JUnit 5

13th Dec 2017

Ingo Mohr

- New major version of JUnit since 2006
- Released in Sep 2017, latest release 5.0.2 Nov 12th
- Available in Eclipse Oxygen 1.a
- Supports Java 8
- Composed out of several components in 3 sub-projects

JUnit 5

=

JUnit Platform + JUnit Jupiter + JUnit Vintage

- Foundation of launching test frameworks on the JVM
- Provides TestEngine API for developing a testing framework
- Provides Console Launcher to…
  - Launch Platform from Console
  - Build plugins for Gradle and Maven
- JUnit 4 based Runner to run any TestEngine

JUnit Jupiter

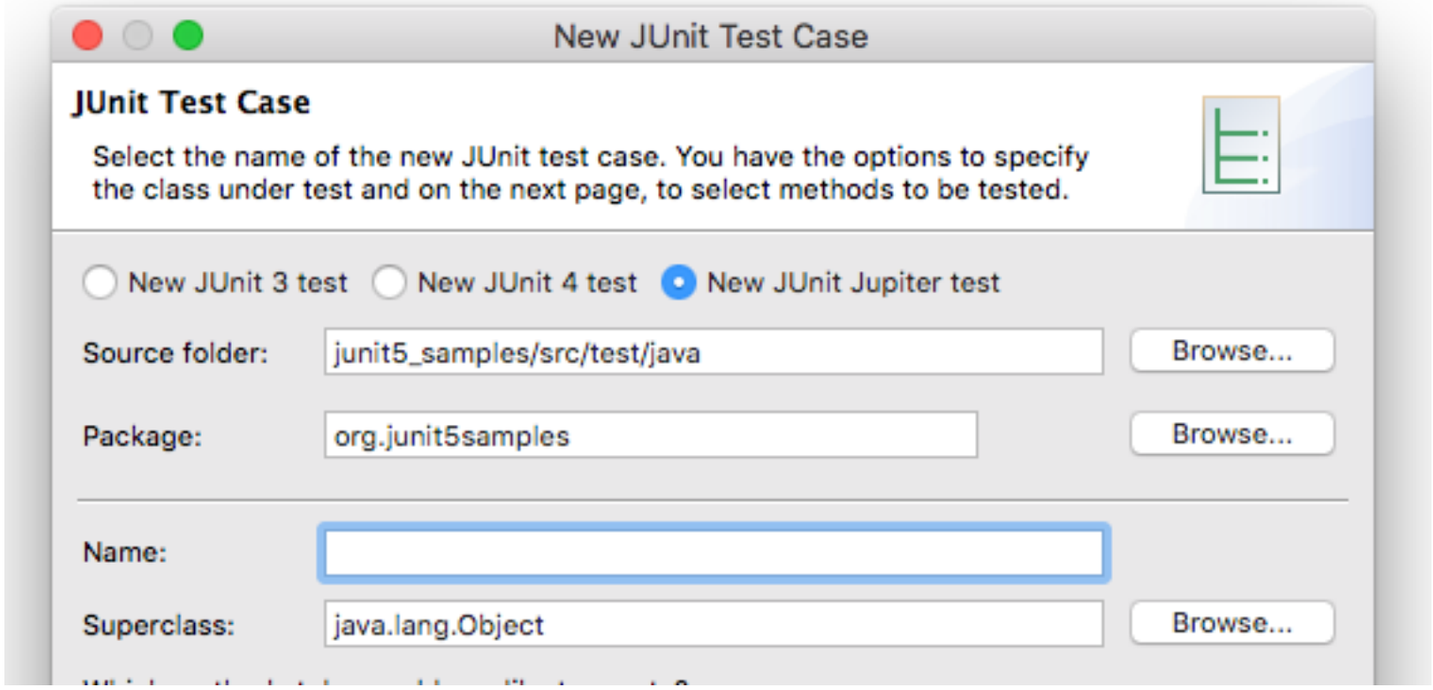
- Jupiter: 5th star from the sun
- For writing tests and extensions in JUnit 5
- Combination of …
  - Programming Model
  - Extension Model

JUnit Vintage

- TestEngine for running JUnit 3 and JUnit 4 based tests on the platform

New Wizard

- You can select what JUnit version you want to use for your test



Basic Stuff

- `@BeforeAll`, `@AfterAll`
- `@BeforeEach`, `@AfterEach`
- `public` access modifier not required anymore
- Message in assert methods is now 3rd parameter
- `@Disabled` for disabled tests

