Beziehungen zwischen Objekten

Florian Adamsky, B. Sc. (PhD cand.)

florian.adamsky@iem.thm.de
http://florian.adamsky.it/

Softwareentwicklung im WS 2014/15

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Outline

Inhaltsverzeichnis

Einführung

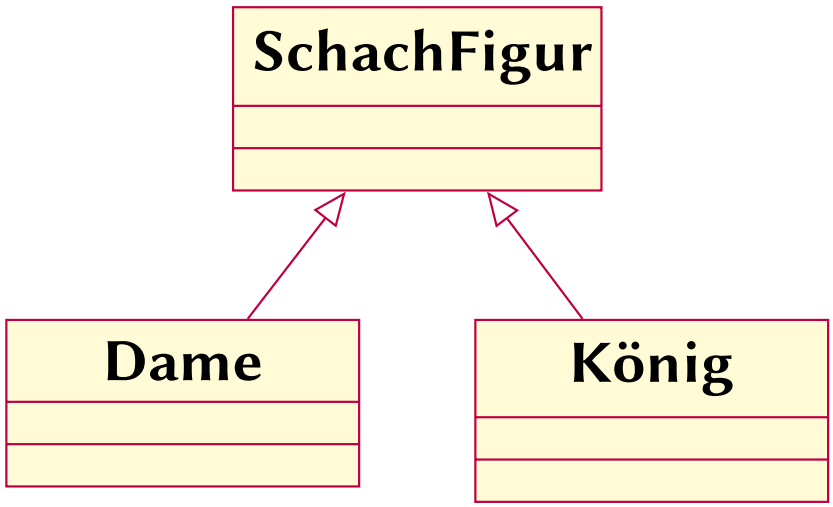
- Vererbung bildet in der OOP eine *ist-ein*-Beziehung ab
  - Der Mensch *ist ein* Säugetier
- Systematische Spezialisierung von oben nach unten
- Die *Unterklasse* erbt damit alle Merkmale der *Oberklasse*

Was wird vererbt und was nicht?

- Unterklassen erben alle zugreifbaren Member der Basisklasse:
  - Konstruktor und Destruktor
  - Attribute / Klassenvariablen
  - Methoden und Operatoren
- Was wird nicht vererbt?
  - `friend` Funktionen
  - Member die als `private` deklariert sind

Beispiel Vererbung

Example (UML)



Example (Quelltext)

```
class SchachFigur {
};

class Dame  : public SchachFigur {
};

class König : public SchachFigur {
};
```

Inhaltsverzeichnis

Komposition

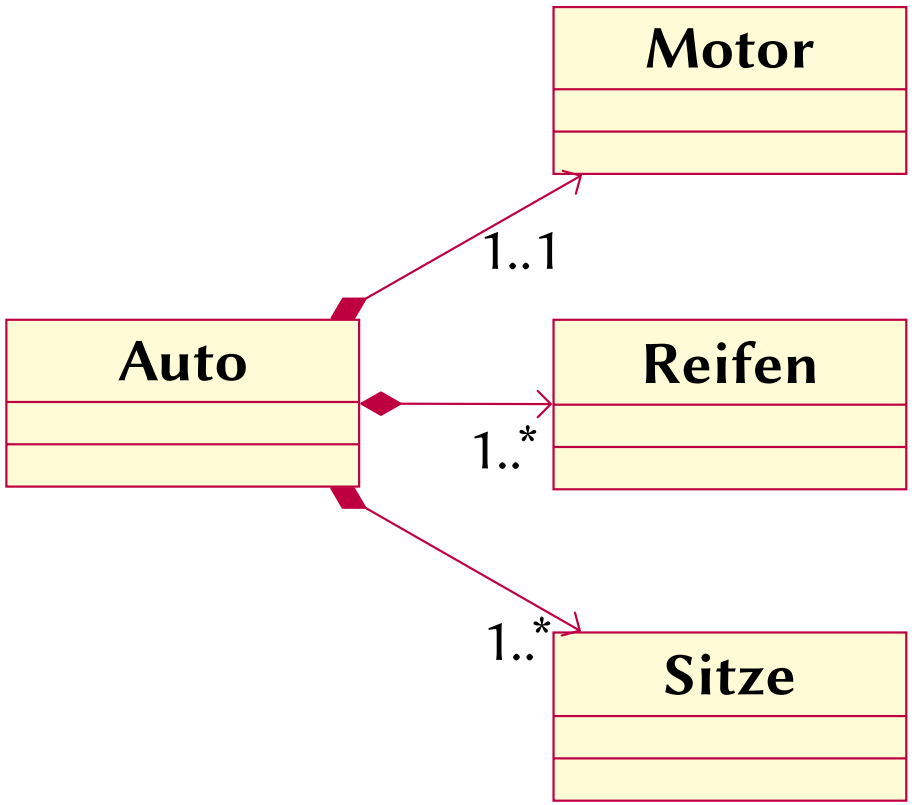
- reale und komplexe Objekte bestehen meist aus kleinen und einfachen Objekte

  Auto besteht aus Reifen, Motor, Sitzen, . . .
  PC besteht aus CPU, Motherboard, RAM, . . .

