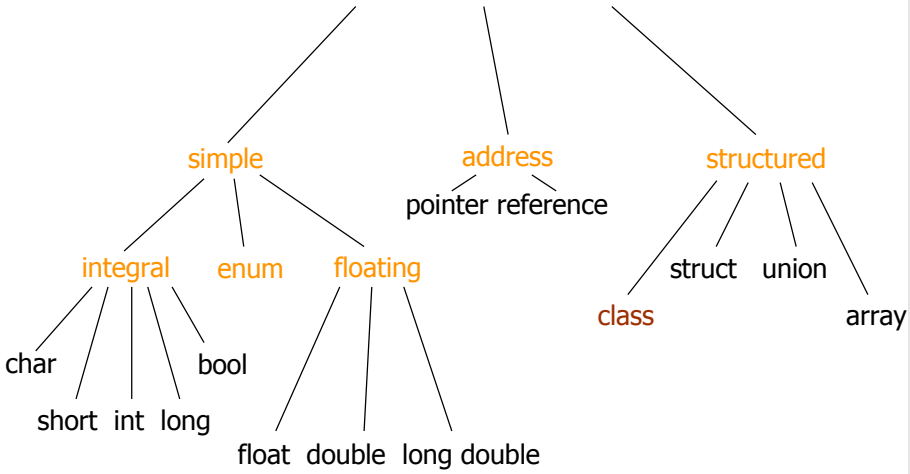



Grundlagen der Programmierung in C
Klassen

Wintersemester 2005/2006
G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de


Das C++ Typsystem



Task: Shoot Yourself in The Foot

The proliferation of modern programming languages (all of which seem to have stolen countless features from one another) sometimes makes it difficult to remember what language you're currently using. This handy reference is offered as a public service to help programmers who find themselves in such a dilemma.

C:

You shoot yourself in the foot.

C++:

Makes it harder, but when you do, you lose your whole leg.

Die Sicht der Dinge

- Programmieren = Modellieren (u.a.)
  - System, bestehend aus Daten und Funktionen
  - Abläufe festlegen
  - Ausführen = Simulieren des Modells
- Bisher: Funktionsbezogene Sicht auf das Gesamtsystem
  - Welche Funktionen werden gebraucht?
  - Welche Daten gehen rein, welche kommen raus?
- Besser: Objektbezogene Sicht
  - Aus welchen Objekten besteht das Gesamtsystem?
  - Welche Fähigkeiten haben diese?
  - Welche internen Zustände (Daten) haben die Objekte?

Vorteile des OO Software-Designs

- Direktere Nachbildung (= Modellierung) der realen Welt
- Datenstrukturen ändern sich seltener als Algorithmen
- Unterstützt besser "Information Hiding" ("encapsulation")
  - → Bessere Wartbarkeit
  - → Leichtere Wiederverwendbarkeit ("code reuse")

Datenstrukturen → Objekte / Klassen

- Bestandteile eines Objektes:
  - Art
  - Attribute / Zustand
  - Fähigkeiten
- Beispiel: Mülleimer
  - Art = Mülleimer
  - Attribute
    - Farbe, Geruch,
    - Inhalt,
    - Raum, in dem er steht
  - Fähigkeiten
    - sich erzeugen
    - Objekt zu Inhalt dazufügen (wegwerfen)
    - Queries (ist der Deckel offen? Ist er voll? …)
    - Sich leeren

- Jedes Objekt separat zu modellieren ist unpraktisch
  - Es gibt zu viele
  - Gemeinsamkeiten werden nicht ausgenutzt
- Lösung: Objekte nach "Art" klassifizieren
  - Pro "Art" eine *Klasse* (Bsp.: "Mülleimer-Klasse")
  - Konkrete Objekte sind *Instanzen* einer Klasse
- Konsequenz:
  - Alle Instanzen einer Klasse haben dieselben Fähigkeiten
  - Haben dieselbe Menge von Attributen (Variablen)
  - Attribute können verschieden belegt sein

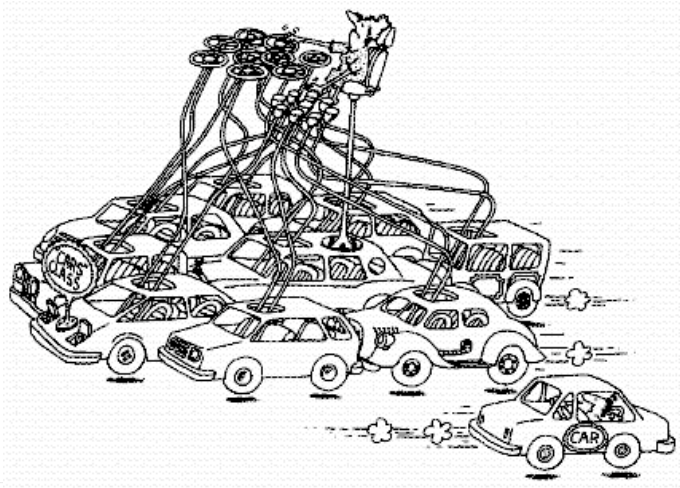
Stempel (Klasse)     Abdrücke (Instanzen)

