Principles of Programming Languages

Dr. Herbert Prähofer
Institute for System Software
Johannes Kepler University Linz

Lecturer

Computer Science
SYSTEM SOFTWARE

a.Univ.-Prof. Dipl.-Ing. Dr. Herbert Prähofer

Address: Altenberger Straße 69, 4040 Linz, Austria
Building: Computer Science Building (Science Park 3) Room: 205
Phone: + 43-732-2468-4352
Fax: + 43-732-2468-4345
Email: **herbert.praehofer@jku.at**

Research and teaching background

- programming languages
- functional programming
- static and dynamic software analysis
- software development and engineering methods in the automation domain

Course Description

- Mandatory course in Major Subject **Software Engineering**
  - 2 KV, 3 ETCS
  - Combined course (KV) with **theoretical** and **practical** part

Software Engineering				
Formal Methods in Software Development	3KV	Schreiner	4.5	WS
Requirements Engineering	2KV	Grünbacher	3.0	WS
Principles of Programming Languages	2KV	Prähofer	3.0	WS
System Software	2KV	Mössenböck	3.0	WS
Software Architectures	3KV	Weinreich	4.5	SS
Model-driven Engineering	2KV	Fischer	3.0	SS
Software Testing	2KV	Plösch, Ramler	3.0	SS
Software Processes and Tools	2KV	Grünbacher	3.0	SS
Project in Software Engineering	5PR		7.5	WS/SS
Seminar in Software Engineering: ...	2SE		3.0	WS/SS

Dates and Lecture Rooms

Datum	Uhrzeit	Raum	Thema
Do. 06.10.2022	15:30 - 17:00	HS 19	
Do. 13.10.2022	15:30 - 17:00	HS 19	
Do. 20.10.2022	15:30 - 17:00	HS 19	
Do. 27.10.2022	15:30 - 17:00	HS 19	
Do. 03.11.2022	15:30 - 17:00	HS 19	
Do. 10.11.2022	15:30 - 17:00	HS 19	
Do. 17.11.2022	15:30 - 17:00	HS 19	
Do. 24.11.2022	15:30 - 17:00	HS 19	
Do. 01.12.2022	15:30 - 17:00	HS 19	
Do. 15.12.2022	15:30 - 17:00	HS 19	
Do. 12.01.2023	15:30 - 17:00	HS 19	
Do. 19.01.2023	15:30 - 17:00	HS 19	
Do. 26.01.2023	15:30 - 17:00	HS 19	Klausur
Do. 26.01.2023	15:30 - 17:00	HS 16	Klausur

Course Material in Moodle

JKU MOODLE Quicklinks Kalender Deutsch (de)

339.019, KV Principles of Programming Languages, Herbert Prähofer, 2022W

Dashboard / Meine Kurse / 2022W339019

Navigation

- Dashboard
  - Startseite
  - Website
  - Meine Kurse
    - MoodleInfo
    - 2020S_sok_Kurzvideos
    - 2022S353003/4/12/34/35
    - 2020W353039
    - **2022W339019**
      - Teilnehmer/innen
      - Bewertungen
      - Informations
      - Slides
      - Assignments
    - 2022W339378
    - 2022W339360
    - sok_MOOC SW 1
    - 2022S339352
    - 2022S339377
    - Mehr ...
  - Kurse

Informations

Course start

we start on Thursday, Oct 6, 15:30 in lecture room HS 19

Slides

Slides will be provided just before the lectures

Assignments

Will be provided just before each lecture

Goals of this Course

- Understand different programming paradigms
- Learn about foundations and key concepts of programming languages
- See the historical development of languages
- Get to know different types of languages
- Experience programming in different types of languages

Non-Goals

- A comprehensive coverage of the language landscape
- Details of particular languages
- Syntax issues
- Compilation and implementation issues

Course Characteristics

- Covering main programming paradigms
  - imperative
  - object-oriented
  - functional
  - logical and rule-based

➔ with a special focus on functional programming

Functional programming paradigm
- is **less known**
- is **different**
- is **powerful**
- is **highly relevant**

- Interplay of
  - Concepts and foundations
  - Concrete programming languages
  - Accompanying exercises

Programming Languages

Scala 3

- Main language in course

Lisp / Scheme

Haskell

Rust

Java

Kotlin

C#

JavaScript

... and others ...

