Programming Language

Lucas Castro (PED) MC346 2021.2
lucas.castro@ic.unicamp.br

PED CLASSES

01 RUST LANGUAGE
09/11

02 GOLANG
11/11

03 CLOJURE
16/11

04 SCALA
18/11

GOLANG HIGHLIGHTS

NETWORK COMMUNICATION PARALLELIZATION

GO ROUTINES

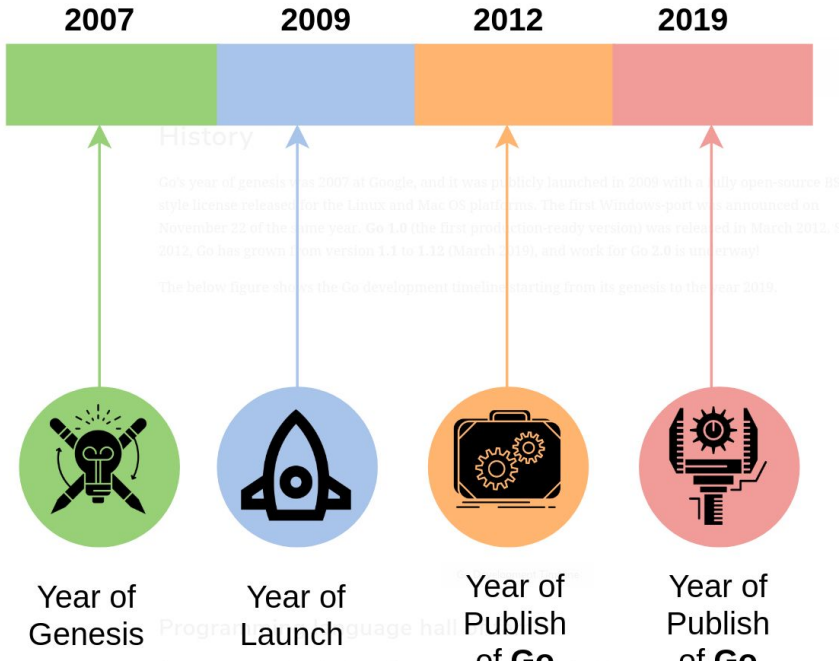
BUILDING SPEED

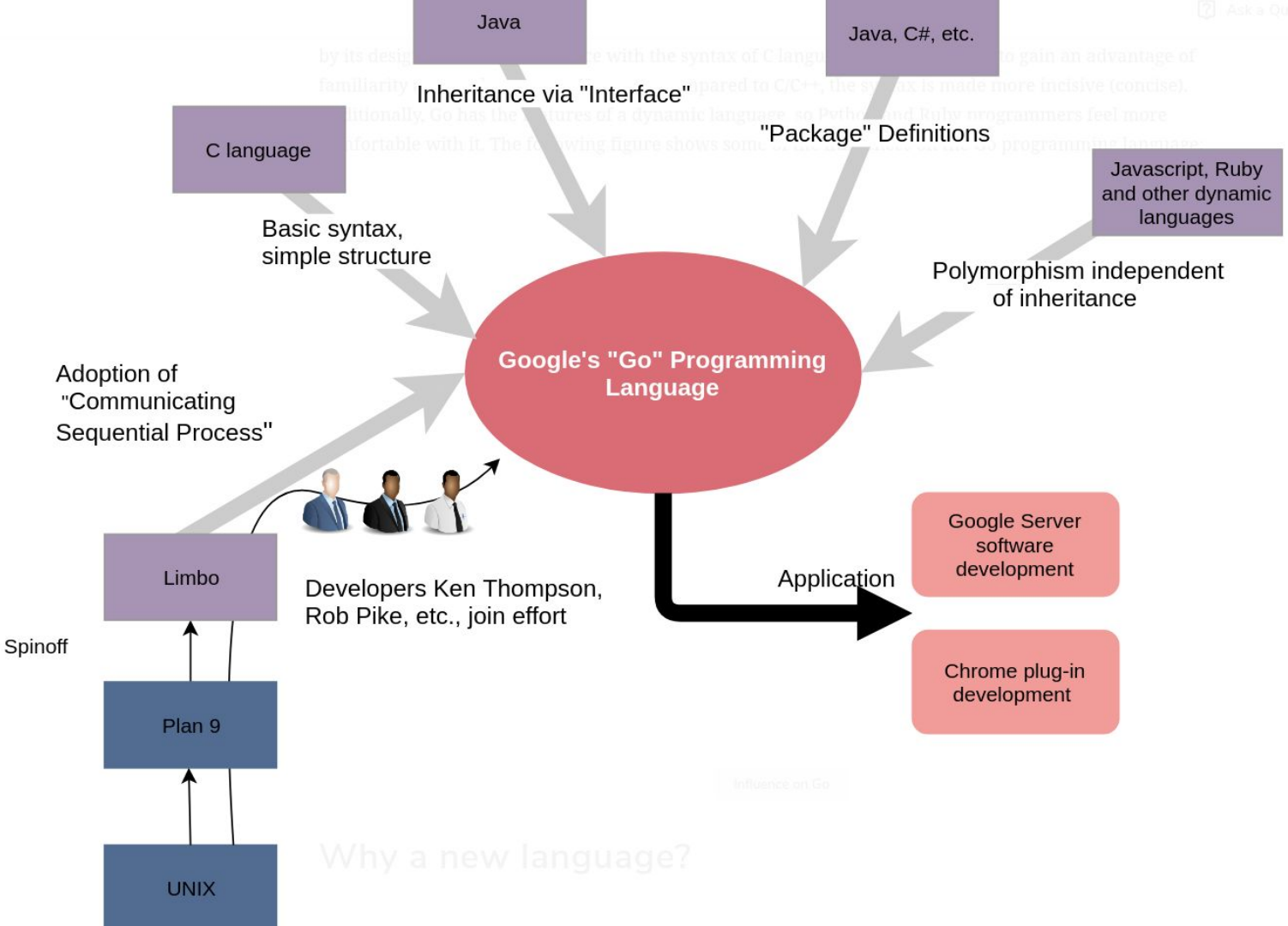
0.5” AND 20” COMPILATION TIME

MEMORY MANAGEMENT

GARBAGE C. & REFLECTION

GOLANG TIMELINE




Java
Java, C#, etc.
Inheritance via "Interface"
"Package" Definitions
C language
Javascript, Ruby and other dynamic languages
Basic syntax, simple structure
Polymorphism independent of inheritance
