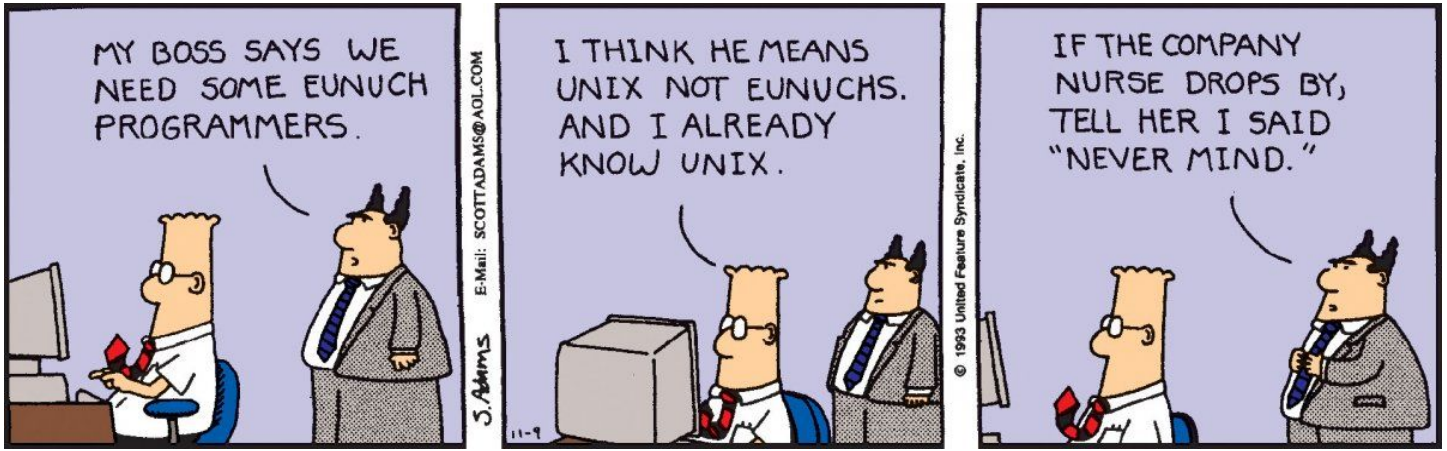
Golang for Programmers

Chuk (chukmunnlee@gmail.com)



Assumption

Workshop is not an introduction to programming

Have working knowledge of 1 programming language

Know basic git operations

Run thru some basic plus plus Golang concepts, not exhaustive

Slide https://bit.ly/golang_for_programmers

Repository https://github.com/chukmunnlee/boardgameatlas.git

What is Golang?

https://medium.com/android-news/magic-lies-here-statically-typed-vs-dynamically-typed-languages-d151c7f95e2b

What is it use for?

Infrastructure and tools

- Runtime and platforms (Kubernetes and Docker), service mesh (Consul, Istio), monitoring,logging,tracing tool (Prometheus, Loki)

Cloud native development

- RESTful web application, gRPC,

Replacement for existing infrastructure

- CoreDNS, Caddy

Data Types

The usual suspect

- string - “hello, world”
- rune - ‘h’
- number - int, int[8, 16, 32, 64], float[32, 64], uint[8, 16, 32, 64]
- complex - complex[32, 64]

```
var name string
var age uint8 = 21
pi := 3.14
```

Arrays

Array - fixed size

```
var name [5]string
name[0] = “fred”
name[1] = “barney”
name[2] = “wilma”
name[3] = “betty”

var name [5]string { “fred”, “barney”, ... }
name := [5]string { “fred”, “barney” }

len(name) = 5
cap(name) = 5
```

Slice - dynamic

```
var name = make([]int, 0, 5)
name = append(name, “fred”)
name = append(name, “barney”)

len(name) = 2
cap(name) = 5
```

Slice from array

```
slice := name[: 3]
```

Loops

```
names := []string{ “fred”, “barney”, “wilma”, “betty” }
```

For loop

```
for i := 0; i < len(names); i++ {
   fmt.Printf(“%d: %s\n”, i, names[i])
}
```

While loop

```
i := 0
for i < len(names) {
   fmt.Printf(“%d: %s\n”, i, names[i])
   i++
}
```

Iterator

```
for i, v := range names {
   fmt.Printf(“%d: %s\n”, i, v)
   i++
}
```

