Programming Languages – Summary

Purposes

- Instructing machines.
- Communicating between programmers.
- Expressing high-level designs.
- Notation for algorithms.
- Tool for experimentation.
- Preventing human errors.

Programming Style

Procedural: Program consists of procedures that are invoked to change the state of the machine.

Object Oriented: Program consists of objects. Methods on objects are invoked to change the state of that object.

Functional: Program consists of functions that return results. These results are then used for future calculations. The function does not otherwise alter the state of the machine.

Language constructs encourage different styles.

Language Families

Imperative: Characterized by procedures that change the state of the system. I.e. have *side-effects*.
- *Procedural*.
- *Object Oriented*.

Declarative: Describe “what” is wanted, not “how” it is done.
- *Functional*.
- *Logic/Relational*.

Importance

There are many tradeoffs in selecting/creating a language:

- *reusability* - higher order functions, currying, inheritance
- *maintainability* - packages, abstract data types, information hiding.
- *portability* - interpreted, compiled to bytecodes.
- *performance* - hybrid Java class model, tail-recursion.
- *complexity* - Scheme vs. Java.
- *expressiveness* - source-to-source transforms, Smalltalk blocks.
- *libraries* - Java class hierarchy, etc.
- *debugging ease* - exceptions, static typing.

Concepts

- garbage collection
- parameter passing
- static vs. dynamic
- side-effects
- persistence
- special forms
- recursion
- procedural abstraction
- higher order functions
- scoping
- type systems
- object oriented
- overriding
- overloading
- polymorphism
- inheritance
- “first-class”
- exceptions
- threads
- pattern matching
- lazy evaluation
- type inference
- constraints
- constraint solvers

Scheme

- Allows side-effects, but encourages a functional style.
- Simple syntax, expression oriented.
- Extensively uses lists (Lisp = LISt Processing).
- Heavy use of recursion.
- Dynamically typed.
- Interpreted (or compiled).
- Read-eval-print loop.

Java

- Object oriented.
- Inheritance.
- Dynamic dispatch.
- Interfaces for polymorphism.
- Statically & dynamically typed, type-safe.
- Extensive class library.
- Threads.
- Interpreted byte-codes.

Miranda

- Pure functional.
- Lazy evaluation.
- Pattern Matching.
- Heavy use of lists and recursion.
- Statically typed: type inference.
- Infinite data structures.

CLP($\mathcal{R}$)

- Declarative.
- Pass by constraint.
- Relations work in both directions.
- Relations combine to restrict solutions.
- Constraint solver searches solution space for solution.
- Can solve linear systems of equations.