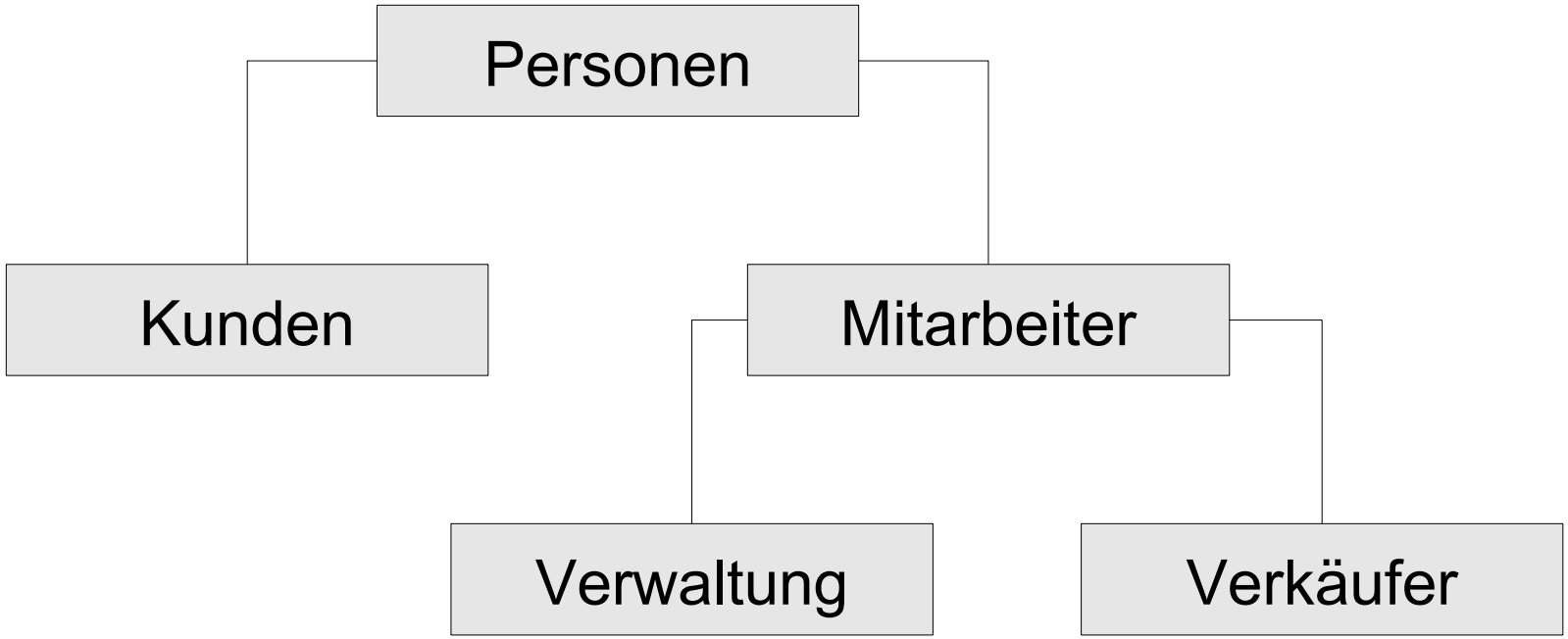
C++ - Objektorientierte Programmierung

Vererbung



Vererbung

- Definition von Klassen auf Basis von bestehenden Klassen.
- Eltern-Kind-Beziehung.
- Ableitung einer Klasse von einer anderen.
- Abbildung von Klassen in einem hierarchischen Modell.
- Implementierung von „ist ein“-Beziehung.

Basisklasse

- Oberklasse, Elternklasse, Superklasse.
- Allgemeine Beschreibung einer Gruppe von Objekten.
- Weitergabe von Attributen und Methoden.

