


|   |                         |               |                         |      |
|---|-------------------------|---------------|-------------------------|------|
|  | Klasse                  | Information   | Thema<br>Klassen in C++ |      |
|   | Lehrer<br>Oliver Kreuer | Lernfeld<br>6 | Datum                   | Name |

## Objektorientierte Programmierung und Klassen in C++

Unter **objektorientierter Programmierung** versteht man eine Art der Programmierung, die Gegenstände oder Begriffe und Gegebenheiten der realen Welt in sogenannten **Klassen** abbildet.

Dabei kann man eine Klasse als einen Bauplan für sogenannte **Objekte** verstehen, die man mit Hilfe dieser Klasse erzeugen kann.

Ein Objekt ist dann eine softwaretechnische Repräsentation eines realen Gegenstands oder eines gedachten Begriffs.

Objekte besitzen **Attribute** und **Methoden**. Die Attribute beschreiben die Eigenschaften eines Objekts. Die Methoden beschreiben die Operationen, die mit dem Objekt durchgeführt werden können.

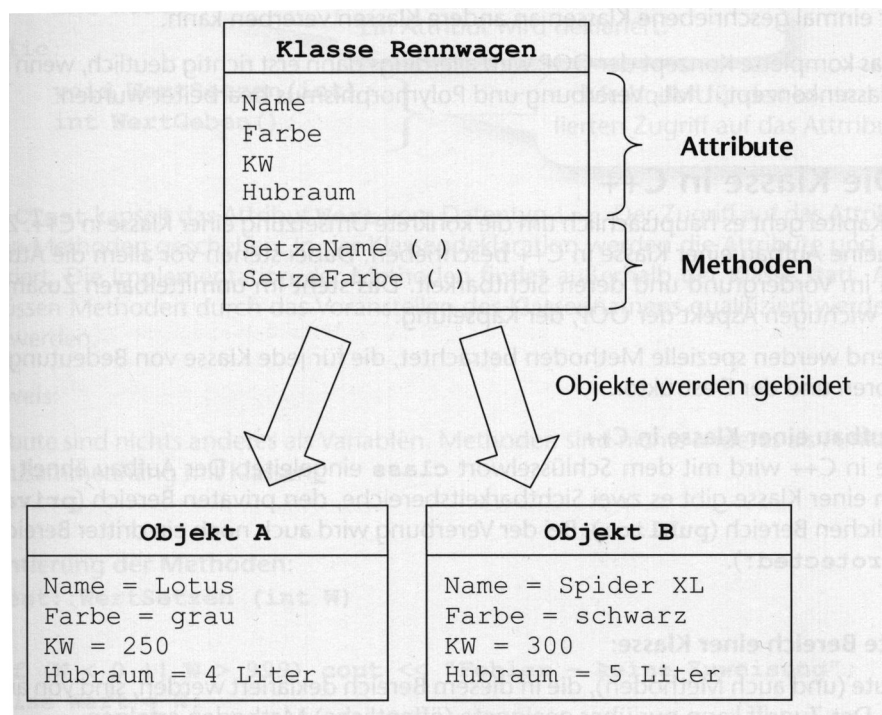



Abbildung 1: Beispiele für eine Klasse und deren Objekte

Innerhalb einer Klasse (und damit in den aus der Klasse erzeugten Objekten) werden zwei Bereiche unterschieden: Ein **privater** und ein **öffentlicher Bereich**.

Attribute und Methoden im öffentlichen Bereich sind außerhalb der Klasse sichtbar. Sie können über ein Objekt dieser Klasse aufgerufen und verwendet werden.

Attribute und Methoden im privaten Bereich sind **außerhalb der Klasse nicht sichtbar und können ausschließlich von Methoden der Klasse selbst verwendet werden**. Ein Zugriff auf Attribute und Methoden im privaten Bereich ist also nur über geeignete öffentliche Methoden möglich.

