


SO GEHTS WEITER

Informationen zur Klausur und zu den nächsten Kursen.

Struktur und Inhalt des Kurses wurden 2012 von Markus Heckner entwickelt. Im Anschluss haben Alexander Bazo und Christian Wolff Änderungen am Material vorgenommen. Die aktuellen Folien wurden von Alexander Bazo erstellt und können unter der **MIT-Lizenz**  verwendet werden.

AKTUELLER SEMESTERFORTSCHRITT (WOCHE 15)

Kursabschnitt	Themen		
Grundlagen	Einführung	Einfache Programme erstellen	Variablen, Klassen & Objekte
	Kontrollstrukturen & Methoden	Arrays & komplexe Schleife	
Klassenmodellierung	Grundlagen der Klassenmodellierung	Vererbung & Sichtbarkeit	
Interaktive Anwendungen	Event-basierten Programmierung	String- & Textverarbeitung	
Datenstrukturen	Listen, Maps & die Collections	Speicherverwaltung	Umgang mit Dateien
Software Engineering	Debugging	Planhaftes Vorgehen bei der Softwareentwicklung	
	Qualitätsaspekte von Quellcode		

DAS PROGRAMM FÜR HEUTE

- Ein kurzer Rückblick auf das vergangene Semester: Was sollten Sie gelernt haben?
- Informationen zum Inhalt und zur Organisation der Klausur
- Ausblick auf die kommenden Semester (aus der Programmier-Perspektive)

DAS HABEN SIE GELERNT!?!

Hinweis: Die folgende Liste an Fragen deckt ab, welche Erkenntnisse Sie auf jeden Fall aus diesem Kurs behalten und mit (in die Klausur) nehmen sollten. Wenn Sie eine oder mehrere der Fragen nicht beantworten können, arbeiten Sie die entsprechenden Inhalte auf jeden Fall nach. **In der Klausur werden solche allgemeinen Fragen nicht gestellt.** Das entsprechende Wissen wird aber zur Beantwortung der konkreten Klausurfragen benötigt.

GRUNDLAGEN

- Was ist eine Variable? Wie wird eine Variable in Java verwendet?
- Was ist der *Scope* einer Variable?
- Was sind Datentypen?
- Was ist ein Ausdruck? Was ist eine Anweisung?
- Was ist eine Methode?
- Was ist ein Rückgabewert?
- Was ist eine Klasse? Was ist ein Objekt? Was ist eine Instanz?
- Was ist ein Interface?
- Was ist ein Wahrheitswert?
- Was sind Kontrollstrukturen?

KLASSEN UND OBJEKTE

- Was versteht man unter objektorientierter Programmierung?
- Wie werden Klassen angelegt?
- Wie werden Instanzen von Klassen erstellt?
- Was ist ein Konstruktor?
- Was sind Sichtbarkeitsbereiche? Wie wirken sich diese aus?
- Wie funktioniert Vererbung in Java?
- Wie unterscheiden sich Instanz- und Klassenvariablen?
- Was sind Konstanten?
- Wie wird die this-Referenz verwendet?
- Wie werden Methoden eines Objekts aufgerufen?

KONTROLLSTRUKTUREN

- Wie unterscheiden sich for- und while-Schleifen?
- Wie werden die break- und continue-Befehle genutzt?
- Wie verhalten sich verschachtelte Kontrollstrukturen?
- Wie werden *Endlosschleifen* mithilfe eines *Sentinel* verwendet bzw. abgebrochen?

DATENSTRUKTUREN

- Wie wird ein *Array* erzeugt?
- Wie iterieren Sie über jede Stelle *eines Arrays*?
- Wie unterscheiden sich *Array* und *Arraylist*?
- Wie werden *HashMaps* verwendet?
- Was ist ein *Iterator*?
- Wie wird der *Datentyp* einer *ArrayList* angegeben?

EVENT-BASIERTE PROGRAMMIERUNG

- Was verstehe man unter eventbasierter Programmierung?
- Welche Arten von *Events* werden in der *GraphicsApp* verwendet?
- Welche Rolle haben Interfaces in der Verarbeitung von *Events*?
- Was ist ein *Observer*? Was ist ein *Observable*?

