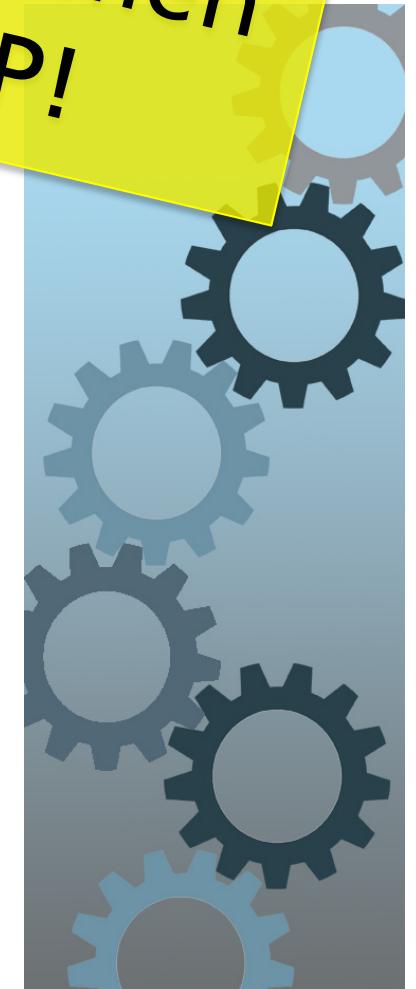


*Herzlich Willkommen
im Modul PCP!*
Programming Concepts & Paradigms

Organisatorisches & allgemeine Informationen

Prof. Dr. Ruedi Arnold

ruedi.arnold@hslu.ch



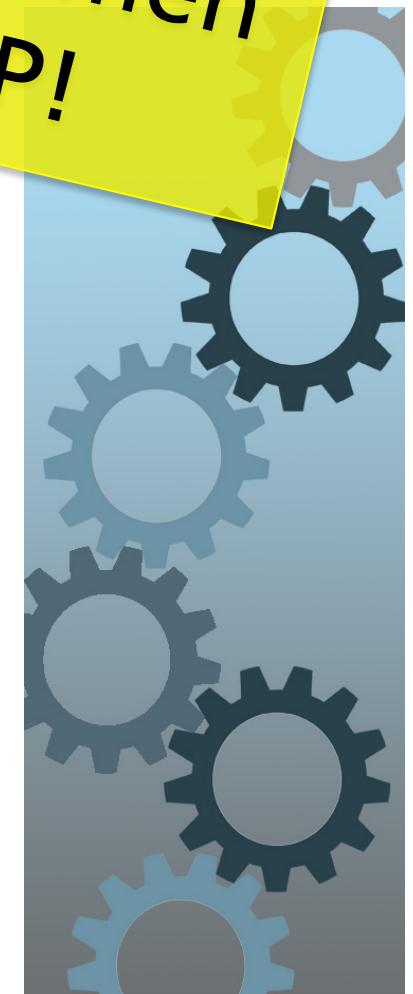
*Herzlich Willkommen
im Modul PCP!*

Programm

O

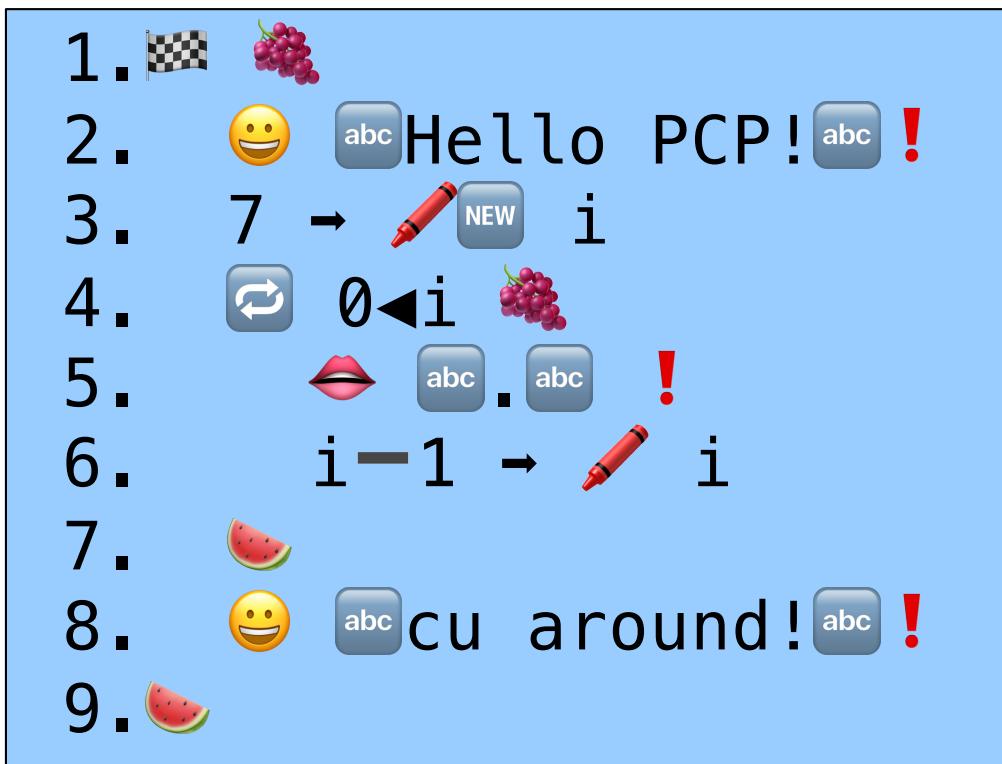
allgemeinen

edi.arnold@nslu.ch



...Emojicode! ;-)

...können Sie Code lesen?



Emojicode

Emojicode is an open-source, full-blown programming language consisting of emojis.

```
$ emojicodec hello.emojic  
$ ./hello  
Hello PCP!  
.....cu around!  
$
```

<https://www.emojicode.org/>

Die Programmiersprache ArnoldC (Falls Sie eher nicht ein Emoji-Typ sind...)

„ArnoldC is an imperative programming language where the basic keywords are replaced with quotes from different Schwarzenegger movies.“

- <http://lhartikk.github.io/ArnoldC/>



<https://www.youtube.com/watch?v=TKTL2EDTFS0>

IT'S SHOWTIME!..

- Was gibt das folgende Programm aus? (Code im Ilia)

```
1. IT'S SHOWTIME
2. HEY CHRISTMAS TREE isLessThan4
3. YOU SET US UP @NO PROBLEMO
4. HEY CHRISTMAS TREE n
5. YOU SET US UP 0
6. STICK AROUND isLessThan4
7. GET TO THE CHOPPER n
8. HERE IS MY INVITATION n
9. GET UP 1
10. ENOUGH TALK
11. TALK TO THE HAND n
12. GET TO THE CHOPPER isLessThan4
