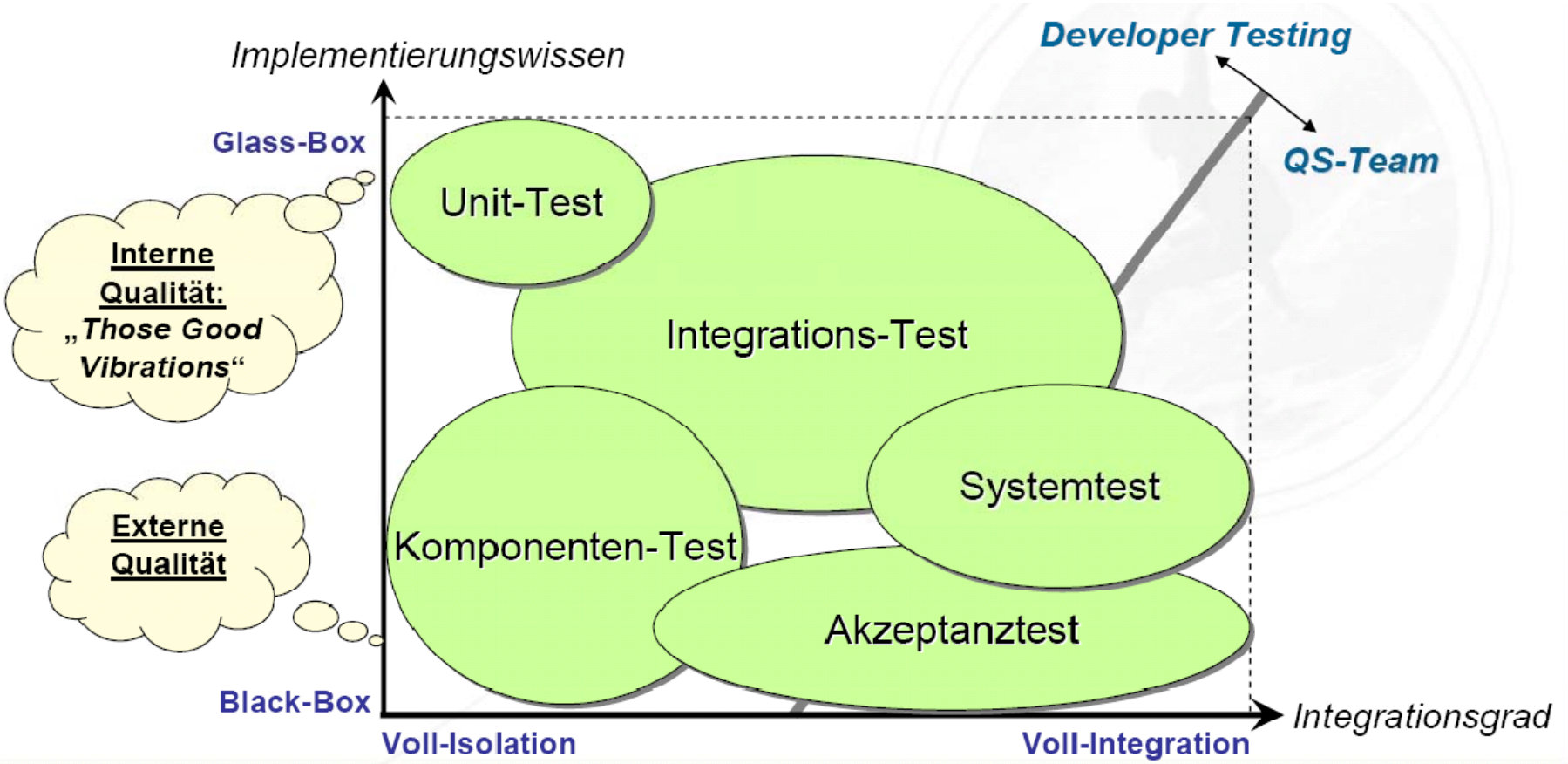
TestNG

Next Generation

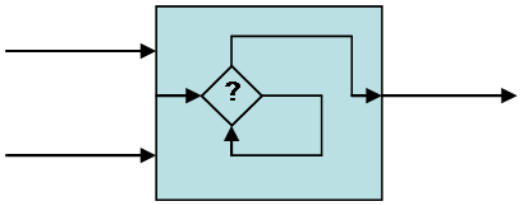
JAVA TESTING

Testen im Software- Lebenszyklus



White-Box vs Black-Box Testing

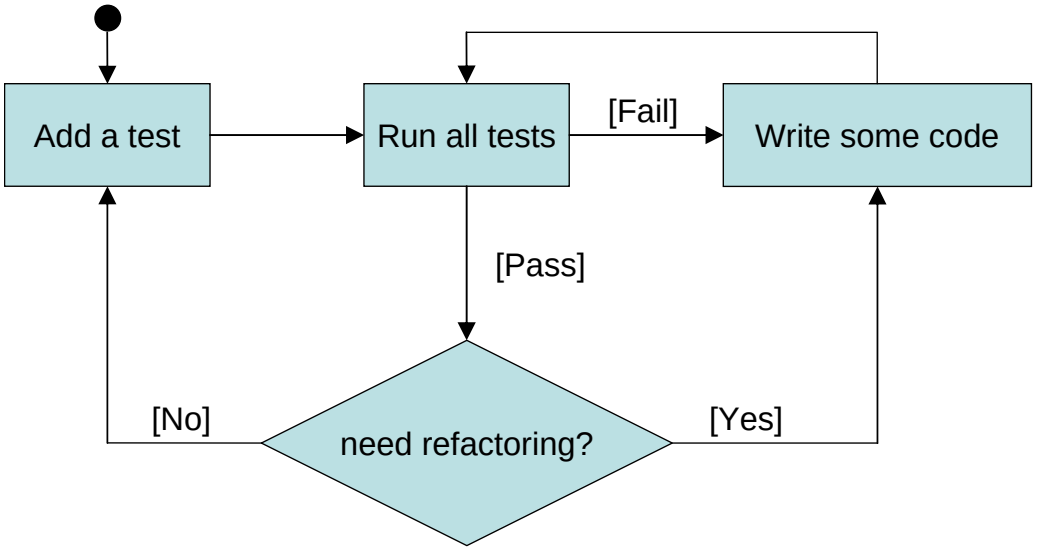
White-Box-Testing



Black-Box-Testing

Test-Driven Development (TDD)

testgetriebene Entwicklung
zuerst werden die Unit-Tests erstellt.
Dann wird implementiert bis alle Tests durchlaufen.
bekannteste Technik: Extreme Programming.



Was sind Developer Tests?

- ***Alle Tests, die ein Entwickler sinnvollerweise selbst macht.***
- ***korrekte Funktionalität des eigenen Codes sicherstellen***
- automatisierte Unit-Tests, auf Stufe Klassen und Methoden
- decken jedoch keinesfalls die kompletten Funktionen oder Anforderungen der Software als Ganzes ab

Regressionstest

- Regression (lat.) = Rückschritt
- versteht man das Durchführen eines Sets von Testfällen, um *Nebenwirkungen* von *Modifikationen* in bereits getesteten Teilen der Software aufzuspüren.
- Voraussetzung für ein ‚unbeschwertes‘ Refactoring

Unit Testing Frameworks

Check

Boost Test Library

Test::Unit

PyUnit

QuickTest

CuTest

csUnit

GNU Autounit

CPPUnit

GUnit

JTiger

CUnit

Unit++

CppUnitLite

JUnit

TUT

NUnit

TestNG

C++test

Cgreen

UnitTest++

NanoCppUnit

cxxtest

MbUnit

MinUnit

JUnit

- JUnit ist der Platzhirsch im Umfeld der Automatisierungs-Frameworks für Unit-Tests
- Einfach erlernbar: test methods, classes suites
- Entwicklung der Konzepte ursprünglich für Smalltalk (SUnit)
- Entwicklung seit 1998
- Hauptentwickler sind Erich Gamma und Kent Beck
- Umsetzung für viele Sprachen verfügbar:
- *CppUnit (C++), PyUnit (Python), NUnit (.NET), PHPUnit (PHP)*
- Viele Erweiterungen für einzelne Testarten und zur Arbeitserleichterung verfügbar
  ant tasks, report Generatoren, DB-, GUI-Testing, …
- Aktuell: Version 4.4 (Juli 2007)

