Informatik I

Uebung 11

Gruppe A, ETZ E91

Patrick Boenzli

dev.orxonox.net/wiki/boenzlip/inf1

Lernziele

Die Studenten:

- lernen was **Vererbung** in OOP ist;
- wissen wie man **static** und **dynamic binding** definiert und verwendet

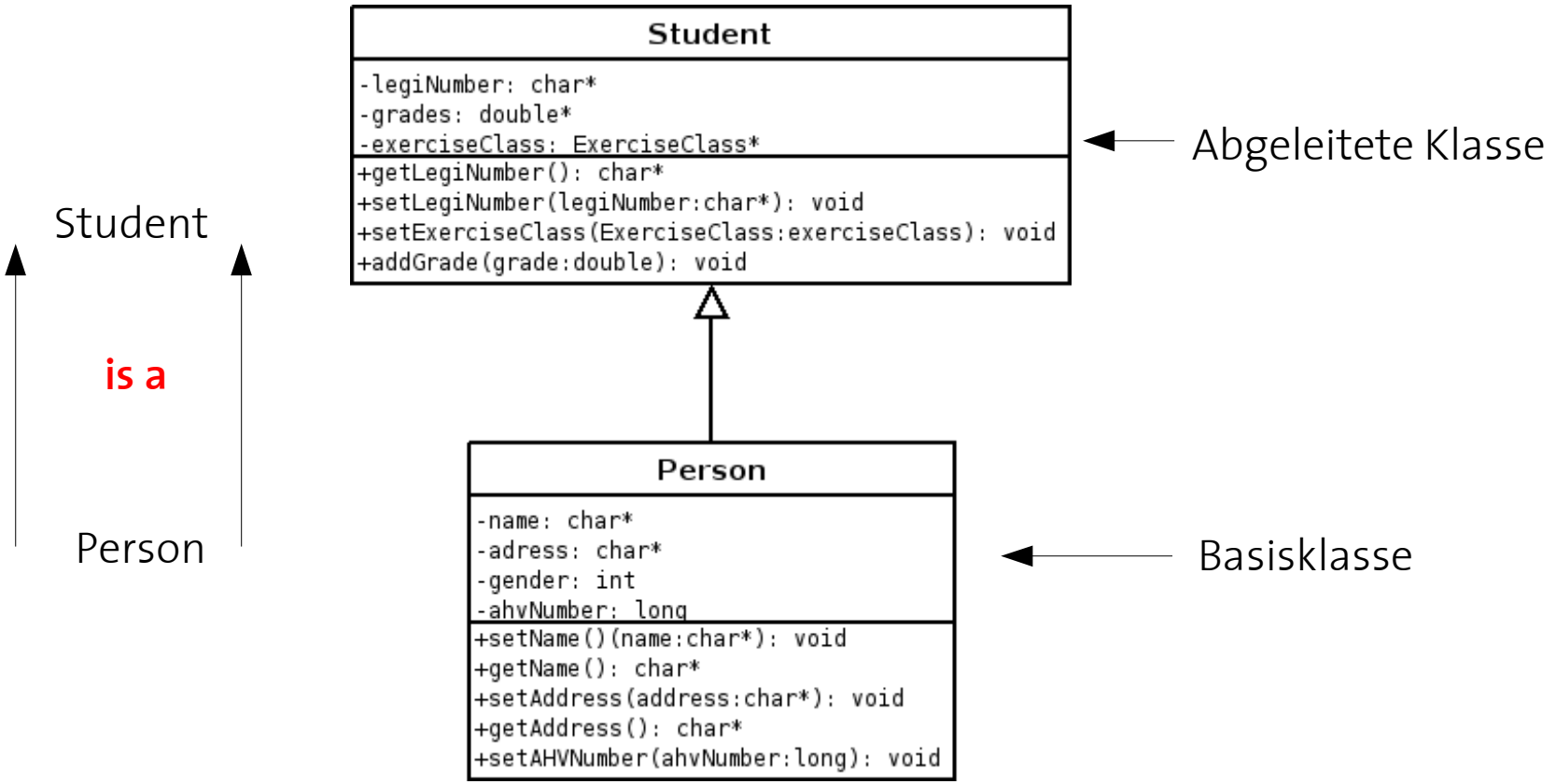
Object Oriented Programming - OOP

a) Vererbung I

- erlaubt es einer Klasse zusaetzliche **Funktionalitaet** zu bestimmen;
- erlaubt es einer Klasse zusaetzliche **Daten** zu verwenden;
- kann **Funktionen veraendern**.

Object Oriented Programming - OOP

b) Beispiel



Object Oriented Programming - OOP

c) Beispiel

Vererbung in C++

```
// header file: person.h

class Person {
public:
  Person(char* name);
  ~Person();


private:
  char* name;
};
```

```
// header file: student.h

#include "person.h"
class Student: public Person {
public:
  Student(char* name,
          char* legiNr);
  ~Student();

private:
  char* legiNr;
};
```

Object Oriented Programming - OOP

c) Beispiel

Aufruf Konstruktor der Basisklasse

```
// source file: person.cpp

#include "person.h"

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

```
// source file: student.cpp

#include "student.h"

Student::Student(char* name, char* legiNr) : Person(name)
{
   this->legiNr = legiNr;
}
```

Object Oriented Programming - OOP

d) Static vs. Dynamic Binding

Gleichnamige Funktionen

```
// header file: person.h

class Person {
public:
