ThinPrint®

www.thinprint.com

Themen

Unit-Tests am Beispiel der JUnit

Dozent

Danny Preußler

Unit-Tests

- Mit Unit-Tests ist es so wie mit dem Essen von Brokkoli:
  Alle wissen, dass es gut wäre, aber kaum jemand tut es.
  (aus „Pragmatic Unit Testing with JUnit„)

Unit-Tests

- Eingeführt durch Kent Beck mit einem Testframework für Smalltalk: SUnit
- Zusammen mit E.Gamma 1999 portiert nach Java → JUnit

- JUnit heute die bekannteste Unit-Test Bibliothek und ein industrieweiter Standard
- Ähnliche Frameworks für fast alle Sprachen verfügbar (NUnit, VbUnit, CppUnit ..)

Was sind Unit-Tests

- Entwicklertests
- Testen auf Modul- oder Funktionsebene
- Oft in Verbindung mit Extreme-Programming „Test first“

Test Early and Often

- The earlier you test
  - The sooner you find the problems
  - The more likely you are to find them
  - And the easier it is to fix them
- Adios debugger

Was sind Unit-Tests?

- Eigentlich nichts neues:
  - Programme enthielten oft Beispielcode zum testen

```
public void DoSelfTest()
{
   ISINVALID("THPR-0002-1-MM9PFF-9TSF", 0);  // Fehler bei SR-2 (Fehler falscher Bugfix)
   ISINVALID("THPS-0002-1-MQ7PUD-AAVF", 0);  // Fehler bei SR-2 (Fehler falscher Bugfix)
   ISINVALID("THPR-0002-1-MU98FF-J57X", 0);  // Fehler vor SR-2 (Fehler mod bei encrypt/decrypt)
   ISINVALID("TPCB-0001-1-TEST01-1WS7", 0);    // gar keine Fehler

   INSTALL_LIC("THPR-0002-1-MM9PFF-9TSF", 0, "S-1-5-21-746137067-179605362-839522115", "413ca03a");
```

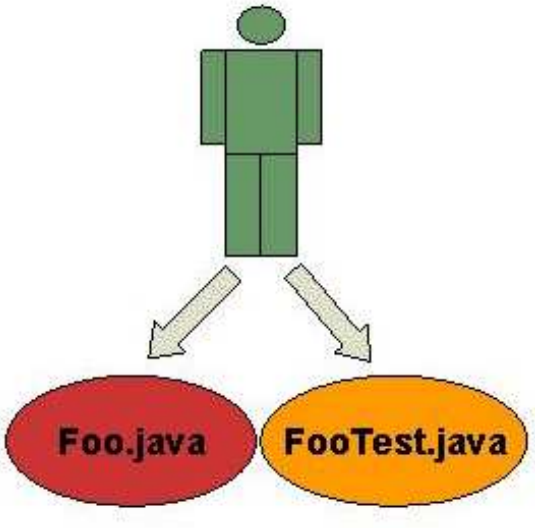
```
isLicenseKeyInvalid(THPR-0002-1-MM9PFF-9TSF): 0 expected: 0
isLicenseKeyInvalid(THPS-0002-1-MQ7PUD-AAVF): 0 expected: 0
isLicenseKeyInvalid(THPR-0002-1-MU98FF-J57X): 0 expected: 0
isLicenseKeyInvalid(TPCB-0001-1-TEST01-1WS7): 0 expected: 0
CLVInstallLicense(THPR-0002-1-MM9PFF-9TSF): 0 expected: 0
```

Was sind Unit-Tests?

- Neu ist:
  - einheitliches Framework für alle
  - Leicht zu benutzen
  - Unterstützt durch viele IDEs, Buildskripten,...

Wie schreibt man Unit-Tests?

- Erzeuge neue Testklasse für zu jede zu testende Klasse
  Klasse erbt von TestCase auf JUnit Framework



Wie schreibt man Unit-Tests?

