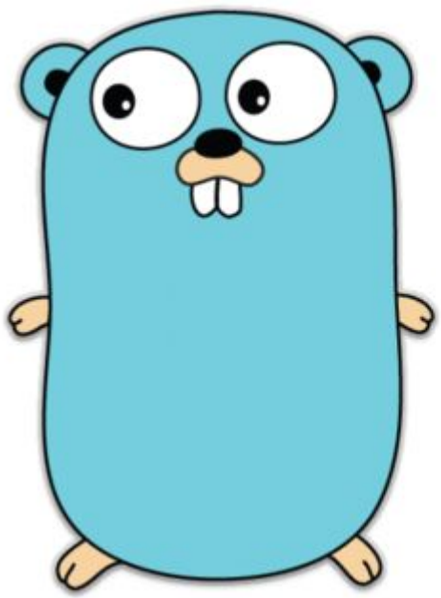
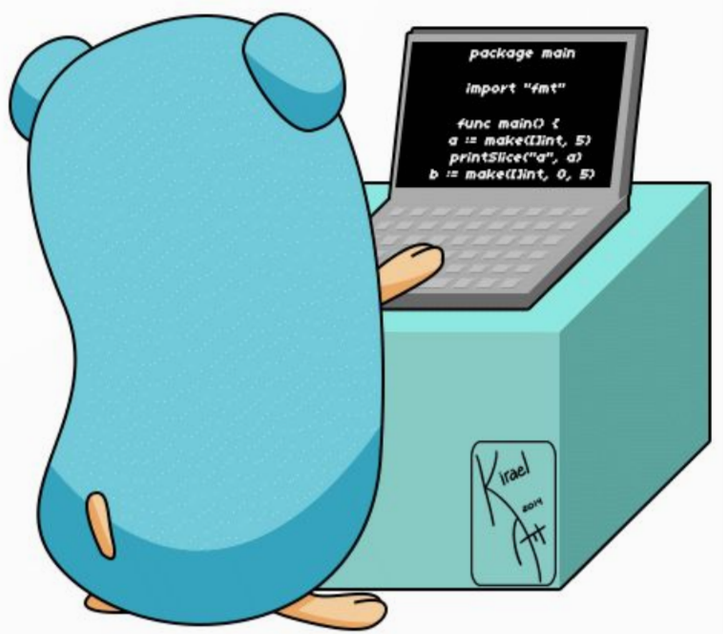


Go

¿Qué es esté Lenguaje?

- Según https://golang.org/doc/:

- “El lenguaje de programación Go es un proyecto de código abierto para que los programadores sean más productivos.
- Go es expresivo, conciso, limpio y eficiente. Sus mecanismos de concurrencia facilitan la escritura de programas que aprovechan al máximo las máquinas multi-núcleo y en red, mientras que su novedoso sistema permite la construcción de programas flexibles y modulares. Go compila rápidamente al código de máquina, pero tiene la comodidad de la recolección de basura y el poder de la reflexión en tiempo de ejecución. Es un lenguaje rápido, estático y compilado que se siente como un lenguaje interpretado de forma dinámica.”



Características

Desarrollado por Google (2009)

