JUnit 5 Fundamentals

INTRODUCING JUNIT 5

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

What are the most important or popular Java frameworks?

“Never in the field of software development was so much owed by so many to so few lines of code.”

Martin Fowler

http://bit.ly/mfowlerquote

Overview

Introducing JUnit 5

Writing tests

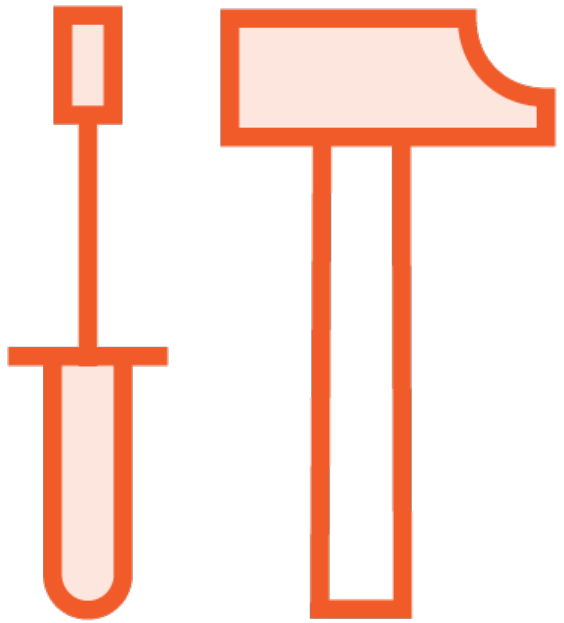
Creating dynamic and parameterized tests

Extending JUnit

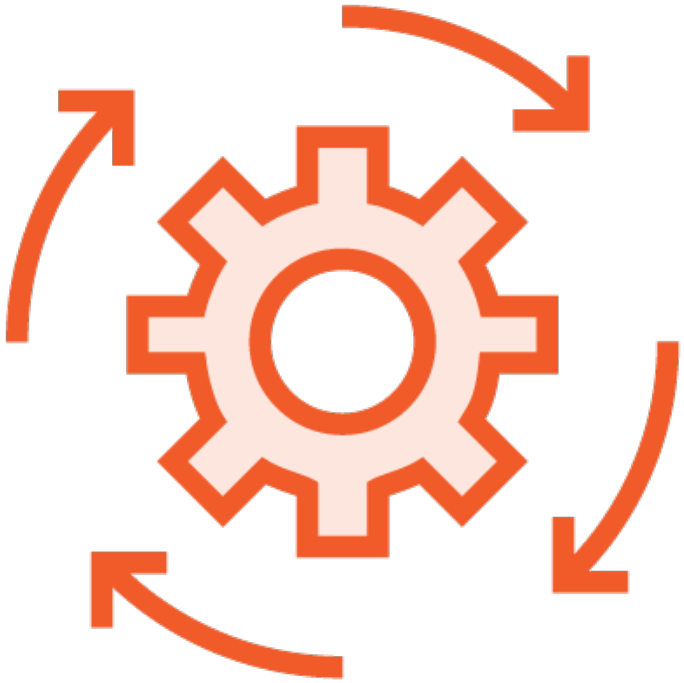
Integrating JUnit

Migrating from JUnit 4

JUnit Is...



Testing Is Feedback



“Programs should be written for people to read, and only incidentally for machines to execute.”

Structure and Interpretation of Computer Programs by Abelson and Sussman.

http://bit.ly/sicppreface

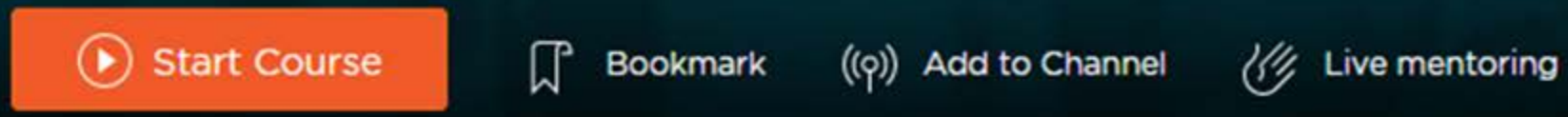
Tests Must Be...

- Easy to understand
- Easy to read
- Easy to modify

Introduction to Testing in Java

by Richard Warburton

The easiest and most pleasant way to get started with unit testing, JUnit, and Test Driven Development (TDD) that you could imagine.



http://bit.ly/introductiontestingjava

Test-Driven Development Practices in Java

by Mike Nolan

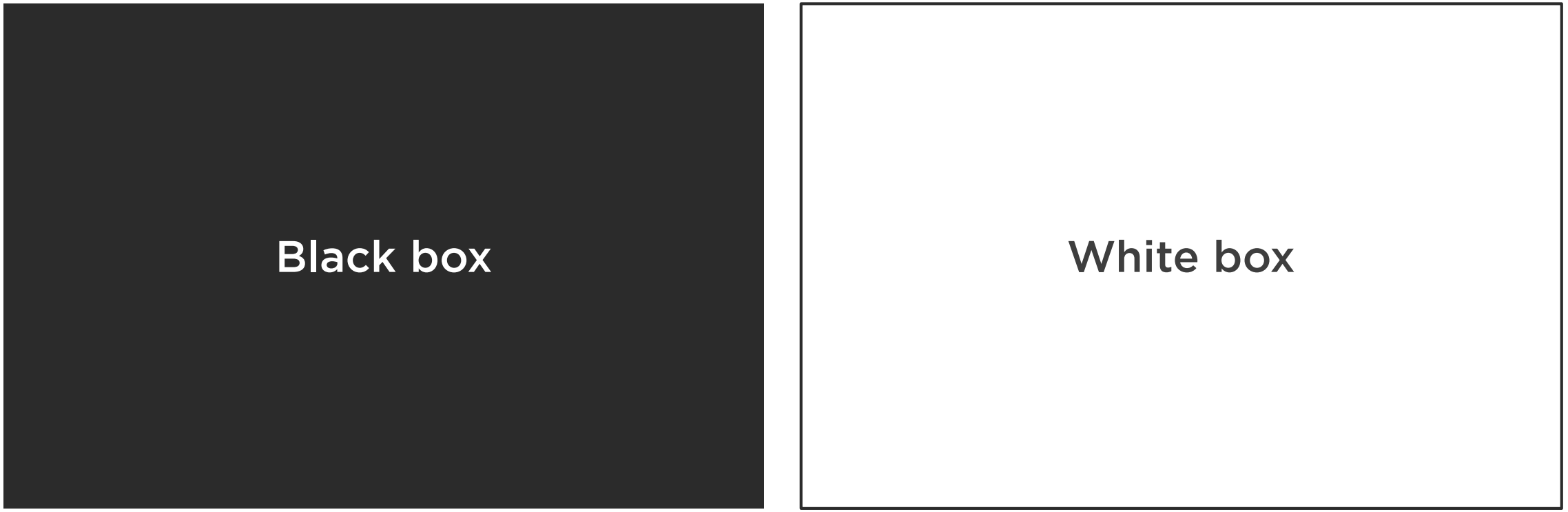
This course covers Test-Driven Development (TDD) practices, and tools supporting TDD on the Java Platform.

http://bit.ly/tddjavaps

Questions?

Types of Tests

According to the Knowledge of the System



Types of Testing

Acceptance

Integration

System

Load

Regression

Unit

Stress

Security

Recovery

Usability

Types of Testing

Unit

Unit Tests

Test a piece of code

```
public BigDecimal calculateCommission() {
   ...
}
```

```
class Sale {
   ...
   public BigDecimal calculateCommission() {
      ...
   }
   ...
}
```

A Good Unit Test Should Be

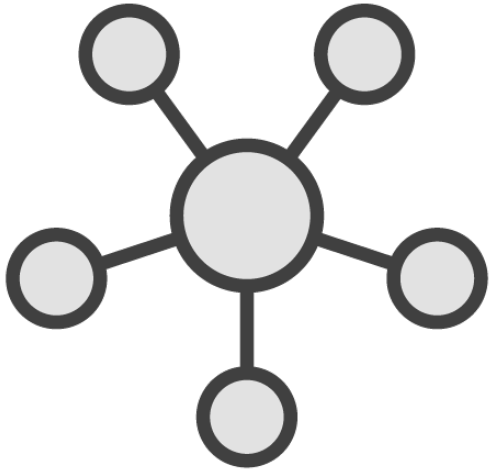
Automated

Repeatable

Fast

```
class Sale {
  private WebService commissionWS;
  ...
  public BigDecimal calculateCommission() {
    double commissionPercentage = commissionWS.getPercentage();
    ...
  }
  ...
}
```

Integration Tests



Test how components work together

```
void testCalculateCommission() {
  Sale sale = new Sale();
  sale.setCommissionWS(new FakeWebService());

  // Test calculateCommission() method ...
}
```

Why Unit Tests Are Important

Benefits

Reduce debugging time

Serve as documentation

Help to improve the design

Would It Be...

Hard?

Time consuming?

Well...

Yes, but not impossible

What If Our Unit Tests Were...

Easy to run

Fast

Complete

Up to date

Courage

What Do You Think Is Better?

Terrible designed system
Good suite of tests

Good designed system
Terrible suite of tests

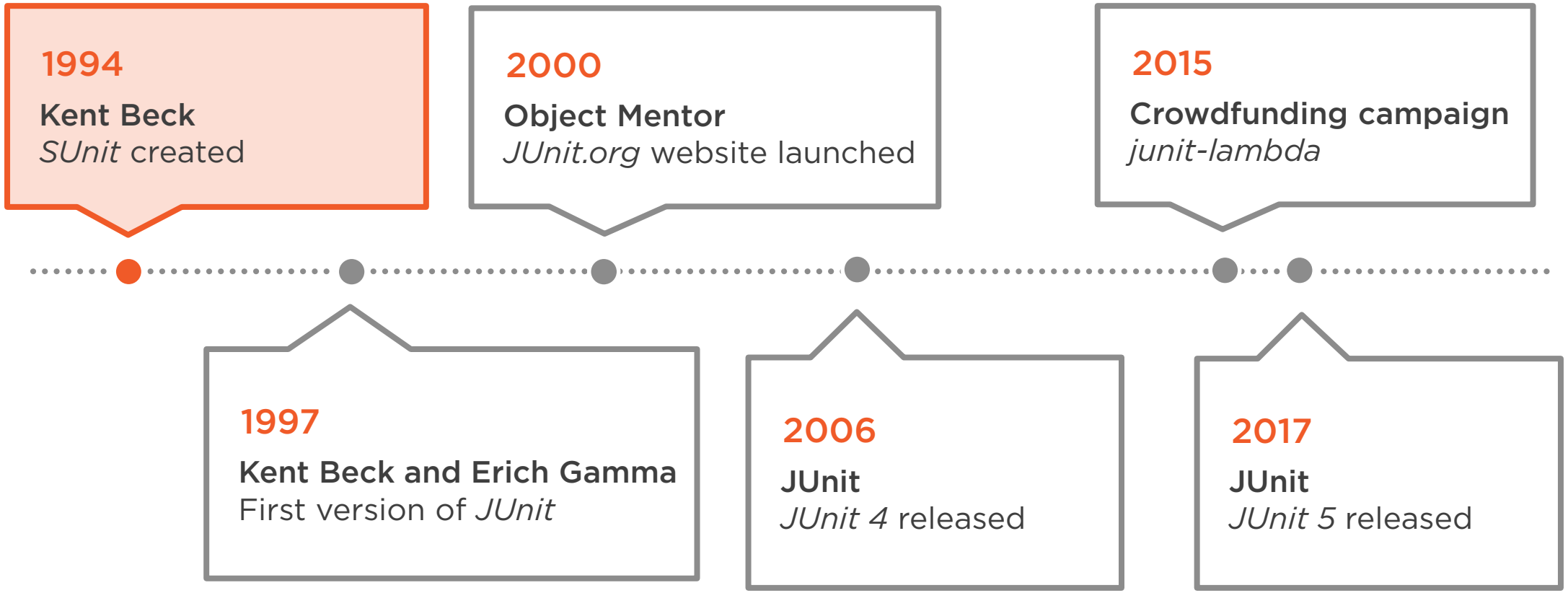
What's JUnit?

