JAVA TO GO

GOOGLE GO FÜR JAVA ENTWICKLER

2.2 MIO ZEILEN CODE

3.082 CONTRIBUTORS

300.000 ZEILEN CODE

776 CONTRIBUTORS

GO

HELLO GOPHER

```go
package main

import "fmt"

func main() {
  fmt.Println("Hello Gopher!")
}
```

Ausführen

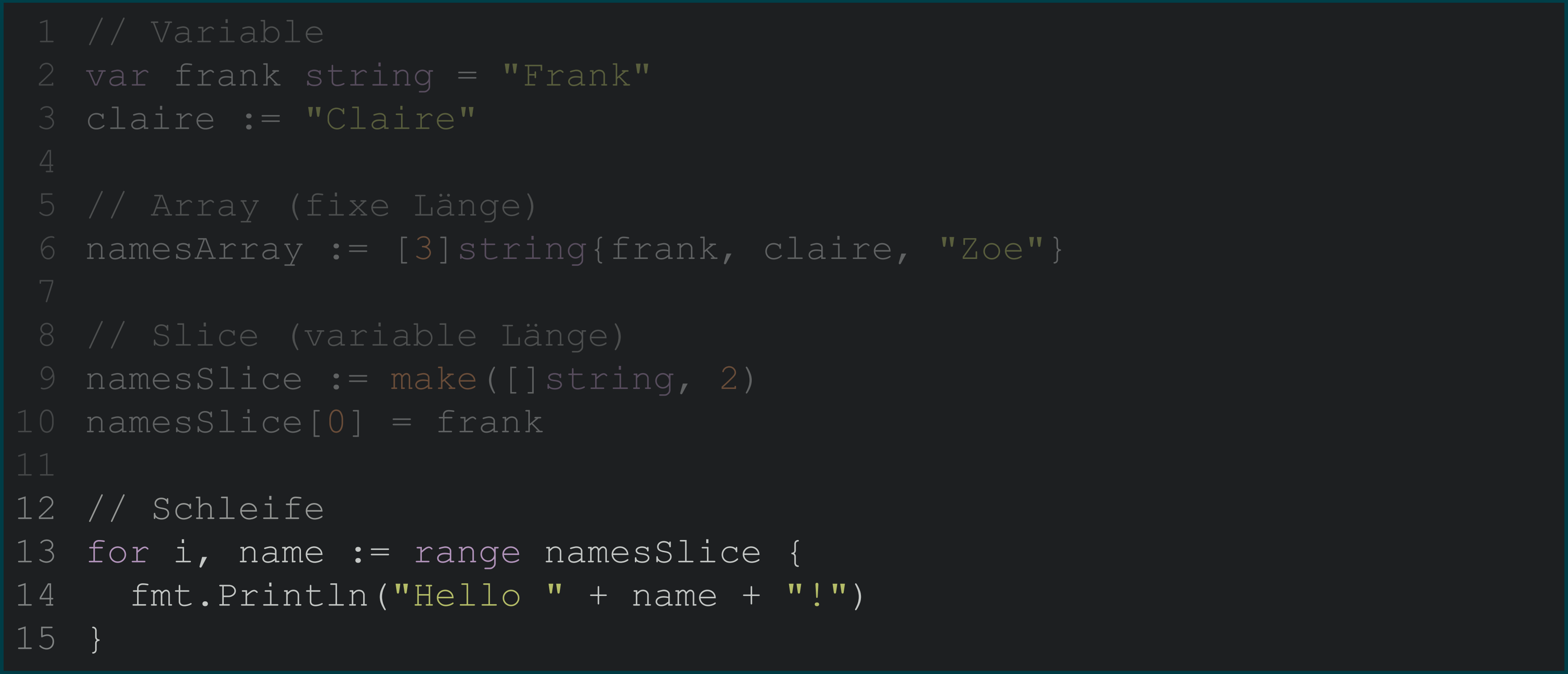
```
go build hellogopher.go // 1. Code kompilieren
./hellogopher           // 2. Binary ausführen
```

```
go run hellogopher.go   // Code kompilieren und ausführen
```

5 FAKTEN ZU GO

1 statisches Typsystem
2 Garbage Collection
3 keine Vererbung
4 Concurrency eingebaut
5 native Ausführung

VARIABLEN, SLICES, SCHLEIFEN



```go
// Variable
var frank string = "Frank"
claire := "Claire"

// Array (fixe Länge)
namesArray := [3]string{frank, claire, "Zoe"}

// Slice (variable Länge)
namesSlice := make([]string, 2)
namesSlice[0] = frank

// Schleife
for i, name := range namesSlice {
  fmt.Println("Hello " + name + "!")
}
```

