Hochschule Esslingen
University of Applied Sciences

Objektorientierte Programmierung mit C++

Einführung in die Objektorientierung

Profs. M. Dausmann, D. Schoop, A. Rößler

Fakultät Informationstechnik, Hochschule Esslingen

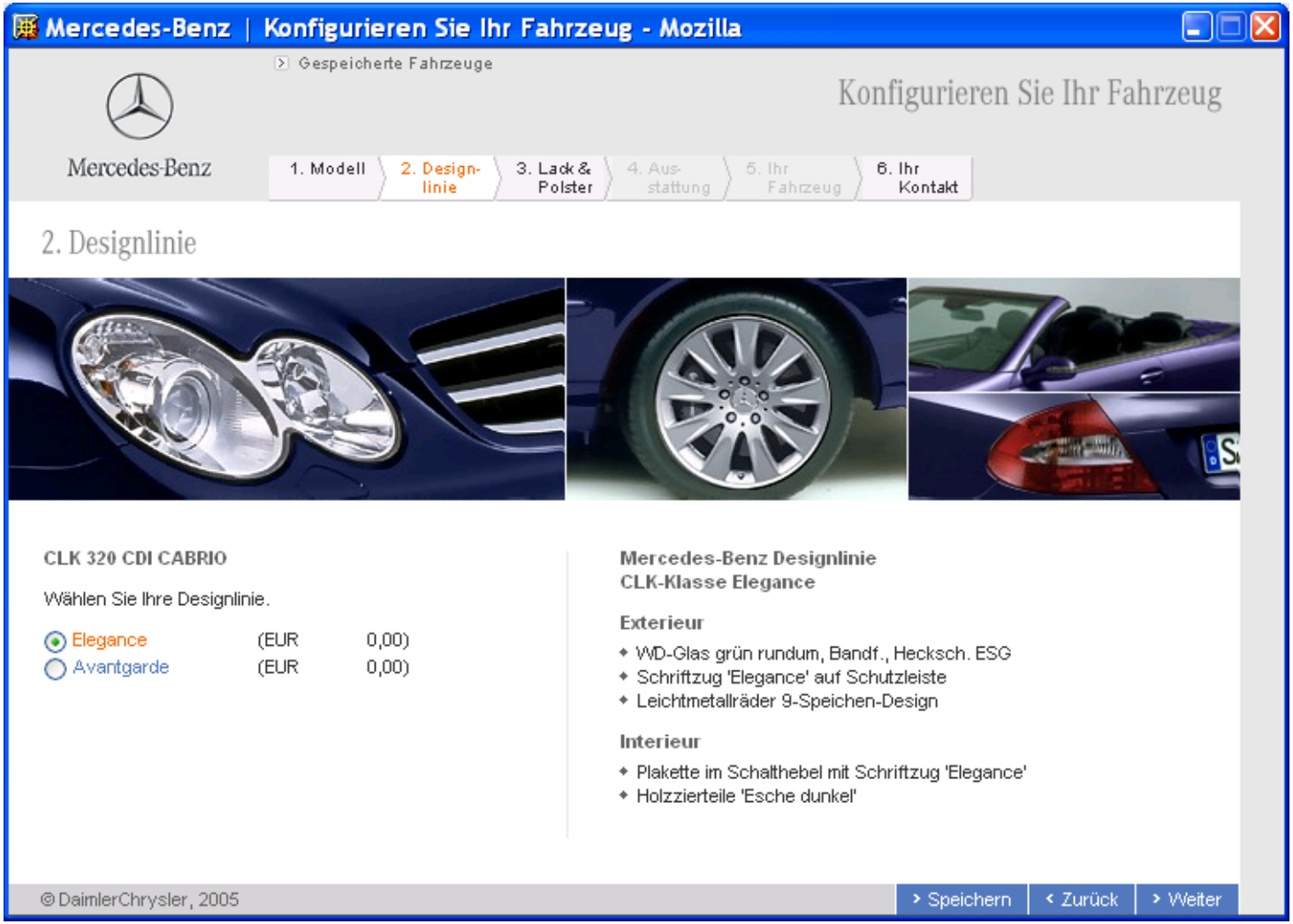
Agenda

- Motivation und Einführung
- Beispielszenario
- Eigenschaften des objektorientierten Ansatzes
- Konzepte des objektorientierten Programmierens

