JUnit

Writing tests with Java 8 and JUnit

Peter Kofler

Professional Software Developer for 15+ years

“fanatic about code quality”

I help development teams!

@codecopkofler · code-cop.org

Görge Albrecht

Software Developer since 1989

Code Mentor - taking care of code

Need help writing simpler code? Contact me!

@g_o_rge · code-mentor.com

The Beginnings

SUnit ⇒ JUnit

- SUnit written by Kent Beck around 1992
- JUnit created by Kent Beck and Erich Gamma around 1997
- JUnit 4 - from framework style to DSL Style
- JUnit 4.4 added dependency to Hamcrest
- JUnit 4.7 added Rules to setting the context of a test

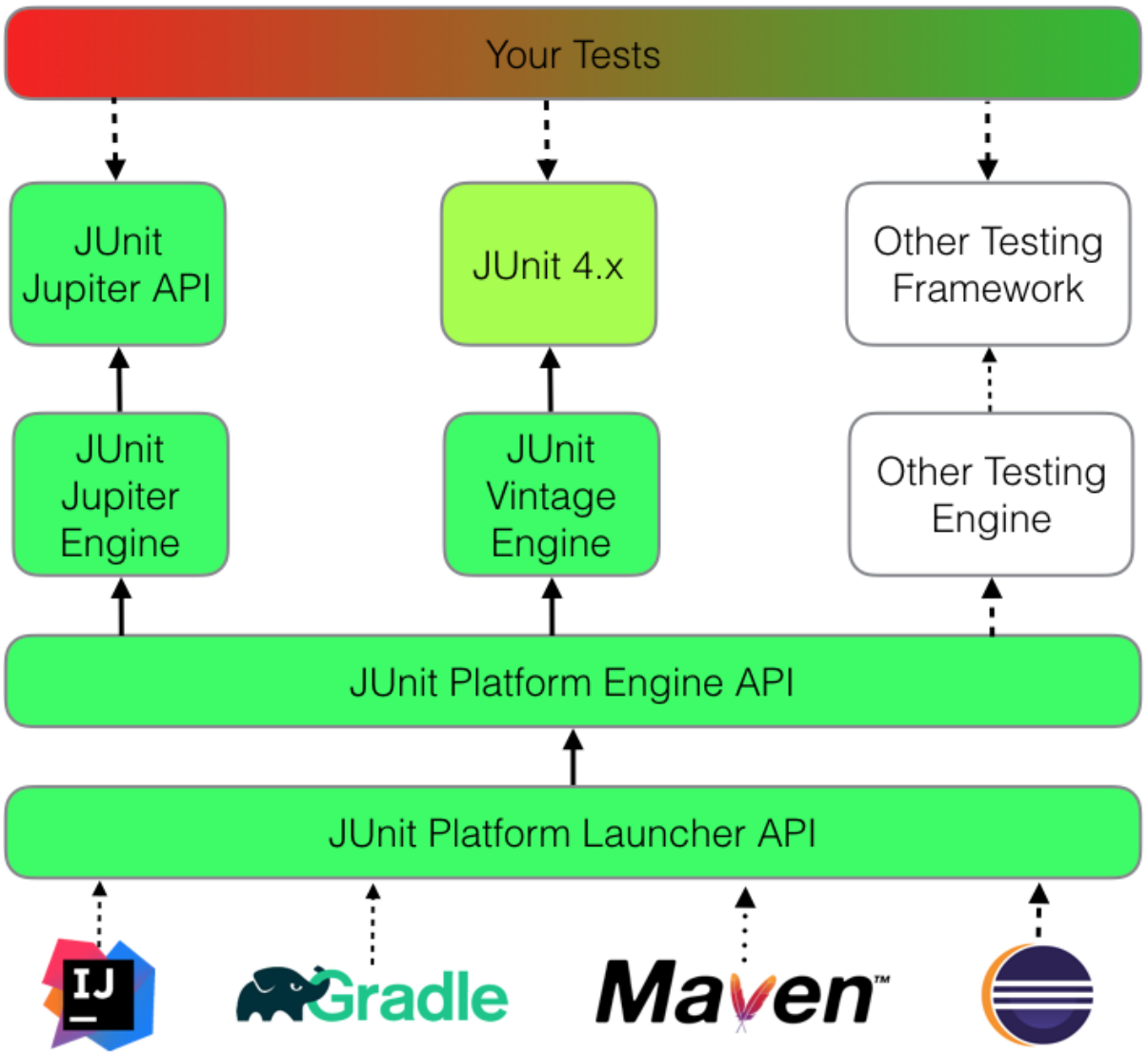
JUnit 5

- 10 Years since release of Junit 4
- JUnit has always been a side project
- ⇒ Crowdfunding Campaign on Indiegogo
- M3 released on 2016-11-30
- JUnit team hopes to release JUnit 5.0 GA during Q1 2017

Vision

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

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

JUnit 5 is ...

- JUnit Platform (Engine API, Launchers)
- JUnit Jupiter API (Test API)
- JUnit Jupiter Engine (the test engine)
