JUnit 5
Die Plattform

Johannes Link

@johanneslink

johanneslink.net

Softwaretherapeut

"In Deutschland ist die Bezeichnung Therapeut allein oder ergänzt mit bestimmten Begriffen gesetzlich nicht geschützt und daher kein Hinweis auf ein erfolgreich abgeschlossenes Studium oder auch nur fachliche Kompetenz." Quelle: Wikipedia

- JUnit-Anwender seit Stunde 1
- JUnit-5 Initiator - zusammen mit Marc Philipp - und Core-Committer im ersten Jahr
- jqwik: Externe JUnit5-Test-Engine für Property-Based Testing

Warum braucht die Welt ein neues JUnit?

A Brief History of JUnit

1997 JUnit 1.0

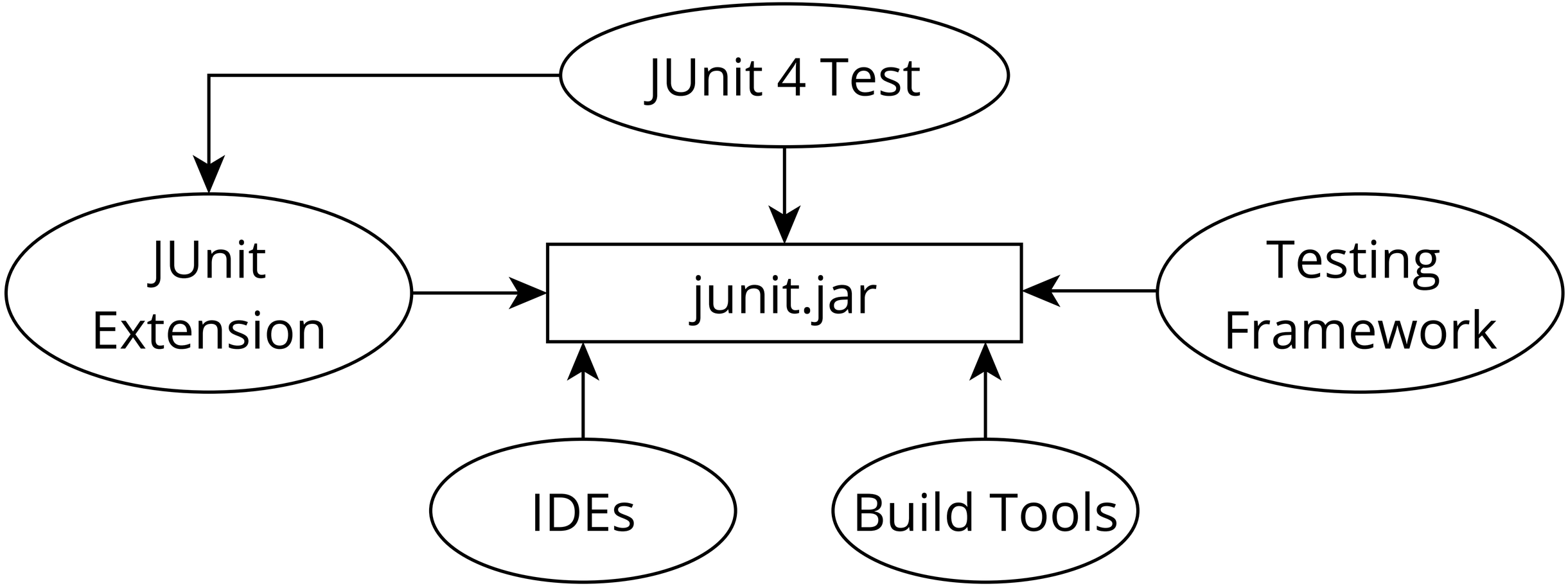
2006 JUnit 4.0 Runner

2009 JUnit 4.7 Rules

2015 JUnit Lambda Campaign

2017 JUnit 5.0.0

JUnit 4 "Architektur"



Wartbarkeit

ComparisonFailure.java

```
12 public class ComparisonFailure extends AssertionError {
13     /**
14      * The maximum length for expected and actual strings.
15      *
16      * @see ComparisonCompactor
17      */
18     private static final int MAX_CONTEXT_LENGTH = 20;
19     private static final long serialVersionUID = 1L;
20
21     /*
22      * We have to use the f prefix until the next major re
23      * serialization compatibility.
24      * See https://github.com/junit-team/junit/issues/976
25      */
26     private String fExpected;
27     private String fActual;
```

4.11

```
Done: 0 of 1  Failed: 1  (6.853 s)

expected:<[Expected] message> but was:<[Actual] messa
Expected :Expected message
Actual   :Actual message
 <Click to see difference>

org.junit.ComparisonFailure: expected:<[Expected] mes
    at ExampleTest.failingTest(ExampleTest.java:9) <
```

ComparisonFailure.java

```
12 public class ComparisonFailure extends AssertionError {
13     /**
14      * The maximum length for expected and actual strings.
15      *
16      * @see ComparisonCompactor
17      */
18     private static final int MAX_CONTEXT_LENGTH = 20;
19     private static final long serialVersionUID = 1L;
20
21     /*
22      * We have to use the f prefix until the next major re
23      * serialization compatibility.
24      * See https://github.com/junit-team/junit/issues/976
25      */
26     private String expected;
27     private String actual;
```

4.12-beta-1

```
Done: 0 of 1  Failed: 1  (7.591 s)

expected: null<null> but was: null<null>
Expected :null
Actual   :null
 <Click to see difference>

org.junit.ComparisonFailure: expected: null<null> but
    at ExampleTest.failingTest(ExampleTest.java:9) <
```