Leibniz Universität Hannover

Basisklasse „Person“

Person
Vorname Nachname Straße Postleitzahl Ort
Setze Personname Lese Personname Setze Adresse Lese Adresse

Deklaration einer Basisklasse

Beispiele/cppOOP_006_...

```
class clsPerson {
{

};
```

- Beginn mit dem Schlüsselwort class.
- Dem Schlüsselwort folgt der Name der Klasse. Der Name ist eindeutig.
- Der Rumpf der Klasse wird durch die geschweiften Klammern begrenzt.
- Jede Deklaration einer Klasse endet mit einem Semikolon.

Basisklasse in C++: Attribute

```
class clsPerson {
public:

protected:
    string vorname;
    string nachname;
    string strasse;
    string postleitzahl;
    string ort;

};
```

Beispiele/cppOOP_006_...

Deklaration der Attribute

string	vorname	;
Datentyp	name	;

- Der Datentyp legt die Art des zu speichernden Wertes fest.
- Der Name des Attributs ist frei wählbar.
- Attribute werden am Anfang oder Ende der Klasse deklariert.
- Jede Deklaration eines Attributs endet mit dem Semikolon.

Zugriffsspezifizierer in einer Klasse

```
class clsPerson {

    public:

    private:

    protected:
};
```

- Lesezeichen für einen bestimmten Abschnitt.
- Wer kann auf die Elemente in der Klasse zugreifen?
- Dem Zugriffsspezifizierer folgt immer ein Doppelpunkt,

Öffentlicher Zugriff (public)

- Attribute können von jeden wie eine Postkarte gelesen und modifiziert werden.
- Die Elemente sind nicht in der Klasse, in der sie deklariert sind, gekapselt.
- Attribute sollten nie öffentlich deklariert werden.

Privater Zugriff (private)

- Die Attribute sind wie in einem Brief verschlossen.
- Ein Zugriff ist nur in der Klasse möglich, in der sie deklariert sind.
- Ein Zugriff von einer anderen Klasse auf die Elemente ist nicht möglich.
- Die Attribute können von einem Kind der Klasse nicht verwendet werden.

Geschützter Zugriff (protected)

- Ein Zugriff ist nur in der Klasse möglich, in der sie deklariert sind.
- Aber: Die Attribute können von einem Kind der Klasse verwendet werden.
- Ein Zugriff von anderen Klasse auf die geschützten Elemente ist nicht möglich. Die Attribute sind vor einen Zugriff von außen geschützt.

Basisklasse in C++: Konstruktoren

Beispiele/cppOOP_006_...

```
class clsPerson {
public:
    clsPerson();
    clsPerson(string, string);

    clsPerson(const clsPerson& orig);
};
```

Basisklasse in C++: Methoden

Beispiele/cppOOP_006_...

```
class clsPerson {
public:

    string getAdresse();
    void setAdresse(string, string plz = "", string strasse = "");

    string getPersonName;
    void setPersonName(string, string vorname = "");
};
```

Subklasse

- Unterklasse, Kindklasse.
- Eltern-Kind-Beziehung.
- Abgeleitete Klasse von ein oder mehreren Basisklassen.
- Erweiterung oder Spezialisierung der Basisklasse.

