Testen mit JUnit

- Motivation, Beispiel
- Ausführen von JUnit-Tests
- Anwendung in GDI

Motivation: Testen

- Woher wissen wir, ob eine Funktion einen korrekten Wert für einen bestimmten aktuellen Parameter liefert?

→Wir testen sie, d.h. wir rufen sie auf und prüfen das Ergebnis

- Beispiel:

```
public class Maximum {

    static int max( int value1, int value2 ) {
        if (value1 > value2)
            return value1;
        else
            return value2;
    }

    public static void main( String[] args ) {

        println(max(readInt(), readInt()));

    }

}
```

Frage:
Mit welchem/n Wert(en) rufen wir die Methode auf?

Antwort:

Frage:
Was machen wir, wenn wir einen Fehler entdecken und die Methode verbessert haben?

Antwort:

Motivation: Automatisierung

- Frage:
  Was machen wir, wenn eine Aufgabe am Computer immer und immer wieder durchgeführt werden soll?

→ Wir schreiben ein Programm, das das für uns tut!

- Am Beispiel: So könnte ein Stück eines **Testprogramms** für die Methode "max" aussehen:

```
public static void main( String[] args ) {

    int result = max(3, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 3);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

}
```

Motivation: Automatisierung

- Das Testprogramm ist praktisch:
  Immer wenn man etwas an "max" geändert hat, sagt es einem "auf Knopfdruck“ (d.h. man muss es nur erneut starten), ob man dabei vielleicht einen Fehler eingebaut hat
- Anmerkung:
  Für „echte“ Funktionen sieht man das nicht so einfach...!

```
public static void main( String[] args ) {

    int result = max(3, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 3);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

}
```

Motivation: Automatisierung

- Das Testprogramm ist aber auch umständlich/geschwätzig
- Die Bibliothek JUnit vereinfacht das Schreiben von Testprogrammen

```
assertEquals(5, max(3, 5));
assertEquals(5, max(5, 3));
assertEquals(5, max(5, 5));
```

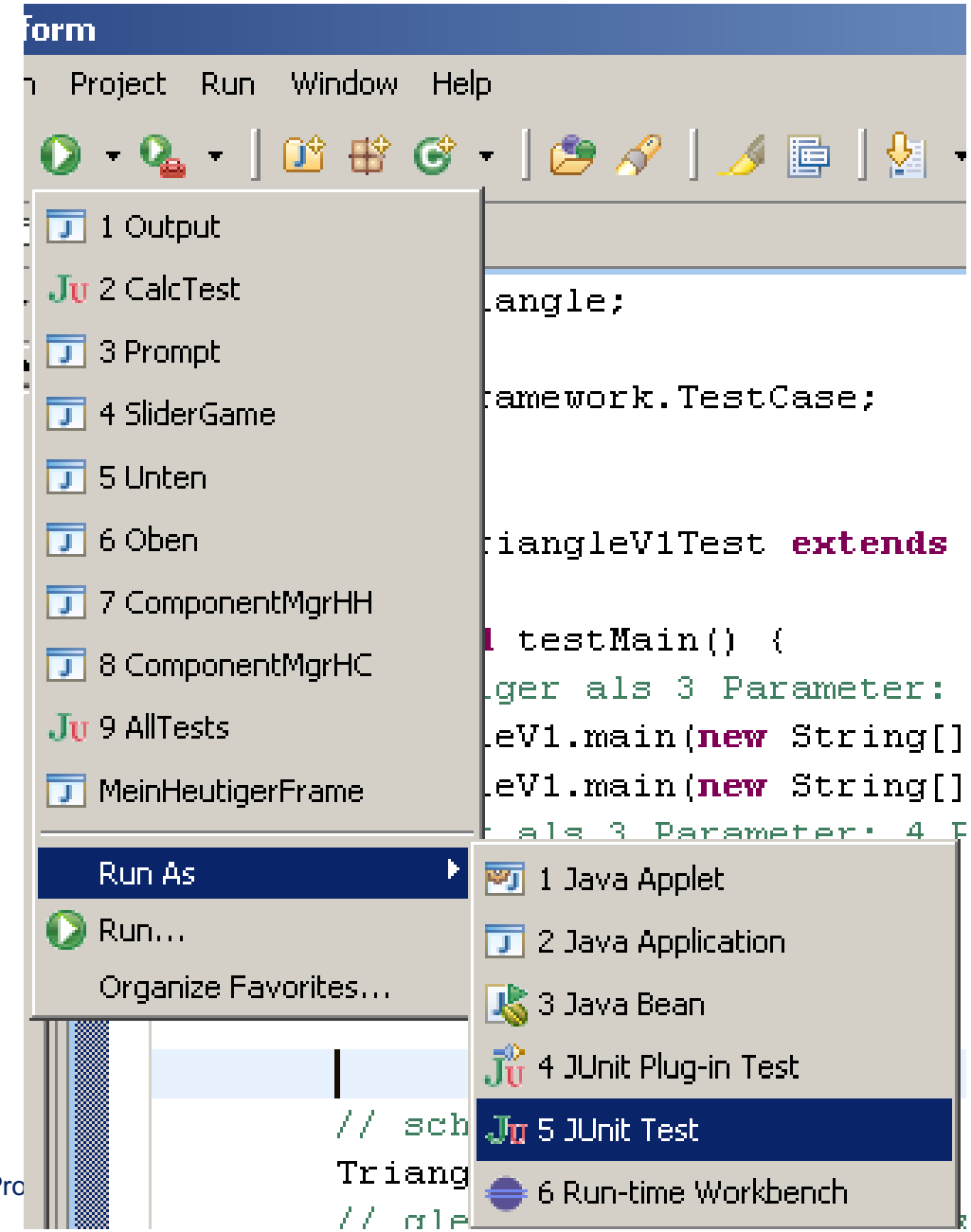
kürzerer Testcode mit JUnit

```
public static void main( String[] args ) {

    int result = max(3, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 3);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

    result = max(5, 5);
    if (result != 5)
        println("Falsches Ergebnis: 5 erwartet, " + result + " geliefert");

}
```