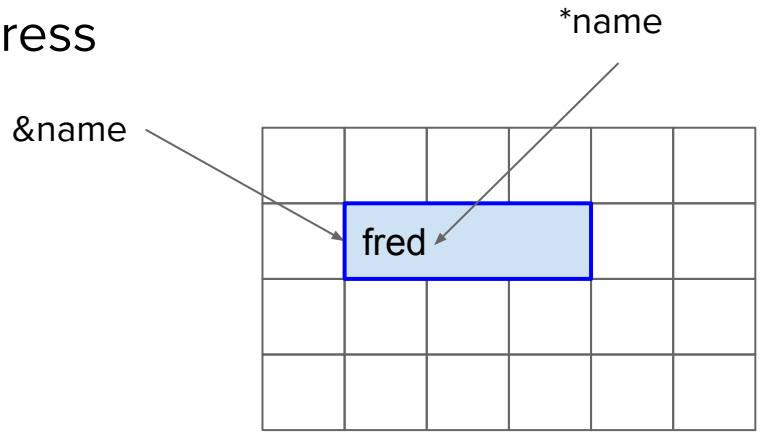
Pointers

name := “fred”

&name // address of

*&name == “fred” // contents of address

name := **new**(string)
*name = “fred”



Custom Types

```
type NaturalNumber unit32

type Customer struct {
   Id NaturalNumber
   Name string
   balance float32
}

type Apply[T any] func(T, T) T
```

Generics! Only supported in >= 1.18

May not be able to access balance
if it is outside of the module

```
cust := Customer{ Id: 5, Name: “fred”, balance: 0.0 }
```

```
func add(a int32, b int32) int32 {
   return a + b
}
func mul(a int32, b int32) int32 {
   return a * b
}

var do Apply[int32] = add
fmt.Printf(“result = %d\n”, do(10, 20))
do = mul
fmt.Printf(“result = %d\n”, do(10, 20))
```

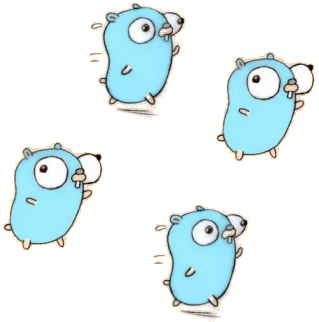
Class Like

```go
type Customer struct {
  Id NaturalNumber
  Name string
  balance float32
}

type AccountOperations interface {
  Deposit(amount float32) error
  Withdraw(amount float32) error
}
```

```go
func (c *Customer) Deposit(amount float32) error {
  c.balance += amount
  return nil
}

func (c *Customer) Withdraw(amount float32) error {
  if c.balance < amount {
    return fmt.Errorf("insufficient fund")
  }
  c.balance -= amount
  return nil
}

fred := Customer{Id: 1, Name: "fred"}
fred.Deposit(10)
```

Goroutine - Lightweight Threads

```
names := []string{ “fred”, “barney”, “wilma”, “betty” }

for _, n := range names {
   go func(name) {
      fmt.Printf(“Hello %s\n”, name)
   } (n)
}
```



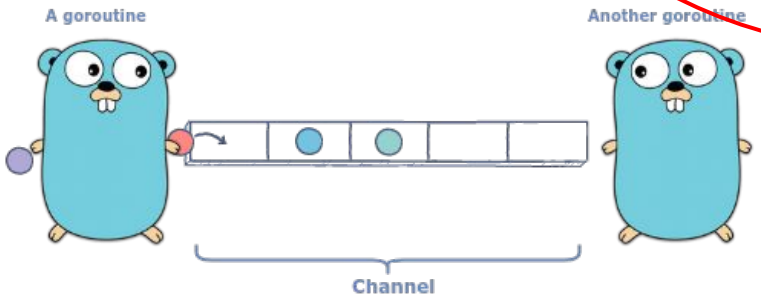
```
greet := func(name) {
   fmt.Printf(“Hello %s\n”, name)
}

for _, n := range names {
   go greet (n)
}
```

Channels - Communications Between Goroutines

```
c := make(chan int32, 1)
```

```
func print(c chan int32, fac int32) {
   result := <-c
   fmt.Printf("The result of %d! is %d", fac, result)
}
```

```
func factorial (c chan int32, fac int32) {
   result := int32(1)
   for n > 0 {
      result *= n
      n--
   }
   c<- result
}
```



Note: the main routine must keep running until all goroutines complete
Otherwise goroutines will terminate when main routine exits

