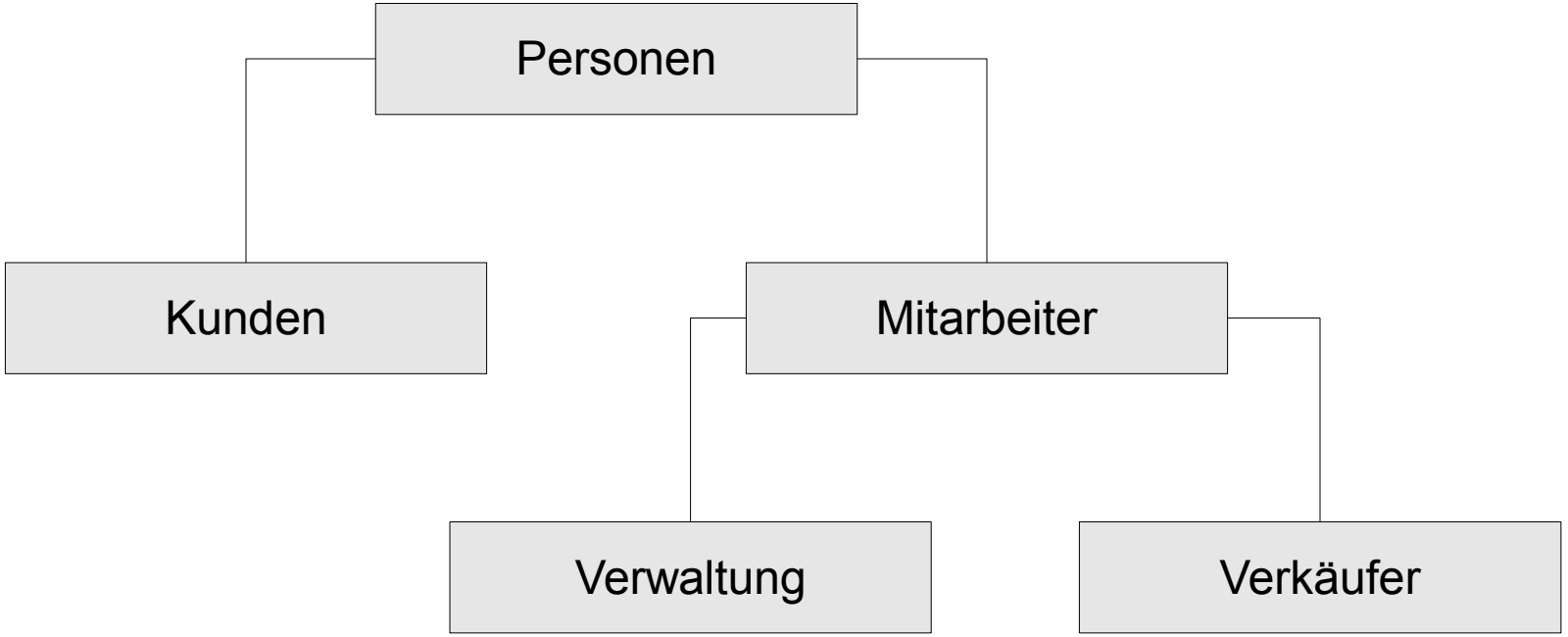
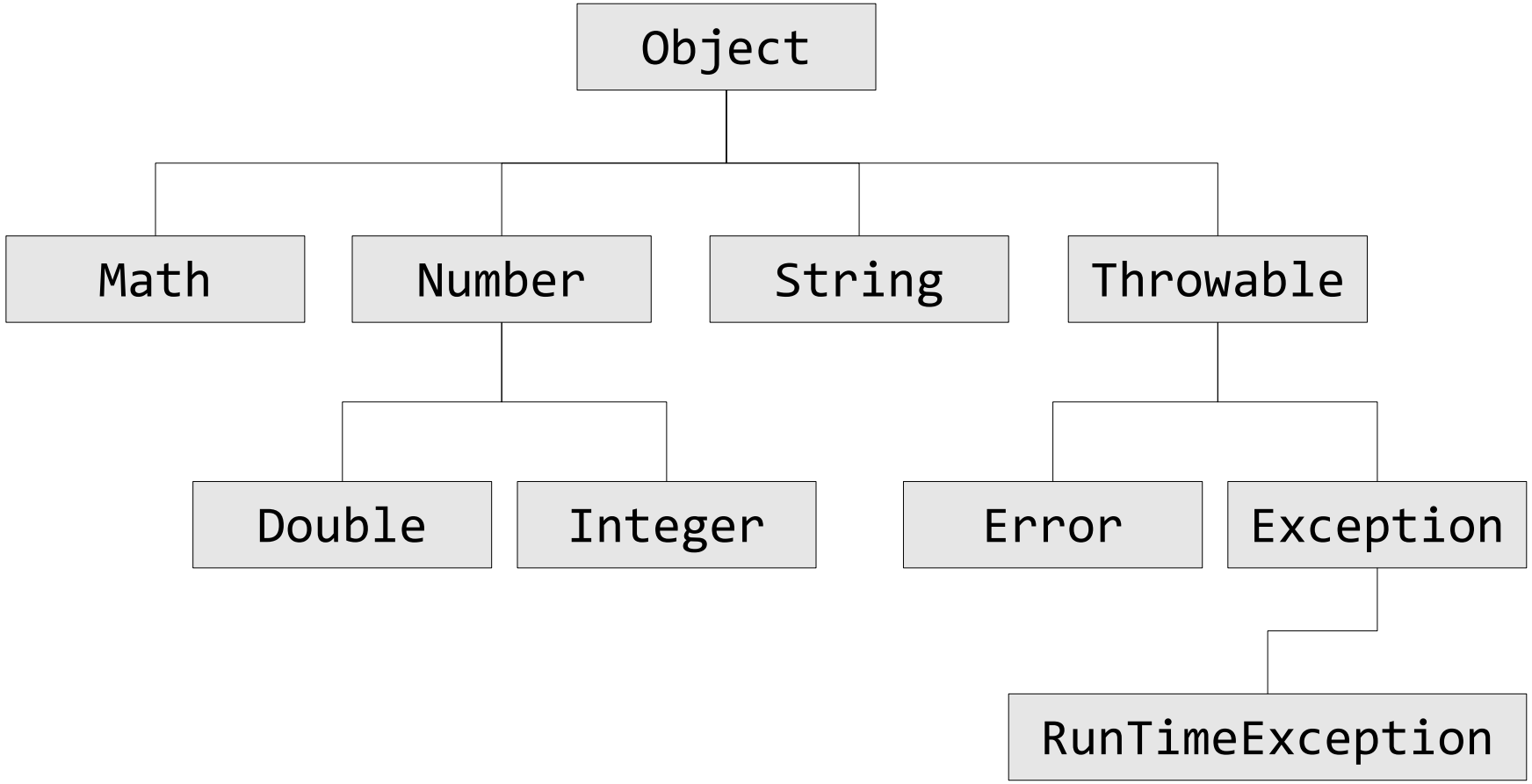
Java - Vererbung