  Person(char* name);
  ~Person();

  void print();

private:
  char* name;
};
```

```
// header file: student.h

#include "person.h"
class Student: public Person {
public:
  Student(char* name,
          char* legiNr);
  ~Student();

  void print();

private:
  char* legiNr;
};
```

Object Oriented Programming - OOP

e) Static vs. Dynamic Binding I

Gleichnamige Funktionen

```
// header file: person.h

class Person {
public:
  Person(char* name);
  ~Person();

  void print();

private:
  char* name;
};
```

```
// header file: student.h

#include "person.h"
class Student: public Person {
public:
  Student(char* name,
          char* legiNr);
  ~Student();

  void print();

private:
  char* legiNr;
};
```

Object Oriented Programming - OOP

e) Static vs. Dynamic Binding II

Welche Funktion wir auferufen:

```
// srouce file: some_source.cpp

#include "student.h"

Person  p("albert");
Student s("Spiderman", "123-456-78");

p.print();
s.print();

Person*  p2 = new Person("albert");
Person*  s2 = new Student(...);

p2->print();
s2->print();
```

Person::print()	Student::print()
✿	
✿	
✿	
✿	

Object Oriented Programming - OOP

e) Static vs. Dynamic Binding III

Gleichnamige virtual Funktionen

```
// header file: person.h

class Person {
public:
  Person(char* name);
  ~Person();

  virtual void print();

private:
  char* name;
};
```

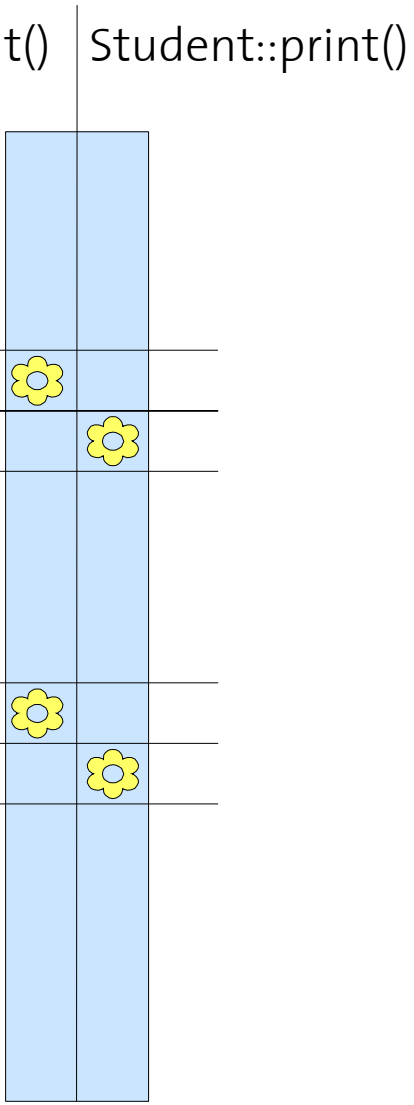
```
// header file: student.h