Klassen

- Implementiert …
  - Fähigkeiten als *Instanzmethoden* (= Funktionen)
  - Attribute als *Instanzvariablen*
- Klasse selbst belegt keinen Speicher! (nur Instanz)
- Ist vielmehr ein *benutzer-definierter Typ*
- Wird manchmal auch *abstrakter Datentyp (ADT)* genannt
- In C++ keine vordefinierten Klassen (Gegensatz zu Java)

Abstraktion

- Fazit: Eine Klasse ist eine Form der Abstraktion



Deklaration

```
class Auto
{
public:
    int starten( void );
    const Person* getFahrer( void );
    enum ColorE { Black, Red, ... };
private:
    ColorE m_farbe;
    Person m_personen[5];
};
```

Instanzmethoden und Typen (=Interface)

Instanzvariablen (=Attribute)

- *Member* = Instanzvariablen + -methoden

Visibility

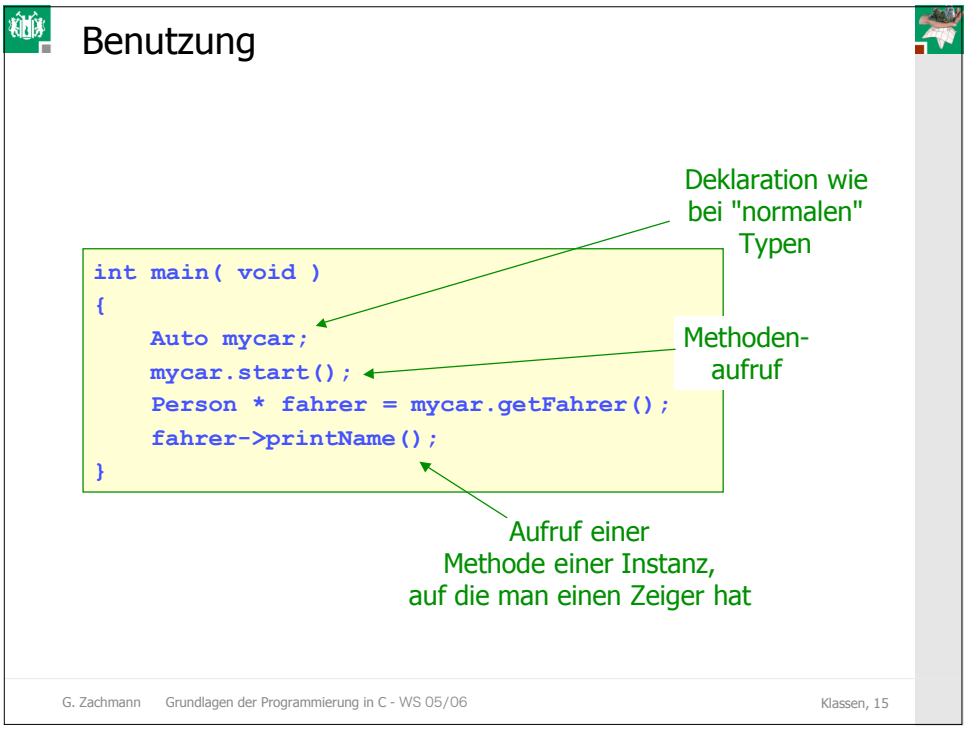
- Public:
  alle Members ab **`public:`** sind nach *außen* sichtbar,
  bis zum nächsten **`private:`**
- Private:
  alle Members ab **`private:`** sind nur in Methoden *innerhalb* der Klasse sichtbar.
- Protected:
  kommt später

Definition

```
int Auto::starten( void )
{
    ...
}

const Person* Auto::getFahrer( void )
{
    ...
}
```

Scope Resolution Operator


Benutzung
Deklaration wie bei "normalen" Typen
int main(void)
{
Auto mycar;
mycar.start();
Person * fahrer = mycar.getFahrer();
fahrer->printName();
}
Methoden-aufruf
Aufruf einer Methode einer Instanz, auf die man einen Zeiger hat
G. Zachmann Grundlagen der Programmierung in C - WS 05/06
Klassen, 15