Vererbung

- Definition von Klassen auf Basis von bestehenden Klassen.
- Eine Klasse erbt Eigenschaften und Methoden von einer anderen Klasse.
- Implementierung von „ist ein“-Beziehungen wie zum Beispiel „Das Auto ist ein Kraftfahrzeug“.
- Eltern-Kind-Beziehung.
- Abbildung der hierarchischen Struktur von Klassen.
- Implementierung von „ist ein“-Beziehung.
- Eine Mehrfachvererbung ist nicht möglich.

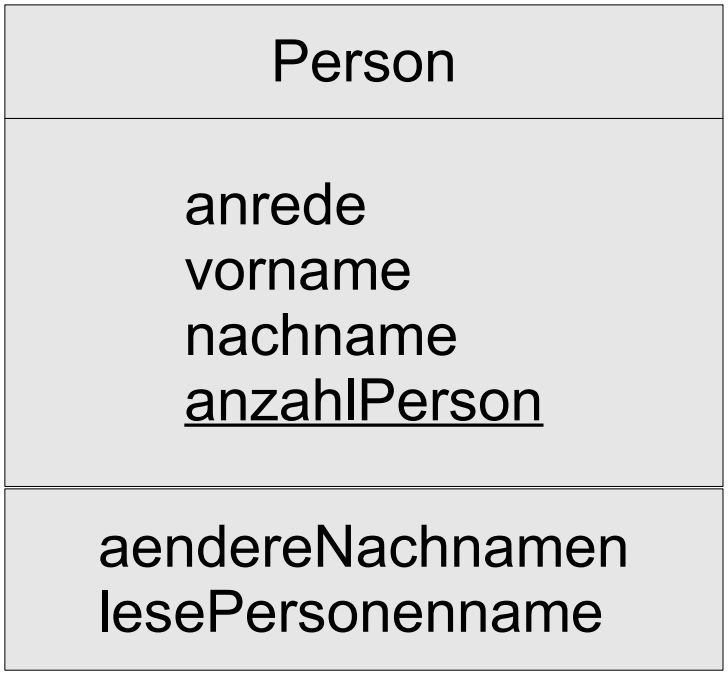
Ausschnitt aus dem Paket „java.lang“



Basisklasse

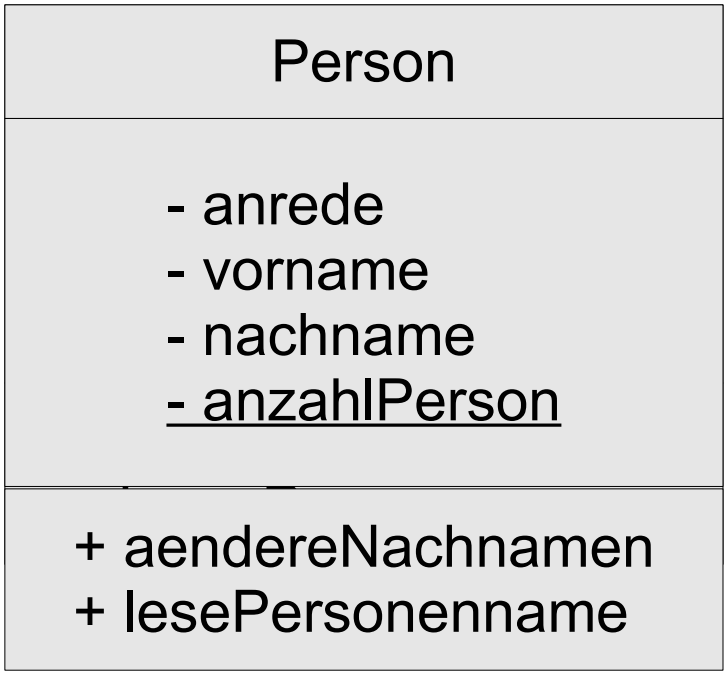
- Oberklasse, Elternklasse, Superklasse.
- Implementierung von „Oberbegriffen“ für eine Objektgruppe.
- Allgemeine Beschreibung eines Objekts, die an eine andere Klasse weitergegeben werden kann.

Klassendiagramm „Person“



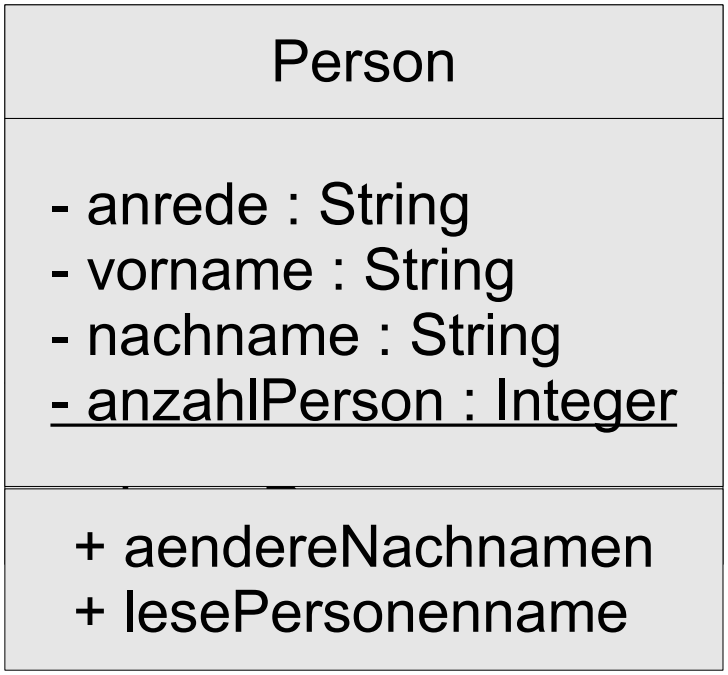
- Attribute beginnen mit einem Kleinbuchstaben.
- Klassenattribute werden unterstrichen.
- Methodennamen sollten ein Verb enthalten, welches die Aktion beschreibt.

