
Evolving JUnit 5
From 5.0 to 5.9
5

Marc Philipp

Software Engineer at Gradle

JUnit committer since 2012
team lead since 2016

Mastodon:
@marcphilipp@chaos.social
Web: marcphilipp.de
Email: marc@junit.org

The JUnit team

JUnit 5 is 5! 🎉

5.0 – September 10, 2017
5.1 – February 18, 2018
5.2 – April 29, 2018
5.3 – September 11, 2018
5.4 – February 7, 2019
5.5 – June 30, 2019
5.6 – January 7, 2020
5.7 – September 13, 2020
5.8 – September 12, 2021
5.9 – July 26, 2022

Thank you to our sponsors!

https://junit.org/sponsoring

IntelliJ IDEA

The Java IDE
for Professional Developers
by JetBrains

GOLD SPONSOR

Since February 2019

Octopus Deploy

Continuous Delivery,
Deployment and
DevOps platform

GOLD SPONSOR

Since September 2020

premium minds

BRONZE SPONSOR

Since July 2019

testmo

BRONZE SPONSOR

Since January 2021

code fortynine

BRONZE SPONSOR

Since July 2021

Ubie

BRONZE SPONSOR

Since February 2022

infoSupport
Solid Innovator

BRONZE SPONSOR

Since August 2022

STILTSOFT

BRONZE SPONSOR

Since August 2022

MICROMATA
Erfolg ist programmierbar.