Testcode für den gründlichen(!) Test von "max" mittels JUnit

```
@Test
public final void testMax() {

    // positive Zahlen: kleiner, größer, gleich
    assertEquals(5, max(3, 5));
    assertEquals(5, max(5, 3));
    assertEquals(5, max(5, 5));

    // negative Zahlen
    assertEquals( 3, max( 3, -5));
    assertEquals( 3, max(-5,  3));
    assertEquals(-3, max(-3, -5));
    assertEquals(-5, max(-5, -5));

    // Null
    assertEquals(0, max( 0, -5));
    assertEquals(0, max(-5,  0));
    assertEquals(5, max( 0,  5));
    assertEquals(5, max( 5,  0));
    assertEquals(0, max( 0,  0));

}
```

Testfall zu einer Klasse ausführen

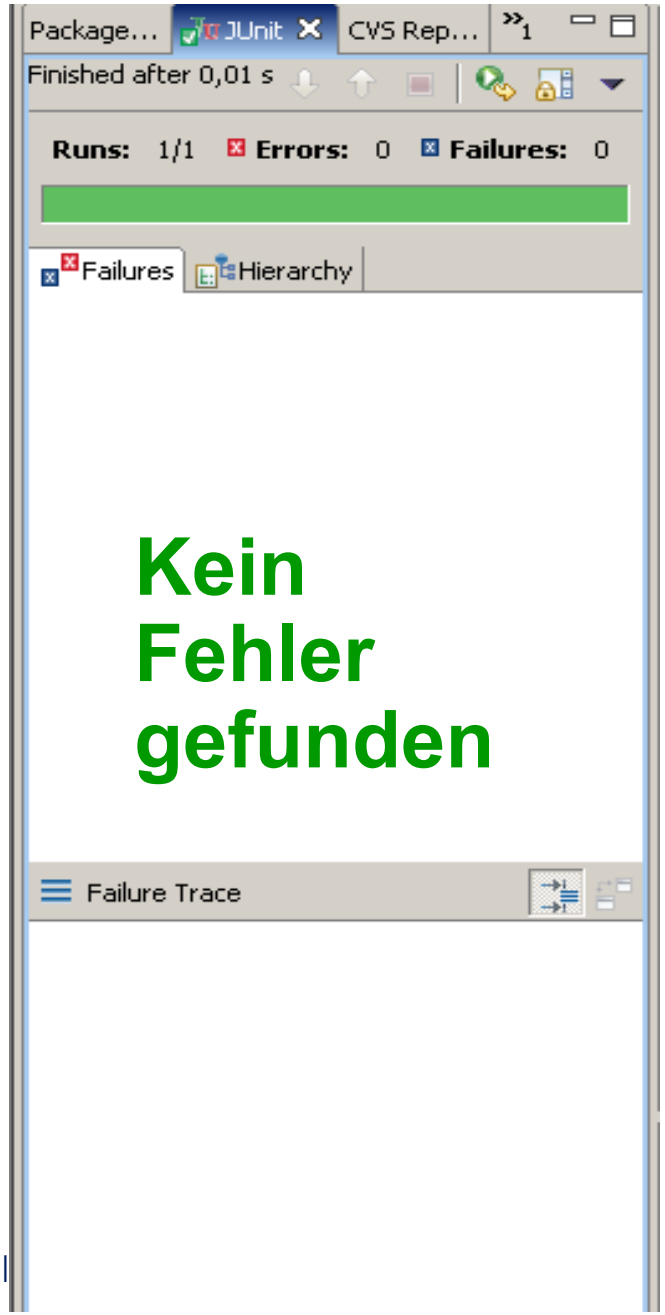


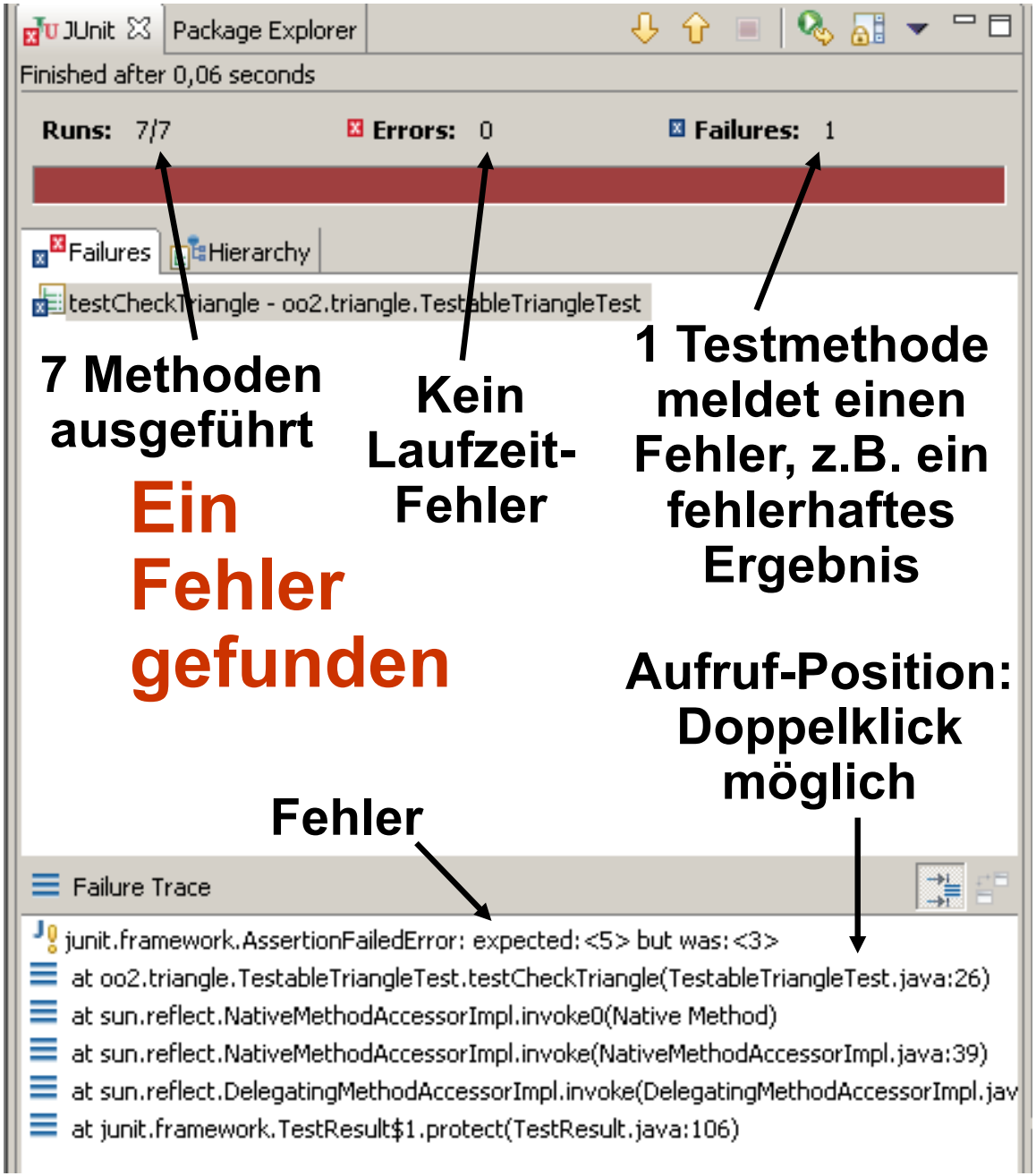
Alle Methoden einer Klasse, die mit

```
@Test
public void...() { ... }
```

annotiert sind,
werden nacheinander ausgeführt

Testfall zu einer Klasse ausführen: negatives / positives Ergebnis





Anwendung

- Ab Übungsblatt 4 werden wir JUnit benutzen, um Ihre Programme aus den Übungen zu testen
- Dazu gibt es jeweils ein "einfaches" Testprogramm für Sie zur Vorbereitung
- Benutzen Sie diese in den Laborstunden!
- Bei den Testaten oder vorher bei der Korrektur verwenden wir "richtige" Testprogramme