# Objektorientierte Systementwicklung:

# Vererbung

# Inhaltsverzeichnis:

# 1 Ausgangssituation

Der Getränkemarkt GeLa GmbH, Esslingen, entschließt sich, in Zukunft auch die Mitarbeiterdaten mit Hilfe einer objektorientierten Software zu verwalten. Zurzeit sind die Mitarbeiter Frank Walser (Gehalt: 1.600,00 Euro), Eva Flink (1.650,00 Euro) und Hans Boss (4.000,00 Euro) beschäftigt. Aufgrund der geplanten Expansion des Unternehmens in Form eines Erweiterungsbaus und der Eröffnung von Filialen im Umland, ist mit einer stark wachsenden Mitarbeiterzahl zu rechnen. Außerdem sollen in Zukunft Auszubildende beschäftigt werden. Mit Sven Vogel (Gehalt: 450,00 Euro) wurde bereits ein Ausbildungsvertrag vereinbart.

Die GeLa GmbH hat folgende Wünsche an die zu entwickelnde Software:

- Name, Vorname und Gehalt eines Mitarbeiters bzw. Auszubildenden sollen erfasst werden können.
- Von den Auszubildendern sollen zusätzlich die Bezeichnung des Ausbildungsberufs und das aktuelle Ausbildungsjahr erfasst werden können.
- Alle Daten müssen einzeln geschrieben werden können.
- Alle Daten müssen einzeln gelesen werden können.
- Bei einem bestehenden Gehalt eines Mitarbeiters bzw. Auszubildendern soll eine Erhöhung um einen absoluten Betrag durchgeführt werden können.

**Aufgaben:**

1 Leiten Sie aus der vorliegenden Beschreibung mögliche Klassen sowie zugehörige Attribute und Methoden ab.

2 Modellieren Sie auf Basis der Situationsbeschreibung ein Klassendiagramm. (Projektname: projektMitarbeiter_1; Paketname: paketMitarbeiter.)

3 Implementieren Sie die Methode gehaltErhoehen.

4 Entwickeln Sie eine geeignete Klasse mit main()-Methode (»StartMitarbeiter«) für das Einlesen und Ausgeben der anfänglichen Attributwerte (Objektnamen: mitarb1, mitarb2, chef und azubi1).
Das Gehalt des Chefs soll nachträglich um 1.000,00 Euro erhöht werden.
Abschließend sollen die Mitarbeiterdaten auf der Konsole ausgegeben werden (siehe Abb.).

```
Mitarbeiter der GeLa GmbH
-------------------------
Mitarbeiter    Frank Walser 1600 Euro
Mitarbeiter    Eva Flink 1650 Euro
Mitarbeiter    Hans Boss 4000 Euro
Auszubildender Sven Vogel 450 Euro

Neues Gehalt des Chefs: 5000 Euro
```

## 2 Generalisierung - Spezialisierung

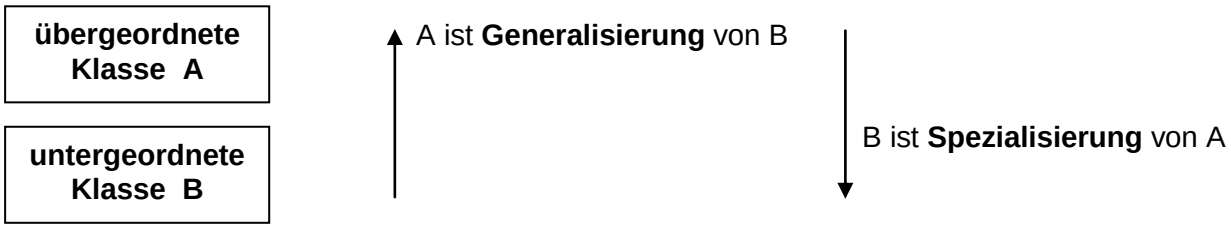
Bei der Modellierung der Klassen »Mitarbeiter« und »Auszubildender« zeigt sich, dass beide Klassen identische Attribute (*vorname*, *name*, *gehalt*) und Methoden (Setter und Getter) aufweisen. Die Klasse »Auszubildender« muss zusätzlich um die Attribute *ausbildungsberuf* und *ausbildungsjahr* erweitert werden. Werden beide Klassen implementiert, treten Redundanzen auf. Ein weiterer Nachteil zeigt sich, wenn ein neues, gemeinsames Merkmal (z.B. Geburtstag) in beiden voneinander unabhängigen Klassen eingefügt werden soll.

Die Lösung dieses Problems besteht darin, eine „gemeinsame“ Klasse »Mitarbeiter« anzulegen und die zusätzliche Klasse »Auszubildender« „erben“ zu lassen, da die Klasse »Auszubildender« eine Spezialisierung der Klasse »Mitarbeiter« darstellt.

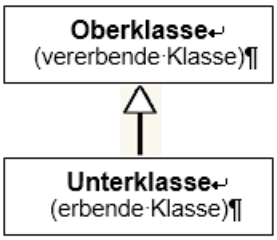
Dies bedeutet, dass Klassen hierarchisch geordnet werden können. Dabei übernehmen (erben) sie die Eigenschaften der ihr übergeordneten Klasse (Vererbungsprinzip).

Das Grundprinzip der **Vererbung** besteht darin, aus bereits bekannten und beschriebenen Elementen eines Objektmodells weitere Elemente abzuleiten. Die Vererbung beschreibt somit eine Beziehung zwischen einer allgemeinen Klasse (Oberklasse, super class) und einer spezialisierten Klasse (Unterklasse, sub class). Eine Unterklasse ist vollständig identisch mit der Oberklasse, ist aber um zusätzliche Attribute, Methoden und gegebenenfalls Assoziationen erweitert und somit spezifiziert. Daher bezeichnet man Vererbungsstrukturen zwischen Klassen auch als Generalisierung/Spezialisierung.

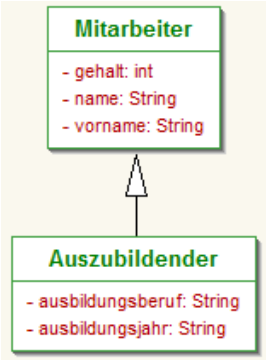
Mit Hilfe des Prinzips der Vererbung wird die Wiederverwendbarkeit als zentrales Merkmal der objektorientierten Systementwicklung realisiert.




Die Darstellung der Vererbungsstruktur im UML-Klassendiagramm erfolgt über einen Vererbungspfeil (mit transparenter Pfeilspitze), der von der erbenden Klasse (Unterklasse, Subklasse) zur vererbenden Klasse (Oberklasse, Superklasse) zeigt.




Für das beschriebene Problem ergibt sich daraus Nebenstehendes UML-Klassendiagramm.
(Auf die Darstellung der Methoden wird verzichtet.)