MICROMATA

SILVER SPONSOR

Since May 2019

QUO CARD

QUO CARD

SILVER SPONSOR

Since March 2022

code intelligence

BRONZE SPONSOR

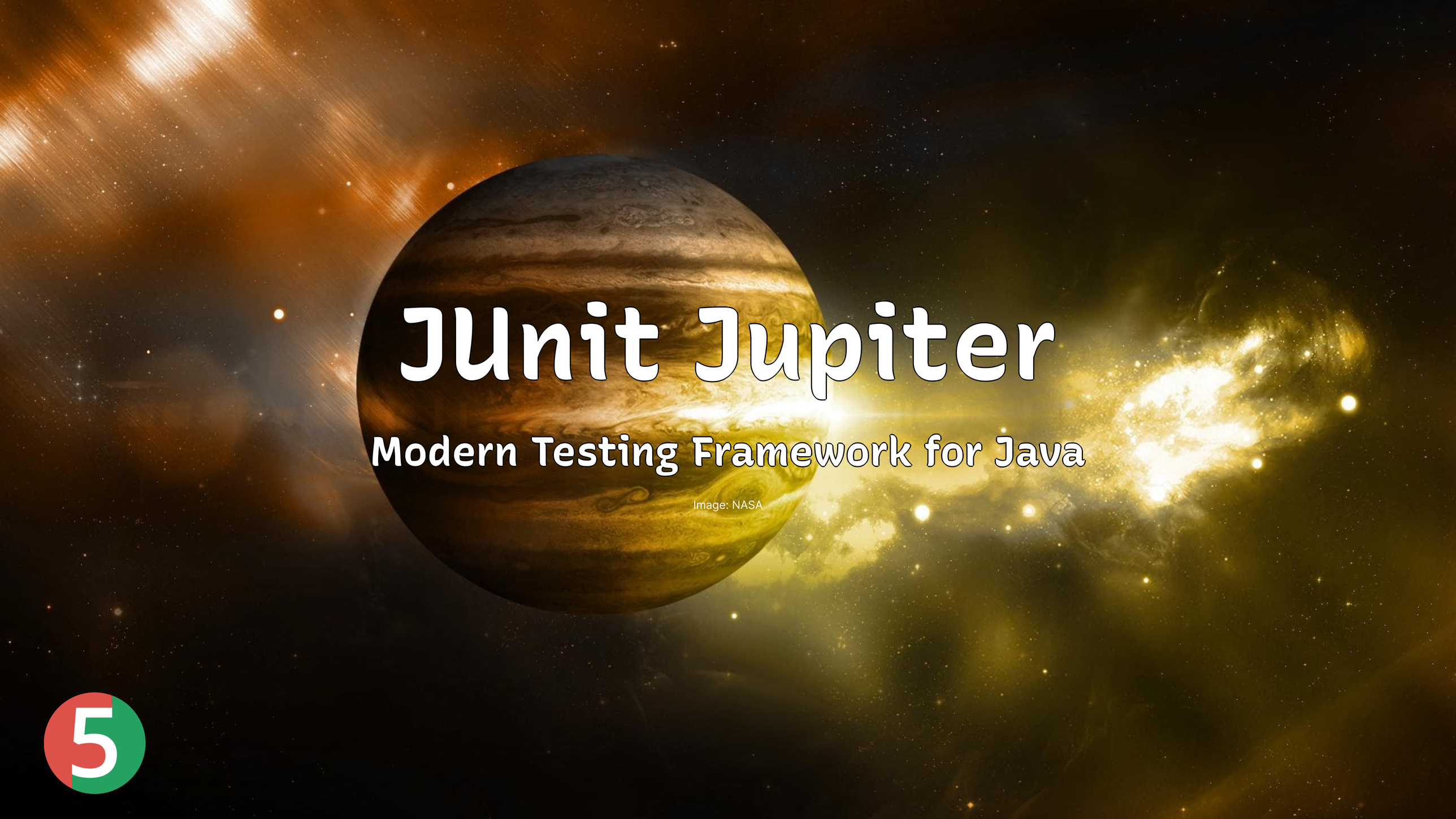
Since November 2022

Agenda

1. How to write tests and extensions using JUnit 5?
2. What is the JUnit Platform and why do we need it?
3. What’s still to come and how to get started?

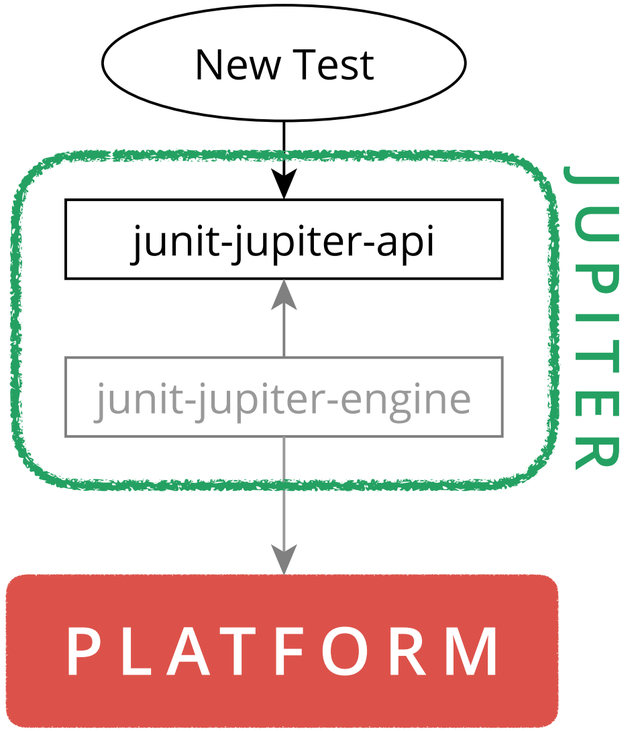
JUnit Jupiter

Modern Testing Framework for Java



JUnit Jupiter

- API for writing tests and extensions
- Requires Java 8 or later
- Tested with Java, Kotlin, and Groovy
- Ships with Java module descriptors and OSGi metadata



Basics

```
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
class Tests {
  @Test void test() {
    assertEquals(2, 1 + 1);
  }
}
```

- `@Test` is now in `org.junit.jupiter.api`
- `Assertions` instead of `Assert` – a few new ones like `assertThrows`, `assertAll`
- `public` modifier is *not* required anymore

Lifecycle Methods

```
class Tests {
  Path resource;
  @BeforeEach void createResource() {
    resource = // ...
  }
  @Test void doSomethingWithResource() {
    assertNotNull(resource); // use resource
  }
}
```

`@BeforeAll`, `@BeforeEach`, `@AfterEach`, `@AfterAll` have new names compared to JUnit 4.x

More Basics

```
@DisplayName("Calculator")
class CalculatorTests {
  @Disabled
  @Tag("feature-addition")
  @DisplayName("should return sum of two numbers when adding")
  void add() {/*...*/}
}
```

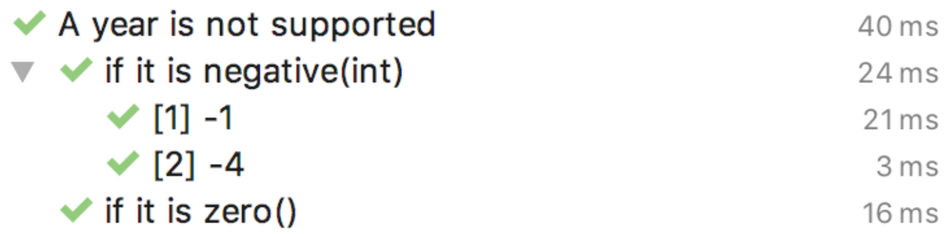
- `@Disabled` instead of `@Ignore` in JUnit 4.x
- `@Tag` instead of `@Category` in JUnit 4.x
- Custom `@DisplayNames`