- Name der neuen Klasse endet mit „Test“:
  *class JCBURLDecoder**Test** extends TestCase*
- Zu testende Methoden beginnen mit „test“
  *public void **test**Decode*()
- Teste Rückgabewerte der Methoden mittels Assert-Funktionen, z.B.:
  void assertEquals(boolean expected, boolean actual)
  void assertEquals(String expected, String actual)
  void assertNotNull(Object object)



Wie schreibt man Unit-Tests?

```
public class JCBURLDecoderTest extends TestCase
{

    /*
     * Test method for 'com.thinprint.j2me.io.JCBURLDecoder.decode(String)'
     */
    public void testDecode()
    {
        try
        {
            assertEquals("hallo", JCBURLDecoder.decode("hallo"));
            assertEquals("hallo du", JCBURLDecoder.decode("hallo%20du"));
            assertEquals("äüöß", JCBURLDecoder.decode("%E4%FC%F6%DF"));
        }
        catch(UnsupportedEncodingException exc)
        {
            fail(exc.toString());
        }
    }
```

Vorteile von Unit-Tests:

- Nie wieder „Gestern ging‘s noch!“
- Saubererer Programmierstil (klare Funktionen mit klarem Aufruf)
- Erleichtern Refactoring
- Sicherheit bei Änderungen -> gutes Gefühl
- Ist Dokumentation und bestimmen Sollzustand
- Testen ohne Gesamtsystem zu benötigen
- Testen von Sonderfällen oder Funktionen, die schwer von außen testbar sind
- Probleme schon vor Testfeld erkennen

Was sollte man testen?

- Test je Klasse mit den wichtigsten Methoden
- Tests für spezielle Funktionalitäten
- Tests für offizielle Bugs („keine Wiederkehr“)
- Keine Tests für simple Funktionen (get/set)
- Tests sollten helfen, nicht aufhalten

Wie testet man …

Funkionen, die internen Status verändern?

- Status abfragen und vergleichen
- Ableiten und Aktionen durch Überladung verifizieren

Wie testet man …

Funktionen, die API- oder Spezial-Funktionen benutzen?

- Wrapper um Funktion
  → kann in Test überladen werden
- Hilfsklassen benutzen und übergeben lassen
  → kann im Test durch Dummy ausgetauscht werden

Testen über den Tellerrand:

- Testen von großen Systemen mit Unit-Tests als gemeinsamer Basis

- Exploratives Unit-Testen:
  Zur Fehlersuche statt traditionellem Messageboxen und/oder Debuggen iterativ Unit-Tests schreiben

Grenzen von Unit-Tests

- Tests sind manuell:
  - Kosten Zeit
  - Gefahr, dass mehr Zeit für Tests als für Entwicklung drauf geht
  - Tests müssen nicht nur geschrieben sondern auch gepflegt werden
- Testfälle vom Programmierer des Testcodes:
  - problematische Fälle in beiden übersehbar?
  - Codeabdeckung evtl. Ungenügend, nicht praxisorientiert?
- Trotz Überladungen, Dummyklassen (Stubs) und generischer Mockobjekte: nicht alles testbar ohne Gesamtsystem
- Ersetzen nicht den Test durch die QA



Wie fängt man an?

Neuer Code:

