Karlsruher Institut für Technologie

Karlsruher Institut für Technologie

Best Practices für „Programmieren“

eezi goIN’ Info-Bites | Yves Kirschner

Das Programmieren-Team

Die Leitenden

Dozentin
Prof. Dr.-Ing. Anne Koziolek

Übungsleiter
Dominik Fuchß, Yves Kirschner, Maximilian Walter

programmieren-vorlesung@cs.kit.edu

Ihre Tutorinnen und Tutoren

- Alexander Kusmin
- Anton Sewergin
- Bastian Franze
- Clemens Dautermann
- Florian Seligmann
- Hannes Takenberg
- Jonathan Schenkenberger
- Julian Keck
- Laura Ruple
- Leon Wittemund
- Liam Wachter
- Moritz Gstür
- Moritz Hertler
- Ralf Hüll
- Robert Brune
- Tobias Thirolf

Wer seid Ihr? → Anonyme Umfrage in Zoom

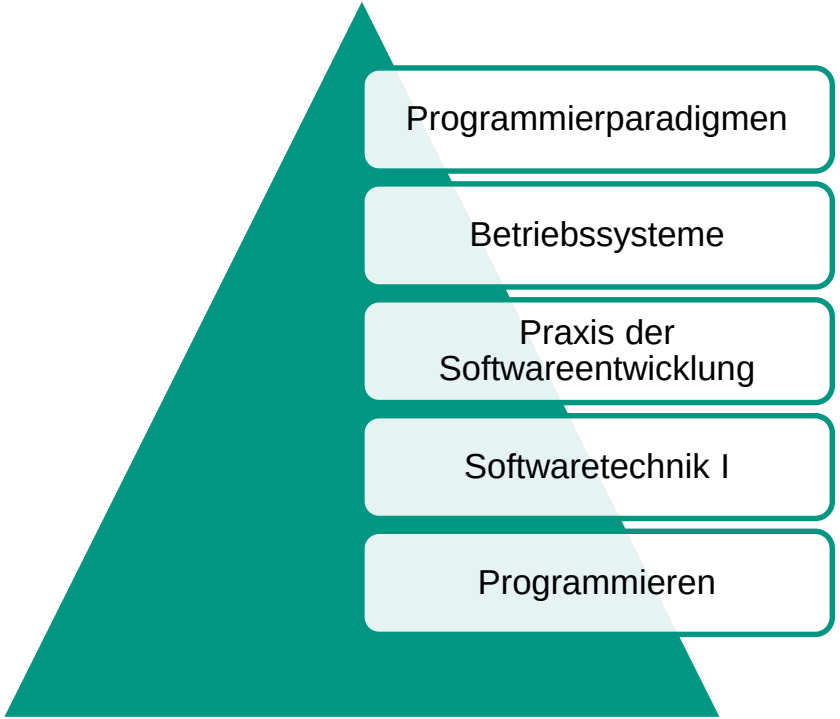
Best Practices für „Programmieren“

- Wir freuen uns über Alle, die erfolgreich das Modul abschließt
  - Teilen von Erfahrung, woran es scheitern könnte
  - Tipps sollen Frust sparen und allen beste Chancen bieten

- Nicht alle Praktiken werden beim ersten Blatt anwendbar sein
  - Es muss nicht unbedingt alles sofort verstanden werden
  - Aber spätestens bei den Abschlussaufgaben solle darauf zurückgegriffen werden

Grundlagenstudium praktische Informatik

- Programmieren bildet die Grundlage für weitere Module
- Fokus auf Orientierungsprüfung
- Programmieren sollte vor SWT abgeschlossen sein



Was Sie aus diesem Modul mitnehmen sollten

- Die Grundlagen der objektorientierten Programmierung in Java
- **Abbildung**: Problemmodellierung
  - Wie man alltägliche Probleme in Programmiersprache modelliert
- **Automatisierung**: Lösungsformulierung
  - Entwicklung von Verfahren (Algorithmen) zur Lösung einfacher Probleme
- **Abstraktion**: Problemformulierung
  - Wie man nicht nur eine Probleminstanz löst, sondern eine allgemeine Lösungsmethode findet, die für viele Probleminstanzen funktioniert
- **Sauber programmieren!**
  - Wichtig für Praxis und Zusammenarbeit

