Syllabus for UCLA Computer Science 131

[131 home]

Texts

Required text

- Adam Brooks Webber, *Modern Programming Languages: A Practical Introduction, 2nd edition*, Franklin, Beedle & Associates, ISBN 978-1-59028-250-2 (2011). Errata [PDF] are available.

Topics

Language design issues

- efficiency, safety, convenience
- programming categories
  - procedural
  - functional
  - object-oriented
  - declarative
  - scripting
- scaling problems
- history and evolution of programming languages

functions

- motivation
- recursion and tail recursion
- activation records
- functions as data
- closures
- debugging via divide and conquer
- persistence

- amortized efficiency

syntax

- tokenization
- grammars

names

- names, binding, visibility, scope, lifetime
- static vs dynamic scope

types

- type, values, operations
- type checking and conversion
- elementary and structured types
- explicit vs implicit storage management
- stack vs heap

control

- expression evaluation
- rewrite rules
- pattern matching, unification, and backtracking
- structured programs

objects

- object-oriented design
- encapsulation and data abstraction
- separating behavior from implementation
- classes and class hierarchies
- inheritance
- polymorphism
- collections and iteration

exceptions

- design issues

- when not to use exceptions
- case study: exceptions in Java

concurrency

- competition and cooperation
- synchronization
- case study: multithreading in Java

semantics

- lambda calculus
- program verification

Language paradigms

Java

- primitive and reference types
- classes and instances
- variables, methods, constructors, and overloading
- inheritance, abstract classes, final classes, and interfaces
- compilation units, packages and visibility
- the `Object` class
- Java class library basics
- collections
- exceptions
- threads

OCaml

- type inference and annotations
- pattern matching
- polymorphism
- higher-order functions and currying
- type constructors

Prolog

- propositional logic
- predicate calculus: instantiation, atoms, variables, structures
- clausal form and Horn clauses
- the resolution principle
- depth-first backward chaining with backtracking
- debugging
- memory management
- the closed world assumption

Python

- the Python shell
- scripting
- functional and object-oriented programming
- modules and packages
- Python library basics
- extending and embedding

Scheme

- syntax
- lists
- comparison (e.g., `eq?` vs `equal?`)
- syntactic forms: core and extension
- `let` and `lambda`
- tail recursion
- continuations

$Id: syllabus.html,v 1.21 2019/09/25 00:54:08 eggert Exp $