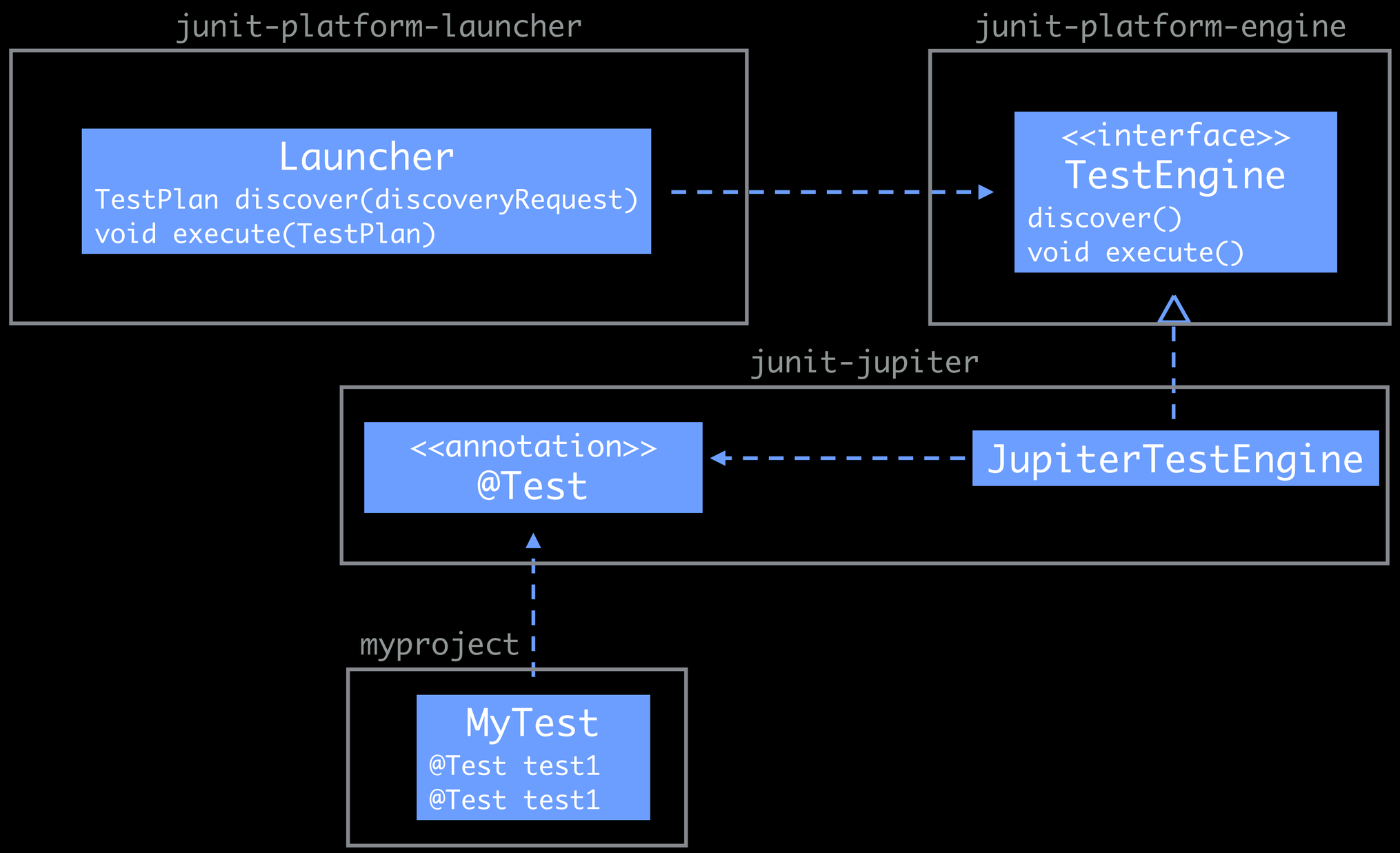
Erfolg von JUnit 4 als Plattform verhindert Weiterentwicklung von JUnit 4 als Werkzeug!

JUnit-5 Design-Ziel

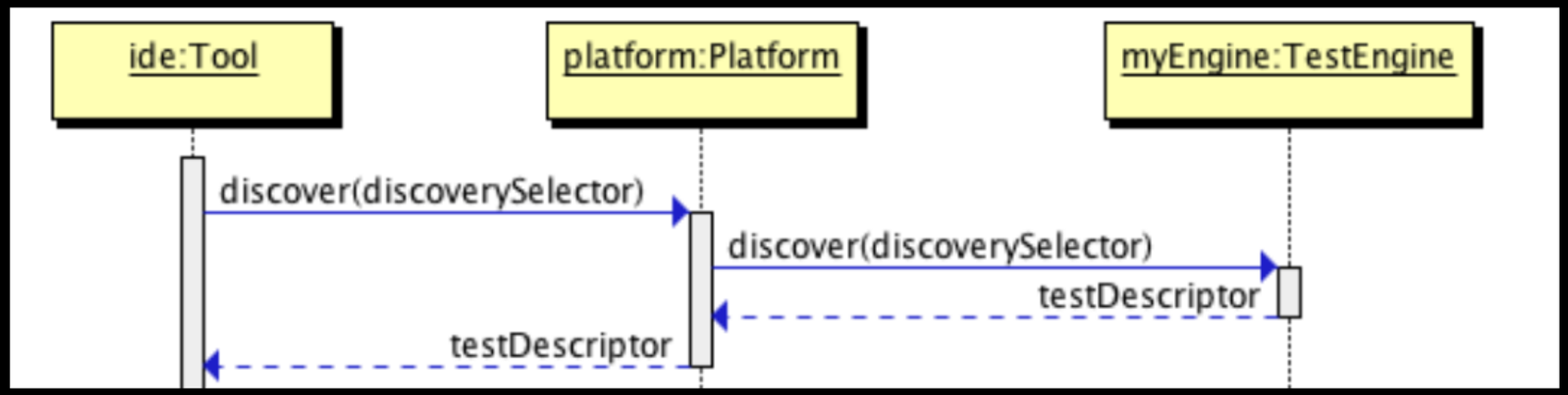
- Trennung der Aspekte
  JUnit als Testwerkzeug und
  JUnit als Plattform
- JUnit 4 und 5 nebeneinander,
  um Adoption und Migration zu erleichtern
- Ergebnis: Leichte IDE- und Tool-Integration für
  beliebige Test-Frameworks