Allgemeine Tipps zur Studienorganisation

- Termine und Fristen verfolgen
  - Bereits zu Beginn des Semesters veröffentlicht: s.kit.edu/programmieren
  - In den eigenen Kalender mit Erinnerung übernehmen
  - Keine nachträglichen Anmeldungen für Prüfungsleistungen

- Studien- und Prüfungsordnung beachten
  - Arbeit muss selbstständig angefertigt werden
  - Der Übungsschein kann mehrfach wiederholt werden
  - Abschlussaufgaben können nach dem Übungsschein erworben werden
  - Die Abschlussaufgaben können nur einmal wiederholt werden
    - Muss bis zum Ende des dritten Semesters bestanden werden
    - Keine mündliche Nachprüfung

Vorläufige Termine und Fristen

- Anmeldung im **Artemis** bis zum 25.04.22, 12:00 Uhr: s.kit.edu/artemis
- Anmeldung **Übungsschein**: 01.05.22 - 01.06.22, jeweils 12.00 Uhr
- Anmeldung **Abschlussaufgaben**: 26.07.22 - 02.08.22, jeweils 12.00 Uhr
- **Präsenzübung** am 24.06.2022 zwischen 17:30 - 19:30 Uhr

- Übungsblatt 1: **25.04.22**, ca. 13:00 Uhr - 10.05.22, 06:00 Uhr
- Übungsblatt 2: 09.05.22, ca. 13:00 Uhr - 24.05.22, 06:00 Uhr
- Übungsblatt 3: 23.05.22, ca. 13:00 Uhr - 14.06.22, 06:00 Uhr
- Übungsblatt 4: 13.06.22, ca. 13:00 Uhr - 28.06.22, 06:00 Uhr
- Übungsblatt 5: 27.06.22, ca. 13:00 Uhr - 12.07.22, 06:00 Uhr
- Abschlussaufgabe 1: 25.07.22, ca. 13:00 Uhr - 23.08.22, 06:00 Uhr
- Abschlussaufgabe 2: 08.08.22, ca. 13:00 Uhr - 07.09.22, 06:00 Uhr

- Mögliche Aktualisierungen werden hier veröffentlicht: s.kit.edu/programmieren

Vorlesung und Übung nachbereiten

- Folien der Vorlesung während des Semesters aktiv nacharbeiten
  - Im Semester nacharbeiten und nicht kurz vor den Abschlussaufgaben
  - Aufzeichnungen der Vorlesung sind auf YouTube verfügbar
  - Vorlesungsfolien sind im ILIAS vorhanden: s.kit.edu/ilias
  - Übungsblätter orientieren sich an der Struktur der Vorlesung

- Beispiellösungen der Übungsblätter nachvollziehen
  - Geben Hinweise zur Bearbeitung der Abschlussaufgaben
  - Nur Beispiele dafür, wie die Aufgabe gelöst werden könnte
  - Verstehen der verwendeten Konzepte in den Beispiellösungen
    - Entwurfsentscheidungen versuchen zu verstehen
    - Das Vorgehen nachvollziehen und verstehen

Unterstützung während des Semesters

Tutorien

- Möglichkeit zum Nachfragen
- Übungsblatt vor- und nacharbeiten
- Vorbereitung auf Präsenzübung
- s.kit.edu/programmieren

eezi

- Mentoring-Stammtische und Info-Bites
- Fachliche und persönliche Unterstützung
- Kennenlernen Anderer
- s.kit.edu/studienstart

MINT

- Wiederholung des Vorlesungsinhalt
- Begrenzte Teilnehmerzahl
- Aufzeichnungen im ILIAS
- s.kit.edu/mint

Forum-Wi / (Aktive) Fachschaft

- Vielfältigen Angebot auf dem Campus
- Klausuren und Prüfungsprotokollen
- Fachliche und persönliche Unterstützung
- s.kit.edu/fsmi
- s.kit.edu/forum

Übungsblätter und Abschlussaufgaben

Übungsschein nicht erworben?

- Vorlesung beginnt bei null
- Übungsblätter orientieren sich an der Vorlesung
- Sehr steile Lernkurve
  - Sehr anspruchsvoll, wenn keine Vorkenntnisse vorhanden sind
  - Fortgeschrittene Studenten sollten für die Methodik am Ball bleiben

Übungsschein bereits erworben?

