

Embracing

JUnit 

with



Noopur Gupta

Eclipse JDT UI co-lead

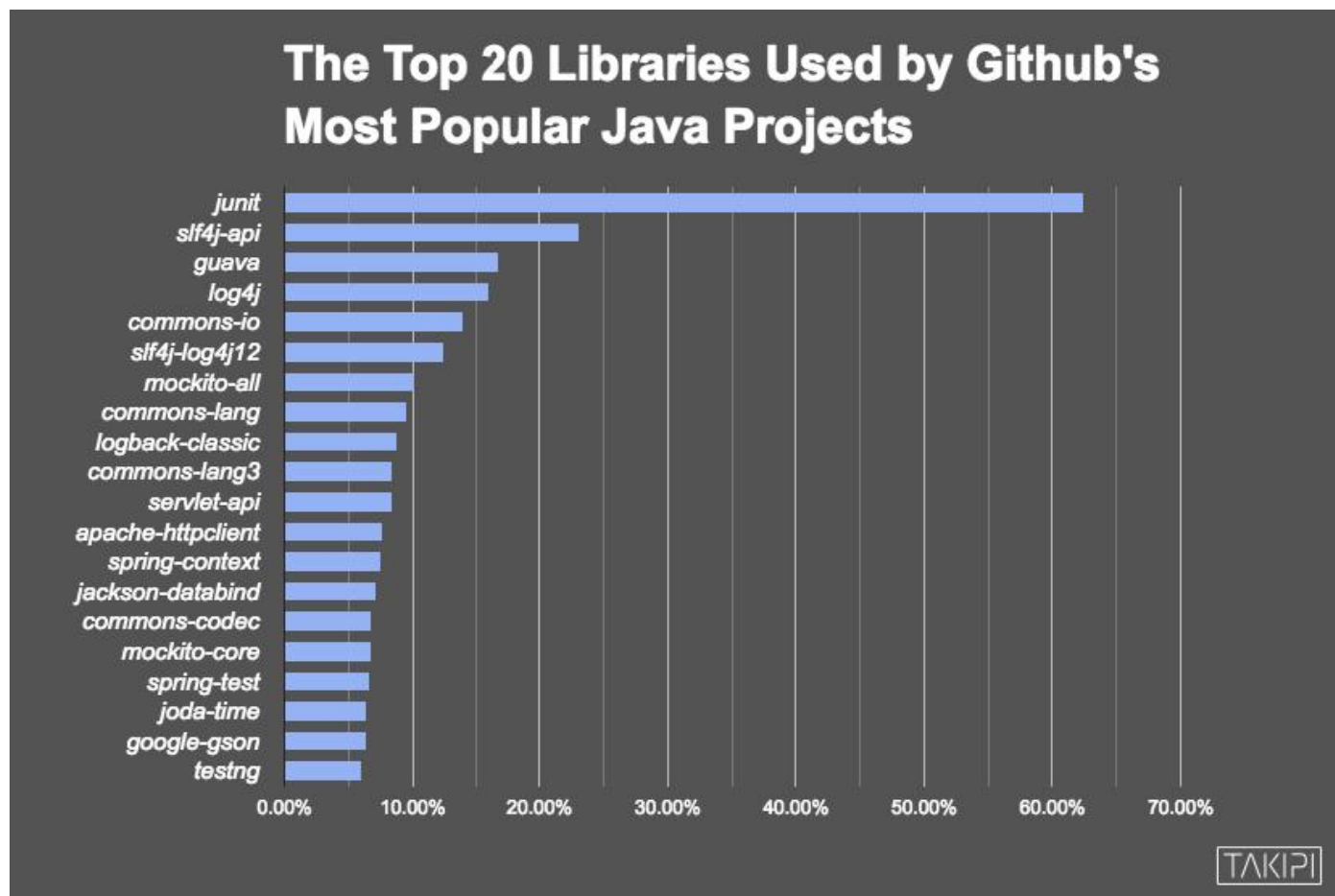
IBM India

noopur_gupta@in.ibm.com

 [@noopur2507](https://twitter.com/noopur2507)



JUnit Framework



Source: <http://blog.takipi.com/the-top-100-java-libraries-in-2016-after-analyzing-47251-dependencies/>

