Praktikum „SEP: Java-Programmierung“ SS 2019

Unit Testing mit JUnit

Thomas Bunk und Karlheinz Friedberger

Überblick

- Warum Testen?
- Was sind Unit Tests?
- Der Teufelskreis des Nicht-Testens
- JUnit
- JUnit in Eclipse

Warum Testen?

- Wie überprüft man, dass ein System das Richtige tut?
  - System in richtigen Zustand bringen
  - Mit Beispiel-Eingaben ausführen
  - Überprüfen ob das erwartete Verhalten auftritt

Warum Testen?

- Wie überprüft man, dass ein System das Richtige tut?
  - System in richtigen Zustand bringen
  - Mit Beispiel-Eingaben ausführen
  - Überprüfen ob das erwartete Verhalten auftritt

- Wie oft muss man überprüfen?
  - Direkt beim Einführen einer neuen Funktionalität
  - Jede Änderung am System kann anderes gewünschtes Verhalten beeinflussen
  - Nach jeder Änderung muss das System überprüft werden!

Warum Testen?

- Wie überprüft man, dass ein System das Richtige tut?
  - System in richtigen Zustand bringen
  - Mit Beispiel-Eingaben ausführen
  - Überprüfen ob das erwartete Verhalten auftritt
- Wie oft muss man überprüfen?
  - Direkt beim Einführen einer neuen Funktionalität
  - Jede Änderung am System kann anderes gewünschtes Verhalten beeinflussen
  - Nach jeder Änderung muss das System überprüft werden!
- ... kann man nicht ein Programm dafür schreiben?
  - Man kann! Und zwar **automatisierte** Tests

Was sind Unit Tests? (1/2)

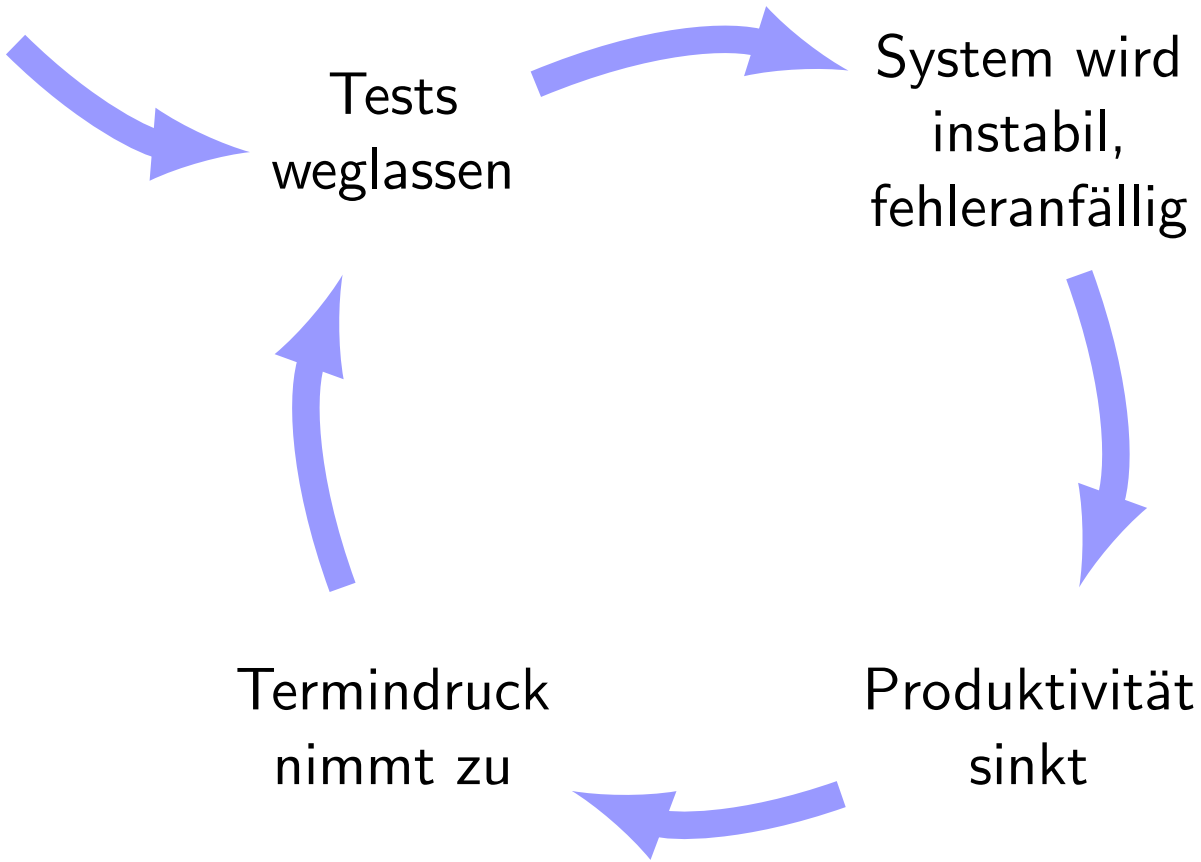
- Test-Programme sind **automatisierte Tests**
  - Genauer: die einzelnen Testfälle werden Test genannt