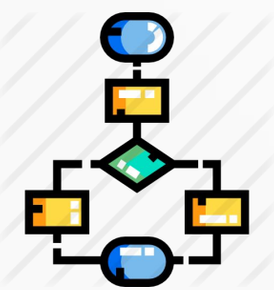
Basado en Leng. C

Compilado

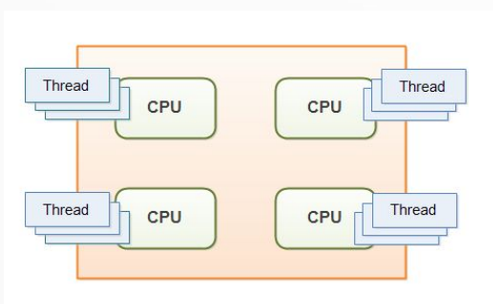
Tipado Estático

Imperativo

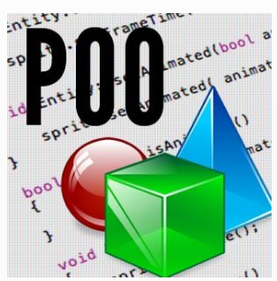
Reflexivo



Estructurado



Concurrente



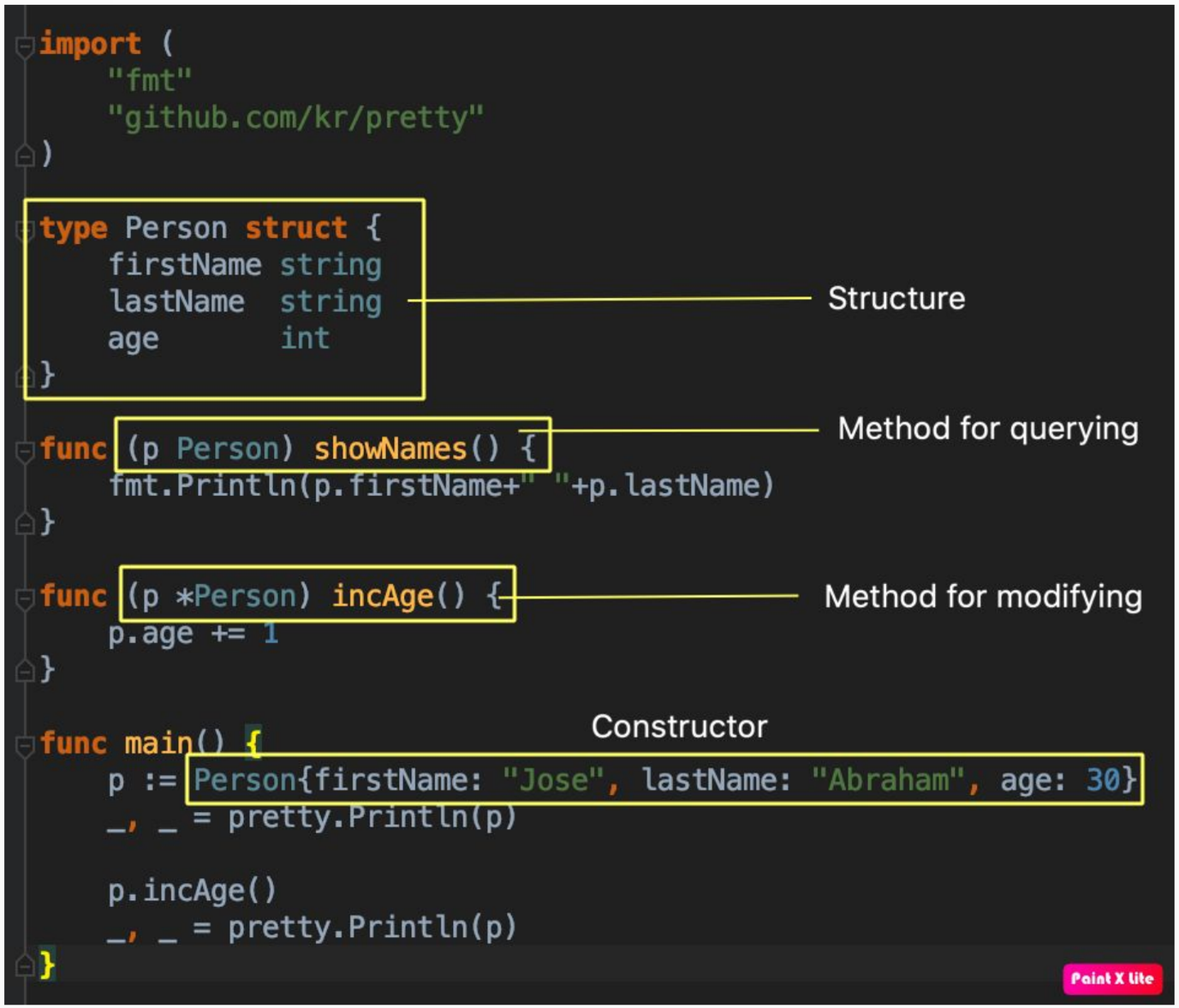
Semi Orientado a Objetos

El manejador de paquetes utiliza la filosofía de Git

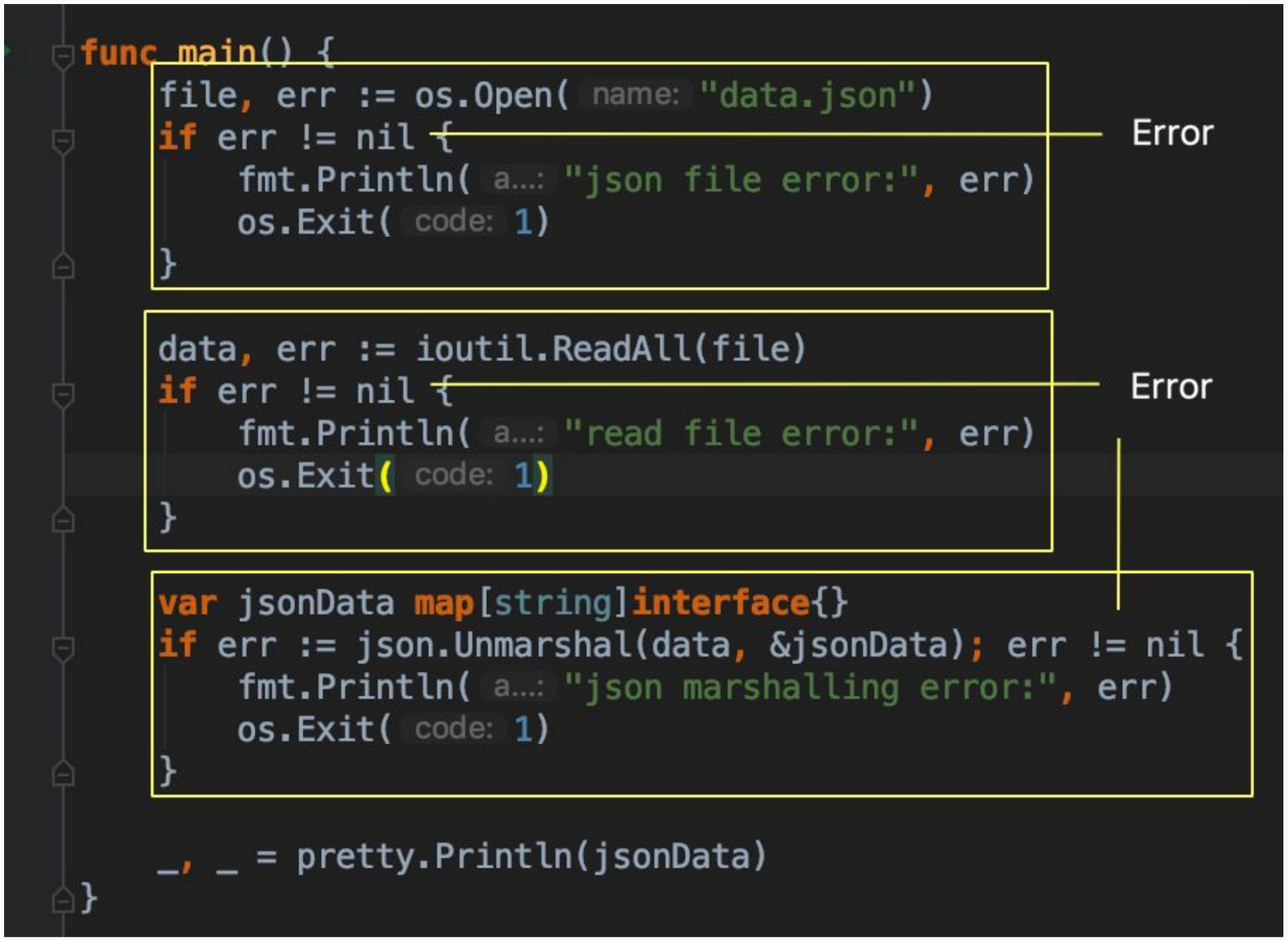
No posee Excepciones

Recolector de Basura

Estructuras

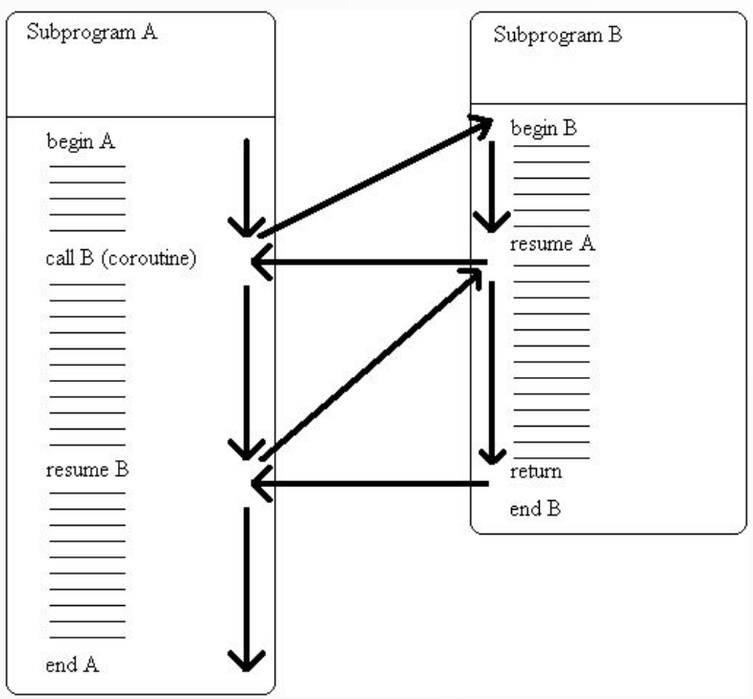


Manejo de errores

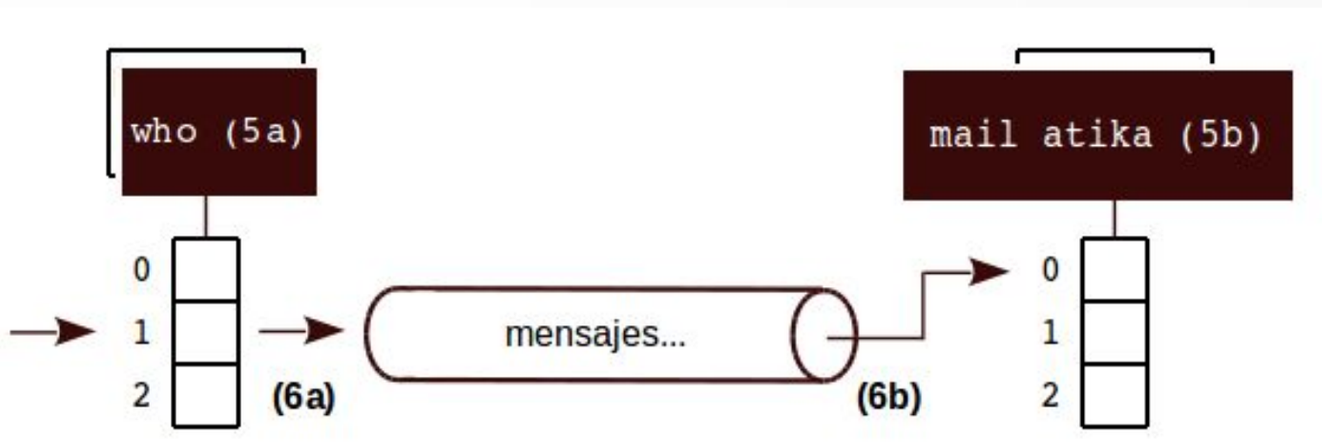


Mecanismos de concurrencia

Goroutines



Channels



MC - Goroutines

```go