Display Name Generators 5.4

```
@DisplayNameGeneration(ReplaceUnderscores.class)
class A_year_is_not_supported {

    @Test
    void if_it_is_zero() {/*...*/}

    @ParameterizedTest
    @ValueSource(ints = { -1, -4 })
    void if_it_is_negative(int year) {/*...*/}
}
```



Default is configurable via configuration parameter 5.5

Nested Tests

```java
@DisplayName("A stack")
class StackTests {
    Stack<Object> stack = new Stack<>();

    @Nested @DisplayName("when new")
    class WhenNew {
        @Test @DisplayName("is empty")
        void isEmpty() {
            assertTrue(stack.isEmpty());
        }

        @Nested @DisplayName("after pushing an element")
        class AfterPushing {
            @BeforeEach
            void pushAnElement() {
                stack.push("an element");
            }
            @Test @DisplayName("returns the element when popped and is empty")
            void returnElementWhenPopped() {
                assertEquals("an element", stack.pop());
                assertTrue(stack.isEmpty());
            }
        }
    }
}
```

- ✔ StackTests
  - ✔ when new
    - ✔ is empty
    - ✔ after pushing an element
      - ✔ returns the element when popped and is empty

Special Assertions for Kotlin 5.1

```kotlin
import org.junit.jupiter.api.Assertions.assertEquals
import org.junit.jupiter.api.assertAll
import org.junit.jupiter.api.assertThrows

class KotlinAssertionsDemo {

    @Test
    fun `expected exception testing`() {
        val exception = assertThrows<ArithmeticException>("Should throw an exception") {
            Calculator().divide(1, 0)
        }
        assertEquals("/ by zero", exception.message)
    }

    @Test
    fun `grouped assertions`() {
        val person = Person("Jane", "Doe")
        assertAll("Person properties",
            { assertEquals("Jane", person.firstName) },
            { assertEquals("Doe", person.lastName) }
        )
    }
}
```

Test Instance Lifecylce

```kotlin
@TestInstance(PER_CLASS)
class KotlinLifecycleDemo {

    private lateinit var calculator: Calculator

    @BeforeAll
    fun `create calculator`() {
        calculator = Calculator()
    }

    @Test
    fun `test something`() {
        // ...
    }
}
```

Default is configurable via configuration parameter

Parallel Execution 5.3

- Tests are run sequentially by default
- Opt-in and configure parallel execution via configuration parameters
- `@Execution(SAME_THREAD` or `CONCURRENT)`
- Declarative synchronization primitives: `@ResourceLock(value = "key", mode = READ)` and `@Isolated` 5.7

More ways to test (Demo)

https://github.com/marcphilipp/junit5-demo/tree/20230412-jug-zurich

More ways to test (Recap)

- `@ParameterizedTest` with different `@Source` annotations
  - `@ValueSource`, `@EnumSource`, `@CsvSource`, `@CsvFileSource`, `@MethodSource`, `@NullSource` 5.4, `@EmptySource` 5.4, `@ArgumentsSource(MyProvider.class)`, `@YourCustomSource`
- `@RepeatedTest` for flaky tests
- `@TestFactory` to produce *dynamic* tests

Extensions

- Allow to hook into test discovery and execution
- Allows extracting reusable behavior and encapsulating it in an extension
- Extensions can make writing tests simpler

Extensions (Demo)

https://github.com/marcphilipp/junit5-demo/tree/20230412-jug-zurich

Extension Registration

- Declarative: `@ExtendWith`
  - on classes or methods
  - on fields and parameters 5.8
- Programmatic: `@RegisterExtension` on fields 5.1
- Global: Via `ServiceLoader` (opt-in via configuration parameter)

