UNIKASSEL VERSITÄT

Research Group Software Engineering

Programmiermethodik

Übung 2

Sommersemester 2010

Fachgebiet Software Engineering

Andreas Scharf

Agenda

- **Organisatorisches zur Abgabe**
- **Besprechung HA 1**
- **JUnit**
- **Praktische Übung:**
  - Eclipse herunterladen & einrichten
  - Projekt erstellen & exportieren
  - Vorarbeit für Hausaufgabe: Bank Beispiel (inkl. JUnit Tests)
- **Vorstellung HA 2**

Organisatorisches zur Abgabe

- **Alles was kein Code ist: Abgabe als PDF**
- **Name und Matrikelnummer (jedes Gruppenmitglieds) auf jedes (am besten in der Kopfzeile) Blatt.**
- **Abgabe von Code: Exportierte Eclipse Projekte**

Besprechung HA 1 I

- **Aufgabe 1 – Abstrakt vs. Konkret**

Abstrakt	Konkret
Baum	Birke
Fahrzeug	Silberner Audi R8
Tier	Schweizer Bergziege
…	….

…aber auch

Backwaren aus der Cafeteria der Universität Kassel Standort WA	Ein Mohnbrötchen belegt mit einer Gurke, zwei saftig grünen Blättern Salat und zwei Scheiben schmackhaftem Schinken auf einem Bett von cremiger Butter zum Preis von 1,10 Euro welches ich gerade verspeise

Besprechung HA 1 II

- **Aufgabe 1 - Abstrakt vs. Konkret: Was gibt‘s zu verbessern?**

Abstrakt	Konkret
Suchmaschine	Beim Eingeben der Wörter „Uni Kassel“ erscheint als 1. die Homepage der Uni-Kassel.
Viele Politiker, die viele Gesetze aufstellen und sehr viel reden	Angela Merkel – Deutsche Bundeskanzlerin

z.B.:

Suchmaschine	Google
Berühmte Person	Angela Merkel – Deutsche Bundeskanzlerin

Besprechung HA 1 III

- **Aufgabe 1 - Abstrakt vs. Konkret**
  - Es gibt verschiedene Abstraktionslevel, z.B.:

Abstrakt	Konkret
Pflanze	Baum
Baum	Birke

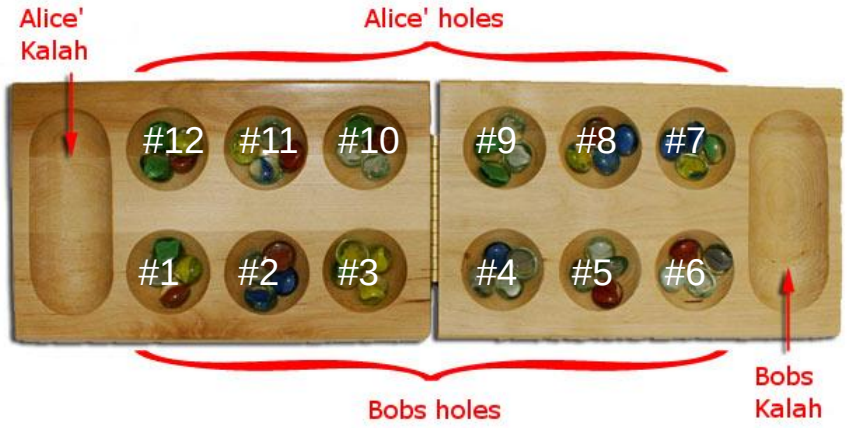
  - **Abstrakt**: Von (lat. *abstractus* – „abgezogen“). Bezeichnet meist das Weglassen von Einzelheiten und das Überführen auf etwas Allgemeineres oder Einfacheres.
  - **Konkret:** Von (lat. *concretus* „dicht, fest“). Bezeichnet etwas, das *wirklich, greifbar, bestimmt, gegenständlich* ist.