- JUnit Vintage Engine (running JUnit 4)
- >= 3 different dependencies.

build.gradle

```
buildscript {
  dependencies {
    classpath 'org.junit.platform:junit-platform-gradle-plugin:1.0.0-M3'
}}

apply plugin: 'org.junit.platform.gradle.plugin'

junitPlatform {
  platformVersion = "1.0.0-M3"
  filters {
    engines {
      include 'junit-jupiter'
}}}

dependencies {
  testCompile "org.junit.jupiter:junit-jupiter-api:5.0.0-M3"
  testRuntime "org.junit.jupiter:junit-jupiter-engine:5.0.0-M3"
}
```

Features of JUnit 5

Annotations

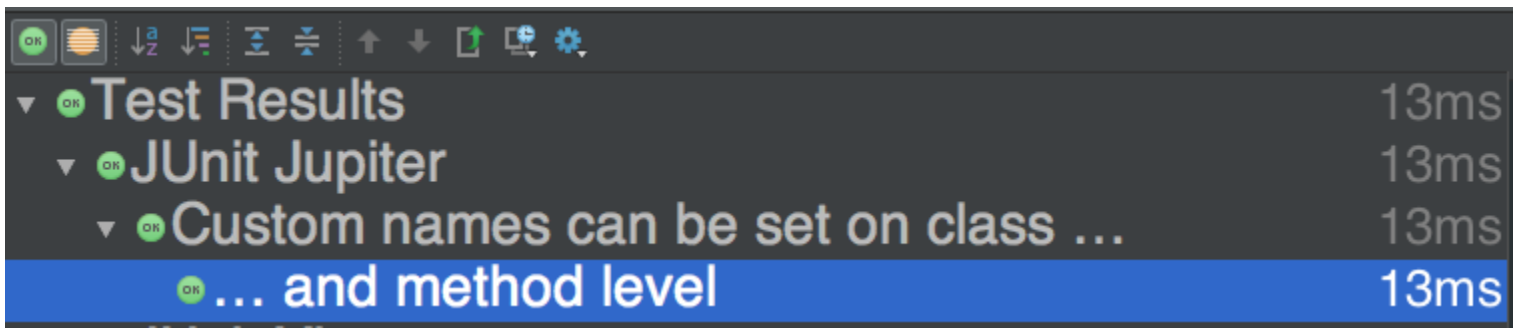
- `@org.junit.jupiter.api.Test`
- `@BeforeEach` / `@AfterEach`
- `@BeforeAll` / `@AfterAll`
- `@Disabled`

no `public` anymore

```
package com.codementor.junit5;

import org.junit.jupiter.api.Test;

class NoPublicTest {

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

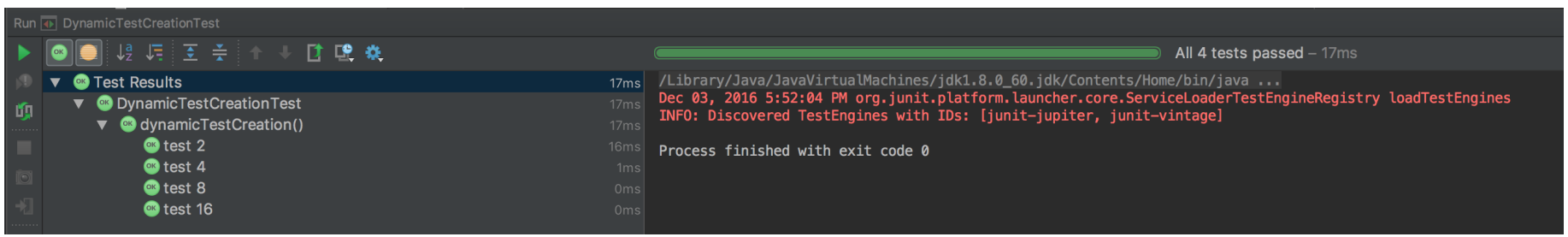
Display Names

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

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



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))
  );
}
```