13. HERE IS MY INVITATION 4
14. LET OFF SOME STEAM BENNET n
15. ENOUGH TALK
16. CHILL
17. TALK TO THE HAND "Welcome to PCP!"
18. YOU HAVE BEEN TERMINATED
```

BEMERKUNG: Sie
können programmieren
und Code lesen - sie
verstehen also
Programme, oder?.. ;-)

...ArnoldC: Stop whining and start coding!



<https://www.youtube.com/watch?v=El8e2ujXe8g>

```
taarnold$ java -jar ArnoldC.jar pcp.arnoldc
taarnold$ java pcp
1
2
3
4
Welcome to PCP!
taarnold$
```

```
1. IT'S SHOWTIME          // Begin Main
2. HEY CHRISTMAS TREE isLessThan4 // Declare Int
3. YOU SET US UP @NO PROBLEMO // Set Initial Value ("@NO PROBLEMO" = TRUE)
4. HEY CHRISTMAS TREE n    // Declare Int
5. YOU SET US UP 0         // Set Initial Value
6. STICK AROUND isLessThan4 // While
7. GET TO THE CHOPPER n   // Assign Variable
8. HERE IS MY INVITATION n // Set Value
9. GET UP 1                // Plus Operator
10. ENOUGH TALK           // End Assign Variable
11. TALK TO THE HAND n    // Print
12. GET TO THE CHOPPER isLessThan4 // Assign Variable
13. HERE IS MY INVITATION 4 // Set Value
14. LET OFF SOME STEAM BENNET n // Greater Than
15. ENOUGH TALK           // End Assign Variable
16. CHILL                 // End While
17. TALK TO THE HAND "Welcome to PCP!" // Print
18. YOU HAVE BEEN TERMINATED // End Main
```