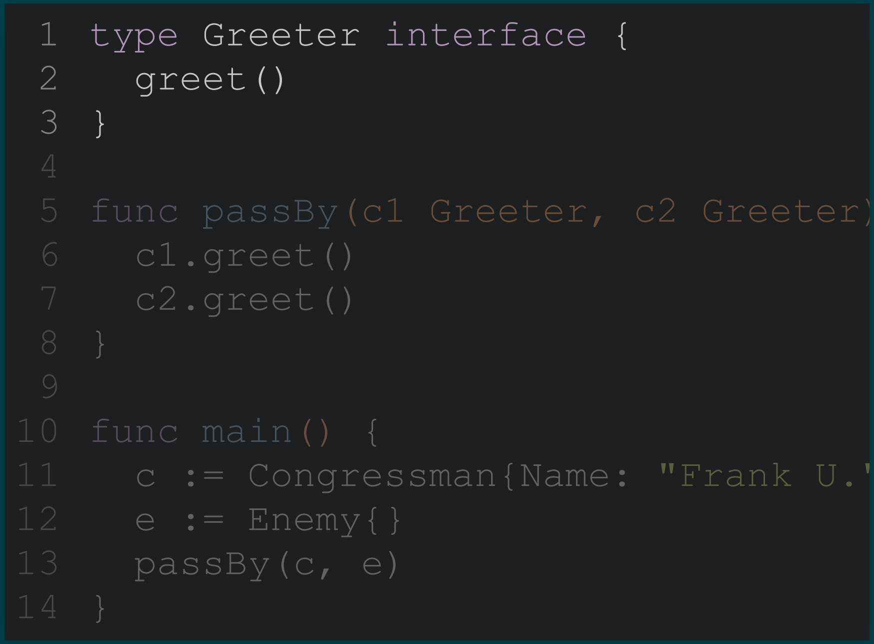
STRUCT STATT KLASSE

```go
type Congressman struct {
  Name string
}

func main() {
  c := Congressman{Name: "Peter Russo"}
  fmt.Println("Hello " + c.Name + "!")
}
```

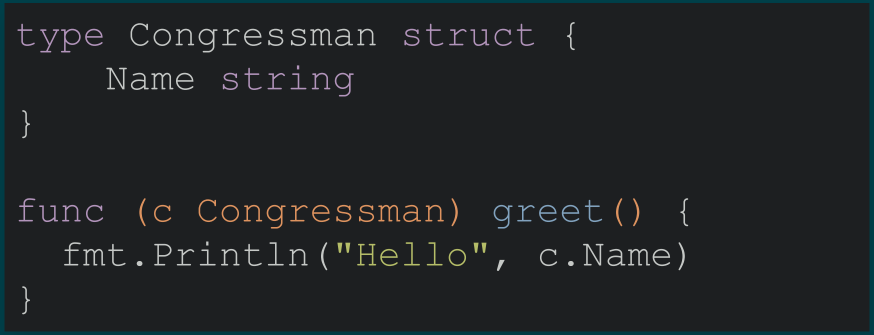
FUNCTION RECEIVER STATT INSTANZMETHODE

```go
type Congressman struct {
  Name string
}

func (c Congressman) swearOathOfOffice() {
  fmt.Printf("I, %v, swear to serve the USA.", c.Name)
}

func main() {
  c := Congressman{Name: "Peter Russo"}
  c.swearOathOfOffice();
}
```

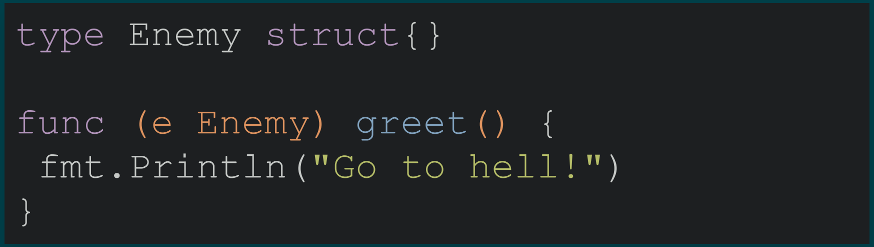
INTERFACE



```go
type Greeter interface {
  greet()
}

func passBy(c1 Greeter, c2 Greeter)
  c1.greet()
  c2.greet()
}

func main() {
  c := Congressman{Name: "Frank U."
  e := Enemy{}
  passBy(c, e)
}
```



```go
type Congressman struct {
    Name string
}

func (c Congressman) greet() {
  fmt.Println("Hello", c.Name)
}
```



```go
type Enemy struct{}

func (e Enemy) greet() {
 fmt.Println("Go to hell!")
}
```