“JUnit is a simple, open source framework to write and run repeatable tests.”

junit.org

How JUnit Works

JUnit Timeline



JUnit 5

JUnit 5 Architecture

The Problem

junit.jar

A single JAR file

Used by everyone

No flexible API

Tools provided great support

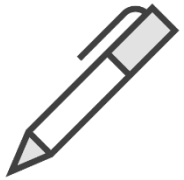
But locked development in

“The success of JUnit as
a platform prevents the
development of JUnit as a tool.”

Johannes Link

http://bit.ly/interviewjlink

Separation of Concerns



An API to write tests

A mechanism to discover and run tests

An API to run tests (for tools)

Design Goals

- Decouple tests
- Preference for extension points
- Support Java 8 features

JUnit 5

JUnit Platform

junit-platform-console

junit-platform-engine

junit-platform-launcher

junit-platform-runner

junit-platform-gradle-plugin

junit-platform-surefire-provider

JUnit Jupiter

junit-jupiter-api

junit-jupiter-engine

junit-jupiter-params

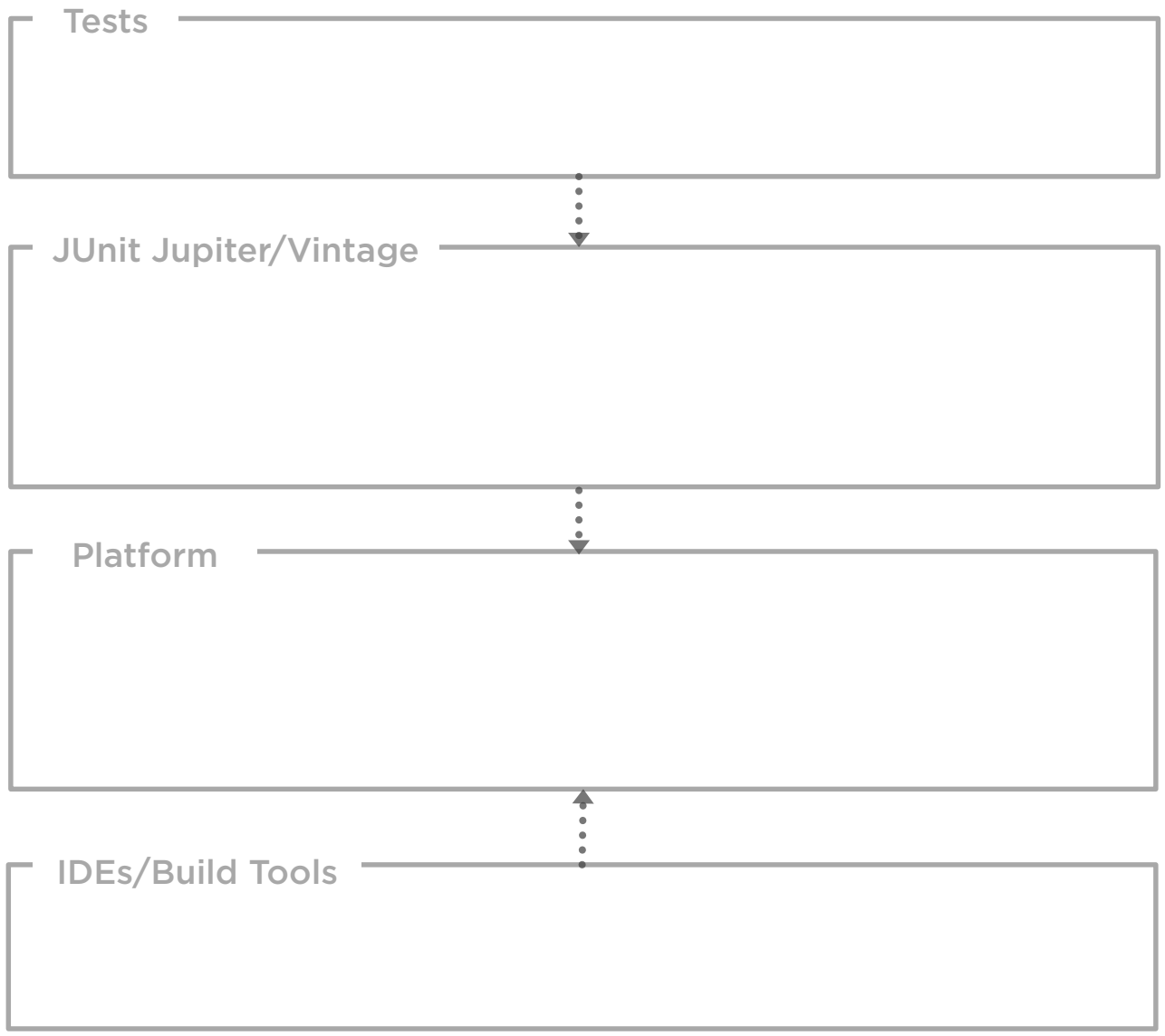
junit-jupiter-migrationsupport

JUnit Vintage

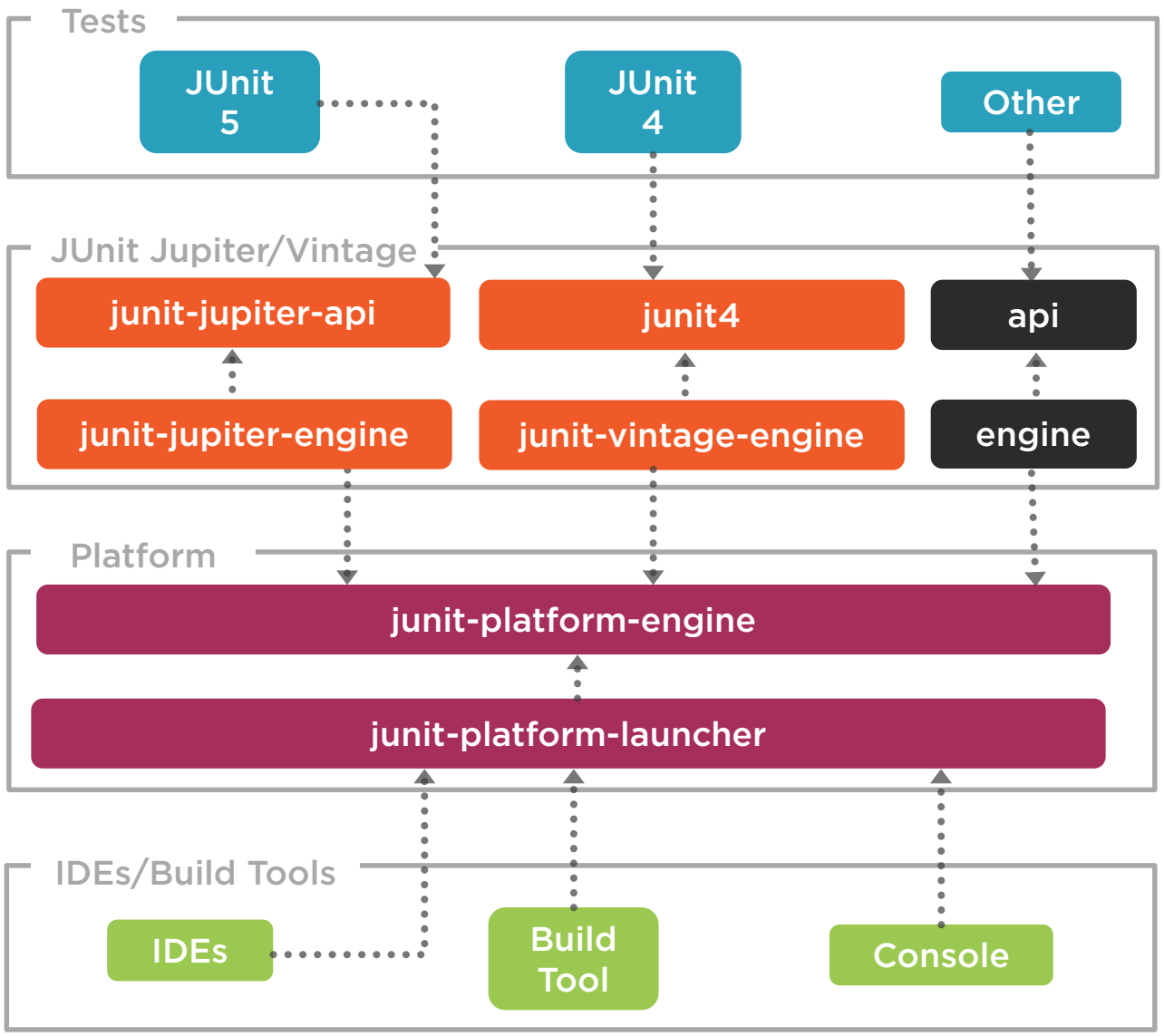
junit-vintage-engine

JUnit 3/4

Architecture

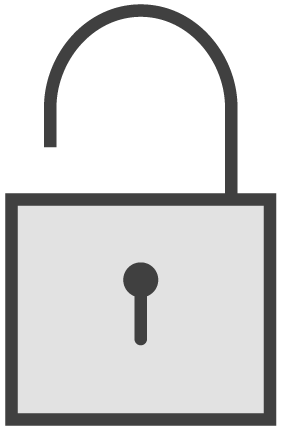


Architecture



The JUnit platform is available to everybody

Opening up the Platform Means

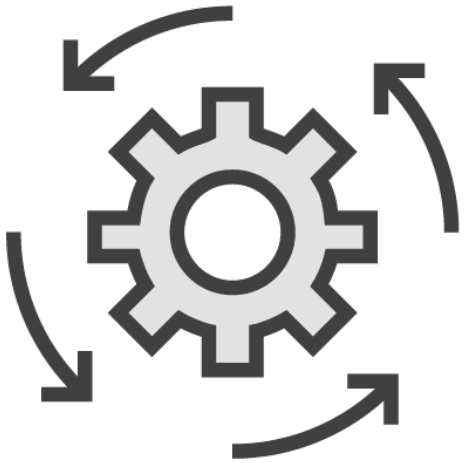


Full tool support

- Other frameworks
- New frameworks

IDEs and Build Tool Support

JUnit 5 Test API



Group ID: org.junit.jupiter

Artifact ID: junit-jupiter-api

Version: 5.0.1

First Test

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

class HelloWorldTest {
    @Test
    void firstTest() {
        System.out.println("First test");
    }
}
```

IDE Support

IntelliJ IDEA

Eclipse

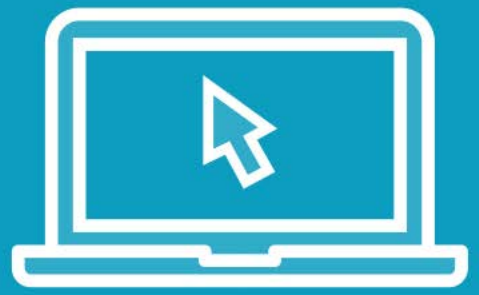
Build Tool Support

Gradle

Maven

Setting up JUnit with Gradle

Demo

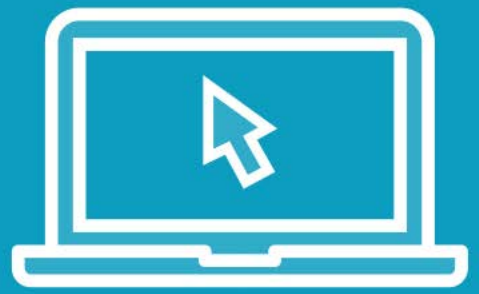


Setting up JUnit

- Gradle

Setting up JUnit with Maven

Demo



Setting up JUnit

- Maven

IDEs Without Support

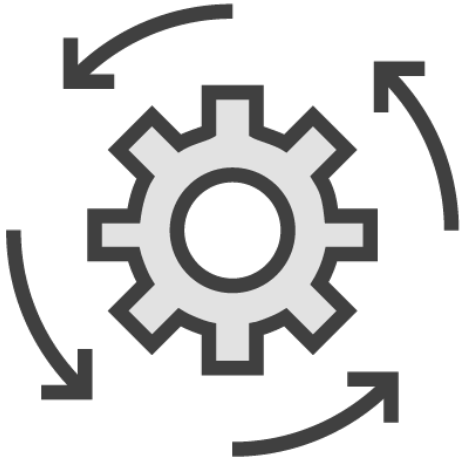
Two Options

Command line