- von vornherein auf Testen ausrichten und mit Tests bestücken

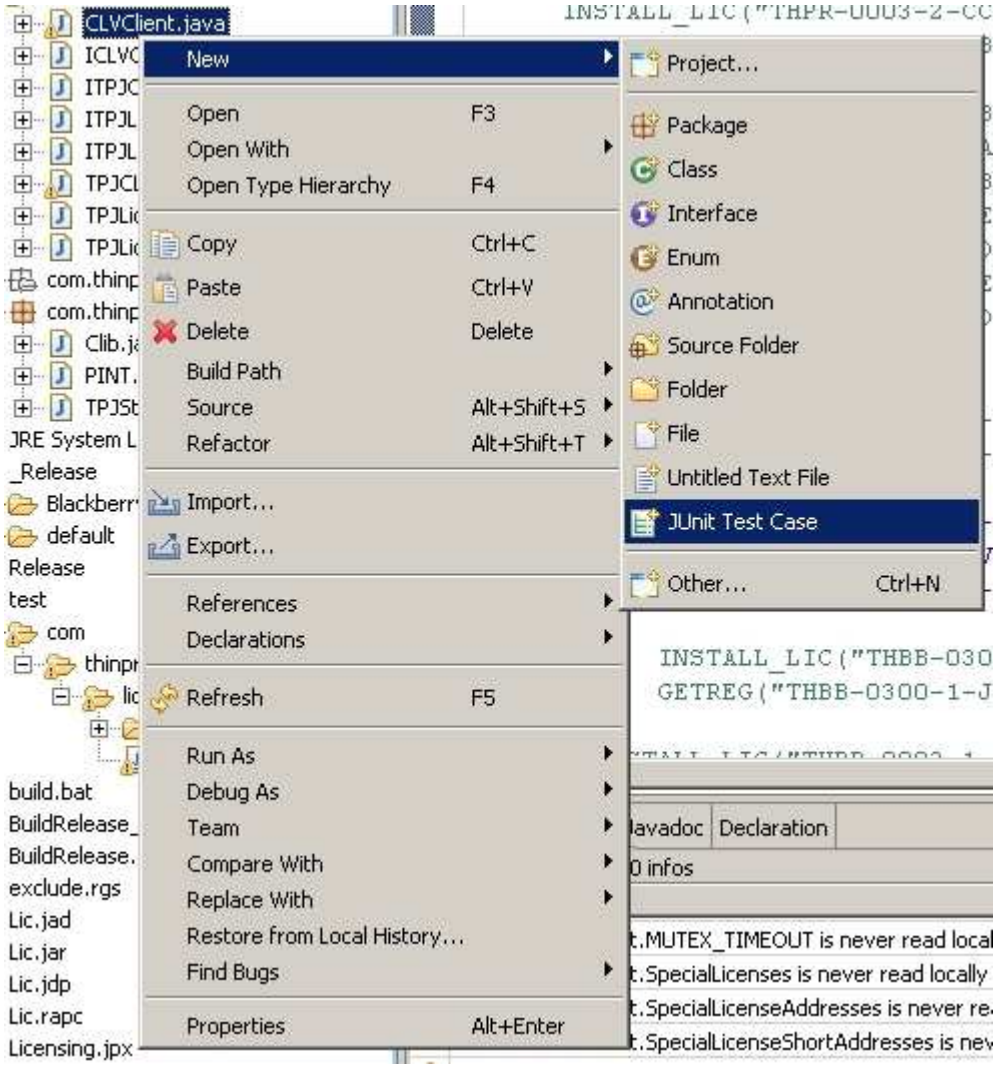
Alter Code:

- Bugs (wenn möglich) durch Tests absichern
  z.B. bei NullPointer-Problemen, Bufferüberläufen usw
- Bei Umbauten / Refactoring: Tests einpflegen

