JUNIT 5 & TESTCONTAINERS

Testen in Zeiten von Java 10 und Docker

TIM RIEMER

- Integration Architect @ FNT GmbH
- Co-Lead Kotlin UG Dusseldorf
- @zordan_f
- github.com/timriemer

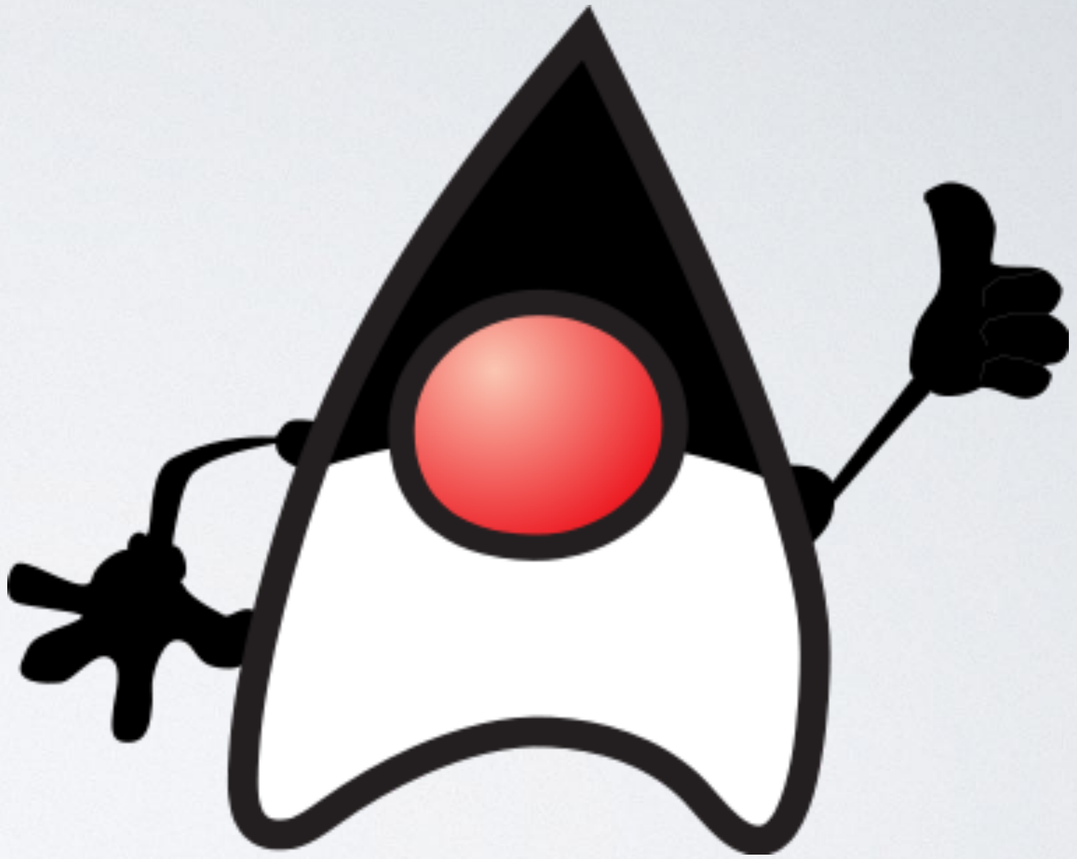
AGENDA

- Java 10
- JUnit 5
- Testcontainers

JAVA 10

What's new?

From a developer's perspective



JAVA 10

- Local-variable type inference (JEP 286) - **var**

JUNIT 5

JUNIT 5

- JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage
- Current version: 5.2.0
- No public modifier needed
- Requires Java 8 or greater
- Support in IntelliJ IDEA and Eclipse IDE

EXAMPLE

```
@Test
public void simpleTest() {
    assertEquals(2, 1+1);
}
```

EXAMPLE

```
// JUnit 4 Imports
import org.junit.Test;

import static org.junit.Assert.assertEquals;
```

EXAMPLE

```
// JUnit 5 Imports
import org.junit.jupiter.api.Test;

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

DEPENDENCIES

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.2.0</version>
    <scope>test</scope>
</dependency>
```