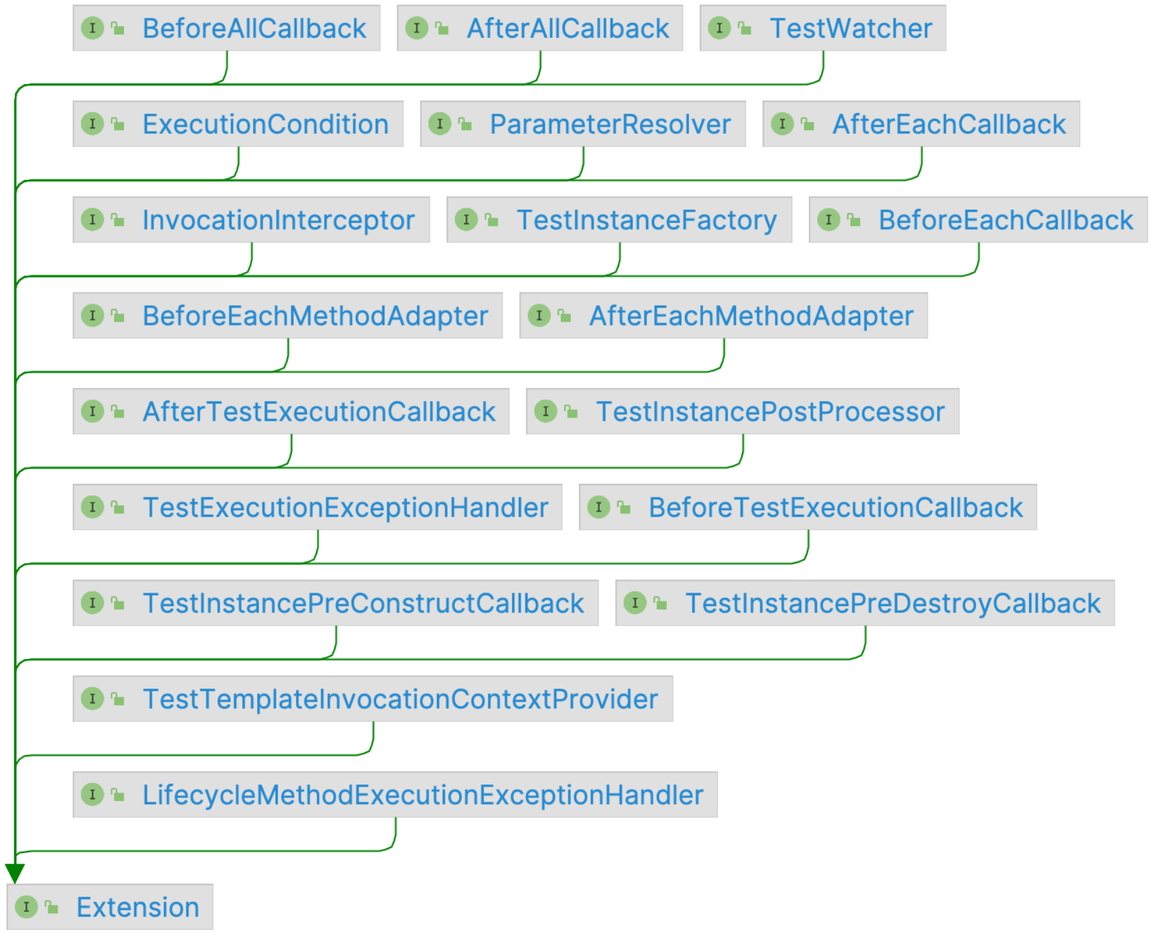
Extension Implementation

```java
package org.junit.jupiter.api.extension;

/**
 * Marker interface for all extensions.
 * ...
 */
public interface Extension {}
```

- `Extension` marker interface
- one extension – *n* extension points/interfaces

Extension Points



Support Classes

Package `org.junit.platform.commons.support` contains:

- `AnnotationSupport` to scan for annotations
- `ReflectionSupport` to scan the class path or look up and execute methods etc.

Composed Annotations

Use Jupiter annotations as meta-annotations to create your own annotations.

```
@Retention(RUNTIME)
@Target(METHOD)
@ExtendWith(DisabledOnWeekdaysExtension.class)
@Tag("example")
public @interface DisabledOnWeekdays {
    DayOfWeek[] value();
}
```

Built-in Temp Dir Support 5.4

```java
import org.junit.jupiter.api.io.TempDir;

@Test
void writeAndReadFile(@TempDir Path tempDir) throws Exception {
    Path testFile = tempDir.resolve("test.txt");

    Files.write(testFile, asList("foo", "bar"));

    List<String> actualLines = Files.readAllLines(testFile);
    assertIterableEquals(asList("foo", "bar"), actualLines);
}
```

