

JAVA

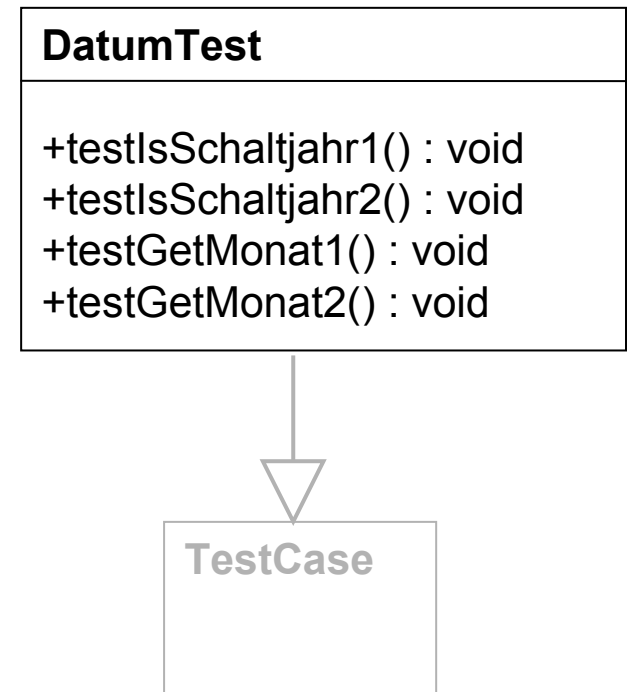
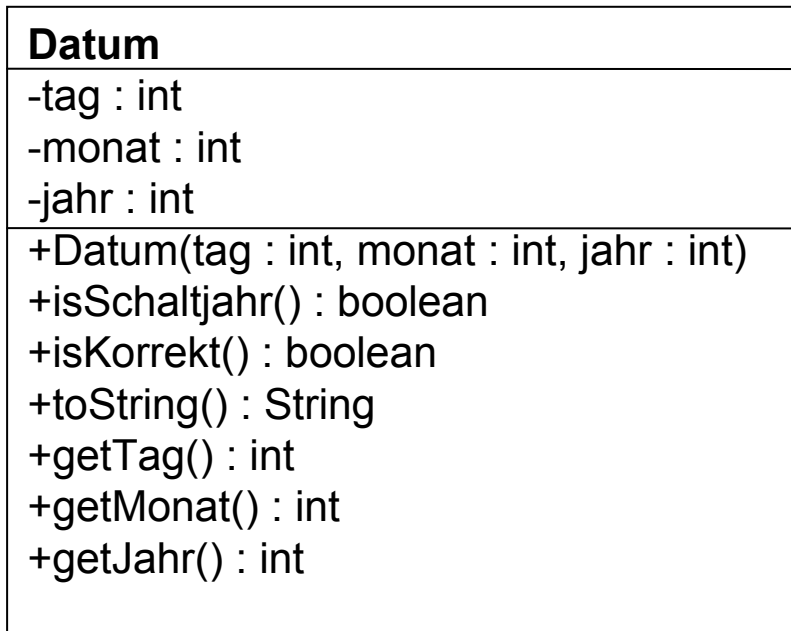
Automatisierte Tests mit JUnit

Testen allgemein

- Testen einer Software
 - ◆ Überprüfung, ob Programm gestellten Anforderungen erfüllt
 - ◆ Testen hat immer das Ziel Fehler zu finden
 - ◆ Keine Fehler gefunden bedeutet Misserfolg für den Tester!
 - ◆ Tests müssen wiederholbar sein
- Verschiedene Arten von Tests
 - ◆ Abnahmetest: Kunde überprüft, ob *gesamte* Software allen seinen Ansprüchen genügt. Nach erfolgreicher Abnahme ist Projekt beendet.
 - ◆ Einzeltest: Entwickler überprüft, ob ein einzelner Softwarebaustein – etwas ein oder mehrere Klassen – die entworfenen Funktionen korrekt ausübt.
 - ◆ Regressionstests: Wiederholt ausführbare Tests (manuell oder automatisch), um zuvor durch Testen als korrekt befundene Funktionalität nach Softwareänderungen erneut zu überprüfen
 - ◆ Lasttests, Benutzbarkeitstests, Sicherheitstests, ...
- Blackbox Testing
 - ◆ Tester kennt Implementierung nicht und entwirft Tests rein auf Basis der Anforderungen
- Whitebox Testing
 - ◆ Tester kennt Implementierung und entwirft Tests auch auf Basis der Implementierung, z.B. Testfälle entwerfen, so dass jede Schleife mindestens einmal durchlaufen wird

