Swift Cheat Sheet

Variablen und Konstanten

```swift
var mutableDouble : Double = 1.0
mutableDouble = 3.0 // Da var = variable -> Wert kann geändert werden

let constantDouble : Double = 20.0
// constantDouble = 30.0 // Fehler da let = Konstant

// Type Inference -> Datentyp Ableitung
var aInt = 30.0
var aString = "Ich bin ein String"
let aDouble = 40.0
```

Schleifen

```swift
for number in 0...10 {
    // ich laufe in einer dauerschleife...
}
```

Datenstrukturen

```swift
var arrayName = [Int]()
arrayName.append(1)
arrayName.append(10)
arrayName.append(100)
arrayName.append(1000)

for number in arrayName {
    print("Zahl \(number)")
}
```

Kontrollsturkturen

```swift
var bedingung = true

if bedingung == true {
    // Falls die bedingung stimmt (true) wird
} else {
    // ansonsten stimmt die bedingung nicht
}
```

Funktionen

```swift
func doIt() {
    // ich erledige meine Aufgabe beim Aufruf
}

doIt()

func doIt(value: Int) {
    // ich erledige meine Aufgabe und erhalten dafür einen Parameter value
}

func doIt(value: Int, value2: Int) -> Int {
    return value + value2
}
```

Swift Cheat Sheet

Begriffe

```
var userName: String =   // Deklaration -> Datentyp angeben / erhalten
```

```
var userName: String = "Max Mustermann" // Initialisierung -> Variable erhält des ersten Wert
```

Operatoren

```
// Mathematische Operatoren (+,-,/,*)
// Zuweisung (=,+=,-=)
// Vergleich (==, !=, <, >)
// Logische Verknüpfungen (&&, ||)
// Range (..., ..<)
```

```
var number1 = 20
var number2 = 10
var result = number1 + number2

number1 -= number2
```

```
if true && number2 < number1 { // stimmt / stimmt nicht
    print("Ich werde aufgerufen, falls  beide Bedingungen stimmen")
}
```

Switch

```
var highScore = 1000

switch highScore {
case 500: print("1 Stern")
case 1000: print("2 Sterne")
case 3000: print("3 Sterne")
default:
    print("Keine Sterne")
}

if highScore == 500 {
    print("1 Stern")
} else if highScore == 1000 {
    print("2 Sterne")
} else if highScore == 3000 {
    print("3 Sterne")
} else {
    print("Keine Sterne")
}
```

While Schleife

```
for zahl in 0...6 { // 7
    print(zahl)
}
```

```
var anzahl = 10

while 0 > 0 { // stimmt nicht
    print("Gegner erstellen")

    anzahl -= 1
}
```

Swift Cheat Sheet

Optionals

```
// Was waren Datentypen? -> allgemein: Gibt den Typ der Information an
// 1. Merke: Swift muss den Datentyp einer Variable Wissen
// 2. Merke: Vor der erste Benutzung einer Variable muss dort ein Wert vorhanden se

var wert1: Int = 4
//wert1 = "vier"
var wert2 = 4

var wert3: Int? = nil// Int? = Optional = Ein eigener Datentyp mit 2 Möglichkeiten
// 1. Hat einen Wert 2. hat keinen Wert (nil)
```

optional binding

```
// 2. Optional Binding - if let Wert vorhanden else ...
var playername : String?

if let name = playername {

    print(name)


} else {

    print("Bitte Namen eingeben")
}
```

force unwrapping

```
// 1. forced unwrapping (!) - MIR SCHEI.. egal ob was drin ist PACKE ES AUS!!!

var email: String?
print(email!)
```

guard statement

```
// 3. guard Statement - guard Wert vorhanden else ..

func testUsername() {

    guard let name = playername else {
        print("Willkommen Namenloser")
        return
    }

    print(name)
}

testUsername()
```

