

A long time ago in a conference hall
far, far away...

JUNIT

REVENGE OF THE 5th

Martin Nachev

JUnit 4

- Release date: Feb 2006 (based on Java 5)
- No modularity: one jar file a.k.a. big ball of mud
- No official API for tool integration
- Extension Mechanism
 - Runners (power) vs Rules (composability)

Introducing JUnit 5

- First General Availability Release: September 2017
 - 5.4.1 - March 2019
- Requirements: Java 8 +

IntelliJ IDEA 2016.2 +

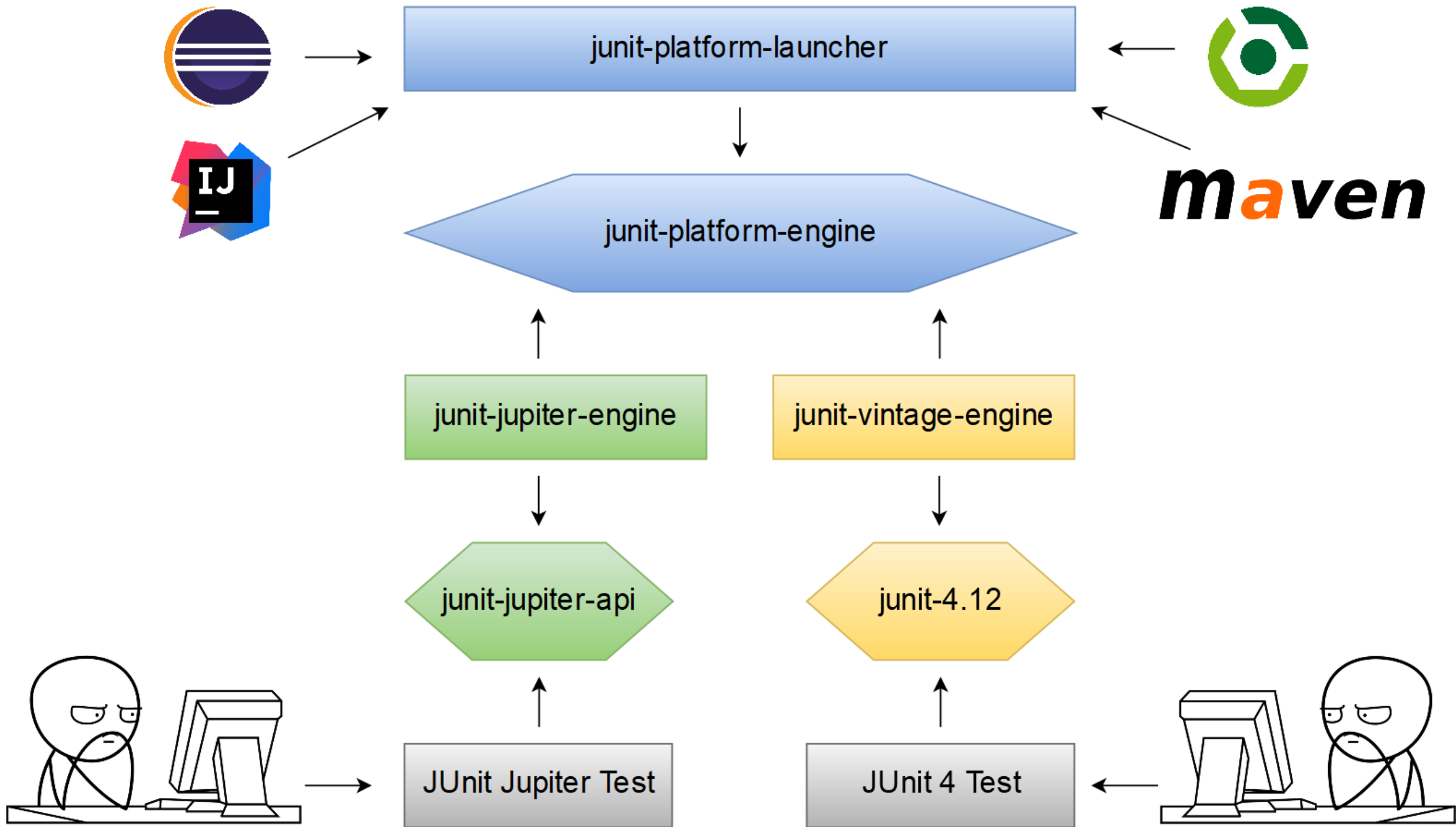
Eclipse Oxygen 4.7.1a +

Apache NetBeans 10

Maven Surefire 2.22.0 +

Gradle 4.6 +

Apache Ant 1.10.3 +



Platform for the JVM

- Third party Test Engine implementations already exist for Scala, Kotlin, Groovy, Cucumber...
- <https://github.com/junit-team/junit5/wiki/Third-party-Extensions>

JUnit 4 vs JUnit Jupiter

```
<dependencies>
```

```
  <dependency>
```

```
-      <groupId>junit</groupId>
```

```
-      <artifactId>junit</artifactId>
```

```
-      <version>4.12</version>
```

```
+      <groupId>org.junit.jupiter</groupId>
```

```
+      <artifactId>junit-jupiter</artifactId>
```

```
+      <version>5.4.1</version>
```

```
+    </dependency>
```

```
+    <!-- optional -->
```

```
+    <dependency>
```

```
+      <groupId>org.junit.vintage</groupId>
```

```
+      <artifactId>junit-vintage-engine</artifactId>
```

```
+      <version>5.4.1</version>
```

```
  </dependency>
```


JUnit 4 & Vintage

`org.junit.*`

`@Before`

`@After`

`@BeforeClass`

`@AfterClass`

`@Ignore`

JUnit Jupiter