Code-Vergleich: ArnoldC vs. Java

```
1. IT'S SHOWTIME
2. HEY CHRISTMAS TREE isLessThan4
3. YOU SET US UP @NO PROBLEMO
4. HEY CHRISTMAS TREE n
5. YOU SET US UP 0
6. STICK AROUND isLessThan4
7. GET TO THE CHOPPER n
8. HERE IS MY INVITATION n
9. GET UP 1
10. ENOUGH TALK
11. TALK TO THE HAND n
12. GET TO THE CHOPPER isLessThan4
13. HERE IS MY INVITATION 4
14. LET OFF SOME STEAM BENNET n
15. ENOUGH TALK
16. CHILL
17. TALK TO THE HAND "Welcome to PCP!"
18. YOU HAVE BEEN TERMINATED
```

```
public static void main(String[] args) {
    boolean isLessThan4 = true;

    int n = 0;

    while (isLessThan4) {
        n++;

        System.out.println(n);
        isLessThan4 = 4 > n;
    }

    System.out.println("Welcome to PCP!");
}
```

...Sprachen & Geschmäcker sind verschieden! ;-)

Demo: Syntax-Fehler...

Brief overview of the keywords

Check the [wiki](#) for more details

False I LIED

True NO PROBLEMO

If BECAUSE I'M GOING TO SAY PLEASE

Else BULLSHIT

EndIf YOU HAVE NO RESPECT FOR LOGIC

While STICK AROUND

EndWhile CHILL

PlusOperator GET UP

1. ITS SNOWTIME
2. HEY CHRISTMAS TREE isLessThan4
3. YOU SET US UP @NO PROBLEMO

```
n0002065:ArnoldC taarnold$ java -jar ArnoldC.jar pcp.arnoldc
Exception in thread "main" org.parboiled.errors.ParsingException: WHAT THE FUCK DID I DO WRONG:
Invalid input 'N', expected 'H' (line 1, pos 7):
IT'S SNOWTIME
^
```

```
at org.arnoldc.ArnoldParser.parse(ArnoldParser.scala:203)
at org.arnoldc.ArnoldGenerator.generate(ArnoldGenerator.scala:10)
at org.arnoldc.ArnoldC$.main(ArnoldC.scala:21)
at org.arnoldc.ArnoldC.main(ArnoldC.scala)
```

14. LET OFF SOME STEAM BENNET n
15. ENOUGH TALK
16. CHILL
17. TALK TO THE HAND "Welcome to PCP!"
18. YOU HAVE BEEN TERMINATED

MethodArguments I NEED YOUR CLOTHES YOUR BOOTS AND YOUR MOTORCYCLE

Return I'LL BE BACK

EndMethodDeclaration HASTA LA VISTA, BABY

CallMethod DO IT NOW

AssignVariableFromMethodCall GET YOUR ASS TO MARS

DeclareInt HEY CHRISTMAS TREE

SetInitialValue YOU SET US UP

BeginMain IT'S SHOWTIME

EndMain YOU HAVE BEEN TERMINATED

Print TALK TO THE HAND

ReadInteger I WANT TO ASK YOU A BUNCH OF QUESTIONS AND I WANT TO HAVE THEM ANSWERED IMMEDIATELY

AssignVariable GET TO THE CHOPPER

SetValue HERE IS MY INVITATION

EndAssignVariable ENOUGH TALK

ParseError WHAT THE FUCK DID I DO WRONG

```
$ java -jar ArnoldC.jar pcp.arnoldc
$ java pcp
```

ArnoldC & TALK TO THE HAND...