func main() {

    run := func (name string, s time.Duration) {
        time.Sleep(s * time.Second)
        fmt.Println(time.Now().String()[:19], name, "executed")
    }

    run( name: "T0",  s: 0)
    run( name: "G1",  s: 3)
    run( name: "G2",  s: 2)

    bufio.NewScanner(os.Stdin).Scan()
}
```

normal functions (imperative funcs)

```
2019-02-13 14:54:42 T0 executed
2019-02-13 14:54:45 G1 executed
2019-02-13 14:54:47 G2 executed
```

G1 / 3s

G2 / 2s

```go
func main() {

    run := func (name string, s time.Duration) {
        time.Sleep(s * time.Second)
        fmt.Println(time.Now().String()[:19], name, "executed")
    }

    go run( name: "T0",  s: 0)
    go run( name: "G1",  s: 3)
    go run( name: "G2",  s: 2)

    bufio.NewScanner(os.Stdin).Scan()
}
```

Go functions (goroutines)

```
2019-02-13 14:55:26 T0 executed
2019-02-13 14:55:28 G2 executed
2019-02-13 14:55:29 G1 executed
```

G2 / 2s

G1 / 3s

MC - Channels

```go
package main

import "fmt"

var done = make(chan bool)
var msgs = make(chan int)

func produce () {
    for i := 0; i < 10; i++ {
        msgs <- i
    }
    done <- true
}

func consume () {
    for {
        msg := <-msgs
        fmt.Println(msg)
    }
}

func main () {
    go produce()
    go consume()
    <- done
}
```