ANNOTATIONS

```
// JUnit 4 @BeforeClass
@BeforeAll
static void tearUp() {
    System.out.println("tearUp!");
}

// JUnit 4 @AfterClass
@AfterAll
static void tearDown() {
    System.out.println("tearDown!");
}
```

ANNOTATIONS

```
// JUnit 4 @Before
@BeforeEach
void before() {
    System.out.println("before!");
}

// JUnit 4 @After
@AfterEach
void after() {
    System.out.println("after!");
}
```

ANNOTATIONS

```
// JUnit 4 @Ignore
@Disable
void simpleTest() {
    assertEquals(2, 1+1);
}
```

ANNOTATIONS

- Nested test classes with @Nested
  - Must be non-static inner classes
  - Can contain test methods, one @BeforeEach and @AfterEach method
  - No limit for depth of class hierarchy

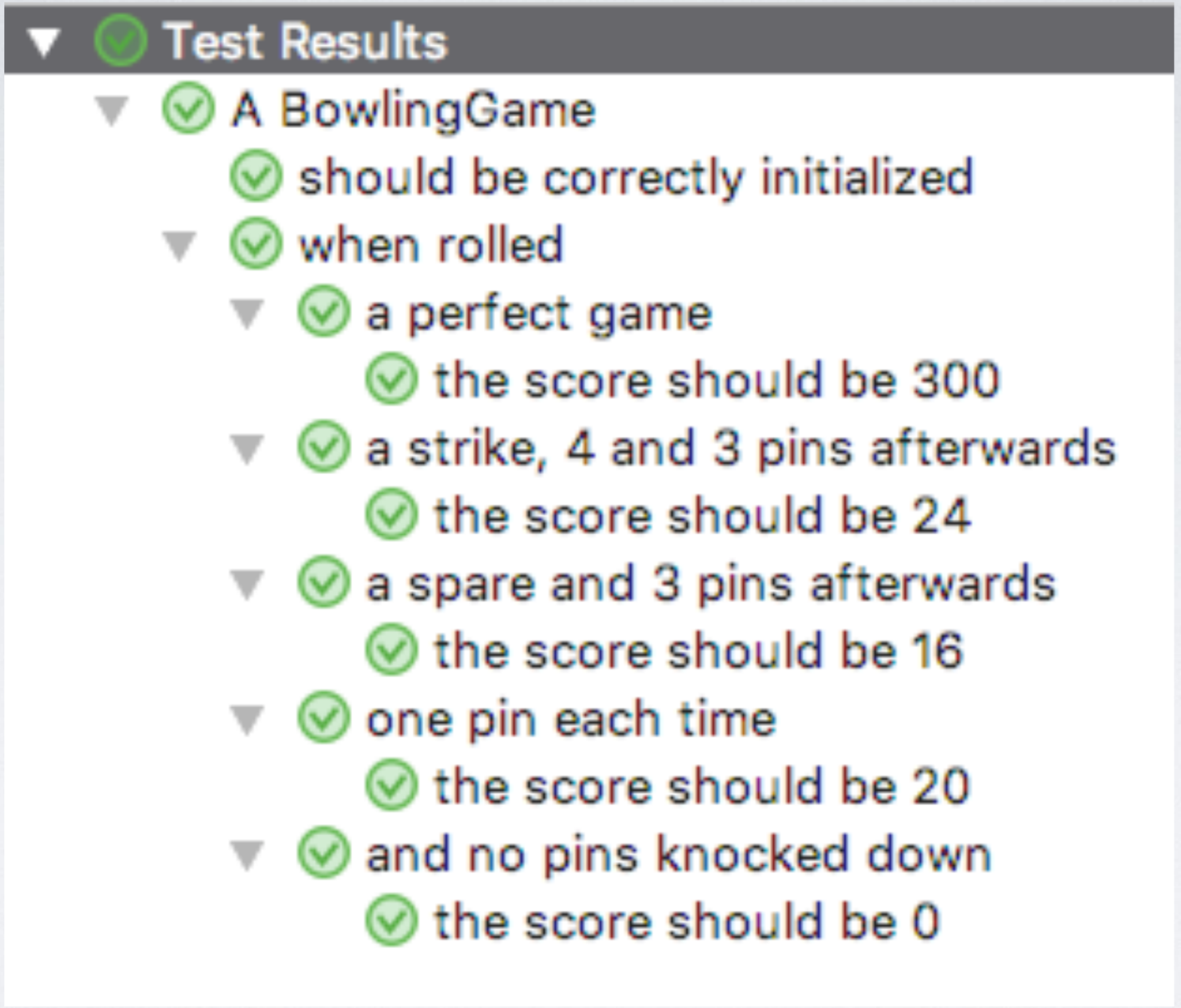
ANNOTATIONS

```
...
@Nested
class InnerClass {

    @Test
    void innerClassTest() {
        assertEquals(4, 2+2);
    }
}
...
```