Beispielszenario: Fahrzeugkonfigurator

Quelle: http://www.mercedes-benz.de/content/germany/mpc/mpc_germany_website/de/home_mpc/passenger_cars.html

Beispielszenario: Fahrzeugkonfigurator



Quelle: http://www.mercedes-benz.de/content/germany/mpc/mpc_germany_website/de/home_mpc/passenger_cars.html

Beispielszenario: Fahrzeugkonfigurator

- Die gewählten Fahrzeugmerkmale sollen durch das Konfigurationsprogramm bearbeitet und gespeichert werden können.
- KFZ-Daten (Beispiel):
  - Typ — Mercedes CLK 320 CDI Cabrio
  - Motorleistung — 165 kW (224 PS)
  - Sitzplatzanzahl — 4
  - Farbe — silber metallisch
  - …

Fahrzeugkonfigurator in C

- Daten als Struktur:

```
struct kfz {
    char typ[20];
    int motorleistung;
    int sitzplatzzahl;
    char farbe[15];
};
```

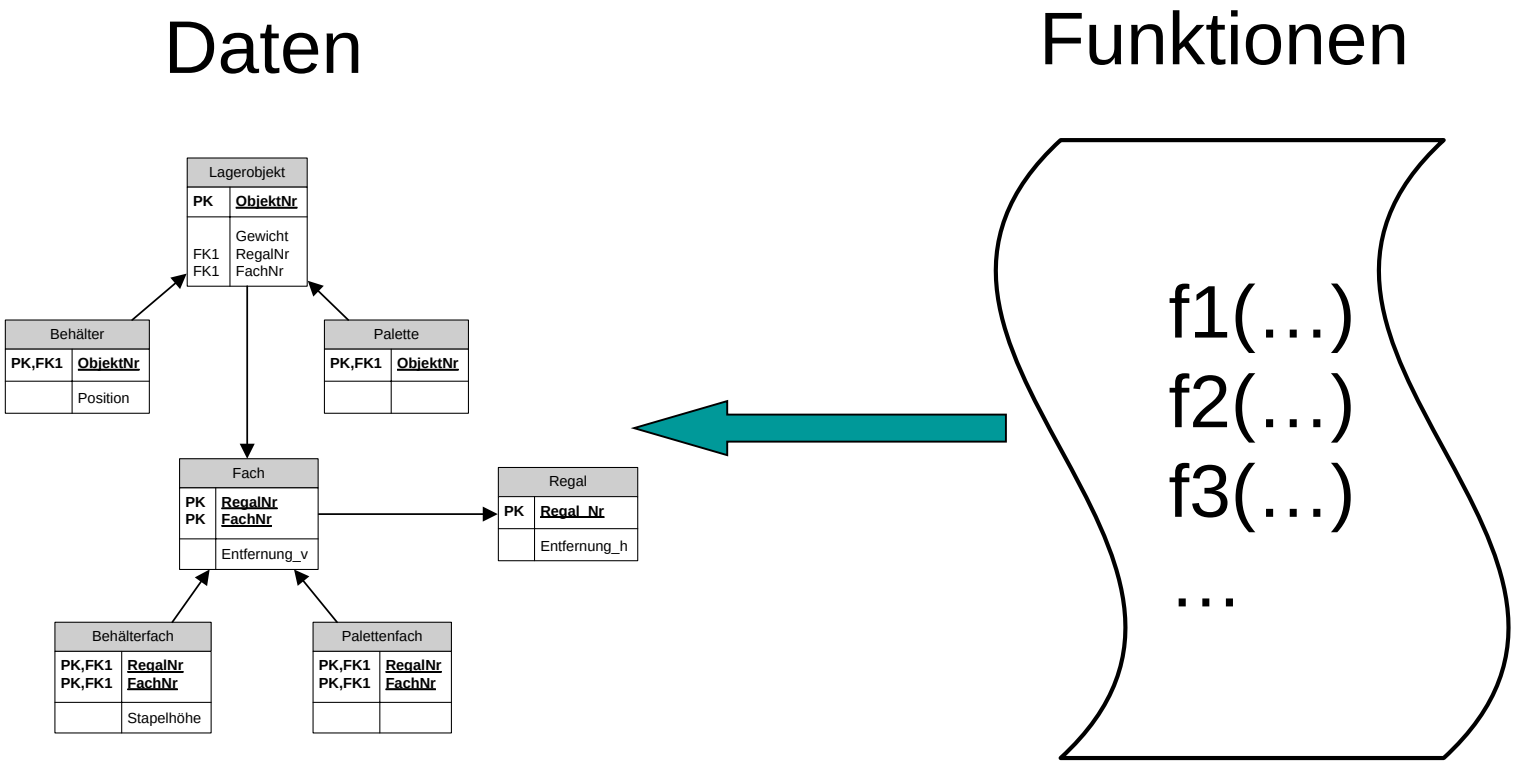
- Veränderung der Daten über eigenständige Funktionen:

```
void setTyp(struct kfz * k, char t[]){
   strncpy(k->typ, t, 20);}
```