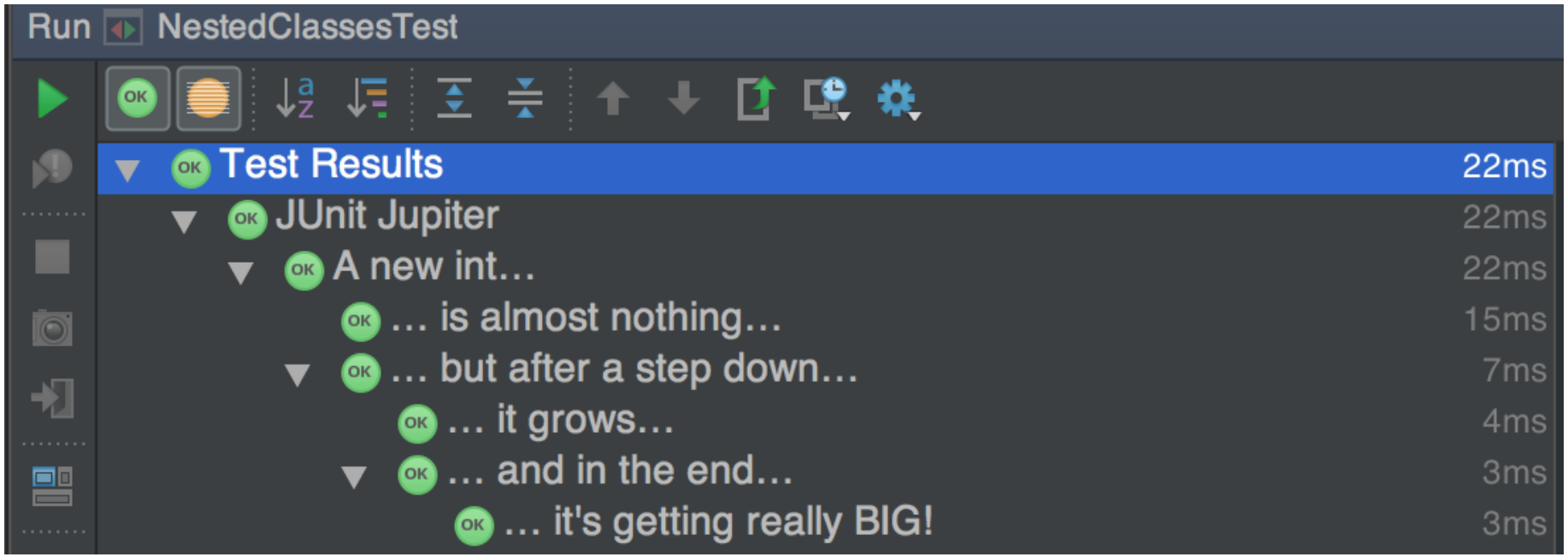


IntelliJ support since version 2016.3

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



Assertion 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);
}
```

Asserting Exceptions

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

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

Asserting Exception Message

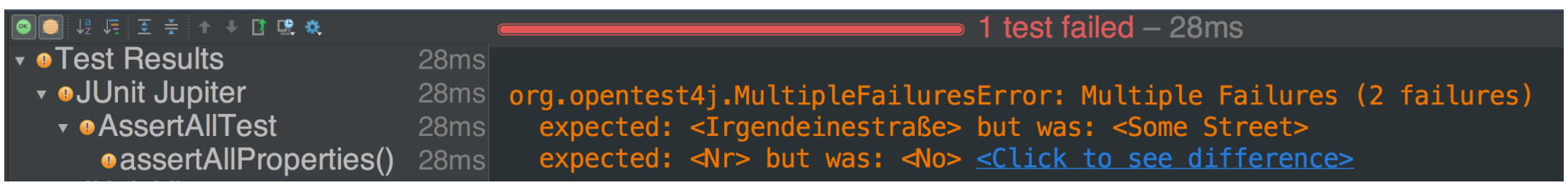
```
@Test
void exceptionMessageAsssertion() {
  final RuntimeException exception =
    assertThrows(RuntimeException.class, this::illegal);
  assertEquals("illegal", exception.getMessage());
}