WEITERE THEMEN

- Welche Informationen werden auf dem *Heap*, welche auf dem *Stack* gespeichert?
- Was versteht man unter *Decomposition*?
- Wie funktioniert der *Top Down*-Ansatz?
- Was ist Codequalität? An was kann diese gemessen werden?
- Warum sind gut gewählte Bezeichner wichtig?
- Welche Techniken können Sie zum Debuggen einsetzen?
- Was ist ein Bug? Was ist ein Syntaxfehler? Was sind *Exceptions*?
- Wie werden Zeichenketten in Java verarbeitet?
- Was ist ein *Immutable*?

INFORMATIONEN ZUR KLAUSUR

DAS WICHTIGSTE

- Die Klausur findet am 18. Februar statt.
- Treffen Sie pünktlich um 12:30 Uhr **vor** den Räumen **H2** und **H3** ein. Wir informieren Sie im Vorfeld, in welchem Raum Sie schreiben werden.
- Die Klausur beginnt, sobald die Anwesenheit kontrolliert wurde und alle wichtigen Informationen verlesen wurden. Sie haben **60 Minuten** für die Beantwortung der Fragen.
- Die Klausur wird *closed book* durchgeführt. Hilfsmittel sind nicht erlaubt. Einzige Ausnahme: In Absprache können Sie ein Wörterbuch verwenden.
- Besteht der Verdacht auf Unterschleif, wird dies im Prüfungsprotokoll vermerkt und führt in der Regel zum **nicht bestehen* der Klausur.

ZULASSUNGSVORAUSSETZUNGEN

Sie müssen zwei von drei Studienleistungen/Aufgaben bestanden haben, um die Modulprüfung ablegen zu können. Studieren Sie nach **neuer Studienordnung (ab WS 2017/18)** wird dies direkt im Flexnow-System kontrolliert. Studieren Sie nach alter Studienordnung, kontrollieren wir die Zulassungsvoraussetzungen im Vorfeld.

Hinweis: Achtung: Eine Teilnahme an der Klausur ist ohne Anmeldung im Flexnow-System nicht möglich. Sollten Sie sich nicht anmelden können, z.B. als Gaststudierender, erinnern Sie mich noch einmal per E-Mail daran.

**MELDEN SIE SICH JETZT IM FLEXNOW-
SYSTEM AN!**

KLAUSURINHALTE


In der Klausur sollen Sie zeigen, dass Sie die Inhalte verstanden haben und eigenständig zur Problemlösung einsetzen können. Zu diesen Inhalten gehören:

- Alle in der Vorlesung erläuterten Themen (inkl. *Bouncer*)
- Alle Übungsaufgaben
- Die Lesetexte (sofern es die in der Vorlesung besprochenen Konzepte vertieft – neue, d.h. nicht in der Vorlesung besprochene Inhalte aus dem Buch sind NICHT für die Klausur relevant)

HINWEISE FÜRS *AUSWENDIGLERNEN*

- *Exotische* API-Methoden werden angegeben, z.B. `public void setBorderWeight(double newWeight)`
- *Standardmethoden* und Syntax werden vorausgesetzt (z.B. Objekterzeugung, Erzeugung und Befüllen von Arrays, Schleifen, Verzweigungen)

TIPPS FÜR DIE KLAUSUR

- Alles in Ruhe lesen, dann erst beginnen
- Schritt für Schritt vorgehen – lieber eine Aufgabe konsequent bearbeiten, anstatt mehrere Aufgaben parallel
- Üben Sie das *Paper Programming* anhand älterer Klausuren, die Sie **online**  finden

SOFTWAREENTWICKLUNG IM WEITEREN VERLAUF IHRES (MEDIENINFORMATIK-) STUDIUMS

SOFTWAREENTWICKLUNG IN DEN NÄCHSTEN KURSEN: HIER WIRD (NUR) PROGRAMMIERT

- OOP
- Anwendungsprogrammierung mit Android (nächstes Semester)
- Algorithmen und Datenstrukturen (nächstes Semester)
- *Multimedia Engineering* (nach OOP, ADP und Android)

Hinweis: Nahezu alle Kurse der Medieninformatik haben direkt oder indirekt mit der Entwicklung von Software zu tun. Der Anteil und Zweck der Programmierarbeit ist dabei jedoch unterschiedlich.