Sichtbarkeit von Attributen und Methoden



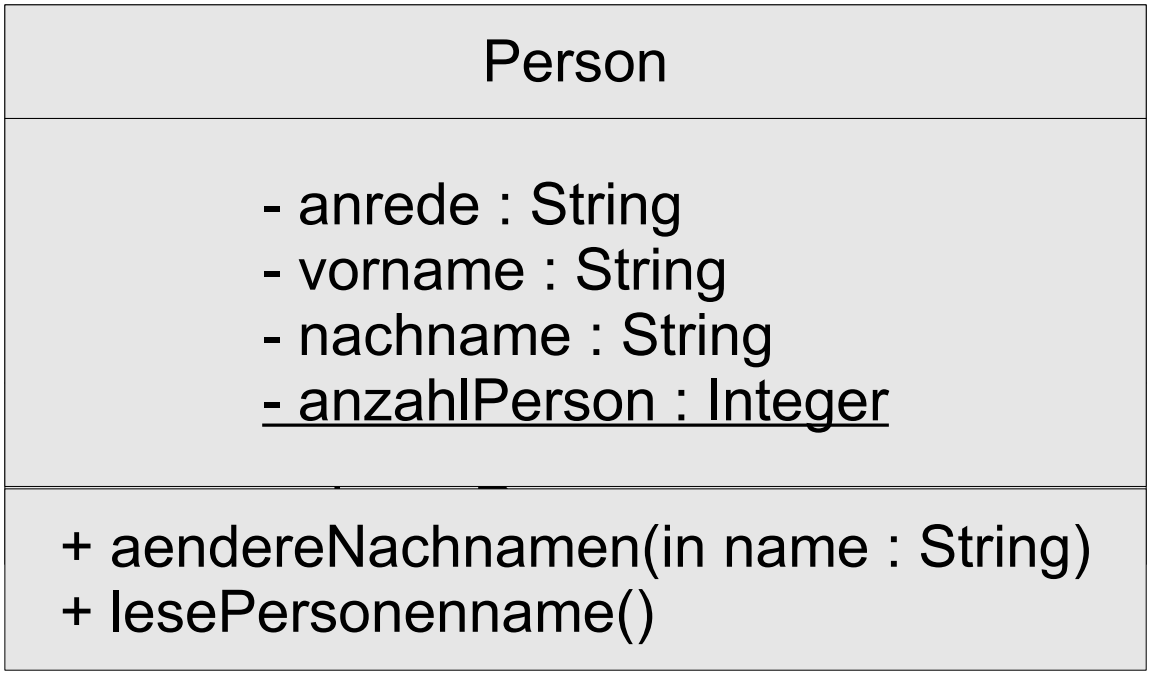
- Private Attribute werden mit einem Minuszeichen gekennzeichnet.
- Öffentliche Methoden werden mit einem Pluszeichen gekennzeichnet.

Datentyp von Attributen



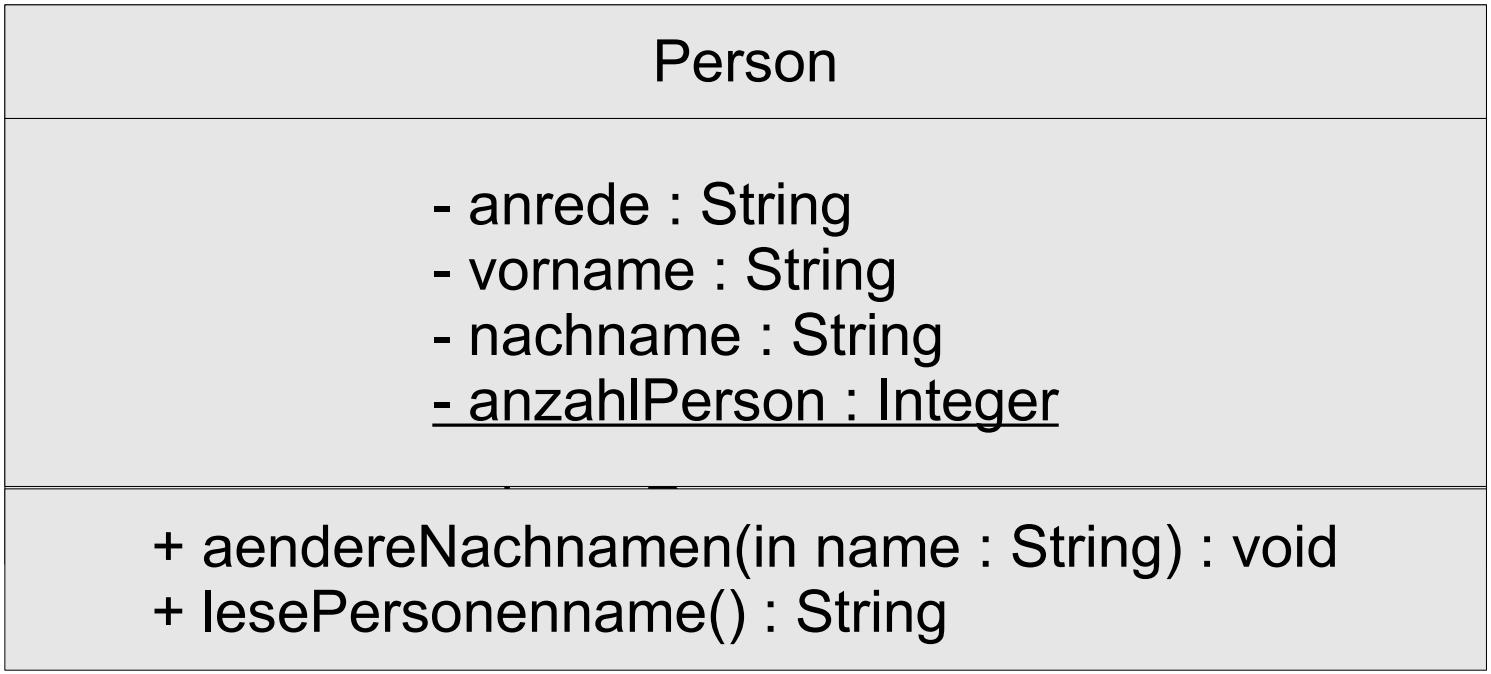
- Ein Attribut und der dazugehörige Datentyp werden durch einen Doppelpunkt getrennt.