# 3 Umsetzung der Vererbung in Java

## 3.1 Quellcode in untergeordneten Klassen

In der Klassendefinition einer Unterklasse (Subklasse) muss darauf hingewiesen werden, dass die Attribute und Methoden der Oberklasse (Superklasse) zur Verfügung stehen sollen. Dies erfolgt mit der Anweisung:

| **public class <<Subklasse>> extends <<Superklasse>>** |
|---|

Für das Projekt *projektMitarbeiter_1* ergibt sich daraus die Java-Anweisung

**public class Auszubildender extends Mitarbeiter {…**

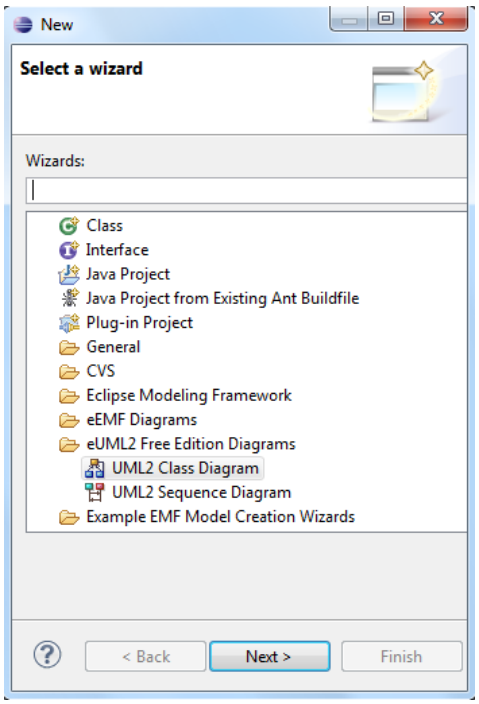
**Erläuterung**: Die Unterklasse **»Auszubildender«** erweitert (= extends) die Beschreibung und Definition der Oberklasse **»Mitarbeiter«**.
Im vorliegenden Beispiel erweitert die Klasse »Auszubildender« unsere Vorstellungen von »Mitarbeitern«.
Sollen Attribute und Methoden einer Oberklasse geerbt werden, die einem anderen Java-Paket zugeordnet ist, so muss der Paketname dieser Oberklasse vorangestellt werden.

public class <<Subklasse>> extends <<Paketname.Superklasse>>

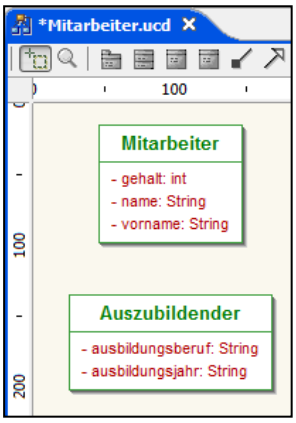
## 3.2 Software unterstützte Modellierung der Vererbungsstruktur

Nachdem ein neues Java-Projekt (projektMitarbeiter_1) mit dem Package paketMitarbeiter erzeugt wurden, wird der UML-Editor geöffnet.

**Befehlspfad:** File
→ New
→ Other
→ eUML2Free Edition Diagrams
→ UML2 Class Diagram



Hier sind die Klassen »Mitarbeiter« und »Auszubildender« nach bekanntem Muster zu erstellen:




**Neue Klasse:**

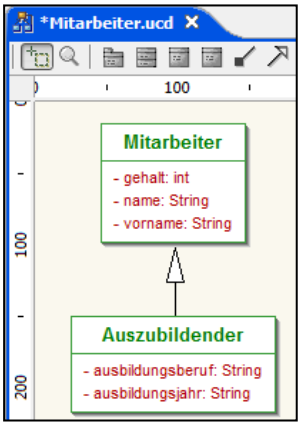
- Kontextmenü (rechte Maustaste): New → Class
- Klasse-Symbol auf Zeichenfläche ziehen

**Neue Attribute:**

- gewünschte Klasse markieren
- Kontextmenü: New → Property
- Attributname und Datentyp festlegen

Die Vererbungsstruktur wird mit dem Symbol  (Generalization) hergestellt:

- Generalization-Symbol aktivieren (anklicken),
- Subklasse markieren,
- Superklasse markieren.




Gleichzeitig hat das System den Quellcode für die Klassen generiert:

**Fachklasse »Mitarbeiter« (Oberklasse)**

```
package paketMitarbeiter;
public class Mitarbeiter {
//Variablendeklaration
    private String name;
    private String vorname;
    private int gehalt;
//Konstruktor
    public Mitarbeiter(){
    }
//set- und get-Methoden
    public void setName(String name) {
        this.name = name;
    }
    public void setVorname(String vorname) {
        this.vorname = vorname;
    }
    public void setGehalt(int gehalt) {
        this.gehalt = gehalt;
    }
    public String getName() {
        return name;
    }
    public String getVorname() {
        return vorname;
    }
    public int getGehalt() {
        return gehalt;
    }
}
```

### Fachklasse »Auszubildender« (Unterklasse, Subklasse)

```
package paketMitarbeiter;
public class Auszubildender extends paketMitarbeiter.Mitarbeiter {
//Variablendeklaration
        private String ausbildungsberuf;
        private String ausbildungsjahr;
//Konstruktor
        Auszubildender()
        { }
//set- und get-Methoden
        public void setAusbildungsberuf(String ausbildungsberuf) {
                this.ausbildungsberuf = ausbildungsberuf;
        }
        public void setAusbildungsjahr(String ausbildungsjahr) {
                this.ausbildungsjahr = ausbildungsjahr;
        }
        public String getAusbildungsberuf() {
                return ausbildungsberuf;
        }
        public String getAusbildungsjahr() {
                return ausbildungsjahr;
        }
}
```

Sofern sich die Ober- und die Unterklasse im selben Paket befinden, kann auf den Zusatz "paketMitarbeiter" verzichtet werden.

```
public class Auszubildender
        extends Mitarbeiter
```

Abschießend ist die Methode zum Erhöhen des Gehalts in der Fachklasse »Mitarbeiter« zu implementieren. Das Vererbungsprinzip stellt sicher, dass auch die Klasse »Auszubildender« über diese Methode verfügen kann.

```
public void gehalterhoehen( int erhoehung){
  this.gehalt = this.gehalt + erhoehung;
}
```

### 3.3 Quellcode zur Erzeugung von Objekten einer untergeordneten Klasse

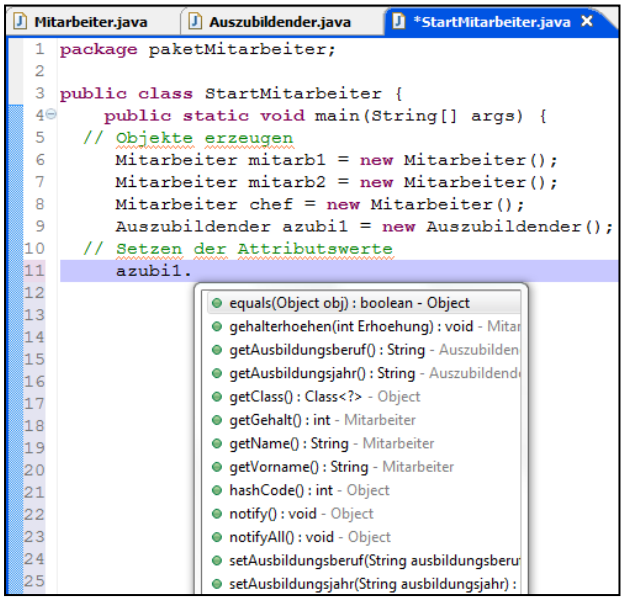
Mit Hilfe einer Klasse mit main()-Methode (»StartMitarbeiter«) sollen nun die Mitarbeiter- und Auszubildendernobjekte (siehe oben, Seite 2, Aufgabe 4) erzeugt und die entsprechenden Attributwerte eingelesen werden. Anschließend sollen die Daten entsprechend der nachfolgenden Abbildung auf der Konsole ausgegeben werden.

```
Mitarbeiter der GeLa GmbH
-------------------------
Mitarbeiter    Frank Walser 1600 Euro
Mitarbeiter    Eva Flink 1650 Euro
Mitarbeiter    Hans Boss 4000 Euro
Auszubildender Sven Vogel 450 Euro

Neues Gehalt des Chefs: 5000 Euro
```

Mit Hilfe der Entwicklungsumgebung Eclipse kann verdeutlicht werden, dass ein Objekt der Unterklasse Auszubildender auch alle Attribute der Klasse »Mitarbeiter« erbt.

In der Klasse »StartMitarbeiter« werden die Objekte *mitarb1*, *mitarb2*, *chef* und *azubi1* erzeugt.
Nach der Eingabe der Befehlszeile “azubi1.“ öffnet sich das Kontextmenü der Code-Ergänzungsfunktion (siehe Abb.).




Darin ist zu erkennen, dass neben dem Ausbildungsberuf und dem Ausbildungsjahr auch das Gehalt, der Name und der Vorname zum aktuellen Objekt (*azubi1*) gehören. **Allerdings kann der Zugriff auf die Attribute der übergeordneten Klasse nur über die set- und get-Methoden der Oberklasse erfolgen**.