Schritt 1: Trennung von Framework-Nutzung und Framework-Anbindern

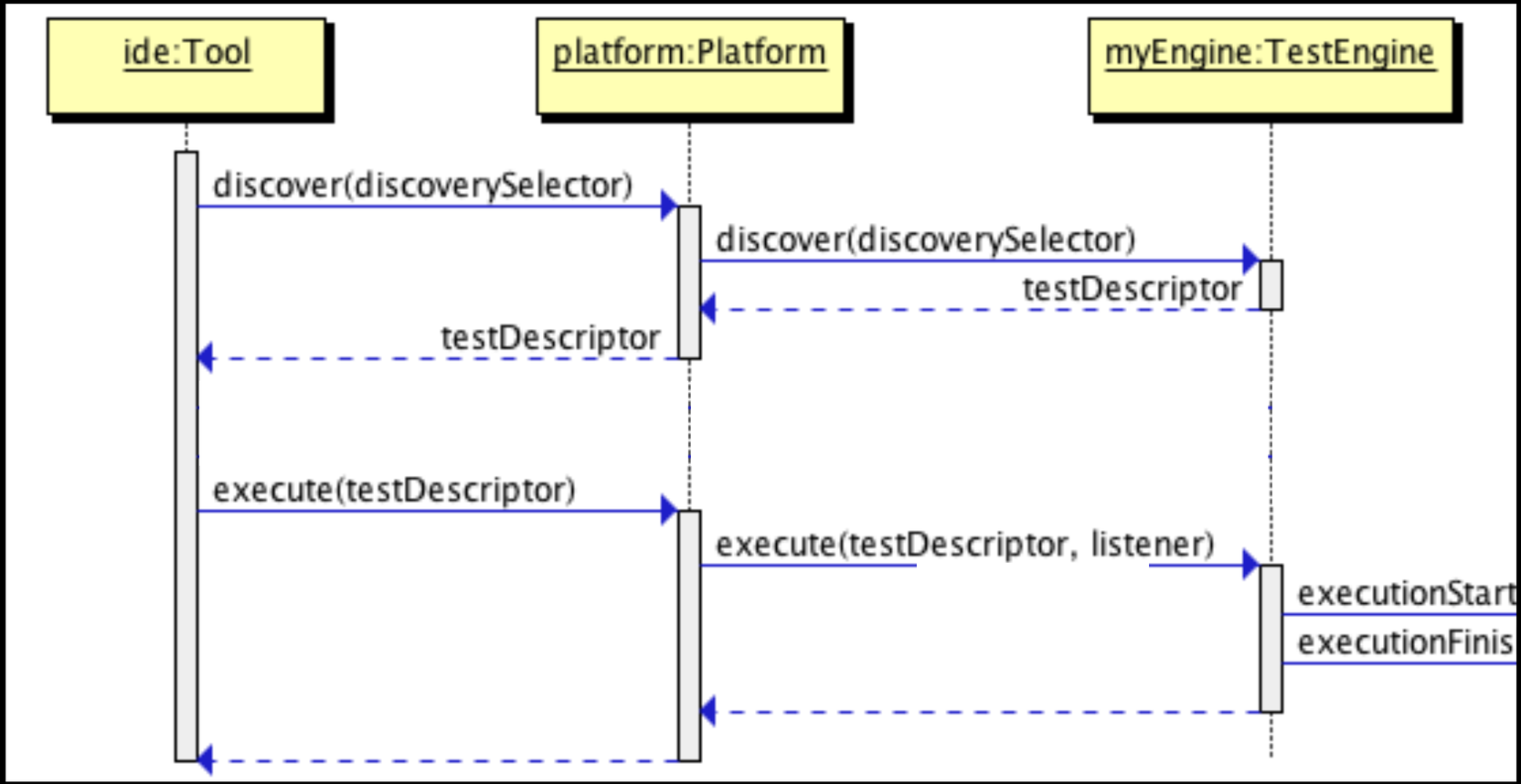
- Nutzung (API):
  - IDE: Anzeigen, Auswahl und Anstoßen von Tests
  - Programmierer: Schreiben/Spezifizieren von Tests
- Anbinder (SPI)
  - Interpretation von Testspezifikationen durch eine Test-Engine


junit-platform-launcher
Launcher
TestPlan discover(discoveryRequest)
void execute(TestPlan)
junit-platform-engine
<<interface>>
TestEngine
discover()
void execute()
junit-jupiter
<<annotation>>
@Test
JupiterTestEngine
myproject
MyTest
@Test test1
@Test test1