- Supports multiple temp dirs 5.8
- Configurable cleanup-mode 5.9

Timeouts

- `assertTimeout{Preemptively}` allows writing assertions for code blocks within a test
- `@Timeout` is a declarative way to specify timeouts for test or lifecycle methods 5.5
  - Configurable thread mode 5.9
- `junit.jupiter.execution.timeout.{...}.default` configuration parameters can be used to configure defaults 5.5

Built-in Conditions (1/2)

- `@Enabled` / `DisabledOnOs({LINUX, MAC, …})` 5.1
  - `architectures = "aarch64"` support 5.9
- `@Enabled` / `DisabledOnJre({JAVA_11, …})` 5.1
- `@Enabled` / `DisabledForJreRange(min = JAVA_9, max = JAVA_10)` 5.6

Built-in Conditions (2/2)

- `@Enabled` / `DisabledIfSystemProperty(named = "someKey", matches = "someValue")` 5.1
- `@Enabled` / `DisabledIfEnvironmentVariable(named = "SOME_KEY", matches = "SOME_VALUE")` 5.1
- `@Enabled` / `DisabledIf("customCondition")` 5.7
- `@Enabled` / `DisabledInNativeImage` 5.9.1

Third-Party Extensions

JUnit Pioneer, Spring, Mockito, Testcontainers, Docker, Wiremock, JPA, Selenium/WebDriver, DbUnit, Kafka, Jersey, GreenMail, S3Mock, Citrus Framework, XWiki, ...

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

Agenda

1. How to write tests and extensions using JUnit 5? ✅
2. What is the JUnit Platform and why do we need it?
3. What’s still to come and how to get started?

Questions?

