

Macoun

OOP ist tot, es lebe POP

Marco Feltmann
@marcofeltmann

Ablauf

- Datenverarbeitung
- Strukturierte Programmierung
- Objektorientierung
- Protokollorientierung

Datenverarbeitung

Eine Zeitreise

Daten und Verarbeitung

- Daten repräsentieren einen Zustand
- Verarbeitung von Daten ermöglichen Zustandsveränderung
- Zusammenhang bereits 1936 von Alan M. Turing formuliert
"On Computable Numbers, with an application to the Entscheidungsproblem"

Am Anfang war das word

1 Byte	8 Bit	byte	(u)int8_t/char
2 Byte	16 Bit	word	(u)int16_t/short
4 Byte	32 Bit	double word	(u)int32_t/float
8 Byte	64 Bit	quad word	(u)int64_t/double
10 Byte	80 Bit	ten byte	long double

Worte verbiegen

MOVW EAX, [bar]

uint16_t foo = bar;

ADDW EAX, 100

bar += (uint16_t)100;

CALL foobar

foobar();

RET

return;

Assembler

- Direktes Arbeiten auf den Registern der CPU
- Herumschieben der Datentypen ist Vertrauenssache
- Operationen sind prozessorabhängig
- Code nahezu nicht portabel

C

- Datentypen entsprechen den Assembler Datengrößen
- Funktionen abstrahieren Prozessorarchitektur
- Übersetzung in Assembler des eigenen Systems
- Übersetzung für andere Systeme möglich („cross compiling“)

Strukturierte Programmierung

Einfach machen.

struct und enum

Strukturen

Liste von Datentypen in einem Datentyp gekapselt

Enumerationen

Liste von Zuständen in einem Datentyp gekapselt

enum

```
enum orientation {  
    north,  
    east,  
    south,  
    west,  
};  
  
orientation direction;  
direction = north; // 0  
direction = east;  // 1
```

```
enum orientation {  
    northwest,  
    north,  
    northeast,  
    east,  
    southeast,  
    south,  
    southwest,  
    west,  
};  
  
orientation direction;  
direction = north; // 1≠0  
direction = east;  // 3≠1
```

struct

```
struct lat_long {  
    double latitude;  
    double longitude;  
}  
  
typedef struct lat_long geocoordinate;
```

```
struct rechts_hoch {  
    double rechtswert;  
    double hochwert;  
    uint false_easting;  
}  
  
typedef struct rechts_hoch geocoordinate;
```



Lasst uns würfeln!

Zwischenspiel – Würfel???

- Seit ca. 5000 Jahren für Gesellschaftsspiele genutzt
"Iran's Burnt City Throws up World's Oldest Backgammon"
- Physikalischer Zufallszahlengenerator
- Geometrisch interessant

Demo: CDice

Fazit

- Datenkapselung ist ein hilfreiches Feature
- Wildwuchs bei den Funktionen
- Hohe Fehleranfälligkeit bei Zeigeroperationen

Objektorientierte Programmierung

Was ist es?

Objekt: Minimalanforderungen

Enumerationen

Strukturen

haben einen Zustand

Funktionen

führen Aktionen durch

Objekt: Positive Nebeneffekte

- Daten und Funktionen in Datentyp gekapselt
- Zugriffskontrolle
- Namenszuordnung
- Erweiterbarkeit
- Ausdruckstärke

Und was ist mit Vererbung?

- JavaScript (Prototypen)
- Go (Protokolle und Erweiterungen)
- Scratch (graphische Blöcke)
- Visual Basic Classic (Klassen, aber keine Vererbung)



Doppelter Einsatz

Demo: ObjCDice

Fazit

- Vererbung nur hilfreich bei konkreter „ist ein“ Beziehung
- Unterscheidung „ist“ und „kann“ nicht gegeben
- Abstrakte Klassen verschlimmern die Situation

Protokollorientierte Programmierung

Was kann ich damit tun?

Was können Würfeln tun?

- Zufallszahlen erzeugen
- Jeder Zustand des Würfels beeinflusst eine Handlung
- Trefferzonen, Götter, Ressourcen, Prozentuale Verteilung...
- Wetteinsatz, Spielstrategie...

Demo: SwiftDice

Protokolle in Swift

- Funktionen werden in einen Datentyp gekapselt
- Entkoppelt wiederverwendbare Funktionalitätsbeschreibung
- Standardimplementierung durch Protokollerweiterung
- Für Objekte, Strukturen und Enumerationen verwendbar

Erfüllte Prinzipien

- Open Close Principle
- Leskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

Gefahren

- Übersicht indirekt proportional zur Anzahl der Protokolle
- Abhängigkeit von `self` oder `associated types` erhöhen die Komplexität
- Mehrfache Implementierung identischer Funktionalität denkbar



Zombieapokalypse

Demo: SwiftZombie



Kleine Schweinereien

Demo: SwiftPiggy

Fazit

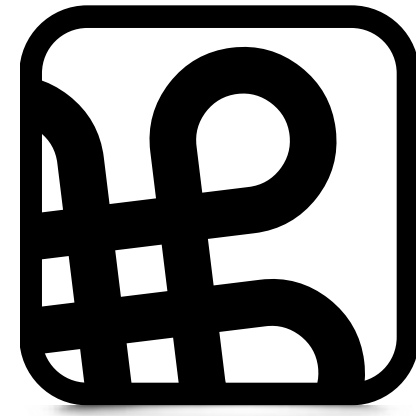
- Kenne Deine Werkzeuge
- Wähle das beste Werkzeug für den Job
- Protokollorientierung ist leider auch nicht die Weltformel...

Leitfaden der Einfachheit

- Nutzung von simplen Datentypen für minimale Komplexität
- Funktionen fügen geringe Komplexität hinzu
- Generische Funktionen sind mitunter komplexer als spezifische Funktionen
- Protokolle als Sammlung der vorherigen Elemente besitzen eine sehr hohe Komplexität
- Die Komplexität von Objekthierarchien lässt sich kaum übertreffen

Fragen?

Vielen Dank



Macoun