Google's "Go" Programming Language
Adoption of "Communicating Sequential Process"
Limbo
Developers Ken Thompson, Rob Pike, etc., join effort
Application
Google Server software development
Chrome plug-in development
Spinoff
Plan 9
UNIX

GOLANG

- Não é puramente orientada a objetos
  - Performance vs orientação a objetos

- Tipagem forte e estática
  - *Keep things explicit*

- No classes (inheritance support)
- No exceptions (recover/panic)
- Variables are mutable

- Golang é funcional, contudo com insights de outros paradigmas
  - Híbrida.

- Programação geral, sistemas e de suporte

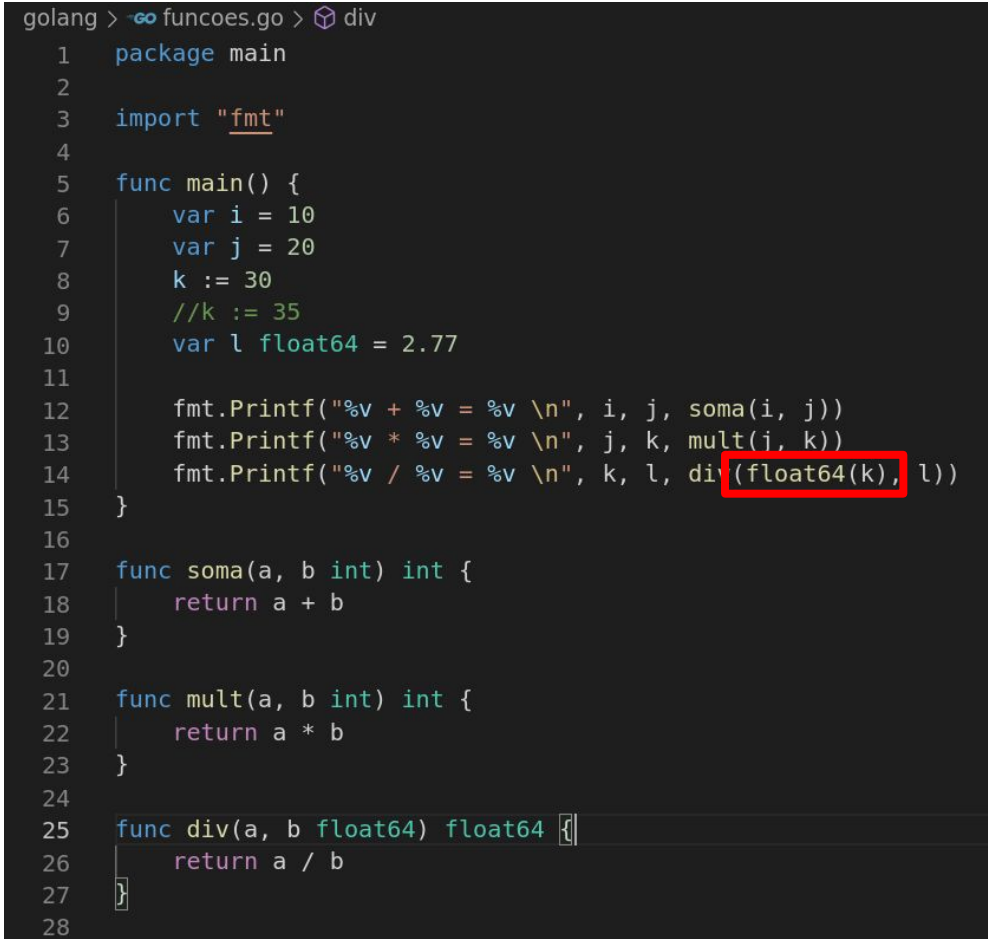
GOLANG - HELLO WORLD!