(Einen Lösungsvorschlag zur beschriebenen Aufgabenstellung finden Sie auf der Folgeseite.)

## 4 Übung

Die Immobiliengesellschaft Immo-GmbH möchte ihre Immobilienangebote mit Hilfe einer objektorientierten Software verwalten. Für jede Wohnung sind Ort, Baujahr, Wohnfläche und Preis zu erfassen. Für jedes Wohnhaus soll zusätzlich noch die Grundstücksfläche erfasst werden.

1 Modellieren Sie zur beschriebenen Situation ein UML-Klassendiagramm. (Projektname: *projektImmoGmbH*; Paketname: *paketImmobilie*.)

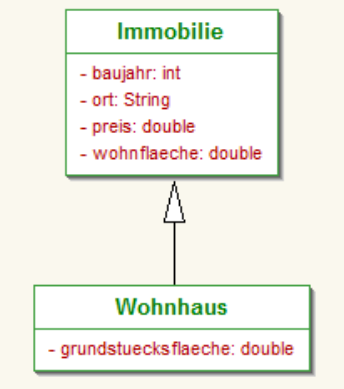
2 Entwickeln Sie eine Klasse mit main()-Methode (»StartImmobilie«), mit deren Hilfe ein Objekt (Ulm, Baujahr 2005, Wohnfläche 85 m², Preis 185.000,00 €) und ein Objekt (Esslingen, Baujahr 1996, Wohnfläche 180m², Preis 575.000,00 €, Grundstücksfläche 450 m²) erzeugt, die Attributwerte eingelesen und auf der Konsole angezeigt werden.

**Lösungsvorschlag zum Quellcode der Klasse »StartMitarbeiter« (projektMitarbeiter_1)**

```
package paketMitarbeiter;
public class StartMitarbeiter {
    public static void main(String[] args) {
 // Objekte erzeugen
        Mitarbeiter mitarb1 = new Mitarbeiter();
        Mitarbeiter mitarb2 = new Mitarbeiter();
        Mitarbeiter chef = new Mitarbeiter();
        Auszubildender azubi1 = new Auszubildender();
 // Setzen der Attributwerte
        mitarb1.setVorname("Frank");
        mitarb1.setName("Walser");
        mitarb1.setGehalt(1600);
        mitarb2.setVorname("Eva");
        mitarb2.setName("Flink");
        mitarb2.setGehalt(1650);
        chef.setVorname("Hans");
        chef.setName("Boss");
        chef.setGehalt(4000);
        azubi1.setVorname("Sven");
        azubi1.setName("Vogel");
        azubi1.setGehalt(450);
        azubi1.setAusbildungsberuf("Industriekaufmann");
        azubi1.setAusbildungsjahr("1. Jahr");
//  Auslesen der Attributwerte
        System.out.println("Mitarbeiter der GeLa GmbH");
        System.out.println(" -------------------------");
        System.out.println("Mitarbeiter    " + mitarb1.getVorname()+" "+
                                mitarb1.getName()+" "+ mitarb1.getGehalt() + " Euro");
        System.out.println("Mitarbeiter    " + mitarb2.getVorname()+" "+
                                mitarb2.getName()+" "+ mitarb2.getGehalt() + " Euro");
        System.out.println("Mitarbeiter    " + chef.getVorname()+" "+
                                chef.getName()+" "+ chef.getGehalt() + " Euro");
        System.out.println("Auszubildender " + azubi1.getVorname()+" "+
                                azubi1.getName()+" "+ azubi1.getGehalt() + " Euro");
        System.out.println("");
//  Gehalt erhöhen chef
        chef.gehalterhoehen(1000);
        System.out.println(" Neues Gehalt des Chefs: " + chef.getGehalt() + " Euro");
    }
}
```

## Lösungsvorschlag zur Übung (Seite 7)

### 1) UML-Klassendiagramm des Projektes *projektImmoGmbH*




### 2) Lösungsvorschlag zum Quellcode der Klasse »StartImmobilie« (projektImmoGmbH)

```
package paketImmobilie;
public class StartImmobilie {
    public static void main(String[] args) {

    //Erzeugen des Objektes ersteImmobilie
        Immobilie ersteImmobilie;
        ersteImmobilie = new Immobilie(2005, "Ulm", 185000, 85);

    //Erzeugen des Objektes erstesHaus
        Wohnhaus erstesHaus = new Wohnhaus();
        erstesHaus.setBaujahr(1996);
        erstesHaus.setGrundstuecksflaeche(450);
        erstesHaus.setOrt("Esslingen");
        erstesHaus.setWohnflaeche(180);
        erstesHaus.setPreis(575000);

    //Rückgabe der Attributwerte und Ausgabe in der Konsole
    //Daten der Wohnung
            System.out.println("Daten der Wohnung:");
            System.out.println("Ort:                " + ersteImmobilie.getOrt());
            System.out.println("Baujahr:            " + ersteImmobilie.getBaujahr());
            System.out.println("Wohnfläche:         " + ersteImmobilie.getWohnflaeche());
            System.out.println("Preis:              " + ersteImmobilie.getPreis());
            System.out.println(" ");

    //Daten des Wohnhauses
            System.out.println("Daten des Wohnhauses:");
            System.out.println("Ort:                " + erstesHaus.getOrt());
            System.out.println("Baujahr:            " + erstesHaus.getBaujahr());
            System.out.println("Wohnfläche:         " + erstesHaus.getWohnflaeche());
            System.out.println("Grundstücksgröße:   " + erstesHaus.getGrundstuecksflaeche());
            System.out.println("Preis:              " + erstesHaus.getPreis());
    }
}
```

# 5 Abstrakte Klasse

## 5.1 Ausgangssituation

Die GeLa GmbH beschäftigt neben den angestellten Mitarbeitern und Auszubildendern zunehmend auch Aushilfskräfte. Eine Analyse der betrieblichen Abläufe ergab, dass folgende Daten der Mitarbeiter verwaltet werden müssen:

Von allen Beschäftigten sind Vorname, Name und Gehalt zu erfassen. Bei den Auszubildendern werden zusätzlich die Informationen zum Ausbildungsberuf und dem Ausbildungsjahr benötigt. Von den Aushilfskräften muss die jeweils geleistete Arbeitszeit erfasst werden können. Bei den Angestellten sind das Jahr der Einstellung sowie der Urlaubsanspruch zu erfassen.

**Aufgaben:**

1 Entwickeln Sie ein UML-Klassendiagramm, das die beschriebenen Sachverhalte berücksichtigt.

2 Implementieren Sie eine Klasse mit main()-Methode, mit der ein Angestelltenobjekt, ein Auszubildendernobjekt und ein Aushilfeobjekt erzeugt, die entsprechenden Attributwerte zugewiesen und folgende Konsolenausgabe erzeugt werden.
(Projekt: projektMitarbeiter_2; Paket: paketMitarbeiter_2))

```
Mitarbeiterdaten

Angestellte:
Heinz Sauer     5000 Euro 28 Tage

Aushilfen:
Heidi Knaller   450  Euro 20 Tage

Auszubildende:
Sven Vogel      600  Euro Industriekaufmann 1. Jahr
```

Aus der Situationsbeschreibung ergibt sich, dass alle Mitarbeiterkategorien die Attribute Vorname, Name und Gehalt aufweisen müssen. Differenzierende Merkmale sind die Arbeitszeit bei den Aushilfen, Einstellungsjahr und Urlaubstage bei den Angestellten sowie Ausbildungsberuf und Ausbildungsjahr bei den Auszubildendern.