`org.junit.jupiter.api.*`

`@BeforeEach`

`@AfterEach`

`@BeforeAll`

`@AfterAll`

`@Disabled`

Environment Conditions

```
@EnabledOnOs({ LINUX, MAC })
```

```
@DisabledOnOs(WINDOWS)
```

```
@EnabledOnJre(JAVA_8)
```

```
@DisabledOnJre({ JAVA_9, JAVA_10 })
```

```
@EnabledIfSystemProperty(named = "os.arch", matches = ".*64.*")
```

```
@DisabledIf("Math.random() < 0.314159")
```

```
...
```

Visibility, Timeout and Exceptions

```
-public class BasicTest {
```

```
+class BasicTest {
```

```
-    @Test(timeout = 1000, expected = RuntimeException.class)
```

```
-    public void test() {
```

```
-        executeLongMethod();
```

```
-        throwException();
```

```
+    @Test
```

```
+    void test() {
```

```
+        assertTimeout(Duration.ofSeconds(1), () -> executeLongMethod());
```

```
+        assertThrows(RuntimeException.class, () -> throwException()); // returns Throwable
```

```
}
```

Extensions

- Replacing Runners and Rules
- Use as many as you want wherever you want them

```
-@RunWith(SpringJUnit4ClassRunner.class)
+@ExtendWith(SpringExtension.class)
+@ExtendWith(SeleniumExtension.class)
public class ExtensionTest {

    @Test
+    @ExtendWith(MockitoExtension.class)
+    void test(@Mock JediService jediService) {
```

JUnit Jupiter New Features

assertAll(...)






```
assertEquals(1, 4);  
assertEquals("Luke", "Rey");
```

org.opentest4j.AssertionFailedError:
Expected :1 Actual :4

```
assertAll(  
    () -> assertEquals(1, 4),  
    () -> assertEquals("Luke", "Rey")  
);
```