Einleitung TestNG

- seit 2003, hauptsächlich von Cédric Beust entwickelt
- Weil nach JUnit3 seeehr lange nichts mehr ging
- basiert auf Annotations (sowohl Javadoc als JDK5)
- flexible plug-in API (Erstellung von Reports, …)
- Hat Junit-Kompatibilitäts-Modus
- Junit Tests einfach in TestNG migrierbar
- Ein Testing-Framework für alle Testebenen.
- TestNG = JUnit 4
- + Nebenläufigkeitstests
- + Testkonfiguration durch XML (Parameter, Suiten, …)
- + Gruppen (*Groups*) als zusätzliches Strukturierungselement
- + Automatisch generierte Testsuite aller fehlgeschlagenen Tests
- + Feingranularere Isolationsklammern
- + Unterstützung von Testsequenzen und abhängigen Testfällen
- + Factory Mechanismus, der eigene Test-Instanziierung zulässt.
- + Erwartete Exceptions (mehrere pro Methode)
- - Pseudo-Isolation durch mehrfache Instanziierung der Testklasse

Vergleich JUnit und TestNG

Feature	JUnit4	TestNG
Konfiguration	Annotations	Annotations und XML
Werkzeugintegration	Ant, Maven, Eclipse, Idea und viele andere wie NetBeans	Ant, Maven, Eclipse, Idea
Organisationsstrukturen	Suite, Test	Suite, Test, Gruppe
Instanziierung von Testklassen	Für jede einzelne Testmethode	Ein Mal pro Testablauf
Parametrierte Testfälle	Ja	Ja
Gruppierung von Tests	Nein	Ja
Definition von Abhängigkeiten zwischen Testmethoden und Gruppen	Nein	Ja
Ausschließlich fehlgeschlagene Testfälle wiederholen	Nein	Ja
Testunterstützung für nicht funktionale Anforderungen	Time-out	Time-out, Stress-Test
Logging und Reporting	Nein	Ja

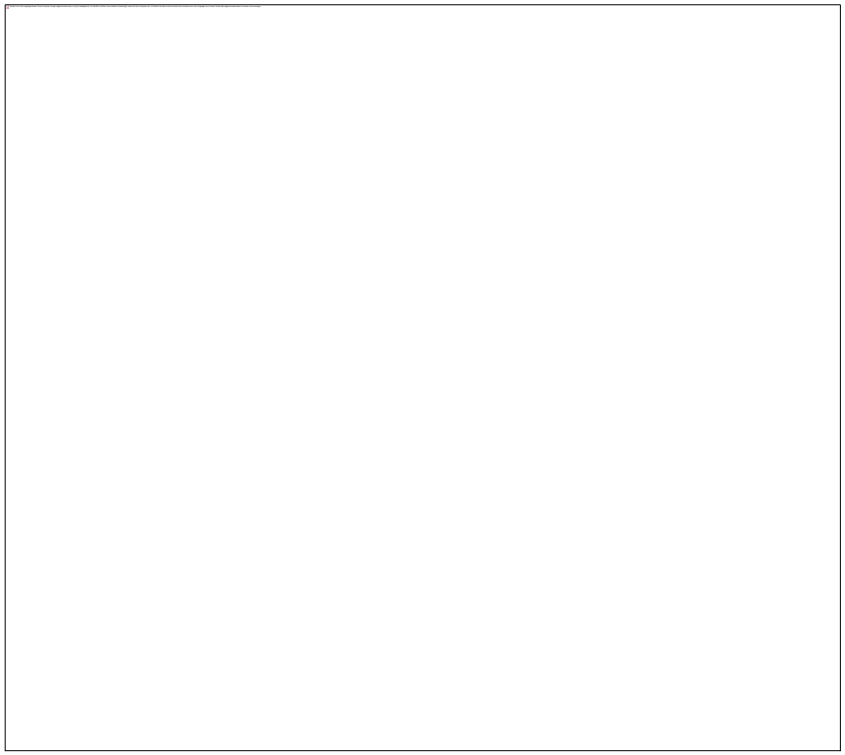
TestNG Beispiel (JDK 5)

```
import org.testng.annotations.*;
public class SimpleTest
{
  @BeforeClass
  public void init()
  {
    // dieser Code wird ausgeführt,
    // bevor der Test ausgeführt wird
    // vergleich JUnit setUp()
  }

  @Test(groups = { "functest" })
  public void serverIsRunning()
  {
    // die eigentliche Test-Logik
  }
}
```