Swift Cheat Sheet

Klasse erstellen

1. Eigenschaften
2. evtl. init()
3. Methoden „Funktionen“

```
class Auto {

    // Eigenschaften
    var marke = "BMW"
    var ps = 140
    var preis = 20000

    // Funktionen -> Methoden
    func starte() {
        print("brum brum")
    }

    func gasgeben() {
        print("Beschleunigt")
    }
}
```

Objekte erstellen

```
let autoOne = Auto()
```

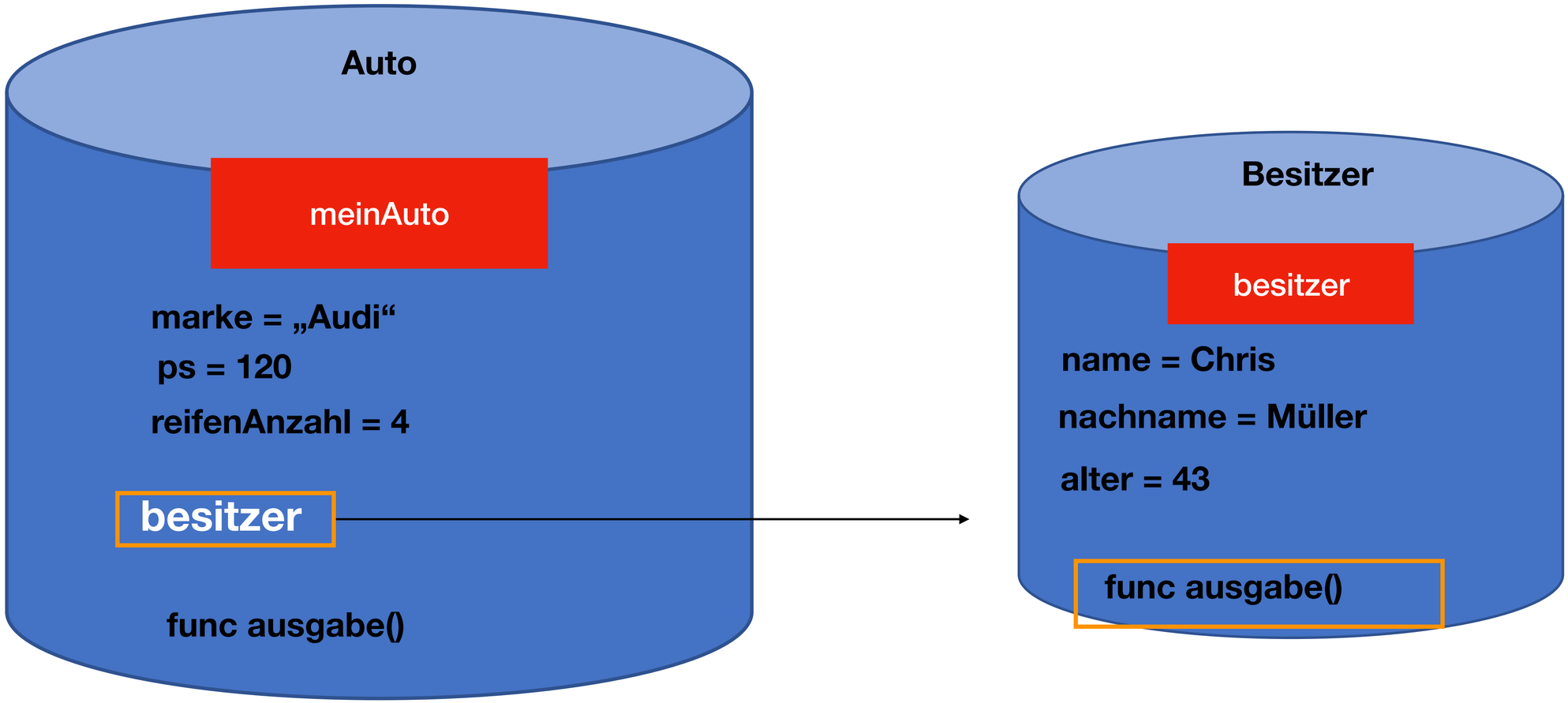
Punktoperator nutzen

```
autoOne.marke = "Audi"
autoOne.starte()
```

Swift Cheat Sheet

```
meinAuto.besitzer.ausgabe()
```

Klasse als Eigenschaft einer anderen Klasse



Swift Cheat Sheet

Vererbung

Was ist Vererbung?