ANNOTATIONS

```
@Test
@DisplayName("Should validate that the sum of 2 + 2 equals 4")
void shouldCheckSumOfCalculation() {
    assertEquals(4, 2+2);
}
```


Test Results
A BowlingGame
should be correctly initialized
when rolled
a perfect game
the score should be 300
a strike, 4 and 3 pins afterwards
the score should be 24
a spare and 3 pins afterwards
the score should be 16
one pin each time
the score should be 20
and no pins knocked down
the score should be 0

ASSERTIONS

- New class: org.junit.jupiter.api.Assertions
- Use of lambdas for lazy evaluation of messages
- Grouping of assertions
- New way to handle exceptions

ASSERTIONS

```
@Test
void simpleTest() {
    assertEquals(5, 2+2, () -> "Result should be 5");
}
```

ASSERTIONS

```
@Test
void simpleTestForAssertAll() {
    assertAll(
        () -> assertEquals(2, 1+1),
        () -> assertEquals(4, 2+2)
    );
}
```

ASSERTIONS

```
@Test
void shouldThrowException() {
    assertThrows(UnsupportedOperationException.class,
        () -> { throw new UnsupportedOperationException
        ("Operation not supported");
    });
}
```

ASSERTIONS

```
@Test
void shouldCheckThrownException() {
    var exception =
    assertThrows(UnsupportedOperationException.class,
        () -> { throw new UnsupportedOperationException
        ("Operation not supported");
    });
    assertEquals("Operation not supported“, exception.getMessage());
}
```

ASSERTIONS

```
@Test
void testAssertTimeout() {
    assertTimeout(ofSeconds(10),
        () -> { Thread.sleep(15000); });
}
```

ASSERTIONS

```
@Test
void testAssertTimeoutPreemptively() {
    assertTimeoutPreemptively(ofSeconds(10),
        () -> { Thread.sleep(15000); });
}
```