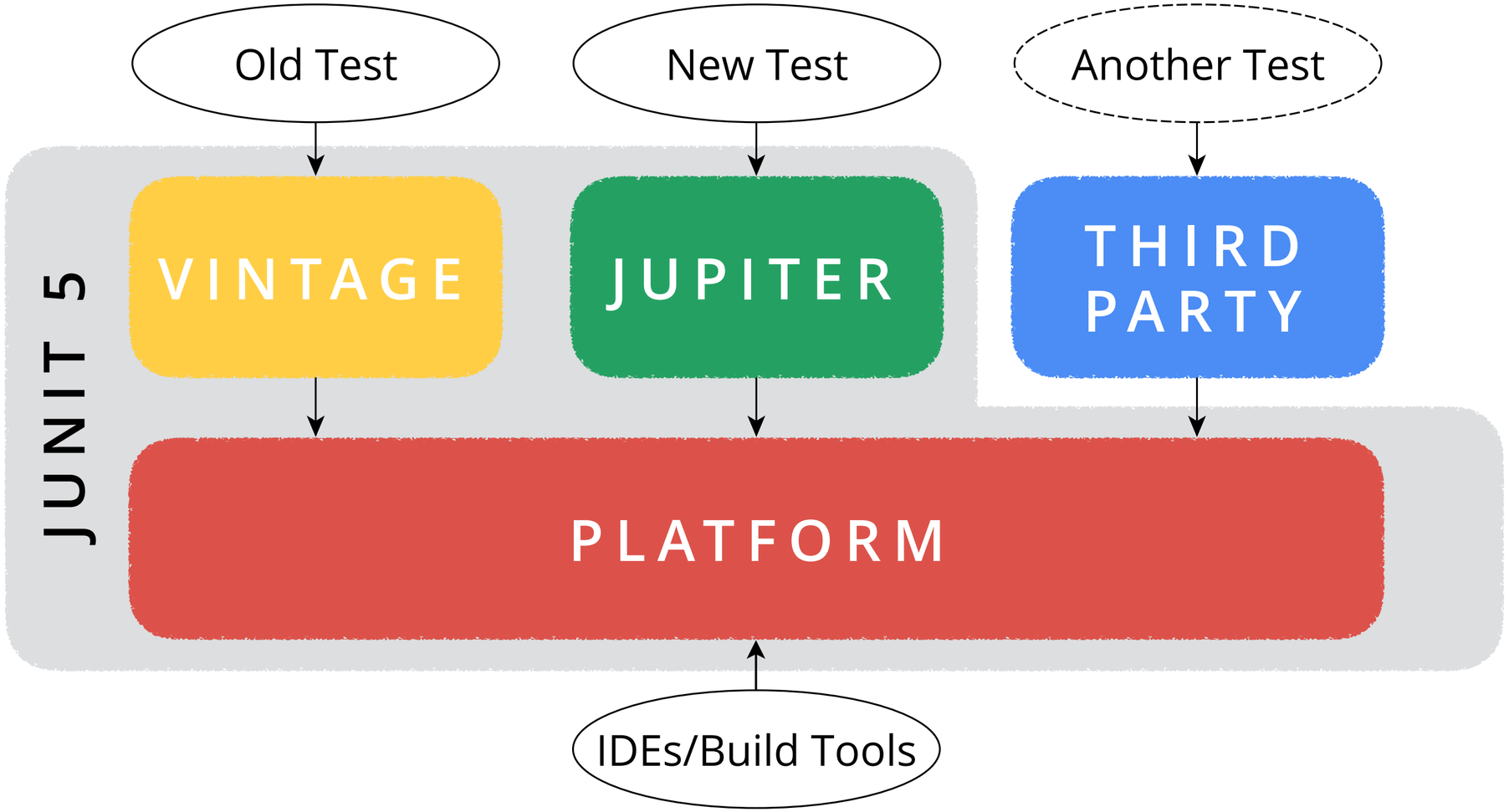
JUnit Platform

Platform for Testing on the JVM

5

Separation of Concerns

1. An API to write tests and extensions (Jupiter API)
2. Extensible mechanisms to discover and execute tests (Test Engine SPI)
3. An API for test execution by tools (Launcher API)


Old Test
New Test
Another Test
JUNIT 5
VINTAGE
JUPITER
THIRD PARTY
PLATFORM
IDEs/Build Tools

JUnit 5
=
Jupiter + Vintage + Platform

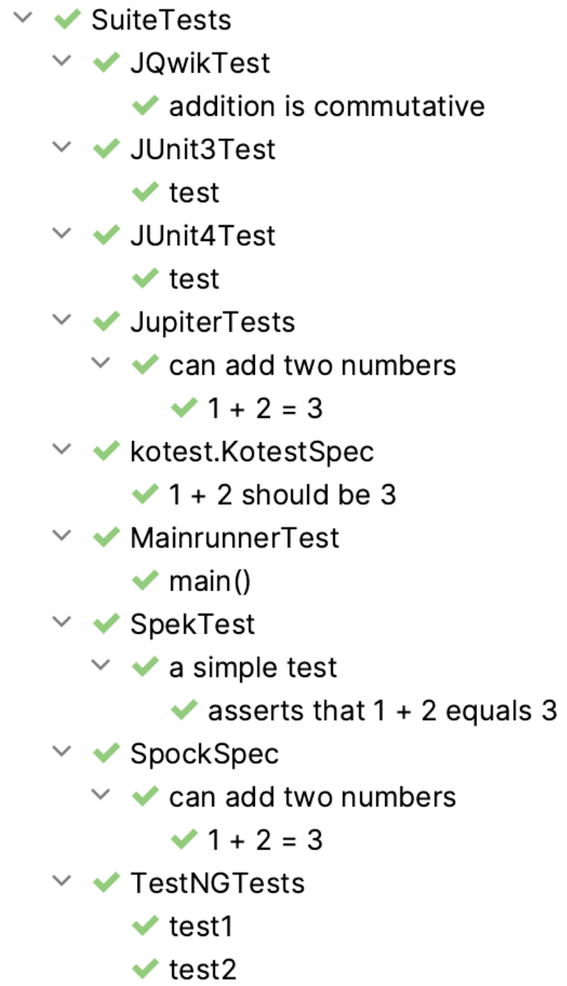
Third-party Engines

Spock, TestNG, jqwik, Cucumber, Kotest, Specsy, Spek, Drools, ScalaTest, Brahms, Mainrunner, ...

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

Mixing Engines

- Multiple test engines can be used in a single test run
- Allows to gradually migrate tests from one test engine to another (e.g. from Vintage to Jupiter)
- *Demo:* junit5-multiple-engines on GitHub



Declarative Test Suites 5.8

```java
@Suite
@SuiteDisplayName("JUnit Platform Suite Demo")
@SelectPackages("example")
@IncludeClassNamePatterns(".*Tests")
class SuiteDemo {
}
```

Made available via `junit-platform-suite-engine`

Tag Expressions 5.1

Precisely specify which tests to run based on tags:

```
test {
    useJUnitPlatform {
        includeTags("(smoke & feature-a) | (!smoke & feature-b)")
    }
}
```