<https://www.youtube.com/watch?v=dQ6m8ztEzfA>

Bemerkungen zu ArnoldC

- ArnoldC-Programme werden nach Java-Bytecode kompiliert mit Hilfe von ArnoldC.jar
 - Interessanter Ansatz unter Verwendung von entsprechenden Java- bzw. Scala-Frameworks für das Parsing, bzw. Byte-Code-Manipulationen
 - Siehe <https://github.com/lhartikk/ArnoldC/wiki/ArnoldC>
 - Mehr zu Compilern und (eigenen) Sprach-Grammatiken später im Modul
- Zwei abschliessende Hinweise
 - ArnoldC i& Emojicode sind NICHT prüfungsrelevant ;-)
 - Sie werden in diesem Modul viel Code lesen & schreiben :-)



Organisatorisches und allg. Infos zu diesem Modul

Programm heute

- Intro: Emojicode & ArnoldC ✓
- Organisatorisches & Infos
 - Motivation & Grobziele
 - Vorstellung
 - Semesterplan
 - Lernziele
- Einführung: Programmierparadigmen
 - (eigener Foliensatz)

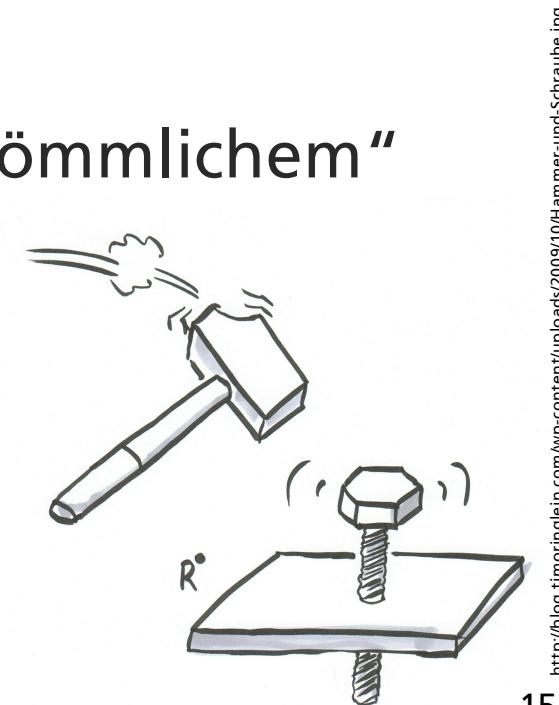
Motivation / Gründe für dieses Modul

- Die (Programmier-)Welt besteht nicht nur aus Befehlen und Objektorientierung wie in Java (resp. C, C#, usw.)
- Programmiersprachen entwickelt sich weiter
 - z.B. Java: Generics, Lambdas, Streams, ...
- Neue Programmiersprachen entstehen
 - z.B. Kotlin (2016), Swift (2014), C# (2000), ArnoldC ;-)
- Viele „moderne“ Programmiersprachen (z.B. Kotlin, Swift oder Scala) setzen auf mehrere Programmierparadigmen
 - Java ist z.B. seit Version 8 (2014) neu auch funktional(er)
- Kompetente ProgrammierInnen kennen und verstehen verschiedene Programmierparadigmen und –konzepte und können das Passende auswählen
 - Handwerk und Werkzeuge kennen & verstehen!

Grobziele für dieses Modul

- Sie kennen verschiedene Programmierparadigmen, können diese einordnen und sinnvoll einsetzen
- Sie setzen sich vertieft mit deklarativer Programmierung auseinander, konkret mit
 - logischer Programmierung (Prolog)
 - funktionaler Programmierung (Scheme)
- Dazu lernen Sie (selbstständig) weitere Programmierkonzepte jenseits von „herkömmlichem“ Mainstream wie Java oder C#
 - z.B. Closures, Optionals oder Matching