- Verwendung in **`main()`**:

```
struct kfz o1; setTyp(&o1, "Mercedes CLK ... ");
```

Trennung von Daten und Funktionen (in C)



Eigenschaften des OO-Ansatzes

- Kapselung

  Objekte kapseln die dazugehörigen Eigenschaften gegenüber der Umwelt und bieten **Schnittstellen** dazu an.

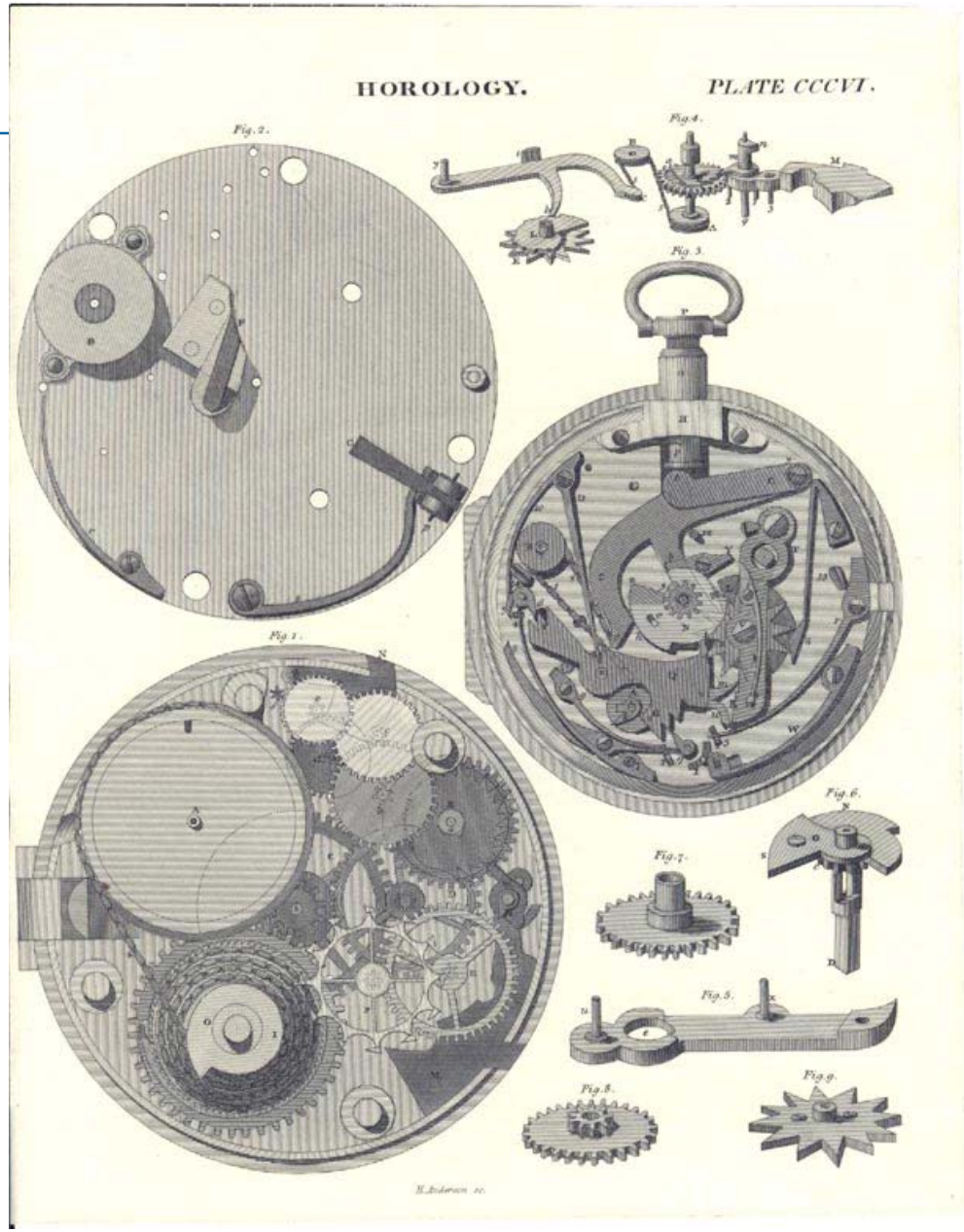
- Identität

  Alle Objekte besitzen eine eigene Identität,
  auch wenn sie nach außen gleich erscheinen.

- Klassifikation / Abstraktion

  Ähnliche Objekte können zu Gruppen zusammengefasst werden und ein Muster (Klasse) steht für die gemeinsamen Eigenschaften dieser Objekte.

Kapselung


HOROLOGY.
PLATE CCCVI.
Fig. 2.
Fig. 4.
Fig. 3.
Fig. 1.
Fig. 6.
Fig. 7.