JUnit 4 runner

- JUnitPlatform

JUnit 5 Platform Runner

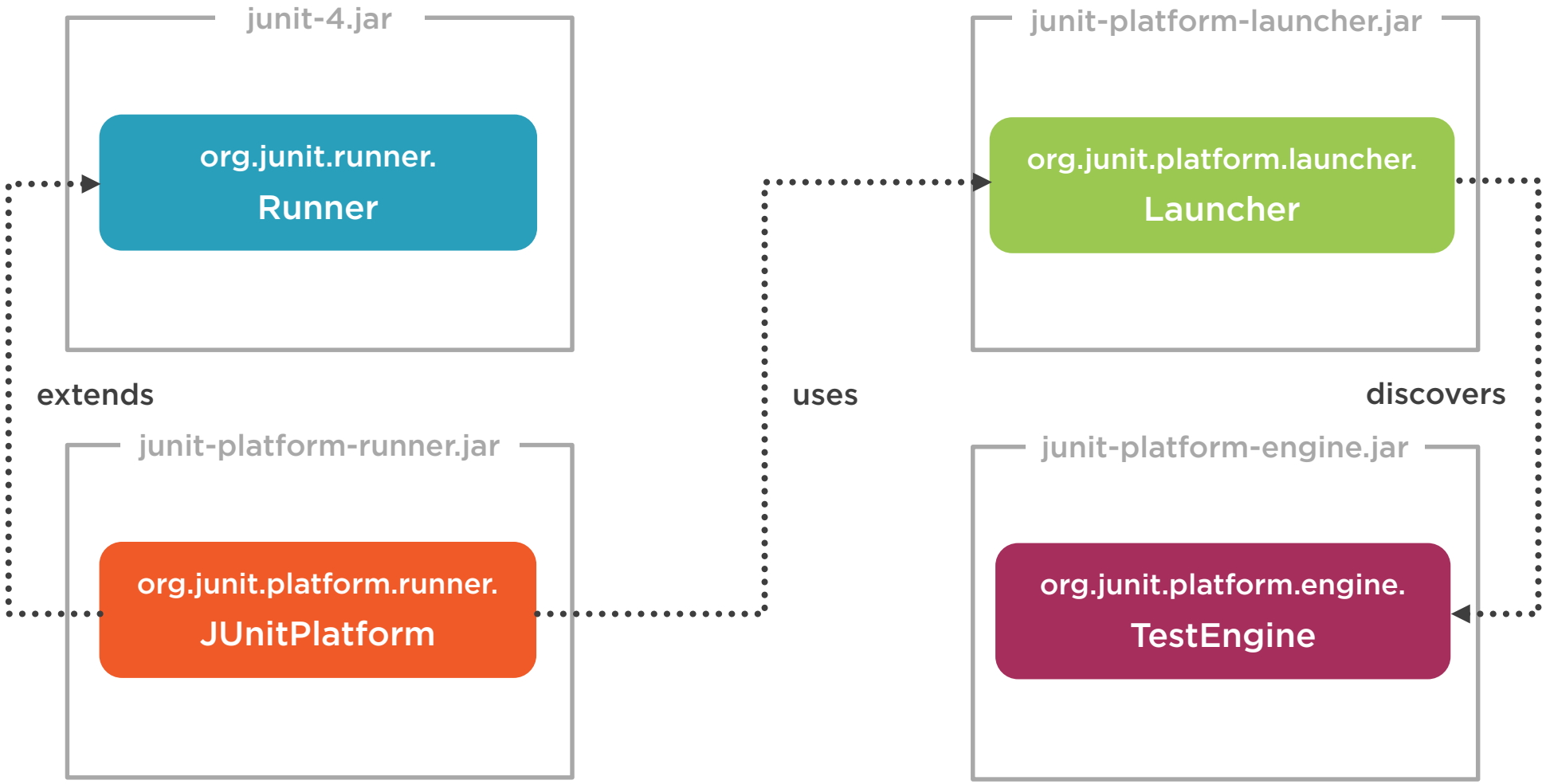


Group ID: org.junit.platform

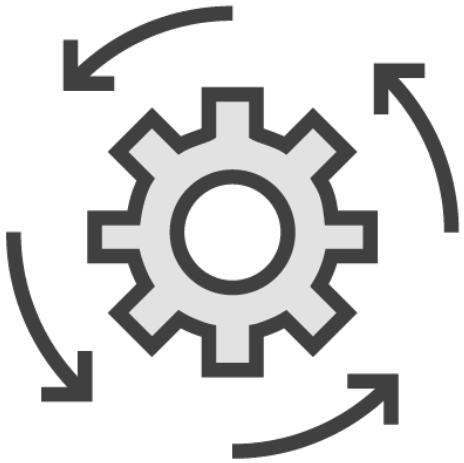
Artifact ID: junit-platform-runner

Version: 1.0.1

How JUnitPlatform Works



JUnit 5 Engine

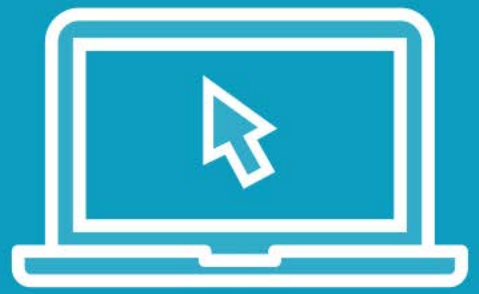


Group ID: org.junit.jupiter

Artifact ID: junit-jupiter-engine

Version: 5.0.1

Demo



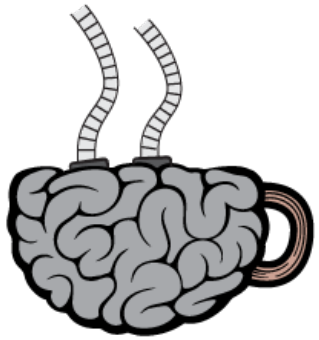
Setting up JUnit

- NetBeans

Create a Test Suite

```
import org.junit.platform.runner.JUnitPlatform;
import org.junit.platform.runner.SelectPackages;
import org.junit.runner.RunWith;

@RunWith(JUnitPlatform.class)
@SelectPackages({"my.package"})
public class TestWithJUnit5 { }
```

Course Scenario



WIRED BRAIN

COFFEE

Loyalty Program

- Every order generates points
- Rewards
  - Conversion
  - Discount
  - Gift

Summary

Types of tests

Why unit tests are important

What's JUnit?

- How it works
- History

JUnit 5 architecture

IDE and build tool support

- IntelliJ IDEA and Eclipse
- Gradle and Maven
- IDEs without support

Writing Tests

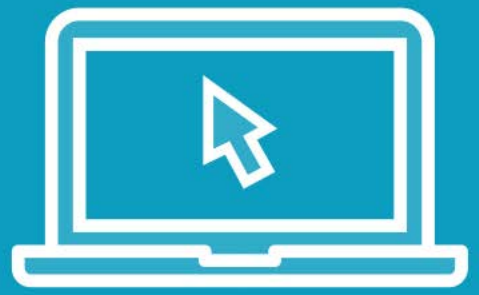
Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Overview

- Test structure
- Lifecycle methods
- Test hierarchies
- Assertions
- Disabling tests
- Assumptions
- Test interfaces and default methods
- Repeating tests

Demo

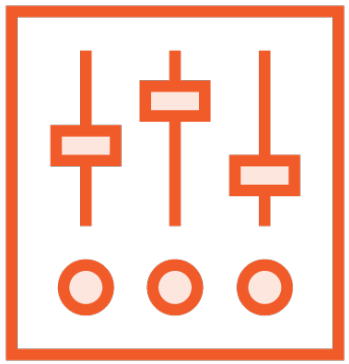


Write tests

How a good unit test is structured

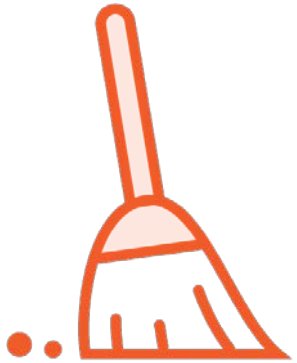
Four Phases of Every Test

Arrange



Act

Assert



Annihilation

Lifecycle Methods

Test Fixture

Everything we need to execute the test

Types for Managing Test Fixtures

Transient Fresh

Persistence Fresh

Persistence Shared

Lifecycle Annotations

Once per method	Once per class
@BeforeEach	@BeforeAll
@AfterEach	@AfterAll

Lifecycle Execution

Per method (default)

Per class

```
@TestInstance(TestInstance.Lifecycle.PER_METHOD)
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
```

Lifecycle Execution

Annotations