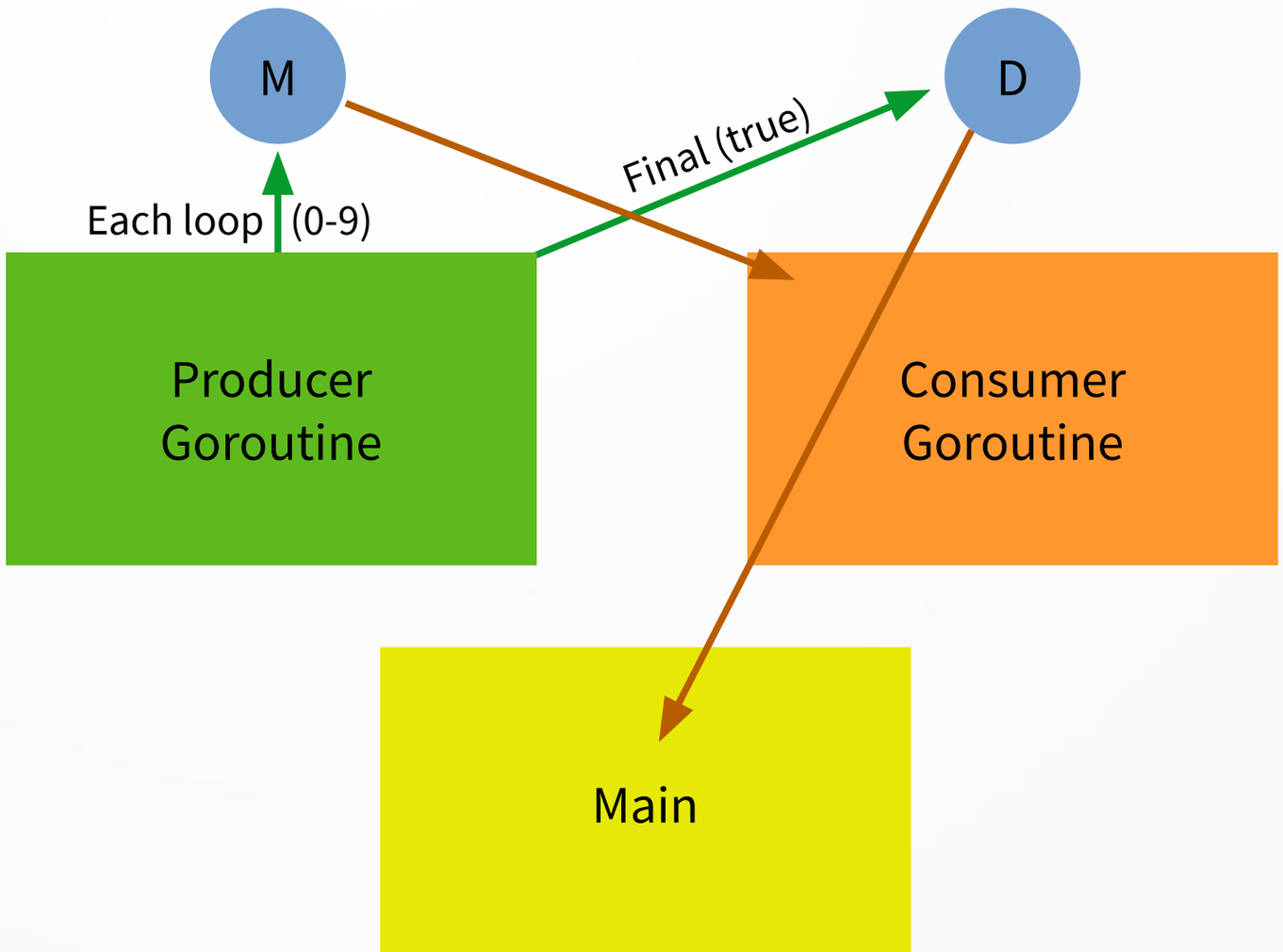
Channel builts

IN - pipe

OUT - pipe

Go funcs

Paint X lite



Conclusiones

- Es un lenguaje bastante fácil de usar
- El manejo de errores es lineal (no existe el concepto de try...catch)
- Es un lenguaje diseñado para el uso de concurrencia
- La ejecución del programa desarrollado se hace sobre un archivo binario
- La construcción de una imagen Docker es mucho más pequeña
- Al usarlo ofusca el código de tu solución