- Einfaches aber hilfreiches Testframework zur Implementierung und Ausführung automatisierter Einzeltests durch Entwickler
- Initial für Java entwickelt, Variante für .NET (NUnit), C++ und andere Programmiersprachen existieren
- Details: www.junit.org
- JUnit
 - ◆ Für Blackbox- oder Whitebox Testing
 - ◆ Tests werden als Java Programm implementiert
 - ◆ Pro Klasse eine Testklasse implementieren
 - ◆ Pro öffentlicher Methode und Konstruktor mindestens eine Testmethode implementieren
- JUnit sehr weit verbreitet und in fast allen Java IDEs integriert

- Testklasse erbt von TestCase
- Testmethoden immer mit „test“ beginnen, kein Rückgabewert
- Konvention: Methodenname mit Name der zu überprüfenden Methode fortführen



```
public class DatumTest extends TestCase {
```

```
    public void testIsshaltjahr1() {  
        Datum datum = new Datum(1, 1, 2000);
```

Testdaten erzeugen

```
        assertTrue(datum.isshaltjahr());
```

Zur überprüfende Methode aufrufen und Ergebnis überprüfen

```
    }
```

```
}
```

- Bei Ausführung Testklasse (wie das geht: gleich):
- Für jede Methode, die mit „test“ beginnt
 - ◆ Neues Objekt DatumTest wird erzeugt
 - ◆ Methode wird ausgeführt
 - ◆ Falls Parameterwert von assertTrue „false“ ist, dann schlägt der Test fehl
 - ◆ Ansonsten ist der Test erfolgreich
- Falls ein Test fehlschlägt, muss Ursache beseitigt werden:
 - ◆ Entweder zu testende Methode in Datum hat einen Fehler oder
 - ◆ Testimplementierung in DatumTest hat (noch) einen Fehler

```
public class DatumJUnitTest extends TestCase {
```

```
    public void testIstSchaltjahr1() {  
        Datum datum = new Datum(1, 1, 2000);  
        assertTrue(datum.istSchaltjahr());  
    }
```

datum.istSchaltjahr() ist true, genau dann, wenn 2000 ein Schaltjahr ist

```
    public void testIstSchaltjahr2() {  
        Datum datum = new Datum(1, 1, 2001);  
        assertTrue(! datum.istSchaltjahr());  
    }
```

! datum.istSchaltjahr() ist true, genau dann, wenn 2001 kein Schaltjahr ist

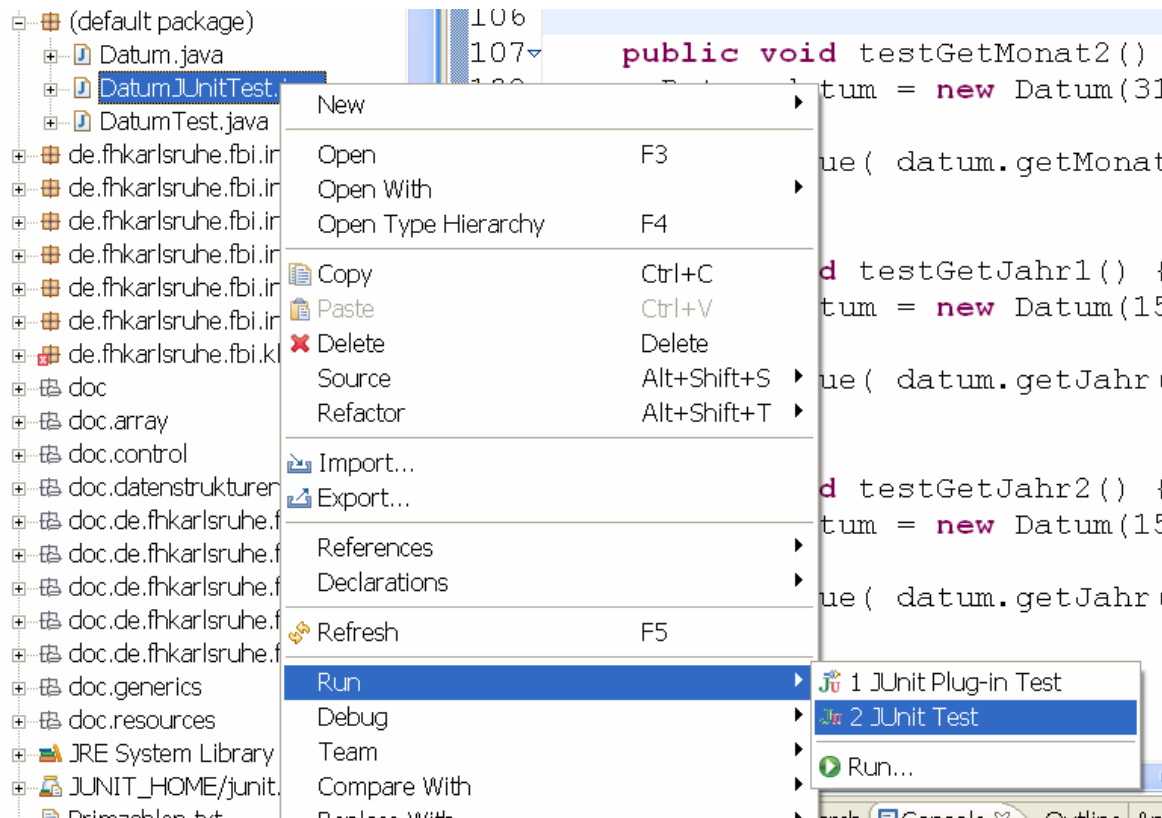
```
    public void testGetMonat1() {  
        Datum datum = new Datum(15, 12, 1996);  
        assertTrue(datum.getMonat() == 12);  
    }
```

```
    public void testGetMonat2() {  
        Datum datum = new Datum(31, 4, 2001);  
        assertTrue(datum.getMonat() == -1);  
    }
```