ASSUMPTIONS

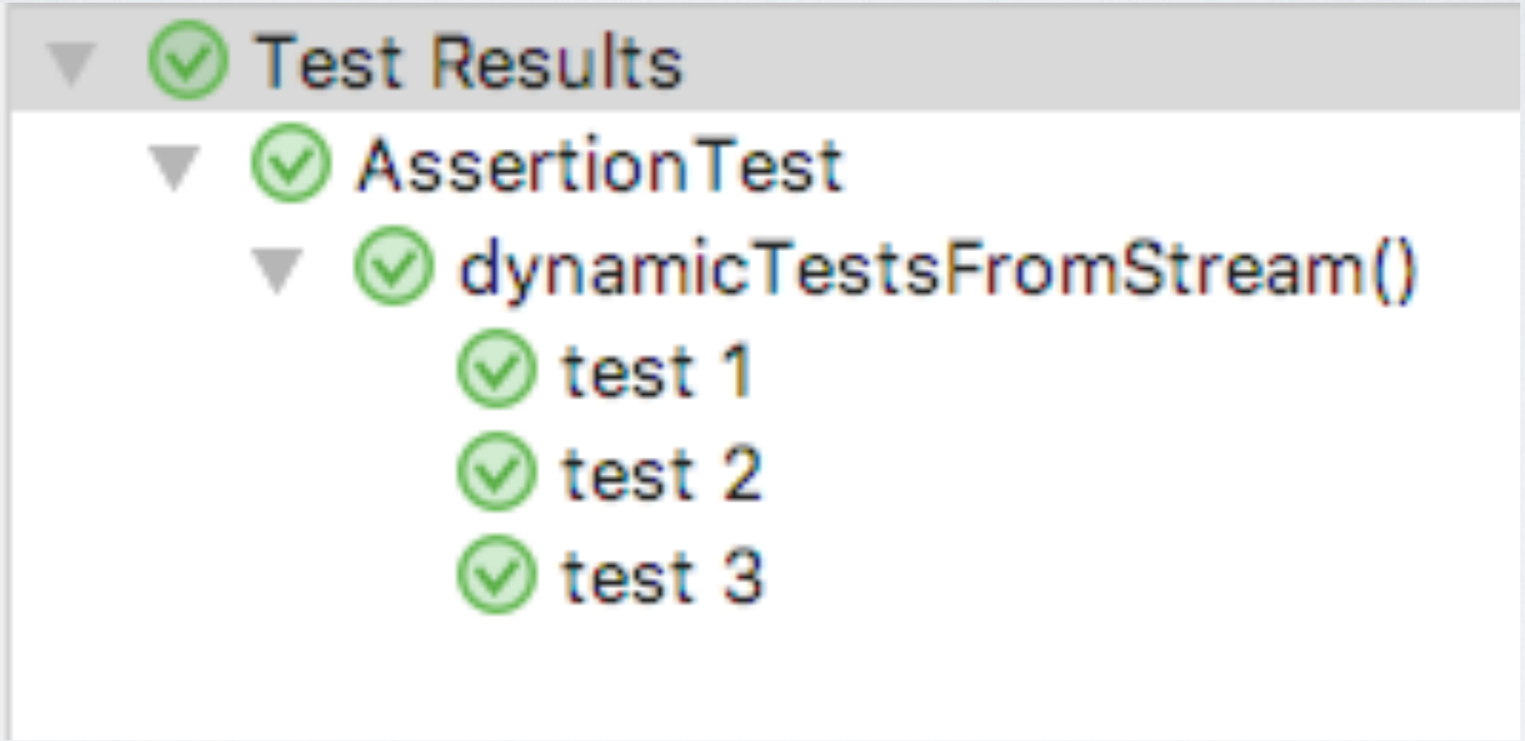
- TestAbortedException if assumption fails -> test is skipped

```
@Test
void assumptionTest() {
    assumeTrue(true);
    assumeFalse(false);
    assumingThat("abc".equals("abc"),
        () -> assertEquals(2, 1+1));
}
```

DYNAMIC TESTS

```
@TestFactory
Stream<DynamicTest> dynamicTestsFromStream() {
    return Stream.of(1, 2, 3)
        .map(i -> dynamicTest("test " + i,
            () -> { assertEquals(i * 2, i + i); }));
}
```

DYNAMIC TESTS



CONDITIONAL TEST EXECUTION

- *ExecutionCondition* as extension API
- *DisabledCondition* simplest example with @Disabled annotation

CONDITIONAL TEST EXECUTION

- @EnabledOnOS(…)
- @EnabledOnJre(…)
- @EnabledIfSystemProperty(named = „“, matches = „“)
- @EnabledIfEnvironmentalVariable(named = „“, matches = „“)
- @EnabledIf(„“), support for scripting languages, EXPERIMENTAL

PARAMETERIZED TESTS

- Experimental feature
- „junit-jupiter-params“ dependency needed

PARAMETERIZED TESTS

```
@ParameterizedTest
@ValueSource(strings = {"java8", "java9", "java10"})
void parameterizedTest(String param) {
    assertTrue(param.contains("java"));
}
```

PARAMETERIZED TESTS

```
@ParameterizedTest(name = "[{index}] => {arguments}")
@ValueSource(strings = {"java8", "java9", "java10"})
void parameterizedTest(String param) {
    assertTrue(param.contains("java"));
}
```

PARAMETERIZED TESTS

- @ValueSource for String, int, long and double
- @EnumSource(…)
- @MethodSource(„methodName“) - method must return Stream, Iterator or Iterable
- @CsvSource({„foo, bar“„„foo2, bar2“})
- @CsvFileSource(resources = …)
- @ArgumentSource(MyArgumentsProvider.class)

WHAT ELSE?