Why Scala

- most advanced language existing
- pure object-oriented language
- full support of functional programming paradigm

Syllabus [1/2]

Part I: Introduction

I.1 Introduction and overview

I.2 Lambda Calculus

Part II: Programming language models and sample languages

II.1 Functional model incl. introduction to Lisp and Haskell

II.2 Imperative/procedural model incl. development history

II.3 Object-oriented model incl. introduction to Scala

Part III: Data types and type systems

III.1 Data types and type systems of imperative languages

III.2 Data types and type systems of object-oriented languages

III.3 Data types and type systems of functional languages

III.4 Generic types

III.5 Type extensions

Syllabus [2/2]

Part IV: Further topics

IV.1 Lambda expressions and higher-order functions

IV.2 Non-strict execution semantics

IV.3 Dynamic languages (guest lecture on JavaScript)

IV.4 ...

Part V: Logic and rule-based programming [optional]

see courses
Artificial Intelligence
and
Knwolege-based Systems

Exercises

- Various assignments
  - in languages
    - Haskell
    - Scala

- Submission through Moodle
  - 80% submissions required

- Assignments will be checked for completeness
  - fully solved (100%) – largely solved (75%) – partially solved (50%) – not solved (0%)

- Individual feedback on your solutions on demand
  - after the lecture
  - consultation hour Zoom meeting (usually Monday 5:00 p.m.)

Exam and Grading

Exam

- Written exam at the end of the semester

Grading

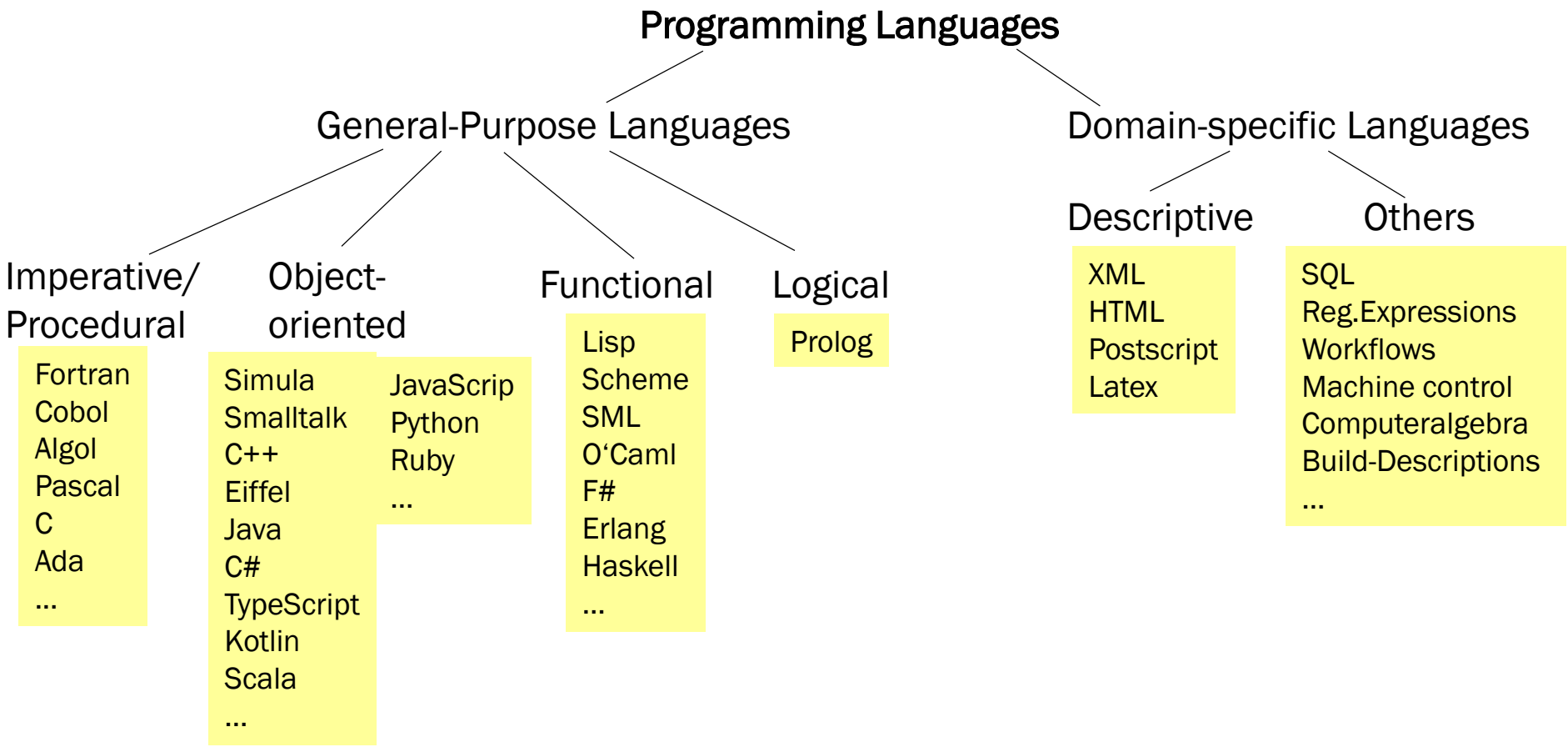
- 2/3 written exam
- 1/3 assignments: number of submissions and quality
  - review of solutions if necessary
- both must be positive
  - >= 50% of written exam
  - 80% submissions
  - at least 50% solved