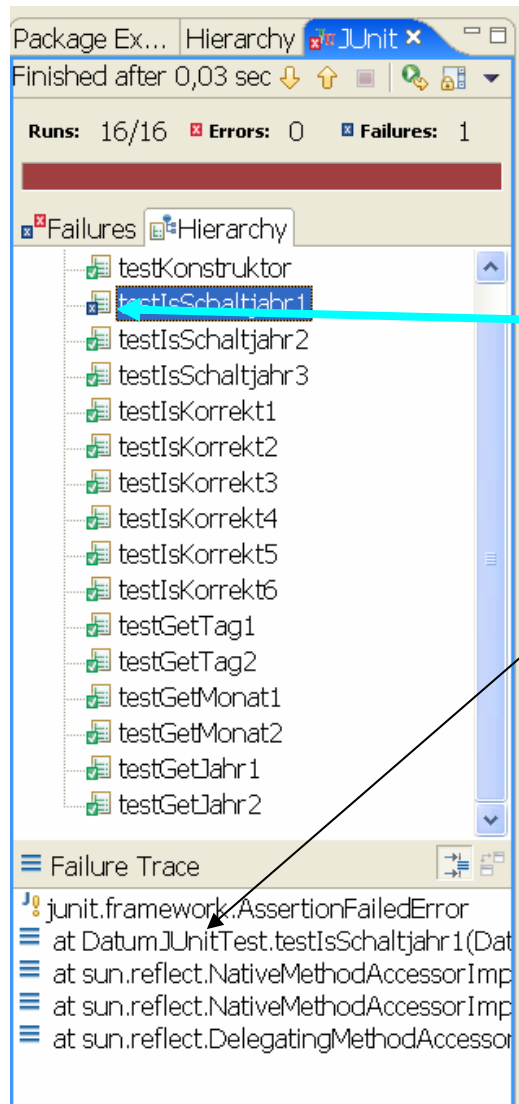
```
}
```

JUnit und Eclipse

- Starten einer JUnit Testklasse in Eclipse
 - ◆ JUnit Testklasse Rechtsklicken, Run -> JUnit Test anwählen
 - ◆ Falls alle Testfälle in Ordnung sind, passiert nichts weiter



JUnit und Eclipse



- Falls ein Test fehlschlägt:
 - ◆ Sicht wechselt zu „JUnit“
 - ◆ Balken ist rot (statt grün)
 - ◆ Fehlgeschlagene Tests sind mit blauem x markiert
 - ◆ Test ist bei einem `assertTrue()` in der Methode fehlgeschlagen
 - ◆ Direkt zur Stelle springen und Testfall überprüfen und korrigieren
- Jeder Testfall in dieser Sicht kann ausgewählt und individuell ausgeführt werden (Rechtsklick -> Run oder Debug)

JUnit und Eclipse

- Für jede Klasse eine Testklasse implementieren
- Ausführen aller Testklassen in Eclipse
 - ◆ Projekt in anwählen, Rechtsklick und Run JUnit
 - ◆ Package anwählen, Rechtsklick und Run JUnit führt jede JUnit Testklasse im ausgewählten Package aus.
- JUnit Testklasse erstmalig erzeugen:
 - ◆ Zur testende Klassen anwählen, Rechtsklick, New -> JUnit Test Case auswählen, Dialog bestätigen, weiteren Dialog folgen (z.B. Auswählen der zu überprüfenden Methoden)
 - ◆ Oder einfache normale Klasse erzeugen mit „extend TestCase“ nach Klassenname.