SOFTWAREENTWICKLUNG IN DEN NÄCHSTEN KURSEN: HIER WIRD SOFTWARE AUS NUTZERPERSPEKTIVE BETRACHTET

- Grundlagen HCI
- *Usability Engineering*

HIER DIENT DIE SOFTWAREENTWICKLUNG ALS GRUNDLAGE FÜR FORTGESCHRITTENE THEMEN

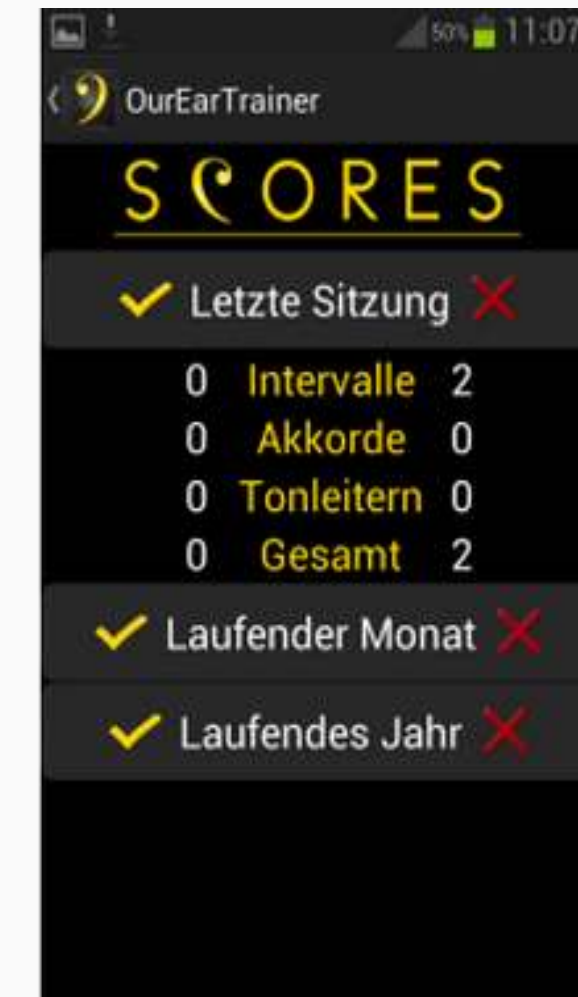
- Multimedia Technology
- Datenbanken
- *Sketching with Hardware*
- Abschlussarbeiten (!)

EINFÜHRUNG IN DIE ANWENDUNGSENTWICKLUNG



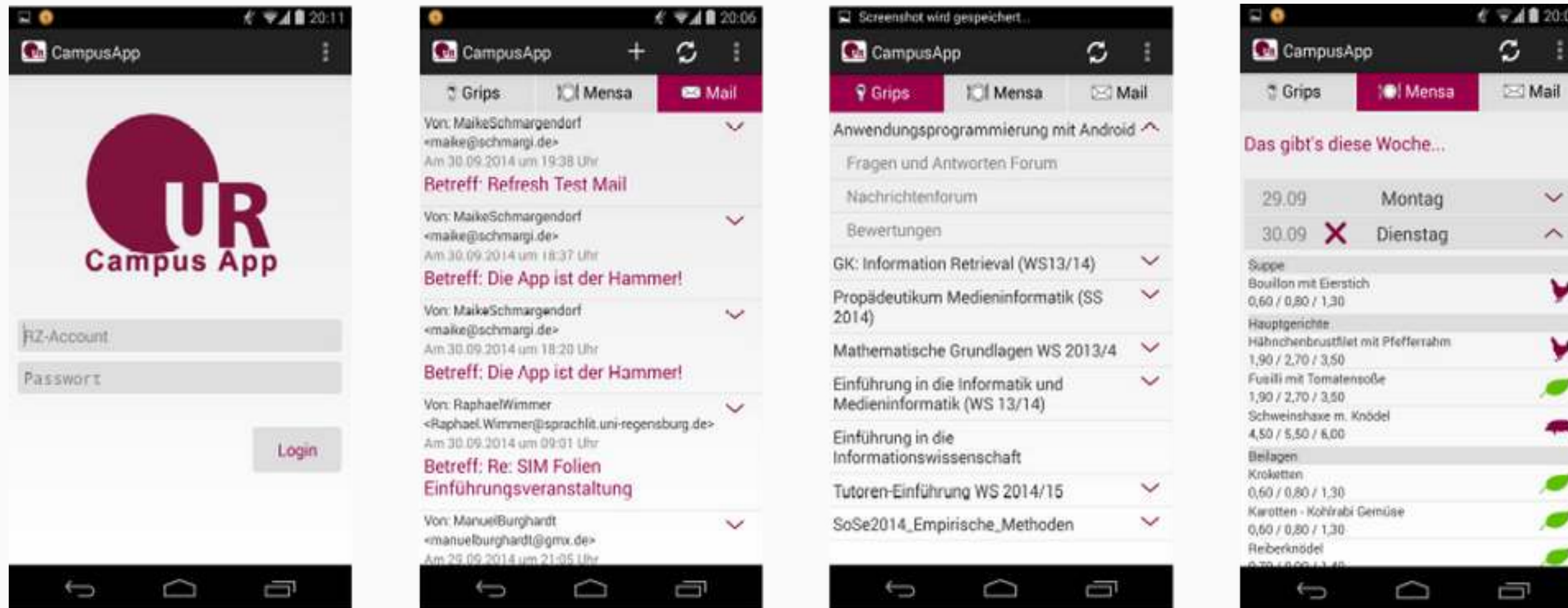
Auf Basis des Android-Frameworks beschäftigen wir uns mit grundlegenden Themen der Anwendungsentwicklung, z.B. mit der Entwicklung graphische Benutzeroberflächen, der praktische Anwendung von Datenbanken, der asynchrone Verarbeitung in *Threads*, Netzwerkkommunikation, *Location Based Services* und Karten und dem Umgang mit Sensordaten. **Android ist dabei ein Beispiel für die Umsetzung allgemeingültiger Methoden und Praktiken des Software Engineerings.**

