

# Programmier-Einführung mit Go

## Eigene Datentypen

Reiner Hüchting

10. November 2025

# Eigene Datentypen – Überblick

Eigene Datentypen

Definition eigener Datentypen

Strukturierte Datentypen

# Definition eigener Datentypen

## *Schlüsselwort type*

- ▶ Definition neuer Namen für Datentypen.
- ▶ Bessere Lesbarkeit und Verständlichkeit.
- ▶ Modellierung von Domänen-spezifischen Typen.

# Definition eigener Datentypen

## *Schlüsselwort type*

- ▶ Definition neuer Namen für Datentypen.
- ▶ Bessere Lesbarkeit und Verständlichkeit.
- ▶ Modellierung von Domänen-spezifischen Typen.

## Beispiel: Längeneinheiten

- ▶ Definiere Datentyp Length für Längenangaben.
- ▶ Ist i.W. ein int .
- ▶ Verhindert Verwechslung mit anderen int -Werten.

# Definition eigener Datentypen

## Längen-Datentyp

```
1 func ExampleLength() {  
2     var a Length = 10  
3  
4     fmt.Println(a)  
5  
6     // Output:  
7     // 10  
8 }
```

# Definition eigener Datentypen

## *Methoden*

- ▶ Spezielle Funktionen, die zu einem Typ gehören.
- ▶ Werden mit einem *Receiver* aufgerufen.
- ▶ Können Besonderheiten des Typs abbilden.

# Definition eigener Datentypen

## Exportmethoden

```
1 func (l Length) Centimeters() int {
2     return int(l)
3 }
4
5 func (l Length) Meters() int {
6     return int(l / 100)
7 }
8
9 func (l Length) Kilometers() int {
10    return l.Meters() / 1000
11 }
```

# Definition eigener Datentypen

## Exportmethoden

```
1 func ExampleLength_conversions() {
2     var a Length = 500000
3
4     fmt.Println(a.Centimeters())
5     fmt.Println(a.Meters())
6     fmt.Println(a.Kilometers())
7
8     // Output:
9     // 500000
10    // 5000
11    // 5
12 }
```

# Definition eigener Datentypen

## *Konstruktoren*

- ▶ Funktionen, die ein Objekt eines Typs erstellen.
- ▶ Verbergen Initialisierungslogik.

# Definition eigener Datentypen

## Konstruktoren

```
1 func FromMeters(m int) Length {  
2     return Length(m * 100)  
3 }  
4  
5 func FromCentimeters(m int) Length {  
6     return Length(m)  
7 }  
8  
9 func FromKilometers(m int) Length {  
10    return Length(m * 1000 * 100)  
11 }
```

# Definition eigener Datentypen

## Konstruktoren

```
1 func ExampleLength_from() {
2     a := FromMeters(5)
3     b := FromCentimeters(5)
4     c := FromKilometers(5)
5
6     fmt.Println(a)
7     fmt.Println(b)
8     fmt.Println(c)
9
10    // Output:
11    // 500
12    // 5
13    // 500000
14 }
```

## Definition eigener Datentypen

Aufgabe: Entwerfen Sie einen Datentyp Duration

- ▶ Modelliert eine Zeitspanne.
- ▶ Speichert Sekunden.
- ▶ Bietet Export/Import als Stunden, Minuten und Sekunden.

# Strukturierte Datentypen

## *Schlüsselwort struct*

- ▶ Definition von zusammengehörigen Variablen.
- ▶ Modellierung von komplexen Datenstrukturen.

# Strukturierte Datentypen

## *Schlüsselwort struct*

- ▶ Definition von zusammengehörigen Variablen.
- ▶ Modellierung von komplexen Datenstrukturen.

## Beispiel: GPS-Koordinaten

- ▶ Definiere struct für Längen- und Breitengrad.
- ▶ Beide sind float64 -Werte.
- ▶ Methode, um Distanz zu einer anderen Koordinate zu berechnen.

# Strukturierte Datentypen

## Struct für Koordinaten

```
1 type Coord struct {  
2     Longitude float64  
3     Latitude  float64  
4 }
```

# Strukturierte Datentypen

## Verwendung

```
1 func ExampleCoord_usage() {
2     a := Coord{0, 0}
3     b := Coord{3, 4}
4
5     fmt.Println(a.Longitude)
6     fmt.Println(b.Latitude)
7
8     a.Latitude = 1
9     fmt.Println(a.Latitude)
10
11    // Output:
12    // 0
13    // 4
14    // 1
15 }
```

# Strukturierte Datentypen

## Distanz-Methode

```
1 func (c Coord) Distance(o Coord) float64 {  
2     x := c.Longitude - o.Longitude  
3     y := c.Latitude - o.Latitude  
4     return math.Sqrt(x*x + y*y)  
5 }
```

# Strukturierte Datentypen

## Distanz-Methode

```
1 func ExampleCoord_Distance() {
2     a := Coord{0, 0}
3     b := Coord{3, 4}
4
5     d := a.Distance(b)
6
7     fmt.Println(d)
8
9     // Output:
10    // 5
11 }
```