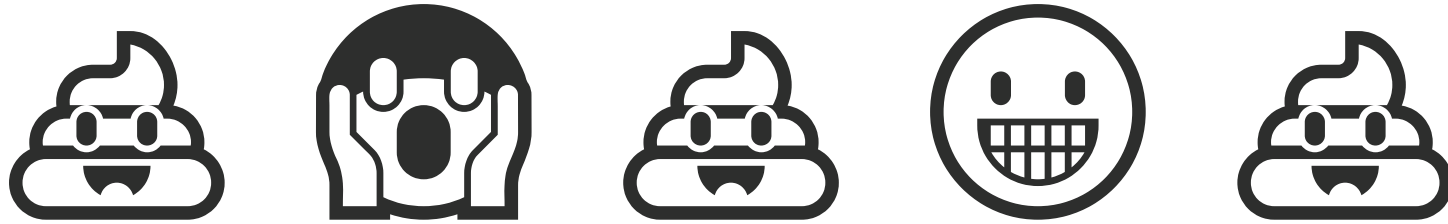
org.opentest4j.MultipleFailuresError:
Multiple Failures (2 failures)
expected: <1> but was: <4>
expected: <Luke> but was: <Rey>

Test Names

- Names default to test class or test method names
 - Characters limited based on Java syntax
- ▼  ItemAddedTest
 -  whenItemExistsCheckAmountMoreThanLimit()
 -  whenItemExistsIncrementAmount()
 -  whenItemDoesntExistCreateNewItem()
 -  whenItemDoesntExistSendNotification()

@DisplayName

- Custom display names for Test classes and methods
- Can contain spaces, special characters, and most importantly...



@Nested

```
@DisplayName("Item is added 😊")
class ItemAddedTest {

    @DisplayName("when item exists 🙌")
    @Nested
    class WhenItemExists {

        @DisplayName("amount++")
        @Test
        void incrementAmount() { }

        @DisplayName("check amount > limit")
        @Test
        void checkAmountMoreThanLimit() {
```

- ▼ ✓ Item is added 😊
- ▼ ✓ when item exists 🙌
 - ✓ amount++
 - ✓ check amount > limit
- ▼ ✓ when item doesn't exist
 - ✓ send notification
 - ✓ create item




@Tag and @RepeatedTest

- Test classes and methods can be tagged to filter test discovery and execution in build tools

```
@Tag("fast")
```

```
@RepeatedTest(3)
```

```
void testRepetitions() {}
```

▼  testRepetitions()
  repetition 1 of 3
  repetition 2 of 3
  repetition 3 of 3

@ParameterizedTest

```
@ParameterizedTest
```

```
@ValueSource(strings = {"KOTOR", "Jedi Knight", "Battlefront"})
```

```
void test(String videoGame) { }
```

- ▼ ✓ test(String)
 - ✓ [1] KOTOR
 - ✓ [2] Jedi Knight
 - ✓ [3] Battlefront

Parameterized with @MethodSource

```
@ParameterizedTest
```

```
@MethodSource("provideArgs")
```

```
void test(String text, int number, List list) {}
```

```
static Stream<Arguments> provideArgs() {
```

```
    return Stream.of(Arguments.of("Hello there!", 3, asList()),
```

```
        Arguments.of("General Kenobi!", 6, null));
```

```
}
```

Dynamic Tests (registered at runtime)

```
@TestFactory
Stream<DynamicTest> dynamicTestsFromStream() {
    return Stream.of("Jedi", "Sith")
        .map(text -> dynamicTest(text, () -> assertNotNull(text)));
}
```

<https://medium.com/@BillyKorando>

What else?

- Assumptions
- AssertJ and Hamcrest
- Extension API
- Meta-annotations
- Display Name Generators
- Test ordering
- Parallel test execution
- And more is coming

<https://junit.org/junit5/>


The new major version of the programmer-friendly
testing framework for Java

 User Guide

 Javadoc

 Code & Issues

 Q & A

 Support JUnit

Thank you!

Questions?