Principles of Programming Languages

I.1 Introduction

Dr. Herbert Prähofer
Institute for System Software
Johannes Kepler University Linz

Programming Language Paradigms

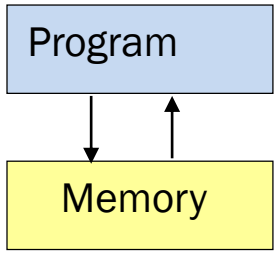


- Many languages today cover object-oriented, imperative and functional features
  - C#, Java, Scala, Kotlin, JavaScript, TypeScript …

Thinking Models

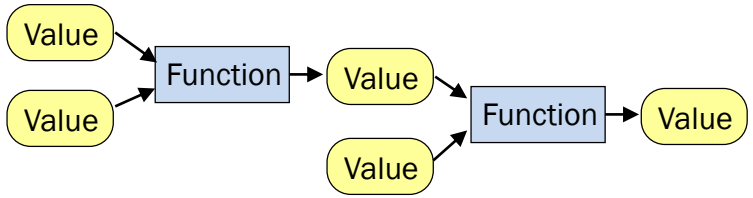
Imperative

- Variables and assignments
- Statements and state transitions
- Control structures and procedures



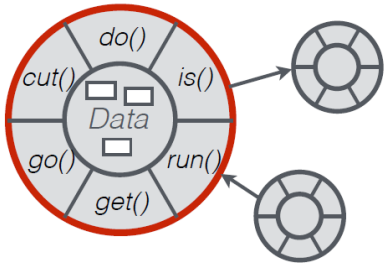
Functional

- Function application
- Expressions
- Immuatble data
- Code as data



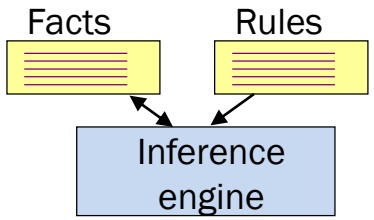
Object-oriented

- Objects = Data + Behavior
- Methods and message passing
- Typ hierarchies and inheritance
- Dynamic binding

