

# Hydroinformatik I - WiSe 2020/2021

## HyBHW-S1-01-V5: Klassen

Prof. Dr.-Ing. habil. Olaf Kolditz

<sup>1</sup>Helmholtz Centre for Environmental Research – UFZ, Leipzig

<sup>2</sup>Technische Universität Dresden – TUD, Dresden

<sup>3</sup>Center for Advanced Water Research – CAWR

<sup>4</sup>TUBAF-UFZ Center for Environmental Geosciences – C-EGS, Freiberg / Leipzig

Dresden, 20.11.2020

## Fahrplan für heute ...

1. Rückblick letzte Veranstaltung (HyBHW-S1-01-V4: Datentypen)

---

2. Objekt-Orientierung: Konzept
3. OpenGeoSys - ein objekt-orientiertes Programm
4. Ihre Fragen
5. Objekt-Orientierte Programmierung (OOP): Daten-Abstraktion, Klassen, Instanzen, ...
6. Übung: Instanzen einer Klasse

---

7. Ausblick auf die nächste Veranstaltung (HyBHW-S1-01-V6: Input/Output (I/O))

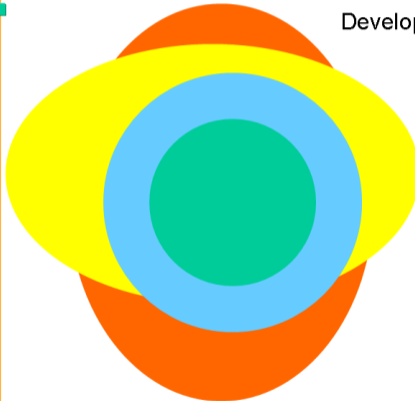
# Objekt-Orientierung



Contents

Definition

## Object-Oriented Programming



Development of software:

- ever increasing functionality
- team work
- user interfaces
  - ...

- ▶ Prozedurales Programmieren
- ▶ ...

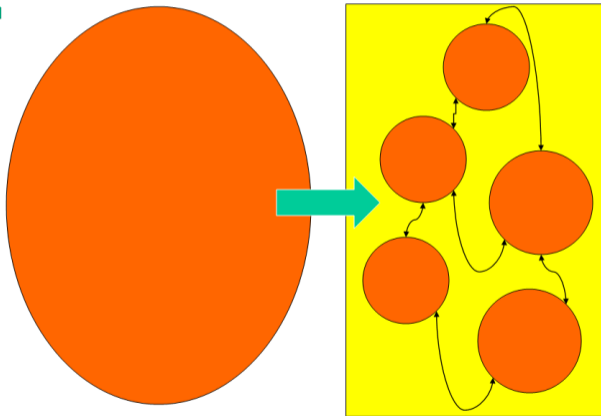
# Objekt-Orientierung



Contents

Definition

## Object-Oriented Programming



# Objekt-Orientierung - Daten- und "Echte" Typen



# Objekt-Orientierung - Typen

Links:

<https://www.youtube.com/watch?v=JBjjnqG0BP8>

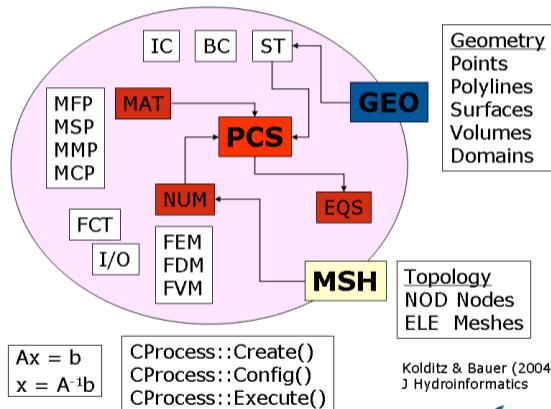
<http://www.stroustrup.com/>

[https://en.wikipedia.org/wiki/Bjarne\\_Stroustrup](https://en.wikipedia.org/wiki/Bjarne_Stroustrup)

# Objekt-Orientierung - OpenGeoSys (OGS)

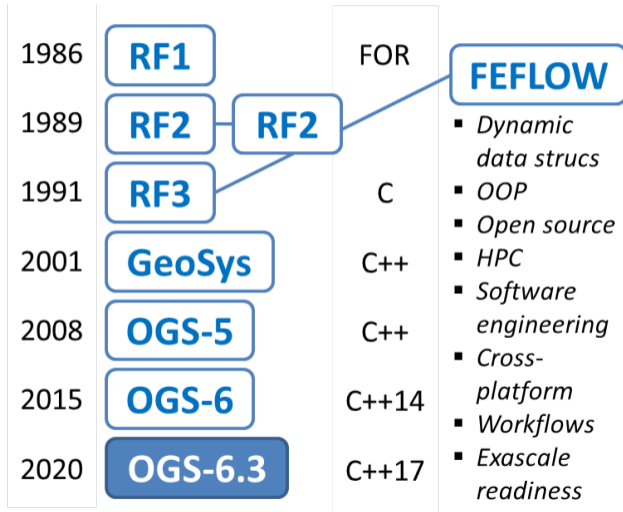


## V4: Object-Orientation: Multifield Problems



Source code reduction: 10 (V3) -> 3 MB (V4)