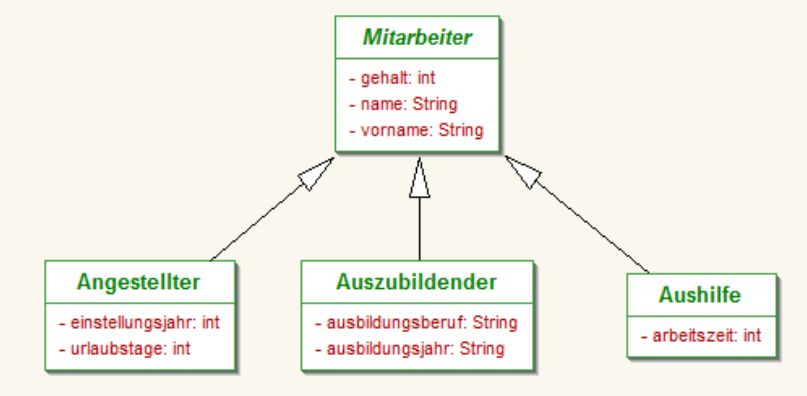
Es ergibt sich aus der Sachlogik, dass es im vorliegenden Projekt kein Objekt der Klasse »Mitarbeiter« geben muss, denn die entstehenden Objekte können nur Angestellte, Aushilfen oder Auszubildender repräsentieren. Diese Klasse, von der keine Objekte instanziiert werden sollen, stellt eine so genannte abstrakte Klasse dar.

Abstrakte Klassen spielen in Vererbungsstrukturen eine wichtige Rolle, wo sie die Gemeinsamkeiten einer Gruppe von Unterklassen definieren. Eine abstrakte Klasse, die keine konkrete Unterklasse hat, ist überflüssig. Von abstrakten Klassen können per Definition keine Objekte erzeugt werden. Mit ihr werden Vorgaben (Attribute, Methoden) festgelegt, die alle Unterklassen erben.

Im UML-Klassendiagramm ist der Klassenname einer abstrakten Klasse üblicherweise kursiv gedruckt.

## 5.2 Lösungshinweise zur Aufgabenstellung

### UML-Klassendiagramm zum Projekt projektMitarbeiter_2




### Quellcode einer abstrakten Klasse

In der Klassendefinition einer abstrakten Klasse muss diese als solche gekennzeichnet werden. Dies erfolgt mit der Anweisung:

| **public abstract class <<Superklasse>>** |
|---|

```
public abstract class Mitarbeiter {
    private String name;
    private String vorname;
    private int gehalt;
    public Mitarbeiter() {
    }
....
}
```

### Quellcode der Unterklassen

Unterklasse »Angestellter«

```
public class Angestellter extends Mitarbeiter {
    private int einstellungsjahr;
    private int  urlaubstage;
    public Angestellter () {
    }
....
}
```

Unterklasse »Auszubildender«

```
public class Auszubildender extends Mitarbeiter {
    private String ausbildungsberuf;
    private String ausbildungsjahr;
    public Auszubildender() {
    }
....
}
```

Unterklasse »Aushilfe«

```
public class Aushilfe extends Mitarbeiter {
    private int arbeitszeit;
    public Aushilfe () {
    }
....
}
```

### Quellcode der Klasse mit main()-Methode zum Projekt projektMitarbeiter_2

```
public class Startklasse
{
    public static void main(String[] args)
    {
     //Angestelltenobjekt
       Angestellter angestellter1 = new Angestellter();
       angestellter1.setVorname("Heinz");
       angestellter1.setName("Sauer");
       angestellter1.setGehalt(5000);
       angestellter1.setUrlaubstage(28);

     //Auszubildendenobjekt
       Auszubildender azubi1 = new Auszubildender();
       azubi1.setName("Vogel");
       azubi1.setVorname("Sven");
       azubi1.setGehalt(600);
       azubi1.setAusbildungsberuf("Industriekaufmann");
       azubi1.setAusbildungsjahr("1. Jahr");

     //Aushilfeobjekt
       Aushilfe aushilfe1 = new Aushilfe();
       aushilfe1.setVorname("Heidi");
       aushilfe1.setName("Knaller");
       aushilfe1.setGehalt(450);
       aushilfe1.setArbeitszeit(20);

     //Konsolenausgabe
       System.out.println("Mitarbeiterdaten");
       System.out.println("");
       System.out.println("Angestellte:");
       System.out.println(angestellter1.getVorname() + " " + angestellter1.getName()+ "\t" +
                          angestellter1.getGehalt() + " Euro " +angestellter1.getUrlaubstage() +"
Tage");
       System.out.println("");
       System.out.println("Aushilfen:");
       System.out.println(aushilfe1.getVorname() + " " + aushilfe1.getName()+ "\t" +
            aushilfe1.getGehalt() + "  Euro " +aushilfe1.getArbeitszeit() +" Tage");
       System.out.println("");
       System.out.println("Auszubildender:");
       System.out.println(azubi1.getVorname() + " " + azubi1.getName()+ "\t" +
            azubi1.getGehalt() + "  Euro " + azubi1.getAusbildungsberuf() + " " +
            azubi1.getAusbildungsjahr());
    }
}
```

## 5.3 Übungsaufgabe

Zur Verwaltung der Personendaten möchte die Schulleitung der Fritz-Erler-Schule Stuttgart eine objektorientierten Software einführen.

Die nachfolgenden Informationen sind dabei zu beachten:

- ➔ Die Schule beschäftigt Lehrer/-innen und Verwaltungspersonal. Die Beschäftigten sind mit Vorname, Nachname und Wohnort zu erfassen. Von den Lehrern ist darüber hinaus die Amtsbezeichnung, von dem Verwaltungspersonal das Monatsgehalt sowie der Urlaubsanspruch zu speichern.
- ➔ Die angemeldeten Schüler sind mit Vor- und Zuname, dem Wohnort sowie dem Jahr des Schuleintritts zu erfassen.

### Aufgabe

Sie erhalten den Auftrag, einen Vorschlag zur Realisierung der Datenerfassung zu unterbreiten. Dazu sollen Sie ein Datenmodell in Form eines UML-Klassendiagramms entwickeln. (Auf die Darstellung der Methoden soll hierbei verzichtet werden.)

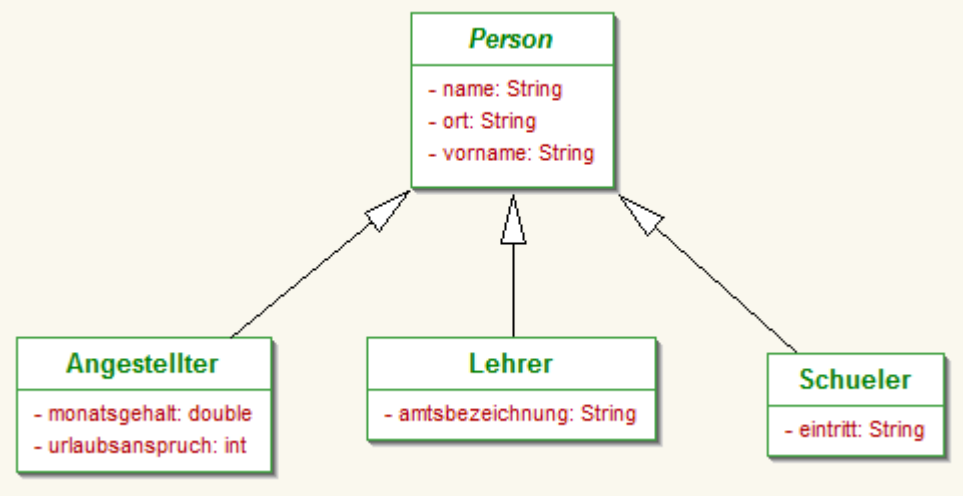
Projektname: *projektSchule*

### Lösungshinweise

Für das Projekt *projektSchule* ergeben sich daraus die Java-Anweisungen

- **public abstract class Person {…**
- **public class Lehrer extends Person {…**
- **public class Schueler extends Person {…**
- **public class Angestellter extends Person {…**

### UML-Klassendiagramm des Projekts *projektSchule*



# 6 Mehrstufige Vererbung

## 6.1 Ausgangssituation

Das Datenmodell der beruflichen Schule (siehe Übungsaufgabe projektSchule) soll um folgenden Sachverhalt erweitert werden:

Die Schule beschäftigt Lehrkräfte im Angestellten- und im Beamtenverhältnis. Von allen Lehrkräften ist die zu leistende Wochenarbeitszeit (Deputatsstunden) zu erfassen. Für die verbeamteten Lehrkräfte (Lehrer_B) ist darüber hinaus die Amtsbezeichnung, für die angestellten Lehrkräfte (Lehrer_A) sind die tarifliche Entgeltgruppe und die Krankenkasse, bei der die jeweilige Lehrkraft versichert ist, zu erfassen

(Hinweis: Unterstellt wird, dass es nicht beabsichtigt ist, angestellte Lehrer in ein Beamtenverhältnis zu übernehmen.)