- Tests können unterschiedlich fein sein
  - **Unit Tests** testen die kleinste testbare Einheit einer Software (z.B. einzelne Methode in Java), typischerweise isoliert vom Rest des Systems.
  - **Integrationstests** testen die Interaktion zwischen Komponenten (z.B. public Interfaces).
  - **Systemtests** testen die Software als Ganzes.

Was sind Unit Tests? (2/2)

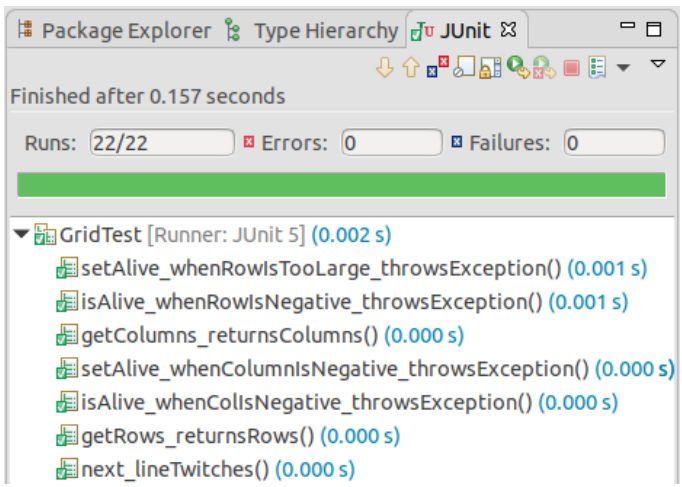
- Jede Code-Änderung kann bestehende Funtionalität kaputt machen
- Angenommen, es gäbe für jedes gewünschte Verhalten einen Test, dann könnte man...
  - per Knopfdruck überprüfen ob eine Änderung eine bestehende Funktionalität kaputt gemacht hat
  - aus dem Test herauslesen, welches Verhalten das System nicht mehr zeigt (evtl. mit dem Debugger)
  - den Code reparieren, so dass der Test wieder erfolgreich durchläuft

Der Teufelskreis des Nicht-Testens



JUnit

- JUnit ist ein Unit Testing Framework für Java
- JUnit stellt verschiedene `assert`-Methoden bereit, die dabei helfen, Tests durchzuführen
  - Die Idee: Die Assertions werden den Test-Methoden hinzugefügt, um das Verhalten der zu testenden Methode zu überprüfen. Ist das Verhalten anders als erwartet, so schlägt der Test fehl.
- Ein Test kann zwei Ergebnisse produzieren:
  - Bestanden (Grün) oder Fehlgeschlagen (Rot)



JUnit in Eclipse

- JUnit 5 einem Projekt in Eclipse hinzufügen:
  - Project ⇒ Properties ⇒ Java Build Path ⇒ Libraries ⇒ Add Library... ⇒ JUnit 5 ⇒ Finish