```
-Djunit.jupiter.testinstance.lifecycle.default=per_method
-Djunit.jupiter.testinstance.lifecycle.default=per_class
```

Lifecycle Execution

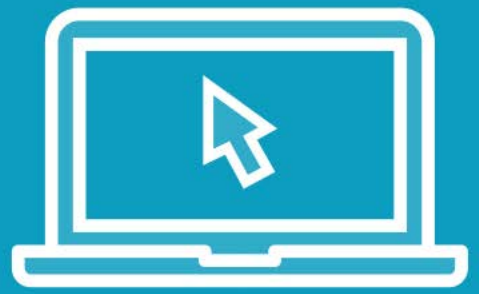
JVM options

```
junit.jupiter.testinstance.lifecycle.default=per_method
junit.jupiter.testinstance.lifecycle.default=per_class
```

Lifecycle Execution

junit-platform.properties

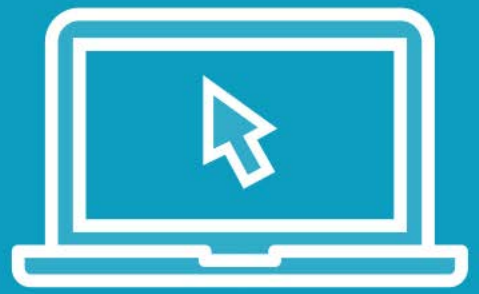
Demo



Lifecycle methods

- Annotations
- Per method/class

Demo



Test hierarchies

Behavior-Driven Development (BDD)

An application is specified and designed by describing how it should behave

BDD Naming Style

Test Phases	BDD
Arrange	Given
Act	When
Assert	Then

Nested Test Classes

Only non-static inner classes

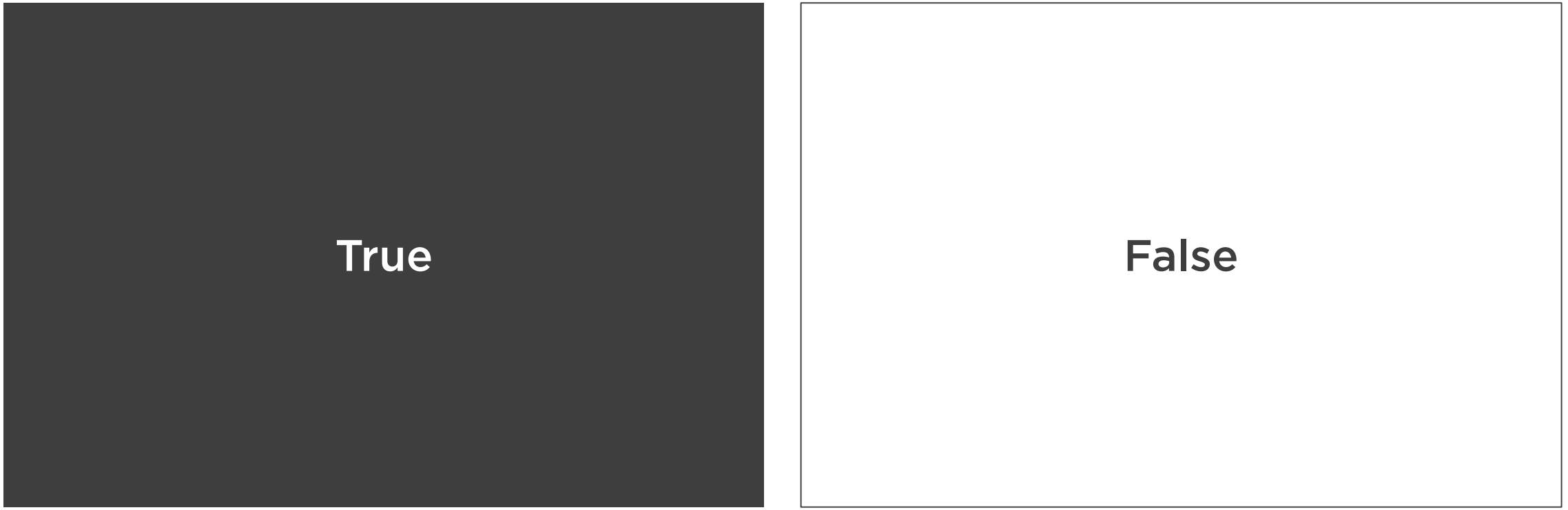
@BeforeAll and @AfterAll don't work by default

- Only with Lifecycle.PER_CLASS

Use them with @DisplayName

Assertions

Test Result



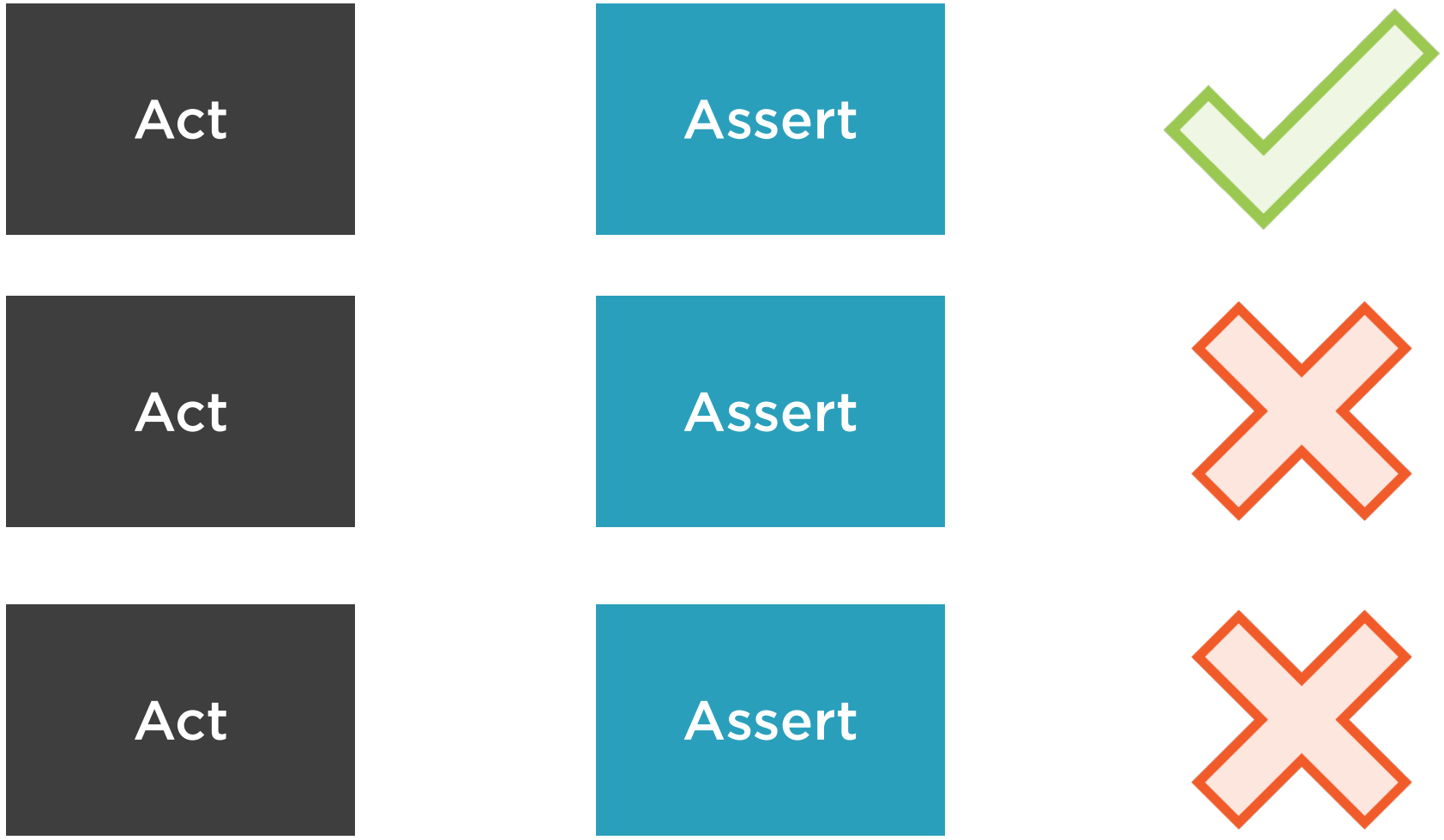
A Single Assertion?

```
void test() {
  ...
  assertTrue(...);
  assertNotNull(...);
  assertEquals(...);
}
```

A Single Assertion?

```
void test() {
  ...

  conditionOne && conditionTwo || conditionThree

}
```

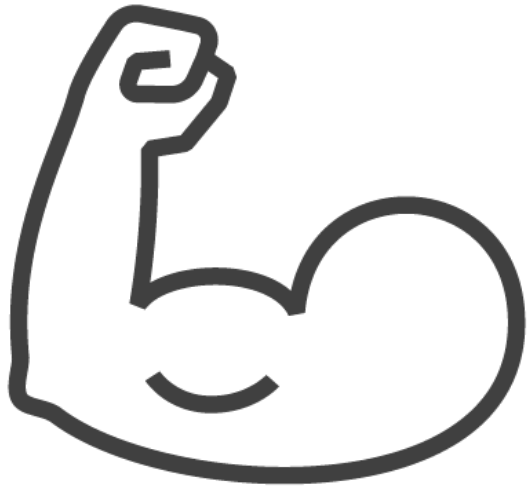
One Act/Assert Operations



JUnit Jupiter Assertions

- assertAll
- assertArrayEquals
- assertEquals
- assertFalse
- assertIterableEquals
- assertLinesMatch
- assertNotEquals
- assertNotNull
- assertNotSame
- assertNull
- assertSame
- assertThrows
- assertTimeout
- assertTimeoutPreemptively
- assertTrue
- fail

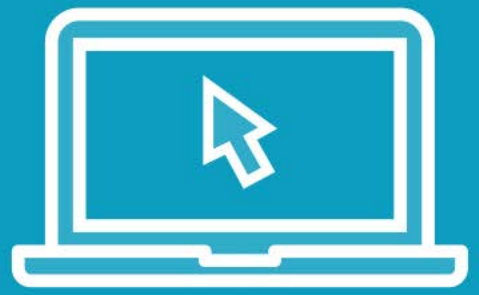
Need More Power?



External assertion libraries

- AssertJ
- Hamcrest

Demo



JUnit Jupiter assertions

- Error messages
- assertAll
- assertThrows
- assertTimeout
- assertTimeoutPreemptively

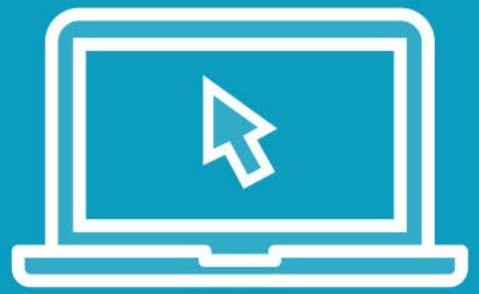
Disabling Tests

The Annotation

@Disabled

- Methods
- Classes

Demo



Disabling tests

Assumptions

Assumptions

Based on conditions

Don’t result in test failure like assertions

Abort the test