- @Tag and filtering in build script
- @RepeatedTest with dynamic placeholder for @DisplayName
- @TestTemplate / TestTemplateInvocationContextProvider
- Extension API, extensions registered via @ExtendWith


TESTCONTAINERS

INTRODUCTION

- Java library to launch Docker containers during JUnit tests
- Integration tests against the data access layer
- Integration tests with external dependencies (e.g. message broker, database, …)
- UI tests with containerized, Selenium compatible, web browsers

INTRODUCTION

- Current version: 1.8.0
- Requires Docker installation
- Requires Java 8
- Compatible with JUnit

DEPENDENCIES

```
<dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>testcontainers</artifactId>
    <version>1.8.0</version>
    <scope>test</scope>
</dependency>
```

OWN MAVEN MODULES

- MySQL
- PostgreSQL
- Oracle XE
- Kafka
- Selenium
- …

JUNIT 4

```
@ClassRule
public static GenericContainer wildfly =  new GenericContainer("jboss/wildfly")
                    .withExposedPorts(8080, 9990);

@Test
public void getExposedPorts() {
    var ports = wildfly.getExposedPorts();
    var expectedPorts = Arrays.asList(8080, 9990);

    assertEquals(expectedPorts, ports);
}
```

JUNIT 5

```
static GenericContainer wildfly = new GenericContainer("jboss/wildfly").withExposedPorts(8080, 9990);

@BeforeAll static void startUp() { wildfly.start(); }

@AfterAll static void tearDown() { wildfly.stop(); }

@Test void getExposedPorts() {
    var ports = wildfly.getExposedPorts();
    var expectedPorts = Arrays.asList(8080, 9990);

    assertEquals(expectedPorts, ports);
}
```

GENERIC CONTAINER

- Offers flexible support for any container image as test dependency
- Reference public docker images
- Internal dockerized services

GENERIC CONTAINER

- withExposedPorts(…)
- withEnv(…)
- withLabel(…)
- getContainerIpAddress()
- getMappedPort(…)

SPECIALISED CONTAINER

- Create images from Dockerfile
  - withFileFromString(…)
  - withFileFromClasspath(…)
- Use Dockerfile DSL to define Dockerfiles in code

SPECIALISED CONTAINER

```
@Test
void testDockerDSLContainer() {
    var container = new GenericContainer( new ImageFromDockerfile()
                        .withDockerfileFromBuilder(builder -> {
                            builder.from(,,alpine:3.2")
                                    .run("apk add --update git")
                                    .cmd("git", ,,version")
                                    .build(); }))
                        .withExposedPorts(80);
    assertEquals(Arrays.asList(80), container.getExposedPorts());
}
```

SPECIALISED CONTAINER

- Use database container to test database specific features
  - No local setup or VM
  - 100% database compatibility instead of in-memory H2
  - MySQL
  - PostgreSQL
  - Oracle XE

SPECIALISED CONTAINER

```
private static PostgreSQLContainer postgres = new PostgreSQLContainer();
// postgresql container start() / stop() omitted …

@Test
void testSimpleSQL() throws SQLException {
    // HikariConfig, HikariDataSource and Statement omitted …
    statement.execute("SELECT 1");
    ResultSet resultSet = statement.getResultSet();

    resultSet.next();
    assertEquals(1, resultSet.getInt(1));
    }
}
```

SELENIUM WEBDRIVER CONTAINER

- Compatible with Selenium 2 / Webdriver tests
- Use of all Selenium docker images from selenium-docker project
- Clean and fixed environment for each browser and test
- Selenium API and browser compatibility assured
- VNC recording (optionally just failed tests)

SELENIUM WEBDRIVER CONTAINER

```
@Rule
public BrowserWebDriverContainer chrome = new BrowserWebDriverContainer()
    .withDesiredCapabilities(DesiredCapabilities.chrome())
    .withRecordingMode(RECORD_ALL, new File(„./build/"));

@Test
public void searchForTestcontainersOnGoogle() {
    var driver = chrome.getWebDriver();
    driver.get(„http://www.google.com");
    driver.findElement(By.name("q")).sendKeys("testcontainers");
    driver.findElement(By.name("q")).submit();
    assertEquals("testcontainers", driver.findElement(By.name("q")).getAttribute("value"));
}
```

FUTURE

- Testcontainers 2.0
  - API cleanup
  - Decoupling from JUnit 4 to support other frameworks directly

ALTERNATIVES

- TestNG
- Other JVM languages
    - Groovy, Spock, Testcontainers-Spock
    - Kotlin, Testcontainers (with workaround)

QUESTIONS

@zordan_f

github.com/timriemer/jdk10_junit_testcontainers