Informatik I

Uebung 10

Gruppe A, ETZ E91

Patrick Boenzli

dev.orxonox.net/wiki/boenzlip/inf1

Lernziele

Die Studenten:

- wissen was **objektorientiertes Programmieren** ist;
- unterscheiden: **objektorientiertes** und **prozedurales** Programmieren;
- wissen wie man **Klassen** und **Objekte** erstellt.

Object Oriented Programming - OOP

a) Begriffserklaerung:

- Prinzip wie man ein Problem abstrahieren kann;
- Spezifikation von Zugriffsschnittstellen (Interfaces);
- Verstecken von unwichtigen Daten/Funktionen (Data Hiding);
- Flexibilitaet und Erweiterbarkeit;
- Standard Programmierstyl fuer grosse Projekte.

Object Oriented Programming - OOP

b) OOP vs. Prozedurales Programmieren

OOP

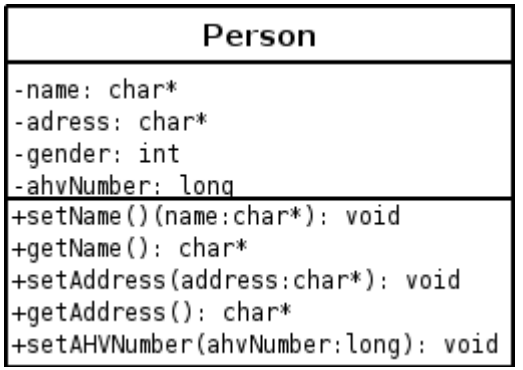
- collection of objects;
- modulariy;
- data structured in objects.

Procedural Propgramming

- collection of functions;
- a bunch of code;
- unstructured data.

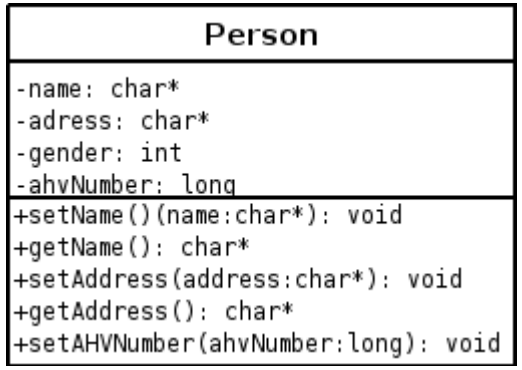
Object Oriented Programming - OOP

c) Beispiel: Class Person



Object Oriented Programming - OOP

d) Beispiel: Implementierung Class Person




```
// header file: person.h
class Person {
public:
  Person();
  ~Person();

  void  setName(char* name);
  char* getName();

  void  setAddress(char* address);
  char* getAddress();

  // more functions

private:
  char* name;
  char* address;
};
```

Konstruktor

Dekonstruktor

Member functions

Member (variables)

Object Oriented Programming - OOP

e) Beispiel: Implementierung Class Person

```
// source file: person.cpp
#include "person.h"

Person::Person(char* name) {
  this->name = name;
  this->address = NULL;
}
Person::~Person(){}

void Person::setName(char* name) {
  this->name = name);
}

char* Person::getName() {
  return this->name;
{
```

Konstruktor

Dekonstruktor

Member functions

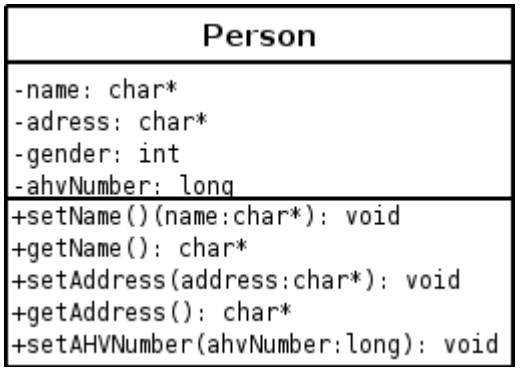
Object Oriented Programming - OOP

f) Beispiel: Implementierung Class Person

```
// Anwendung

#include "person.h"


// Variante 1
Person p("Albert");

// Variante 2
Person p1 = Person("Albert");

// Variante 3
Person* p2 = new Person("Albert");

// functions:
char* name = p.getName();
```