- Übungsblätter als erneute Vorbereitung nutzen
- Das Abgabesystem anschauen
- Programme selbstständig implementieren
  - Aufgaben aus dem Internet
  - Algorithmen aus den Mathematikvorlesungen

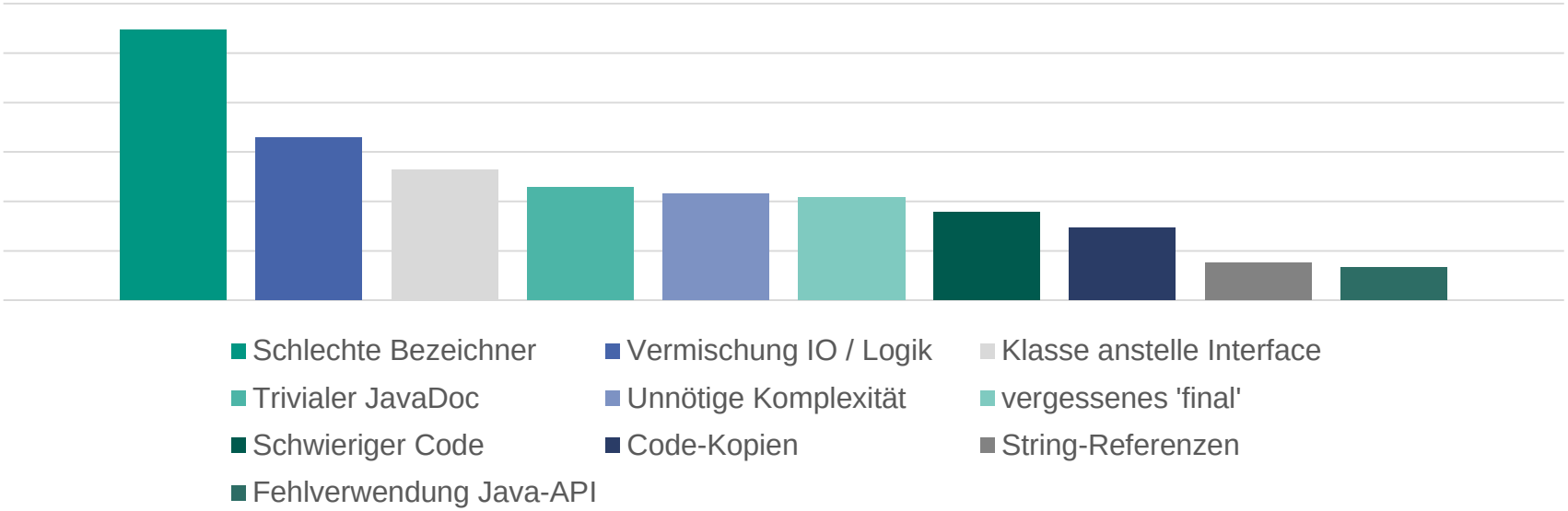
Organisatorische Tipps für die Übungsblätter

- Übungsblätter als Vorbereitung nutzen
  - Durch das eigenständige Bearbeiten lernt man am meisten
  - Praktisch fast kein zusätzlicher Lernaufwand für die Präsenzübung
  - Das letzte Übungsblatt als Vorbereitung auf die Abschlussaufgaben nutzen
  - Von Anfang an „saubere“ Programmierung denken und sich daran gewöhnen
- Früh genug anfangen, die Übungsblätter zu bearbeiten
  - Unklarheiten können so frühzeitig beseitigt werden
  - Die Übungsblätter werden schnell deutlich anspruchsvoller
  - Auf den ersten Blättern so viele Punkte wie möglich sammeln
- Früh genug anfangen, die Übungsblätter abzugeben
  - Frühzeitiges erproben von VPN und Artemis
  - Lösungen können beliebig häufig hochgeladen werden
  - Tutoren sieht nur die letzte gültige Abgabe

Praktische Tipps für die Aufgaben

- Quelltextklone vermeiden
  - Sich nicht Wiederholen und alles nur einmal im Quelltext implementieren
  - Aus doppelten Quelltextstellen versuchen, Methoden zu extrahieren
- Lange Methoden vermeiden
  - Methoden zerteilen und auf mehrere kleine (private) Hilfsmethoden verteilen
  - Dokumentation der Hilfsmethoden durch Javadoc-Kommentare
- Overengineering vermeiden
  - Quelltext so einfach wie möglich halten
  - Keine nicht benötigte Funktionalität
- Trennung der Anliegen
  - Datenstrukturen nicht nach außen geben
  - Trennung von Anwendungslogik und Benutzungsschnittstelle
- Geeignete Datentypen verwenden
  - Enums bei abgeschlossenen Mengen
  - Primitive Datentypen verwenden
  - Final, wenn nur einmal zugewiesen
  - Nur Klassenvariablen sollten statisch sein
- Herangehensweise
  - Aufgabestellung genau lesen
  - Exakt die erwartete Ausgabe liefern
  - Bei Unklarheiten sofort nachfragen