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)
  );
}
```



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

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) {}
}
```

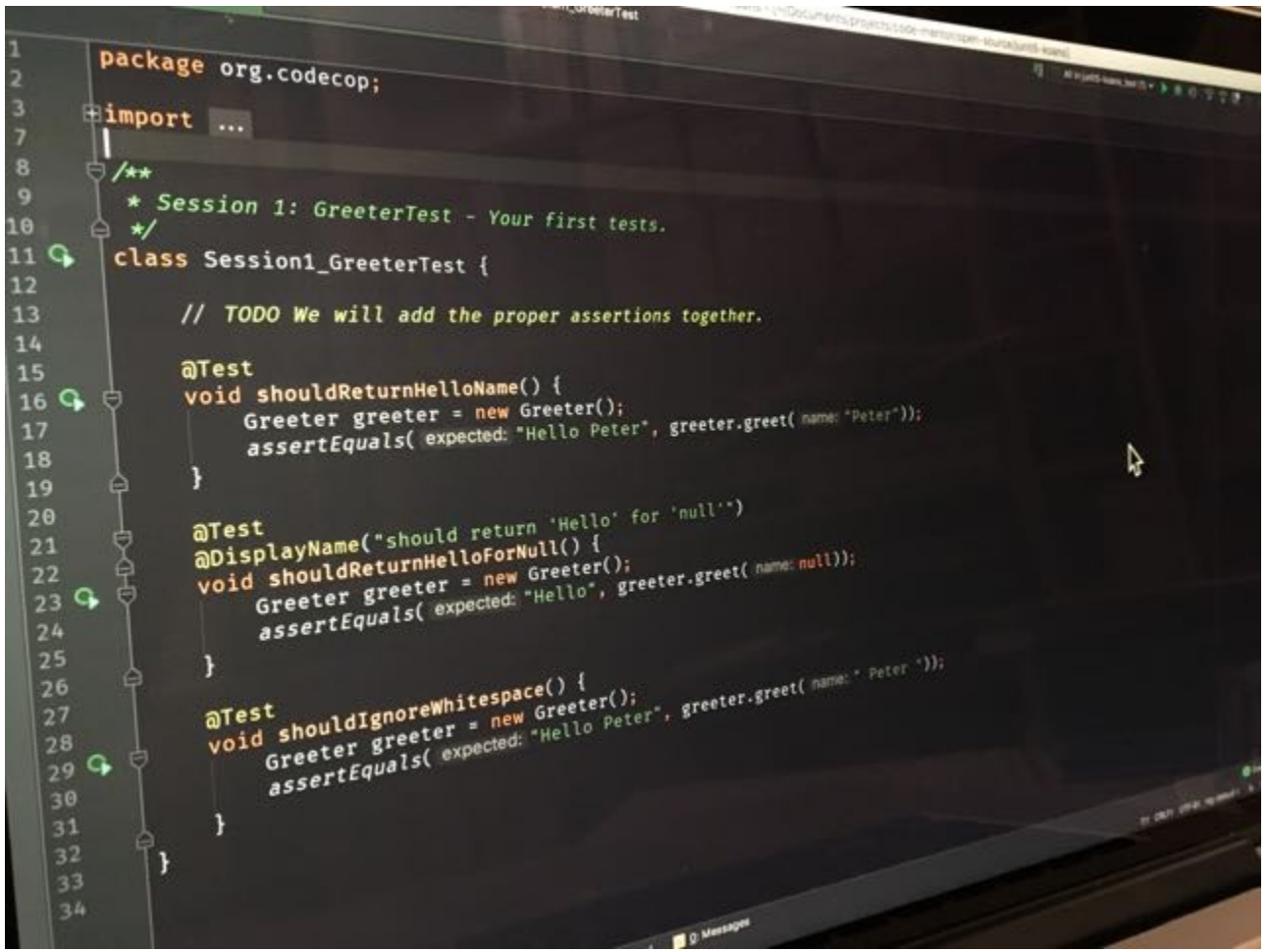
Try yourself

www.flickr.com/photos/ninahiironniemi/497993647

Assignment

- Find a pair.
- Get the code (`junit5-koans.zip`)
- Run JUnit. (Maven: `mvnw test`; Gradle: `gradlew test`)
- Go through the test code.
- Assertions are missing/incomplete.
- Complete assertions, make tests pass.

Coding



```
package org.codecop;

import ...

/**
 * Session 1: GreeterTest - Your first tests.
 */
class Session1_GreeterTest {

    // TODO We will add the proper assertions together.

    @Test
    void shouldReturnHelloName() {
        Greeter greeter = new Greeter();
        assertEquals( expected: "Hello Peter", greeter.greet( name: "Peter"));
    }

    @Test
    @DisplayName("should return 'Hello' for 'null'")
    void shouldReturnHelloForNull() {
        Greeter greeter = new Greeter();
        assertEquals( expected: "Hello", greeter.greet( name: null));
    }

    @Test
    void shouldIgnoreWhitespace() {
        Greeter greeter = new Greeter();
        assertEquals( expected: "Hello Peter", greeter.greet( name: " Peter "));
    }
}
```

Sessions

Hints and solutions.

Session 1

Your first tests.

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

assertEquals();
```

Session 2a

Basic assertions.

```
import static org.junit.jupiter.api.Assertions.*;

assertTrue();
assertFalse();
assertNull();
assertNotNull();
assertArrayEquals();
assertEquals(., ., 0.01);
```

Session 2b

Assertions new in JUnit 5.

```