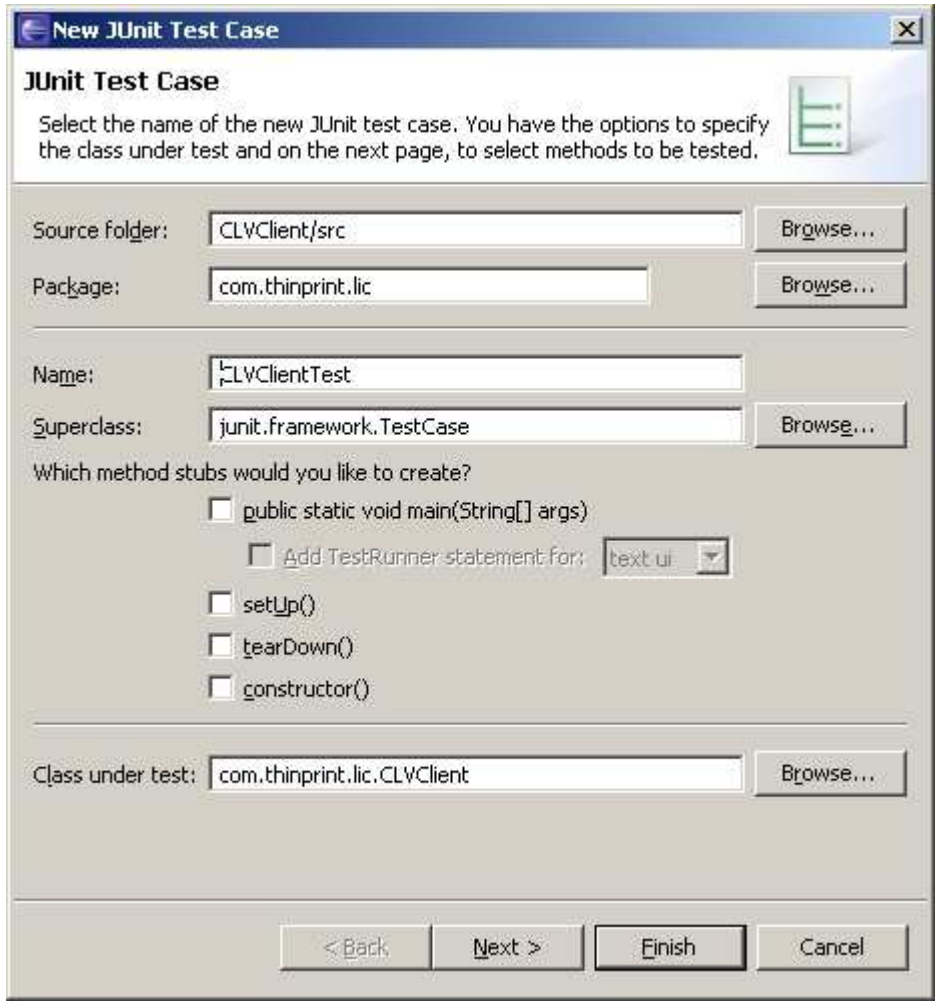
Wie fängt man an?

```
/*
 * checks problem for BUG-4316
 *
 * problem was: creating of list from addrbook should only
 * happen if user selects "from addressbook"
 * because long addressbooks could cause waiting time
 * -> should not happen if user wants to select "use once"
 *
 * @author dapre
 *
 */
public class CheckFaxNoDlgWithLongListTest extends TestCase
{
     public void testgetFaxNoFromUser()
.....
```

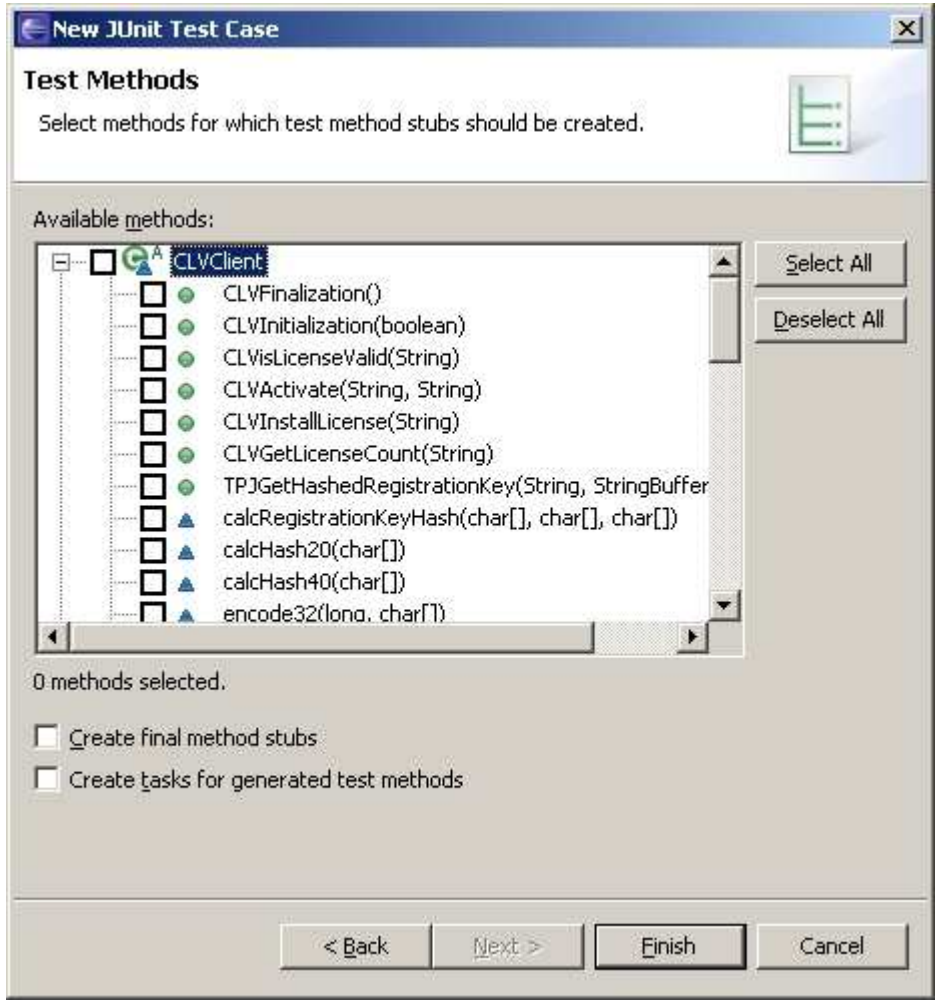
Wie fängt man an?