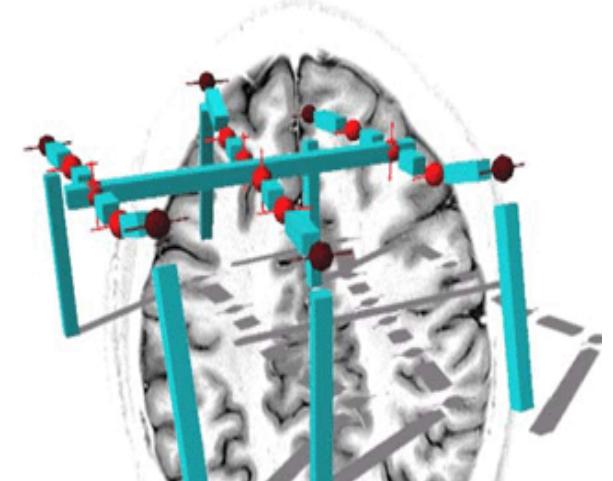
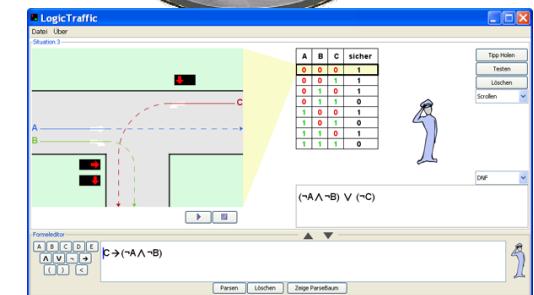
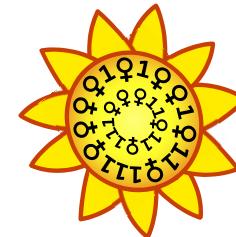
**...und sie schlagen so möglichst Schrauben
in Zukunft nicht mit dem Hammer ein!**



<http://blog.timoringlain.com/vvp-content/uploads/2009/10/Hammer-und-Schraube.jpg>

getShortCV("Ruedi Arnold")

- 1996 Matura in Altdorf UR
- 2002 Dipl. Informatik Ing. ETH
- 2003 Höheres Lehramt
- 2007 Dr. sc. ETH Zürich
- 2008-11 Ergon Informatik AG, Zürich
 - Entwickler, Projektleiter & Lehrlingsbetreuer
- 2009-11 EB Zürich (Kursleiter)
- 2012- HSLU (Dozent Informatik)

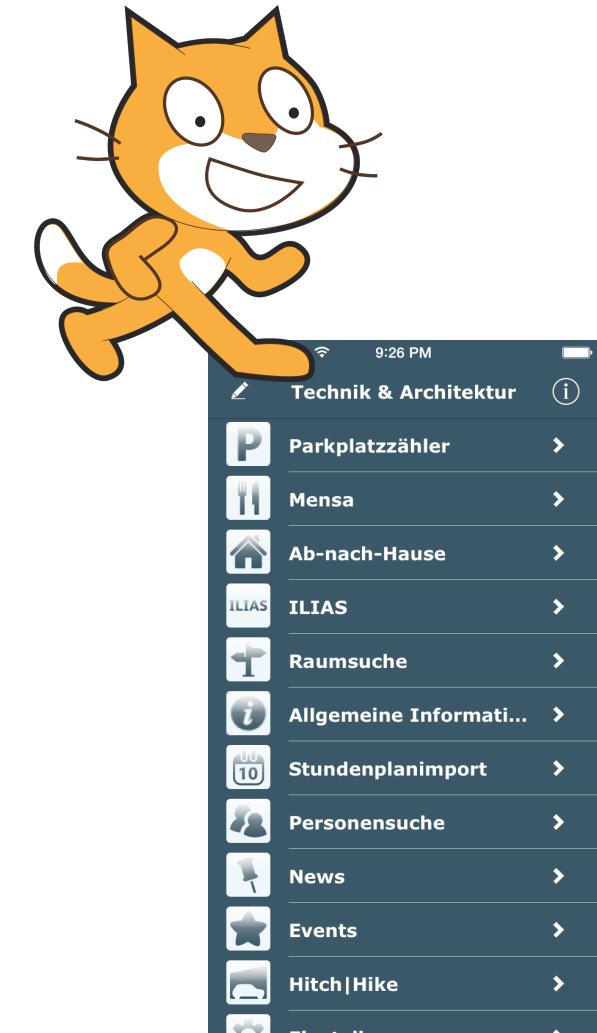
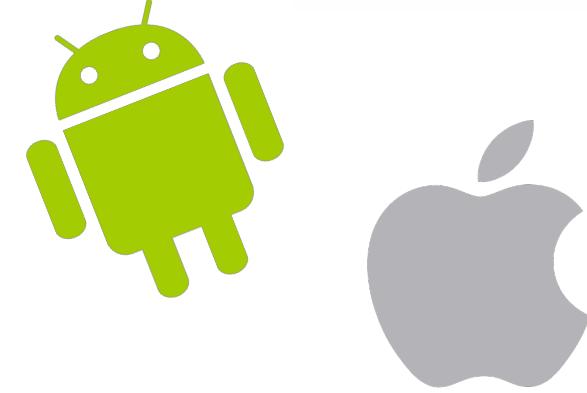




