



# Softwaretesten mit **J**Unit

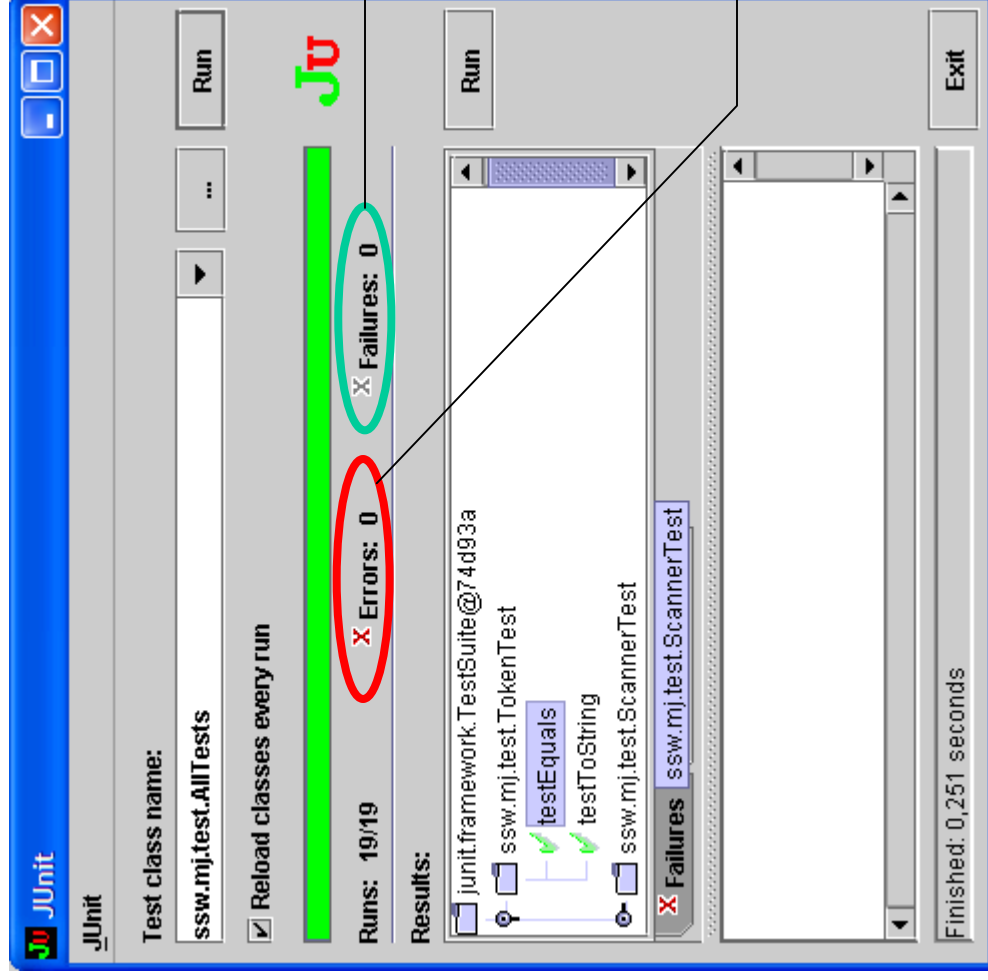
Eine Einführung  
für die  
Übungen aus Übersetzerbau

# 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:
    - in bestehender Testfallmethode
  - ScannerTest3:
    - in neuer Testfallmethode in bestehender Testfallklasse
  - MyTest:
    - in neuer Testfallklasse
    - zeigt Unterschied Error  $\Leftrightarrow$  Failure

# Hilfsfunktionen in Klasse Scanner



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

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

# 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`)

# Informationsquellen



- JUnit-Home: <http://junit.org>
- Johannes Link: *Unit Tests mit Java*. dpunkt.verlag, 2002
- Christoph Steindl: *Testen von Softwaresystemen*. KV am SSW
- JUnit-Dokumentation:
  - Kent Beck, Erich Gamma: *JUnit Cookbook*
  - *JUnit Test Infected: Programmers Love Writing Tests*
  - *JUnit: a Cook's Tour*
  - *JUnit FAQ*
  - *JUnit JavaDoc*
- 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>



# JUnit - Das Framework