Trennung von Entdecken und Ausführen

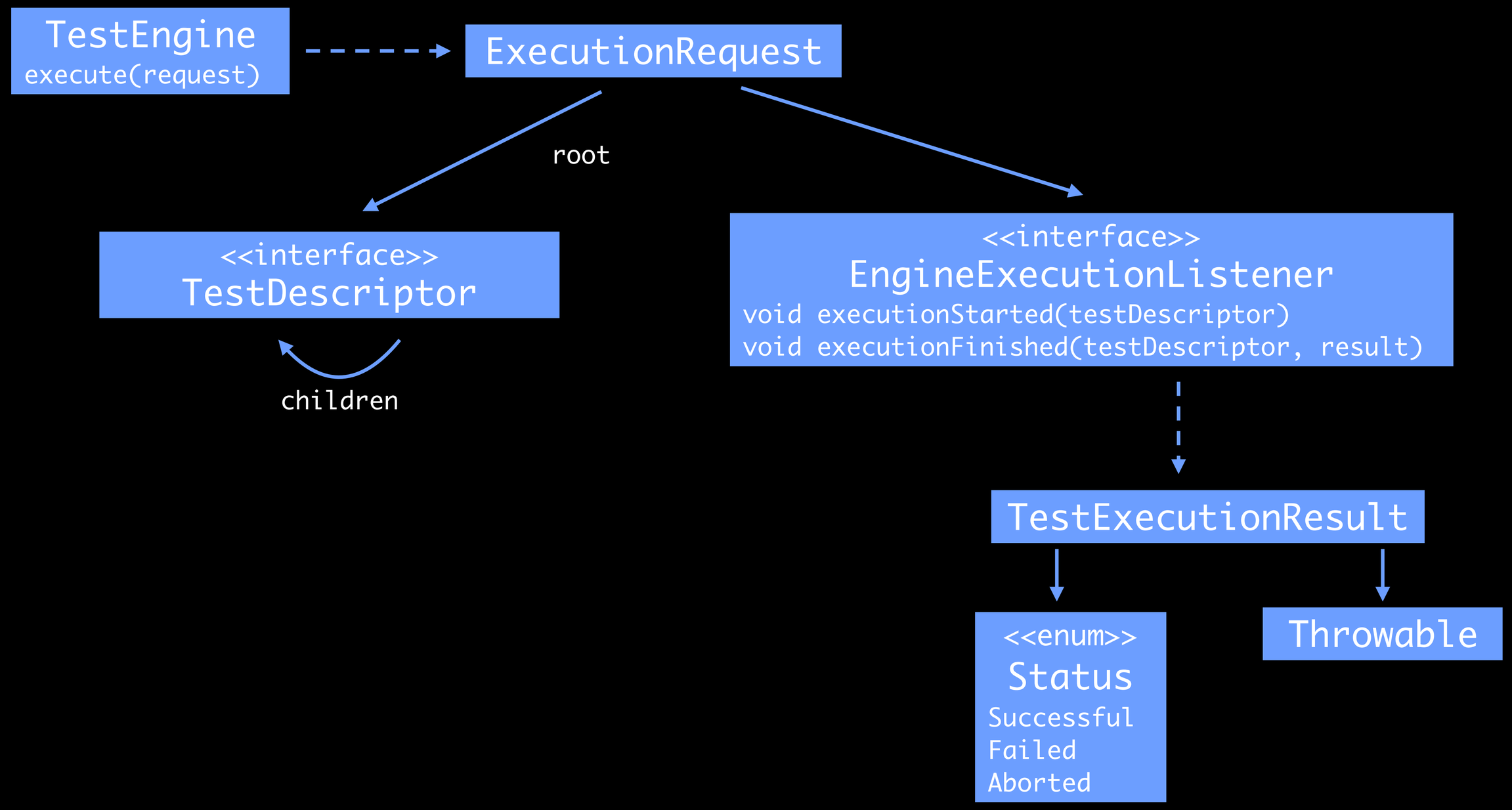


Trennung von Entdecken und Ausführen



Schritt 2: Ständiger Fortschrittsbericht eines Testlaufs

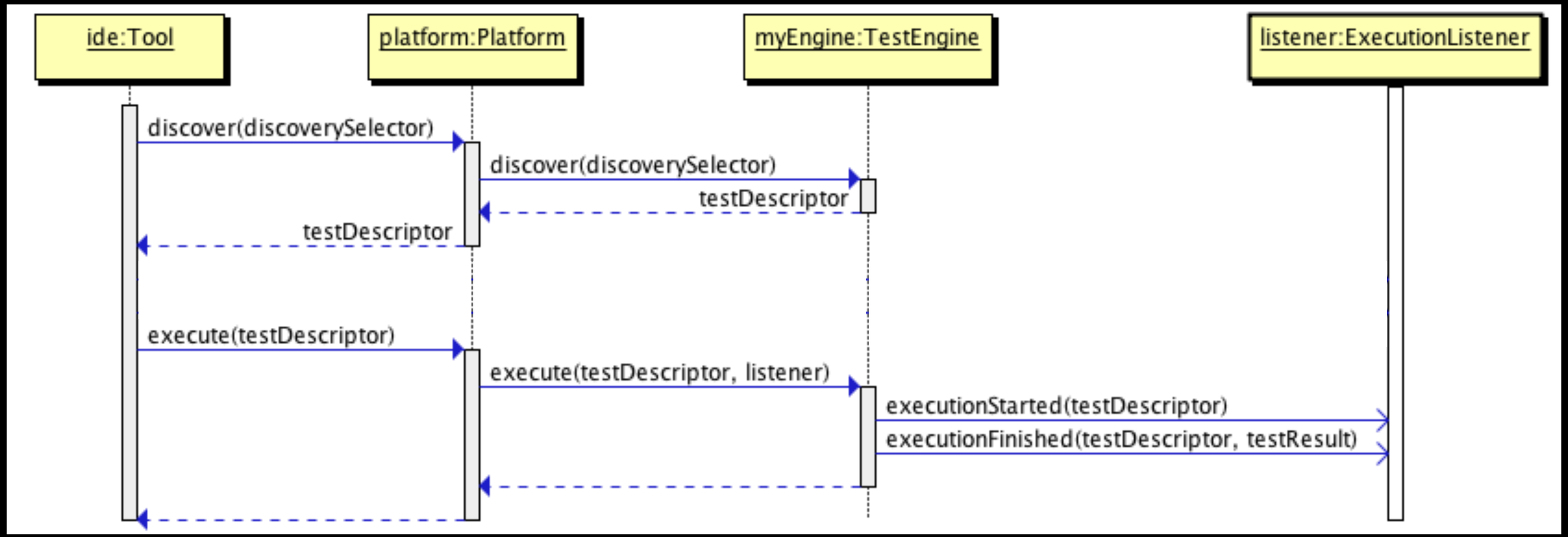
- Events / Nachrichten
  statt Rückgabe eines Ergebnisses


TestEngine
execute(request)
ExecutionRequest
root
<<interface>>
TestDescriptor
children
<<interface>>
EngineExecutionListener
void executionStarted(testDescriptor)
void executionFinished(testDescriptor, result)
TestExecutionResult
<<enum>>
Status
Successful
Failed
Aborted
Throwable

EngineExecutionListener	
dynamicTestRegistered(TestDescriptor)	void
executionSkipped(TestDescriptor, String)	void
executionStarted(TestDescriptor)	void
executionFinished(TestDescriptor, TestExecutionResult)	void
reportingEntryPublished(TestDescriptor, ReportEntry)	void

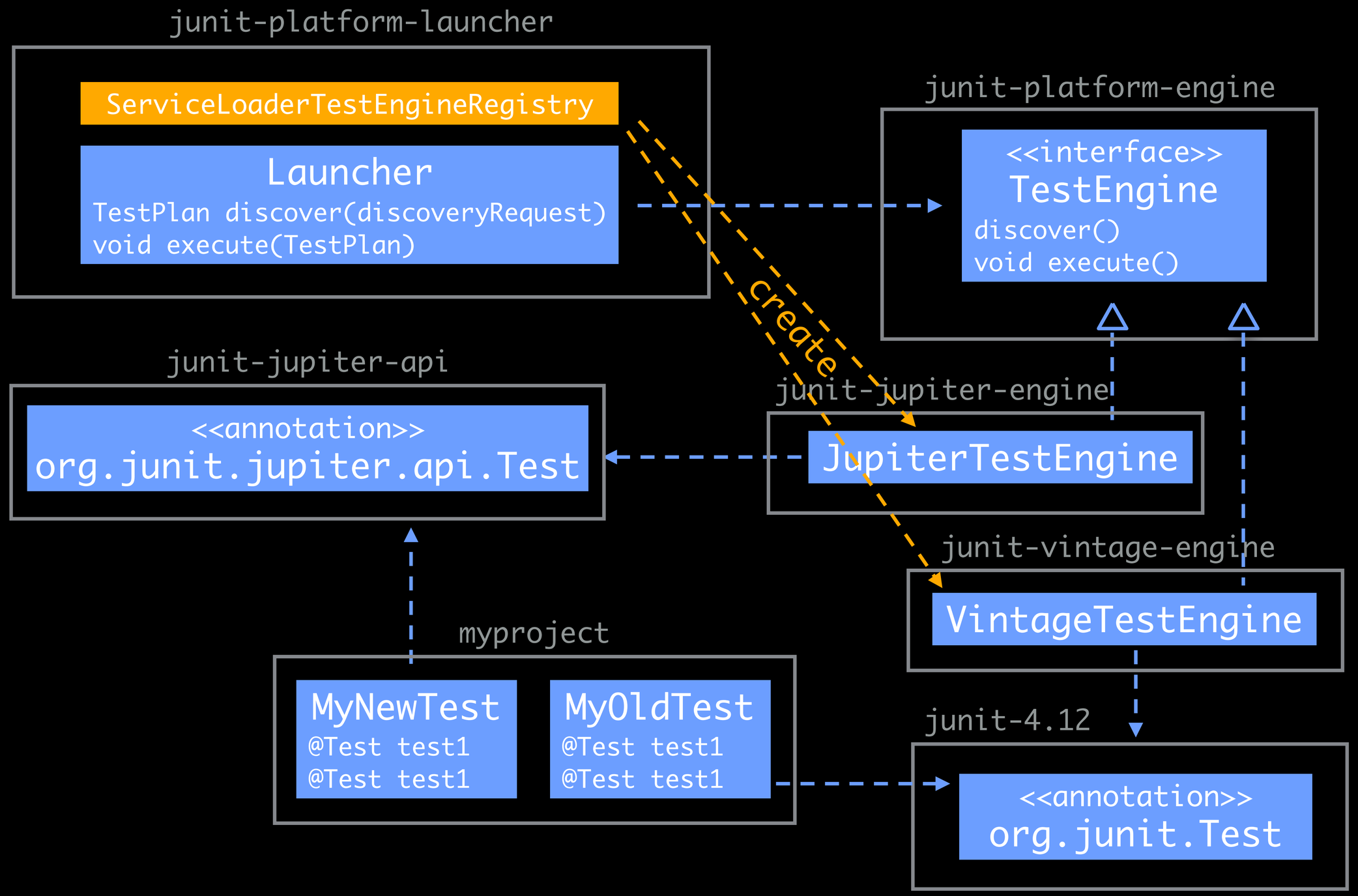
TestExecutionResult	
successful()	TestExecutionResult
aborted(Throwable)	TestExecutionResult
failed(Throwable)	TestExecutionResult
getStatus()	Status
getThrowable()	Optional<Throwable>
toString()	String