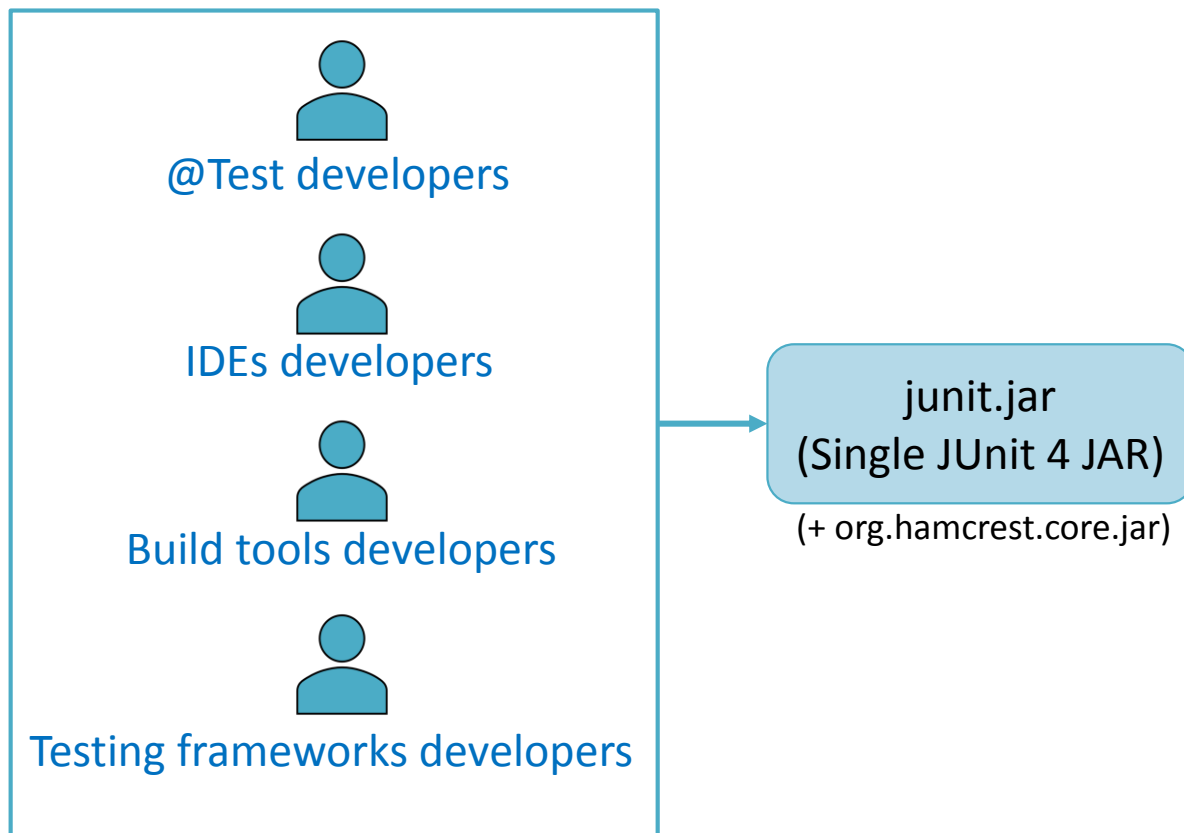
❑ JUnit 4.0

Released in 2006

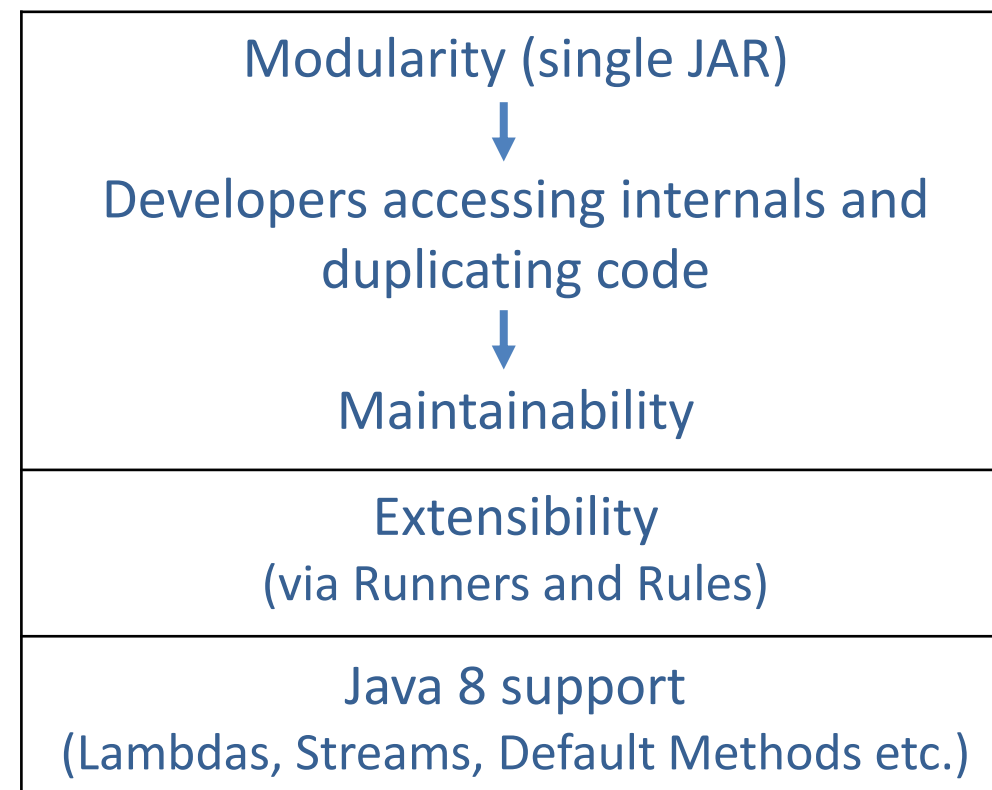
❑ JUnit 5.0

Release in Q3 2017
(expected)