### Aufgaben

1 Ergänzen Sie das UML-Klassendiagramm Ihres Projektes *projektSchule*.

2 Implementieren Sie die Fachklasse des erweiterten Projekts *projektSchule*.

3 Implementieren Sie eine Klasse mit main()-Methode, die folgende Objekte erzeugt, Attributwerte zuweist und die Daten auf der Konsole anzeigt.
(Aus Vereinfachungsgründen sollen nicht alle Attribute angesprochen werden!):

```
Daten Lehrer (verbeamtet):
Name:            Karl Huber
Ort:             Stuttgart
Stunden:         25
Amtsbezeichnung: Studienrat
```

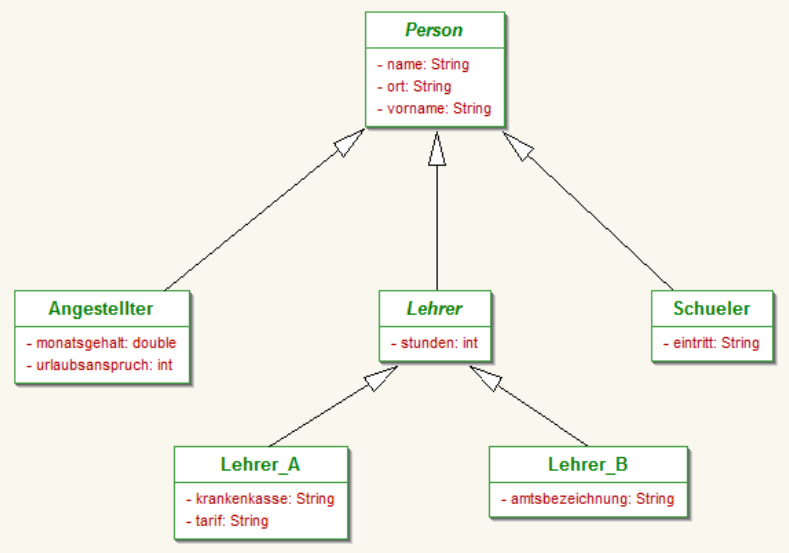
```
Daten Lehrer (angestellt):
Name:          Uwe Kandler
Ort:           Ostfildern
Stunden:       13
Tarif:         E 12
Krankenkasse:  Barmer GEK
```

```
Daten der Verwaltungsangestellten:
Name:      Franz Ganz
Ort:       Renningen
Gehalt:    2700.0
Urlaub:    28
```

```
Daten der Schüler:
Name:      Swen Vogel
Ort:       Ludwigsburg
Eintritt:  2012
```

## 6.2 Lösungshinweise zur Ausgangssituation

**Erweitertes UML-Klassendiagramm des Projekts *projektSchule***



Für das erweiterte Projekt *projektSchule* ergeben sich daraus die Java-Anweisungen

- **public abstract class Lehrer {…**
- **public class Lehrer_A extends Lehrer {…**
- **public class Lehrer_B extends Lehrer {…**

### Quellcode der Klasse mit main()-Methode - Projekt *projektSchule*

```
public class Startklasse {
    public static void main(String[] args)            {
          Lehrer_B ersterL = new Lehrer_B();
          Lehrer_A zweiterL = new Lehrer_A();
          Angestellter ersterA = new Angestellter();
          Schueler ersterS = new Schueler();
          ersterL.setName("Huber");
          ersterL.setVorname("Karl");
          ersterL.setOrt("Stuttgart");
          ersterL.setStunden(25);
          ersterL.setAmtsbezeichnung("Studienrat");
          zweiterL.setName("Kandler");
          zweiterL.setVorname("Uwe");
          zweiterL.setOrt("Ostfildern");
          zweiterL.setStunden(13);
          zweiterL.setTarif("E 12");
          zweiterL.setKrankenkasse("Barmer GEK");
          ersterA.setName("Ganz");
          ersterA.setVorname("Franz");
          ersterA.setOrt("Renningen");
          ersterA.setMonatsgehalt(2700);
          ersterA.setUrlaubsanspruch(28);
          ersterS.setName("Vogel");
          ersterS.setVorname("Sven");
          ersterS.setOrt("Ludwigsburg");
          ersterS.setEintritt("2012");
          System.out.println("Daten der Lehrer (verbeamtet:)");
          System.out.println("Name:      " + ersterL.getName());
          System.out.println("Ort:       " + ersterL.getOrt());
          System.out.println("Stunden:   " + ersterL.getStunden());
          System.out.println("Besoldung: " + ersterL.getAmtsbezeichnung());
          System.out.println(" ");
          System.out.println("Daten der Lehrer (angestellt:)");
          System.out.println("Name:      " + zweiterL.getName());
          System.out.println("Ort:       " + zweiterL.getOrt());
          System.out.println("Stunden:   " + zweiterL.getStunden());
          System.out.println("Besoldung: " + zweiterL.getTarif());
          System.out.println("Funktion:  " + zweiterL.getKrankenkasse());
          System.out.println(" ");
          System.out.println("Daten der Verwaltungsangestellten:");
          System.out.println("Name:      " + ersterA.getName());
          System.out.println("Ort:       " + ersterA.getOrt());
          System.out.println("Gehalt:    " + ersterA.getMonatsgehalt());
          System.out.println("Urlaub:    " + ersterA.getUrlaubsanspruch());
          System.out.println(" ");
          System.out.println("Daten der Schüler:");
          System.out.println("Name:      " + ersterS.getName());
          System.out.println("Ort:       " + ersterS.getOrt());
          System.out.println("Eintritt:  " + ersterS.getEintritt());
          }
}
```

## 6.3 Übung zur mehrstufige Vererbung

Eine Geschäftsbank möchte in Zukunft die Konten für Privatkunden mit einer objektorientierten Software verwalten. Zur Zeit können Privatkunden Girokonten und verschiedene Geldanlage-konten eröffnen.
Bei allen Konten sind der Name des Konteninhabers, die Kontennummer, der Guthabenzinssatz sowie der Kontostand zu erfassen. Für Girokonten müssen zusätzlich ein Kreditlimit sowie ein Überziehungszinssatz festgehalten werden können. Privatkunden können bei Geldanlagen zwischen Termingeldkonten und Sparplankonten wählen. Für alle Geldanlagekonten ist deren Laufzeit zu erfassen. Bei den Termingeldkonten muss darüber hinaus der Mindestanlagebetrag, bei den Sparplankonten die monatliche Sparrate gespeichert werden.

