

Programmieren I

Überblick

Institut für Automation und angewandte Informatik

```
final List<String> allResults = new ArrayList<String>();  
final Map<String, Integer> typeWordResultCount = new HashMap<String, Integer>();  
final Map<String, Integer> typePoints = new HashMap<String, Integer>();  
evaluation.put(type, typePoints);  
  
for (final Sheet sheet : this.sheets) {  
    final String sheetResult = sheet.getPlayerInput(type);  
    if (sheetResult.startsWith(start) && this.isValidWord(sheetResult, type)) {  
        validWordCountForType++;  
        allResults.add(sheetResult);  
    }  
}
```

Übersicht „Programmieren“

- Programmieren I (1. Semester)
 - 4 Vorlesungs- und Übungsstunden / Woche
 - 1 Stunde begleitetes Selbststudium / Woche
 - 7,5 (!) Stunden Selbststudium / Woche
 - Keine Prüfung

- Programmieren II (2. Semester)
 - 4 Vorlesungs- und Übungsstunden / Woche
 - 7,5 (!) Stunden Selbststudium / Woche
 - 1 Stunde begleitetes Selbststudium / Woche
 - Prüfung für beide Semester (Programmieraufgabe am Rechner)

- Σ : 8+2 Semesterwochenstunden, 9 ECTS Creditpoints

Das sagt der Studienplan (1) – Ziele/Kompetenzen Karlsruher Institut für Technologie

- Siehe Modulbeschreibung "Programmieren (T3INF1004)", S. 286/287
- Grundelemente
 - der **prozeduralen** und
 - der **objektorientierten** Programmierung kennen lernen
- Selbständig ein Programmdesign entwerfen
- Programme codieren
- Systematisches Testen und Fehlersuche durchführen
- Verschiedene **Strukturierungsmöglichkeiten** in einer modernen Programmiersprache kennen lernen
- Verschiedene **Datenstrukturen** und ihre Anwendungsmöglichkeiten
- Existierenden Code analysieren und beurteilen
- Beispielhaft in eine **Programmierungsumgebung** einarbeiten und diese einsetzen
- Logik und Boolesche Algebra in der Programmierung anwenden können



Das sagt der Studienplan (2) – Inhalte

- Kenntnisse in *prozeduraler* Programmierung
 - Algorithmenbeschreibung (z.B. Struktogramm)
 - Datentypen
 - E/A-Operationen
 - Operatoren
 - Kontrollstrukturen
 - Funktionen
 - Stringverarbeitung
 - Strukturierte Datentypen
 - Dynamische Datentypen
 - Dateiverarbeitung
 - Rekursion
 - Speicherverwaltung

Das sagt der Studienplan (3) – Inhalte

- Kenntnisse in objektorientierter Programmierung
 - Objektorientierter Programmentwurf (z.B. Klassendiagramme)
 - Idee der objektorientierten Programmierung
 - Klassenkonzept
 - Operatoren / Methoden
 - Überladen von Operatoren / Methoden
 - Vererbung und Überschreiben von Operatoren / Methoden
 - Polymorphismus
 - Template
 - Klassenbibliotheken
 - Speicherverwaltung (Garbage Collector)

Und was machen wir tatsächlich? (1)

■ Programmieren I

- Programmiersprache *Java*
- Entwicklungsumgebungen
- Primitive Datentypen
- Kontrollstrukturen
- Arrays
- Klassen und Objekte
- Strings und Wrapperklassen
- Vererbung, Methoden, Polymorphismus, Pakete
- Fehlerbehandlung (Exceptions)
- Input/Output
- Programmdokumentation



Java-Logo
Bild-Quelle: Wikipedia

Und was machen wir tatsächlich? (2)

■ Programmieren II

- Abstrakte Klassen, Interfaces und innere Klassen
- Grafische Benutzeroberflächen (Swing)
- Ereignisverarbeitung (Events)
- Datenstrukturen, Collection-Framework, generische Datentypen
- Aufzählungstypen (enum)
- Nebenläufigkeit (Threads)
- Arbeiten mit XML- und JSON-Dokumenten
- JUnit-Testing
- Neuerungen seit Java 8 (Lambda-Expressions etc.)

Was machen wir nicht?

- Entwurf größerer Programmsysteme
➔ Software-Engineering
- Algorithmen-Entwurf und -bewertung
➔ Theoretische Informatik I+II
- Services (SOAP, REST, ...)
➔ Verteilte Systeme

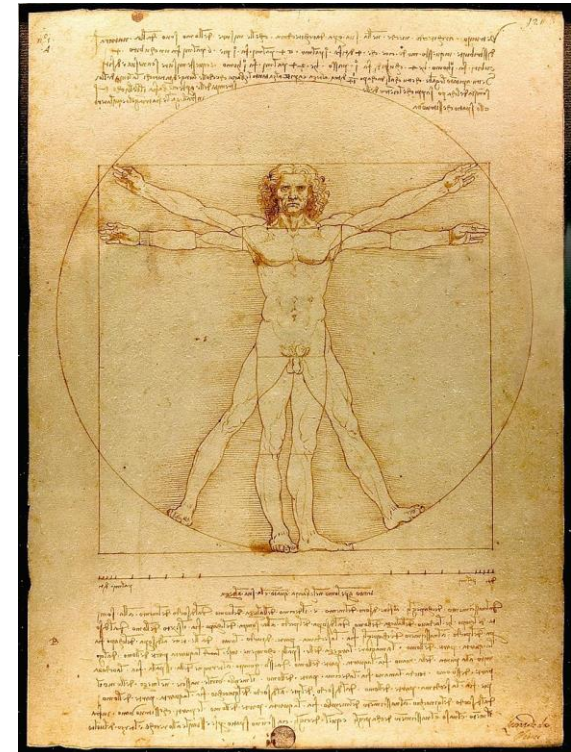
Ablauf der „Vorlesungen“ (4h)