...Informatik ist für mich ein sehr spannender Beruf, privat habe ich andere "Hobbys"

Prof. Dr. Ruedi Arnold @ HSLU

- Bsc-Module: PCP, MOBLAB, MOBPRO, IOS, IDCLAB, PLAB, ...
- Teil-Studiengangleitung "Master in Arts Fachdidaktik Medien & Informatik"
- Forschungs-Projekte Team "Algorithmic Business" (ABIZ.ch)
- MSE-Advisor - Master was für Sie?
- Programmierworkshops für Jugendliche
 - <http://hslu.ch/scratch>
- Betreuung Mobile Apps für HSLU
 - Blog: <http://blog.hslu.ch/mobapp/>



...now who are you?

- Name, Vorname, Wohnort
 - HSLU-Semester
 - "Background"
 - Job / Firma
 - Programmiererfahrungen
 - Sprachen, Technologien, ...
- Teilnehmenden?!**

→ Motivation für PCP?



<https://ksassets.timeincuk.net/wp/uploads/sites/55/2018/07/Dave-Grohl-Live-920x584.jpg>

Umfrage...

...digital & online! ☺

<http://etc.ch/naBc>

→ Bitte jetzt gleich
ausfüllen, Danke!



...now who are you?

Spontanmeldungen zu:



<https://ksassets.timeincuk.net/wp/uploads/sites/55/2018/07/Dave-Grohl-Live-920x584.jpg>

- Lieblings-Programmiersprache?
- Job? (Coole Entwicklerbude?!)
- Motivation für PCP?
- Whatelse?..
 - ...ArnoldC?! ;-)

Programming Concepts & Paradigms

- | | |
|------------------|---|
| ■ Typ | Wahl-Kernmodul |
| ■ Anzahl Credits | 6 ECTS (d.h. ~180h!) |
| ■ Tag & Zeit | Donnerstag 12:50 - 15:10
Freitag 09:05 - 11:25 |
| ■ Ort | S1A.202 |
| ■ Dozenten-Team | Roger Diehl (roger.diehl@hslu.ch)
Ruedi Arnold (ruedi.arnold@hslu.ch,
modulverantwortlicher Dozent) |
| ■ Assistent | Pascal Wullschleger
(pascal.wullschleger@hslu.ch) |



Modulkurzbeschrieb

Einführung in verschiedene Programmierparadigmen. Einsatz von imperativen, objektorientierten, deklarativen, funktionalen und logischen Programmiersprachen. Diskussion von charakteristischen Programmierkonzepten und deren typischen Einsatzgebieten. Formale Grammatiken und Grundkonzepte vom Compiler-Bau, Erstellung eines eigenen Compilers. Übungen mit verschiedenen Sprachen zur praktischen Vertiefung.

→ Fokus: verschiedene Programmierparadigmen kennen, verstehen und praktisch anwenden können!

Lernziele Fachkompetenz

- Die beiden deklarativen Programmierparadigmen funktional und logisch verstehen und diese an Beispielen erklären und exemplarisch anwenden können
- Die folgenden sieben Programmierparadigmen einordnen und erläutern können: imperativ, strukturiert, prozedural, objektorientiert, deklarativ, funktional und logisch
- Die sieben oben genannten Programmierparadigmen exemplarisch anhand von geeigneten Programmiersprachen illustrieren und anwenden können
- Die mit Java 8 eingeführten funktionalen Sprachkonstrukte von Java kennen und diese in Beispielen anwenden können
- Wissen, was eine formale Sprache ist und mit formalen Grammatiken von Programmiersprachen arbeiten können
- Verstehen, wie Compiler funktionieren und mit geeigneteren Hilfsmitteln selbstständig eigene Compiler bauen und anwenden können

