Karlsruher Institut für Technologie

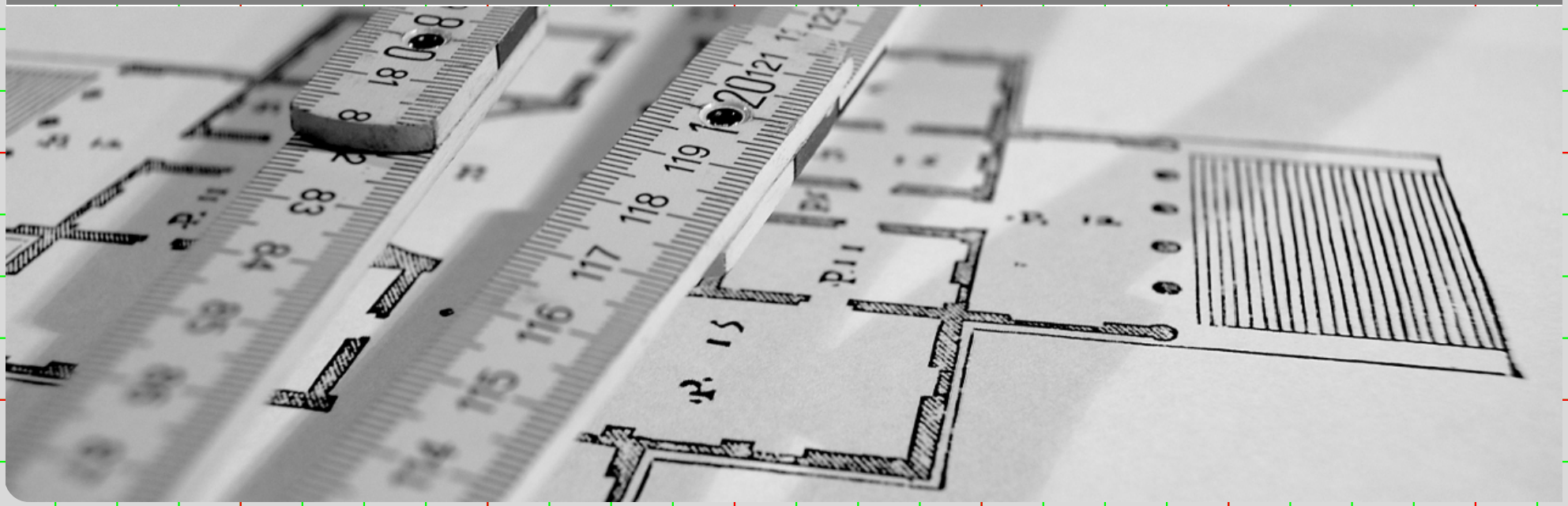
mm 40 60 80 100 120

Softwaretechnik 1 - 0. Tutorium

Felix Bachmann | 25. April 2017

40

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)