Subklassen

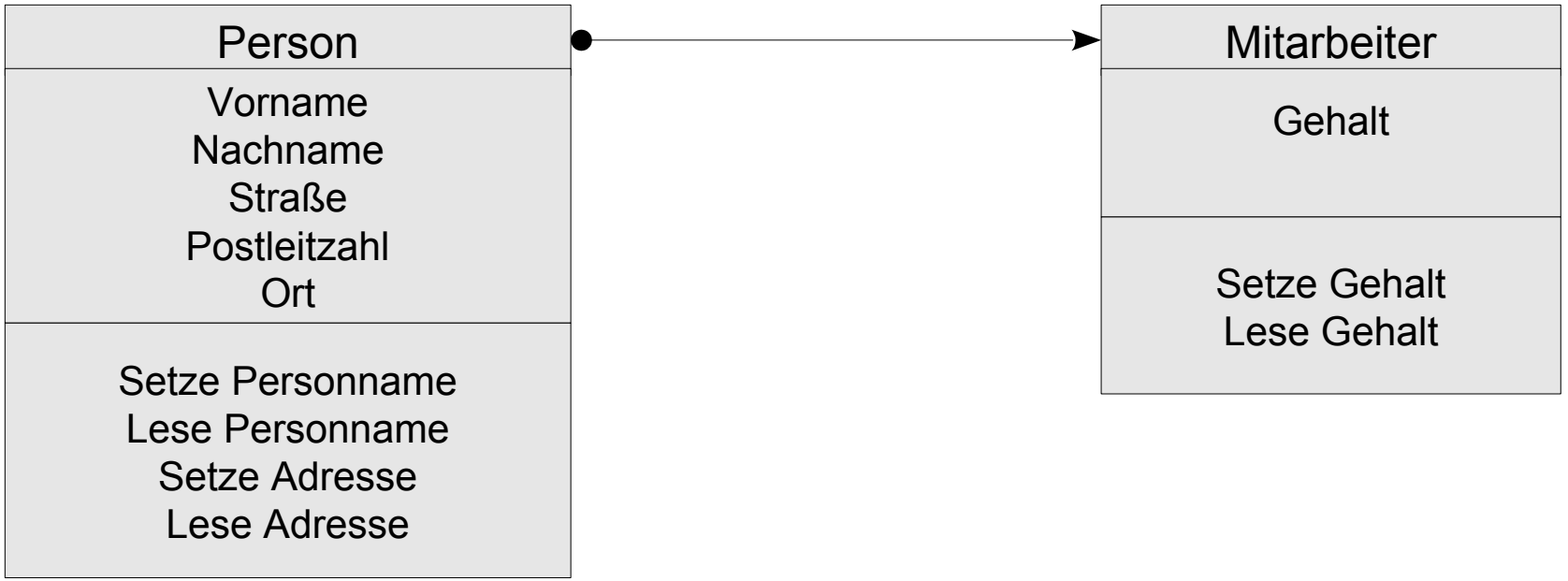
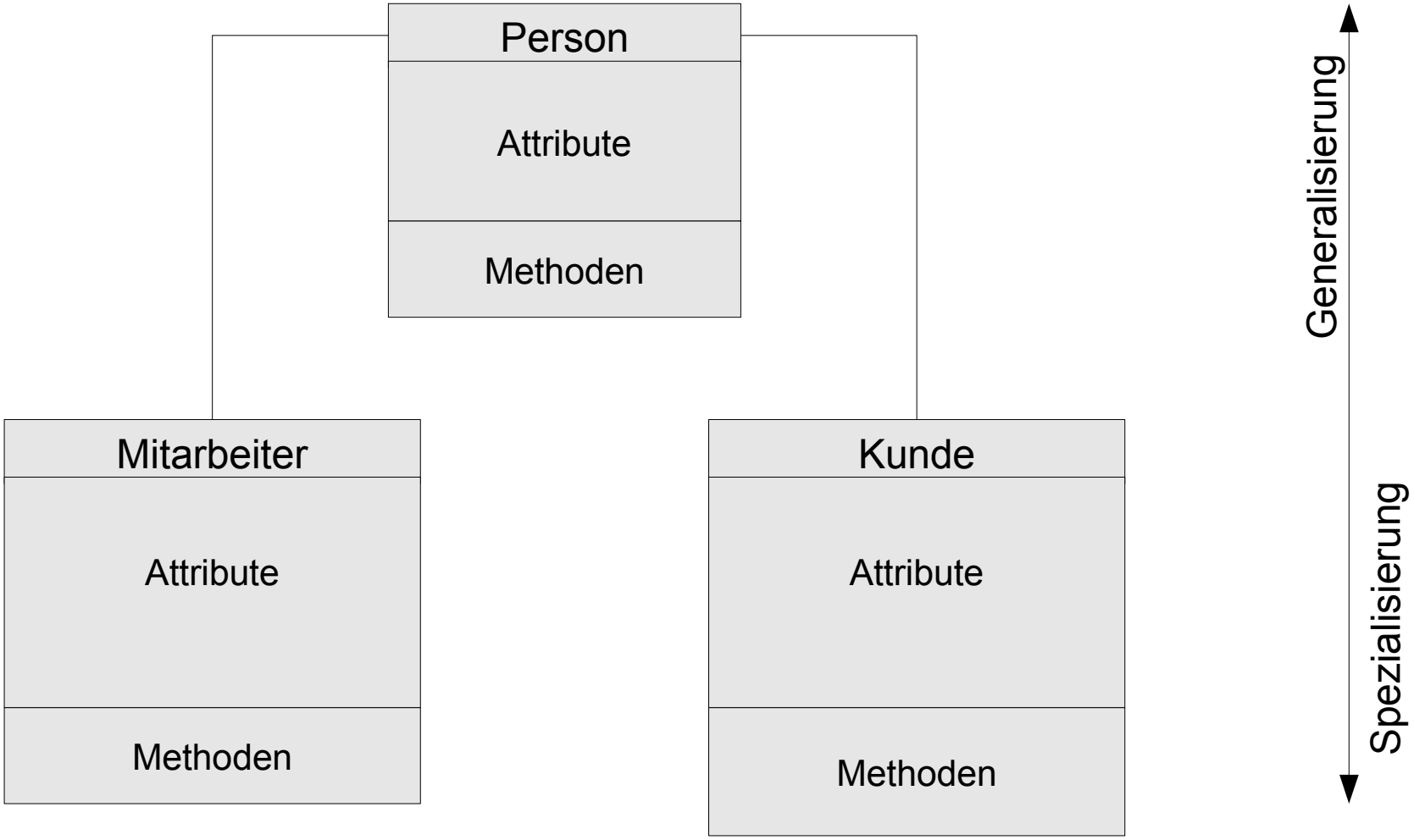