TestNG Beispiel

- Müssen von keiner spezifische Klasse ableiten
- Test-Methoden müssen nicht mehr mit “test” beginnen
- Konfigurations-Methoden können beliebig benannt werden
  (vergleiche JUnit3: setUp()/ tearDown().
- Beliebig viele Konfig-Methoden erlaubt
  können sich sowohl auf Methoden wie auch Klassen beziehen

TestNG Terminologie



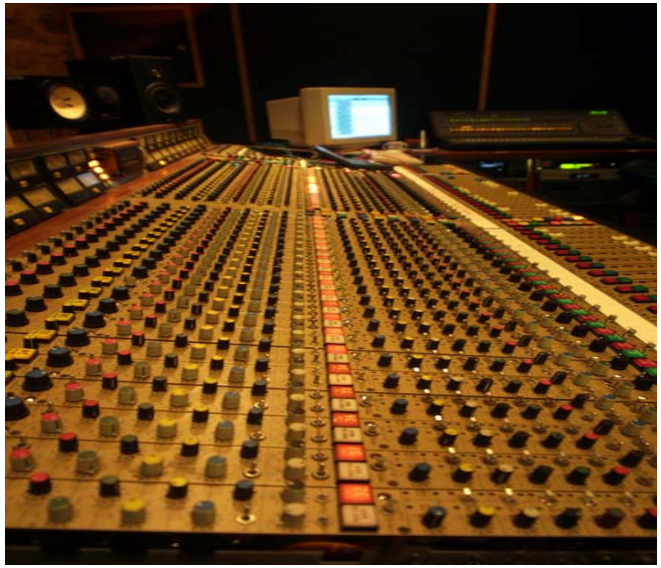
```
@BeforeTest
public void initTest() { … }

@AfterSuite
public void cleanUp() { … }
```

Je nach TestNG-Annotation kann die Granularität auf Stufe Suite / Test / Klasse / Methode liegen

TestNG annotations

Konfigurations-Methoden:

- @BeforeMethod/AfterMethod **(setUp/tearDown)**
- @BeforeClass/@AfterClass **(kein JUnit Equivalent)**
- @BeforeSuite/@AfterSuite **(kein JUnit Equivalent)**
- @BeforeTest/@AfterTest **(kein JUnit Equivalent)**



- Es können beliebig viele Konfigurations-Methoden definiert werden
- Konfigurations-Methoden können Gruppen angehören und von anderen Gruppen abhängig sein