JUnit 4 Architecture



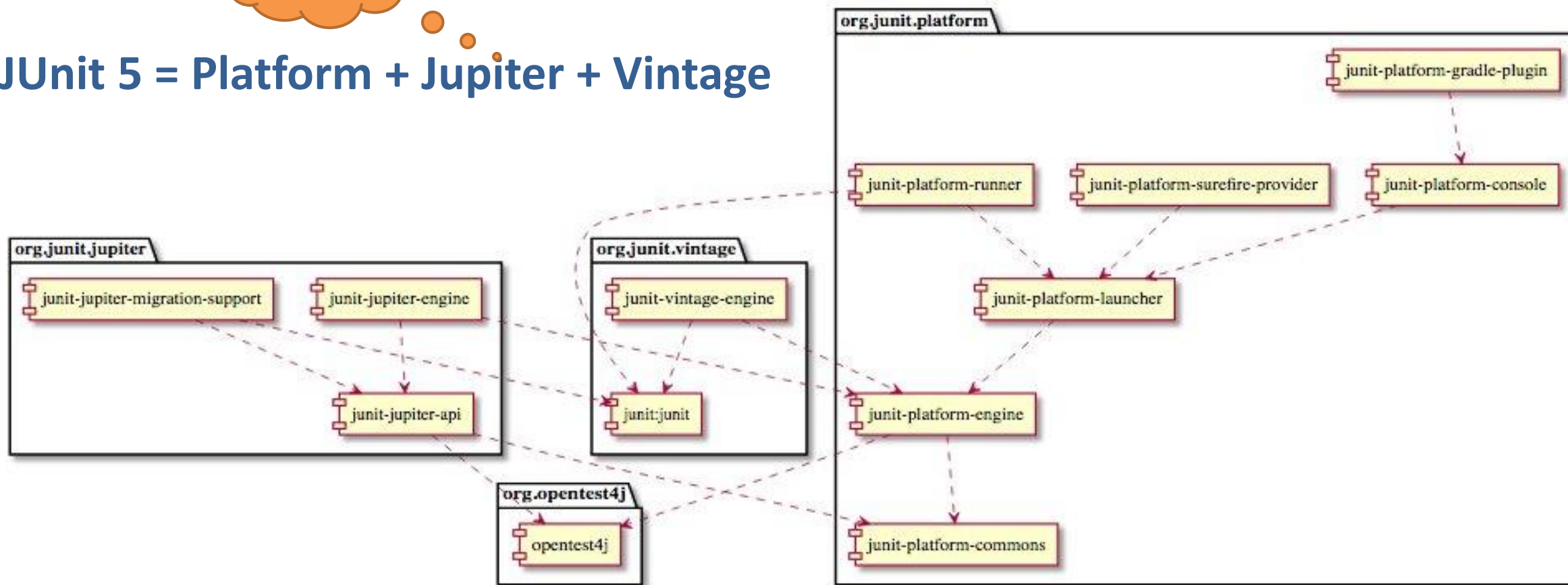
Problems



JUnit 5 Architecture

5th planet
in Solar
System!

JUnit 5 = Platform + Jupiter + Vintage



Source: (JUnit 5 M3 User Guide) <http://junit.org/junit5/docs/current/user-guide/#dependency-diagram>

JUnit 5 Architecture

❑ JUnit Platform

- **junit-platform-launcher**
Launcher APIs used by IDEs and build tools to launch the framework.
Finds test engines via Java's *ServiceLoader* mechanism.
- **junit-platform-engine**
TestEngine APIs for integration of any testing framework like JupiterTestEngine.
- ...

❑ JUnit Jupiter

- **junit-jupiter-api**
APIs for the new programming and extension model.
- **junit-jupiter-engine**
To discover and execute Jupiter tests.

❑ JUnit Vintage

- **junit-vintage-engine**
To discover and execute JUnit 3 and JUnit 4 tests on JUnit 5 platform.



A sneak peek into the major interesting
features of JUnit Jupiter with
Eclipse beta support for JUnit 5.

Eclipse beta support for JUnit 5

Follow the steps provided in the [wiki](#) to setup JUnit 5 support in Eclipse.



Demo: Eclipse 4.7 M6 build with JUnit 5 support based on JUnit 5 M3 release (using JUnit5-20170313-snapshot JARs).