import static org.junit.jupiter.api.Assertions.*;

assertNotEquals();
assertIterableEquals();

assertAll(
    () -> assertEquals(),
    () -> assertEquals()
);

assertTimeout();
```

Session 2c

Assertions with Hamcrest.

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.collection.IsArrayContaining.hasItemInArray;
import static org.hamcrest.collection.IsArrayWithSize.arrayWithSize;
import static org.hamcrest.core.IsNot.not;

assertThat(counter.uniqueWords(), hasItemInArray("bar"));
assertThat(counter.uniqueWords(), not(hasItemInArray("foo")));

assertThat(counter.uniqueWords(), arrayWithSize(3));
```

Session 3

Fixtures.

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

@BeforeEach
void createFreshTestFileForEachTest() throws IOException { }

@AfterEach
void deleteTestFile() { }
```

Session 4

Exceptions and ignoring tests.

```
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.function.Executable;

assertThrows(Exception.class, Executable);
Exception exception = assertThrows(Exception.class, Executable);

@Test
@Disabled("message")
void shouldCountUniqueWordsCaseInsensitive() { }
```

Session 5

Parameterised/table driven tests.

```
import org.junit.jupiter.api.DynamicTest;
import org.junit.jupiter.api.TestFactory;

@TestFactory
List<DynamicTest> createTests() {
    return TEST_CASES.stream().
            map(testCase -> DynamicTest.dynamicTest(
                    testCase.name(),
                    testCase::shouldReturnRatioOfGivenWord)
                ).
            collect(Collectors.toList());
}
```

Session 6

Reuse fixtures in Extensions.

```
import org.junit.jupiter.api.extension.ExtendWith;
import org.codecop.TempFile.Temp;