Fig. 5.
Fig. 8.
Fig. 9.

Identität

Abstraktion

- Ein VW Käfer ist ein Auto
- … ist ein Fahrzeug
- … ist ein Transportmittel

- Eine Maus ist ein Nagetier
- … ist ein Säugetier
- … ist ein Lebewesen

Eigenschaften des OO-Ansatzes

- Vererbung / Hierarchisierung / Abstraktion

  Objekt-Muster (Klassen) können ihre Eigenschaften an spezifischere Objekt-Muster vererben. Es entsteht eine Hierarchie von Objekt-Mustern, die eine verschieden tiefe Abstraktion widerspiegeln.

- Polymorphismus

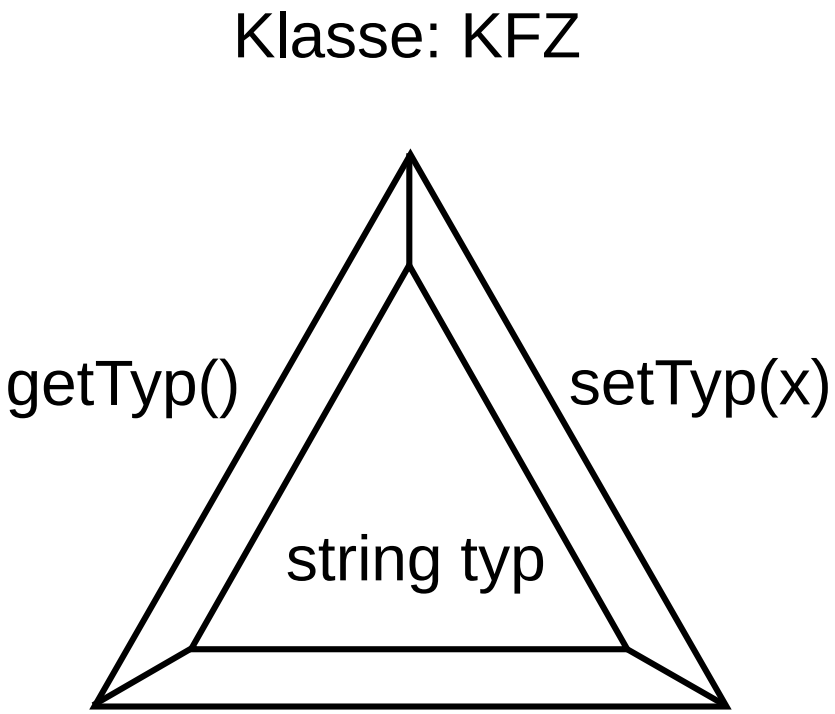
  Eine Funktion kann mehrmals implementiert sein. Beim Funktionsaufruf wird die passende Funktion ausgewählt und ausgeführt. (→ Kapselung, Abstrahierung)