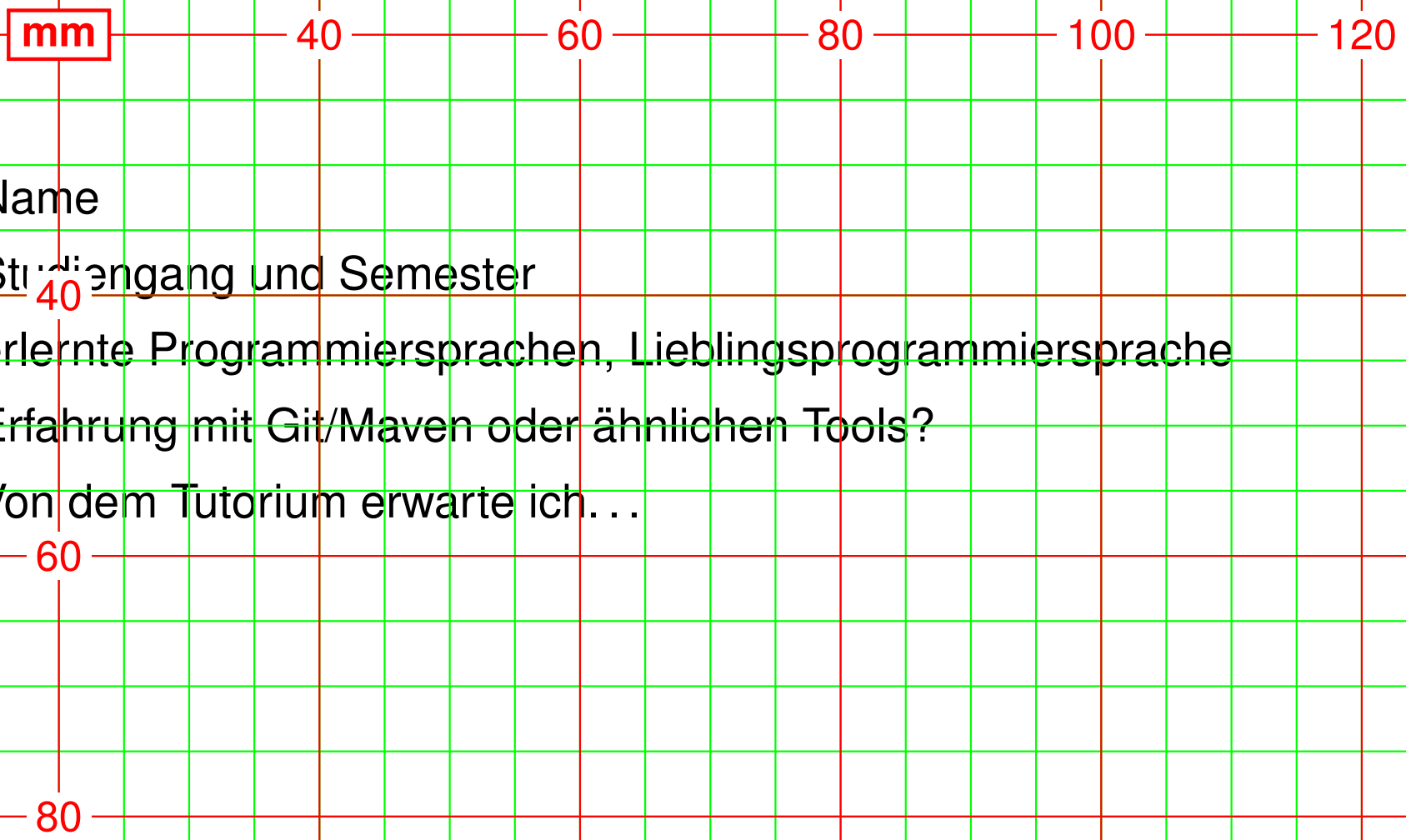


Themenübersicht

Das bin ich

- Felix Bachmann
- Informatikstudent im 4. Semester
- erstes Tutorium
- E-Post-Adresse: felix.bachmann@ewetel.net

. . . und ihr?



- Name
- Studiengang und Semester
- erlernte Programmiersprachen, Lieblingsprogrammiersprache
- Erfahrung mit Git/Maven oder ähnlichen Tools?
- Von dem Tutorium erwarte ich. . .

Verhalten im Tutorium

cool

- **mitdenken**
- **Fragen stellen**
- **Fragen beantworten**
- essen & trinken
- gehen
- schlafen

!cool

- laut sein
- stören
- andere ablenken

mm 40 60 80 100 120

Verhalten im Tutorium

cool

- **mitdenken**
- **Fragen stellen**
- **Fragen beantworten**
- essen & trinken
- gehen
- schlafen

!cool

- laut sein
- stören
- andere ablenken

Übungsbetrieb

- Bestehen des Scheins Voraussetzung zum Bestehen des Moduls
- neue Übungsblätter ungefähr alle 2 Wochen $\Longrightarrow$ 1+6 Blätter
- Übungen i.d.R. jeweils am Tag der Abgabe
- ab 50% der Punkte habt ihr sicher bestanden
- Besprechung der Musterlösung
- Abgaben
  - Theorieaufgaben+Deckblatt im 3.Stock
  - Programmieraufgaben auf http://lez.ipd.kit.edu

Tutoriumsbetrieb

- Wann?: ab dem 15.05 14-tägig
- Wo?: Raum -107
- Was?:
  - Wiederholung des VL-Stoffs
  - “Rechnen“ von Aufgaben (Altklausuren)
  - ggf. Tipps für die Übungsblätter
- Folien gibt’s im Ilias und auf `www.github.com/malluce/swt1-tut`
- Fragen stellen !!

