UE 2: Lexikalische Analyse (*Scanner*)

- Testen
  - siehe Testanleitung auf Homepage!
  - diese Woche in der UE:

    "**Softwaretesten mit JUnit – Eine Einführung**"
- UB-UE2-Angabe.zip: (neu seit 21.10.2004)
  - Compilerklassen (*Token.java*, *Messages.java*, Gerüst von *Scanner.java*, ...)
  - Testklassen (*AllTests.java*, *IdentifiedTestCase.java*, *TokenTest.java*, *ScannerTest.java*, *TestPrintWriter.java*, ...)
- Abgabe
  - siehe Abgabeanleitung auf Homepage!
  - bis **Mi, 20.10.2004, 12:00**: Aufgabe 2, elektronisch 
  - bis Do, 28.10.2004, 8:15: Aufgabe 1, auf Papier & elektronisch

Softwaretesten mit **JUnit**

Eine Einführung

für die

Übungen aus Übersetzerbau

Tests ausführen

- Compilieren des Compilers und der Testfälle:

```
> javac -classpath .;junit.jar
            ssw/mj/*.java ssw/mj/test/*.java
```

- Ausführen aller Testfälle (= AllTests.main)

```
> java -cp .;junit.jar ssw.mj.test.AllTests
```

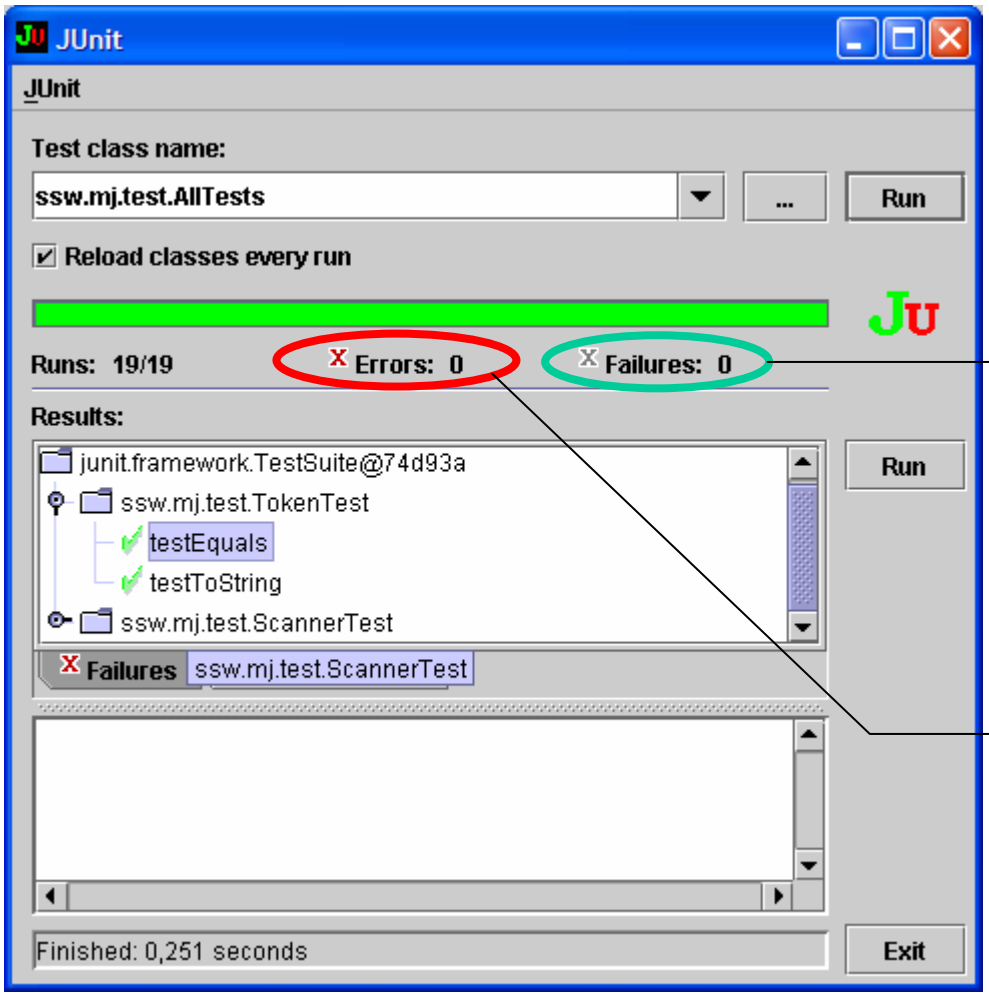
- Ausführen der Token Tests mit dem Text UI

```
> java -cp .;junit.jar junit.textui.TestRunner
            ssw.mj.test.TokenTest
```

