Topic 10
Subroutines
Ch 9 in Scott

return value — function
no return — procedure
associated with structure/class — method (regardless of return)

Skip 9.1 & 9.2 Mostly.
Quickly, What happens when you make a subroutine call

1. Compute (as needed) value of arguments to called subroutine
1a. Hold onto the "dynamic link" ie the place where the program execution returns to when function completes
2. if a nested subroutine — the "static link" ie what variables are available to nested sub from its surrounding subroutine

see funinfun_go/ff.go

Java does not support nested subroutines. But you can make an instance of a class with a function defined at that time!

```
public class Nested {
   private abstract class Aa {
      public abstract void printer();
   }
   public void outer(int zz) {
      int bb = zz * zz;
      Aa aA = new Aa() {
         public void printer() {
            System.out.println(bb);
         }
      };
      aA.printer();
      bb *= bb;
      aA.printer();
   }
}
```

This does not compile!!! Problem is with the line "bb *= bb". Comment is the bb must be final, or effectively final. Discuss

see

how is static link distinct from closure? Closures are about functions that are defined, but then executed later. Static link is about functions executed now. So they refer to largely the same thing; but with different uses. Also closures can get outside the function in which they were defined.

the callee — on starting:

3. Pass return location of the calling function
4. Add all of this to the "call stack"

Parameter passing: (sec 9.3)

Define

/ reference param — a param that is a pointer
\ value param — a param that is a value
/ formal param — the thing in the function definition
\ actual param — the value in the function call

Two basics

pass by value
pass by reference
These generally make the most sense in discussion of value-model languages because they determine what gets sent in a function call.
call by sharing
applies to a reference model language, essentially “share” the reference with the called function.

Java — primitive types == pass by value
classes == sharing
Go — values — pass by value
pointers — pass by value
Can change the thing pointed to but if try to change the passed pointer, that change does not survive!
So the pointer itself is passed by value!!
see passpoint_go/
Rust
pass ownership except
1. item implements the “copy trait”
2. reference — NON mut see **passref_rust — rec_refr_mut_value**

3. Reference — mut see **passref_rust — rec_refr_mut_reference**
4. Reference to struct — if try to replace reference with struct created in the method “// compile error : `bb` does not live long enough
// `bb` dropped here while still borrowed
“ WHAT????

```
fn rec_refr_mut_reference_s(mut aa : &aa) {
   println!("Before  int {:?}", aa);
   let mut bb = aa{aval:42};  //not allowed when behind a &
   aa = &bb; // compile error : `bb` does not live long enough
         // `bb` dropped here while still borrowed
   //aa = &12; // cannot mutate aa
   println!("After  int {:?}", aa);
}
```

problem is that the struct is allocated from the stack and thus when the function ends, the struct goes away. As a result the pointer would become null. But Rust hates this and will not allow (by default). Leads to one of the truly unique things in rust, reasoning about object lifetimes. We will NOT discuss this further — one of the things this does is get out of this trap

Why doesn’t Go have this issue?? Structs are allocated from the heap!!!!

Java — “call by sharing”
see passref_java/PassRef.java
Note that in Java, when you cannot change the location of the pointer, you can change what is inside
When pass (or share) a reference
effectively allows function to return multiple values
mret_go/mreg.go
Why bother?
PbV on large objects can be expensive

Read-only parameters
Java formal parameter marked as final
covariance
Go No
Are read-only params needed?

Closures — again
when functions are passed in Go they are passed with their closure (if closure exists) (when will it not?)
Question what happens when change values in closure AFTER function

Java — functions are NOT first class so cannot be passed so closures are not a thing
Instead “Object closures”. Idea, define an object with method and vars. Then pass the objects. The receiver knows to use the method defined within the object. This is used a lot in Android to get “callbacks” and more generally in multithreadedprograms.
Illustrative example of callback?
For instance, downloading stuff from web. By rule in Android, this must be done in a separate thread. When download complete, want to notify the “main thread” and have it put up a “complete/fail” message. So, pass in an instance of something like
Will come back to this

Rust closures: move and FnMut
Closures and mutables functions.
Closures can be passed from a function to another function
see **move_rust/src/main.rs**
But closures cannot be returned! WHY? Rust can not deal with extent! (if nothng else, violates ownership). Also, for vars allocated on the stack, rust requires them to die with their stack frame.
To return use “move” which causes the function to take ownership of the vars. Thus games with vars and closures that are possible in Go cannot happen in rust. (EG, cannot change var in returning func after the func is created.)