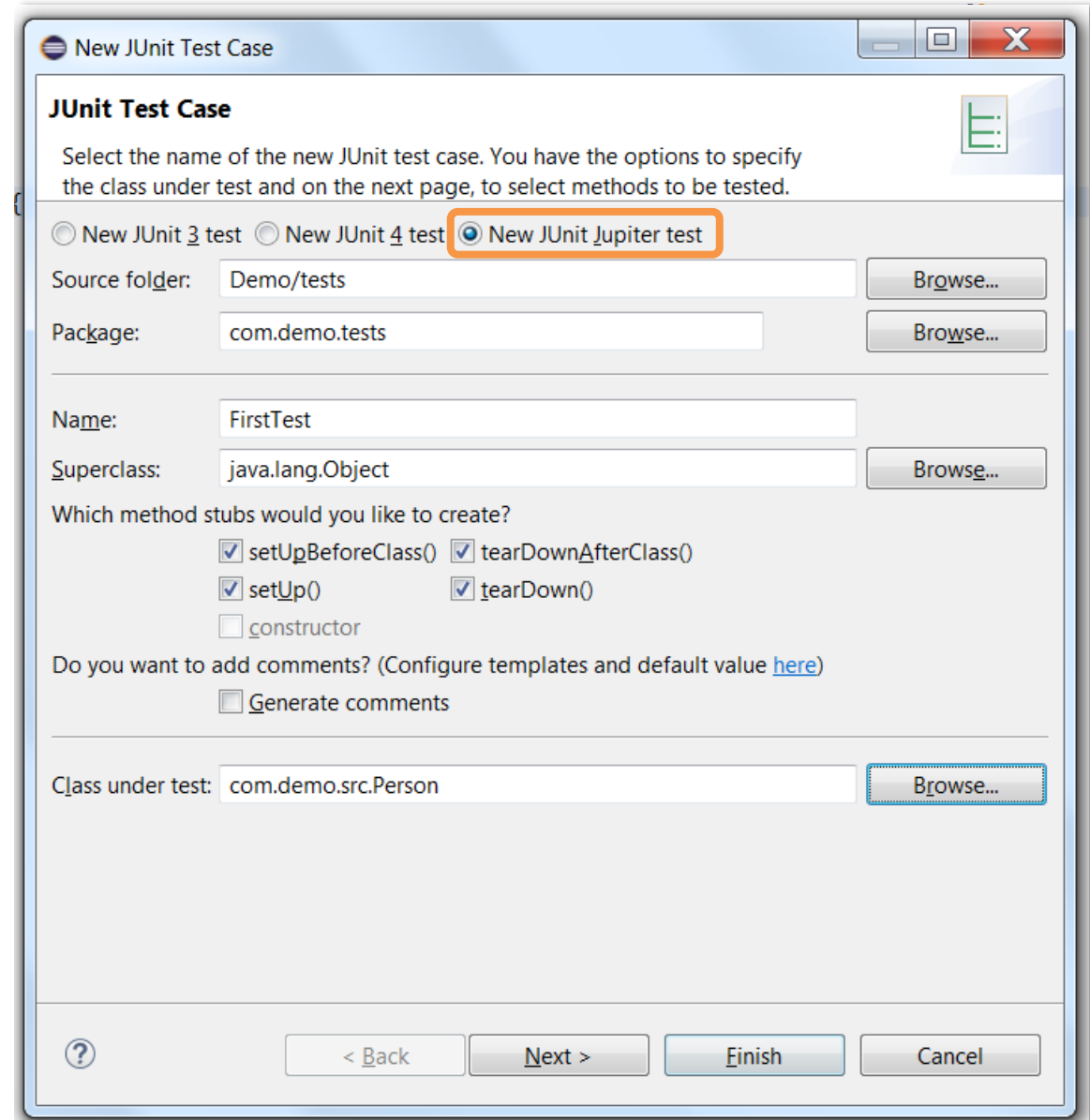
Demo

JUnit Jupiter - Programming Model

Demo

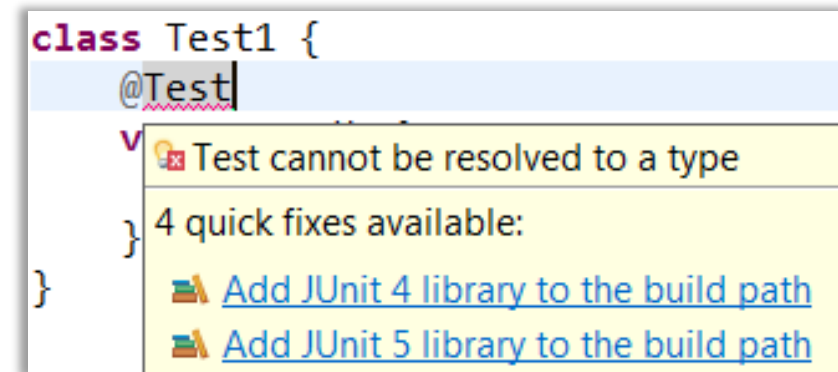
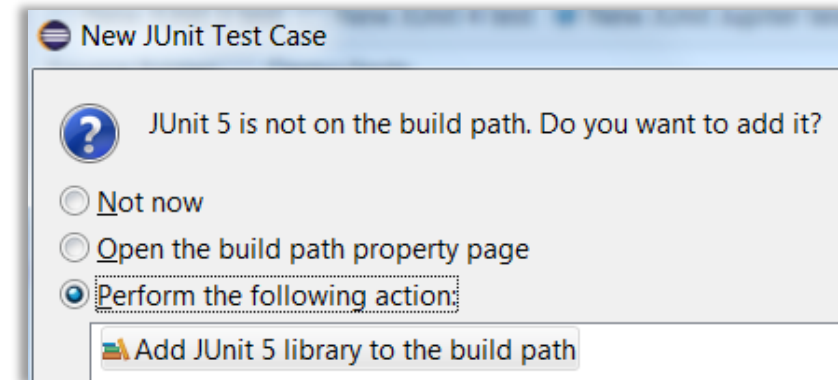
- Create new JUnit Jupiter test in Eclipse with setUp and tearDown methods for a class and its methods being tested:

**New JUnit Test Case wizard
> New JUnit Jupiter test**



Demo

- Add JUnit 5 library to the build path
- **New JUnit Test Case** wizard offers to add it while creating a new JUnit Jupiter test.
- **Quick Fix (Ctrl+1)** proposal on @Test.