```
assumeTrue(boolean assumption)

assumeTrue(boolean assumption, String message)

assumeTrue(BooleanSupplier assumptionSupplier)

assumeTrue(boolean assumption, Supplier<String> message)

assumeTrue(BooleanSupplier assumptionSupplier, String message)

assumeTrue(BooleanSupplier assumptionSupplier, Supplier<String> message)
```

assumeTrue

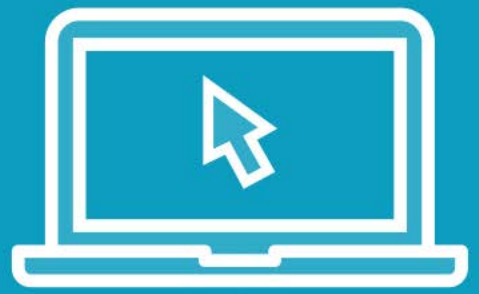
```
assumeFalse(boolean assumption)

assumeFalse(boolean assumption, String message)

assumeFalse(BooleanSupplier assumptionSupplier)

assumeFalse(boolean assumption, Supplier<String> message)

assumeFalse(BooleanSupplier assumptionSupplier, String message)

assumeFalse(BooleanSupplier assumptionSupplier, Supplier<String> message)
```

assumeFalse

```
assumingThat(boolean assumption, Executable message)

assumingThat(boolean assumption, Executable executable)

assumingThat(BooleanSupplier assumptionSupplier, Executable executable)
```

assumingThat

Demo



Assumptions

Test Interfaces and Default Methods

What to Include in Interfaces?

@Test

@BeforeEach

@AfterEach

What to Include in Interfaces?

@RepeatedTest

@ParameterizedTest

@TestFactory

@TestTemplate

@ExtendWith

@Tag

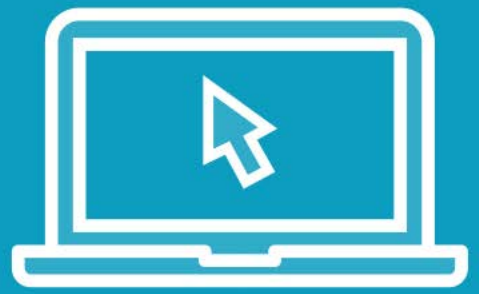
What to Include in Interfaces?

@BeforeAll

@AfterAll

@TestInstance(Lifecycle.PER_CLASS)

Demo



Test interfaces and default methods

- Extract methods

Repeating Tests

@RepeatedTest

Repeat a test

Fixed number of repetitions

Full support of lifecycle

{displayName}

{currentRepetition}

{totalRepetitions}

Custom Display Name

Placeholders

```
RepeatedTest.LONG_DISPLAY_NAME
{displayName} :: repetition {currentRepetition} of {totalRepetitions}
Ex: My Test :: repetition 1 of 10

RepeatedTest.SHORT_DISPLAY_NAME
repetition {currentRepetition} of {totalRepetitions}
Ex: repetition 1 of 10
```

Custom Display Name

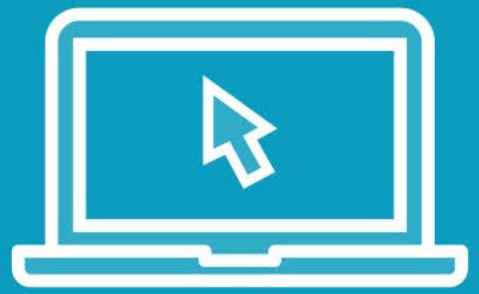
Predefined formats

```
int getCurrentRepetition();
int getTotalRepetitions();
```

RepetitionInfo Interface

@RepeatedTest, @BeforeEach, and @AfterEach

Demo



Repeating tests

- @RepeatedTest
- Custom display name
- RepetitionInfo interface

Summary

- Test structure
- Lifecycle methods
- Test hierarchies
- Assertions
- Disabling tests
- Assumptions
- Test interfaces and default methods
- Repeating tests

Creating Dynamic and Parameterized Tests

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Overview

Dynamic Tests

- @TestFactory
- Sources

Parameterized Tests

- Setup
- Argument sources
- Argument conversion

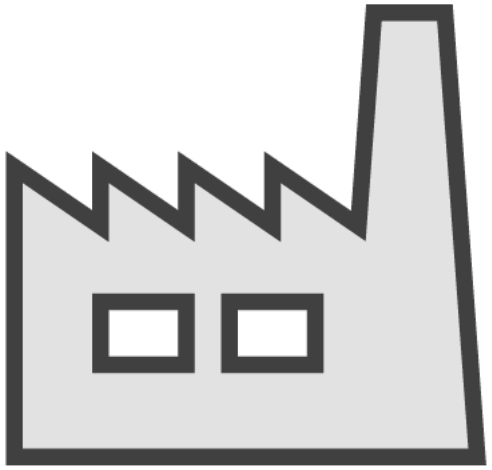
Dynamic Tests

```
@Test
void testRewardProgram() {
  // ...
  assertEquals(type, reward.getType());
}
```

```java
@Test
void testRewardProgram() {
  // ...
  List<TestData> list = createTestData();
  for(TestData data : list) {
    // ...
    assertEquals(type, reward.getType());
  }
}
```

```java
@RepeatedTest(10)
void testRewardProgram(RepetitionInfo repetitionInfo) {

  // ...

  TestData data = list.get(
       repetitionInfo.getCurrentRepetition() - 1
  );

  // ...

  assertEquals(type, reward.getType());

}
```

Dynamic Tests



@TestFactory

- Factory of tests

No private or static methods

Experimental API

Sources

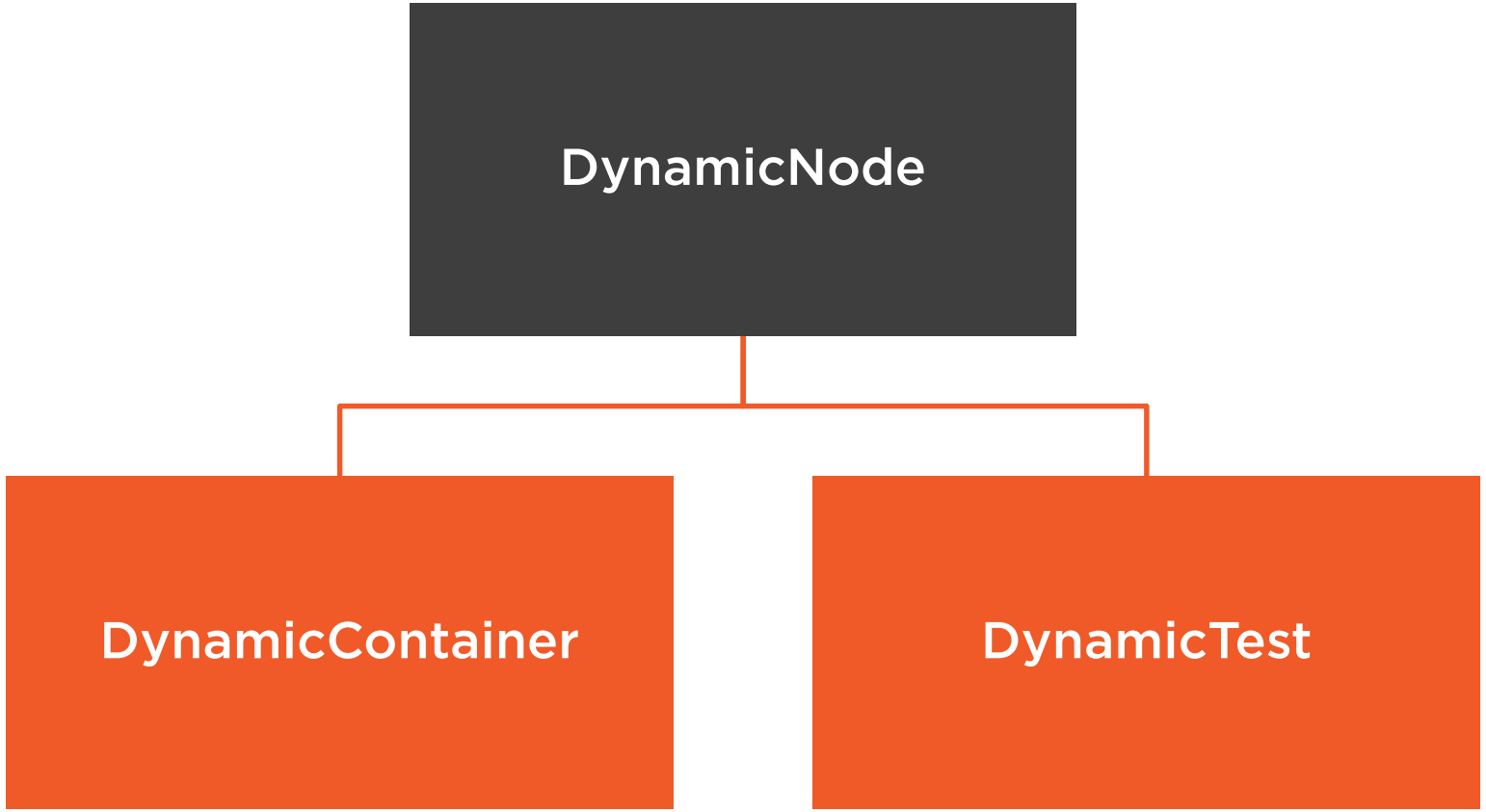
Collection

Iterable

Iterator

Stream

DynamicNode Subclasses



DynamicContainer

Display name

Iterable (or Stream)

- DynamicNodes
  - DynamicContainer
  - DynamicTest

DynamicTest

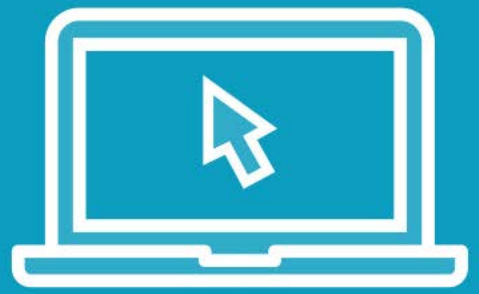
Display name

Executable

- Functional interface

@BeforeEach and
@AfterEach methods
are not executed for
each dynamic test

Demo

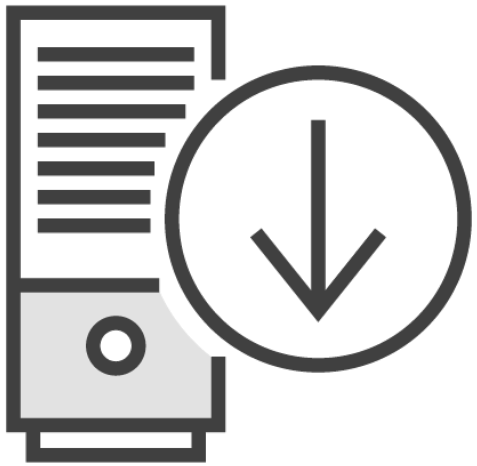


Dynamic Tests

- Lifecycle
- DynamicTest
- DynamicContainer

Parameterized Tests

Parameterized Tests

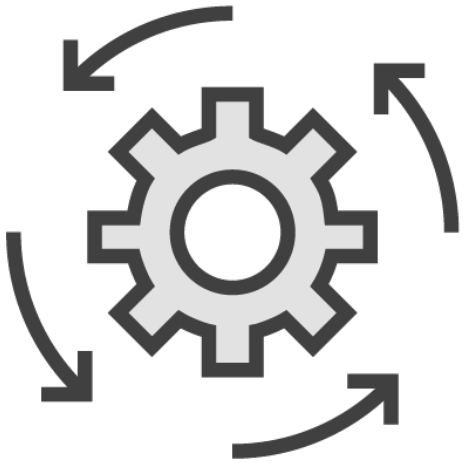


@ParameterizedTest

- Like regular test
- Declare at least one source

Experimental API

Dependency



Group ID: org.junit.jupiter

Artifact ID: junit-jupiter-params

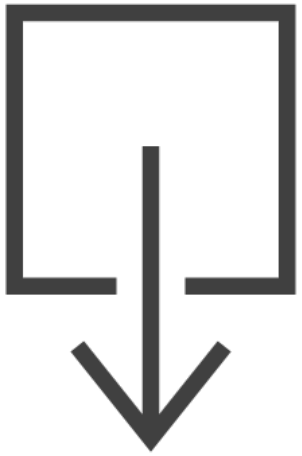
Version: 5.0.1