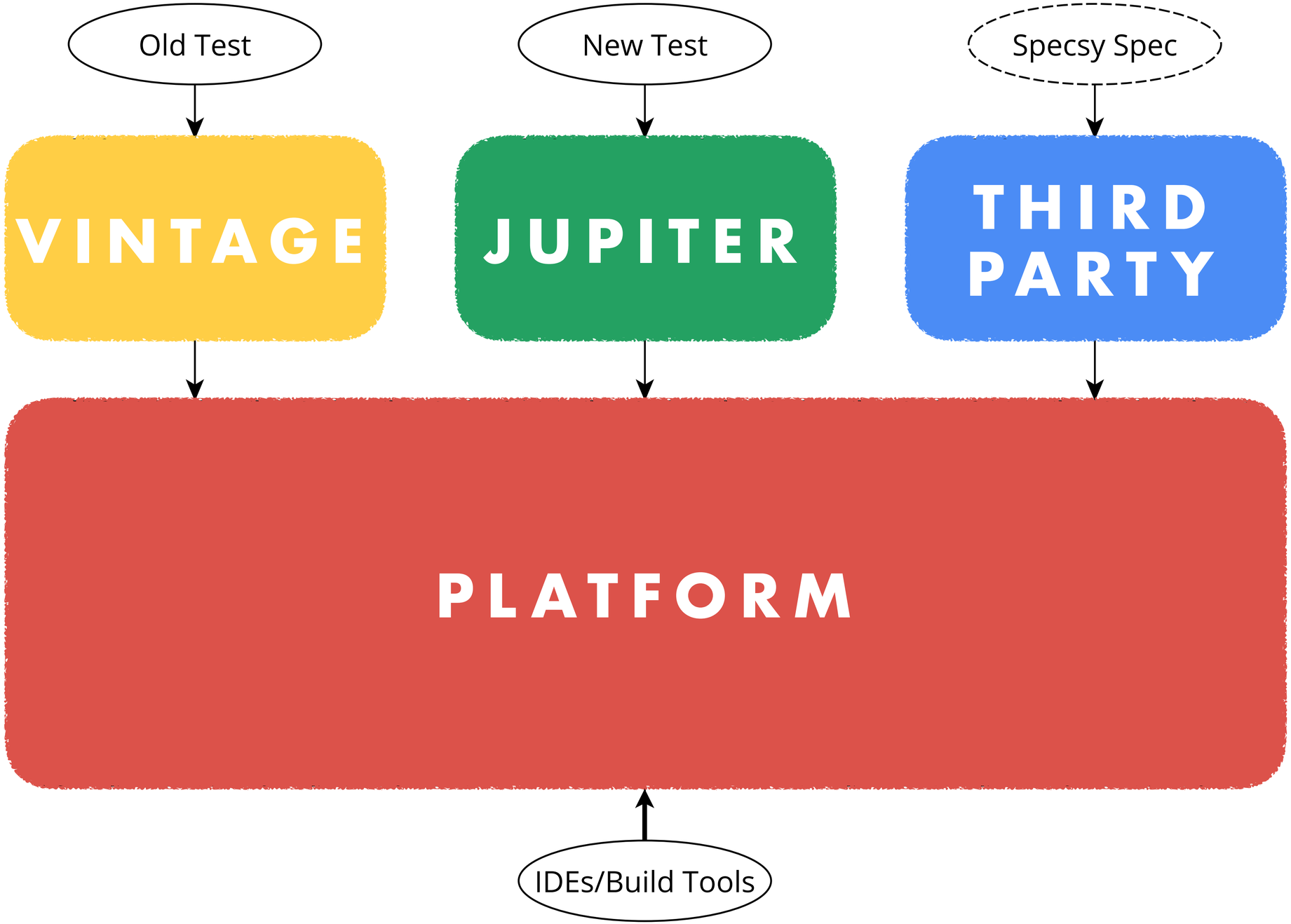
Execution Listener

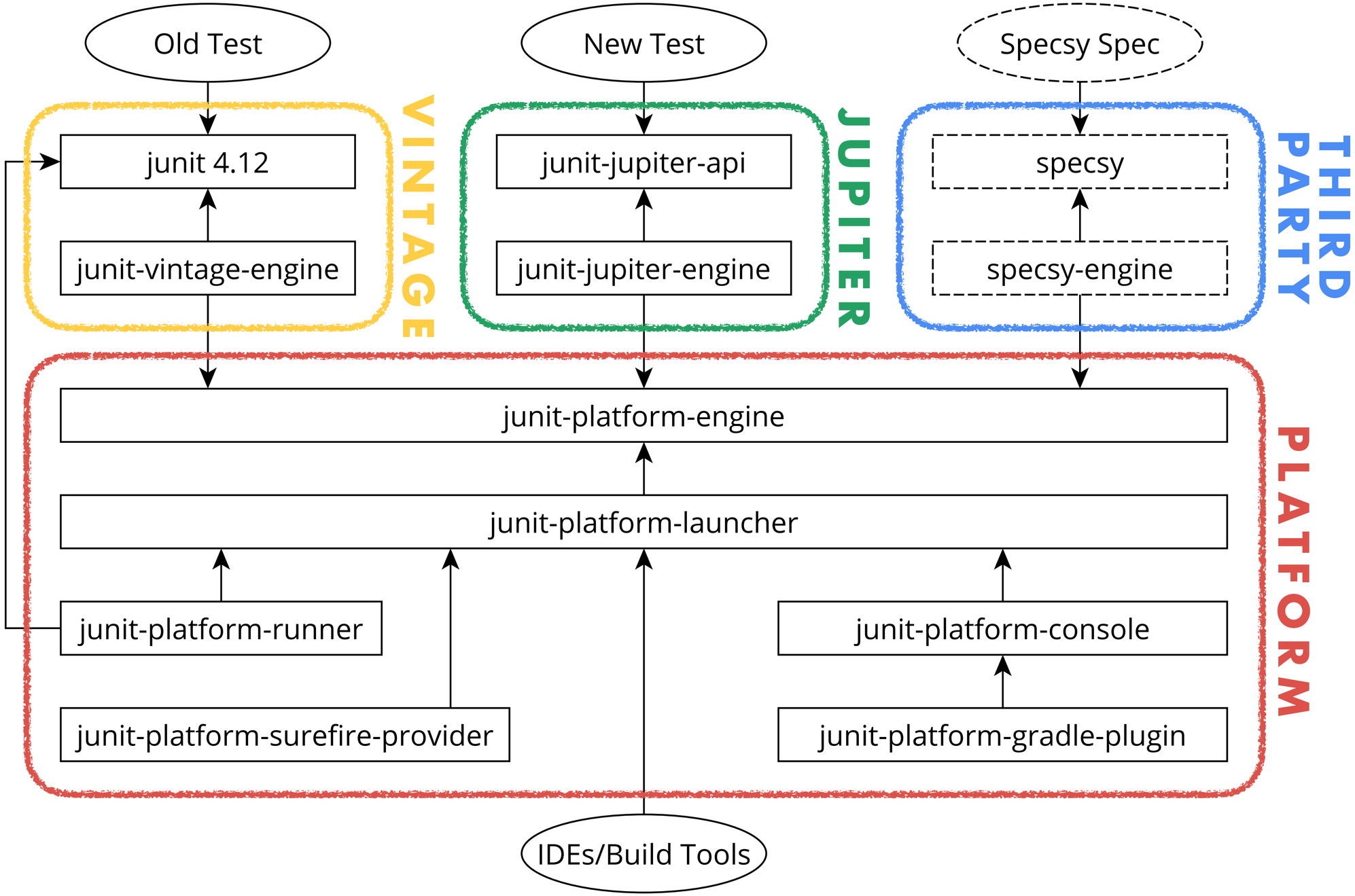


Schritt 3: Gleichzeitiger Einsatz mehrerer Test-Engines

- Jede Test-Engine hat ihre eigene API zur Testfallspezifikation
- Sevice-Provider-Mechanismus von Java zur Registrierung einer Test-Engine


junit-platform-launcher
ServiceLoaderTestEngineRegistry
Launcher
TestPlan discover(discoveryRequest)
void execute(TestPlan)
junit-platform-engine
<<interface>>
TestEngine
discover()
void execute()
create
junit-jupiter-api
<<annotation>>
org.junit.jupiter.api.Test
junit-jupiter-engine
JupiterTestEngine
junit-vintage-engine
VintageTestEngine
myproject
MyNewTest
@Test test1
@Test test1
MyOldTest
@Test test1
@Test test1
junit-4.12
<<annotation>>
org.junit.Test


Old Test
New Test
Specsy Spec
VINTAGE
JUPITER
THIRD PARTY
PLATFORM
IDEs/Build Tools


Old Test
New Test
Specsy Spec
junit 4.12
junit-vintage-engine
VINTAGE
junit-jupiter-api
junit-jupiter-engine
JUPITER
specsy
specsy-engine
THIRD PARTY
junit-platform-engine