#include "person.h"
class Student: public Person {
public:
  Student(char* name,
          char* legiNr);
  ~Student();

  virtual void print();

private:
  char* legiNr;
};
```

Object Oriented Programming - OOP

e) Static vs. Dynamic Binding II

Welche Funktion wir auferufen:

```
// srouce file: some_source.cpp

#include "student.h"

Person  p("albert");
Student s("Spiderman", "123-456-78");

p.print();
s.print();

Person*  p2 = new Person("albert");
Person*  s2 = new Student(...);

p2->print();
s2->print();
```



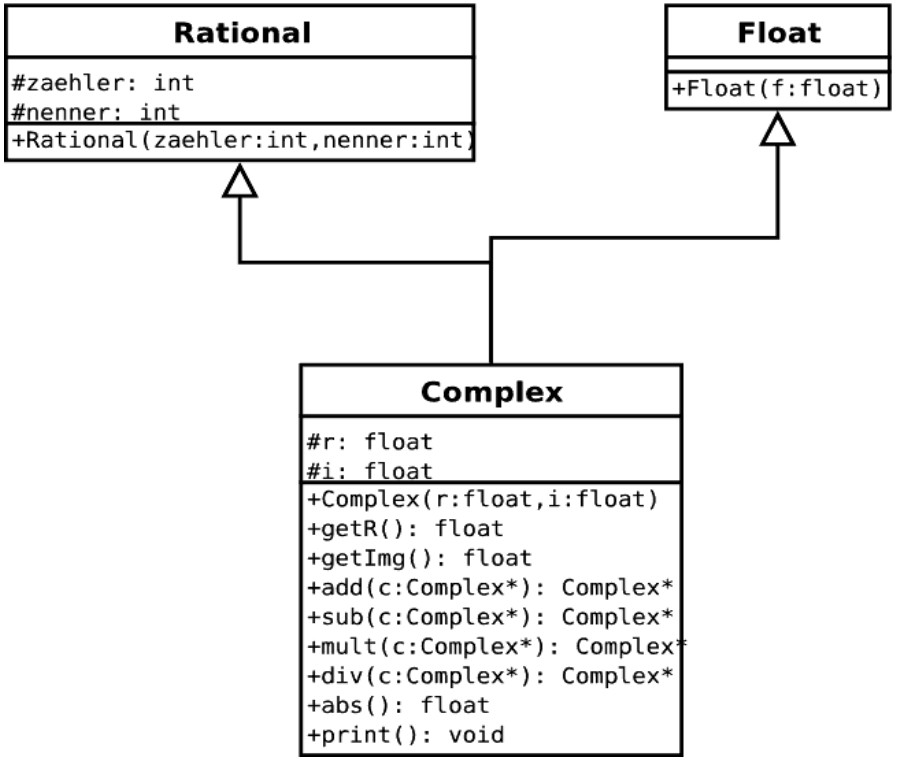
Aufgabe 1 – UPNCalc Revisited

a) Aufgabe verstehen

1) Schreibe eine Klasse fuer rationale Zahlen und Gleitkommazahlen

2) Die Klassen erben von der Complex Klasse aus der Uebung 10

3) UPNCalc wird den rest machen



Aufgabe 2 - Polymorphismus

a) Aufgabe verstehen

Polymorphismus bedeutet Mehrfachvererbung. Eine Klasse kann also von mehreren anderen Klassen erben.