Optional parameters
“Go has no concept of default parameters nor any way to specify arguments by name”
Nor does Java … exactly
Why not?
Go does not allow overloading of function names
Why Not?
FROM GO DOCUMENTATION
Why does Go not support overloading of methods and operators?
Method dispatch is simplified if it doesn't need to do type matching as well. Experience with other languages told us that having a variety of methods with the same name but different signatures was occasionally useful but that it could also be confusing and fragile in practice. Matching only by name and requiring consistency in the types was a major simplifying decision in Go's type system.
Regarding operator overloading, it seems more a convenience than an absolute requirement. Again, things are simpler without it.

Many languages allow both named parameters and overloading.
Do you really need both?
Java overloading only
Variable number of args
Go — YES

the print statement
see also **variadic_go**
clever user of functions — but is this clearer??

Rust???
no overloading
“Generics and function overloading don’t play along well.” ????
fn does not do variadic, macros do

Function returns
1. Should functions allow return statement other than at the end?
Go: multiple return values; see return_go
RUST: sure, through tuple creation and unpacking, You do not really need to know that the tuple ever existed.

Go: named return values; see return_go
are these good?
Rust: only in macros
Java: not at all

Exceptions:

Text — exception Handling
3 methods of dealing with exceptions without explicit exception handling mechanism
1. Return one value with special values indicating that things went wrong. For instance, if function is supposed to return a positive integer, return -1 to indicate an error
2. Return a status in addition to value (in a multiple return language). The is the general approach taken by Go, with the Panic as a bonus
3. Caller returns a closure containing an error handling function. (or null if no error.) This is kind of Java — Object Closures!!!
see **ObjClo_java/ObjClo.java**

From booK: 1 is difficult to generalize, 2 and 3 can introduce new errors and obfuscatory ..

https://radio-weblogs.com/0122027/stories/2003/04/01/JavasCheckedExceptionsWereAMistake.html

Handling in Go
basic approach, lots of functions return two values
desired value, error
see **basicexcep_go/basicexcept.go**
when normal, err value is 0 (nil).
when problem, normal value is 0 (possibly) and error value is non-0
So, how does func set that non-nil error?
1. error prevention — do not just open a file and try to read, then get exception when file does not exist. First check that file exists. Yes, programmer has to do this.
2. PANIC. panics are the equivalent of a thrown exception in Java
if not “caught”, program dies. BUT catch without try catch???
Note: “giving up is usually the right response to a panic, but not always”
More generally, problems. Answer, catch with a function that is guaranteed to run at the end of a function kind of like try {} finally{} in Java

in Go “defer” == run just before function completes.

So defer is part of the language

inside deferred function call “recover()”. If execute while NOT in panic, return null. if return NOT null, then panicing!!!

So, how do you get a func to return an error value is the only way to catch the error val is done by a function that must run as the last thing in a function??? A By having the actual work done in an inner function!! **see except_go/except1.go .. basicPanic()**
except_go/except1.go .. betterPanic()

OR

use go named return values and set the value of the named returns inside the deferred fun see **except1/betterPanic2()**

This use case may be logic behind named returns

So why does Go take option 2?

it does not/ fully. It also kind of does option 3 —

Programmer should do stuff to prevent problems. If unable to prevent, only then, maybe, panic and die.

Problems with catching exceptions someplace other than the func in which they occurred — need to unwind the stack!!!!

3 goals of exception handlers:

A. compensate for exception to allow program to recover
B. Clean up and re-thow so someone else can hanlde
C. Print an error message.

Events

also “event-driven programming”

Point — the program might be doing other things, it is NOT just waiting on the event

events are “something to which a running program needs to responds, but which occurs outside the program, at an unpredictable time.”

GUI input

Network events (read URL)

Synchronous handlers

program is single threaded, event interrupts whatever the program had been doing; does its work, then is gone. Can be handled kind of like any subroutine call.

Threaded Handlers

most modern programming languages (except Javascript!!!!)

Problem, how to handle data structure use clashes between threads.

Java synchronized. ArrayList vs Vector. All the methods of Vector is synchronized. But, the methods of ArrayList is not synchronized.

rewrite this in Java??

For example

events_java/SD.java

Java GUI Idea is that we have a background thing that is listening for “events” and the program tells the background thing what events it is interested in. In this case 2 events

button push

window close

handling is much like the Object closure That is, define an instance of a class with a customized function

events_java/GTReder.java

Here we have our own event and handler Idea is that we want to read from the internet. But we want to keep doing stuff. So do the read in a separate thread. When complete, tell the main thread that reading is complete.