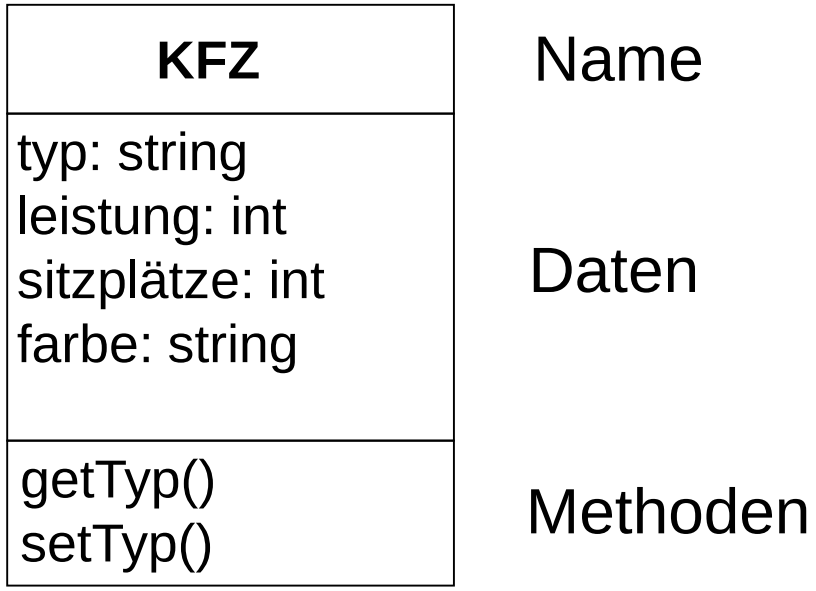
Vorteile des OO-Ansatzes

- Kein Bruch zwischen Modellierung und Implementierung
- Nachvollziehbarkeit der Implementierung
- Programmstabilität durch Kapselung
- Wiederverwertbarkeit

Kapselung und Klassen

- Kapselung (information hiding, Geheimnisprinzip) von Daten und Methoden (=Funktionen)