- Starten der JUnit-Swing-Oberfläche

```
> javaw -cp .;junit.jar junit.swingui.TestRunner
```

JUnit Swing UI



Failure:
gibt Punkteabzüge

Error:
Aufgabe wird nicht korrigiert (= 0 Punkte)

Errors & Failures in JUnit

- Error
  - unvorhergesehene *Errors* oder *Exceptions* (= `java.lang.Throwable`)
- Failure
  - vom Tester vorausgesehener Fehlerfall
  - durch `assert`-Methode abgefragt
  - durch `junit.framework.AssertionFailedError` angezeigt

In TestCase.run-Methode:

```
try {
    runTest();
} catch (AssertionFailedError e) {
    result.addFailure(this, e);
} catch (Throwable e) {
    result.addError(this, e);
}
```

Testfälle hinzufügen

- siehe Demo
  - ScannerTest2:
    - Neuer Test in bestehender Testfallmethode (`testInvalidSymbols`, `"ö"`)
  - ScannerTest3:
    - Neuer Testfall (= Testmethode) in bestehender Testfallklasse (`testMyProg`)
  - MyTest:
    - neue Testfallklasse
    - zeigt Unterschied Error ⇔ Failure

Basistestklasse IdentifiedTestCase

- Gibt in *setUp()* vor jedem Testfall einen Header mit dem Testfallnamen auf der Console aus.

Hilfsklasse TestPrintWriter

- print- und println-Methode aus java.io.PrintWriter überschrieben ⇨ speichert Ausgabe zeilenweise in einer Liste
- erlaubt Hinzufügen von erwarteten Zeilen (addExpectedLine)
- automatisiert Vergleich der tatsächlichen mit der erwarteten Ausgabe (verify)

Hilfsfunktionen in Klasse ScannerTest

- *initScanner* setzt
  - Eingabe von String statt Datei
  - Ausgabe auf `TestPrintWriter` statt `System.out`

```
void initScanner (String s) {
  output = new TestPrintWriter();
  Scanner.init(new StringReader(s), output);
}
```

- *checkNext*
  - zum sequentiellen Überprüfen der gelieferten Tokens (überladen)

```
checkNext (int kind, int line, int col, int val, String str) {
    Token expected = new Token(kind, line, col, val, str);
    Token actual = Scanner.next();
    assertEquals(expected, actual);
}
```

Zahlen-Konstanten

- Gültige Zahlen
  - Normale Zahlen: `123;`
    - Ein Token: *number*
  - Negative Zahlen: `-123;`
    - Zwei Tokens: *minus* und *number*
  - Buchstaben: `123abc;`
    - Zwei Tokens: *number* und *ident*
- Ungültige Zahlen
  - Zu große Zahlen: `2147483648;` BIG_NUM
    - Spezialfall: `-2147483648;`
- Neuer JUnit-Testfall *testNumberIdent*

Zeichen-Konstanten

- Gültige Zeichen-Konstanten
  - Normale Zeichen: `'A';`
  - Escape-Zeichen: `'\r';` und `'\n';`
  - Backslash: `'\';`
- Fehlerhafte Zeichen-Konstanten
  - Kein Zeichen: `'';` EMPTY_CHARCONST
  - Fehlendes Ende: `'A;` MISS_SQUOTE
  - Escape-Zeichen: `'\A';` UNDEF_ESC
    - Backslash: `'\\';` UNDEF_ESC
  - Kombination: `'\A;` UNDEF_ESC + MISS_SQUOTE
    - Spezialfall: `'\;` UNDEF_ESC + MISS_SQUOTE

      Der Strichpunkt wird als Escape-Zeichen interpretiert und daher nicht als Token erkannt.
- Neuer JUnit-Testfall *testCharErrors*

Informationsquellen

- JUnit: http://junit.org
  - Doku: Kent Beck, Erich Gamma: *JUnit Cookbook*, *JUnit Test Infected: Programmers Love Writing Tests*, *JUnit: a Cook's Tour*, *JUnit FAQ*, *JUnit JavaDoc*
- WWW:
  - Frank Westphal: *Unit Testing mit JUnit*, http://www.frankwestphal.de/UnitTestingmitJUnit.html
  - Jeff Langr: *Evolution of Test and Code Via Test-First-Design.* Practitioner Report, OOPSLA 2001, http://www.objectmentor.com/resources/articles/tfd.pdf
- Bücher:
  - Johannes Link: *Unit Tests mit Java.* dpunkt.verlag, 2002
  - Kent Beck: *Test-Driven Development*, Addison-Wesley, 2003
- LVA:
  - Christoph Steindl: *Testen von Softwaresystemen*. KV am SSW

JUnit - Das Framework