Parameterliste von Methoden



- Die Richtung des Parameters wird mit dem Wort in angegeben.
- Der Parameter und der dazugehörige Datentyp werden durch ein Doppelpunkt getrennt.

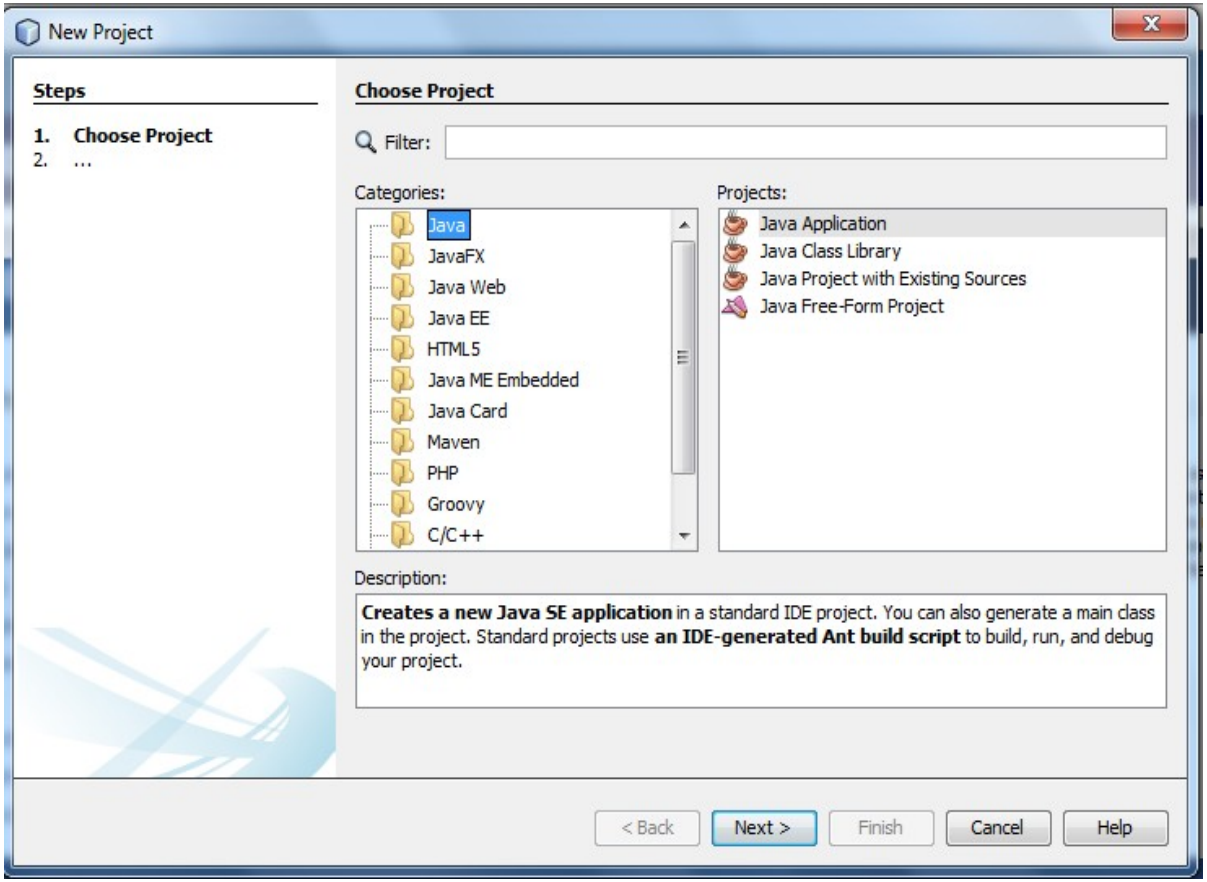
Rückgabewert von Methoden



- Der Parameterliste folgt der Datentyp des Rückgabewertes.
- Der Datentyp des Rückgabewertes wird durch den Doppelpunkt einer Methode zugewiesen.