TestNG-Annotations

Annotation	Bedeutung
`@BeforeSuite`	Die annotierte Methode läuft, bevor alle Tests dieser Suite laufen.
`@AfterSuite`	Die annotierte Methode läuft, nachdem alle Tests dieser Suite laufen.
`@BeforeTest`	Die annotierte Methode läuft vor einem Test.
`@AfterTest:`	Die annotierte Methode läuft nach einem Test.
`@BeforeGroups:`	Die annotierte Methode läuft, bevor einer der Gruppentests läuft.
`@AfterGroups:`	Die annotierte Methode läuft, nachdem einer der Gruppentests läuft.
`@BeforeClass:`	Die annotierte Methode läuft, bevor alle Testmethoden dieser Testklasse laufen.
`@AfterClass:`	Die annotierte Methode läuft, nachdem alle Testmethoden dieser Testklasse laufen.
`@BeforeMethod`	Die annotierte Methode läuft, bevor eine Testmethode läuft.
`@AfterMethod`	Die annotierte Methode läuft, nachdem eine Testmethode läuft.
`@DataProvider`	Markiert eine Methode als Datenlieferant für eine Testmethode.
`@Factory`	Markiert eine Methode als Factory, die von TestNG als Testklasse genutzt wird.
`@Parameters`	Beschreibt, wie Parameter einer Testmethode übergeben werden.
`@Test`	Markiert eine Klasse oder eine Methode als Teil eines Tests.

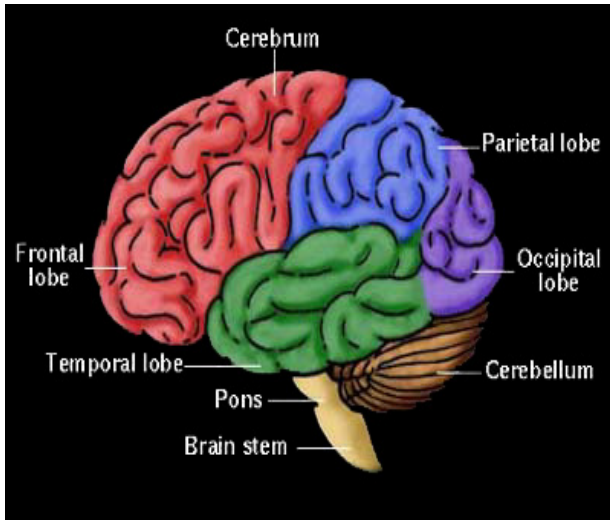
TestNG annotations

`@Test`

- `groups`
  Definiert die Gruppenzugehörigkeit dieser Methode
- `parameters`
  Die Parameter welche an die Test-Methode weitergereicht werden, so wie sie im File testng.xml vorgefunden werden.
- `dependsOnGroups`
  Die Liste der Gruppen von welcher diese Methode Abhängig ist.
  TestNG stellt sicher, dass alle Methoden welche von diesen Gruppen Abhängig sind vorgängig aufgerufen werden.
- `timeOut`
  Wie lange soll TestNG warten, bis die Test-Methode als failed betrachtet wird.

```
@Test(groups = { “functional” },
      timeOut = 10000,
      dependsOnGroups = { “serverIsUp” })
public void sendHttpRequest() {
  // …
}
```

testng.xml



Hier wird der Testverlauf zentral parametriert:

- Die Test Methoden, Klassen, Packages
- Welche Gruppen sollen ausgeführt werden
- (include-groups)
- Gruppen ausschliessen (exclude-groups)
- Zusätzliche Gruppen festlegen (“groups of groups”)
- Sollen die Tests parallel abgearbeitet werden
- Parameter vorgeben (-> Testdaten definieren)
- JUnit mode

Beispiel testng.xml

```
<test name=“Simple">
    <groups>
      <run>
        <include name=“functest"/>
      </run>
    </groups>
    <classes>
      <class name=“SimpleTest" />
    </classes>
  </test>
```

Hinweis: *testng.xml ist optional*
Alternativen sind ant, Kommando-Zeile

Parameter expectedExceptions

```
public int getConn(String arg)
{
   if(arg==null)
      throw new IllegalArgumentException();
   return ...;
}
```

```
@Test(expectedExceptions={IllegalArgumentException.class})
public void checkGetConn()
{
   assert tmp.getConn(null,...)==...;
}
```

Total tests run: 1, Failures: 0, Skips: 0

Expected an exception in test method de.test..