Person
Vorname
Nachname
Straße
Postleitzahl
Ort
Setze Personname
Lese Personname
Setze Adresse
Lese Adresse
Mitarbeiter
Gehalt
Setze Gehalt
Lese Gehalt

Abbildung als hierarchisches Modell



Erläuterung

- Beschreibung eines Stammbaums.
- Von oben nach unten in der Hierarchie: Um so tiefer man kommt, um so spezieller werden die Attribute der Klasse. Die Beschreibung eines Objekts wird detaillierter.
- Von unten nach oben in der Hierarchie: Um so höher man kommt, um so allgemeiner werden die Attribute der Klasse. Die Gemeinsamkeit einer Gruppe von Objekten wird allgemein beschrieben.

Subklasse in C++

Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:

private:
};
```

Deklaration der Klasse

Beispiele/cppOOP_006_...

```
class clsMitarbeiter{
public:

private:
};
```

- Beginn mit dem Schlüsselwort class.
- Dem Schlüsselwort folgt der Name der Klasse. Der Name ist eindeutig.
- Der Rumpf der Klasse wird durch die geschweiften Klammern begrenzt.
- Jede Deklaration einer Klasse endet mit einem Semikolon.

Ableitung von der Basisklasse

Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:

private:
};
```

- Dem Namen der Subklasse folgt ein Doppelpunkt.
- Dem Doppelpunkt folgt ein Zugriffsspezifizierer für den Zugriff auf die Basisklasse.
- Im Anschluss daran folgt der Name der Basisklasse.

Öffentlicher Zugriff (public)

- Ist-ein-Beziehung.
- Elemente, die in der Basisklasse public- oder protected-Zugriff haben, behalten ihre Zugriffsrechte.