```
{index}
{arguments}
{0}, {1}, ...
```

Custom Display Name

@ParameterizedTest placeholders

Parameter Injection



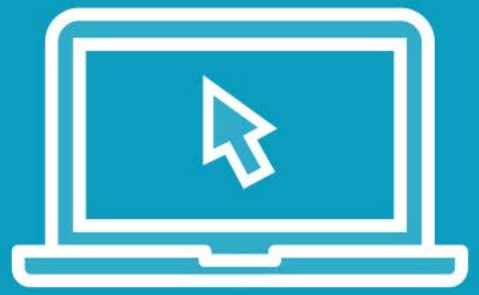
@ParameterizedTest parameters

- Can’t be injected into lifecycle methods

Test information parameters

- TestInfo
- TestReporter
- After parameters injected to the test

Demo

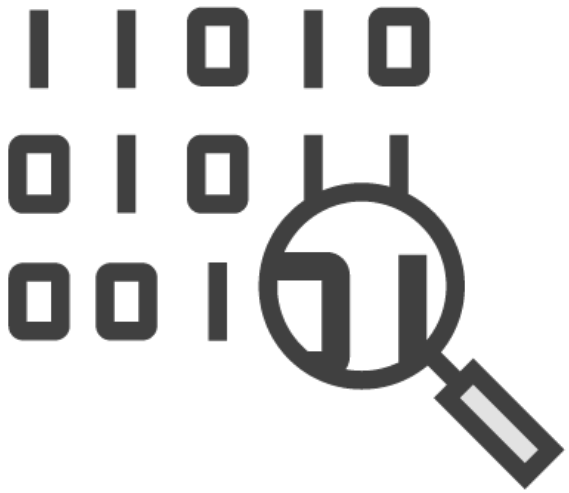


Parameterized Tests

- Setup
- Lifecycle
- Custom display name
- Inject test information parameters

Argument Sources

Sources Rules



At least one source

Provide values for all parameters

One execution for each group of arguments

@ValueSource

Arrays of type:

- String
- int
- long
- double

For single parameter methods

@EnumSource

Values of an enum

Optional parameters:

- names
- mode

For single parameter methods

@MethodSource

Refer one or more methods

For tests with a single parameter:

- Return a Stream of parameter type
- Return a Stream of primitive types

For tests with multiple parameters:

- Return a Stream, Iterable, Iterator, or array of type Arguments

The methods used:

- Must be static
  - Unless you're using @TestInstance(Lifecycle.PER_CLASS)
- And optionally private

@CsvSource

Comma-separated String literals

Parameter:

- Delimiter

Uses a single quote (') as quote character

@CsvFileSource

CSV files from classpath

Parameters:

- Encoding
- Line separator
- Delimiter

Each line results in one invocation

Uses a double quote (") as quote character

@ArgumentsSource

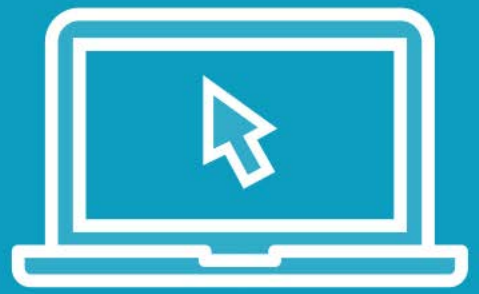
For custom sources

Use an ArgumentsProvider implementation

```
interface ArgumentsProvider {
    Stream<? extends Arguments>
        provideArguments(ExtensionContext context)
                throws Exception;
}
```

@ArgumentsSource

Demo



Argument Sources

Argument Conversion

Sources

@ValueSource

@CsvSource

@CsvFileSource

Implicit Conversion

String to:

- Primitive values
- Enum
- java.time classes

```
interface ArgumentConverter {
    Object convert(Object source, ParameterContext context)
            throws ArgumentConversionException;
}
```

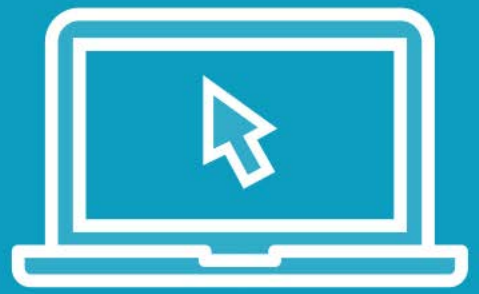
Custom Converter

@ConvertWith

```
abstract class SimpleArgumentConverter implements ArgumentConverter {
    protected abstract Object convert(Object source,
                                      Class<?> targetType)
                               throws ArgumentConversionException;
    // ...
}
```

SimpleArgumentConverter

Demo



Custom Converter

Summary

Dynamic Tests

- @TestFactory
- No support for lifecycle methods
- Collection, Iterable, Iterator, Stream
- DynamicContainer and DynamicTest

Parameterized Tests

- Support for lifecycle methods
- Dependency junit-jupiter-params
- Multiple sources
- Custom display name
- Custom converters

Extending JUnit

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Overview

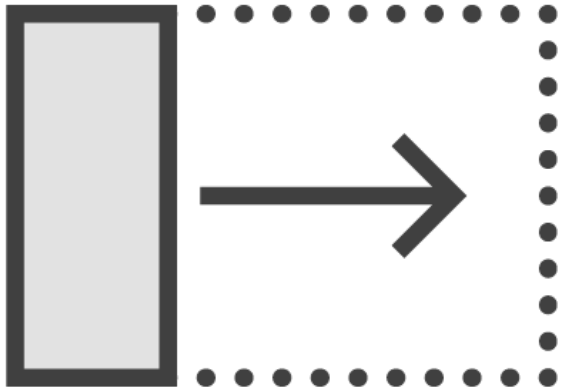
- Extension points
- Parameter resolution
- Meta-annotations
- Keeping state
- Sample extensions

Extension Points

“Prefer extension points over features.”

JUnit design principles

Extension Points



Marker interface ⟶ Extension

One extension point ⟶ one interface

Called by the JUnit Jupiter engine

General Purpose

TestInstancePostProcessor

ParameterResolver

TestExecutionExceptionHandler

Conditional

ExecutionCondition

Lifecycle Callbacks

BeforeAllCallback / AfterAllCallback

BeforeEachCallback / AfterEachCallback

BeforeTestExecutionCallback / AfterTestExecutionCallback

Extension Registration

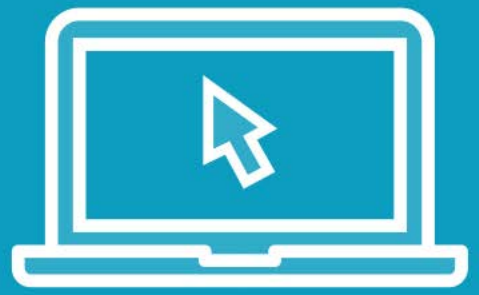
Explicit

- @ExtendWith

Global

- java.util.ServiceLoader mechanism
  - /META-INF/services
  - org.junit.jupiter.api.extension.Extension
  - junit.jupiter.extensions.autodetection.enabled

Demo

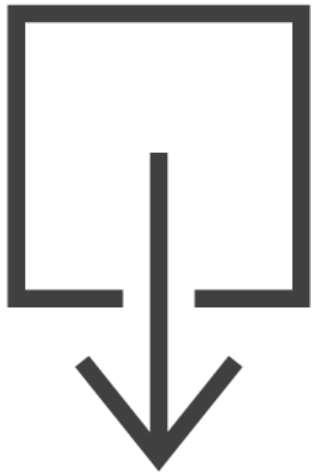


Implementing an extension

- Lifecycle callbacks

Parameter Injection

Parameter Injection



Test information parameters

- RepetitionInfo
- TestInfo
- TestReporter

ParameterResolver

- RepetitionInfoParameterResolver
- TestInfoParameterResolver
- TestReporterParameterResolver

```
boolean supportsParameter(ParameterContext parameterContext,
                          ExtensionContext extensionContext)
        throws ParameterResolutionException


Object resolveParameter(ParameterContext parameterContext,
                        ExtensionContext extensionContext)
        throws ParameterResolutionException
```

ParameterResolver

Methods That Can Have Parameters Injected

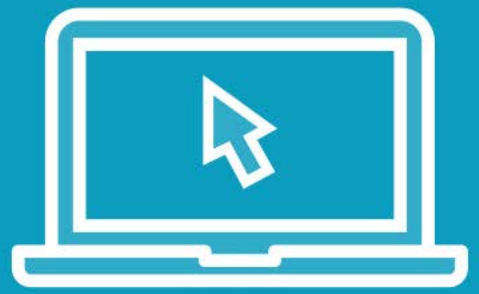
@Test

@TestFactory

Lifecycle Methods

Constructors

Demo



ParameterResolver extension

Meta-annotations

Meta-annotations

```
@Test
void testRewardProgram() {
   ...
}
```

Meta-annotations

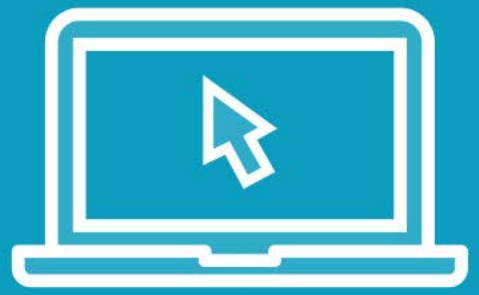
```
@TestWithErrorHandler
void testRewardProgram() {
   ...
}
```

Meta-annotations

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Test
@ExtendWith({ExceptionHandler.class})
public @interface TestWithErrorHandler() { }
```

```
@TestWithErrorHandler
void testRewardProgram() {
  // ...
}
```