Java-Projekt anlegen

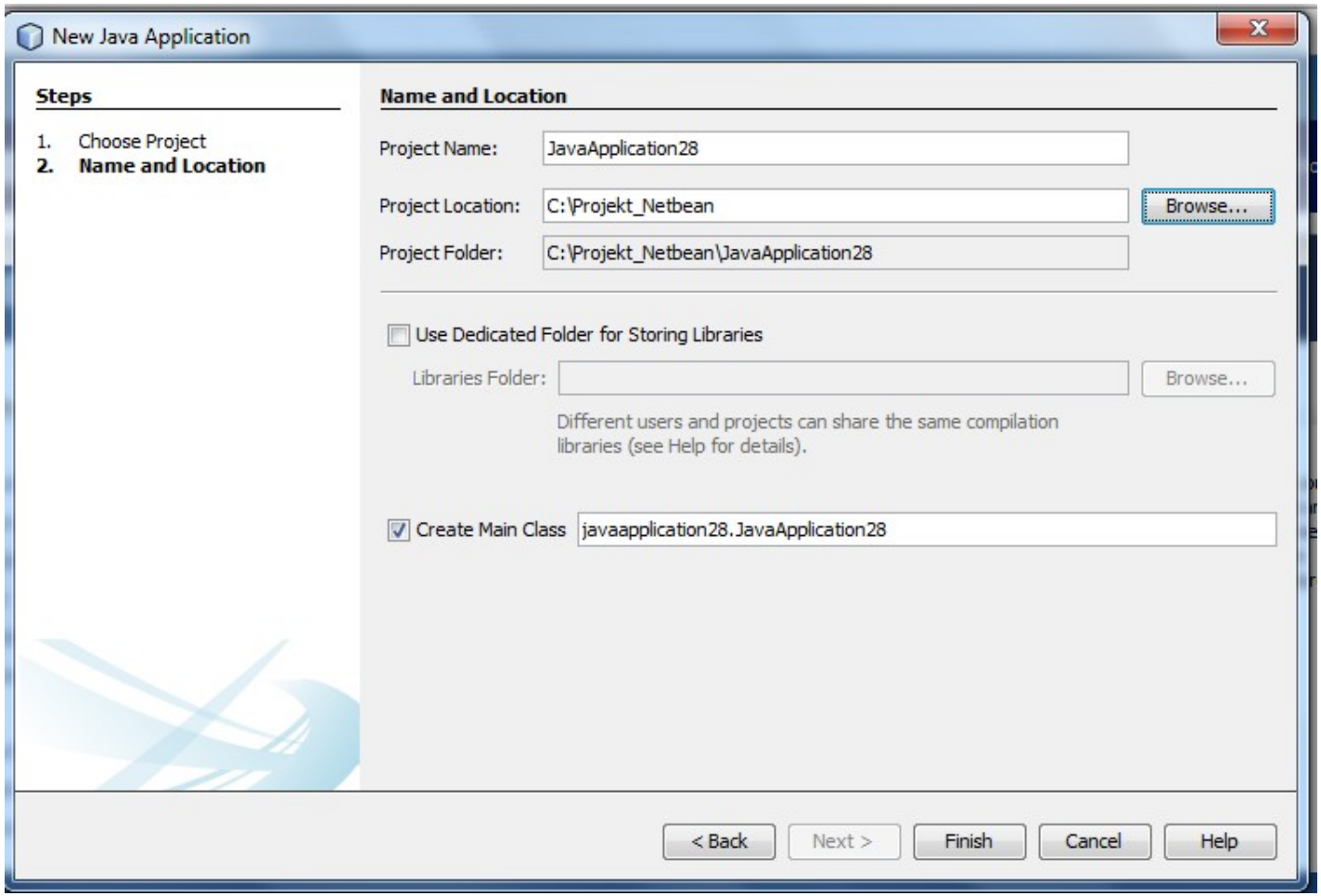
- *File – New Project*.



1. Schritt: Auswahl einer Projektkategorie

- *Categories* „Java“. *Projects* „Java Application“.
- Das Grundgerüst eines Projekts wird entsprechend der ausgewählten Kategorie erzeugt.
- In diesem Beispiel werden die Ordner und Dateien für eine Konsolen-Anwendung automatisiert erstellt.

2. Schritt: Name und Speicherort des Projekts



Erläuterung

- *Project Name* legt den Namen des Ordners fest. Der Paketname des Java-Projekts wird festgelegt.
- *Project Location* und *Project Folder* legen den Speicherort des Java-Projekts fest.
- *Create Main Class* erzeugt die Klasse, die die Start-Methode enthält. Der Klassennamen beginnt mit einem Großbuchstaben und entspricht standardmäßig dem Paketnamen.

Hinzufügung einer Klasse

- Rechter Mausklick auf den Projektnamen im Projekt-Explorer.
- *New* – *Java Class*.
- Eingabe eines Klassennamens in das Textfeld *Class Name.* Klassennamen beginnen immer mit einem Großbuchstaben.
- Klick auf die Schaltfläche *Finish*. Das Gerüst einer Klasse wird automatisiert erstellt.

Attribute einer Basisklasse

```
public class clsBasis_Person {
    private static int anzahlPerson;
    private String vorname;
    private String nachname;
    private String anrede;

}
```

Attribute

- Instanzvariablen sind an die Existenz eines Objekts gebunden. Die Variablen beschreiben ein Objekt. Jede Instanz einer Klasse unterscheidet sich in mindestens einem Attribut-Wert von allen andern Instanzen.
- Klassenvariablen sind an die Existenz einer Klasse gebunden. Klassenvariablen beschreiben Attribute, die von allen Instanzen einer Klasse gemeinsam genutzt werden. Klassenvariablen sind statische Attribute.
- Ein Zugriff von außen direkt auf Instanzvariablen und Klassenvariablen ist durch das Schlüsselwort `private` nicht möglich.

Konstruktoren einer Basisklasse

```
public class clsBasis_Person {
    public clsBasis_Person(String anrede,
                String vorname, String nachname){
        this.anrede = anrede;
        this.vorname = vorname;
        this.nachname = nachname;
        clsBasis_Person.anzahlPerson++;
    }

    public clsBasis_Person(){
        this("", "", "");
    }

    public clsBasis_Person(String anrede, String nachname){
        this(anrede, "", nachname);
    }
}
```

Konstruktoren

- Erzeugen eine Instanz von einer Klasse.
- Spezielle Methoden die durch das Schlüsselwort `new` automatisiert aufgerufen werden.
- Setzen von Anfangswerten für eine Instanz von einer Klasse.