Demo

- Visibility: Test classes and methods can have any access modifier (other than private).
- Annotations

JUnit 4	JUnit 5
@org.junit.Test	@org.junit.jupiter.api.Test (No <i>expected</i> and <i>timeout</i> attributes)
@BeforeClass	@BeforeAll
@AfterClass	@AfterAll
@Before	@BeforeEach
@After	@AfterEach
@Ignore	@Disabled

```
Person.java  FirstTest.java
1 package com.demo.tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.AfterAll;
6 import org.junit.jupiter.api.AfterEach;
7 import org.junit.jupiter.api.BeforeAll;
8 import org.junit.jupiter.api.BeforeEach;
9 import org.junit.jupiter.api.Test;
10
11 class FirstTest {
12
13     @BeforeAll
14     static void setUpBeforeClass() throws Exception {
15     }
16
17     @AfterAll
18     static void tearDownAfterClass() throws Exception {
19     }
20
21     @BeforeEach
22     void setUp() throws Exception {
23     }
24
25     @AfterEach
26     void tearDown() throws Exception {
27     }
28
29     @Test
30     void testGetFirstName() {
31         fail("Not yet implemented");
32     }
33
34 }
```

Demo

- Create a JUnit Jupiter test method in Eclipse with the new template:

test5

```
class First {  
    test5  
}  
  
@org.junit.jupiter.api.Test  
void testName() throws Exception {  
}
```

test5 - test method (JUnit Jupiter)
test5() : void - Method stub

Demo - Assertions

(org.junit.jupiter.api.Assertions class)

- Failure message comes at the end of arguments list. `assertTrue("").isEmpty(), "Should be empty.");`
- Failure message can be retrieved lazily. `assertTrue("").isEmpty(), () -> evaluateFailureMessage());`
- Grouped assertions to execute all assertions first and then report all failures together.

```
assertAll("Person details",
    () -> assertEquals("Johnny", p.getFirstName()),
    () -> assertEquals("Depp", p.getLastName()),
    () -> assertEquals(53, p.getAge()));
```



Failure Trace

org.opentest4j.MultipleFailuresError: Person details (2 failures)

expected: <Johnny> but was: <Johny>
expected: <53> but was: <30>

Result Comparison

testGroupedAssertions(com.demo.tests.AssertionsDemo)

Expected	Actual
1 Johnny	1 Johny
2	2
3 53	3 30

- Exception testing to assert and evaluate a thrown exception.

```
Exception exception = assertThrows(StringIndexOutOfBoundsException.class, () -> printSubstring(-1));
assertEquals(exception.getMessage(), "String index out of range: -1");
```

- Asserting that the given task completes before the given timeout.

```
assertTimeout(Duration.ofMillis(10), () -> {
    Thread.sleep(50); // task taking more than 10 ms
});
```

Demo - Assumptions

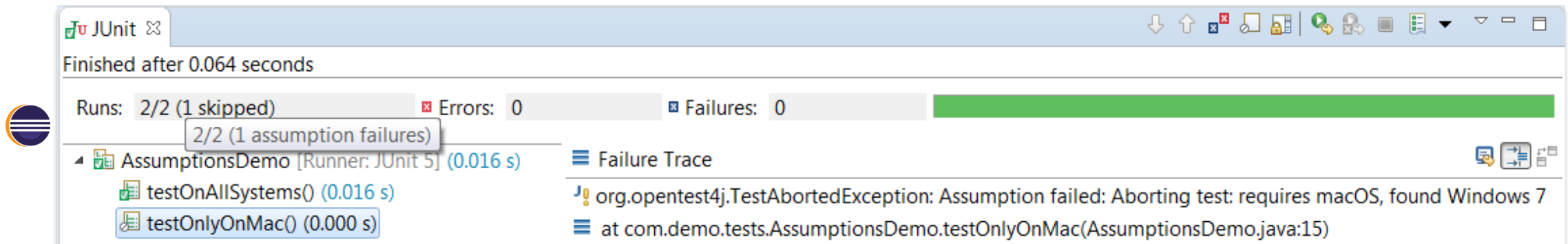
(org.junit.jupiter.api.Assumptions class)

```
@Test
void testOnlyOnMac() throws Exception {
    assumeTrue("macOS".equals(System.getProperty("os.name")), () -> {
        return "Aborting test: requires macOS, found " + System.getProperty("os.name");
    });

    // test code for macOS
    assertTrue(new File("/Volumes/").exists());
}
```

```
@Test
void testOnAllSystems() throws Exception {
    assumingThat("macOS".equals(System.getProperty("os.name")), () -> {
        assertTrue(new File("/Volumes/").exists());
    });

    // test code for all systems
    assertTrue(1 + 1 == 2);
}
```



