C++: Vererbung und Polymorphie

Carsten Gips (HSBI)

Vererbung: “IS-A”-Beziehung zw. Klassen

```
class Student : public Person { ... }
```

Vererbung: “IS-A”-Beziehung zw. Klassen

```
class Student : public Person { ... }
```

```
Student(const string &name = "", double c = 0.0)
: Person(name), credits(c) { }

Student(const Student &s)
: Person(s), credits(s.credits) { }
```

Polymorphie: Was passiert im folgenden Beispiel?

```
class Person { ... }
class Student : public Person { ... }

Student s("Heinz", "heizer");
Person &p = s;

cout << s.toString() << endl;
cout << p.toString() << endl;
```

Konsole: polyStat.cpp

Polymorphie: statisch und dynamisch

- C++ entscheidet zur **Kompilierzeit**, welche Methode aufgerufen wird
  - `p` ist vom Typ `Person` => `p.toString()` => `Person::toString()`
  - Dieses Verhalten wird **statisches Binden** genannt.

- Von Java her bekannt: **dynamisches Binden**
  - Typ eines Objektes wird zur **Laufzeit** ausgewertet

Dynamisches Binden geht auch in C++ ...

1. Methoden in **Basisklasse** als **virtuelle Funktion** deklarieren
   => Schlüsselwort `virtual`
2. Virtuelle Methoden in Subklasse normal überschreiben (gleiche Signatur)
3. Objekte mittels Basisklassen-Referenzen bzw. -Pointer zugreifen (siehe nächste Folie)

```
class Person {
    virtual string toString() const { ... }
};
```

Konsole: polyDyn.cpp

Vorsicht Slicing

```
Student s("Heinz", 10.0);
Person p("Holger");

p = s;
cout << "Objekt s (Student): " << s.toString() << endl;
cout << "Objekt p (Person):  " << p.toString() << endl;
```

Konsole polySlicing.cpp

Vorsicht Slicing

```
Student s("Heinz", 10.0);
Person p("Holger");

p = s;
cout << "Objekt s (Student): " << s.toString() << endl;
cout << "Objekt p (Person):  " << p.toString() << endl;
```

Konsole polySlicing.cpp

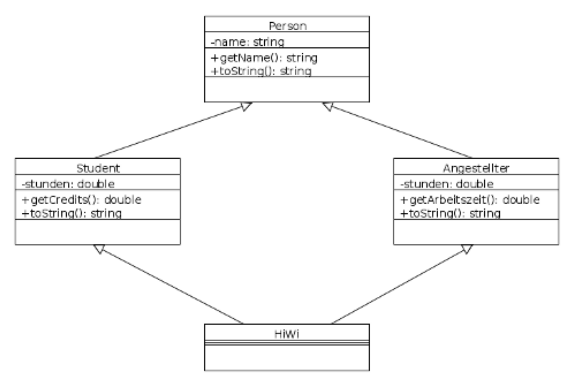
=> **Dyn. Polymorphie** in C++ immer über **Referenzen** (bzw. Pointer) **und virtuelle Methoden**

Mehrfachvererbung in C++

```
class HiWi: public Student, public Angestellter {...};
```

Mehrfachvererbung in C++

```
class HiWi: public Student, public Angestellter {...};
```



Hinweis Speicherlayout ...

Mehrfachvererbung in C++: Virtuelle Basisklassen

```
class Angestellter: virtual public Person {...};
class Student: virtual public Person {...};

class HiWi: public Student, public Angestellter {...};
```

- `Person` ist jetzt eine virtuelle Basisklasse
- Auswirkungen erst in Klasse `HiWi`
- Dadurch sind gemeinsam genutzte Anteile nur einfach vorhanden

Konsole vererbungMultiVirtual.cpp

Wrap-Up

- `Subklasse : public Superklasse`
- Keine gemeinsame Oberklasse wie `Object`, kein `super`
- Verkettung von Operatoren und *struktoren
- Abstrakte Klassen in C++

- Statische und dynamische Polymorphie in C++
  - Methoden in Basisklasse als `virtual` deklarieren
  - Dyn. Polymorphie nur mittels Pointer/Referenzen
  - Slicing in C++ (bei Call-by-Value)

- Konzept der Mehrfachvererbung
- Problem bei rautenförmiger Vererbungsbeziehung: Attribute und Methoden mehrfach vorhanden
- Virtuelle Basisklassen: Gemeinsam genutzte Attribute nur noch einfach vorhanden

LICENSE