junit-platform-launcher
junit-platform-runner
junit-platform-surefire-provider
junit-platform-console
junit-platform-gradle-plugin
PLATFORM
IDEs/Build Tools

Platform Usability

Test-Engines nutzen und selbst entwickeln

Warum braucht die Welt mehr als eine Test-Engine?

- Andere JVM-Sprache
- Anderes Spezifikationsmodell
- Anderes Ausführungsmodell

- Aber: Für *Erweiterungen* des Test-Modells genügt häufig eine Jupiter-Extension

Test-Engine benutzen

1. TestCompile-Dependency hinzufügen:

   `org.myorg:my-engine:x.y.z`

2. Tests schreiben

Test-Engine Kochrezept

1. Compile-Dependencies hinzufügen
2. `TestEngine`-Interface implementieren
3. Engine registrieren
4. Tests in IDE starten

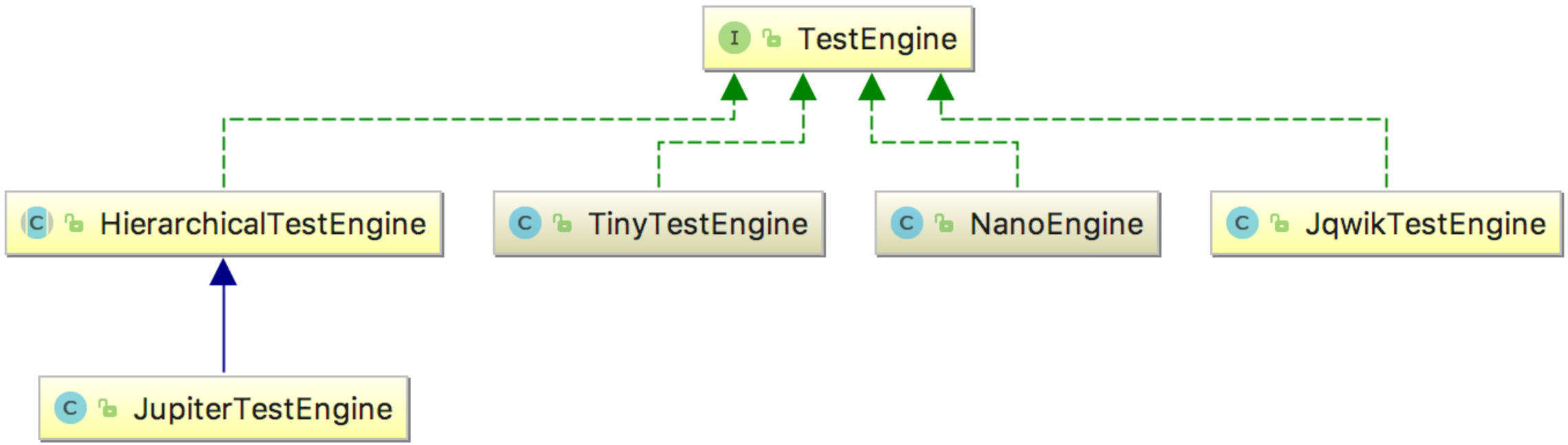
DEMO: Wir bauen unsere eigene Test-Engine

```
dependencies {
  compile("org.junit.platform:junit-platform-engine:1.2.0“)
  compile("org.junit.platform:junit-platform-commons:1.2.0")

  // For writing integration tests
  testCompile("org.junit.platform:junit-platform-launcher:1.2.0")
}
```