Lernziele Methodenkompetenz

- In gemischt programmier-sprachlicher Umgebung analytisch und problembezogen vorgehen können
- Englische und deutsche Fachliteratur lesen und interpretieren können
- Selbstständig ein Fachthema aufarbeiten und im Plenum präsentieren können

Lernziele Personalkompetenz

- Aktiv, konstruktiv und zielorientiert im Team arbeiten und lernen können
- Sich zeitlich organisieren und verbindlich verhalten können
- Selbstständig Wissenslücken und Unsicherheiten identifizieren und beheben können

Voraussetzung: Basic-Module & VSK besucht

- Alle Pflicht-Module Stufe Basic abgeschlossen
- Modul VSK besucht
- Hinweis: Programmierwissen und -erfahrung in Java wird vorausgesetzt, konkret:
 - Java/C Basiswissen: Syntax & Semantik von Bedingungen, Schleifen, Variablen, Typen
 - OOP-Grundkonzepte: Klassen & Objekte, Instanziierung, Vererbung & Polymorphismus

Semesterplan im Bild

SW 1	SW 2	SW 3	SW 4	SW 5	SW 6	SW 7
Einführung Paradigmen Imperative vs. prozedurale Programmierung, Diskussion OOP, ADTs, objektorientiert) (Ruedi Arnold)	Logische Programmierung I	Logische Programmierung II	Logische Programmierung III	Funktionale Programmierung I	Funktionale Programmierung II	Funktionale Programmierung III
	Einführung Prolog, Fakten, Regeln, Abfragen, Beweissuche, Backtracking, Operatoren	Optimierungen (Endrekursion, Aufbau, Anwendungen), Autoren (HTTP, JSON)	Schwache Negation, Finite Domänen, Sudoku, usw., Modelle (HTTP, JSON)	Einführung Scheme, Sprachelemente, funktionale Modellierung, strukturierte Daten und Datenabstraktion	Lambda-Kalkül, rekursive Datentypen, Rekursionsfunktionen höherer Ordnung	lokale Definitionen, lexikalisches Scoping, Spezialitäten und Bindungen
	logische Programmierung: Prolog (Ruedi Arnold)			funktionale Programmierung: Scheme (Roger Diehl)		
	LF1, LF2, LF3					
SW 8	SW 9	SW 10	SW 11	SW 12	SW 13	SW 14
Java 8 & „functional Thinking“ I	Java 8 & „functional Thinking“ II	Compiler I & Kickoff Team-Projekt	Compiler II	Team-Projekt I	Team-Projekt II	Abschluss Team-Projekt
Lambdas, funktionale Interfaces, Methodenreferenzen, default und statische Interface-Methode Streams	Lambdas & Streams im Einsatz, weitere Java „functional Thinking“, funktionaler Bausteine	Compiler, Interpret, formale Sprache, Grammatiken, virtuelle Maschinen, Parser-Generatoren	Parser, Abstract Syntax Tree (AST), generierter Compiler	Coaching, Besprechung Abschlusspräsentation	Coaching, Besprechung Abschlusspräsentation	Abschlusspräsentation inkl. Demos
Java 8 & „Functional Thinking“ (Ruedi Arnold)		Compiler & Grammatiken (Roger Diehl)		individuelle Team-Projekte (Ruedi Arnold & Roger Diehl)		
		Themenwahl		LF1, LF2, LF3		
FS 2020 V1.0		PCP - Org. & Infos				

Grobüberblick Kursinhalt

- SW1: Einführung Programmierparadigmen (imperativ, strukturiert, prozedural, objektorientiert, deklarativ, funktional, logisch), Diskussion C und Java
- SW2-4: Logische Programmierung mit Prolog
- SW5-7: Funktionale Programmierung mit Scheme
- SW8-9: Java 8 & „Functional Thinking“
- SW10-11: Compiler & Grammatiken
- SW10-14: Individuelle Team-Projekte