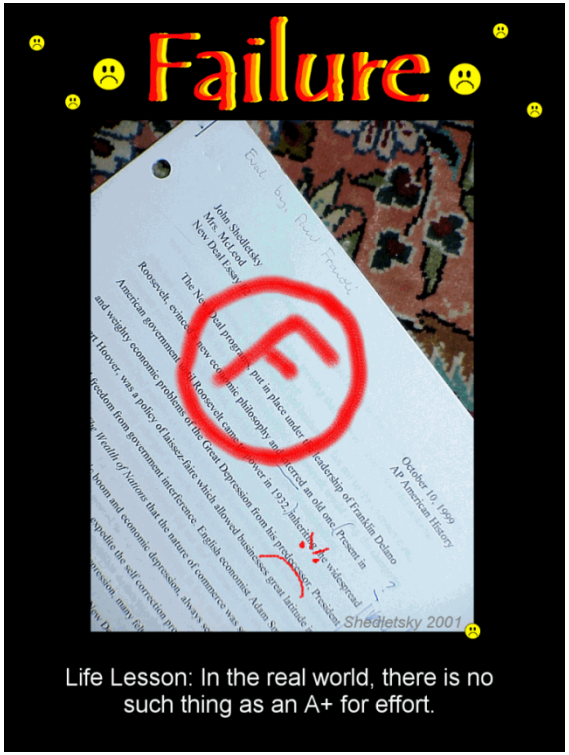
Gescheiterte Tests wiederholen

- Es wird viel Zeit damit verwendet gescheiterte Tests zu wiederholen
- TestNG registriert die gescheiterten Tests eines Test-Laufs und kann exakt nur diese nochmals starten
- ➔ testng-failed.xml

- Typisches Vorgehen:

  $ java org.testng.TestNG testng.xml

  $ java org.testng.TestNG testng-failed.xml



Parameter dataProvider

- Wenn Datenübergabe aus testng.xml nicht ausreicht
- DataProviders ermöglichen die Trennung zwischen Test-Daten und Test-Logik
  - Data Driven Testing
- Test-Daten können aus folgenden Quellen stammen: Java, Flat-File, Datenbank, Netzwerk, etc…
- beliebig viele DataProvider
  - “korrekte-strings-provider”, “gemeine-strings-provider”
- Typen:
  - feste Argumentanzahl
  - Iteratoren
  - parametrisierte DataProvider
- Test Factories setzen dem noch eins drauf !

Beispiel dataProvider

Beispiel: Ermittlung des HTTP Error-Codes anhand des ’Browser Strings’

```
@Test
public void verifyUserAgentSupport()
{
assertEquals(200,getReturnCodeFor("MSIE"));
assertEquals(200,getReturnCodeFor(FireFox);
assertEquals(301,getReturnCodeFor(Safari);
assertEquals(-1, getReturnCodeFor(WAP));
}
```

```
@Test(dataProvider = “user-agents”)
public void verifyUserAgent
            (String s, int code) {
  assert getReturnCodeFor(s) == code;
}
```

```
@DataProvider(name = “user-agents")
public Object[][] createUserAgents()
{
  return new Object[][] {
    new Object[] { “MSIE”, 200},
    new Object[] { “FireFox”, 200};
    new Object[] { “Safari”, 301};
    new Object[] { “WAP”, -1};
}
```

Ausschliessen von Gruppen

- Manchmal schafft man es nicht alle Tests zu fixen
- **JUnit:** entsprechend Tests auskommentieren und hoffen, dass jemand diese frühzeitig vor Auslieferung wieder aktiviert
- **TestNG:** erstellen einer “broken” Gruppe und diese bei allen Test-Runs excluden. Alle fehlerhaften Tests dieser Gruppe zuweisen
- **Später:** alle Tests aus der “broken” Gruppe lokalisieren, Code fixen und betreffende Test aus dieser Gruppe herausnehmen

```
<test name=“DBTest">
   <groups>
     <run>
       <exclude name=“broken.*" />
       <include name=“functional.*” />
     </run>
   </groups>
```