golang > helloWorld.go > ...

```
1  package main
2
3  import fm "fmt"
4
5  func main() {
6      fm.Println("Hello World!")
7  }
8
```

GOLANG - FUNÇÕES E VARIÁVEIS

- Inferência de tipos
- Pacote fmt
- Casting explícito
- Escopo das variáveis
- 3 tipos elementários principais
  - bool, numerical, character



GOLANG - TIPOS E O PACOTE "OS"

- Podemos definir diferentes tipos
- Pacote "OS", "Fmt"

golang > general.go > ...

```go
package main

import (
    "fmt"
    "os"
    "strconv"
)

type Celsius float32
type Fahrenheit float32

func main() {

    //argsWithoutProg := os.Args[1:]

    if temp, err := strconv.ParseFloat(os.Args[1], 32); err == nil {
        fmt.Printf("%f in Farenheit is %f", os.Args[1], toFahrenheit(Celsius(temp)))
    } else {
        fmt.Printf("Error")
    }

}

func toFahrenheit(t Celsius) Fahrenheit {
    var temp Fahrenheit
    temp = Fahrenheit((t * 9 / 5) + 32)
    return temp
}
```

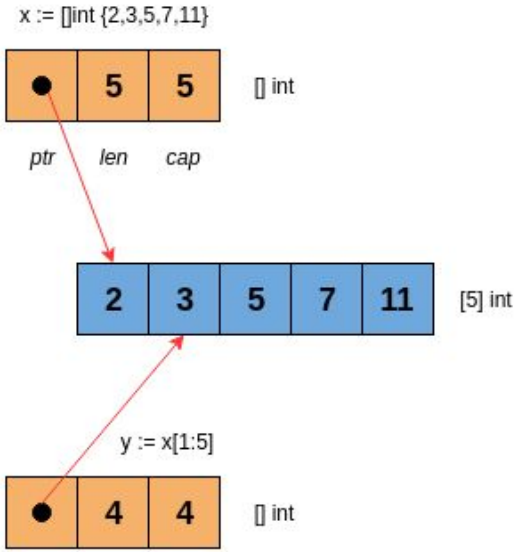
GOLANG - ARRAYS & SLICES

golang > arrays.go > toFahrenheit

```go
package main

import (
    "fmt"
    "os"
    "strconv"
)

type Fahrenheit float32

func main() {

    array := toFahrenheit os.Args[1:6])

    for i := 0; i < len(array); i++ {
        fmt.Printf("%sC is %vF\n", os.Args[i+1], array[i])
    }

}

func toFahrenheit(temp []string) [5]Fahrenheit {
    i := 0

    var tempsFarenheit [5]Fahrenheit

    for range temp {
        if tempFloat, err := strconv.ParseFloat(temp[i], 32); err == nil {
            tempsFarenheit[i] = Fahrenheit((tempFloat * 9 / 5) + 32)
        } else {
            break
        }
        i++
    }
    return tempsFarenheit

}
```

