Modulbezeichnung:	**Geschichte und Prinzipien der Programmiersprachen (PS-PoPL-GdP)** (History and Principles of Programming Languages)	**7.5 ECTS**
Modulverantwortliche/r:	Hans Jürgen Schneider	
Lehrende:	Hans Jürgen Schneider, Ronald Veldema	

Startsemester: WS 2014/2015	Dauer: 2 Semester	Turnus: halbjährlich (WS+SS)
Präsenzzeit: 90 Std.	Eigenstudium: 135 Std.	Sprache: Deutsch

Lehrveranstaltungen:

Die Vorlesungen bauen **nicht** aufeinander auf und können daher in beliebiger Reihenfolge gehört werden. Daher kann auch die Modul-Prüfung ebenfalls nach jedem beliebigen Semester abgelegt werden.

Geschichte der Programmiersprachen (WS 2014/2015, Vorlesung, 2 SWS, Hans Jürgen Schneider)
Principles of Programming Languages (SS 2015, Vorlesung, 2 SWS, Ronald Veldema)
Exercises for Principles of Programming Languages (SS 2015, Übung, 2 SWS, Ronald Veldema)

Inhalt:

Part I: History of Programming Languages:

We study the historical development of programming languages from the very beginning in the 1950s. We examine the first languages, such as Fortran, Cobol, Algol, Lisp, and which alterations language concepts had to pass through until today.
We examine the development of the programming paradigms and in which way they have influenced programming languages. This includes structuring in the small (data structures and control flow) as well as in the large (modules, object classes, type classes, and asynchronous processes).
We also discuss the rise of description methods (description of syntax and semantics) and the invention of compiler techniques, especially the predecessors of the LR-analysis.

Part II: Principles of Programming Languages

We will examine the ideas and concepts that are used and seen in programming languages. In this it is NOT important to us how a feature is implemented, but only how to **use** a language idea/paradigm. We will examine both current and past programming language concepts and approach the lecture by catagorization over ideas and with lots of languages to examplify the ideas.
Sample language concepts we will examine include: logic, functional, fuzzy logic, statistical, imperative, object-oriented, specification/modeling, annotational dialects, access-oriented, etc, etc, etc. Sample languages that will will look at: Occam, Haskel, Ada, Chuck, C, Spec#, Pascal, Modula, Basic, NuSMV, Sisal, Promela, Python, Icon, sh, etc, etc, etc.
Rationale 1: it turns out that there is no single programming language 'to rule them all'. For each problem domain, there is a programming language that is best suited.
Rationale 2: many applications currently written use multiple interacting programming languages.
Rationale 3: language and thought are tightly interwoven. Using/thinking in another language might give you new insights in how problems can be solved. This course may give you some new views.
Prerequisite: knowledge/experience with at least one programming language.

Lernziele und Kompetenzen:

The students

- explain the advantages of today's language concepts
- judge upcoming language proposals in the light of previous development
- analyze papers related to former language concepts
- explain different typical properties and means of expression of programming languages, compilers and interpreters
- name basic concepts (wrt. control flow, data types, modularization, type safety, etc.) of imperative programming languages and their concrete forms in typical languages (e.g. C, Ada, Pascal, Modula)
- explain the main difference between imperative and object oriented programming and languages
- illustrate typical concepts of object oriented languages (e.g. classes, inheritance, overloading, polymorphism) and their concrete forms in typical languages (e.g. C++, Java)

- describe the principles of functional programming languages (referential transparency, type inference, currying, lazy evaluation, pattern matching, monads) and their concrete forms in typical languages (e.g. Miranda, Haskell, Lisp, OCaml)
- explain how "programs" in logic programming are being evaluated and know the role of unification
- describe typical problems of and solutions for parallel programming (atomicity, races, synchronization, message passing)
- select an adequate domain specific language depending on the problem domain to be solved
- apply aspect oriented programming (e.g. AspectC++, AspectJ) and web languages (e.g. Perl, PHP, JavaScript)
- explain the purpose and benefits of specification languages (e.g. Coq, JML, OCL, NuSMV)

Studien-/Prüfungsleistungen:

Geschichte und Prinzipien der Programmiersprachen (Prüfungsnummer: 128686)
(englische Bezeichnung: History and Principles of Programming Languages)

Prüfungsleistung, mündliche Prüfung, Dauer (in Minuten): 30
Anteil an der Berechnung der Modulnote: 100%

Erstablegung: WS 2014/2015, 1. Wdh.: SS 2015
1. Prüfer: Hans Jürgen Schneider
