runner`/`Rule` to Extension
- ask your `Runner`/`Rule` provider to do so
- test your favorite IDE and file bugs
- give feedback to JUnit 5 team

Thanks for listening

@g_o_rge · http://code-mentor.com