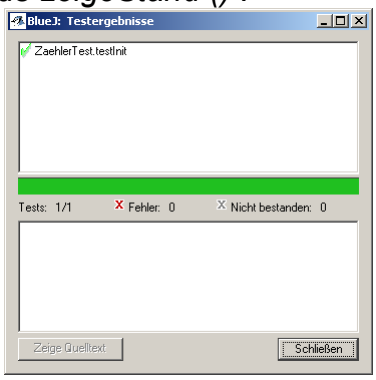
JUnit und test – driven developement

Unser BlueJ enthält ein sehr leistungsfähiges Werkzeug zum Testen von Projekten. Software ist – wenn sie nicht belanglos ist – nie fehlerfrei. Das hat in der Softwareentwicklung zu vielen Ansätzen geführt, die Qualität durch Tests zu verbessern. Ein Mittel dazu bei JAVA ist JUnit (Quelle: www.junit.org), das in BlueJ integriert ist. Damit kann man Testklassen schreiben, mit den Softwaretests automatisiert werden können.
Ein radikaler Ansatz bei der Softwareentwicklung ist die *Test – getriebene Softwareentwicklung*, bei der man mit der Definition von Tests beginnt und dann erst die Klassen schreibt.
Beginnen wir mit einem einfachen Zählerprojekt, indem wir zunächst eine Standardklasse Zaehler vom System erzeugen lassen und eine Testklasse dazu und nun zunächst in dieser Testklasse definieren, welche Anforderungen die Klasse Zaehler erfüllen soll. Z.B. verlangen wir, dass sich ein Objekt erzeugen lässt und dass es eine Methode *zeigeStand ()* gibt, die uns einen ordnungsgemäß mit 0 initialisierten Wert des Zählerstandes anzeigt:

```
/**
 * 1.  Test auf funktionierende Initialisierung mit 0.
 */                                   der Name muss mit test beginnen !
public void testInit()
{
    Zaehler zaehler = new Zaehler();
    assertNotNull(zaehler);
    assertTrue(zaehler.zeigeStand()==0);
}
```

Dieser Test muss zu diesem Zeitpunkt fehlschlagen – schlimmer, wir bekommen sie wegen der fehlenden Methode *zeigeStand ()* nicht einmal übersetzt. Wir wissen „nun“, dass wir die Klasse Zaehler entsprechend ergänzen müssen. Wir ersetzen im Standardklassentext die Methode beispielMethode(int y) durch die Methode *zeigeStand ()* :

```
/**
 * Zeigt den Stand des Zaehlers an.
 */
public int zeigeStand()
{
      return x;
}
```

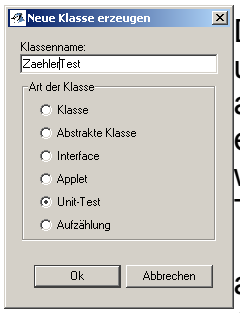


Tun wir das und testen nun, bekommen wir eine Erfolgsmeldung.
Natürlich soll die Klasse auch zählen können. Dazu gehen wir wie eben vor: Testmethode schreiben und Klassentext ergänzen.

```
/**
 * 2.  Test auf funktionierendes Zaehlen.
 */
public void testZaehlen()
{
    Zaehler zaehler = new Zaehler();
    zaehler.zaehle();
    assertTrue(zaehler.zeigeStand()==1);
    assertFalse(zaehler.zeigeStand()==0);
}
```

Der Test führt wieder zu einem Erfolgsfall. Wir ergänzen – wieder erst den Test – und dann eine Methode zum Zurücksetzen des Zählers. Macht man diese falsch, z.B. dadurch, dass sie nichts tut, bekommt man beim Testen eine Fehlermeldung.

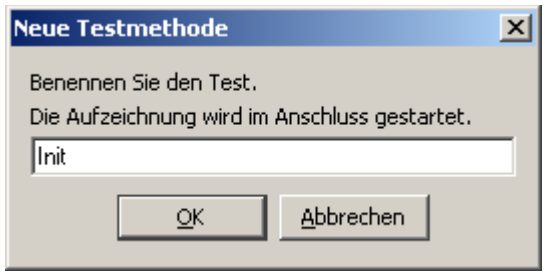
Automatisiertes Erstellen von Testklassen



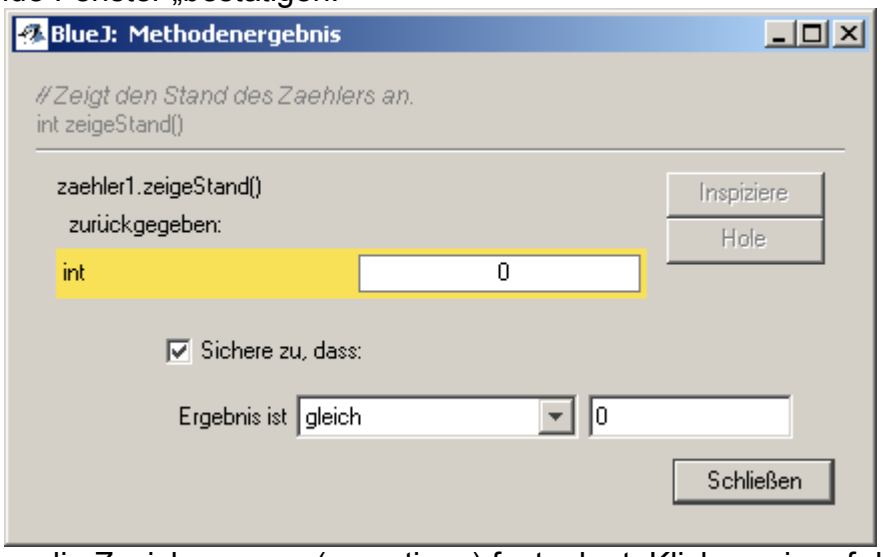
Der Vorgang lässt sich mit BlueJ auch automatisieren. Starten wir mit unserer fast „fertigen“ Zählerklasse allein ein neues Projekt und fügen eine Testklasse ein, dann können wir im Kontextmenü dieser Testklasse den Punkt

ErzeugeTestmethode

auswählen und erhalten das rechts dargestellte Fenster.



Nun können wir den Test „aufzeichnen“: Objekt erzeugen, Methode zeigeStand() aufrufen und das folgende Fenster „bestätigen.



Hier werden nun die Zusicherungen (assertions) festgelegt. Klicken wir auf den Button Beenden, dann ist unsere Testmethode fertig und kann wie oben beschrieben verwendet werden. Der automatisch erstellte Text ist geringfügig anders, inhaltlich aber vergleichbar:

```
public void testInit()
{
      Zaehler zaehler1 = new Zaehler();
      assertEquals(0, zaehler1.zeigeStand());
}
```

Wieder gibt es natürlich – wie wir erwartet haben – einen Fehler.
(Siehe Fenster rechts)