Wie fängt man an?



Wie fängt man an?



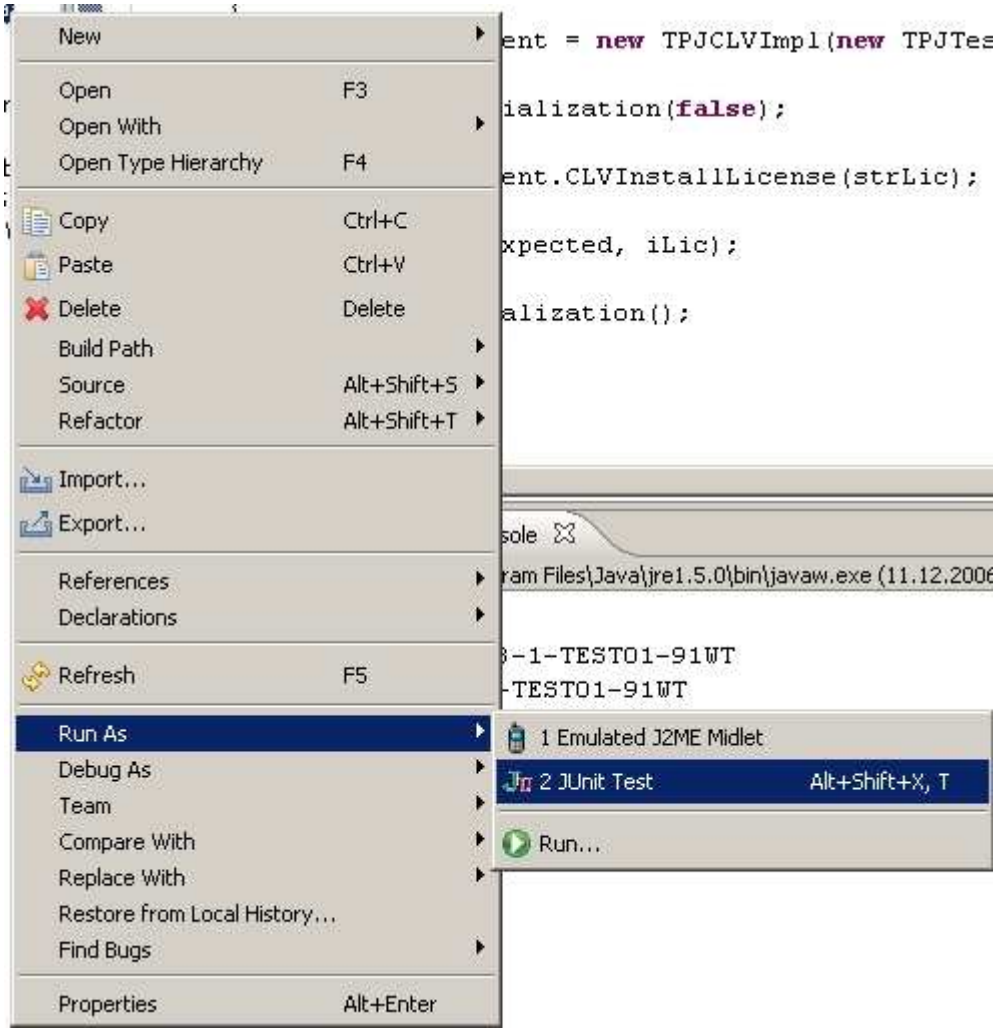
Wie fängt man an?

```
/*
 *  Test method for 'com.thinprint.lic.CLVClient.CLVInstallLicense(String)'
 *
 *  falsche checksum
 */
public void testCLVInstallLicenseWithWrongCheckSum()
{
      INSTALL_LIC("THPR-0003-2-CCCCCC-SOFO", ICLVClient.CLV_ERR_LICENSEKEY, "S-1-5-21-746137067-179605362
}

/*
 *   Test method for 'com.thinprint.lic.CLVClient.CLVInstallLicense(String)'
 *
 *   kurze seriennummer
 */
public void testCLVInstallLicenseWithShortSerialNo()
{
    INSTALL_LIC("THPR-0002-1-BBBB-G8U", 0, "S-1-5-21-746137067-179605362-839522115", "413ca03a");
    INSTALL_LIC("THPR-0003-1-BBBB-GD7X", ICLVClient.CLV_ERR_LICENSEKEY, "S-1-5-21-746137067-179605362-83
}

/*
 * Test method for 'com.thinprint.lic.CLVClient.CalcRegistrationKey(char[], char[])'
 */
public void testCalcRegistrationKey()
{
    CHECKREGKEY("THPR-0002-1-MM9PFF-9TSF", "68G671YW", "S-1-5-21-746137067-179605362-839522115", "413ca0
    CHECKREGKEY("THPR-0002-1-MM9PFF-9TSF", "68G671YW", "S-1-5-21-746137067-179605362-839522115", "413ca0
}
```