Test-Engine registrieren

/META-INF/services/
org.junit.platform.engine.TestEngine

```
nano.NanoEngine
```


TestEngine
HierarchicalTestEngine
TinyTestEngine
NanoEngine
JqwikTestEngine
JupiterTestEngine

TestEngine-Interface implementieren

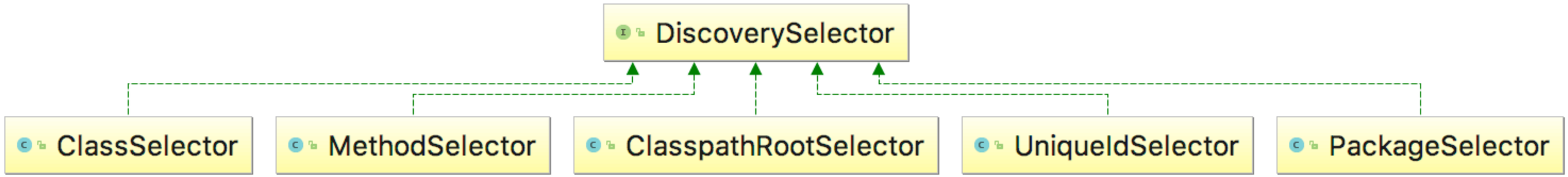
1. Create TestEngine class
2. Implement discover()
3. Implement execute()

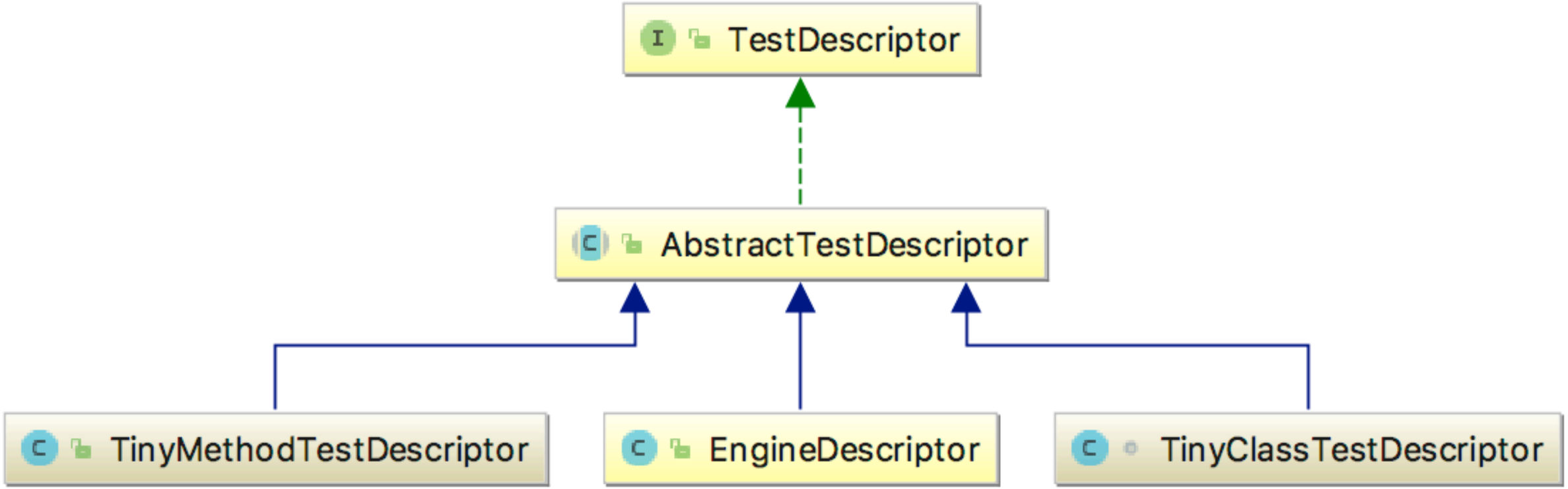
Was fehlt zu stärkerer Test-Engine?

- Nano kann von IDE ausgeführt werden, aber ist sehr beschränkt
- Test-Spezifikation durch Klassen und Methoden
  - Navigation in IDE's Test-Runner

TinyTest

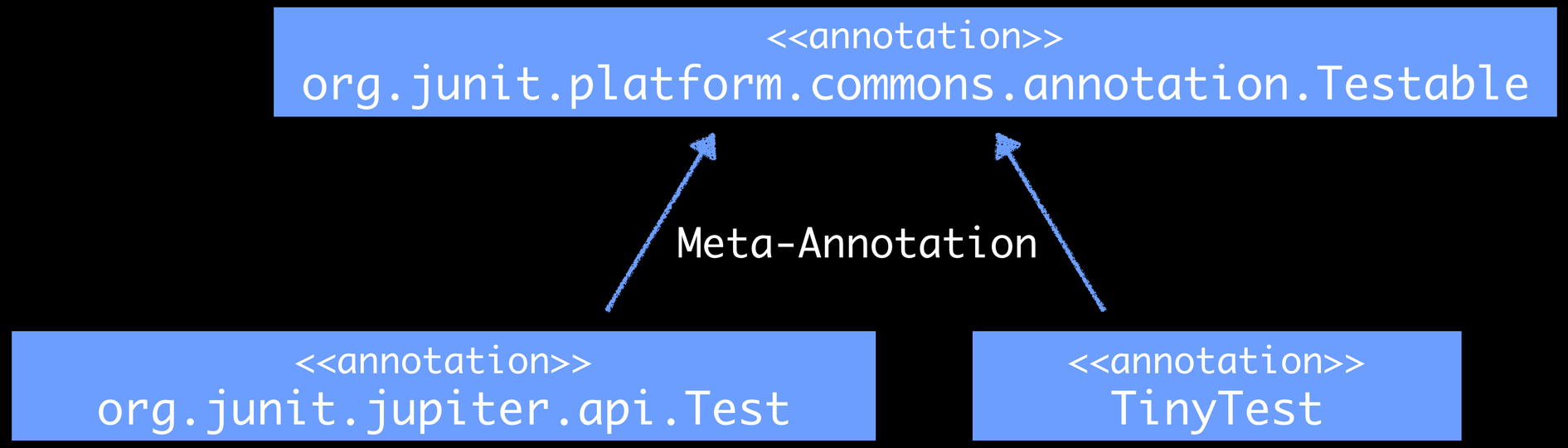
```
@TinyTest
public class A_tiny_test {

  final int theAnswer = 42;

  public boolean this_should_return_true() {
    return theAnswer == 42;
  }

  public boolean this_returns_FALSE() {
    return theAnswer == 43;
  }
}
```


DiscoverySelector
ClassSelector
MethodSelector
ClasspathRootSelector
UniqueIdSelector
PackageSelector


TestDescriptor
AbstractTestDescriptor
TinyMethodTestDescriptor
EngineDescriptor
TinyClassTestDescriptor

Tests in IDE identifizieren

- Theoretisch ist jedes zur Laufzeit ermittelbare Spezifikationsmodell umsetzbar
- Aber: Alles immer zu kompilieren ist für (manche) IDEs nicht möglich



Was geht noch?

