



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences



MAD5

Protokolle

Prof. Dr. Dragan Macos



- Eine Menge von Methoden, Properties, Konstanten und anderen Eigenschaften, die ein Teil des Softwarecodes beinhalten muss.
- Syntax

```
protocol SomeProtocol {  
    // Protokolldefinition  
}
```





- .. Oder „Protokoll einhalten“

```
struct SomeStructure: FirstProtocol,  
    AnotherProtocol {  
    // Strukturdefinition  
}
```

Diese Struct würde
gerne das Protokoll
FirstProtokoll einhalten

```
class SomeClass: SomeSuperclass,  
    FirstProtocol, AnotherProtocol {  
    // Klassendefinition  
}
```

Zuerst die Oberklasse,
wenn es welche gibt,
danach Protokolle.



- Properties (Typ- und Instanz-) mit bestimmten Namen und Typen
- Ein Protokoll sagt nicht, ob ein Property „computed“ oder „stored“ ist.
- Ein Protokoll spezifiziert, ob ein Property einen setter haben muss oder nicht.
- Beispiel

```
protocol SomeProtocol {  
    var mustBeSettable: Int { get set }  
    var doesNotNeedToBeSettable: Int { get }  
}  
  
protocol AnotherProtocol {  
    class var someTypeProperty: Int { get set }  
}
```

Es muss ein
getter und ein
setter geben.

Anforderung nach einem Type-Property.
Immer mit „class“ trotz den anderen
Bezeichnungen für Type-Props bei enum
und struct.



```
protocol FullyNamed {  
    var fullName: String { get }  
}
```

```
struct Person: FullyNamed {  
    var fullName: String  
}
```

```
let john = Person(fullName: "John Appleseed")  
// john.fullName is "John Appleseed"
```



- Protokolle können als Typen verwendet werden.
- Dies bedeutet „Irgendein Typ, der das Protokoll einhält“.
- Protokollelemente, die nicht zwingend umgesetzt werden sollen, werden mit „@optional“ gekennzeichnet.





- Nachträgliche Protokollkonformität deklarieren durch „extension“

```
protocol TextRepresentable {  
    func asText() -> String  
}
```

```
extension Dice: TextRepresentable {  
    func asText() -> String {  
        return "A \((sides)-sided dice"  
    }  
}
```

- Protokolle können (ver)erben



Die Klasse Dice wird um die Methode „asText“ erweitert, um das Protokoll TextRepresentable einzuhalten.



- „is“ – liefert true, wenn eine Instanz zu einem Protokoll konform ist.
- „as“ – Forciertes Casting zur Protokollkonformität





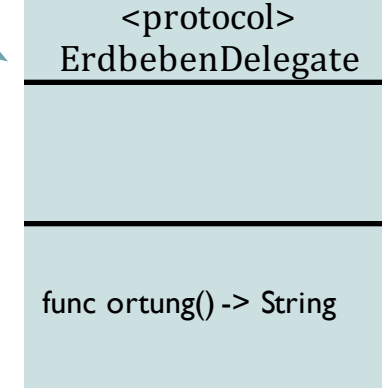
- Ein Entwurfsmuster (Design Pattern)
- Eine Möglichkeit, Funktionalitäten an einen anderen Typen zu delegieren
- Wenn wir etwas an jemanden delegieren, müssen wir sicher sein, das er das auch kann 😊



Beispiel



Das Protokoll
„ErdbebenDelegate“, das das
delegate
braucht



Protokoll als Typ

Das delegate. An dem wird die
Funktion „ertung“ aufgerufen,
um die Hilfe dorthin zu schicken.

ErdbebenManagement

```
var delegate:
  ErdbebenDelegate
```

```
func schickeHelfer()-> ()
```

```
{ print("Hilfe senden nach:" +
  self.delegate.ertung() + " ") }
```

Beispiel



Das Protokoll „ErdbebenDelegate“, der Typ von *delegate..*

ErdbebenManagement

```
var delegate:  
    ErdbebenDelegate
```

```
func schickeHelfer()-> ()
```

Das delegate. An dem wird die Funktion „ortung“ aufgerufen, um die Hilfe dorthin zu schicken.

```
{ print("Hilfe senden nach:" +  
    self.delegate.ortung() + " ") }
```

<protocol> ErdbebenDelegate

```
func ortung() -> String
```

Sie halten das Protokoll „ErdbebenDelegate“ ein.