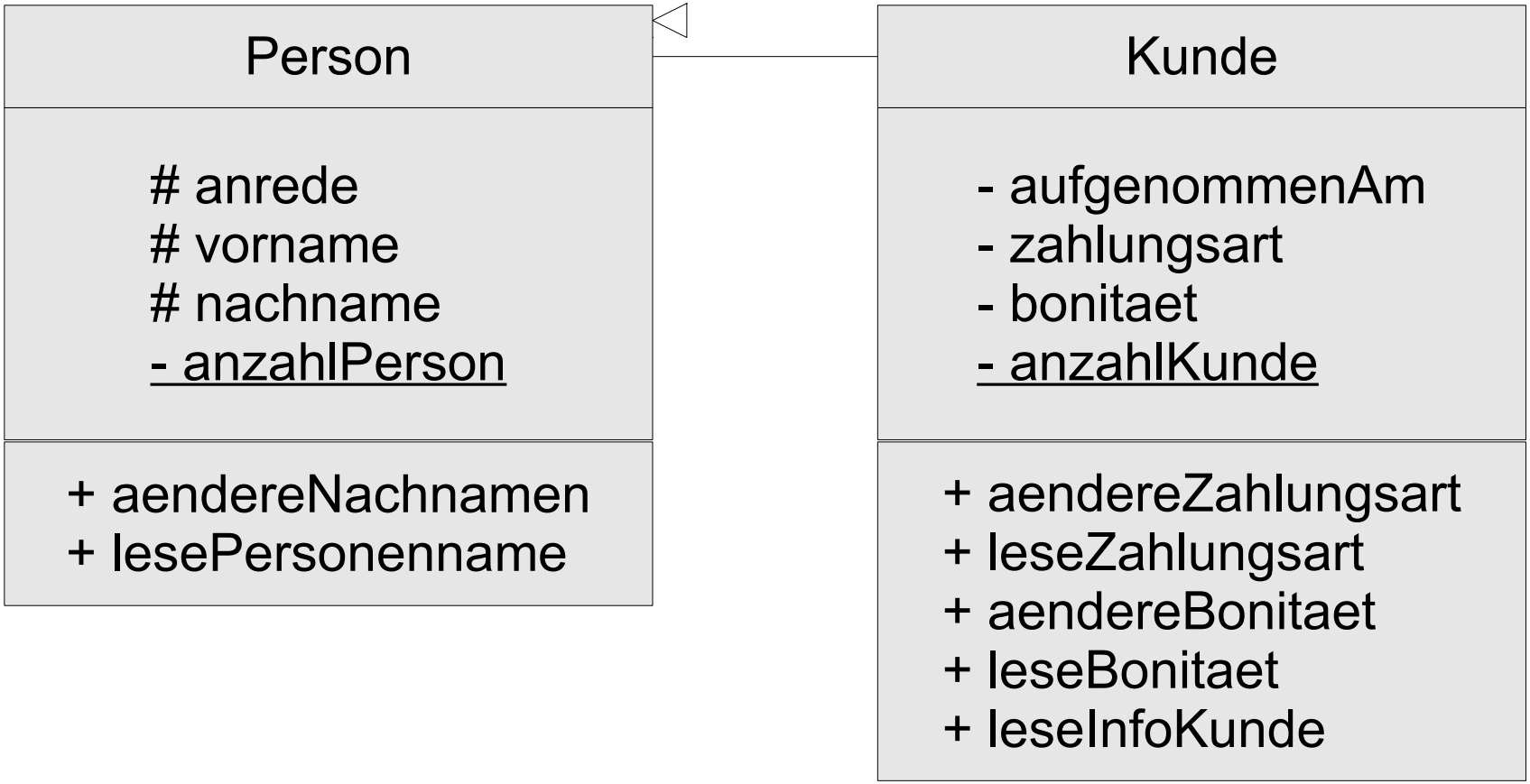
Methoden

- Tätigkeiten, die eine Instanz ausführen kann.
- Statische Klassenmethoden können ohne das eine Instanz der Klasse existiert ausgeführt werden.
- Methoden sind häufig öffentlich. Methoden können außerhalb der Klasse aufgerufen werden.

Subklasse

- Unterklasse, Kindklasse.
- Spezialisierung einer Basisklasse.
- Die Klasse erbt Attribute und Methoden aus der Basisklasse.
- Die Klasse kann geerbte Methoden überschreiben.
- In einer „ist ein“- Beziehung wie in dem Satz „Der Kunde ist eine Person.“ beschreibt die Subklasse das Subjekt. Die Basisklasse wird als Objekt des Satzes abgebildet.

