K C++

- C++ ist
  - ➤ eine Erweiterung von C um einige moderne Sprachkonstrukte
  - ➤ eine objektorientierte Programmiersprache, die auf C aufbaut

➔ C++ kann sehr unterschiedlich eingesetzt werden
  - ➤ als "schöneres" C
  - ➤ als objektorientierte Sprache

- Probleme:
  - ➤ C++ hat auch alle "unschönen Dinge" von C geerbt
  - ➤ Programmieren in C++ wird oft mit objektorientierter Programmierung gleichgesetzt
    - wenn man objektorientiert programmieren will, ist C++ eine geeignete Sprache
    - aber: bloß weil man in C++ programmiert, muss das noch lange nicht objektorientiert sein

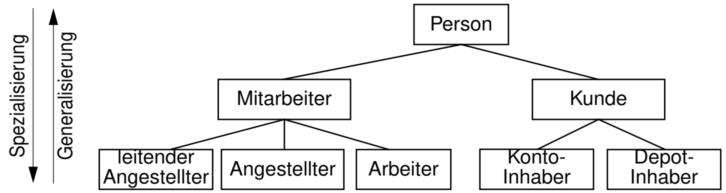
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.1 Objektorientierter Softwareentwurf

Bertrand Meyer:

Rechner führen Operationen auf bestimmten Objekten aus;
um flexiblere und wiederverwendbare Systeme zu erhalten,
ist es daher sinnvoller, die Software-Struktur
an diesen Objekten statt an den Operationen zu orientieren.

- Softwaresystem wird als Sammlung kooperierender Objekte modelliert
- einzelne Objekte sind Instanz einer Klasse in einer Hierarchie von Klassen
- Beispiel einer Klassenhierarchie:



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.1 Objektorientierter Softwareentwurf (2)

- Grundlage: objektorientierte Dekomposition
  - ➤ zentrale Frage ist nicht:
    "was ist zu tun und in welche Teilaufgaben kann ich das zerlegen?"
  - ➤ sondern:
    "aus welchen Komponenten besteht mein Problembereich, welchen Zustand haben diese Komponenten, was ist ihre Aufgabe, wie arbeiten sie zur Lösung des Gesamtproblems zusammen?"
- Vorteile:
  - + Wiederverwendung gemeinsamer Mechanismen
    - ➔ Software wird kleiner
  - + Software leichter zu ändern und weiterzuentwickeln
  - + Ergebnisse weniger komplex
  - + Besseres Verständnis des Auftraggebers für die Problemlösung

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.2 Objektorientierte Programmierung

1 Definition

OOP ist eine Methode der Implementierung, in der Programme in Form von

Mengen kooperierender Objekte

organisiert sind, wobei jedes Objekt

Instanz einer Klasse

ist und die Klassen Bestandteil einer über

Vererbungsbeziehungen

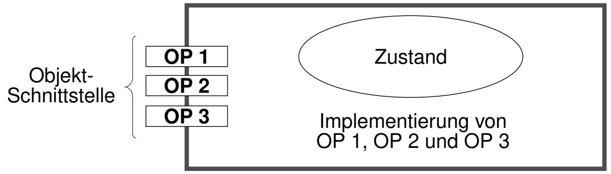
definierten Hierarchie von Klassen sind.

- Grundbegriffe
  - ➔ Objekt
  - ➔ Klasse
  - ➔ Vererbung

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Objekte & Methoden

- **Objekt** = Variablen (Zustand) + Operationen



- **Methoden** = Operationen eines Objekts (Schnittstelle)

3 Klassen

- **Klasse** = Objektbeschreibung (Schablone für Objekte)
- **Instanz** = aus einer Klasse erzeugtes Objekt

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.3 Objekte und Klassen in C++

- Klassendeklaration vergleichbar mit C-Strukturdeklaration
- Zugriff auf Objektkomponenten (Instanzvariablen und Methoden) wie auf Strukurkomponenten mit den Operatoren `.` bzw. `->`
- Beispiel:

```
// Class counter
class counter
{
  private:
    int value;
  public:
    void incr() { value++; }
    void decr() { value--; }
    int get_value() { return value; }
};
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.4 Instantiierung in C++

- Instantiierung von Objekten erfolgt
  - ➤ statisch zur Übersetzungszeit
  - ➤ dynamisch zur Laufzeit

1 statische Instantiierung

- durch Objektdefinition
- Beispiel:

```
void main()
{
  counter c1;      // Objekt c1 der Klasse counter
  counter *pc1;    // Zeiger auf ein counter-Objekt
  ...
}
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 dynamische Instantiierung

- C++ -Operatoren `new` und `delete`
  - ➤ statt Funktion malloc( ) in C
- Beispiele:

```
class counter
{ ... };
void main()
{
  counter c1;      // Objekt c1 statisch anlegen
  counter *pc1;    // Zeiger auf ein counter-Objekt
...
  pc1 = new counter;
  pc1->incr();
  c1.incr();
  ...
  delete pc1;
  ...
}
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Konstruktoren

Funktionen zum Initialisieren von Instanzen

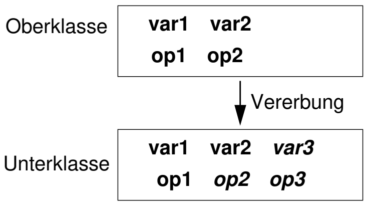
- Konstruktor: Methode mit Methodenname = Klassenname
  - ➤ wird bei Instantiierung automatisch aufgerufen
- Beispiel:

```
class counter {
  private:
    int value;
  public:
→   counter(int c) { value = c; } // Konstruktor
    void incr()    { value++; }
   ...
};
...
counter c1(20);        // c1 instantiieren, value mit 20 initialisieren
cp = new counter(30);
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.5 Vererbung

- **Vererbung**
  = Konstruktion von Klassen aus existierenden Klassen
- ➔ Oberklasse (Basisklasse) & Unterklasse



- ➤ Die Implementierung der Oberklasse wird nicht verändert
- ➤ Gleiche Funktionalität muß nicht mehrfach implementiert werden
- ➤ Veränderungen der Oberklasse wirken auf alle Unterklassen
- ➤ Beziehungen zwischen Klassen werden dokumentiert

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

K.6 Vererbung in C++

1 Überblick

- Unterklasse erbt Variablen und Methoden von der Basisklasse
- Unterklasse kann Basisklasse modifizieren
  - ➤ zusätzliche Methoden und Variablen
  - ➤ veränderte Methoden
- Methoden der Unterklasse haben Zugriff auf *public*- und *protected* -Bereich der Basisklasse
- *private*-Daten und -Methoden der Basisklasse für Methoden der Unterklasse nicht sichtbar

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Beispiel

```
// Class counter
class counter
{
  protected:
    int value;
  public:
    void incr() { value++; }
    void decr() { value--; }
    int get_value() { return value; }
};


// Subclass resettable counter
class rcounter : public counter
{
  private:
    int initial;
  public:
    rcounter(int v)       { initial = v; value = v; }
    void reset()          { value = initial; }
};
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.