Didaktisches Konzept

- SW1-11: „Vorlesung + Übungen“
 - 2 x ca. 2 Lektionen geführtes Studium
 - 2 x ca. 1 Lektion geführtes Selbststudium
 - Lektüre / Repetition Folien inkl. Code-Bsp.
 - Aufgaben lösen / besprechen
 - Aufgaben abgeben / vorzeigen
- SW10-14 „Team-Projekt“
 - 2er Gruppen
 - Individuelle Themen zu diversen Programmiersprachen
 - Abschlussbericht, Präsentation & Diskussion im Plenum

Zulassung und Nachweis

- Zulassung (Testat)
 - Min. 5 Wochenübungen (von 9) zufriedenstellend gelöst und einem Dozenten oder Assistenten individuell präsentiert, davon zwingend Übung SW4 und SW7
 - 2 Ilias-Übungen gelöst (je 1 zu Prolog & Scheme)
- Nachweis (**neu und anders als im Modulbeschrieb!**)
 - **1/3:** Aktive Teilnahme an einem individuellen 2er-Team-Projekt, inkl. Präsentation und Diskussion der erarbeiteten Resultate im Plenum.
 - **2/3:** Schriftliche Modulendprüfung, **2h (3h)**

→ **Neuer Modus ok für Sie?**

... Dozenten/Assistenten Übungen präsentieren?

- Voraussetzung: Sie haben die Wochenübung gelöst
- Ablauf: Doz./Ass. wählt die zu präsentierenden Aufgaben aus, Studierende(r) zeigt ihre/seine Lösung und beantwortet Fragen vom Doz./Ass.
 - Falls Lösung ok und Fragen zufriedenstellend beantwortet, wird Übung als Teil vom Testat akzeptiert
- Übungspräsentation ist möglich bis max. 2 Wochen nach Ausgabe der Übung
 - z.B.: die Übung der SW4 muss bis spätestens Ende SW6 präsentiert sein

Modus Übungspräsentationen

- Koordination mittels Doodle-Umfrage (FIFO)
 - URL jeweils auf Leinwand/Whiteboard anfangs Übungsblock
- Präsentation jeweils zu zweit, Fragen von Dozierenden/Assistenten werden individuell beantwortet
- Wann können Übungen präsentiert werden?
 - Do & Fr nach Vorlesung, d.h.:
 - **Do ab ca. 14:30**
 - **Fr ab ca. 10:45**

Unterlagen

- Folien
- Übungsblätter
- Empfohlene Bücher / Kapitel

- Hinweis: Es werden keine Lösungsvorschläge zu den Übungen abgegeben
 - Besprechung der Übungen in den Übungsstunden, ggf. dort nachfragen
 - Weitere Fragen können im Forum gestellt werden

Prüfungsrelevanter Stoff

- Vorlesung: Folien inkl. Code & Demo dazu
- Übungen
 - Selber lösen / Lösungen nachvollziehen
- Empfohlene Literatur (Bücher & Links)
- Weitere Quellen (Internet, Bücher, ...)

...etwa in dieser Reihenfolge

- D.h. primär wichtig: Vorlesungs- und Übungsstoff verstanden bzw. nachvollzogen haben!

Benötigte Software

- In SW 1-9 benötige Software:
 - C-Compiler
 - z.B. gcc, <http://www.mingw.org/>, oder built-in von Visual Studio oder Xcode
 - SWI-Prolog (<http://www.swi-prolog.org/>)
 - Dr. Racket (<http://racket-lang.org/>)
 - JDK ≥ 8.0
 - Empfehlung: das aktuelle LTS JDK 11.0.6
(<https://adoptopenjdk.net/?variant=openjdk11&jvmVariant=hotspot>)

Fragen, Anmerkungen?

- Fragen während der Veranstaltung jederzeit gerne an Dozierende, bzw. Assistierende! (Danach im Forum)

→ Mein Tipp: Aktive Teilnahme, mitdenken, Fragen stellen, programmieren, über Tellerränder schauen, diskutieren, ausprobieren, Übungen lösen, ...

– Programmieren lernt man, indem man es tut!



http://www.mcstudy.de/newsletter/archiv/images/bild_tellerrand.jpg

PCP - Org. & Infos



http://fc09.deviantart.net/f550ff/2009/301/d/0/Raging_Code_Monkey_by_Plognark.jpg

37