Demo



Meta-annotations

Keeping State

Extensions have to be stateless

Storing State

Store

- Key-value structure

Namespace

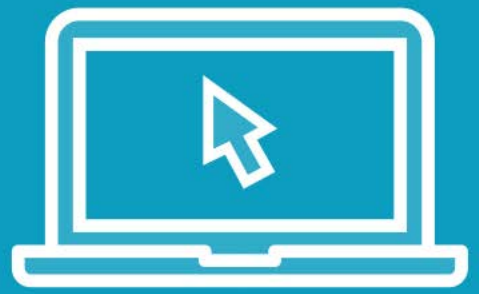
Hierarchy

- MethodExtensionContext
- ClassExtensionContext
- JupiterEngineExtensionContext

```
Object get(Object key)
<K,V> Object getOrComputeIfAbsent(K key, Function<K,V> defaultCreator)
void put(Object key, Object value)
Object remove(Object key)
```

ExtensionContext.Store

Demo



Implementing an extension

- Store
- ExecutionCondition

Sample Extensions

Summary

Extension points

- One extension point ⟶ one interface
- Register explicitly or globally
- Lifecycle callbacks

Parameter resolution

- ParameterResolver

Meta-annotations

Keeping state

- Extensions are stateless
- Store and namespaces
- Hierarchical contexts

Sample extensions

Integrating JUnit 5

Esteban Herrera
JAVA ARCHITECT

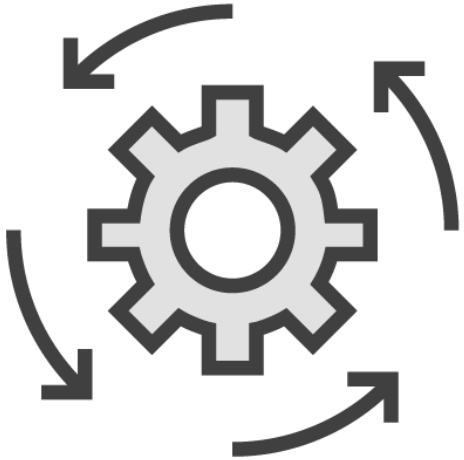
@eh3rrera www.eherrera.net

Overview

- Running tests from the console
- Running tests with Gradle
- Running tests with Maven
- Include/exclude tests with tags
- Code coverage

Running Tests from the Console

JUnit 5 Console Launcher



Group ID: org.junit.platform

Artifact ID: junit-platform-console-standalone

Version: 1.0.1

http://bit.ly/mavenjunitconsole

```
java -jar junit-platform-console-standalone-1.0.1.jar \
    --cp ${PATH_TO_COMPILED_CLASSES} \
    --scan-classpath
```

Run Tests

Demo

Running tests from the console

Running Tests with Gradle

Enabling the JUnit Gradle Plugin

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'org.junit.platform:junit-platform-gradle-plugin:1.0.1'
    }
}

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

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {




}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  logManager 'org.apache.logging.log4j.jul.LogManager’




}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  logManager 'org.apache.logging.log4j.jul.LogManager'
  reportsDir file('build/test-results/junit-platform')




}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  logManager 'org.apache.logging.log4j.jul.LogManager'
  reportsDir file('build/test-results/junit-platform')
  enableStandardTestTask true




}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  logManager 'org.apache.logging.log4j.jul.LogManager’
  reportsDir file('build/test-results/junit-platform’)
  enableStandardTestTask true
  configurationParameter 'junit.jupiter.testinstance.lifecycle.default','per_class'




}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  logManager 'org.apache.logging.log4j.jul.LogManager’
  reportsDir file('build/test-results/junit-platform’)
  enableStandardTestTask true
  configurationParameter 'junit.jupiter.testinstance.lifecycle.default','per_class’
  configurationParameters([
      'junit.jupiter.testinstance.lifecycle.default': 'per_class’,
      'junit.jupiter.conditions.deactivate': '*'
  ])

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...
  selectors {









  }
}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...
  selectors {
    uri 'file:///data.txt'
    file 'data.json'
    directory 'resources/data'
    aPackage 'com.wbc.rewards'
    aClass 'com.wbc.rewards.PointsTest'
    method 'com.wbc.sale.SaleTest#test(java.lang.String[])'
    resource '/com/wbc/program.properties'
  }
}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...
  selectors {
    uris 'file:///data.txt', 'file:///data2.txt'
    files 'data.json', 'data2.json'
    directories 'resources/data', 'resources/data2'
    packages 'com.wbc.rewards', 'com.wbc.customer'
    classes 'com.wbc.rewards.PointsTest', 'com.wbc.rewards.PointsTest'
    methods 'com.wbc.sale.SaleTest#test(java.lang.String[])', 'com.wbc.Main#main'
    resources '/com/wbc/program.properties', '/com/wbc/points.properties'
  }
}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {




  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    engines {
      include 'junit-jupiter', 'junit-vintage'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    engines {
      exclude 'junit-jupiter', 'junit-vintage'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    tags {
      include 'test-ci', 'integration'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    tags {
      exclude 'test-ci', 'integration'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    packages {
      include 'com.wbc.rewards', 'com.wbc.customer'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    packages {
      exclude 'com.wbc.rewards', 'com.wbc.customer'
    }

  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    includeClassNamePattern '.*Tests'




  }

}
```

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    includeClassNamePattern '.*Tests'
    includeClassNamePatterns '.*Test', '.*Case'


  }

}
```

Demo

Running tests with Gradle

Running Tests with Maven

Configuring the Maven Surefire Plugin

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
            <dependencies>
                <dependency>
                    <groupId>org.junit.platform</groupId>
                    <artifactId>junit-platform-surefire-provider</artifactId>
                    <version>1.0.1</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

Configuring the Maven Surefire Plugin

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
            <dependencies>
              ...
            </dependencies>




        </plugin>
    </plugins>
</build>
```

Configuring the Maven Surefire Plugin

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.19.1</version>
            <dependencies>
              ...
            </dependencies>
            <configuration>



            </configuration>
        </plugin>
    </plugins>
</build>
```

Configuring the Maven Surefire Plugin

```
<configuration>



</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <includes>
    <include>**/Test*.java</include>
    <include>**/*Test.java</include>
  </includes>
</configuration>
```

Default Inclusions

```
"**/Test*.java"
"**/*Test.java"
"**/*Tests.java"
"**/*TestCase.java"
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <includes>
    <include>**/Test*.java</include>
    <include>**/*Test.java</include>
  </includes>
</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <excludes>
    <exclude>**/Test*.java</exclude>
    <exclude>**/*Test.java</exclude>
  </excludes>
</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <properties>
    <includeTags>integration</includeTags>
    <excludeTags>slow, test-ci</excludeTags>
  </properties>
</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <properties>


  </properties>
</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <properties>
    <configurationParameters>


    </configurationParameters>
  </properties>
</configuration>
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <properties>
    <configurationParameters>
    junit.jupiter.conditions.deactivate=*
    junit.jupiter.testinstance.lifecycle.default=per_class
    </configurationParameters>
  </properties>
</configuration>
```

Demo

Running tests with Maven

Include/exclude Tests with Tags

Tags

```
import org.junit.jupiter.api.Tag;
...
@Tag("v1")
@Tag("reward")
class TestRewards {
    @Test
    @Tag("fast")
    void testRewardProgram() { /* ... */ }
}
```

Interfaces

```
@Tag("error")
@ExtendWith({ExceptionHandler.class})
public interface ErrorHandler {
   ...
}
```

```
class TestRewardProgram implements ErrorHandler {
   ...
}
```

Meta-annotations

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Test
@Tag("error")
@ExtendWith({ExceptionHandler.class})
public @interface TestWithErrorHandler() { }
```

```
@TestWithErrorHandler
void testRewardProgram() {
   ...
}
```

Syntax Rules for Tags

A tag must not be null or blank

A *trimmed* tag must not contain:

- Whitespace
- ISO control characters
- Any of the following *reserved characters*: , ( ) & | !

Configuring the JUnit Gradle Plugin

```
...
junitPlatform {
  ...

  filters {
    tags {
      include 'slow'
      exclude 'test-ci', 'integration'
    }

  }

}
```

Configuring the Maven Surefire Plugin

```
<configuration>
  <properties>
    <includeTags>integration</includeTags>
    <excludeTags>slow, test-ci</excludeTags>
  </properties>
</configuration>
```

Demo

Tags

- Gradle
- Maven

Code Coverage

Code coverage

Measure used to describe the degree to which the code of your program is covered by your tests

Demo

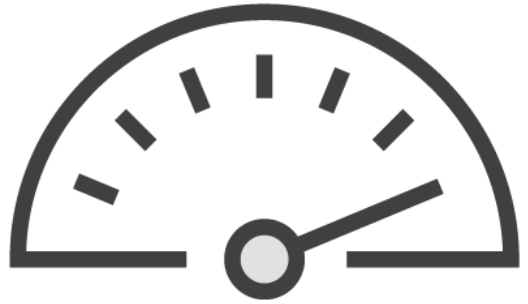
Code Coverage

- JaCoCo
  - Maven
  - Gradle

What Percentage Should Be Our Goal?

60%?

80%?



100%?

What Percentage Should Be Our Goal?

100%

100% code coverage
doesn’t mean your code
works perfectly

Just Remember

100% must be the goal

But it only means how much code is tested

It doesn't necessarily mean it's well-tested

Mutation Testing

Small modifications

AND instead of OR

If the unit test fails

It works fine

If the test succeeds

It is not testing the right thing

The result is the percentage of failing tests

Summary

- Running tests from the console
- Running tests with Gradle
- Running tests with Maven
- Include/exclude tests with tags
- Code coverage

Migrating from JUnit 4

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Overview

Differences between JUnit 4 and 5

Running JUnit 4 tests in JUnit 5

Rule support in JUnit 5

Differences Between JUnit 4 and 5

org.junit.jupiter

All classes and annotations are now under this package

```
public class Test {
   @Test
   public void myTest() {
      ...
   }
}
```

Public Classes/Methods

```
class Test {
  @Test
  void myTest() {
    ...
  }
}
```

No More Public Classes/Methods

```
@BeforeClass
public static void setUpOnce() { /* ... */ }
@Before
public void setUp() { /* ... */ }
@After
public void teardown() { /* ... */ }
@AfterClass
public static void tearDownOnce() { /* ... */ }
```

Lifecycle Annotations

```
@BeforeClass
static void setUpOnce() { /* ... */ }
@Before
void setUp() { /* ... */ }
@After
void teardown() { /* ... */ }
@AfterClass
static void tearDownOnce() { /* ... */ }
```

Lifecycle Annotations

```
@BeforeAll
static void setUpOnce() { /* ... */ }
@BeforeEach
void setUp() { /* ... */ }
@AfterEach
void teardown() { /* ... */ }
@AfterAll
static void tearDownOnce() { /* ... */ }
```

Lifecycle Annotations

```
@BeforeAll
void setUpOnce() { /* ... */ }
@BeforeEach
void setUp() { /* ... */ }
@AfterEach
void teardown() { /* ... */ }
@AfterAll
void tearDownOnce() { /* ... */ }
```

Lifecycle Annotations

@TestInstance(Lifecycle.PER_CLASS)