<class>

KleingartenPerson: ErdbebenDelegate

```
var name = " "  
var ort = " "  
var steuernummer = 0
```

```
func ortung() -> String {  
    return ort  
}
```

<struct>

UebungsgelaendeBW: ErdbebenDelegate

```
var gpsKoordinaten =  
    "0.0.0"
```

```
func ortung() -> String {  
    return gpsKoordinaten  
}
```

<struct>

Hippie: ErdbebenDelegate

```
var name = " "  
var mq:  
    MarihuanaQualitaet
```

```
func ortung() -> String {  
    return "++Nirvana++"  
}
```

Beispiel



ErdbebenManagement

```
var delegate:
    ErdbebenDelegate
```

```
func schickeHelfer()-> ()
```

⋮

```
{ print("Hilfe senden nach:" +
    self.delegate.ortung() + " ") }
```

<protocol>

ErdbebenDelegate

```
func ortung() -> String
```

```
let erdbebenManager = ErdbebenManagement()
let buerger = KleingartenPerson(name: "WilliSpieß", ort: "Kleingarten", steuernummer: 1)
let hippie = Hippie(name: "Krishna", mq: nil)
let bwGelaendeBrandenburg = UebungsgelaendeBW(gpsKoordinaten: "GPS: <123.234.23>")
erdbebenManager.delegate = buerger
erdbebenManager.schickeHelfer()
erdbebenManager.delegate = hippie
erdbebenManager.schickeHelfer()
erdbebenManager.delegate = bwGelaendeBrandenburg
erdbebenManager.schickeHelfer()
```

<class>

KleingartenPerson:
ErdbebenDelegate

```
var name = " "
var ort = " "
var steuernummer = 0
```

```
func ortung() -> String {
    return ort
}
```

<struct>

UebungsgelaendeBW:
ErdbebenDelegate

```
var gpsKoordinaten =
    "0.0.0"
```

```
func ortung() -> String {
    return gpsKoordinaten
}
```

<struct>

Hippie:
ErdbebenDelegate

```
var name = " "
var mq:
    MarihuanaQualitaet
```

```
func ortung() -> String {
    return "++Nirvana++"
}
```

Beispiel



ErdbebenManagement

var delegate:
ErdbebenDelegate

func schickeHelfer()-> ()

```
{ print("Hilfe senden nach:" +  
self.delegate.ortung() + " ") }
```

×

Console Output

```
Hilfe senden nach:Kleingarten  
Hilfe senden nach:++++Nirvana++++  
Hilfe senden nach:GPS: <123.234.23>
```

<protocol>

ErdbebenDelegate

func ortung() -> String

```
let erdbebenManager = ErdbebenManagement()  
let buerger = KleingartenPerson(name: "WilliSpieß", ort: "Kleingarten", steuernummer: 1)  
let hippie = Hippie(name: "Krishna", mq: nil)  
let bwGelaendeBrandenburg = UebungsgelaendeBW(gpsKoordinaten: "GPS: <123.234.23>")  
erdbebenManager.delegate = buerger  
erdbebenManager.schickeHelfer()  
erdbebenManager.delegate = hippie  
erdbebenManager.schickeHelfer()  
erdbebenManager.delegate = bwGelaendeBrandenburg  
erdbebenManager.schickeHelfer()
```

<class>

KleingartenPerson:
ErdbebenDelegate

var name = " "
var ort = " "
var steuernummer = 0

```
func ortung() -> String {  
    return ort  
}
```

<struct>

UebungsgelaendeBW:
ErdbebenDelegate

var gpsKoordinaten =
"0.0.0"

```
func ortung() -> String {  
    return gpsKoordinaten  
}
```

<struct>

Hippie:
ErdbebenDelegate

var name = " "
var mq:
 MarihuanaQualitaet

```
func ortung() -> String {  
    return "++Nirvana++"  
}
```

Delegate, das Hauptprinzip



ErdbebenManagement

```
var delegate:  
    ErdbebenDelegate
```

```
func schickeHelfer()-> ()
```

Das delegate-Objekt wird unterschiedlich gesetzt. Egal welches Objekt das ist, kann er die Funktion „ortung“ ausführen.
Die „ortung“ oder „Sag wo Du bist“, wird ausgelagert. Delegiert.

Die meisten Sourcecode-Beispiele und die Sprachdefinition der Sprache Swift wurden aus:

Apple Inc. „The Swift Programming Language.“ iBooks. <https://itun.es/de/jEUH0.I>

genommen.

Eventuelle andere Quellen bzw. eigene Beispiele werden an den entsprechenden Stellen direkt angegeben bzw. gekennzeichnet.