Beispiel

Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:
    clsMitarbeiter();

    clsMitarbeiter(string, string);
    clsMitarbeiter(string, double, string vorname = "");

    clsMitarbeiter(const clsMitarbeiter& orig);

    void setGehalt(double);
    double getGehalt();

private:
    double gehalt;
};
```

Zugriff „class subClass: public basicClass“

Basisklasse	Subklasse	
Zugriffsspezifkatoren	class subclass {}	subclass objekt
public	public	public
private	private	private
protected	protected	protected

Geschützter Zugriff (protected)

- Implementiert durch-Beziehung.
- Elemente, die in der Basisklasse public- oder protected-Zugriff haben, haben in der Subklasse einen protected-Zugriff.
- Objekte der abgeleiteten Klasse können nicht mehr auf die Elemente der Basisklasse zu greifen.

Beispiel

```
class clsKunden: protected clsPerson {
public:
    clsKunden();
    clsKunden(string, string, string);
    clsKunden(string, string);
    clsKunden(string, string, double);
    clsKunden(const clsKunden& orig);
    string getKundenname();
    void setBestelllimit(double);
    double getBestelllimit();

private:
    double bestelllimit;
    string anrede;
};
```

Beispiele/cppOOP_006_...

Zugriff „class subClass: protected basicClass“

Basisklasse	Subklasse	
Zugriffsspezifkatoren	class subclass {}	subclass objekt
public	public	protected
private	private	private
protected	protected	protected

Zugriffsbeschränkung aufheben

```
class clsKunden: protected clsPerson {
public:

    using clsPerson::getPersonName;
    using clsPerson::getAdresse;
```

- Über Objekte der Klasse clsKunden können keine öffentlichen Elemente aufgerufen werden.
- Durch die using-Anweisung wird die Zugriffsbeschränkung für die Methoden getPersonName und getAdresse aufgehoben.

using-Anweisung

using	clsPerson	::	getPersonName	;
using	Klasse	::	methodename	;

- Links vom Bereichsoperator: In welcher Basisklasse ist die Methode, auf die die Subklasse zugreifen möchte, deklariert?
- Links vom Bereichsoperator: Auf welche Methode möchte die Subklasse zugreifen?

Privater Zugriff (private)

- Standard-Zugriff, wenn keine Angaben gemacht werden.
- Elemente, die in der Basisklasse public- oder protected-Zugriff haben, haben in der Subklasse einen private-Zugriff.

Zugriff „class subClass: private basicClass“

Basisklasse	Subklasse	
Zugriffsspezifkatoren	class subclass {}	subclass objekt
public	public	private
private	private	private
protected	protected	private

Subklasse in C++: Attribute

Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:

private:
    double gehalt;
};
```

Hinweis

- Die Attribute der Basisklasse werden um spezifische Attribute der Subklasse ergänzt.
- Es werden spezielle Attribute für die Subklasse definiert.
- Die Subklasse kann auf die Attribute der Basisklasse entsprechend des Zugriffsspezifizierers des Attributs und der Ableitung zugreifen.

