

# Methods

Maximilian Arthofer

---

# Allgemein

---

- Funktionen die einem bestimmten Typ zugeordnet werden
  - Class
  - Structure
  - Enum

---

# Instance Methods

---

- Methoden die einer Instanz angehören
- In den Klammern des entsprechenden Blockes geschrieben
  - Greifen auf Werte zu
  - Stellen den Inhalt des Block betreffende Funktionen zur Verfügung
- Hat Zugriff auf andere Methods oder Properties
- Kann nur über eine Instanz aufgerufen werden

# Instance Methods

- Deklaration ist gleich der einer Funktion

```
class Counter {  
    var count = 0  
    func increment() {  
        count += 1  
    }  
    func increment(by amount: Int) {  
        count += amount  
    }  
    func reset() {  
        count = 0  
    }  
}
```

- Zugriff wie bei Properties

```
let counter = Counter()  
// the initial counter value is 0  
counter.increment()  
// the counter's value is now 1  
counter.increment(by: 5)  
// the counter's value is now 6  
counter.reset()  
// the counter's value is now 0
```

# Self Property

- Zugriff auf die eigene Instanz
- Gleich „this“ unter Java
- Normalerweise nicht notwendig
  - Ausnahme: Property-Name==Parameter-Name

```
func increment() {  
    self.count += 1  
}
```

```
struct Point {  
    var x = 0.0, y = 0.0  
    func isToTheRightOf(x: Double) -> Bool {  
        return self.x > x  
    }  
}  
  
let somePoint = Point(x: 4.0, y: 5.0)  
if somePoint.isToTheRightOf(x: 1.0) {  
    print("This point is to the right of the line where x == 1.0")  
}  
  
// Prints "This point is to the right of the line where x == 1.0"
```

---

# Mutating Methods

---

- Zum Verändern von Property-Werten innerhalb einer Methode

```
struct Point {  
    var x = 0.0, y = 0.0  
    mutating func moveBy(x deltaX: Double, y deltaY: Double) {  
        x += deltaX  
        y += deltaY  
    }  
}  
  
var somePoint = Point(x: 1.0, y: 1.0)  
somePoint.moveBy(x: 2.0, y: 3.0)  
print("The point is now at \(somePoint.x), \(somePoint.y)")  
// Prints "The point is now at (3.0, 4.0)"
```

# Mutating Methods

- Sich selbst einen neuen Wert zuweisen

```
struct Point {  
    var x = 0.0, y = 0.0  
    mutating func moveBy(x deltaX: Double, y deltaY: Double) {  
        self = Point(x: x + deltaX, y: y + deltaY)  
    }  
}
```

```
enum TriStateSwitch {  
    case off, low, high  
    mutating func next() {  
        switch self {  
            case .off:  
                self = .low  
            case .low:  
                self = .high  
            case .high:  
                self = .off  
        }  
    }  
}
```

```
var ovenLight = TriStateSwitch.low  
ovenLight.next()  
// ovenLight is now equal to .high  
ovenLight.next()  
// ovenLight is now equal to .off
```

---

# Type Methods

---

- Methoden die einem Typen angehören
- Gekennzeichnet durch *static* oder *class*

```
class SomeClass {  
    class func someTypeMethod() {  
        // type method implementation goes here  
    }  
}  
SomeClass.someTypeMethod()
```

- *self* verweist jetzt auf den Typen
- Properties die so verändert werden sind für alle Instanzen gleich



# Type Methods

```
struct LevelTracker {
    static var highestUnlockedLevel = 1
    var currentLevel = 1

    static func unlock(_ level: Int) {
        if level > highestUnlockedLevel { highestUnlockedLevel =
level }
    }

    static func isUnlocked(_ level: Int) -> Bool {
        return level <= highestUnlockedLevel
    }

    @discardableResult
    mutating func advance(to level: Int) -> Bool {
        if LevelTracker.isUnlocked(level) {
            currentLevel = level
            return true
        } else {
            return false
        }
    }
}
```

```
class Player {
    var tracker = LevelTracker()
    let playerName: String
    func complete(level: Int) {
        LevelTracker.unlock(level + 1)
        tracker.advance(to: level + 1)
    }
    init(name: String) {
        playerName = name
    }
}
```

---

# Type Methods

---

```
var player = Player(name: "Argyrios")
player.complete(level: 1)
print("highest unlocked level is now
      \((LevelTracker.highestUnlockedLevel)")
// Prints "highest unlocked level is now 2"
```

```
player = Player(name: "Beto")
if player.tracker.advance(to: 6) {
    print("player is now on level 6")
} else {
    print("level 6 has not yet been unlocked")
}
// Prints "level 6 has not yet been unlocked"
```

---

# Beispiel

---

- Erstelle eine Klasse Fahrzeug mit den Attributen: Typ (String), Motorleistung und Sitzanzahl.
- Erstelle eine Methode die es ermöglicht die Typen aller Fahrzeuge gleichzeitig zu ändern.
- Ein Fahrzeug soll ganz einfach durch ein anderes ausgetauscht werden können weshalb die Methode changeVehicle() ein neues Fahrzeug entgegennimmt und die aktuelle damit ersetzt.
- Zwei Getter- bzw. Setter-Methoden sollen dazu dienen die Instanz eines Fahrzeugs zu bearbeiten.