Fragen zu Übung(sblättern), Vorlesung

erst im Forum, auf Google oder Stackoverflow nachschauen, dann

- neuen Forum-Thread anlegen
- falls nicht öffentlich postbar: Mail an mich oder swt1@ipd.kit.edu (nur im Notfall)

Was ihr bisher getan haben solltet..

Installation von:

- Eclipse (incl. CheckStyle und EclEmma)
- Git

Überblick über:

- Maven
- Git

Probleme mit der Installation? $\Longrightarrow$ kommt nach dem Tut nach vorne

JUnit4 - Überblick

- Test-Tool für Java-Klassen
- Nur öffentliche Methoden testen
- Konventionen:
  - Für Klasse Hallo Testklasse HalloTest schreiben
  - Methode hallo(Object o) wird z.B. durch die Methode testHalloWithNull() getestet

JUnit4 - Aufbau einer Testklasse

Methoden können mit Annotationen (@XYZ) versehen werden

Aufbau:

- @BeforeClass (wird als erstes einmal ausgeführt)
- @Before (wird vor jeder Test-Methode einmal ausgeführt)
- @Test (vergleichen erwartetes und reales Ergebnis, schlagen ggf. fehl, Ausführung in beliebiger Reihenfolge)
- @After (wird nach jeder Test-Methode einmal ausgeführt)
- @AfterClass (wird am ende einmal ausgeführt)

JUnit4 - Aufbau einer Testklasse

Methoden können mit Annotationen (@XYZ) versehen werden

Aufbau:

- @BeforeClass (wird als erstes einmal ausgeführt)
- @Before (wird vor jeder Test-Methode einmal ausgeführt)
- @Test (vergleichen erwartetes und reales Ergebnis, schlagen ggf. fehl, Ausführung in beliebiger Reihenfolge)
- @After (wird nach jeder Test-Methode einmal ausgeführt)
- @AfterClass (wird am ende einmal ausgeführt)

JUnit4 - Aufbau einer Testklasse

Methoden können mit Annotationen (@XYZ) versehen werden

Aufbau:

- @BeforeClass (wird als erstes einmal ausgeführt)
- @Before (wird vor jeder Test-Methode einmal ausgeführt)
- @Test (vergleichen erwartetes und reales Ergebnis, schlagen ggf. fehl, Ausführung in beliebiger Reihenfolge)
- @After (wird nach jeder Test-Methode einmal ausgeführt)
- @AfterClass (wird am ende einmal ausgeführt)

JUnit4 - Aufbau einer Testklasse

Methoden können mit Annotationen (@XYZ) versehen werden

Aufbau:

- @BeforeClass (wird als erstes einmal ausgeführt)
- @Before (wird vor jeder Test-Methode einmal ausgeführt)
- @Test (vergleichen erwartetes und reales Ergebnis, schlagen ggf. fehl, Ausführung in beliebiger Reihenfolge)
- @After (wird nach jeder Test-Methode einmal ausgeführt)
- @AfterClass (wird am ende einmal ausgeführt)

JUnit4 - Aufbau einer Testklasse

Methoden können mit Annotationen (@XYZ) versehen werden

Aufbau:

- @BeforeClass (wird als erstes einmal ausgeführt)
- @Before (wird vor jeder Test-Methode einmal ausgeführt)
- @Test (vergleichen erwartetes und reales Ergebnis, schlagen ggf. fehl, Ausführung in beliebiger Reihenfolge)
- @After (wird nach jeder Test-Methode einmal ausgeführt)
- @AfterClass (wird am ende einmal ausgeführt)

JUnit4 - Assert

org.junit.Assert bietet diverse Methoden, um Ergebnis mit Erwartung abzugleichen

zu jeder Methode kann als erstes Argument ein String mitgegeben werden (wird bei Fehlschlag angezeigt)

Beispiele:

- assertArrayEquals(int[] expected, int[] actual)
- assertNotNull(Object obj)
- assertSame(Object expected, Object actual)

JUnit4 - eine Testmethode

Zu testende Methode in der Klasse Hallo

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

Testmethode in der Klasse HalloTest

```
@Test
public void testAdd() {
Assert.assertEquals(7, Hallo.add(5 + 2));
}
```