Test Kit 5.4

`EngineTestKit` allows testing `Extension` or `TestEngine` implementations.

```
EngineExecutionResults results = EngineTestKit
    .engine("junit-jupiter")
    .selectors(selectClass(ExampleTestCase.class))
    .execute();

results.testEvents()
    .assertThatEvents()
        .haveExactly(1, event(test("skippedTest"),
            skippedWithReason("for demonstration purposes")));
        .haveExactly(1, event(test("failingTest"),
            finishedWithFailure(message("on purpose"))));
```

New XML reporting format 5.9

- New Open Test Reporting format
- Enabled via `junit.platform.reporting.open.xml.enabled=true` configuration parameter
- Full support for all features of the JUnit Platform such as hierarchical test structures, display names, tags, …
- Extensible via additional XML schemas

Event-based format 5.9

```
<?xml version="1.0" ?>
<e:events xmlns="https://schemas.opentest4j.org/reporting/core/0.1.0" xmlns:e="https://schemas.oper
<infrastructure><hostName>...</hostName><userName>marc</userName><operatingSystem>Mac OS X</operat:
<e:started id="766" name="JUnit Jupiter" time="2022-09-23T07:54:50.324086Z"><metadata><junit:unique
<e:started id="767" name="ColorPaletteTests" parentId="766" time="2022-09-23T07:54:50.324275Z"><met
<e:started id="768" name="DemonstratePalettesTests" parentId="767" time="2022-09-23T07:54:50.32445(
<e:started id="769" name="flat_default()" parentId="768" time="2022-09-23T07:54:50.324589Z"><metada
<e:finished id="769" time="2022-09-23T07:54:50.326039Z"><result status="SUCCESSFUL"></result></e:f
<e:finished id="768" time="2022-09-23T07:54:50.332254Z"><result status="SUCCESSFUL"></result></e:f
<e:finished id="767" time="2022-09-23T07:54:50.333862Z"><result status="SUCCESSFUL"></result></e:f
<e:finished id="766" time="2022-09-23T07:54:50.812066Z"><result status="SUCCESSFUL"></result></e:f
</e:events>
```