# Objekt-Orientierung - OGS - History





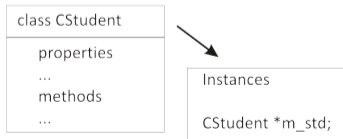
# Klassen

Das Sprachelement der Klassen sind das entscheidende Kriterium von objekt-orientierten Konzepten und die objekt-orientierte Programmierung (OOP). Klassen sind eine Art Schablone für einen benutzerdefinierten Datentypen. Darüber hinaus enthält die Klasse neben den Daten auch alle Methoden (Funktionen), um mit den Daten der Klasse operieren zu können. Unser Beispiel für Klassen, das uns im Verlaufe der Vorlesung beschäftigen wird, ist - wie könnte es anders sein - CStudent (Abbildung). Für die Konzipierung von Klassen spielt die Abstraktion der Daten einer Klasse eine besonders wichtige Rolle.



source:  
TU Dresden

Data abstraction ↓



# Daten-Abstraktion

Die Abbildung illustriert uns, dass eine Abstraktion von Daten (d.h. Eigenschaften) der Klasse Studenten eine durchaus vielschichtige Angelegenheit sein kann. Eine Aufstellung von Daten / Eigenschaften, die es aus ihrer Sicht zu berücksichtigen gilt, ist ihre nächste Hausaufgabe.

## C vs. C++

Der nächste Block zeigt Ihnen das Schema der Syntax der Klassen-Definition CStudent. Das Schlüsselwort für die Klassen-Definition ist `class`, der Name ist `CStudent`. Der Klassen-Rumpf ist in geschweifte Klammer eingebettet. Wichtig ist der Abschluss mit einem Semikolon. Wie bereits erwähnt, eine Klasse enthält Daten (Eigenschaften) und Methoden (Funktionen) auf den Daten. Prinzipiell geht diese Datenabstraktion auch mit anderen Sprachen wie C.

```
1 typedef struct
2 {
3     char* name_first;
4     char* name_last;
5     long matrikel_number;
6 } TDStudent;
7 TDStudent *student = NULL;
```

Listing 1: C - Type definition

```
1 class CStudent
2 {
3     data:
4     ...
5     methods:
6     ...
7 };
```

Listing 2: C++ - Class definition

## Daten-Abstraktion

Ein weiterer Vorzug von OO-Sprachen ist z.B. die Sichtbarkeit / Zugreifbarkeit von Daten zu regeln. Der nachfolgende Block zeigt das Datenschutz-Konzept von C++ (Sicherheitsstufen): Daten können öffentlich sein (public) oder gezielt für 'Freunde' verfügbar gemacht werden (protected) oder nur exklusiv für die eigene Klasse sichtbar zu sein (private).

```
1 class CStudent
2 {
3     private:
4     ...
5     protected:
6     ...
7     public:
8     ...
9 };
```

**Listing 3:** C++ - Data protection

# Klassen-Deklaration

Im vorangegangenen Abschnitt haben wir uns mit der Datenabstraktion mittels Klassen beschäftigt. So sollte konsequenterweise jede Klasse auch ihre eigenen - sorry - eigenen Quelldateien besitzen. Die Deklaration von Klassen erfolgt üblicherweise in einer sogenannten Header-Datei \*.h. Für die Methoden / Funktionen der Klasse ist eine \*.cpp Datei reserviert. Für uns bedeutet dies, zwei Dateien anlegen:

- ▶ student.h - die Deklaration der Klasse CStudent
- ▶ student.cpp - die Methoden der Klasse CStudent

# Klassen-Deklaration

Um mit der Klasse arbeiten zu können, müssen wir das entsprechende Header-File inkludieren. Dies erfolgt mit der Anweisung `#include "student.h"` am Anfang unseres Main-Files.

```
1 #include "student.h"
2 int main
3 {
4     return 0;
5 }
```

**Listing 4:** C++ - Declararion

# Instanzen einer Klasse

An dieser Stelle möchten wir unsere Eingangsgraphik erinnern. Instanzen sind Kopien einer Klasse mit denen wir arbeiten können, das heißt diese bekommen echten Speicher für ihre Daten (die natürlich für jede Instanz einer Klasse unterschiedlich sein können).



# Instanzen einer Klasse

Es gibt zwei Möglichkeiten, Instanzen einer Klasse zu erzeugen:

```
1 #include "student.h"
2 void main()
3 {
4     // Creating an instances of a class - 1
5     CStudent m_std_A;
6     // Creating an instances of a class - 2
7     CStudent *m_std_B;
8 }
```

**Listing 5:** Creating instances

# Instanzen einer Klasse

Der direkte und der mittels eines sogenannten Zeigers (hierfür gibt ein Extra-Kapitel). Wir werden sehen, dass der zweite Weg oft der bessere ist, da wir z.B. die Initialisierung und das Speichermanagement für unsere Daten selber in die Hand nehmen können. Dies können wir mittels sogenannter Konstruktoren und Destruktoren erledigen. Damit beschäftigen wir uns im nächsten Abschnitt.

# Instanzen einer Klasse: EX05a

```
1 #include "student.h"
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     CStudent *m_std_cpp; // pointer to an instance
7     cout << "E41: Instances of classes" << endl;
8     cout << "What have we created?\t : " << m_std << endl;
9     cout << "What size has it?\t : " << sizeof(m_std) << endl;
10    TDStudent *m_std_c; // pointer to TD
11    return 0;
12 }
```

**Listing 6:** Exercise: Creating instances

# Übungen: Übersicht

- ▶ EX05a: Instanzen einer Klasse - Kreieren
- ▶ EX05b: Instanzen einer Klasse - Speicher
- ▶ EX05c: Eigenschaften einer Klasse
- ▶ EX05d: Weitere Eigenschaften einer Klasse
- ▶ EX05e: Aktion "Gummibärchen" (Datenbank häcken) ...