Subklasse in C++: Konstruktoren

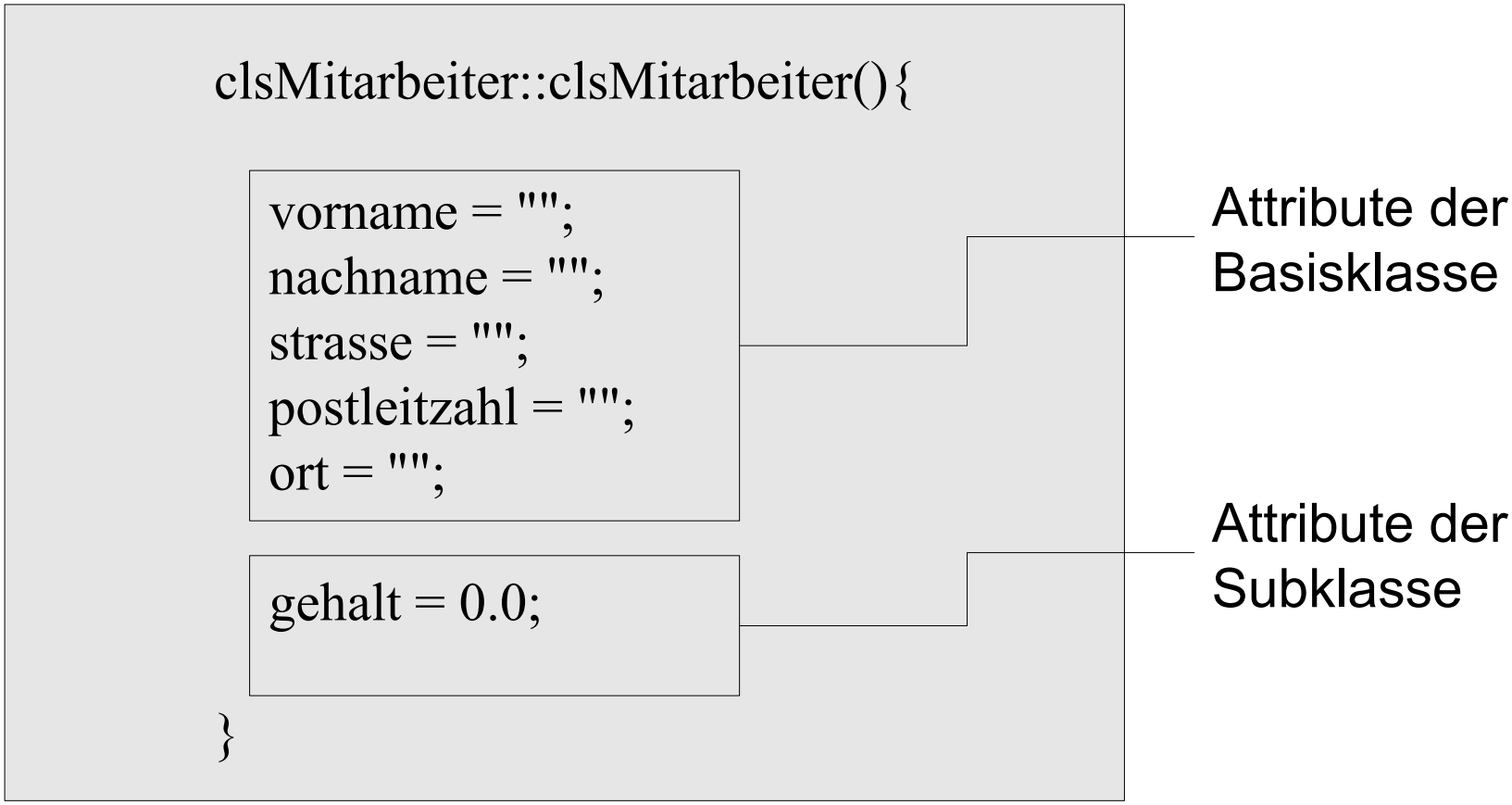
Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:
    clsMitarbeiter();

    clsMitarbeiter(string, string);
    clsMitarbeiter(string, double, string vorname = "");

    clsMitarbeiter(const clsMitarbeiter& orig);
};
```

Hinweis

- Konstruktoren und Destruktoren der Basisklasse werden nicht vererbt.
- Mindestens ein Standardkonstruktor sollte vorhanden sein.

Standardkonstruktor



Beispiele/cppOOP_006_...

Nutzung der Initialisierungsliste

Beispiele/cppOOP_006_...