PROJEKTE AUS DEM ANDROID-KURS: EARTRAINER



Hinweis: [Brem & Schäbel, 2014] entwickelten im Android-Kurs eine Anwendung für das individuelle Gehörtraining. NutzerInnen werden akustische Signale (Tonintervalle) vorgespielt, die anschließend über ein Piano korrekt wiedergegeben werden müssen.

PROJEKTE AUS DEM ANDROID-KURS: CAMPUS-APP



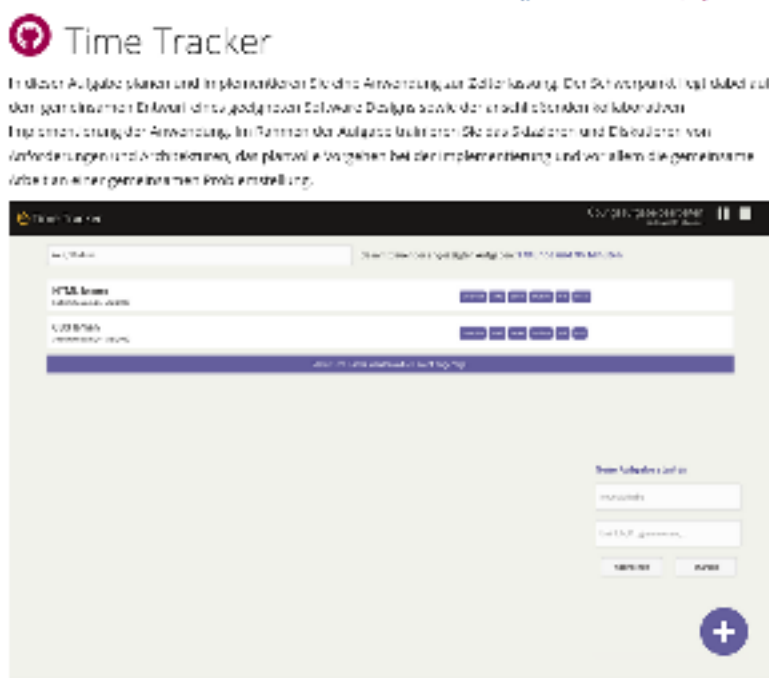
Hinweis: [Deller, Schmargendorf, Schuller & Weber, 2014] ermöglichten den gebündelten Zugriff auf viele Uni-Dienste (GRIPS, E-Mail und den Speiseplan der Mensa über eine zentrale Android-Anwendung.

PROJEKTE AUS DEM ANDROID-KURS: CAMPUS-APP



Hinweis: [Ackermann, Kugler, Geblich & Schlott 2014] erstellten ein GPS-basiertes Spiel für die Android-Plattform, das das *Pokemon-Prinzip* für den Uni-Campus umsetzt.

MULTIMEDIA ENGINEERING



Time Tracker

In dieser Aufgabe planen und implementieren Sie eine Anwendung zur Zeiterfassung. Der Schwerpunkt liegt dabei auf dem gemeinsamen Entwurf eines geeigneten Software Designs sowie den erforderten Implementierungen. Die Implementierung der Anwendung soll in Form von Klassen und Methoden von Anforderungen und Anforderungen, die planmäßig vorgehen bei der Implementierung und schließlich die gemeinsame Arbeit an einer gemeinsamen Implementierung.