GOLANG - SLICES

- Slices
  - C/C++ Array type
  - Python list




```go
type Fahrenheit float32

func main() {
    sl := toFahrenheit(os.Args[1:len(os.Args)])
    fmt.Println("All conversions", sl)
    fmt.Println("First conversion", sl[1])
    fmt.Println("Middle conversions", sl[1:len(sl)-1])
    fmt.Println("Last conversion", sl[len(sl)-1:])

    slBkp := make([]Fahrenheit, len(sl)+10)
    copy(slBkp, sl)
    slBkp = append(slBkp, 32.55)
    fmt.Println("Slice backup", slBkp)
}

func toFahrenheit(temp []string) []Fahrenheit {
    i := 0
    tempsFarenheit := make([]Fahrenheit, len(temp))
    for range temp {
        if tempFloat, err := strconv.ParseFloat(temp[i], 32); err == nil {
            tempsFarenheit[i] = Fahrenheit((tempFloat * 9 / 5) + 32)
        } else {
            break
        }
        i++
    }
    return tempsFarenheit

}
```

GOLANG - STRUCTS &POINTERS

- “Objetos”
- Overloading? Overwriting?
- Interfaces

```
type tempStruct struct {
    celcius    float64
    farenheit float64
}

func main() {

    tempAC := new(tempStruct)
    tempAC.celcius = 20

    sl := toFahrenheit(os.Args[1:len(os.Args)]
    fmt.Println(sl)

    fmt.Println(toFahrenheitSingle(*tempAC))
}

func toFahrenheit(sliceTemp []string) []tempStruct {
    i := 0
    tempsFarenheit := make([]tempStruct, len(sliceTemp))
    for range sliceTemp {
        if tempFloat, err := strconv.ParseFloat(sliceTemp[i], 32); err == nil {
            tempsFarenheit[i].celcius = tempFloat
            tempsFarenheit[i].farenheit = ((tempFloat * 9 / 5) + 32)
        } else {
            break
        }
        i++
    }
    return tempsFarenheit

}

func toFahrenheitSingle(singleTemp tempStruct) tempStruct {
    singleTemp.farenheit = (singleTemp.celcius * 9 / 5) + 32
    return singleTemp
}
```

GOLANG - METHODS

- Métodos são diferentes que funções
- Feature que o Goland provê para uma "orientação a objetos"

```go
type tempStruct struct {
    celcius   float64
    farenheit float64
}

func (temp *tempStruct) toFahrenheit() tempStruct {
    temp.farenheit = (temp.celcius * 9 / 5) + 32
    return *temp
}

func (temp *tempStruct) toCelcius() tempStruct {
    temp.celcius = (temp.farenheit - 32) * 5 / 9
    return *temp
}

func main() {

    tempAC_BR := new(tempStruct)
    tempAC_BR.celcius = 30
    fmt.Println(tempAC_BR.toFahrenheit())

    tempAC_US := new(tempStruct)
    tempAC_US.farenheit = 0
    fmt.Println(tempAC_US.toCelcius())
}
```

GOLANG - DEFER

- Adia até a função mais prox. Retornar
- Qual vai ser o output?
-

```go
package main

import "fmt"

func mult(a, b int) int {

    res := a * b
    fmt.Println("Result: ", res)
    return 0
}

func show() {
    fmt.Println("MC346")
}

func main() {

    defer mult(23, 45)
    defer mult(10, 10)
    show()
}
```

GOLANG - GOROUTINES

- Paralelismo
- Channels

```go
package main

import (
    "fmt"
)

func display(str string) {
    for w := 0; w < 2; w++ {
        fmt.Println(str)
    }
}

func main() {

    go display("abcdefghijklmnopqrstuvxwyz123456789")
    display("MC346ICUNICAMPBR")
}
```

```go
import (
    "fmt"
    "time"
)

func display(str string) {
    for w := 0; w < 2; w++ {
        time.Sleep(1 * time.Second)
        fmt.Println(str)
    }
}

func main() {

    go display("abcdefghijklmnopqrstuvxwyz123456789")
    display("MC346ICUNICAMPBR")
}
```

GOLANG - CHANNELS

```go
func function(ch chan int) {

    fmt.Println(234 + <-ch)
}

func function2(ch chan int) {
    time.Sleep(1 * time.Second)
    ch <- 50

}
func main() {
    ch := make(chan int)

    fmt.Println("Starting...")
    go function(ch)
    go function2(ch)
    time.Sleep(2 * time.Second)
    fmt.Println("Ending")

}
```

ONDE APRENDER + SOBRE GOLANG

- Tour on Go
- Educative.io - The way to go
- Go by Examples

QUEM USA GOLANG NO BRASIL?

- Go Users Worldwide