Exceptions

- `assertThrows`(Class<T>, Executable) : T

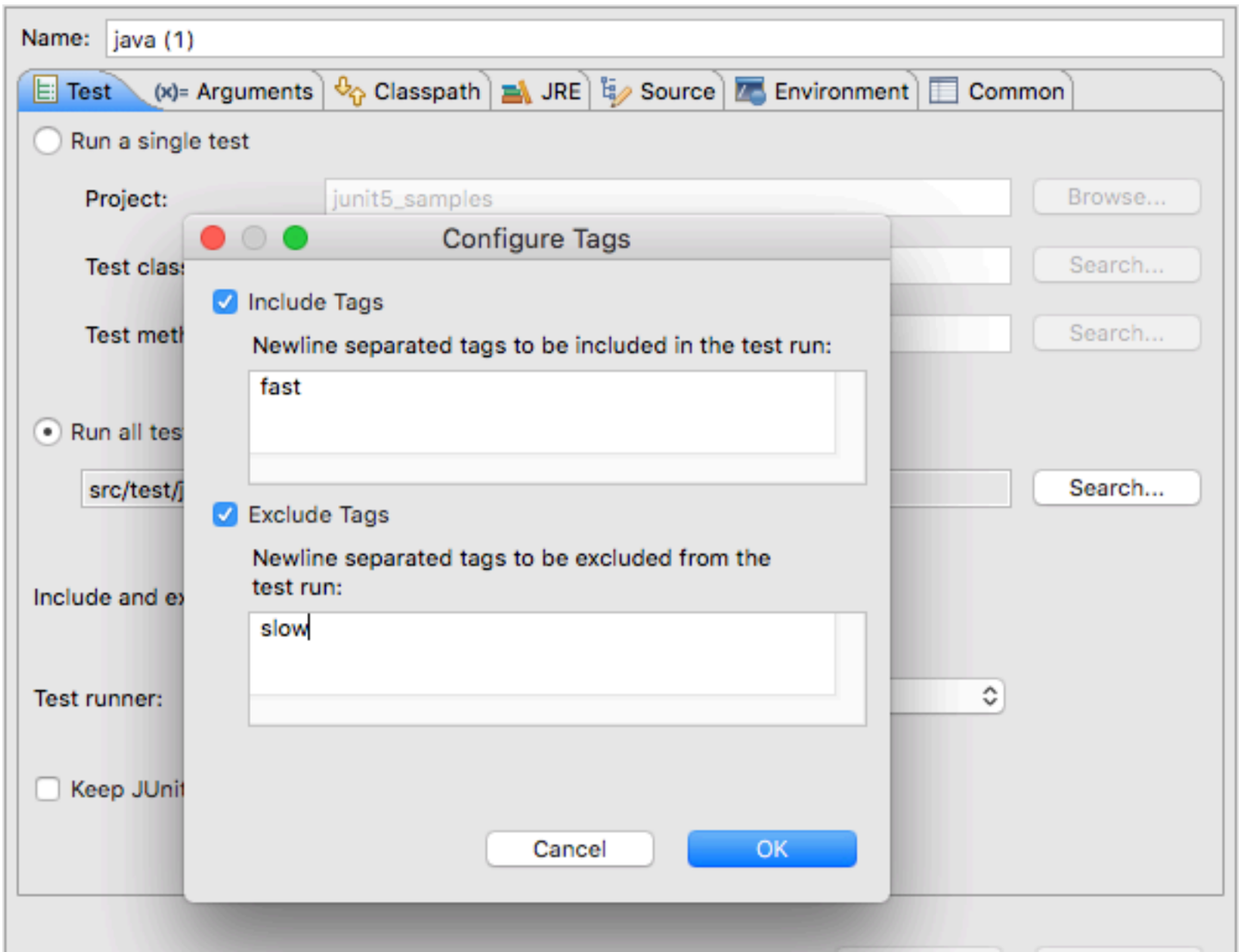
- You can continue working on the `Throwable` object

AssertAll

- Perform a group of checks and notify about all failures

Tags

- Annotate `@Tag`s for filtered execution



Parameterized Tests

- Use `@ParameterizedTest` and ...
  - `@ValueSource` for values
  - `@EnumSource` for enum values
  - `@MethodSource` for simple or complex values provided by Streams
  - `@Csv(File)Source` for comma-separated values
  - `@ArgumentsSource` for arguments provider

Display Name

- Use `@DisplayName` to specify a name to be shown in the result tree

- Can be combined w/ parameterized tests

Dynamic Tests

- New in Jupiter to create and run tests at runtime
- Use `@TestFactory`
- `@Before/AfterEach` are executed for each `@TestFactory`, but not for each test
- Currently Experimental Feature

Nested Tests

- For better grouping of tests

- Use `@Nested` for inner tests

Dependency Injection

- Both test constructors and methods may now have parameters
- This enables DI for constructors and parameters
- 3 built-in `ParameterResolver`s
  - `TestInfoParameterResolver` for info on the test
  - `RepetitionInfoParameterResolver` for repetition info
  - `TestReporterParameterResolver` to publish additional info
- Write your own with `@ExtendWith`

Repeated Tests

- Use `@RepeatedTest` for test repetition

- Can be combined w/ `@DisplayName`

Thank you