LIEBEN ODER HASSEN

STRUCT EMBEDDING STATT VERERBUNG

```go
type Congressman struct {
  Name string
}

type President struct {
  Congressman  // Embedded

  NuclearWeaponCode string
}

func main() {
  p := President{NuclearWeaponCode: "123"}
  p.Name = "Frank Underwood"
  p.swearOathOfOffice();
}
```

FEHLER

```go
// Fehler als Rückgabewert
func (c Congressman) bribe(amount float64) error {
  if c.Name != "Peter Russo" {
      return errors.New("Not corrupt!")
  }
  c.AccountBalance += amount
  return nil
}

func main() {
  c := Congressman{Name: "Jackie Sharp", AccountBalance: -10.0}

  // Fehler behandeln
  err := c.bribe(5000.0)
  if err != nil {
      fmt.Printf("%v is not bribable.", c.Name)
  }
}
```

GENERICS

```
func printSliceOfInts(numbers []int) {
    for _, num := range numbers {
        fmt.Print(num, " ")
    }
}
```

Generics kommen in Go 2

```
fun                         ings []string) {
                        ange strings {
          e.Print(num, " ")
    }
}
```

STÄRKEN

CONCURRENCY

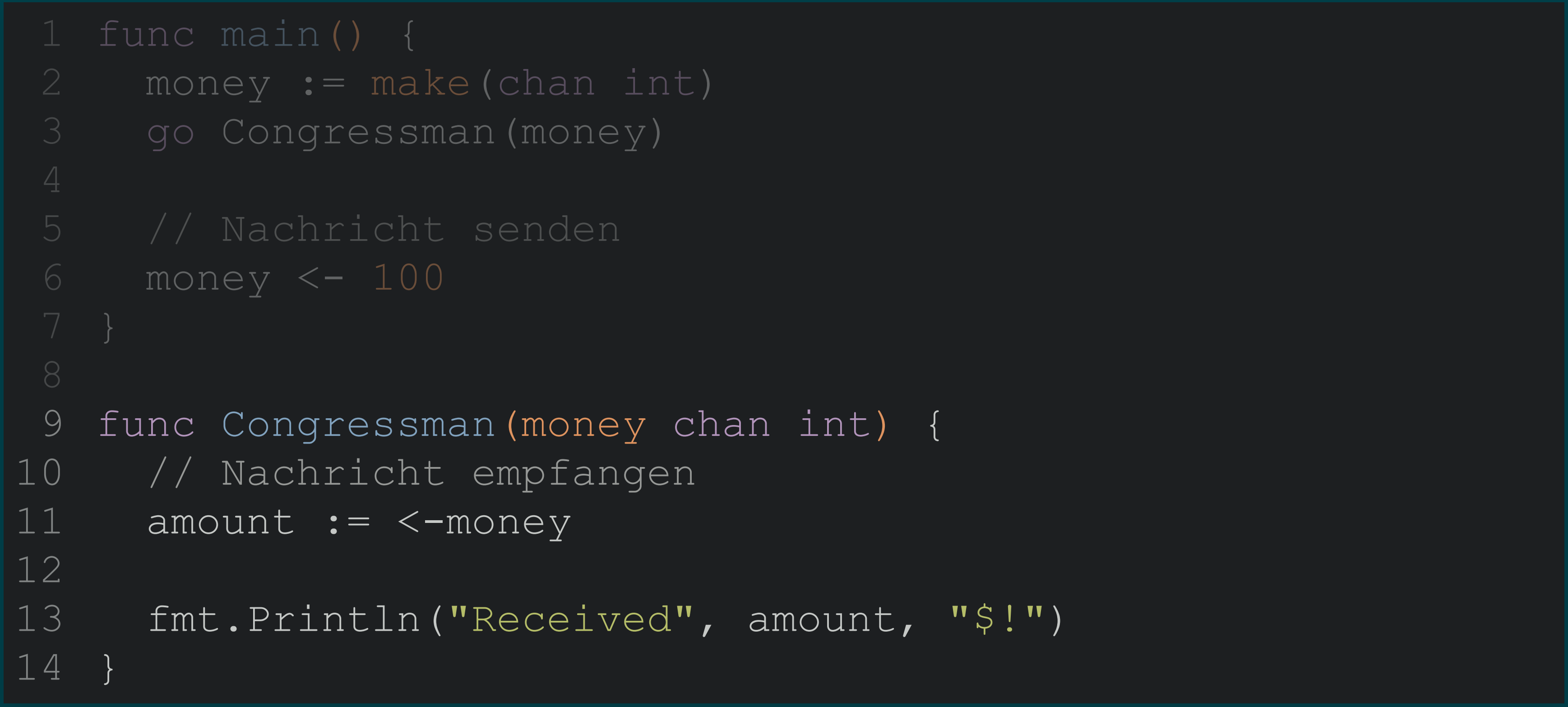
GOROUTINE
leichtgewichtiger Thread

CHANNEL
Kanal für Nachrichten

GOROUTINE

```go
func HelloCongressman(name string) {
  fmt.Println("Hello Congressman", name)
}

func main() {
  go HelloCongressman("Russo")
}
```