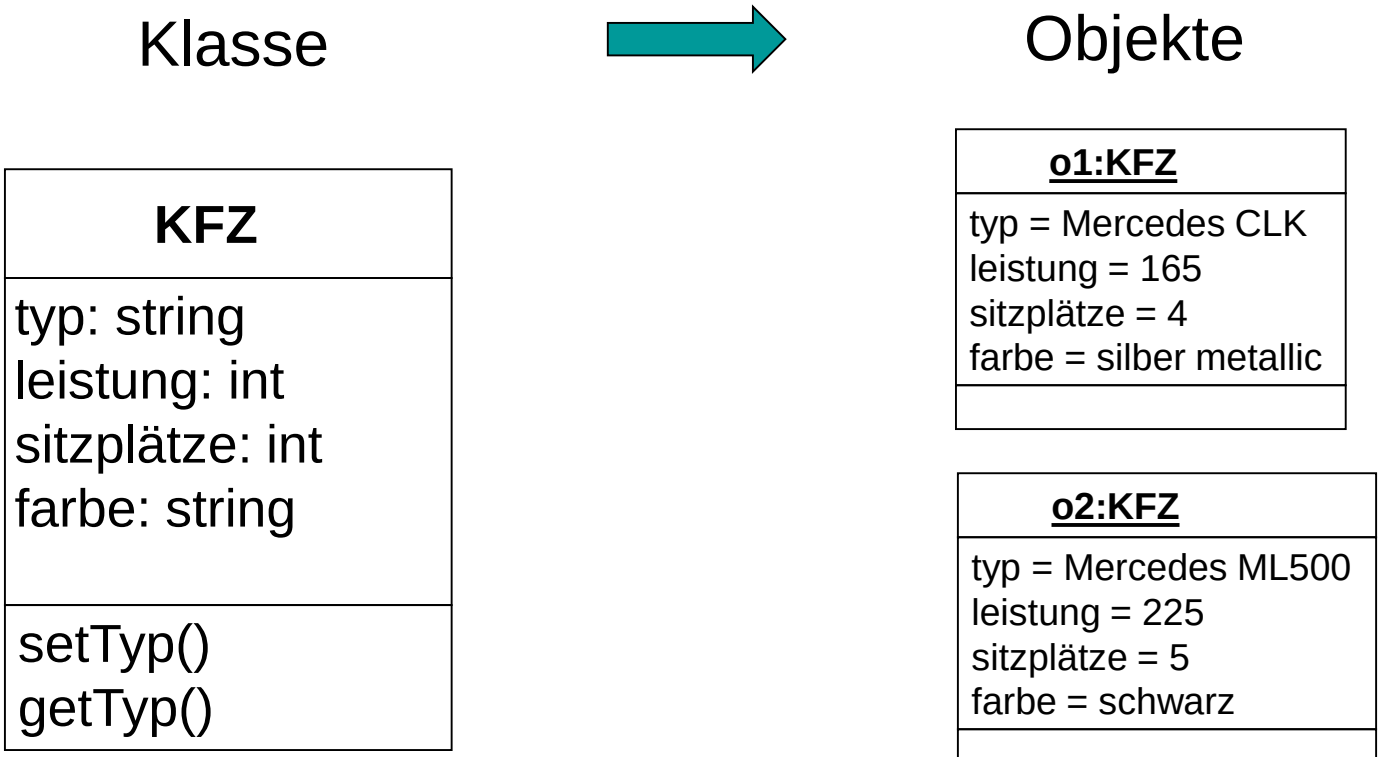
Kapselung und Klassen

- Daten **und** Funktionen für KFZ in einer Klasse

```
class KFZ {                              // Klassenname
   char typ[20];                         // Daten
   int motorleistung;                    // Daten
   int sitzplatzzahl;                    // Daten
   char farbe[15];                       // Daten
public:
   void setTyp(char t[]) {               // Methode
      strncpy(typ, t, 20); }
};
```

Objekt (Instanz, Exemplar)

- Objekt = Behältnis für Daten und Methoden (Funktionen)
- Klasse = Bauplan für Objekte



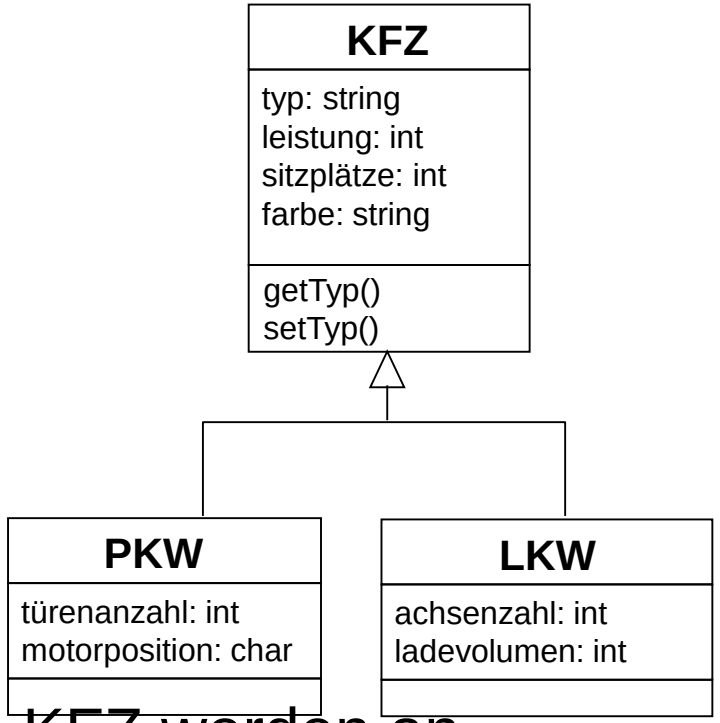
Objekt und Methodenaufruf

```
void main() {
   ...
   KFZ o1, o2;  // Objekte werden definiert
   ...
   // Methoden von Objekten werden aufgerufen:
   o1.setTyp("Mercedes CLK");
   o2.setTyp("Mercedes ML500");
   ...
}
```

Vererbung/Abstraktion

- Zusätzliche Daten **und** Funktionen für PKW und LKW jeweils in einer eigenen **Klasse**

```
class PKW : public KFZ {
    int tuerenanzahl;
};
class LKW : public KFZ {
    int achsenzahl;
    int ladevolumen;
};
```



- Restliche Daten und Funktionen von KFZ werden an PKW und LKW **vererbt**.

Zusammenfassung

- Die objektorientierte Programmierung setzt den Ansatz des objektorientierten Modellierens fort. Ein Bruch zwischen Modellierung und Implementierung wird vermieden.
- Objekte kapseln Daten und Funktionen und fördern dadurch
  - Nachvollziehbarkeit der Implementierung
  - Programmstabilität
  - Wiederverwertbarkeit
- Gleiche Objekte werden durch eine Klasse beschrieben. Eine Klasse enthält den Bauplan für Objekte.
- Gemeinsame Eigenschaften von ähnlichen Objekten werden in einer eigenen Klasse definiert und den Klassen mittels Vererbung zur Verfügung gestellt.