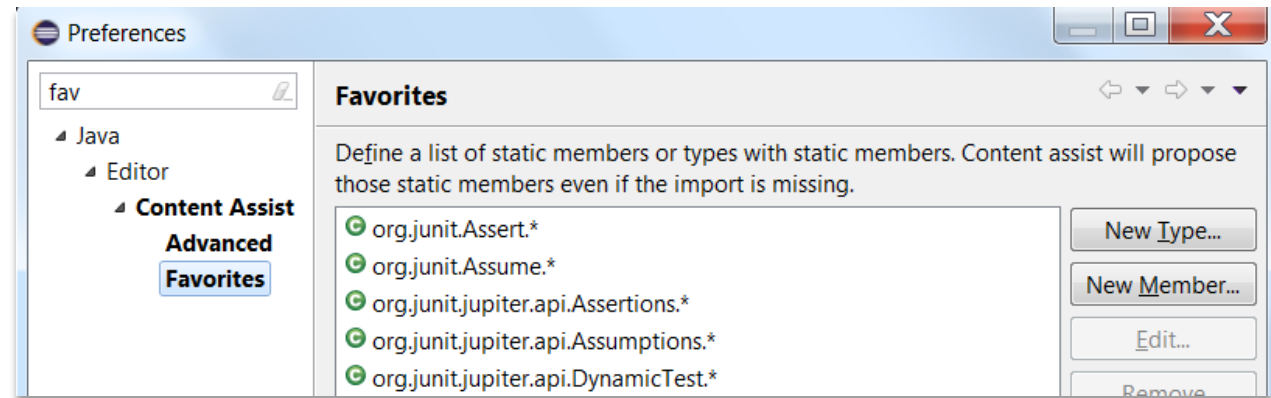
The screenshot shows the JUnit 5 test runner interface. At the top, it says "Finished after 0.064 seconds". Below this, a summary bar shows "Runs: 2/2 (1 skipped)", "Errors: 0", and "Failures: 0". A tooltip over the "Runs" section indicates "2/2 (1 assumption failures)". The test list on the left shows "AssumptionsDemo [Runner: JUnit 5] (0.016 s)" expanded, with "testOnAllSystems() (0.016 s)" and "testOnlyOnMac() (0.000 s)". The "Failure Trace" on the right shows the error: "org.opentest4j.TestAbortedException: Assumption failed: Aborting test: requires macOS, found Windows 7" at "com.demo.tests.AssumptionsDemo.testOnlyOnMac(AssumptionsDemo.java:15)".

Demo

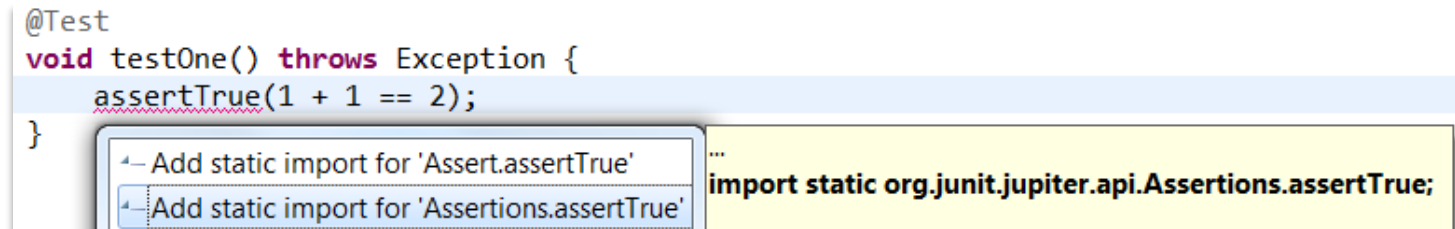
- JUnit Jupiter's *Assertions*, *Assumptions* & *DynamicTest* classes are now added to Eclipse Favorites by default.

Preferences

> *Java* > *Editor*
> *Content Assist*
> *Favorites*



Import static methods in your code from favorite classes via Content Assist (*Ctrl + Space*) and Quick Fix (*Ctrl + 1*).



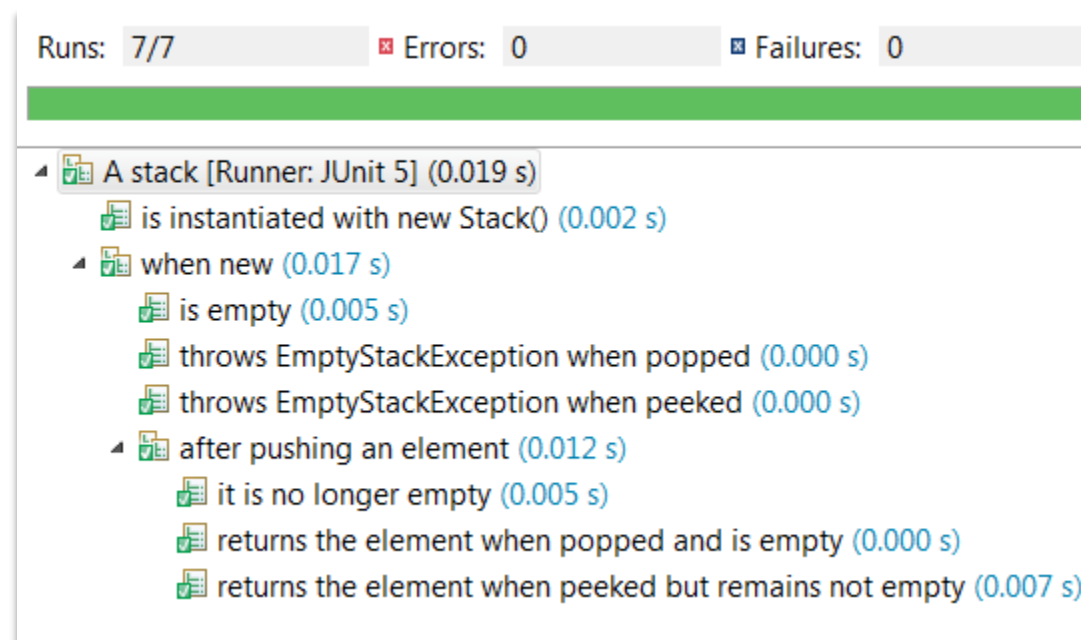
Configure the number of static member imports needed before type.* is used.