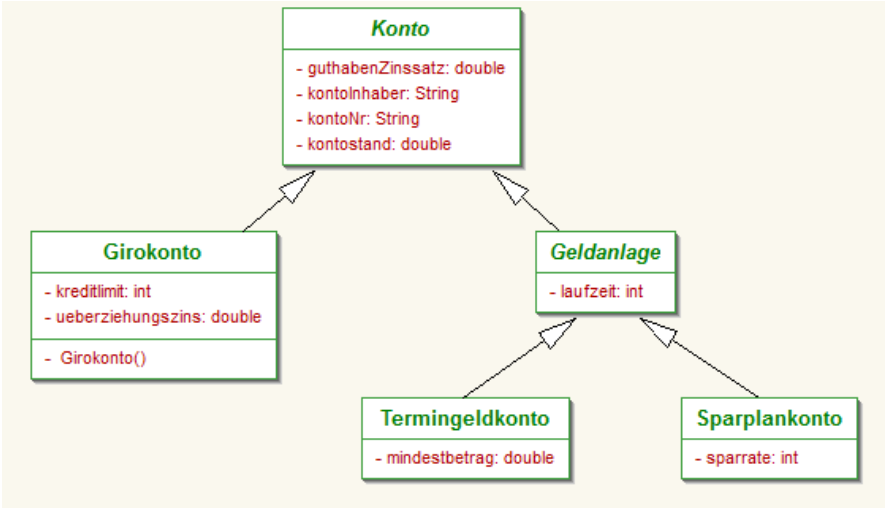
1 Modellieren Sie ein UML-Klassendiagramm zur Lösung der beschriebenen Sachverhalte. Nutzen Sie dabei die Möglichkeit der Generalisierung. (Projektname: projektGeschaeftsbank)

2 Erstellen Sie ein Objektdiagramm anhand der nachfolgenden Informationen:

Herr Schindel hat auf dem Konto 123987 einen aktuellen Kontostand von 50.000,00 Euro zu einem Zinssatz von 0,75% für 60 Monate angelegt. Die monatliche Sparrate beträgt 500,00 Euro.

### Lösungsvorschlag zur Übungsaufgabe

### UML-Klassendiagramm zum Projekt "Geschäftsbank"




### Objektdiagramm

**erstesKonto: Sparplankonto**
kontoNr = 123987
kontoInhaber = Schindel
kontostand = 50000
guthabenZinssatz = 0.75
anlagedauer = 60
sparrate = 500

# 7 Konstruktoren in der Vererbung

**Situation:**

Das Projekt *projektMitarbeiter_1* soll dahingehend erweitert werden, dass mit der Instanzierung eines neuen Objekts auch sofort die jeweiligen Attributwerte übergeben werden.

Zur Erinnerung: Mitarbeiter Heinz Sauer; 5.000,00 € Gehalt und Auszubildender Sven Vogel; 600,00 €; Industriekaufmann; 1. Ausbildungsjahr

In der Startklasse des Projekts Mitarbeiter_1 wird ein neues Objekt der Klasse »Mitarbeiter« mit folgender Anweisung erzeugt:

```
Mitarbeiter ersterMitarbeiter = new Mitarbeiter();
```

Der Ausdruck `Mitarbeiter()` ist eine besondere Methode, die den Namen der korrespondierenden Klasse trägt. Diese Methode erzeugt ein Objekt der Klasse und wird Konstruktor genannt. Ist der Konstruktor nicht definiert, so hat die Klasse dennoch einen Konstruktor, den man Standardkonstruktor nennt.

In der folgenden Tabelle sind mögliche Konstruktoren der Klasse »Mitarbeiter« aufgeführt. Sie unterscheiden sich durch den Aufruf. Die Reihenfolge der Übergabeparameter in der Klammer ist für den Aufruf entscheidend.

| Erklärung | Quellcode | Erzeugung |
|---|---|---|
| Ohne Übergabeparameter | `public Mitarbeiter() { }` | `mitarb1 = new Mitarbeiter();` |
| Parameter mit einem alphanumerischen Wert für *Name* | `public Mitarbeiter(String name) { this.name = name; }` | `mitarb1 = new Mitarbeiter ("Sauer");` |
| Parameter mit einem alphanumeri-Werte für *Name* und einem Zahlenwert für *Gehalt* | `public Mitarbeiter(String name, int gehalt) { this.name = name; this.gehalt = gehalt; }` | `mitarb1 = new Mitarbeiter ("Sauer", 5000);` |
| usw. | | |

In der Klammer stehen die Parameter. Sie sind Werte, die nach dem Aufruf des Konstruktors aus dem Hauptspeicher gelöscht werden. Sie werden an die neu zu erstellenden Objekte übergeben und dort in den jeweiligen Attributen abgespeichert.
Das Wort „this“ stellt in diesem Zusammenhang einen Verweis auf das aktuelle Objekt dar. Dies ist das gerade aktive Objekt.

Laut Aufgabenstellung muss der Quellcode des Projekts Mitarbeiter_1 nachfolgende Ergänzungen erhalten:

**Fachklasse »Mitarbeiter«**

```
//Attributdeklaration
.....
//Konstruktor
public Mitarbeiter(String vorname, String name, int gehalt)
{
   this.vorname = vorname;
   this.name = name;
   this.gehalt = gehalt;
}
//set- und get-Methoden
.....
```

**Fachklasse »Auszubildender«**

Im Gegensatz zu Methoden werden Konstruktoren nicht vererbt. Beim Aufruf des Konstruktors einer Unterklasse wird immer der Standard-Konstruktor der Oberklasse automatisch aufgerufen. Sofern in der Oberklasse parametrisierte Konstruktoren festgelegt wurden, muss dort ein zusätzlicher Konstruktor mit leeren Argumentationsliste definiert werden.

Wenn ein spezieller, parametrisierter Konstruktor der Superklasse ausgeführt werden soll, so muss dieser in der Unterklasse explizit mit dem Ausdruck **super()** aufgerufen werden. Da der parametrisierte Konstruktor der Superklasse Argumente erwartet, müssen diese mit den Aufruf des super-Konstruktors übergeben werden.

Für die Fachklasse Auszubildender ergibt sich folgender Quellcode:

```
//Attributdeklaration
.....
//Konstruktor
public Auszubildender(String vorname, String name, int gehalt,
String ausbildungsberuf, String ausbildungsjahr) {
      super(vorname, name, gehalt);
      this.ausbildungsberuf = ausbildungsberuf;
      this.ausbildungsjahr = ausbildungsjahr;
      }
//set- und get-Methoden
.....
```

Die Klasse »Auszubildender« kann nun die Methoden der Oberklasse verwenden. Auf die Attribute der Oberklasse wird mit den set- und get-Methoden zugegriffen.

Beispiel: Auszubildende erhalten bei überdurchschnittlichen Leistungen in der Berufsschule einen Bonus von 3 % ihrer Ausbildungsvergütung.

```
public double berechneBonus() {
     double bonus = this.getGehalt()* 0.03;
     return bonus;
}
```

Auf das Schlüsselwort „this“ kann in diesem Falle auch verzichtet werden.

### Startklasse

Der Aufruf zur Erzeugung der Objekte in der Startklasse erfolgt nun mit folgenden Anweisungen:

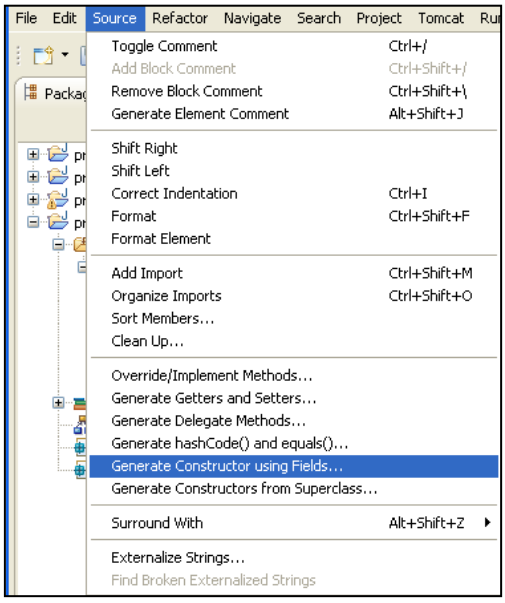
```
//Erzeugen des Objektes mitarb1
     Mitarbeiter mitarb1;
     mitarb1= new Mitarbeiter("Heinz", "Sauer", 5000);

//Erzeugen des Objektes ersterAuszubildender
     Auszubildender azubi1;
     azubi1= new Auszubildender("Sven", "Vogel", 600,
"Industriekaufmann",       "1. Jahr");
```