- im objektorientierten Paradigma nennt man diese Beziehung: Komposition
- bildet eine *besteht aus* oder *hat ein* Beziehung ab
  - Komposition wird in C++ mit Attributen umgesetzt

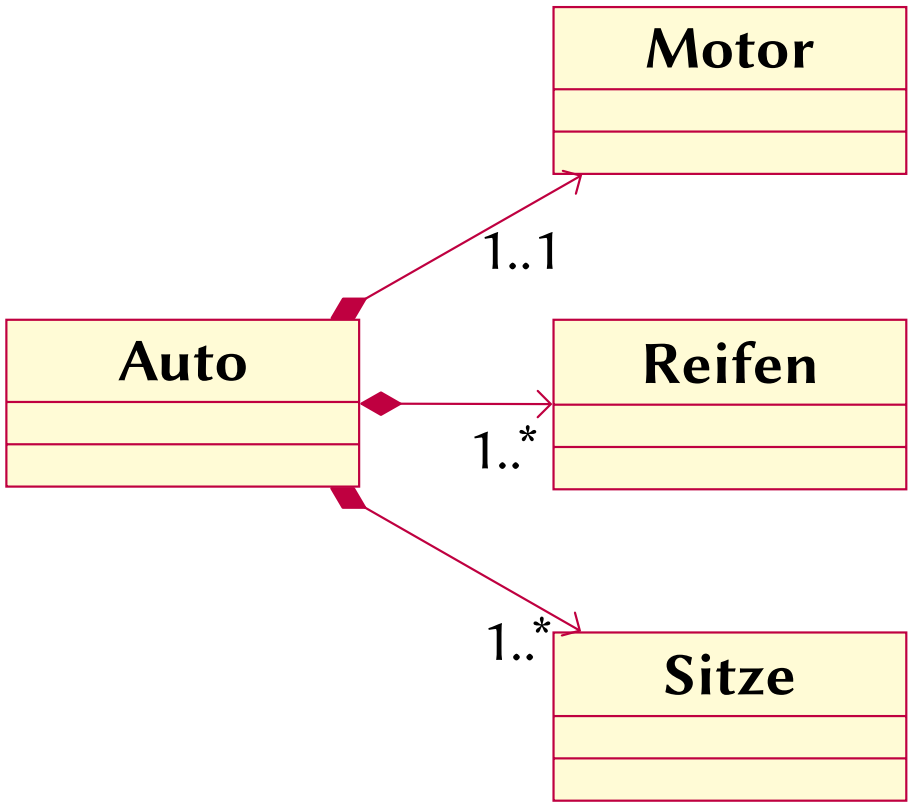
Beispiel Komposition in C++

Example (UML)



Beispiel Komposition in C++

Example (UML)



Example (Auto)

```
#include "Reifen.h"
#include "Benzinmotor.h"
#include "Sitze.h"

class Auto {
private:
    Reifen      reifen[4];
    Benzinmotor motor;
    Sitze       sitze[4];
};
```

Besonderheiten der Komposition

- Oberklasse *besitzt* die Subklassen
  - entscheidet über deren Leben und Tod
- Subklassen werden erstellt wenn die Oberklasse erstellt wird
- Subklassen werden gelöscht wenn die Oberklasse gelöscht wird

Implementierung der Komposition in C++

Example (Stack)

```
class Auto {
private:
    Reifen      reifen[4];
    Benzinmotor motor;
    Sitze       sitze[4];
};
```

Example (Heap)

```
class Auto {
private:
    Motor*  m;
    Reifen* reifen[4];

public:
    Auto() {
        Motor* m = new Motor();
        for (int i = 0; i < 4; i++)
            reifen[i] = new Reifen();
    }

    ~Auto() {
        delete motor;
        delete[] reifen;
    }
};
```

Vorteile der Komposition

Example (Vorteile)

- jede Klasse kann klein gehalten werden
- Subklassen können wieder verwendet werden
  - z.B. Fahrrad braucht auch Reifen
- Komplexität verteilt sich dadurch auf verschiedenene Klassen, statt auf einer einzigen