Demo - @Nested Test Classes

Non-static nested classes (i.e. inner classes) can serve as *@Nested* tests for logical grouping of test cases.

Example: TestingAStackDemo in JUnit 5 user guide.

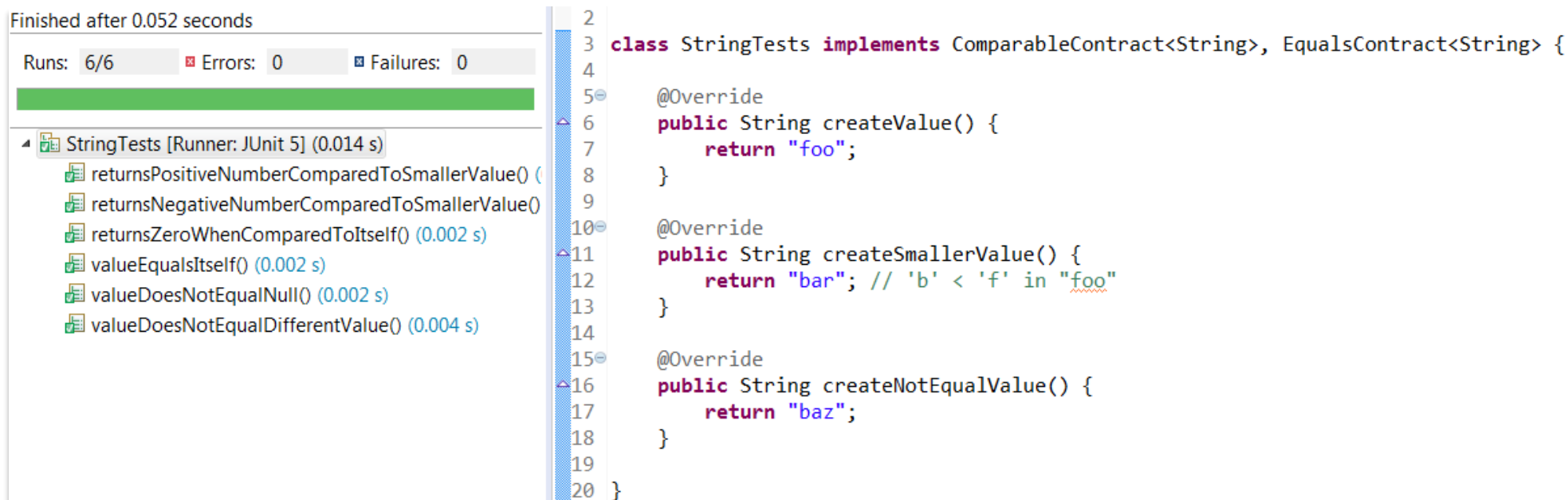


Demo - Interface Default Methods

- Default test methods can be inherited by implementing test classes.
- Enables multiple inheritance in tests classes.

Example: StringTests in JUnit 5 user guide.

```
@Test
default void valueEqualsItself() {
    T value = createValue();
    assertEquals(value, value);
}
```



The screenshot displays the Eclipse IDE interface. On the left, the 'JUnit 5' test runner shows a successful run of 'StringTests' with 6/6 tests passed, 0 errors, and 0 failures. The test results list includes: 'returnsPositiveNumberComparedToSmallerValue()', 'returnsNegativeNumberComparedToSmallerValue()', 'returnsZeroWhenComparedToItself() (0.002 s)', 'valueEqualsItself() (0.002 s)', 'valueDoesNotEqualNull() (0.002 s)', and 'valueDoesNotEqualDifferentValue() (0.004 s)'. The main editor shows the source code for 'StringTests', which implements 'ComparableContract<String>' and 'EqualsContract<String>'. The code includes three overridden methods: 'createValue()' returning 'foo', 'createSmallerValue()' returning 'bar', and 'createNotEqualValue()' returning 'baz'. A callout box above the code shows a default test method 'valueEqualsItself()' that uses 'createValue()' and 'assertEquals()'.

```
2
3 class StringTests implements ComparableContract<String>, EqualsContract<String> {
4
5     @Override
6     public String createValue() {
7         return "foo";
8     }
9
10    @Override
11    public String createSmallerValue() {
12        return "bar"; // 'b' < 'f' in "foo"
13    }
14
15    @Override
16    public String createNotEqualValue() {
17        return "baz";
18    }
19
20 }
```

Demo - Tagging, Meta-Annotations and Composed Annotations

- Tag test classes and test methods with *@Tag*. Tags can later be used to filter test execution.

```
@Tag("performance")
@Test
void testPerformance() {
}
```

- JUnit Jupiter annotations can be used as *meta-annotations*.
- Create custom *composed annotation* inheriting semantics of its meta-annotations.

```
@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Tag("performance")
@Test
@interface PerformanceTest {
}
```

```
@PerformanceTest
void testPerformance() {
}
```

Demo - Dynamic Tests

- Dynamic tests are generated at runtime by a **@TestFactory** method.