Klassendiagramm „Kunde“



Klassendiagramm „Mitarbeiter“

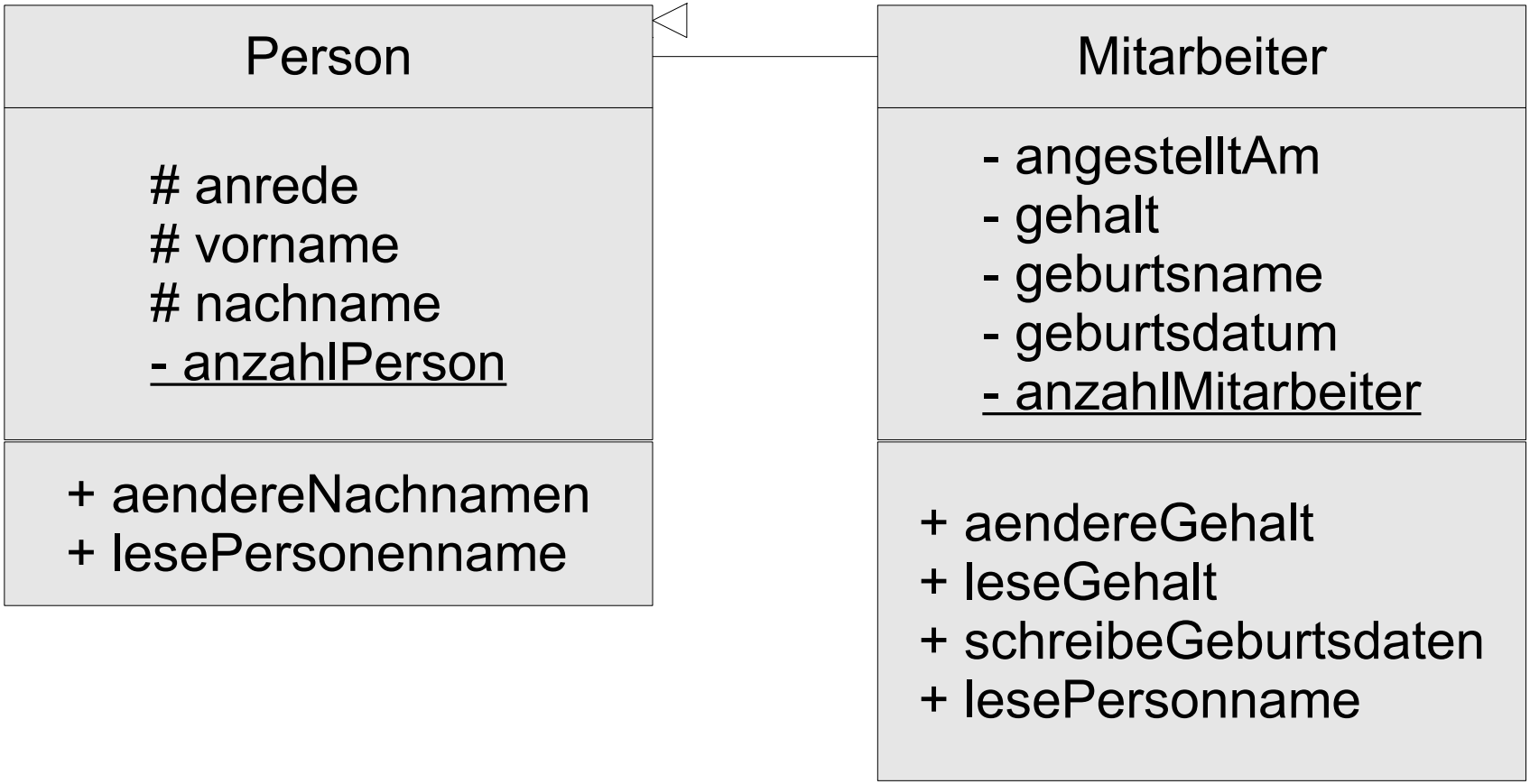
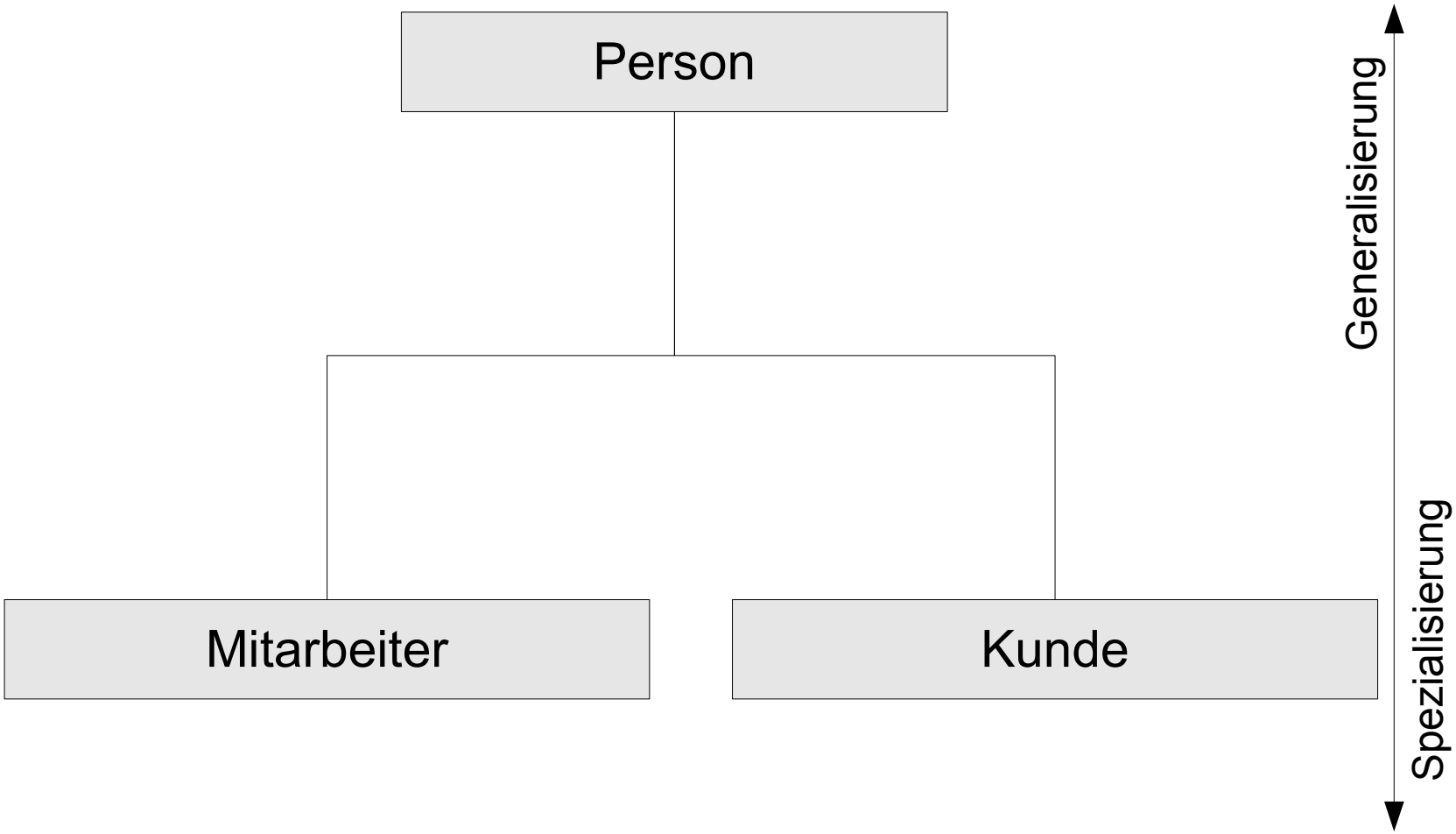


Abbildung als hierarchisches Modell



Hinweise

- Beschreibung einer hierarchischen Struktur.
- Von der Wurzel nach unten werden die Klassen immer spezieller. Die Beschreibung einer Instanz wird immer detaillierter.
- Von den Blättern zur Wurzel werden die Klassen immer allgemeiner. Die Beschreibung einer Instanz wird generalisiert. Objektarten werden mit Hilfe eines Oberbegriffs zusammengefasst.