```
clsMitarbeiter::clsMitarbeiter(string nachname, string vorname)
  : clsPerson(nachname, vorname){
    gehalt = 0.0;
}

clsMitarbeiter::clsMitarbeiter(string nachname,
                               double gehalt, string vorname)
    : clsPerson(nachname, vorname), gehalt(gehalt){

}
```

Initialisierungsliste im Kopf des Konstruktors

bereich	::	konstruktor	(	parameterliste	)	:	initialisierungsliste

- Die Initialisierungsliste wird vom Kopf des Konstruktors durch einen Doppelpunkt getrennt.
- Die Liste kann aus beliebig vielen Elementen, getrennt durch ein Kommata, bestehen.

Initialisierung der Attribute der Basisklasse

konstruktor	(	parameterliste	)	:	initialisierungsliste		
					Konstruktor Basisklasse	,	attribut(parameter)

- Als erster Parameter muss ein ein passender Konstruktor der Basisklasse angegeben werden.
- Mit Hilfe dieses Konstruktors werden die, in der Basisklasse deklarierten Attribute initialisiert.
- Der Konstruktor wird über den Namen der Basisklasse aufgerufen.

Leibniz Universität Hannover

Basisklassen-Initialisierer

konstruktorname	(		)	;
konstruktorname	(	parameterliste	)	;
clsPerson	(	nachname, vorname	)	;

- Es kann der Standard-Konstruktor oder ein allgemeiner Konstruktor der Basisklasse aufgerufen werden.
- Die Parameterliste wird durch die runden Klammern begrenzt. Die Parameter in der Liste werden durch Kommata getrennt.

Regeln

- Wenn eine Basisklasse keinen Standard-Konstruktor hat, muss die Subklasse einen Basisklassen-Initialisierer für diesen angeben.
- Es werden immer zuerst die Attribute der Basisklasse und dann die Attribute der Subklasse initialisiert.

Initialisierung der Attribute der Subklasse

konstruktor	(	parameterliste	)	:	initialisierungsliste		
					Konstruktor Basisklasse	,	attribut(parameter)

- Anschließend werden die Attribute der Subklasse initialisiert.
- Dem Attribut wird in runden Klammern der Startwert übergeben.
- Konstante Instanzvariablen müssen über eine Initialisierungsliste initialisiert werden.
- Die Angabe der Attribute entspricht der Auflistung der Attribute in der Klasse.

Subklasse in C++: Methoden

Beispiele/cppOOP_006_...

```
class clsMitarbeiter: public clsPerson {
public:
    void setGehalt(double);
    double getGehalt();

private:
};
```

Hinweis

- Methoden von der Basisklasse können in der Subklasse entsprechend des Zugriffsspezifizierers der Ableitung aufgerufen werden.
- Methoden, die nur in der Subklasse genutzt werden, können ergänzt werden.
- Methoden der Basisklasse können überschrieben werden.

Objektvariablen der Basisklasse