(mehr Beispiele später)

Maven - Überblick

- Maven ist Jiddisch und heißt „Sammler des Wissens“
- Build-Management-Tool (Automatisierung von möglichst vielen Schritten)
- Maven ist in jeder Eclipse-Installation integriert
  $\Longrightarrow$ keine manuelle Installation nötig
- Aufgaben von Maven
  - Strukturierung (durch vorgegebene Verzeichnisstruktur)
  - Kompilieren
  - Testen
  - Verwalten von Abhängigkeiten
  - Verpacken

Maven - Überblick

Verzeichnisstruktur:

- src
  - main
    - java
    - resources
  - test
    - java
    - resources
- target
  - classes
  - test-classes
  - *.jar / *.war / *.zip . . .
  - . . .
- pom.xml

Maven -pom.xml

- pom steht für “Project Object Model“
- konfiguriert euer Maven Projekt im XML-Format (gefüllt durch default-Werte)
  - Wo sucht Maven Tests?
  - Wohin speichert Maven Build-Dateien?
  - In welches Format soll das Projekt verpackt werden?
  - …
- Eclipse-Plugin bietet GUI

Maven - Überblick

Wichtige Befehle

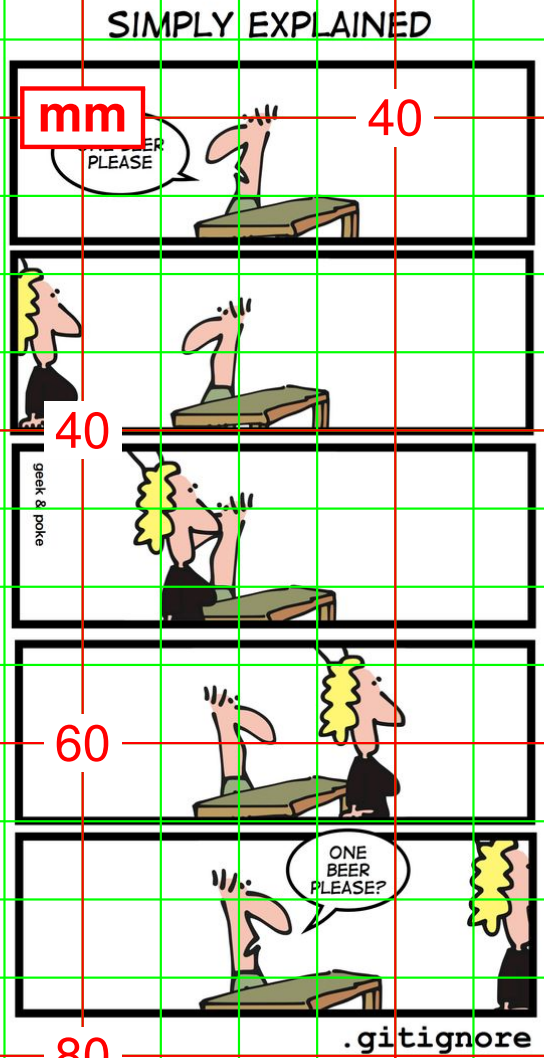
`mvn compile`	kompiliert Quelltexte zu .class-Dateien
`mvn test`	kompiliert Test-Quelldateien zu .class-Dateien, führt Tests aus und zeigt Ergebnisse an
`mvn package`	verpackt euer Projekt in eine Datei (.war/.jar/.zip)
`mvn clear`	leert euren target-Ordner

Maven - Fehler finden

Lösungsansätze:

- Rechtsklick auf Projekt $\Longrightarrow$ Maven $\Longrightarrow$ Update Maven Project $\Longrightarrow$ Haken bei “Force Update...”
  - Synchronisiert pom.xml mit Projekt, aktualisiert Abhängigkeiten
- mvn clean
  - vielleicht war der target-Ordner verschmutzt
- C:/Users/MeinName/.m2/ löschen und mvn compile (oder mvn package) ausführen
  - löscht alle Dependencies und lädt sie neu runter (ab und zu lädt man leider korrupte Dateien runter oder Dateien fehlen)

Bis dann! (dann=15.05.17)



geek-and-poke.com/geekandpoke/2012/11/7/simply-explained.html