🌀 Create a @TestFactory method in Eclipse with the new template: **testfactory**

```
class DynamicTestsDemo {
    testfact
}

testfactory - test factory method (JUnit Jupiter)
testfact() : void - Method stub
@TestFactory - org.junit.jupiter.api
TestFactoryTestDescriptor - org.junit.jupiter.engine.c

@org.junit.jupiter.api.TestFactory
Stream<DynamicTest> testFactoryName() throws Except
// TODO: generate dynamic test cases
return null;
}
```

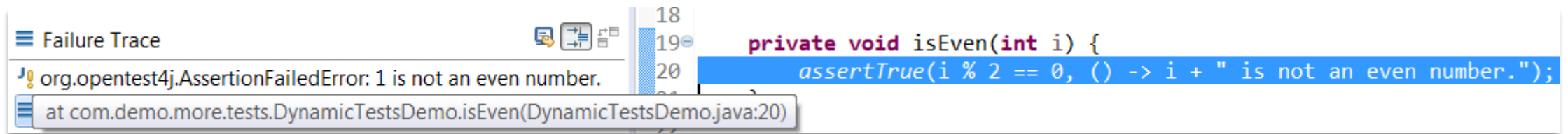
- Dynamic test is composed of a display name and an Executable.

```
DynamicTestsDemo [Runner: JUnit 5] (0.005 s)
├─ dynamicTests() (0.005 s)
│   └─ dynamic test for: 1 (0.005 s)
│       └─ dynamic test for: 2 (0.005 s)
└─

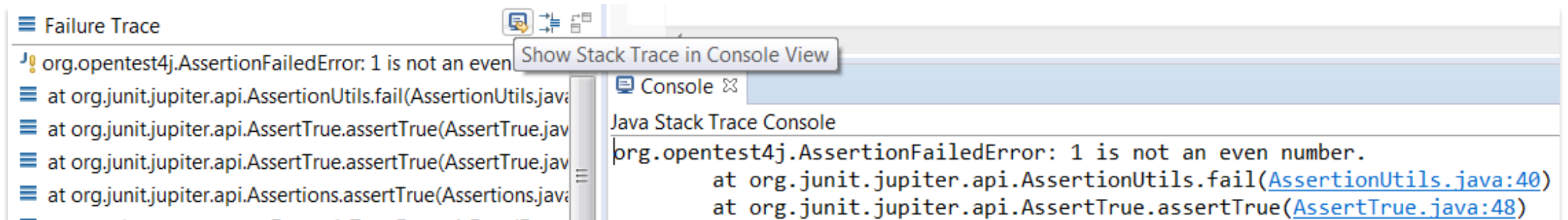
13 @TestFactory
14 Stream<DynamicTest> dynamicTests() throws Exception {
15     return Stream.of(1, 2).map(
16         i -> dynamicTest("dynamic test for: " + i, () -> isEven(i)));
17 }
18 }
```

Demo - Navigate to failing source location

- Double-click an entry in JUnit view's failure trace to jump to the corresponding source location.



- Click "Show Stack Trace in Console View" and use the hyperlinks. It can also be used to copy parts of the stack trace.



Demo – Dependency injection

- Test constructors and methods are now permitted to have parameters enabling Dependency Injection.
- **ParameterResolver** is the extension API to provide a parameter resolver which dynamically resolves a parameter at runtime.
- Two built-in resolvers by JUnit Jupiter that are registered automatically:
 - **TestInfoParameterResolver** for parameter type **TestInfo**
 - **TestReporterParameterResolver** for parameter type **TestReporter**

```
@Test
@DisplayName("my test")
@Tag("my tag")
void testDependencyInjection(TestInfo testInfo) throws Exception {
    assertEquals("my test", testInfo.getDisplayName());
    assertTrue(testInfo.getTags().contains("my tag"));
}
```

JUnit Jupiter - Extension Model

- ❑ Provides extension points as interfaces in **org.junit.jupiter.api.extension** package to be implemented by extension providers.
- ❑ Register one or more extensions on a test class, test method, or composed annotation with **@ExtendWith(...)**.

```
@ExtendWith({ FooExtension.class, BarExtension.class })  
class MyTest {  
    // ...  
}
```

See the [JUnit 5 User Guide](#) for more details.

Resources

➤ JUnit 5 Project

<http://junit.org/junit5>

- User Guide

<http://junit.org/junit5/docs/current/user-guide>

- Javadoc

<http://junit.org/junit5/docs/current/api>

- GitHub

<https://github.com/junit-team/junit5>

- Q&A

<http://stackoverflow.com/questions/tagged/junit5>

➤ Eclipse support for JUnit 5

https://wiki.eclipse.org/JDT_UI/JUnit_5



eclipse
CONVERGE

Evaluate the Sessions

Sign in and vote at eclipseconverge.org

-1 0 +1



eclipse
CONVERGE

Thank You!

That's all Folks!

