Leftover from Names Scopes and Bindings
ObjectOrientedLanguages
  Inheritance
  Dynamic method Dispatch
  see DMD_java

Topic 5 — Control Flow Ch 6 Scott
sequencing
selection
iteration
procedures
recursion
concurrency
exception handling
nondeterminacy (skip)

“sequencing is central to imperative languages but plays a relatively minor role in functional”
Question, what order are args to a function evaluated?
  Go: left to right at least usually. see order_go
  Book notes that optimizing compilers might want to change order. So, even if
you test — like order_go — probably best to not rely.
in infix notation, what order are operators evaluated (note no precedence problems in pre/postfix)
  C: On the order of 15 levels of precedence — about that many in Java and Go
also need to know “associativity”
    10-5-5 is this 10 or 0??
    17+MAX_INT-50
      if r->l then MAX_INT-33 else
In a program I wrote with math ALL I had was one level of precedence and L-R
associativity
  Good idea?? Smalltalk, APL do it

“side effect” if expression evaluation influences subsequent computation in any way other than
by returning a value
  NOTE—by this definition printing is not a side effect
  if there are no side effects that a lot of sequencing is not important.
ALL assignment statements are done for side effect. If you did not care about the value
being stored, do not store it.
  purely functional languages have NO side effects.

Value model of variables
  a = b+c
  l-value — the location of the variable. Where a thing is to be stored. (a) Think of L as
standing for both Left and Location
usually l-value is simple thing but it can get complex due to arrays a[b[5]+3] and structs/objects
  r-value — an actual value (b,b,b+c)

reference model of variables
  still have l-value and r-values
  every variable is an l value so when an r-value is needed need to “dereference” — that
is turn it into an r-value.

What model does Go use? Java? design an experiment to determine
Go almost always uses value model — exception data structures like slices
A slice is a reference. A sub slice is a reference to a location within a slice!

Value model languages DO NOT have aliases. They can’t. But even in value-model langs, references and aliases can be really handy. So they have pointers! Reference-model does not need because everything is a pointer.

Go: need to tell that you have a pointer when passing into funcs, but after that value and pointer are treated the same — from programmers perspective.

Value model languages DO NOT have aliases. They can’t. But even in value-model langs, references and aliases can be really handy. So they have pointers! Reference-model does not need because everything is a pointer.

Go: need to tell that you have a pointer when passing into funcs, but after that value and pointer are treated the same — from programmers perspective.

see pointer0_go — usign pointer to get an alias
pointer_go — using pointers in function calls
nullpointer_go — even pointers in Go have a “zero value”

Initialization and the problem of uninitialized vars
Not that value and reference model langs have different issues
Java: every value model starts as 0. Every reference starts as null
Go: every type has a defined “zero” state. Every var initialized to zero state. pointers?
Java — definite assignment guarantees that no variable is uninitialized. Unneeded in
Go. Is it really needed in Java?
see Definite.java in short_go

Short circuit boolean
see short_go
Note unlike Java Go does not allow assignment with boolean
also a = (b=6)

Flow
Goto BAD
why a label in a program is not GoTo
return — should it be allow from anywhere or only at end!
should I be able to return from more than one func.
very rare
Crossover between exception and multilevel return. Note that can simulate a
multilevel return in Java using exceptions — Write Example

SEQUENCING
“sequencing is central to imperative programming”

SELECTION
the if statement
the switch statement
switch_go and switch.go, Switch.java therein
“the principle motivation is to facilitate the generation of efficient target code”
switch in Go
NO fall through
but allows listing of multiple cases
any type that allows an == comparison
tagless optionall

LOOPS
another imperative concept
Iterators and enumeration controlled loops
rather than just using numbers allows programmer to do a loop for everything in a collection

Have seen this in GO slic_go/slic.go
for idx,val := range slice {}
Java: looping over array or any collection
Labeled loops and break:
  break allow you to get out of loop early
  labeled loop allows you to not just get out of inner loop
    GO: break cannot get out of current function — WHY?
      break_go
Using closures to simulate enumerated loop
  see cloloop_go
    problem, this is specific to a int slice. Would really like generics.
    but Go does not have them — not yet!
Any number of syntaxes for for loops

Recursion
  Advantage: no special syntax (but does require support for recursion)
  Tail recursion:
    see tailrec_go
    “additional calculation never follows recursive call”
    In this case you can do the recursion without adding to the stack. Just use/
    overwrite the stack. Since new stack frame is one of the principle costs of recursion...