```
@Test
@Ignore
public void myTest() {
   ...
}
```

Ignore Annotation

```
@Test
@Disable
void myTest() {
   ...
}
```

Disable Annotation

```
@Test
@Category({ Version1.class, Important.class })
void myTest() {
   ...
}
```

Categories

```
@Test
@Tag("Version1")
@Tag("Important")
void myTest() {
   ...
}
```

Tags

`org.junit.Assert`

Assertions Package

`org.junit.jupiter.api.Assertions`

Assertions Package

```
assertEquals("Error message", expected, actual)
```

Parameter Order

```
assertEquals(expected, actual, "Error message")
```

Parameter Order

```
assertEquals(expected, actual, () -> "Error message")
```

Lazy Strings

`org.junit.Assert.assertThat`

assertThat

~~`org.junit.Assert.assertThat`~~

assertThat Is Gone

Use it directly from Hamcrest

```
@Rule
ErrorCollector collector = new ErrorCollector();
@Test
public void myTest() {
  collector.checkThat("aa", equalTo("a"));
  collector.checkThat(1, equalTo(11));
}
```

Continue Test Execution After Failure

```
@Test
public void myTest() {
  collector.checkThat("aa", equalTo("a"));
  collector.checkThat(1, equalTo(11));
}
```

Continue Test Execution After Failure

```
@Test
void myTest() {
  collector.checkThat("aa", equalTo("a"));
  collector.checkThat(1, equalTo(11));
}
```

Continue Test Execution After Failure

```
@Test
void myTest() {
  assertAll(
  );
  collector.checkThat("aa", equalTo("a"));
  collector.checkThat(1, equalTo(11));
}
```

Continue Test Execution After Failure

```
@Test
void myTest() {
  assertAll(
    () -> assertThat("aa", equalTo("a")),
    () -> assertThat(1, equalTo(11))
  );
}
```

Continue Test Execution After Failure

Timeout in JUnit 4 (One Method Version)

```
@Test(timeout=1000)
public void testWithTimeout() {
   // ...
}
```

Timeout in JUnit 4 (All Methods Version)

```
@Rule
public Timeout globalTimeout = Timeout.seconds(10);

@Test
public void longTest() {
  // ...
}
```

Timeout in JUnit 5

```
@Test
public void longTest() {
  assertTimeout(ofSeconds(10), () -> {
    // ...
  }, "The longTest method takes more than 10 seconds");
}
```

Timeout in JUnit 5 (Preemptively Version)

```
@Test
public void longTest() {
  assertTimeoutPreemptively(ofSeconds(10), () -> {
    // ...
  }, "The longTest method takes more than 10 seconds, aborted");
}
```

Exception Testing in JUnit 4 (Try-catch Version)

```
@Test
public void catchTheException() {
  try {
    // Code that may throw an exception
    fail("Shouldn't get here");
  } catch(RuntimeException e) {
    // Assert something about the exception
  }
}
```

Exception Testing in JUnit 4 (Annotation Version)

```java
@Test(expected = RuntimeException.class)
public void annotationBasedApproach() {
  // Code that may throw an exception
}
```

Exception Testing in JUnit 4 (Rule Version)

```
@Rule
ExpectedException thrown = ExpectedException.none();

@Test
public void ruleBasedApproach() {
  thrown.expect(RuntimeException.class);
  thrown.expectMessage(containsString("..."));
  // Code that may throw an exception
}
```

Exception Testing in JUnit 5

```
@Test
void newAssertThrows() {
  assertThrows(RuntimeException.class, () -> {
    // Code that may throw an exception
  });
}
```

Exception Testing in JUnit 5

```
@Test
void newAssertThrows() {
  RuntimeException e = assertThrows(RuntimeException.class, () -> {
    // Code that may throw an exception
  });
  assertEquals("...", e.getMessage());
}
```

```
@RunWith(Suite.class)
public class MyTest {
  // ...
}
```

Extension Mode in Junit 4

@RunWith

```
@Rule
public final TemporaryFolder folder = new TemporaryFolder();


@ClassRule
public static final ExternalResource resource =
  new ExternalResource() {
      // ...
}
```

Extension Mode in Junit 4

@Rules

@ClassRule

```
class MyExtension implements BeforeTestExecutionCallback {
  // ...
}

@ExtendWith(MyExtension.class)
void myTest() { /* ... */ }
```

Extension Model in JUnit 5

@ExtendWith

New Features

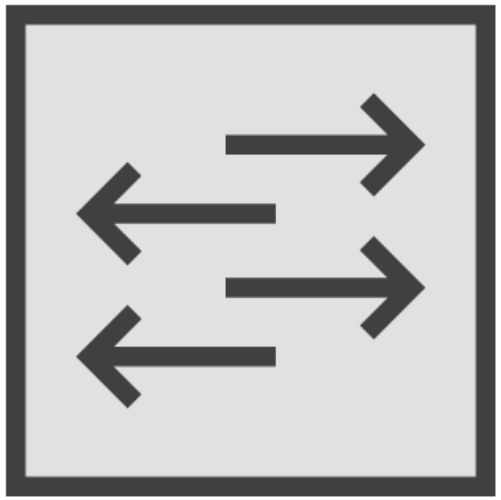
- Nested tests
- Custom display names
- Java 8 support
- Parameter injection
- Dynamic and parameterized tests
- Meta-annotations

Don’t migrate all your tests to JUnit 5 at once

Do it gradually

Running JUnit 4 Tests in JUnit 5

Compatibility



Backward compatibility

- JUnit Vintage engine

Forward compatibility

- JUnitPlatformRunner

Gradual migration to the Jupiter API

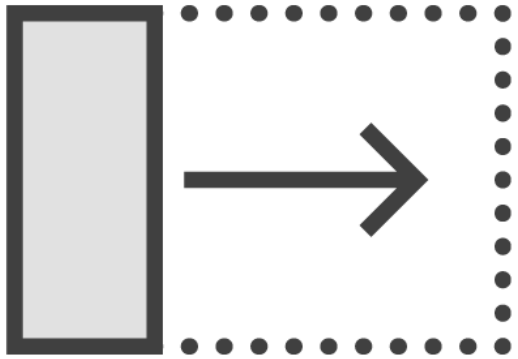
Demo

Running JUnit 4 tests in JUnit 5

- JUnit Vintage

Rule Support in JUnit 5

JUnit 4 Extension Mechanism



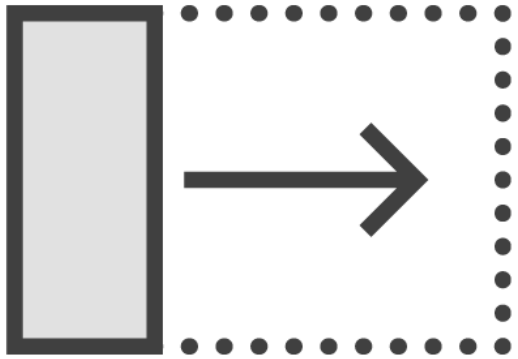
Runners

- @RunWith

Rules

- Public, non-static field
- Subtype of TestRule
- @Rule

In JUnit5...

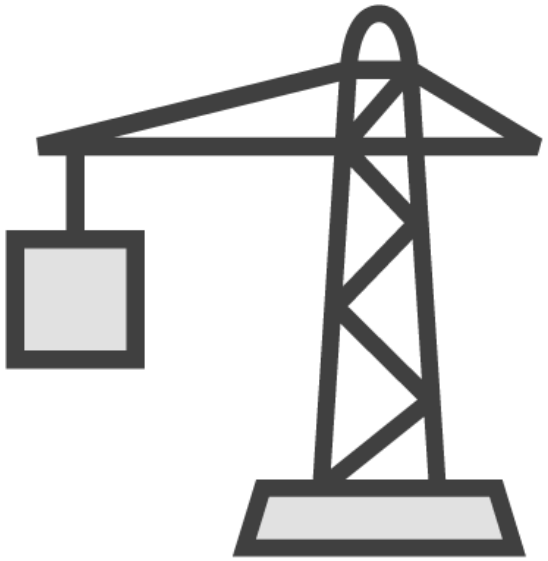


Extension points

Runners can be implemented as extensions

Limited Rule support

JUnit 5 Rule Support



ExternalResource

- TemporaryFolder

Verifier

- ErrorCollector

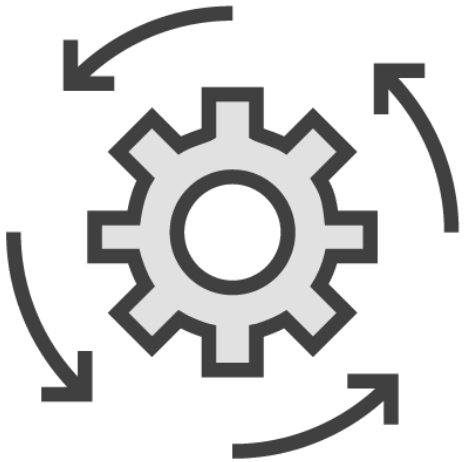
ExpectedException

```
@Target(value=TYPE)
@Retention(value=RUNTIME)
@API(status=EXPERIMENTAL, since="5.0")
@ExtendWith(value=ExternalResourceSupport.class)
@ExtendWith(value=VerifierSupport.class)
@ExtendWith(value=ExpectedExceptionSupport.class)
public @interface EnableRuleMigrationSupport { /* ... */ }
```

Annotation to Enable Support

@EnableRuleMigrationSupport

JUnit 5 Migration Support API



Group ID: org.junit.jupiter

Artifact ID: junit-jupiter-migrationsupport

Version: 5.0.1

Demo

Rule support in JUnit 5

- ErrorCollector

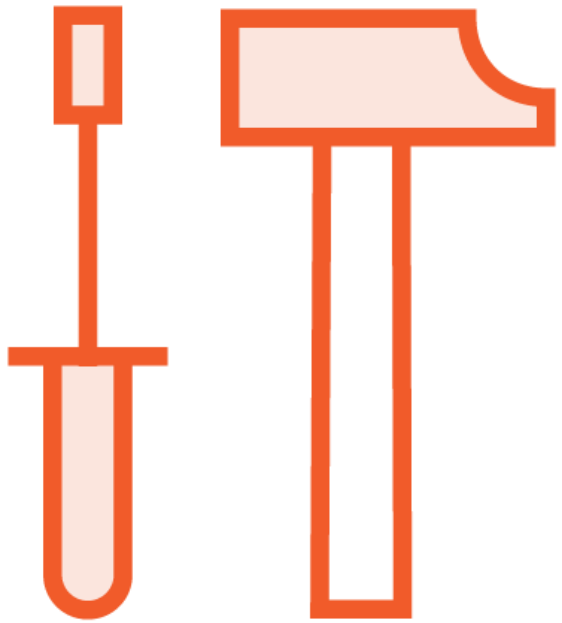
Summary

Differences between JUnit 4 and 5

Running JUnit 4 tests in JUnit 5

Rule support in JUnit 5

JUnit Is...



Thank you