Wie fängt man an?

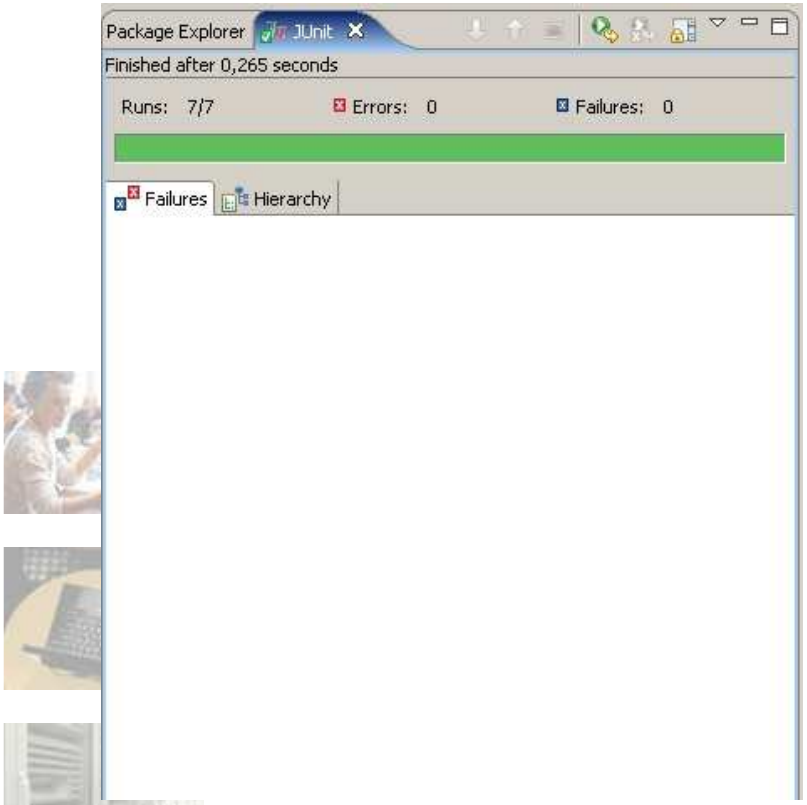
```
private void ISINVALID(String strLic, int iExpected, TPJCLVImpl oClient)
{
    assertEquals(strLic,
                 iExpected,
                 oClient.isLicenseKeyInvalid(strLic.toCharArray()));
}
```