- Ausführen von Dateien, URLs, und anderen Ressourcen
- jqwik.net: Property Testen in Java
- Specsy: Alles mit Lambdas
- Wrapper für Cucumber, Spock, Fitnesse etc

Jupiter Extension API

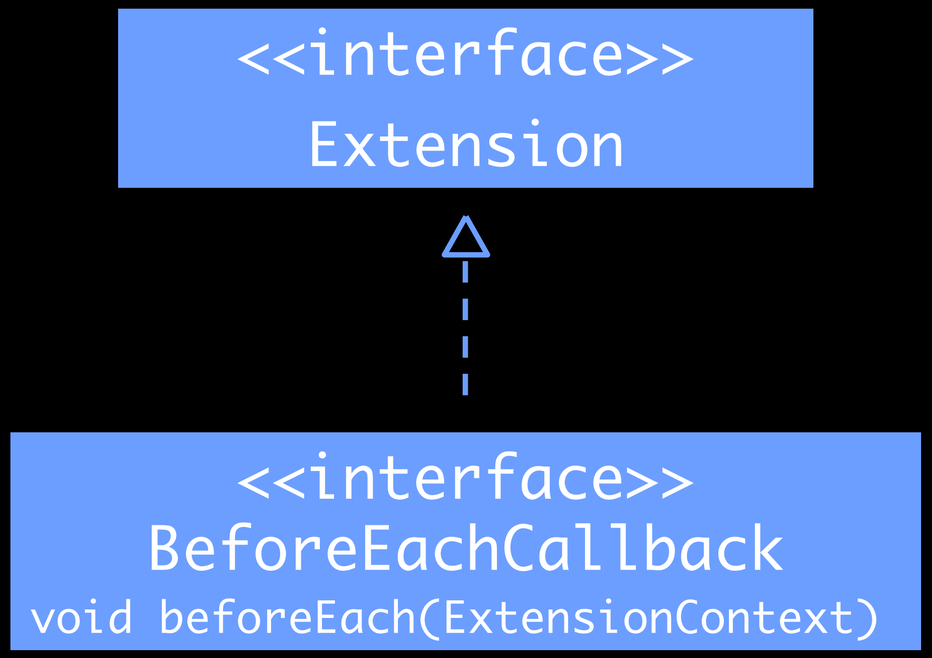
Jupiter Extensions

```java
@API(Experimental)
public interface Extension {
}
```

```java
@API(Experimental)
public @interface ExtendWith {

    Class<? extends Extension>[] value();

}
```

```java
@Test
@ExtendWith(CustomTypeParameterResolver.class)
@ExtendWith(CustomAnnotationParameterResolver.class)
void testMethodWithExtensionAnnotation(CustomType customType,
                                       @CustomAnnotation String value) {...}
```

Jupiter Callbacks



Hierarchy

Hierarchy Subtypes of Extension

Scope: Production

- Extension (org.junit.jupiter.api.extension)
  - AfterEachMethodAdapter (org.junit.jupiter.engine.execution)
  - BeforeEachMethodAdapter (org.junit.jupiter.engine.execution)
  - AfterTestExecutionCallback (org.junit.jupiter.api.extension)
  - ContainerExecutionCondition (org.junit.jupiter.api.extension)
    - DisabledCondition (org.junit.jupiter.engine.extension)
  - BeforeTestExecutionCallback (org.junit.jupiter.api.extension)
  - TestExecutionCondition (org.junit.jupiter.api.extension)
    - DisabledCondition (org.junit.jupiter.engine.extension)
  - AfterEachCallback (org.junit.jupiter.api.extension)
    - ExternalResourceSupport (org.junit.jupiter.migrationsupport.rules)
    - ExpectedExceptionSupport (org.junit.jupiter.migrationsupport.rules)
    - AbstractTestRuleSupport (org.junit.jupiter.migrationsupport.rules)
    - VerifierSupport (org.junit.jupiter.migrationsupport.rules)
  - BeforeAllCallback (org.junit.jupiter.api.extension)
  - AfterAllCallback (org.junit.jupiter.api.extension)
  - BeforeEachCallback (org.junit.jupiter.api.extension)
    - ExternalResourceSupport (org.junit.jupiter.migrationsupport.rules)
    - AbstractTestRuleSupport (org.junit.jupiter.migrationsupport.rules)
  - TestTemplateInvocationContextProvider (org.junit.jupiter.api.extension)
    - RepeatedTestExtension (org.junit.jupiter.engine.extension)
    - ParameterizedTestExtension (org.junit.jupiter.params)
  - TestExecutionExceptionHandler (org.junit.jupiter.api.extension)
    - ExpectedExceptionSupport (org.junit.jupiter.migrationsupport.rules)
    - AbstractTestRuleSupport (org.junit.jupiter.migrationsupport.rules)
      - TestRuleMethodSupport (org.junit.jupiter.migrationsupport.rules)
      - TestRuleFieldSupport (org.junit.jupiter.migrationsupport.rules)
  - ParameterResolver (org.junit.jupiter.api.extension)
    - ParameterizedTestParameterResolver (org.junit.jupiter.params)
    - TestInfoParameterResolver (org.junit.jupiter.engine.extension)
    - RepetitionInfoParameterResolver (org.junit.jupiter.engine.extension)
    - TestReporterParameterResolver (org.junit.jupiter.engine.extension)

Tool-Support: State of the Union

- IntelliJ
  - Unterstützung seit beinahe 2 Jahren
- Eclipse
  - Eingebaut seit 4.7.1
- Gradle
  - Nativ unterstützt seit 4.7
- Maven
  - Nativ unterstützt seit kurzem

https://github.com/jlink/junit5-the-platform

http://johanneslink.net/downloads/junit5-the-platform.pdf

Links

- Artikel über JUnit-5-Architektur
  https://blog.codefx.org/design/architecture/junit-5-architecture/
- Artikel über Jupiter-Extension-Modell
  https://blog.codefx.org/design/architecture/junit-5-extension-model/
- Framework Design Principles: http://www.davidtanzer.net/node/118
- JUnit 5: http://junit.org/junit5
- Jqwik-Engine: http://jqwik.net