- ca. 10-20 % Wiederholung (Lösung Übungsaufgaben)
 - ca. 40-60 % Vorlesung (neuer Stoff)
 - ca. 20-40 % Übungen (Übungsaufgaben)
-
- ca. 1 Std. „begleitetes Selbststudium“ (Übungen)
-
- ca. 0-5h Heimarbeit (Fertigstellung Übungsaufgaben)
[Zur Erinnerung:
Studienplan: „+7,5h“]



Prinzipien

- Fragen stellen (jederzeit!), Diskussionen anzetteln
- Learning by doing
(„Programmieren lernt man nur durch Programmieren!“)
 - Laptop notwendig
- Vertraute Umgebung nutzen
 - eigener Laptop
 - Betriebssystem und IDE nach eigenem Gusto
- Gegenseitige Hilfe
 - s. „anderen erklären“ ☺
- Hilfe bei Fragen und Problemen
- Eigenverantwortung („Selber groß!“)



Der vitruvianische Mensch (Leonardo da Vinci)
Quelle: Luc Viatour / <http://www.Lucnix.be>

Prüfung (Programmentwurf)

- Nach dem 2. Semester und der folgenden Praxisphase (Voraussichtlich Anfang Oktober)
 - 120 Minuten
 - Programmieraufgabe
 - Bearbeitung am Rechner, Abgabe: Quellcode
 - Hilfsmittel: Folien, Übungen, Beispiele

- Angebot: Übungs-Programmentwurf am Ende des 1. und 2. Semester
 - „Echte“ Bedingungen
 - Standortbestimmung
 - Ohne Bewertung

■ Motivationsschreiben-1

- „Frühzeitig nach eigenen Projekten ... schauen, bei denen Dein Interesse geweckt wird.“
- „Idealerweise schließt Du dich für das Projekt mit einem Kommilitonen oder anderem Freund zusammen.“

■ Motivationsschreiben-2

- „Während eures Studiums werdet ihr immer wieder Projekte bearbeiten, die verschiedenste Kenntnisse und Skills erfordern.“
- Am Anfang eures Studiums habt ihr noch verhältnismäßig viel Zeit. Nutzt diese, um diese Erfahrung zu sammeln.

■ Motivationsschreiben-3

- „Programmieren lernt man meiner Erfahrung nach zu 20% durch Information von außen, 80% aber durch selbst ausprobieren.“
- „Es ist egal wie du Programmieren lernst, Hauptsache du beschäftigst dich damit und hältst dein Interesse dafür aufrecht.“

Literatur

- „**Grundkurs Programmieren in Java**“, Ratz et al., ISBN 978-3-446-45212-1
<https://www.hanser-elibrary.com/doi/book/10.3139/9783446453845>
(Das Herunterladen funktioniert vermutlich nur aus dem KIT/DHBW-Netz)
- „**Java (1. u. 2. Band)**“, Heusch, RRZN Hannover
<https://www.luis.uni-hannover.de/fileadmin/buecher/leseproben/java-b1-a10-lese.pdf> (nur .1 Band)
- „**Java ist auch eine Insel**“ (14. Aufl.), Christian Ullenboom, ISBN 978-3-8362-6721-2
<http://www.tutego.de/javabuch/Java-ist-auch-eine-Insel/12/> (12. Aufl.)
- „**Java als erste Programmiersprache**“, Joachim Goll et al., ISBN 978-3-658-12117-4
- **Java-Tutorials** (von Oracle) (zu diversen Themen)
<https://docs.oracle.com/javase/tutorial/>
- **API-Dokumentation** (von Oracle) <https://docs.oracle.com/javase/8/docs/api/>
<https://docs.oracle.com/en/java/javase/13/docs/api/>
- Und: Das Netz ist Dein Freund 😊



**Heusch
Ratz**

Online-Kurse für verschiedene Programmiersprachen (1)

■ Verspielte:

CodinGame: <https://www.codinggame.com/start> (alle Sprachen)

CodeCobalt: <https://codecombat.com/play> (Python, JavaScript)

■ Ernsthafte:

a) Community-unterstützt:

freeCodeCamp: <https://www.freecodecamp.org/> (Front-end, Back-end, Git, JavaScript; Frameworks wie Node.js, React.js)

Codewars: <https://www.codewars.com/>

b) Alleine (asynchron) oder "ge-timed" - mit Lehrer:

edX: <https://www.edx.org/course/java-fundamentals-for-android-development-0>

Coursera: <https://www.coursera.org/courses?query=Java> (Bsp. Java)

c) Alleine:

Sololearn: <https://www.sololearn.com/>

Online-Kurse für verschiedene Programmiersprachen (2)

■ Kostenpflichtige:

Udemy:

<https://www.udemy.com/courses/search/?ref=home&src=ukw&q=java>

(auf Deutsch)

Codecademy: <https://www.codecademy.com/learn/learn-java>

Online-Ressourcen von uns Dozenten

- Vorlesungsfolien
- Übungsblätter
- Musterlösungen

- → www.iai.kit.edu/javav1/

- Kontakt:
 - werner.geiger@partner.kit.edu
 - wolfgang.suess@kit.edu
 - thorsten.schlachter@kit.edu
 - christian.schmitt@kit.edu