CHANNEL



```go
func main() {
  money := make(chan int)
  go Congressman(money)

  // Nachricht senden
  money <- 100
}

func Congressman(money chan int) {
  // Nachricht empfangen
  amount := <-money

  fmt.Println("Received", amount, "$!")
}
```

CHANNEL MIT SELECT

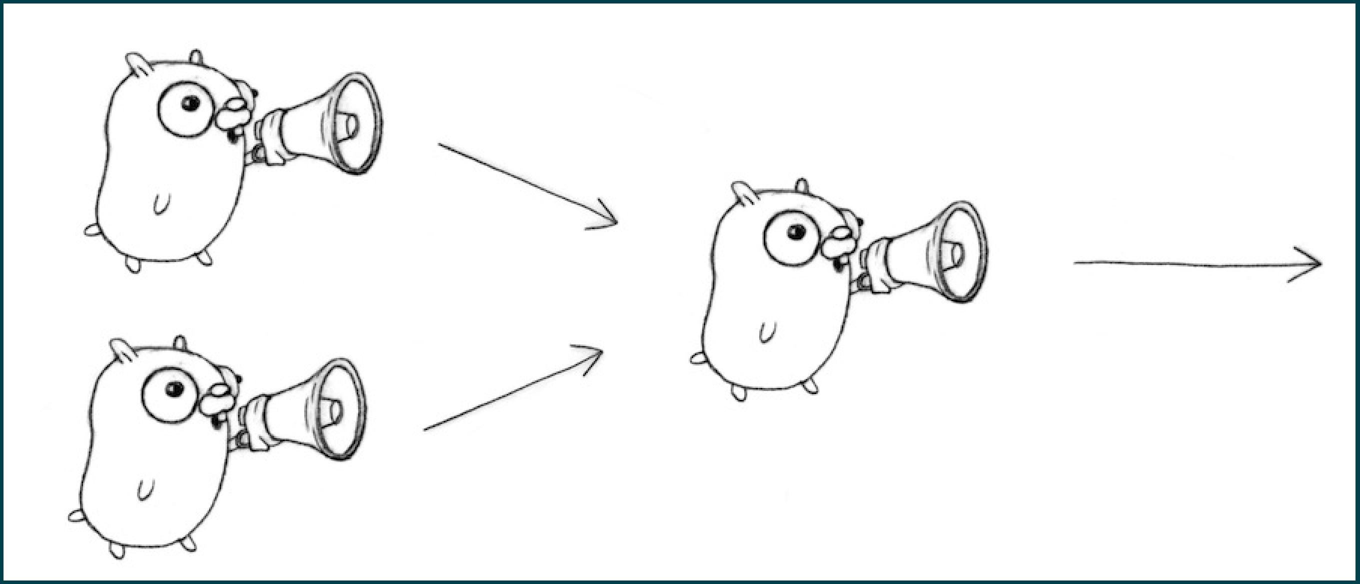
```go
func main() {
  money := make(chan int)
  go Congressman(money)

  // Nachricht senden
  money <- 100
}

func Congressman(money chan int) {
  select {
  case amount := <-money:
    fmt.Println("Received", amount, "$!")
  }
}
```

CHANNEL MIT TIMEOUT

```go
func main() {
  money := make(chan int)
  go Congressman(money)

  time.Sleep(2 * time.Second)
}

func Congressman(money chan int) {
  select {
  case amount := <-money:
    fmt.Println("Received", amount, "$!")
  case <-time.After(1 * time.Second):
    fmt.Println("Got nothing ...!")
  }
}
```

CONCURRENCY

mit Goroutinen und Channels



STANDARDBIBLIOTHEK

#   Tests
HTTP(2) Server und Router
JSON
Logging

TESTS hellogopher_test.go

```
package main

import "testing"

func TestSayHelloGopher(t *testing.T) {
    if HelloGopher() != "Hello Gopher!" {
        t.Error("Unerwartetes Ergebnis.")
    }
}
```

BASICS

Variablen, Slices, Schleifen
Struct
Interface

LIEBEN ODER HASSEN

Struct Embedding
Fehler
Generics

STÄRKEN

Goroutine
Channel
Standardbiblioth

GO!
GO!

JAN STAMER

Solution Architect

jan.stamer@comdirect.de