Nachteile

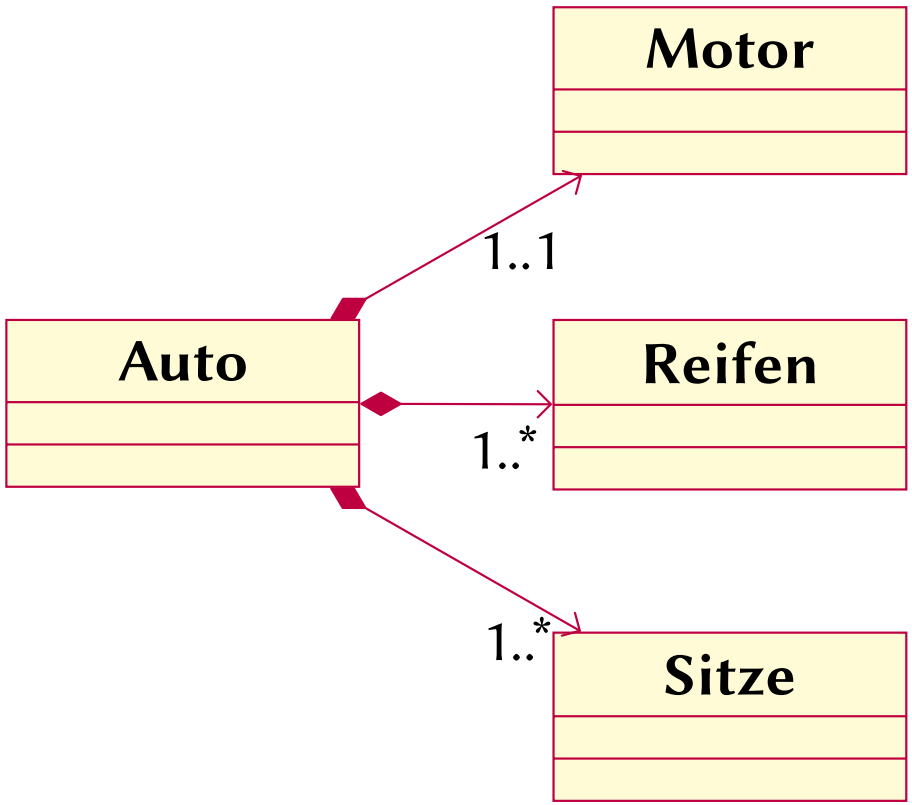
- Oberklassen sind schlechter wiederverwendbar

Aggregation

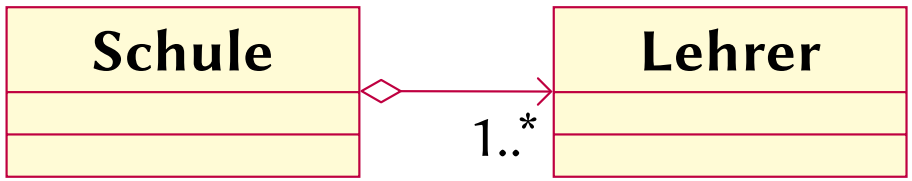
- Aggregation ist eine spezielle Form der Komposition
- bildet auch eine *besteht aus* oder *hat ein* Beziehung ab
- Oberklasse hat jedoch keine Besitzansprüche
  - Subklassen leben weiter und werden nicht zerstört wenn die Oberklasse zerstört wird
  - Subklassen werden auch nicht automatisch erstellt wenn die Oberklasse erstellt wird

Unterschied Aggregation/Komposition

Example (Komposition)



Example (Aggregation)



Beispiel Schule

Example (Klasse)

```
class Lehrer {
private:
    string name;
public:
    Lehrer(string name) : name(name){}
};

class Schule {
private:
    Lehrer* lehrer;
public:
    Schule(Lehrer* m) : lehrer(m){}
};
```

Example (Main)

```
int main(void) {
    Lehrer* bob = new Lehrer("Bob");

    {
        Schule schule(bob);
    }

    // Lehrer existiert immer noch obwohl
    // die Schule schon zerstört ist
    delete bob;
}
```

Inhaltsverzeichnis

Assoziation

- lose Verbindung zwischen Objekten
- Objekte kennen sich
- Umsetzung in C++ über Argumente in Methoden

Beispiel Assoziation in C++

Example (UML)

Beispiel Assoziation in C++

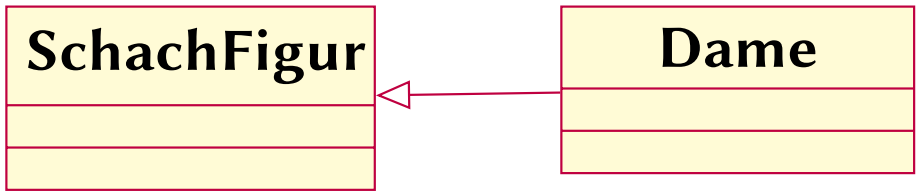
Example (UML)

Example (Kreis)

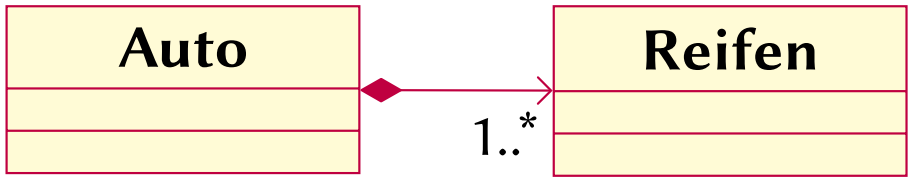
```
class Kreis {
public:
    setzeMittelpunkt(Punkt p);
};
```

Zusammenfassung

Vererbung



Komposition



Assoziation

Aggregation