Besteht aus einer Superklasse(Mutterklasse) die Ihre Eigenschaften, init und Methoden an die Subklassen(Kindklassen) weiter gibt.

Die Subklasse erbt die Eigenschaften, init und die Methoden der Superklasse

```swift
class Tier { // Superklasse

    // Eigenschaften
    var name = " "
    var gewicht = 0.0
    var alter = 0

    // init
    init(_name: String, _gewicht: Double, _alter: Int) {
        self.name = _name
        self.gewicht = _gewicht
        self.alter = _alter
    }

    // Methoden
    func machLaute() {
        print("Das Tier macht Laute")
    }
}
```

So wird die Vererbung angegeben

```swift
class Hund: Tier { // Subklasse

}

var meinHund = Hund(_name: "Bello", _gewicht: 5.0, _alter: 4)
meinHund.machLaute()


class Katze: Tier { // Subklasse


}
```

Swift Cheat Sheet

Protokolle

Ein Protokol besteht nur aus:

- Nur Eigenschaften ohne Initialisierung
- Methodensignatur ohne Implementierung

- Klassen oder Strukturen und Enum können/müssen Protokolle implementieren

```swift
protocol MyProtocol {

    // Nur Eigenschaften ohne initialisirung
    var str: String {get set}

    // Nur Funktionenköpfe keine Implementierung
    func summe(zahl1: Int, zahl2: Int) -> Int
}
```

So wird ein Protocol angegeben

```swift
class MyClass: MyProtocol {

    var str = "Hallo"

    func summe(zahl1: Int, zahl2: Int) -> Int {
        return zahl1 + zahl2
    }
}
```

Swift Cheat Sheet

Datenstrukturen: Daten in einer Struktur speichern

Array:

```
Array(Liste)

Eigenschaften:
- Zuordnung durch einen index
- feste Reihenfolge
```

```swift
// Array erstellen
var namen = [String]()
var zahlen : [Int] = [1,2,3,4,10]

// Anzahl der Elemente
namen.count

// Element hinzufügen
namen.append("Peter")
namen.append("Michael")
namen.append("Sven")

// Element an einer bestimmten Stelle hizufügen
namen.insert("Chris", at: 0)
namen.insert("Petra", at: 2)


// Ausgabe der Elemente
for name in namen {
    print(name)
}
```

Set:

```
Set(Menge)

Eigenschaften:
- Keine feste Ordnung
- Jedes Element ist Einzigartig
```

```swift
// Set erstellen
var numbers = Set<Double>()
var musikGenres: Set<String> = ["Rock", "Klassik", "Pop"]

// Anzahl der Elemente
numbers.count

// Element hinzufügen
musikGenres.insert("Jazz")

// Ausgabe der Elemente
for gerne in musikGenres {
    print(gerne)
}
```

Dictionary:

```
Dictionary

Eigenschaften:
- Keine feste Ordnung
- Jedes Element hat seine
Einzigartigen Schlüssel
- Elemente nur über Schlüssel
erreichbar
```

```swift
// Dictionarie erstellen
var numbersAndString = [Int: String]()

// Anzahl der Elemente
numbersAndString.count

// Element hinzufügen
numbersAndString[1] = "Peter"
numbersAndString[10] = "Sven"
numbersAndString[4] = "Katja"
numbersAndString[400] = "Sabrina"

// Ausgabe nur der Keys
for number in numbersAndString.keys {
    print(number)
}

// Ausgabe nur der Values
for string in numbersAndString.values {
    print(string)
}
```