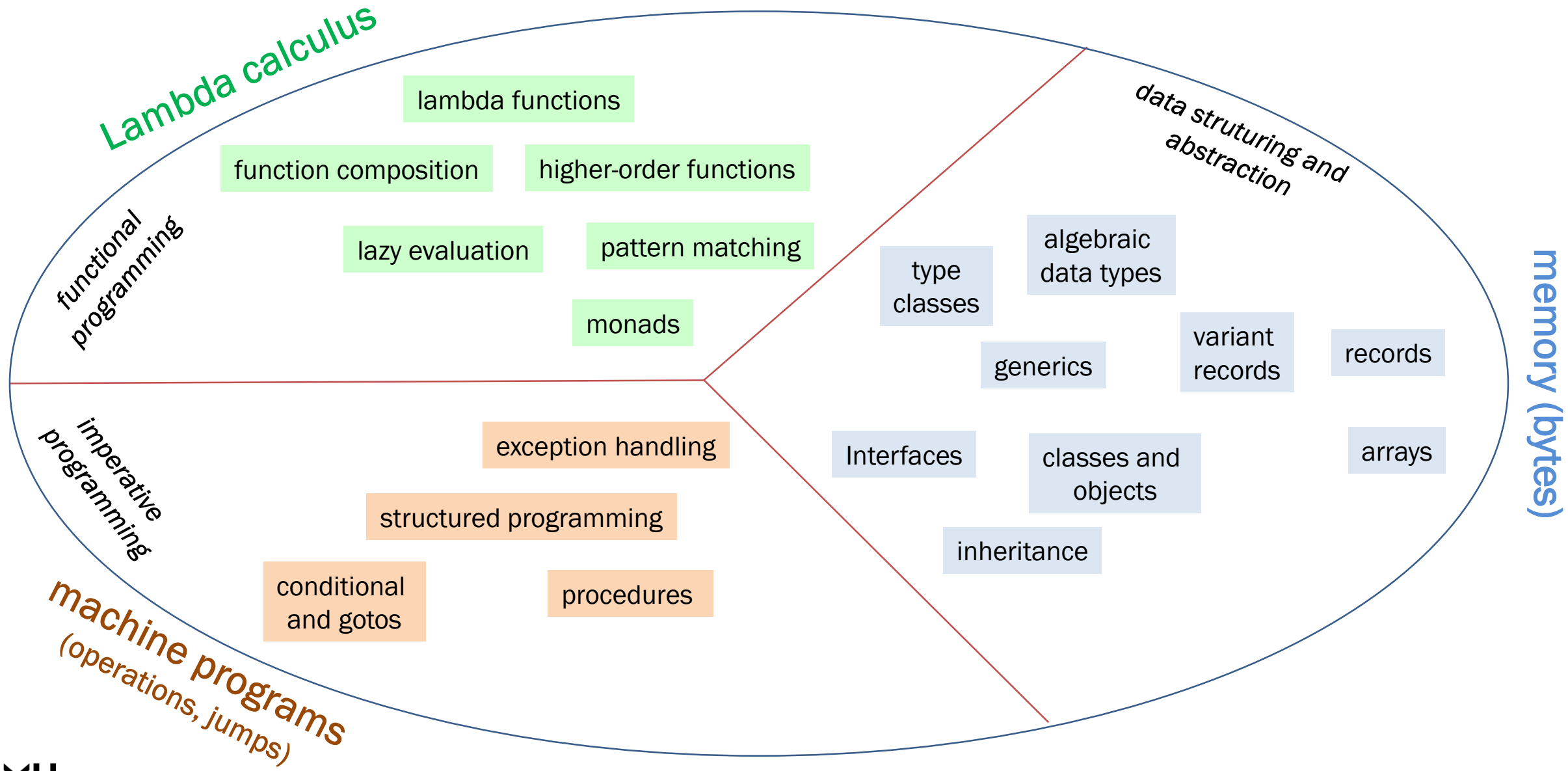


Logical

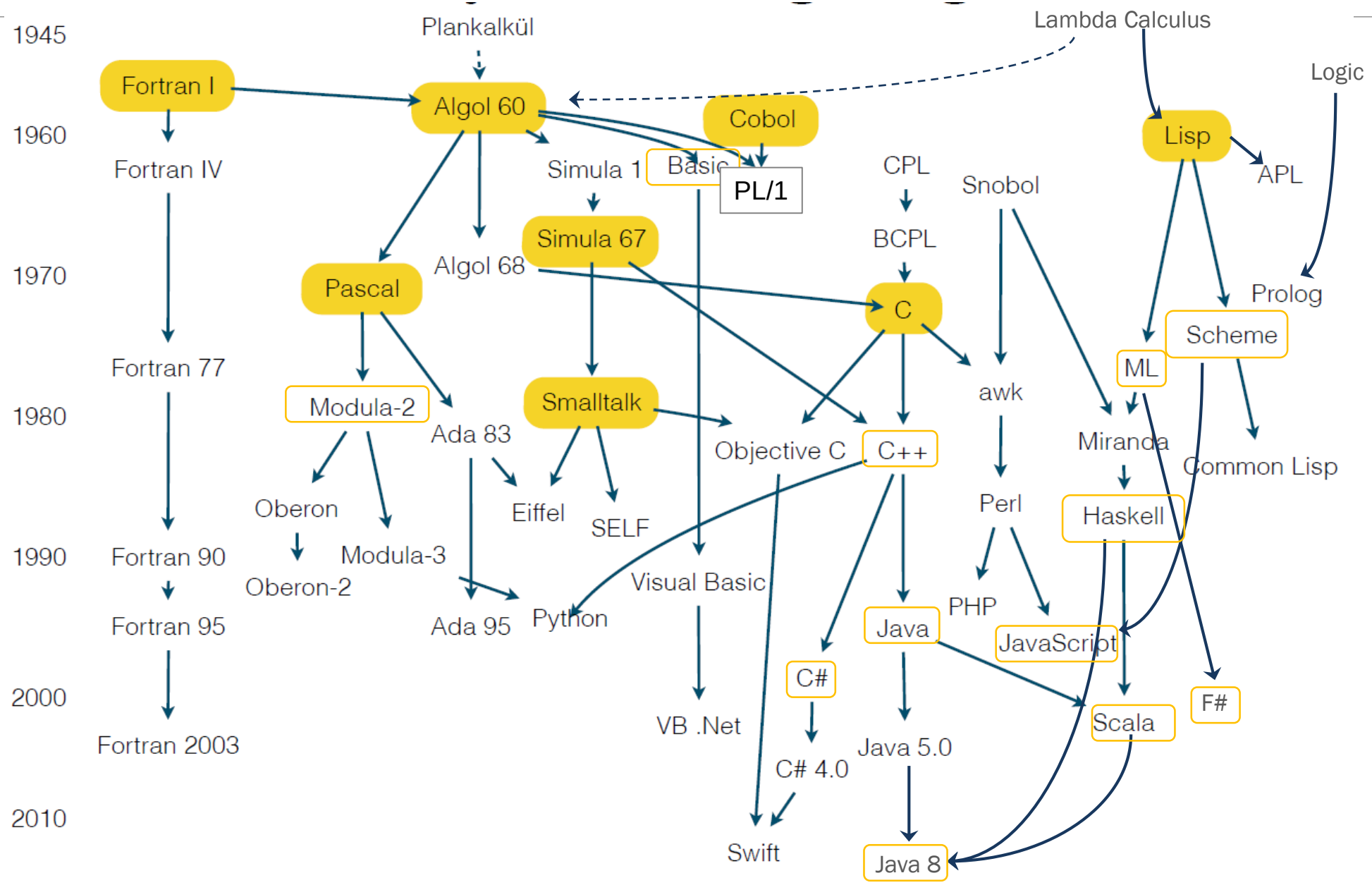
- Problem description
- Logical inference



Programming Language Concepts



History of Programming Languages



Categories and Terms

Typing

- Statically typed ➔ types are checked at compile time
- Dynamically typed ➔ types are checked at run time