```
class clsPerson person("Person Nachname", "vorname");
```

Beispiele/cppOOP_006_...

- Variable vom Typ „Klasse“.
- Die Klasse muss durch die Anweisung #include eingebunden werden.
- In der Variablen wird als Wert die Speicheradresse des erzeugten Objektes abgelegt.

Objektvariablen der Subklasse

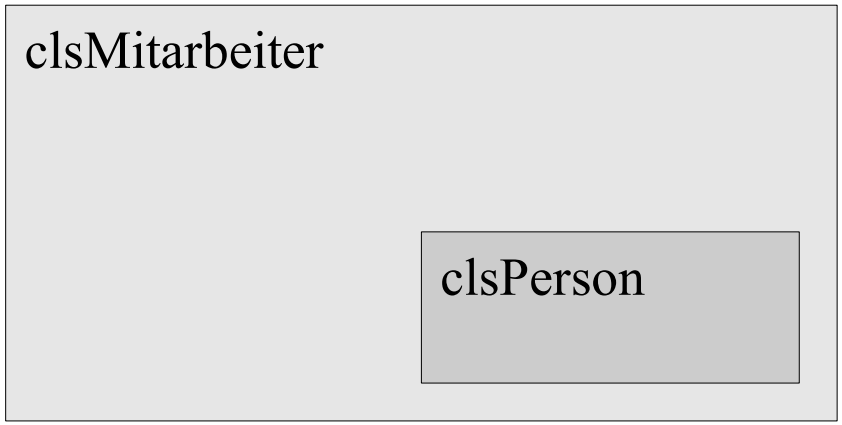
```
class clsMitarbeiter mitarbeiter("Mitarbeiter Nachname", 2400, "vorname");
class clsKunden kunde("Herr", "Kunde", 1000);
```

Beispiele/cppOOP_006_...

- Variable vom Typ „Klasse“.
- Die Klasse muss durch die Anweisung #include eingebunden werden.
- In der Variablen wird als Wert die Speicheradresse für ein erzeugtes Objekt der Subklasse abgelegt.
- Die Subklasse hat als Unterobjekt die Basisklasse, von der sie abgeleitet ist.

Instanziierung

- Das Basisklassen-Unterobjekt wird zuerst mit Hilfe eines passenden Konstruktors angelegt. Die geerbten Elemente bilden ein Unterobjekt in dem Objekt vom Typ „Subklasse“.
- Anschließend werden die Instanzvariablen der Subklasse, in der Reihenfolge angelegt, in der sie in der Klasse deklariert sind. Den Variablen kann mit Hilfe einer Initialisierungsliste ein Anfangswert zugewiesen werden.



Objekte dynamisch erzeugen

```
class clsMitarbeiter *ptrMitarbeiter;
ptrMitarbeiter = new clsMitarbeiter("Mitarbeiter Nachname"
                                    , 2400, "vorname");

ptrMitarbeiter->setAdresse("ort", "12345", "strasse");

cout << "\nPointer to Mitarbeiter - Info: "
                          << ptrMitarbeiter->getInformationen();
delete ptrMitarbeiter;
```

Beispiele/cppOOP_006_...

Arbeiten mit new

```
class clsMitarbeiter *ptrMitarbeiter;
ptrMitarbeiter = new clsMitarbeiter("Mitarbeiter Nachname"
                                    , 2400, "vorname");
```

- Mit Hilfe des Schlüsselwortes new wird Speicher angefordert. Der Speicher wird im Heap des Rechners bereitgestellt.
- Dem Schlüsselwort folgt die benötigte Größe des Speicherbereichs. In diesem Beispiel wird Speicher für ein Objekt der Klasse clsQueue angefordert.
- In runden Klammern folgt die Parameterliste. In Abhängigkeit der Parameterliste wird der passende Konstruktor aufgerufen.

Objekt löschen

```
delete ptrMitarbeiter;
```

- Objekte, die mit Hilfe von new erzeugt wurden, müssen mit delete gelöscht werden.
- Das Schlüsselwort delete ruft den Destruktor einer Klasse auf.

Zeiger vom Typ „Subklasse“

```
class clsMitarbeiter *ptrMitarbeiter;
ptrMitarbeiter = new clsPerson("Mitarbeiter Nachname",
                               "vorname");
```

- Zeiger vom Typ „Subklasse“ können nicht auf ein Objekt der Basisklasse zeigen.
- Der Fehler „invalid conversion from“ wird vom Compiler gemeldet.

Zeiger vom Typ „Basisklasse“

```
class clsPerson *ptrPerson;
ptrPerson = new clsMitarbeiter("Mitarbeiter Nachname",
                               2400, "vorname");
```

- Zeiger vom Typ „Basisklasse“ können nicht auf ein Objekt der Subklasse zeigen.
- Der Zeiger vom Typ „Basisklasse“ kann nur Methoden der Basisklasse aufrufen, aber keine Methoden der Subklasse.
- Das Objekt der Subklasse wird automatisiert in ein Objekt der Basisklasse konvertiert.