Die häufigsten methodischen Fehler der letzten Abschlussaufgaben



Weitere Bewertungsrichtlinien im Wiki: s.kit.edu/wiki

Testen

- Testen hilft, Fehler zu finden!
  - Vergleich von tatsächlichem und erwünschtem Verhalten
  - Modularer Quelltext hilft beim Testen
- Ausführliches Testen vordem Hochladen
  - Nicht auf den Public-Test verlassen
  - Den Public-Test lokal ausführen
- Selber Tests schreiben
  - Für die Kommandozeileninteraktion
  - Über eigene Main-Methode
  - JUnit-Tests schreiben

Teststrategien

- Datenbasiert
  - Beispiele auf Aufgabenblättern
- Kontrollflussbasiert
  - Suche in Datenstrukturen
  - Alle Entscheidungen einmal treffen
- Grenzwertbasiert
  - Wertebereich bei Berechnungen
  - Off-by-one-Error

Rückfragen

- Fragen und Unklarheiten können immer auftreten
  - Wenn ihr eine Frage habt, haben andere vielleicht die gleiche Frage
  - Wir bieten viele Möglichkeiten, Fragen zu stellen
- Bei Rückfragen werden Aufgaben gegebenenfalls aktualisiert
  - Lediglich minimale Aktualisierungen oder weitere Klarstellungen
  - Schaut immer mal wieder, ob die aktuelle Aufgabe aktualisiert wurde

Reihenfolge bei Fragen

- Websuche
- FAQ / Programmieren-Wiki prüfen
  - *s.kit.edu/faq*
  - *s.kit.edu/wiki*
- Diskussionsforum prüfen
  - Suchfunktion verwenden
- Im Diskussionsforum nachfragen
  - Aussagekräftigen Titel verwenden
- Eure Tutoren fragen
- Übungsleitung fragen
  - *programmieren-vorlesung@cs.kit.edu*

Weitere digitale Quellen

YouTube

- **Prof. Dr.-Ing. Anne Koziolek – Wintersemester 2020/21**: https://www.youtube.com/watch?v=LmgKH0KPH3g&list=PLfk0Dfh13pBOkHJEvgdI1zLOLeOUn2cnq
- **Prof. Dr.-Ing. Anne Koziolek – Wintersemester 2019/20**: https://www.youtube.com/watch?v=pBexdqQDFcI&list=PLfk0Dfh13pBPtDhFFeFoa9wEclDKNQsX0
- **Prof. Dr. Ralf H. Reussner – Wintersemester 2018/19**: https://www.youtube.com/watch?v=aDuHbYxGviU&list=PLfk0Dfh13pBMZa4rJWYaR8HRQLJyHT5Vy
- **Prof. Dr.-Ing. Anne Koziolek – Wintersemester 2017/18**: https://www.youtube.com/watch?v=nTlwyd1OyK8&list=PLfk0Dfh13pBP3AkkIGjCHuI7ugnJl91Na

Bücher (kostenlos über das KIT-Netzwerk erhältlich)

- **Peter Pepper – Programmieren lernen**: https://www.springer.com/de/book/9783540723639
- **Dietmar Ratz et al. – Grundkurs Programmieren in Java**: https://www.hanser-elibrary.com/doi/book/10.3139/9783446453845
- **Elisabeth Jung – Java Übungsbuch**: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&db=nlabk&AN=2347097
- **Christian Ullenboom – Java ist auch eine Insel**: https://openbook.rheinwerk-verlag.de/javainsel/
- **Kathy Sierra & Bert Bates – Java von Kopf bis Fuß**: https://oreilly.de/produkt/java-von-kopf-bis-fuss/

Tutorials

- **Oracle – The Java™ Tutorials**: https://docs.oracle.com/javase/tutorial/
- **Bradley Kjell – Introduction to Computer Science using Java**: https://chortle.ccsu.edu/java5/index.html
- Zahlreiche zusätzliche Informationsquellen sind kostenlos im Internet zugänglich: https://duckduckgo.com/?q=Java+Tutorial

Fragen und Antworten

Also gladly in English