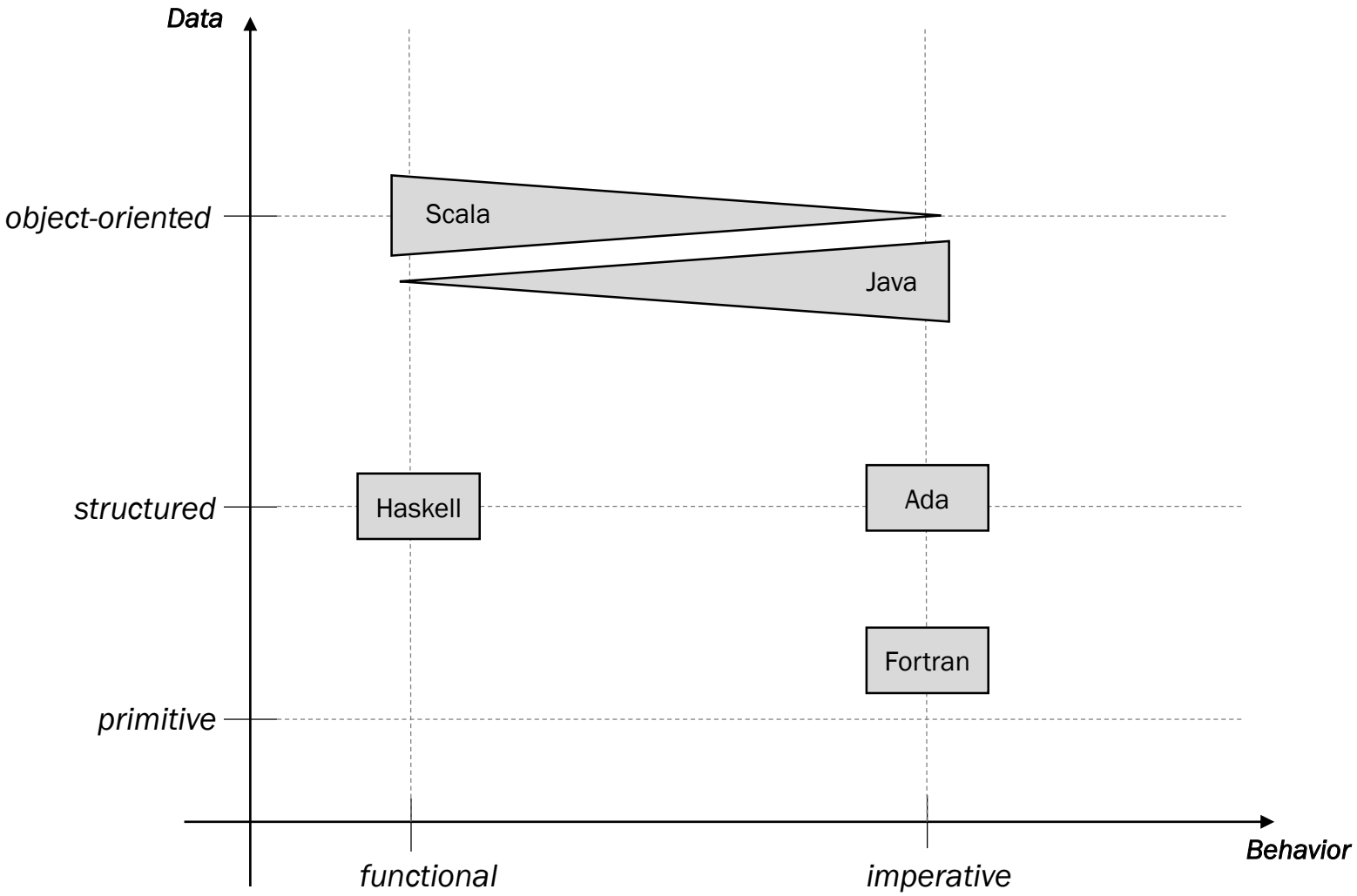
Execution

- Compiled
- Interpreted
- ➔ Combinations, e.g. Java, JavaScript

Memory management

- Managed ➔ VM with garbage collection
- Unmanaged

Classification along different Dimensions



Classification along different Dimensions