@ExtendWith(TempFile.class)
class SomeTest {

@Test
void shouldDoSomething(@Temp File file) {
    // use file
}
```

```
import org.junit.jupiter.api.extension.AfterEachCallback;
import org.junit.jupiter.api.extension.BeforeEachCallback;
import org.junit.jupiter.api.extension.ParameterResolver;

class TempFile implements BeforeEachCallback,
                          ParameterResolver,
                          AfterEachCallback {

public void beforeEach(TestExtensionContext context) {
    Object testInstance = context.getTestInstance();
    createTempFileFor(testInstance);
}

public Object resolve(...) {
    return getTempFile();
}

public void afterEach(...) {
```

Migration from JUnit 4

Remarks

- JUnit 5 is a complete rewrite
- neither compile nor binary compatibility
- bridges provided down- and upwards for smooth migration
- ⇒ 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`

Changing Code

- 80% search/replace package and class names
- only partial support for Rules
- fails with parameterized / DynamicTest

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

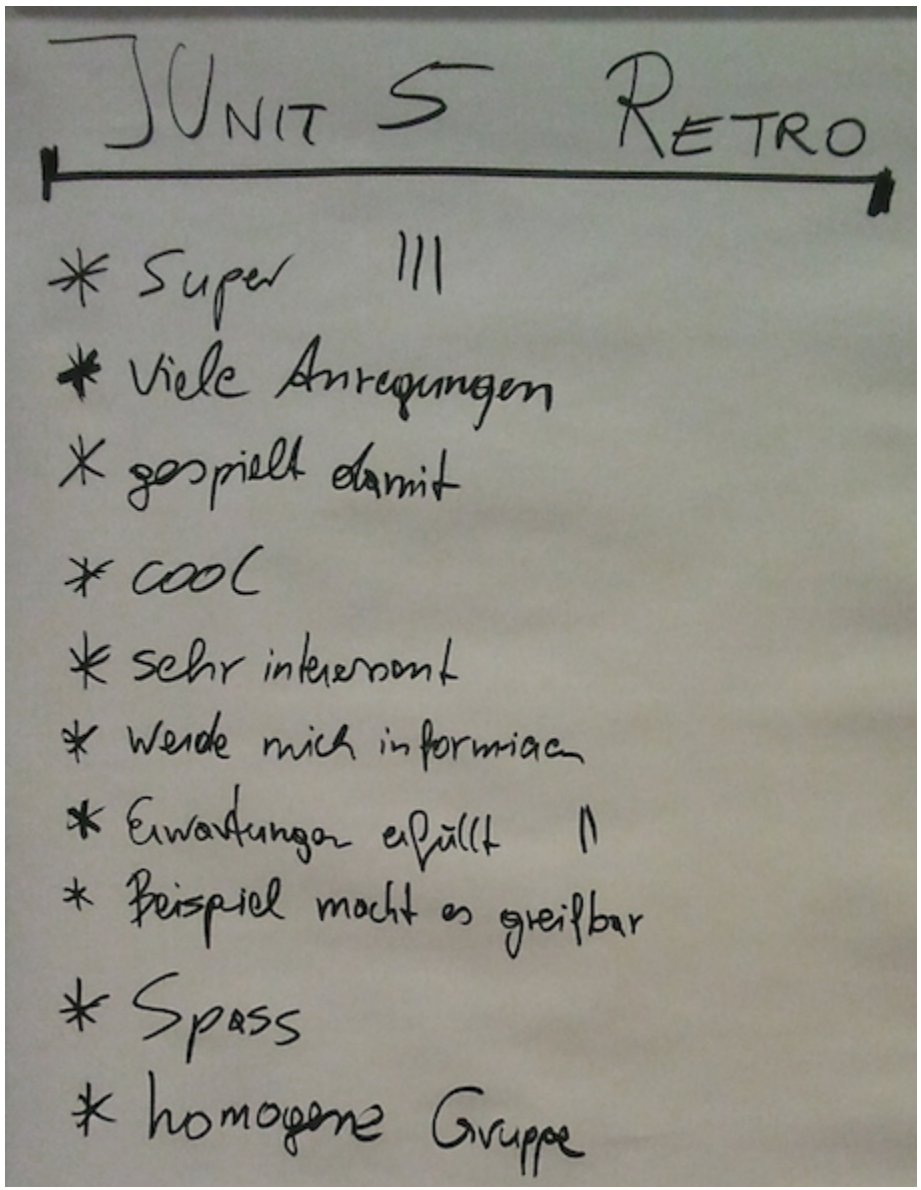
but ...

- better testing for exception 👍
- Dynamic tests are more powerfull 👍
- solves "more Runners problem" 👍
- 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

What did you learn today?



Thanks for participating

@codecopkofler · code-cop.org

@g_o_rge · code-mentor.com