Diese Eigenschaft von Klassen bezeichnet man als **Datenkapselung**.

|   |                         |               |                         |      |
|---|-------------------------|---------------|-------------------------|------|
|  | Klasse                  | Information   | Thema<br>Klassen in C++ |      |
|   | Lehrer<br>Oliver Kreuer | Lernfeld<br>6 | Datum                   | Name |

## Syntaktischer Aufbau einer Klasse in C++

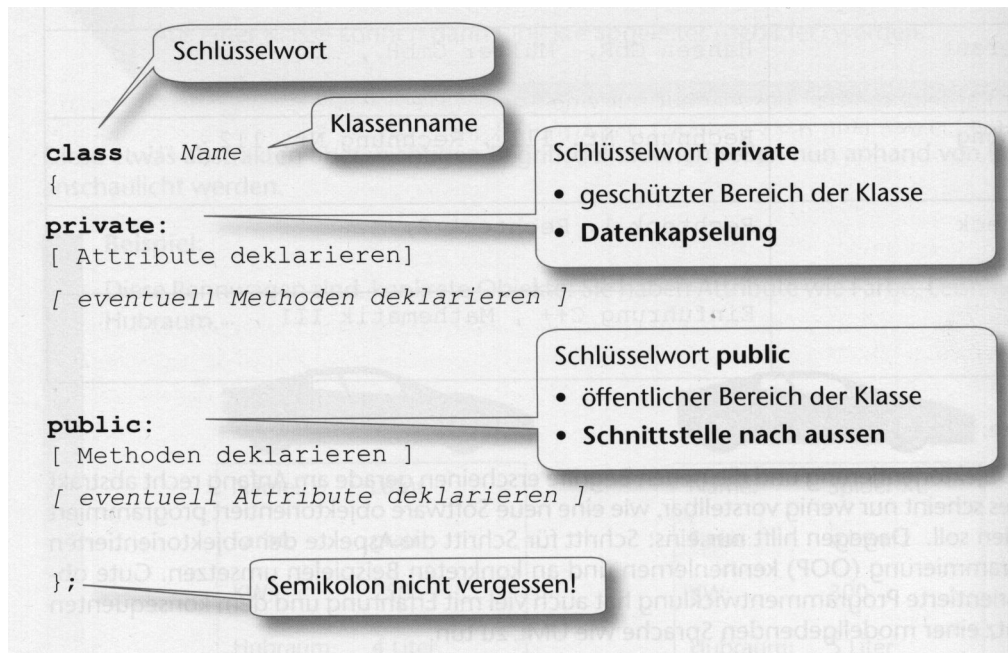


Abbildung 2: Syntaktischer Aufbau einer Klasse in C++

## Beispiel für das Definieren einer Klasse in C++ und die Implementation der zugehörigen Methoden

```

class Angestellter
{
private:
    char cName[21];
    char cVorname[21];
    double dGehalt;
public:
    double getGehalt();
    void setGehalt(double);
};


double Angestellter::getGehalt(void)
{
    return dGehalt;
}

void Angestellter::setGehalt(double dLohn)
{
    dGehalt=dLohn;
}

```

Attribute sind nichts anderes als Variablen; Methoden sind nichts anderes als Funktionen.

Qualifizierung der Methode durch Voranstellen des Klassennamens.

|   |                         |               |                         |      |
|---|-------------------------|---------------|-------------------------|------|
|  | Klasse                  | Information   | Thema<br>Klassen in C++ |      |
|   | Lehrer<br>Oliver Kreuer | Lernfeld<br>6 | Datum                   | Name |

### Beispiel für das Erzeugen und Verwenden eines Objekts dieser Klasse

```
int main(int argc, char *argv[])
{
    Angestellter hans;
    // Objekt der Klasse Angestellter erzeugen.

    // Gehalt von hans setzen und wieder ausgeben mittels cout.
    hans.setGehalt(42000.30);
    cout << "Gehalt des Angestellten Hans: " << hans.getGehalt() << endl;
}
```

#### Zu Beachten:

- Die Klassendefinition wird üblicherweise in einer eigenen Headerdatei vorgenommen, die den Namen der Klasse erhält. In obigem Beispiel wäre dies die Datei „Angestellter.h“.
- Die Methoden werden üblicherweise in einer zur Headerdatei passenden Datei implementiert. Hier wäre dies die Datei „Angestellter.cpp“. In dieser Datei muss mit Hilfe der Anweisung  

```
#include „Angestellter.h“
```

die Klassendefinition eingebunden werden. Danach folgen die Implementationen der einzelnen Methoden wie oben gezeigt.
- Um im Hauptprogramm (Funktion „main“) die Klasse verwenden zu können ist auch hier mittels include-Anweisung die Headerdatei einzubinden.

#### Quelle:

Die Abbildungen und Teile der Formulierungen stammen aus dem Buch „C++ und UML“ von Dirk Hardy, Europa Lehrmittel Verlag, 1. Auflage 2007, ISBN 978-3-8085-8546-7