Abhängige Methoden (depend)

- Problem: einige Tests-Methoden sind vom Erfolg von vorgängigen Tests abhängig.
- Beispiel: testen eines Web-Servers:
  - Eine Test-Methode welche den Server startet (launch())
  - Eine Test-Methode welche den Server anpingt (ping())
  - 20 Methoden welche verschiedene Aspekte des Servers prüfen (test1() … test20())
- Problem: Server ist gestartet aber der ping() scheitert
- Dieses Szenario ist mit JUnit nicht einfach abzuhandeln
  JUnit Resultat: 1 PASSED and 21 FAILURES
- Möglichkeit Ablauf-Reihenfolge zu steuern
  zuerst werden die abhängigen Methoden ausgeführt
- Gescheiterte Tests infolge nicht erfüllter Dependencies werden als ‘skiped’ und nicht ‘failed’ ausgewiesen

Abhängige Methoden (depend)

- Umsetzung des Web-Server Testing Beispiels:
  - Dependencies: launch → init → tests

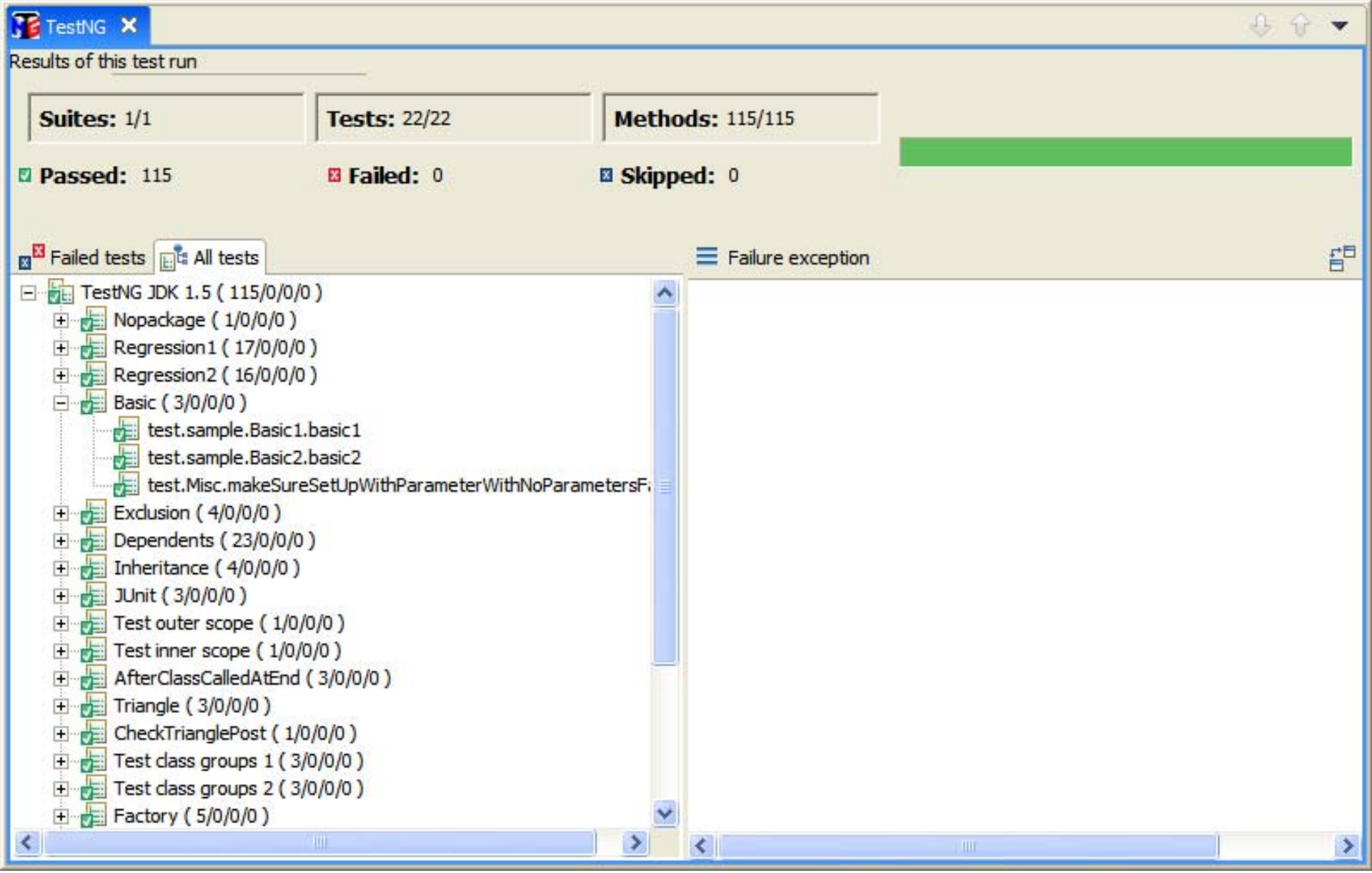
```
@Test(groups = “launch”)
public void launchServer() {…}

@Test(groups = “init”,
      dependsOnGroups = { “launch” })
public void ping() { …}

@Test(dependsOnGroups = { “init” })
public void test1() { … }
```