Besprechung HA 1 IV

- **Aufgabe 2 – Textuelle Szenarien**
- **Bestehen aus:**
  - Titel
  - Startsituation
  - Ablauf
  - Endsituation
- **Szenarien sollten voneinander verschiedene Situationen beschreiben**
- **Sollten nicht voneinander abhängig sein**

Besprechung HA 1 V

- **Am Beispiel Mancala:**



Szenario: Normal turn

Alice and Bob are playing Mancala. Alice owns the upper 6 holes with 4 Stones in each hole and Bob owns the lower 6 holes with 4 Stones in each hole as well. Bob and Alice don‘t have any stones in their Kalah. It‘s Bobs turn: He grabs all 4 stones from hole number 1 and distributes them one by one to the holes 2, 3, 4 and 5. Now there is no stone in hole number 1, 5 stones in the holes 2, 3, 4 and 5 and 4 Stones in hole number 6. Bobs move is finished and it‘s Alice turn now.

Besprechung HA 1 V

- **Analyse**

  Szenario: Normal turn

  Alice and Bob are playing Mancala. Alice owns the upper 6 holes with 4 Stones in each hole and Bob owns the lower 6 holes with 4 Stones in each hole as well. Bob and Alice don‘t have any stones in their Kalah. It‘s Bobs turn: He grabs all 4 stones from hole number 1 and distributes them one by one to the holes 2, 3, 4 and 5. Now there is not stone in hole number 1, 5 stones in the holes 2, 3, 4 and 5 and 4 Stones in hole number 6. Bobs move is finished and it‘s Alice turn now.

  - Titel
  - Startsituation
  - Ablauf
  - Endsituation

JUnit – Motivation

- **Testen ist Ausprobieren?**
- **Unit Tests prüfen systematisch, ob das Programm das tut was es soll**
- **Vorteile:**
  - Reproduzierbar
  - Automatisierbar und selbst auswertend
  - Dokumentierend und Anwendungsbeispiel

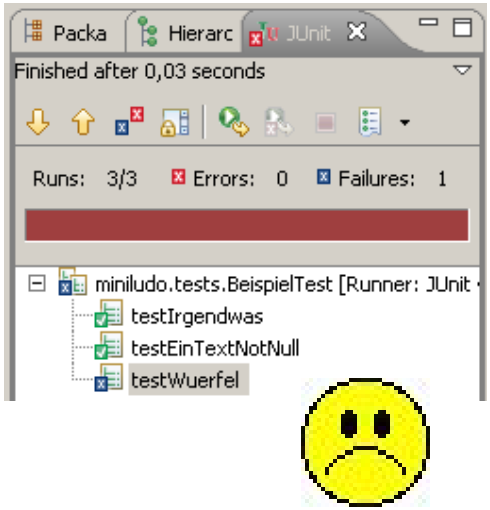
- **=> Weniger Fehler, Debugging und Fehlersuche**

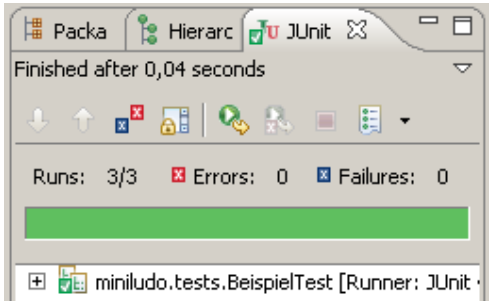
JUnit - Vorgehen beim Testen

- **Test vor der Implementierung:**
  - Funktionalität abprüfen
  - Alle möglichen Fehlerfälle prüfen
  - aber keine Trivialitäten
- **So wenig wie möglich implementieren**
  - Genau soviel, dass der Test erfolgreich ist
- **Umfang vom Testcode:**
  - 15-50% Anteil am Gesamtcode

JUnit - Framework (1)

- **Freies Open Source Framework**
  - http://www.junit.org/
  - aktuell: 4.8.2
- **Autoren: Kent Beck & Erich Gamma**
- **Grundeinstellung:**
  - Das Feature funktioniert erst, wenn ein Test geschrieben wurde!