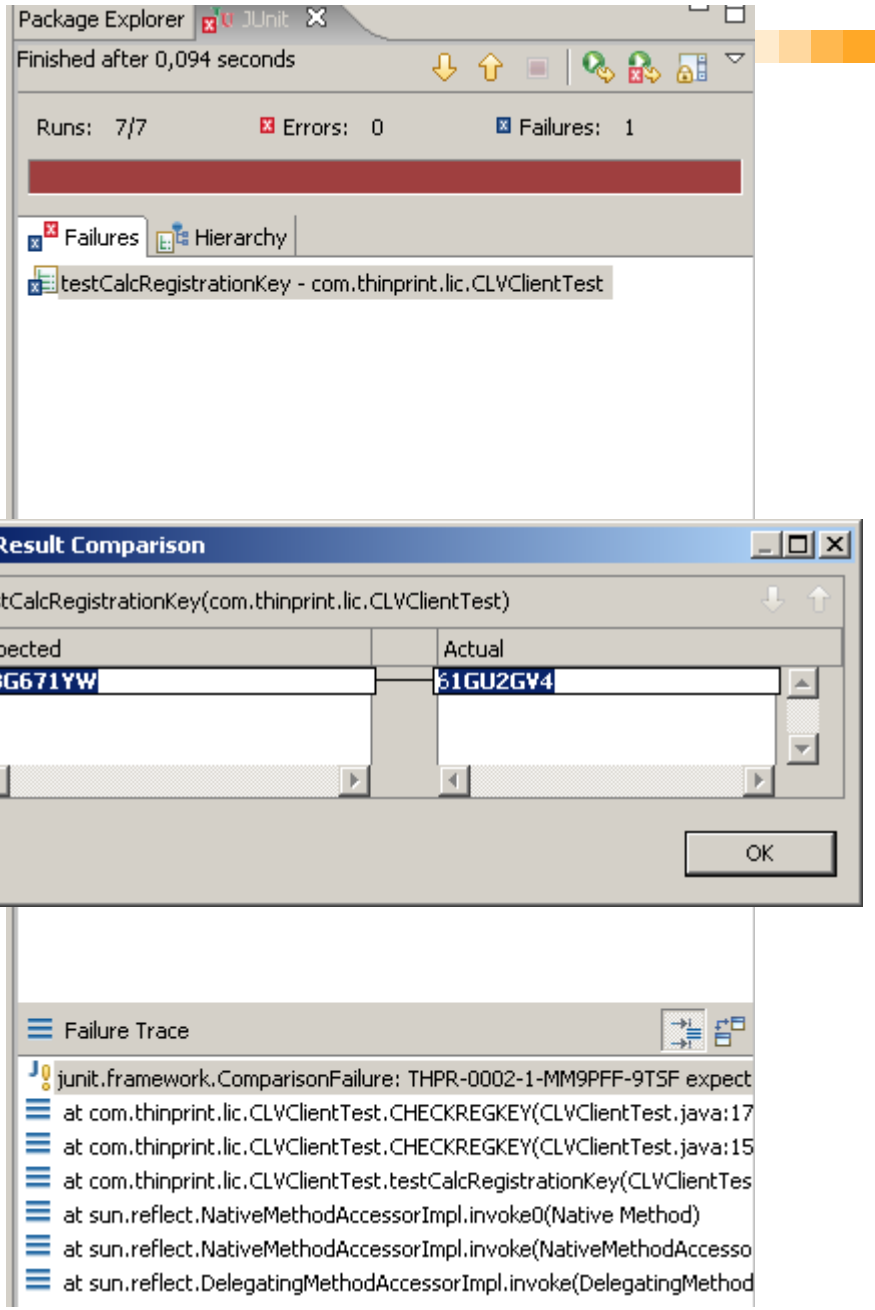
Wie fängt man an?



ThinPrint®

Ob ihr wirklich richtig steht…





ThinPrint®

Viel Spaß beim Testen