Aufgabenbeschreibung

Grundlage für die Arbeit an der Anwendung ist ein gemeinsames GUT (Modellierung). In dem die verschiedenen Teammitglieder die Aufgaben des Projekts definieren. Die Entwicklung soll in zwei Phasen. Zu Beginn entwickeln die Teams - ausgehend von den definierten Anforderungen - ein Design für die Software. Dazu gehören die Identifikation der notwendigen Komponenten, deren Aufgaben und die Kommunikation zwischen den verschiedenen Komponenten. In der zweiten Phase (Implementierung) wird das Design in Code umgesetzt.

Auf Basis moderner Webtechnologien beschäftigen wir uns mit fortgeschrittenen Themen der Anwendungsentwicklung im speziellen und des Software Engineerings im speziellen. Dazu gehören das *Software Design* und die Softwarearchitektur, der Einsatz von Entwurfsmustern, und fortgeschrittene Methoden und Praktiken der Versionsverwaltung.

Mehr unter www.regensburger-forscher.de/mme □

PROJEKTE AUS DEM MME-KURS: ZOMBIE MAZE

Hinweis: Ein 2D-Adventure mit Taschenlampen-Effekt von [Seipel & Wrobel, 2015].



PROJEKTE AUS DEM MME-KURS: CINEPHILIA

Hinweis: Ein Mehrspieler-Quiz mit Filmfragen und -Rätseln von [Bachl, Schifferl & Will, 2015].



PROJEKTE AUS DEM MME-KURS: BODY-SPORT

Hinweis: Eine Webanwendung für Fitnessstudio mit personalisiertem Trainingsplan und -Tagebuch von [Bräuer, Brem & Gürbüz, 2015].



VON OOP ZU DEN NÄCHSTEN KURSEN (1/6)

Programmieren im Speziellen und Softwareentwicklung im Allgemeinen sind elementarer Bestandteil Ihres Studiums: Ab dem nächsten Semester werden Sie selbstständig kleine Anwendungsprojekte umsetzen müssen.

VON OOP ZU DEN NÄCHSTEN KURSEN (2/6)

Versuchen Sie die grundlegenden Techniken und Qualitätsaspekte, die Sie hier gelernt haben auf die kommenden Kurse zu übertragen.

VON OOP ZU DEN NÄCHSTEN KURSEN

(3/6)

Komplexere Software wird in der Regel in einem Team entwickelt:
Beginnend mit dem *Android*-Kurs müssen Sie lernen, gemeinsam
an einer Lösung zu programmieren.

VON OOP ZU DEN NÄCHSTEN KURSEN (4/6)

Zur Softwareentwicklung gehören auch weitere Themen, die Sie in den nächsten Semestern kennenlernen werden:
Versionsverwaltung, Projektmanagement, Dokumentation, ...




VON OOP ZU DEN NÄCHSTEN KURSEN (5/6)

Programmieren ist ein Handwerk, das in der Regel nicht gelehrt sondern nur gelernt werden kann: Viele Dingen - auch der Umgang mit komplexeren Problemen - werden Ihnen mit wachsender Erfahrung leichter fallen als noch in diesem Semester.

VON OOP ZU DEN NÄCHSTEN KURSEN (6/6)

Wenn Sie nach Ihrem Studium in der Softwareentwicklung arbeiten, werden Sie auch dort stetig an Ihren Fähigkeiten arbeiten (müssen).

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT. WENN SIE MÖCHTEN, SEHEN WIR UNS IM ANSCHLUSS IN DER ZENTRALÜBUNG!

Fragen oder Probleme? In allgemeinen Angelegenheiten und bei Fragen zur Vorlesung wenden Sie sich bitte an Alexander Bazo (alexander.bazo@ur.de ). Bei organisatorischen Fragen zu den Studienleistungen und Übungsgruppen schreiben Sie bitte Florin Schwappach (florin.schwappach@ur.de ). Bei inhaltlichen Fragen zu den Übungen, Beispielen und Studienleistungen schreiben Sie uns unter mi.oop@mailman.uni-regensburg.de .

QUELLEN

Eric S. Roberts, *The art and science of Java: an introduction to computer science, New international Edition*, 1. Ausgabe, Pearson, Harlow, UK, 2014.