Outcome: 1 SUCCESS, 1 FAIL, 20 SKIPS

Eclipse und IDEA

plug-ins für Eclipse und IDEA:

- Starten von Test- Methoden, Klassen, Gruppen (testng.xml)
- Einfache Selektion von Gruppen und Suite-Files
- Anzeige des vertrauten rot/grün Stati
- Direktes ‘Anspringen’ von Test-Fehlern
- Automatische Konvertierung von JUnit nach TestNG


TestNG
Results of this test run
Suites: 1/1
Tests: 22/22
Methods: 115/115
Passed: 115
Failed: 0
Skipped: 0
Failed tests
All tests
Failure exception
TestNG JDK 1.5 (115/0/0/0)
Nopackage (1/0/0/0)
Regression1 (17/0/0/0)
Regression2 (16/0/0/0)
Basic (3/0/0/0)
test.sample.Basic1.basic1
test.sample.Basic2.basic2
test.Misc.makeSureSetUpWithParameterWithNoParametersF
Exclusion (4/0/0/0)
Dependents (23/0/0/0)
Inheritance (4/0/0/0)
JUnit (3/0/0/0)
Test outer scope (1/0/0/0)
Test inner scope (1/0/0/0)
AfterClassCalledAtEnd (3/0/0/0)
Triangle (3/0/0/0)
CheckTrianglePost (1/0/0/0)
Test class groups 1 (3/0/0/0)
Test class groups 2 (3/0/0/0)
Factory (5/0/0/0)

Integration von Frameworks

- Maven plug-in (v1 and v2)
- Spring
- DBUnit

Vergleich: Popularität

Treffer am 11.4. und 28.4. 2006

- „JUnit“: **7‘300‘000, 7‘660‘000** (+4,9%)
- „JUnit 4“: **40‘500, 52‘200** (+28,9%)
- „TestNG + Java“: **189‘000, 214‘000** (+13,2%)

Treffer am 1.2.2008

- „JUnit“: **3'330'000**
- „JUnit 4“: **42'800**
- „JUnit4“: **64'600**
- „TestNG“: **251'000**

Weiterführende Themen

- Mock Objects
- Metriken (Lines of Code, Kopplung, Kohäsion,
- *Coverage Tools*
  (JCoverage, Clover, EMMA, Coverlipse)

Links

- TestNG Homepage, www.testng.org
- http://en.wikipedia.org/wiki/Unit_testing
- JUnit Homepage, www.junit.org
- Buch ‘NextGeneration Testing mit TestNG & Co‘ entwickler.press
  ISBN-10 3-939084-02-6


Sch(l)uss & Aus