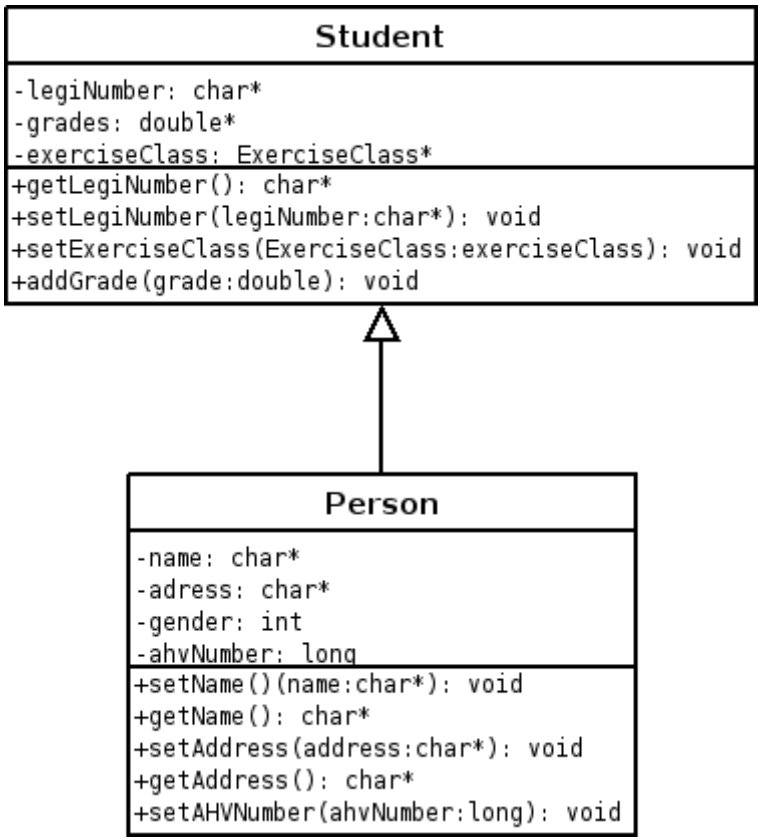
Object Oriented Programming - OOP

g) Beispiel: Class Person



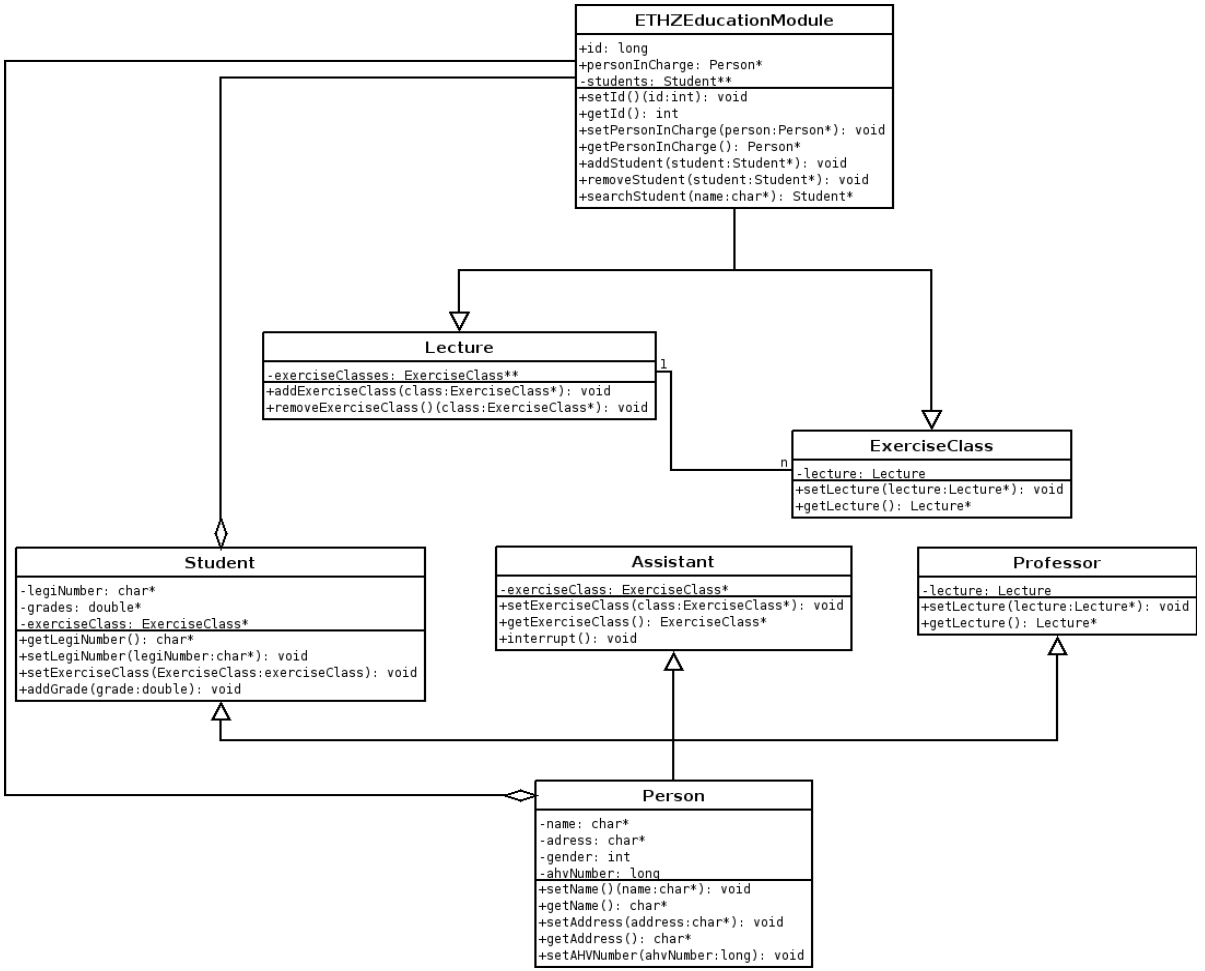
Object Oriented Programming - OOP

h) Beispiel: Class Student Vererbt



Object Oriented Programming - OOP

i) Beispiel: ETHZ Framework :)



Aufgabe 1 – Complex Number

a) Aufgabe verstehen

1) Schreibe eine Klasse fuer komplexe Zahlen

2) Membervariablen: Imaginar- und Realteil

3) Memberfunktionen: `add, sub, mult, div, abs, print`

```
// header file: complex.h

class Complex {
public:
  Complex();

  ...
  ...
};
```

Aufgabe 2 - Operators

a) Aufgabe verstehen

Operatoren ueberladen:

```
// header file: complex.h

class Complex {
   ...
   Complex operator+(Complex& c);
   ...
};
```

```
// source file: complex.cpp

Complex Complex::operator+(Complex& c)
{
  ...
}
```

Aufgabe 3 – UPN Calculator

a) Aufgabe verstehen

1) Schreibe den UPN Calculator so um, dass er komplexe Zahlen versteht
2) Schreibe ListStack so um, dass es komplexe Zahlen speichern kann
3) Achtung: Persistenz von Daten