### Software gesteuertes Generieren der Konstruktoren

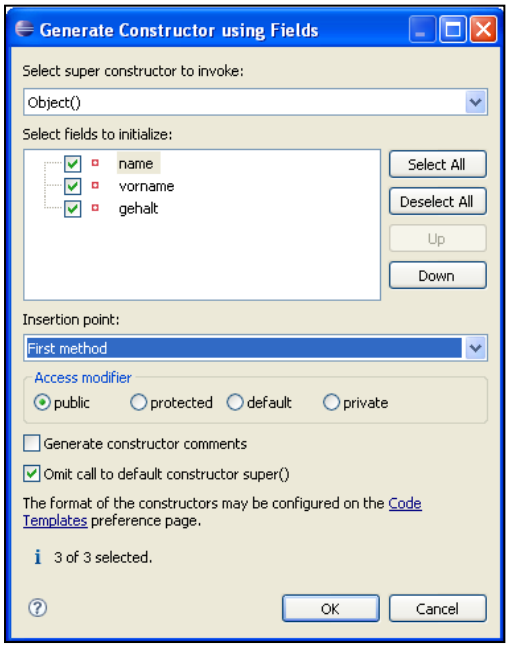
Der oben beschriebene Konstruktor der Klasse »Mitarbeiter« kann wie folgt generiert werden:

- Quellcode der Klasse »Mitarbeiter« aktivieren.
- Im Befehlsmenü die Auswahl
  *Source → Generate Constructor using Fields.….*
  treffen.




Die für den Konstruktor gewünschten Felder auswählen.

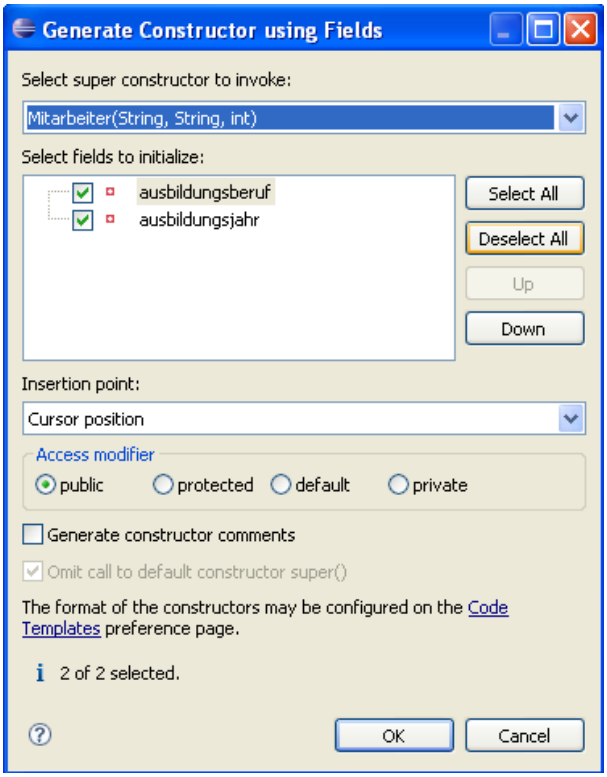
Damit der Konstruktor einer möglichen Oberklasse nicht aufgerufen wird, muss unten das Kontrollkästchen
„*Omit call to default constructor super()*“
aktiviert sein.



Der Konstruktor der Subklasse »Auszubildender« benötigt den Aufruf des Konstruktors der Superklasse. Das Kontrollkästchen

*"Omit call to default constructor super()",*

muss somit deaktiviert werden.




**Übungsaufgabe**

Das Projekt *Geschaeftsbank* soll dahingehend erweitert werden, dass beim Erzeugen eines Objekts vom Typ

- Girokonto auch die Werte für das Attribut kontoNr,
- Termingeldkonto auch die Werte für die Attribute kontoNr und mindestbetrag,
- Sparplankonto auch die Werte für die Attribute kontoNr, laufzeit und sparrate

übergeben werden.
Alle übrigen Attributwerte sind mit Hilfe der entsprechenden set-Methoden zu übergeben.

Implementieren Sie Ihre Lösung unter Verwendung folgender Testdaten:

```
Girokonto:
Kontonummer:    111 333 66
Kontoinhaber    Maier
Zinssatz        0.2
Kreditlimit:    50000

Termingeldkonto:
Kontonummer:    999 888 77
Kontoinhaber:   Schneider
Zinssatz:       1.0
Mindestbetrag:  10000.0

Sparkonto:
Kontonummer:    555 333 99
Kontoinhaber:   Maier
Zinssatz:       1.5
Laufzeit:       24
Sparrate:       100
```

**Lösungsvorschlag Übungsaufgabe**

**Folgende Konstruktoren sind anzulegen:**

**Superklasse »Konto«**

```
public Konto(String pKontoNr) {
      this.kontoNr = pKontoNr;
   }
```

**Subklasse »Girokonto«**

```
public Girokonto(String pKontoNr) {
      super(pKontoNr);
   }
```

**Superklasse »Geldanlage«**

```
public Geldanlage(String pKontoNr) {
      super(pKontoNr);
}

public Geldanlage(String pKontoNr, int pLaufzeit){
      super(pKontoNr);
      this.laufzeit = pLaufzeit;
}
```

**Subklasse »Termingeldkonto«**

```
public Termingeldkonto(String pKontoNr, int pMindestbetrag) {
      super(pKontoNr);
      this.mindestbetrag = pMindestbetrag;
   }
```

**Subklasse »Sparplankonto«**

```
public Sparplankonto(String pKontoNr, int pLaufzeit, int pSparrate) {
      super(pKontoNr, pLaufzeit);
      this.sparrate = pSparrate;
}
```

**Quellcode der Klasse »Startklasse«**

```
//Erzeugen des Objektes erstesGirokonto
   Girokonto erstesGiro;
   erstesGiro = new Girokonto("111 333 66");
   erstesGiro.setKontoInhaber("Maier");
   erstesGiro.setGuthabenzinssatz(0.2);
   erstesGiro.setKreditlimit(50000);

//Erzeugen des Objektes erstesTermingeldkonto
   Termingeldkonto erstesTermin;
   erstesTermin = new Termingeldkonto("999 888 77", 10000);
   erstesTermin.setKontoInhaber("Schneider");
   erstesTermin.setGuthabenzinssatz(1.0);

//Erzeugen des Objektes erstesSparkonto
   Sparplankonto erstesSpar;
   erstesSpar = new Sparplankonto("555 333 99", 24, 100);
   erstesSpar.setKontoInhaber("Huber");
   erstesSpar.setGuthabenzinssatz(1.5);

//Rückgabe der Attributwerte und Ausgabe in der Konsole
//Daten des Mitarbeiters
   System.out.println("Girokonto:");
   System.out.println("Kontonummer:   " + erstesGiro.getKontoNr());
   System.out.println("Kontoinhaber   " +
                                  erstesGiro.getKontoInhaber());
   System.out.println("Zinssatz       " +
                                  erstesGiro.getGuthabenZinssatz());
   System.out.println("Kreditlimit:   " + erstesGiro.getKreditlimit());
   System.out.println(" ");

   System.out.println("Termingeldkonto:");
   System.out.println("Kontonummer:   " + erstesTermin.getKontoNr());
   System.out.println("Kontoinhaber:  " +
                                  erstesTermin.getKontoInhaber());
   System.out.println("Zinssatz:      " +
                                  erstesTermin.getGuthabenZinssatz());
   System.out.println("Mindestbetrag: " +
                                  erstesTermin.getMindestbetrag());
   System.out.println(" ");

   System.out.println("Sparkonto:");
   System.out.println("Kontonummer:   " + erstesSpar.getKontoNr());
   System.out.println("Kontoinhaber:  " +
                                  erstesGiro.getKontoInhaber());
   System.out.println("Zinssatz:      " +
                                  erstesSpar.getGuthabenZinssatz());
   System.out.println("Laufzeit:      " + erstesSpar.getLaufzeit());
   System.out.println("Sparrate:      " + erstesSpar.getSparrate());  }
```

# 8 Zusammenfassende Übungen

**Aufgabe 1**

Ein Softwarehaus wird beauftragt, die betrieblichen Abläufe des KFZ-Vermietung Ritter KG in ein Softwaresystem abzubilden. Hierzu soll in einem ersten Schritt mit Hilfe der UML eine objektorientierte Analyse durchgeführt werden.

