

---

## Summary

- Looked at important programming languages concepts and their implementation, and how they are realized in specific programming languages.
- Important concepts:
  - What is a language? Separation of syntactic and semantic aspects of languages.
  - Variables, bindings, and expressions: order of evaluation of expressions.
  - Notion of types, type equivalence, and type checking, and why they are needed.
  - Software development concepts such as information hiding, generalization and specialization of information, and genericity, and how they are supported in languages through ADT, class, inheritance, and template mechanisms.
  - Scoping
  - Implementation Issues:
    - Scanning, parsing, code generation
    - Storage layout: static, stack, heap
  - Separate models of computations: functional and logic based approaches.

---

## Summary - cont'd.

- Languages:
  1. Java
    - Programming environment
    - Notion of class, inheritance
    - Inheritance
  2. LISP
    - Elegant
    - Uniform view: both code and data are lists. Also model of computation is slightly different.
  3. Prolog:
    - Elegant
    - Logic based approach, relational programming
    - backtracking, unification, “input”/“output” parameters
- Use of languages:
  - Java: becoming industry standard.
  - LISP and PROLOG: big in AI, but also being used for more general purpose computing. Interpretive style of software development for imperative languages as well. Prolog has influence design of many concurrent programming languages.

---

## Summary - cont'd.

- Programming assignments: Goal was to learn more than just learn a language – do something conceptual with it.
  - C++: Syntax, parsing
  - Java: Inheritance, Data abstraction, Iterators, Public and Private inheritance.
  - LISP: Program transformation and simplification
  - Prolog: Backtracking and searching
- What I hope you got out of the course:
  - Broaden your view of programming languages and programming
  - More flexible and better programmer
  - Note: did not become an expert in any of the languages, at least not just through the course.
  - easier to pick up new languages.
  - Appreciation of kinds of languages and paradigms, plus inter-relationships.
  - Some ideas about process of language design and how various issues influence them.
  - Some ideas about language design.

---

## Where do we go from here?

- ECS140B: more languages, currently Icon, SR, Sisal; imperative string processing, with backtracking; concurrent programming, functional scientific computing. Prof Olsson.
- ECS142 compilers: Learn lots more about implementation and optimization techniques, e.g. details of parsing, code generation, optimization data-flow analysis etc.
- ECS240: graduate course in programming languages, focusing on formal semantics.
- ECS244: concurrent programming.
- ECS247: Design and implementation of concurrent programs.