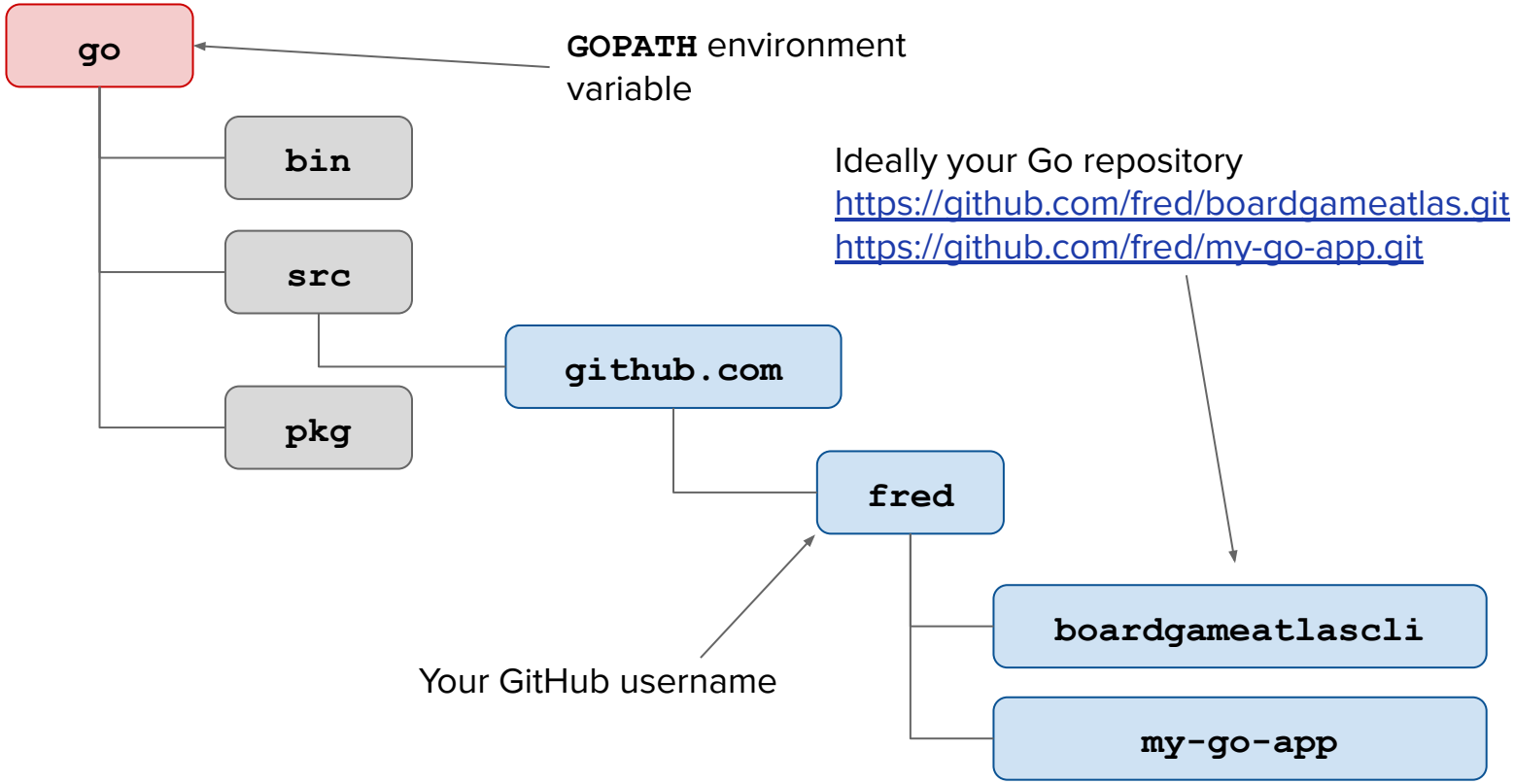
Workshop - What Will We Be Building?

Command line tool to search for boardgames
https://www.boardgameatlas.com/api/docs/search

Task

- CLI options to pass to the tool
- Learn to make HTTP GET request with query parameters
- Deserializing JSON into struct
- Make the search function reusable/integration by others
- Use context to control the HTTP request
- Pretty print the result
- Cross compile to other platforms
- Write a multi threaded version, if time permits

Golang Work Directory



Unused

Golang Environment Variables

Download from https://go.dev/dl/

GOROOT - location of your Golang SDK

GOPATH - defines the location of Go's workspace

- bin - utilities and your binaries are located here
- pkg - compiled packages
- src - go source code