In einem Projektgespräch stellt die Ritter OHG dem Softwarehaus folgende Informationen bezüglich der zu modellierenden Klassen zur Verfügung:

- ➔ Die Ritter KG vermietet PKWs, Transporter und Motorräder.
- ➔ Herr Ritter beschäftigt acht Mitarbeiter. Davon sind drei Mitarbeiter in der Werkstatt beschäftigt. Bei den übrigen Mitarbeitern handelt es sich um kaufmännische Angestellte. Drei von ihnen sind im Außendienst und fünf im Innendienst beschäftigt.
- ➔ Innendienstmitarbeitern kann Prokura erteilt werden.
- ➔ Die drei Mitarbeiter der Werkstatt erhalten einen Stundenlohn, die übrigen Mitarbeiter erhalten ein monatliches Festgehalt.
- ➔ Von allen Mitarbeitern sind Vorname, Name, Anschrift mit Straße, Postleitzahl und Ort sowie das jeweilige Entgelt zu erfassen.
- ➔ Jedem Außendienstmitarbeiter wird ein Mobiltelefon zur Verfügung gestellt, dessen Rufnummer gespeichert werden soll.
- ➔ Bei den Innendienstmitarbeitern muss erkennbar sein, ob ihnen Prokura erteilt wurde oder nicht.
- ➔ Von den Vermietgegenständen sind der Hersteller, der Fahrzeugtyp, das amtliche Kennzeichen und das Baujahr zu speichern.
- ➔ Für PKWs sollen vorhandene Besonderheiten (Cabrio, Anhängerkupplung etc.) erfasst werden können.
- ➔ Bei den Transportern sind zusätzlich die Größe der Ladefläche und die Anzahl der Sitzplätze zu speichern.
- ➔ Die Motorräder sind mit ihrem Hubraum in ccm zu erfassen.

Entwickeln Sie ein UML-Klassendiagramm, das die beschriebenen Sachverhalte berücksichtigt. Nutzen Sie dabei die Möglichkeit der Vererbung.

**Aufgabe 2**

Um die Abläufe in der Bäckerei Weizmann zu unterstützen, soll eine objektorientierte Software eingeführt werden. In einem Interview, das Sie mit dem Produktionsleiter führen, ergibt sich die Notwendigkeit, den Backwarenstamm neu zu strukturieren.

Die Backwaren lassen sich in Konditoreiwaren, Brote und Frühstücksbackwerk einteilen.

Für Konditoreiwaren sind insbesondere die notwendige Kühltemperatur und das Mindesthaltbarkeitsdatum zu speichern.

Bei Broten sind Gewicht und Mehlsorte zu erfassen.

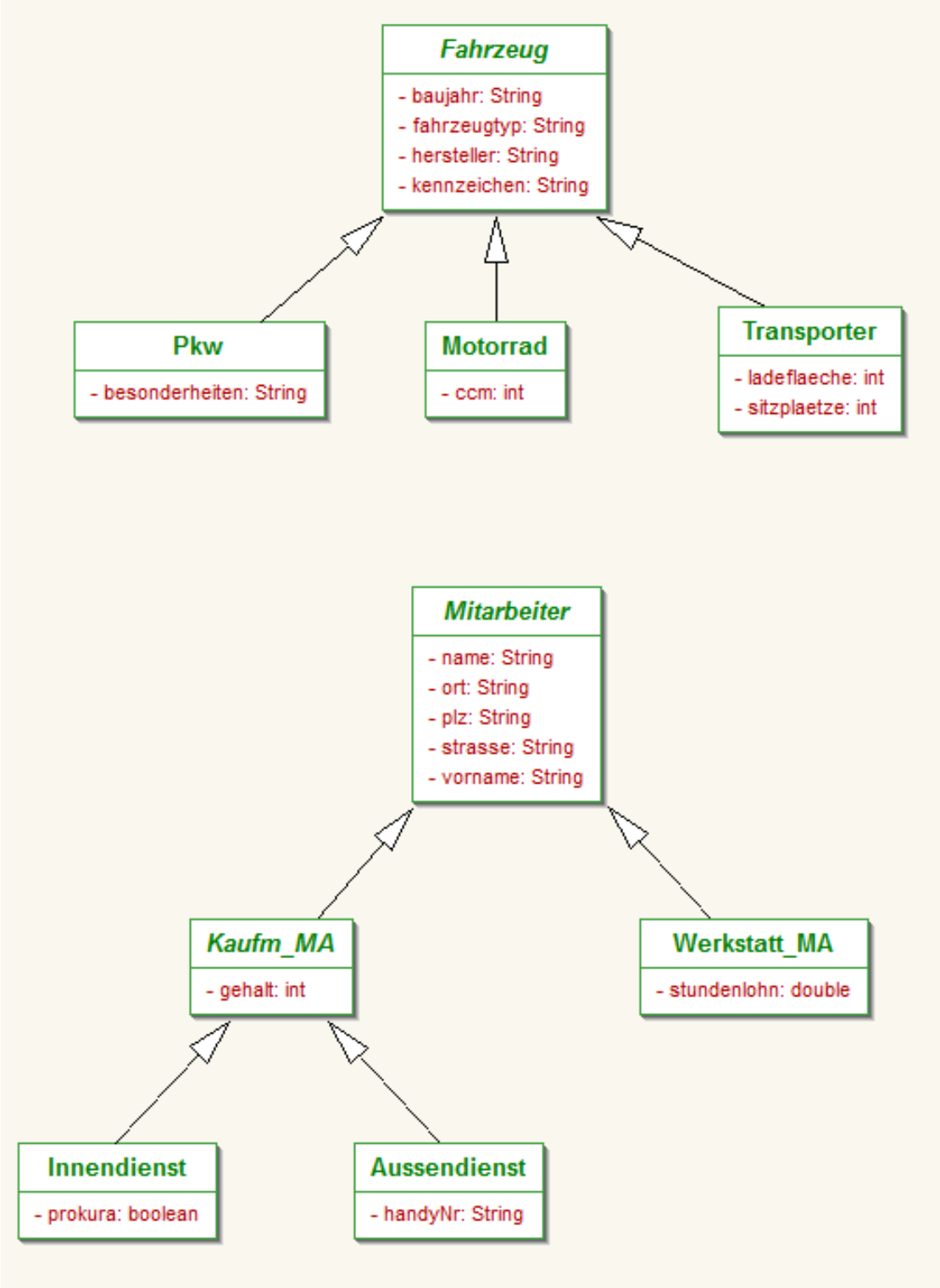
Das Frühstücksbackwerk lässt sich wiederum in Salz- und Süßbackwaren unterteilen. Bei Salzbackwaren wird Salzgehalt in Prozent gespeichert. Süßbackwaren werden mit Zuckergehalt und Lagertemperatur gespeichert. Bei beiden Warengruppen ist die Abkühldauer zu erfassen.

Gemeinsam sind allen Backwaren eine aussagekräftige Bezeichnung und ein Verkaufspreis.

Erstellen Sie einen objektorientierten Entwurf in Form eines Klassendiagramms. Beachten Sie hierbei die Möglichkeit der Generalisierung.

## Lösungsvorschlag zur Übungsaufgabe 1

## UML-Klassendiagramm zum Projekt "KFZ_Vermietung"



## Lösungsvorschlag zur Übungsaufgabe 2

## UML-Klassendiagramm zum Projekt "Bäckerei Weizmann"