JUnit - Framework (2)

- **Tests in separaten Test-Klassen**
- **Funktionsweise der Test-Methoden**
  - Beispiel-Situation aufbauen
    - Eventuell Ausgangssituation merken
  - Eine Aktion durchführen
    - Ein oder mehrere Methodenaufrufe
  - Rückgabewert oder Veränderungen prüfen
    - Mit Hilfe von Assertion-Methoden
    - Prüfung von Fehlerbehandlung (Exceptions)

JUnit - Assertion-Methoden

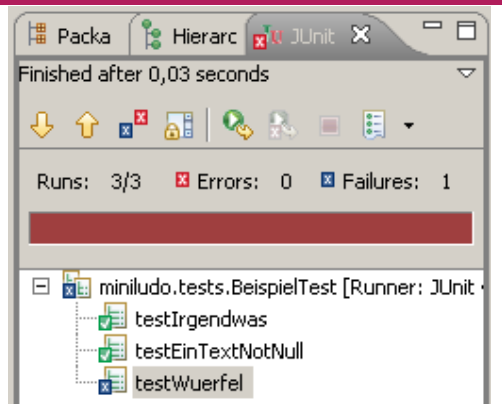
- **Ausdrücke, die wahr sein müssen, sonst Abbruch der aktuellen Test-Methode**
- ***assertEquals(soll, ist)***
  - Object, primitive Typen wie boolean, int, long, ...
- ***assertTrue(boolean)***
- ***assert[Not]Null(Object), assert[Not]Same(obj1, obj2)***
- ***fail()* (in Zusammenhang mit Kontrollfluss)**
- **Bei Fehlschlag oder *fail()*:**
  - Wirft *junit.framework.AssertionFailedError*
- **Alle Methoden auch mit ‚*String message*' Parameter**
  - Beispiel: `assertEquals(„Würfel zeigt 6", 6, wuerfel.getWert())`

JUnit - Version 4

- Keine separaten Testklassen nötig (im Vergleich zu JUnit 3)
- Tests werden per Annotation markiert

```
import static org.junit.Assert.*;

public class Addierer {

    public int add(int a, int b) { return a + b; }

    private int summand;

    @Before
    public void init() throws Exception { summand = 1; }

    @Test
    public void einsPlusEinsGleichZwei() {
        int ergebnis = add(summand, summand);
        assertEquals(2, ergebnis);
    }
}
```

Testen - Vorgehen bei Rot



- **Funktionalität fertig implementiert?**
- **Ansonsten:**
  - Fehlermeldung!
  - Stacktrace
- **Debugging - Einkreisen eines Fehlers**
  - Breakpoints an „spannenden“ Stellen setzen
  - Variablenbelegung / Objektstruktur überprüfen
  - Methode schrittweise ausführen
  - in interessante Methoden reinsteppen
- **Tipp:**
  - Fehler gefunden => reproduzierenden Test schreiben

Praktische Übung I

- **Eclipse herunterladen & einrichten**
  - Download unter http://eclipse.org/downloads/
  - „Eclipse IDE for Java Developers“
  - In ein beliebiges Verzeichnis entpacken
  - Starten
- **Projekt erstellen und Exportieren -> DEMO**

Praktische Übung II

- **Entwickeln: Test-first!**
  - Am Beispiel eines Bankkontos (Vorarbeit für Hausaufgabe)
- **„Gerüst“ schreiben: Klassen, Methodensignaturen**
- **JUnit Tests schreiben**
  - Einzahlen
  - Abheben
  - Überweisen
  - …
  - Randfälle beachten!! Z.B.: Abheben/Überweisen wenn Konto nicht gedeckt
- **Methoden implementieren -> JUnit Balken grün werden lassen**
- **DEMO**

Vorstellung HA 2

- **Download unter http://seblog.cs.uni-kassel.de/wp-content/uploads/2010/04/HA-21.pdf**
- **Abgabe bis Mittwoch, 28.04.2010 um 23.59 Uhr**

Los geht‘s

Ihr seid dran!