- 👍 Suitable for writing and streaming
- 👎 Not very human-readable

Hierarchical format 5.9

Converted from event-based format via CLI tool

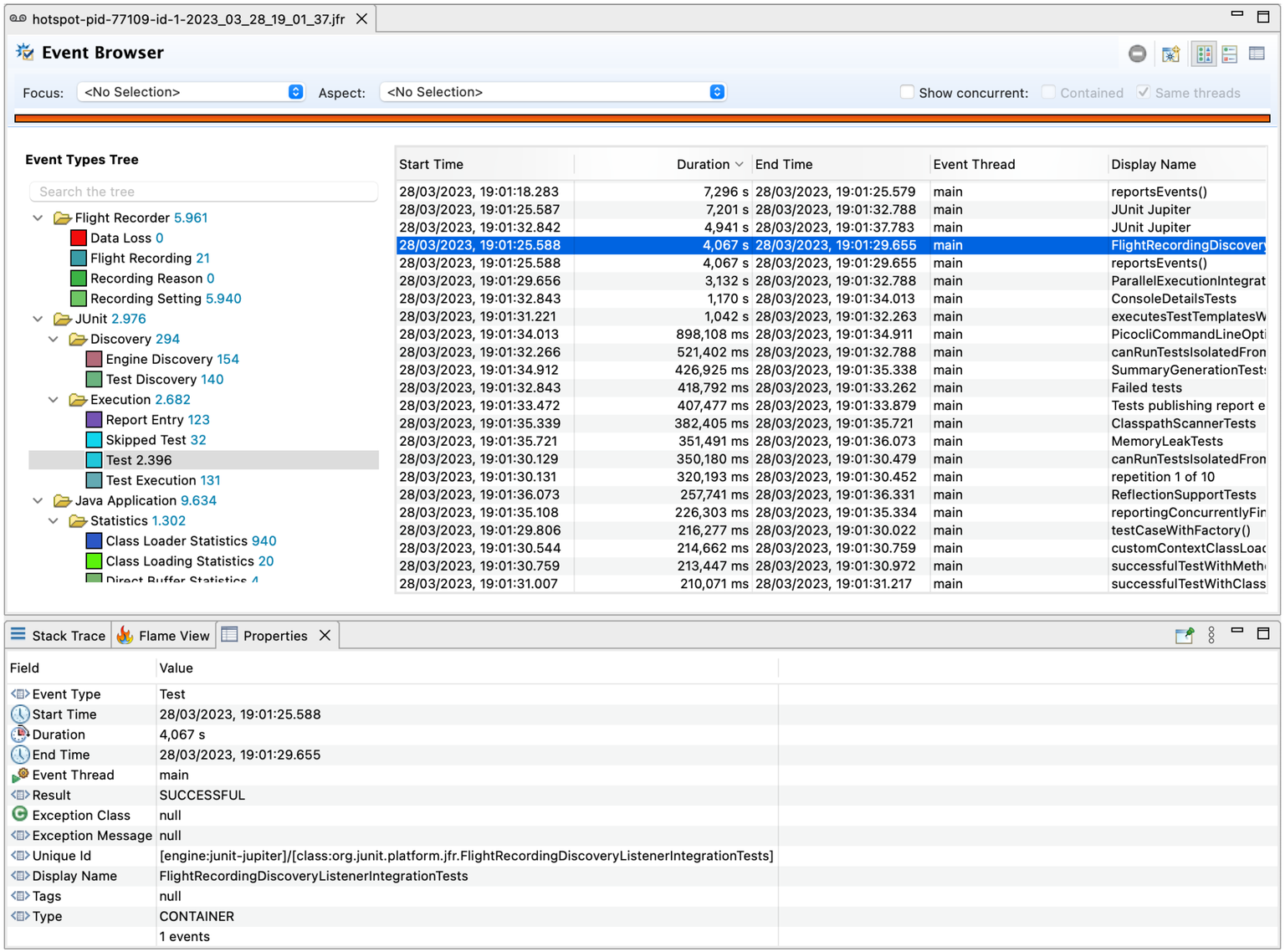
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<h:execution xmlns:h="https://schemas.opentest4j.org/reporting/hierarchy/0.1.0" xmlns="https://sche
  <infrastructure><!-- ... --></infrastructure>
  <h:root duration="PT0.48798S" name="JUnit Jupiter" start="2022-09-23T07:54:50.324086Z">
    <metadata>
      <junit:uniqueId>[engine:junit-jupiter]</junit:uniqueId>
      <junit:legacyReportingName>JUnit Jupiter</junit:legacyReportingName>
      <junit:type>CONTAINER</junit:type>
    </metadata>
    <result status="SUCCESSFUL"/>
    <h:child duration="PT0.009587S" name="ColorPaletteTests" start="2022-09-23T07:54:50.324275Z">
      <metadata><!-- ... --></metadata>
      <sources><java:classSource className="org.junit.platform.console.tasks.ColorPaletteTests"/></
      <result status="SUCCESSFUL"/>
      <h:child duration="PT0.007798S" name="DemonstratePalettesTests" start="2022-09-23T07:54:50.32
        <metadata><!-- ... --></metadata>
        <sources><java:classSource className="org.junit.platform.console.tasks.ColorPaletteTests$De
        <result status="SUCCESSFUL"/>
        <h:child duration="PT0.00145S" name="flat_default()" start="2022-09-23T07:54:50.324589Z">
          <metadata><!-- ... --></metadata>
          <sources><java:methodSource className="org.junit.platform.console.tasks.ColorPaletteTests
          <result status="SUCCESSFUL"/>
        </h:child>
      </h:child>
```

Java Flight Recorder Support 5.7

To correlate test execution with GC and other JVM events

1. Add `junit-platform-jfr` dependency
2. Add `-XX:StartFlightRecording=...` JVM arg
3. Open in Java Mission Control etc.

Java Flight Recorder Support 5.7



Agenda

1. How to write tests and extensions using JUnit 5? ✅
2. What is the JUnit Platform and why do we need it? ✅
3. What’s still to come and how to get started?

Roadmap and Resources

Image: NASA

On the horizon…

- Dry-run mode and discovery-only console launcher
- Additions to the reporting format (screenshots, …)
- Parameterized test classes
- Built-in support for GraalVM native images
- *Your ideas?*

Getting Started

- User Guide:
  docs.junit.org
- Sample projects (Ant/Bazel/Gradle/Maven/sbt):
  start.junit.org
- Javadoc:
  api.junit.org

Today's Example Code

https://github.com/marcphilipp/junit5-demo/tree/20230412-jug-zurich

Questions?

More Questions or Feedback?

- Questions: junit5 tag on StackOverflow
- Code & Issues: junit-team/junit5 on GitHub
- Chat: junit-team/junit5 on Gitter
- Twitter: @junitteam

Thanks! 👋