Subklasse

```
public class clsSub_Mitarbeiter extends clsBasis_Person{
    String angestelltAm;
    String geburtsdatum;
    String geburtsname;
    BigDecimal gehalt;

    public clsSub_Mitarbeiter(String anrede,
     String nachname, String angestelltAm, String gehalt)
    {
    }

   public String leseGehalt(String gehalt){
        return(this.gehalt.toString());
    }
 }
```

Attribute / Eigenschaften

Konstruktor

Methoden

Kopf einer Subklasse

```
public class clsSub_Mitarbeiter
        extends clsBasis_Person{
```

- Jede Klasse wird durch das Schlüsselwort `class` gekennzeichnet.
- Jede Klasse hat einen eindeutigen Namen. In diesem Beispiel wird die Klasse `clsSub_Mitarbeiter` implementiert.
- Vor dem Namen wird ein Zugriffsmodifikator angegeben.
- Nach den Namen wird die Basisklasse angegeben.

Klassenname

- Beginn mit einem Großbuchstaben. Zum Beispiel „Mitarbeiter“.
- Nutzung der Kamel-Notation. Zum Beispiel „MitarbeiterIstPerson“. Jedes Wort beginnt mit einem Großbuchstaben.
- Der Name spiegelt die Bezeichnung eines Dinges in der realen Welt wieder.
- Der Name ist innerhalb eines Paketes eindeutig.

Angabe der Basisklasse

```
public class clsSub_Mitarbeiter
        extends clsBasis_Person{
```

- Dem Schlüsselwort `extends` folgt der Name der Basisklasse.
- Die Subklasse erweitert die Basisklasse.

Öffentliche Klassen

```
public class clsSub_Mitarbeiter
        extends clsBasis_Person{
```

- Das Schlüsselwort `public` kennzeichnet öffentliche Klassen.
- Die Klasse kann von jedem Paket genutzt werden.
- Die Klasse kann von allen anderen Klassen verwendet werden.

Private Klassen

```
private class clsSub_Mitarbeiter
        extends clsBasis_Person{
```

- Das Schlüsselwort `private` kennzeichnet eine private Klasse.
- Wenn kein Zugriffsmodifikator angegeben wird, ist die Klasse nicht öffentlich.
- Die Klasse kann nur in dem Paket genutzt werden, in dem sie definiert ist.

Attribute einer Subklasse

```
public class clsSub_Mitarbeiter
       extends clsBasis_Person{

    final double MINDESTGEHALT = 1500.0;
    String angestelltAm;
    String geburtsdatum;
    String geburtsname;
    BigDecimal gehalt;

    static int anzahlMitarbeiter;
```

Deklaration von Instanzvariablen

`String`	`angestelltAm`	`;`
`typ`	`variablenname`	`;`

- Der Variablenname ist frei wählbar.
- Entsprechend des angegebenen Datentyps wird Speicher bereitgestellt.
- Es können primitive Datentypen oder Klassen genutzt werden.
- Die Variablen können im Konstruktor initialisiert werden. Variablen werden in Methoden verändert.

Deklaration von Klassenvariablen

`static`	`int`	`anzahlMitarbeiter`	`;`
`static`	`typ`	`variablenname`	`;`

- Klassenvariablen beginnen mit dem Schlüsselwort `static`.
- Klassenvariablen sind abhängig von der Existenz der Klasse.
- Klassenvariablen beschreiben Attribute, die alle Instanzen einer Klasse gemeinsam nutzen.

Deklaration von Konstanten

`final`	`double`	`MINDESTGEHALT`	`=`	`1500.0`	`;`
`final`	`Datentyp`	`Konstanten-Name`	`=`	`wert`	`;`

- Deklarationen von Konstanten beginnen mit dem Schlüsselwort `final`.
- Konstanten können von Methoden nicht verändert werden.
- Konstanten können gleichzeitig deklariert und initialisiert werden. Konstanten können aber auch in einem Konstruktor initialisiert werden.

Zugriffsmodifikatoren für Instanzvariablen

- Wenn kein Zugriffsmodifikator angegeben ist, kann das Attribut von allen Objekten aller Klassen im selben Paket genutzt werden.
- Private Attribute (`private`) können von keinen Objekten verwendet werden. Sie sind in der Klasse gekapselt. Die Attribute können nur mit Hilfe von öffentlichen Methoden verändert werden.
- Öffentliche Attribute (`public`) können von allen anderen Objekten verwendet werden. Attribute in einer Klasse sollten nie öffentlich sein.

Attribute einer Basisklasse

```
public class clsBasis_Person {
    private static int anzahlPerson;
    protected String vorname;
    protected String nachname;
    protected String anrede;
```

Hinweise

- Die Subklassen der Basisklasse erbt alle Instanzvariablen.
- Klassenvariablen werden nicht vererbt.

Zugriffsmodifikatoren für Instanzvariablen

- Wenn kein Zugriffsmodifikator angegeben ist, kann das Attribut von allen Objekten aller Klassen im selben Paket genutzt werden.
- Attribute in Basisklassen können wie in der Subklasse privat (`private`) oder öffentlich (`public`) sein.
- Geschütze Attribute (`protected`) können nur von Subklassen der Basisklasse und der Basisklasse selbst genutzt werden. Sie sind aber in der Basisklasse gekapselt. Von außen können die Attribute nur mit Hilfe von Methoden verändert werden.

Zusammenfassung

Definition	Zugriff aus			
in der Basisklasse	der Basisklasse	der Subklasse	dem gleichen Paket	einem beliebigen Paket
Keine Angaben	•	Im gleichen Paket	•	
`private`	•			
`public`	•	•	•	•
`protected`	•	•	•	

Methoden einer Basisklasse

```
public class clsBasis_Person {
    public void aendereNachname(String nachname){
        this.nachname = nachname;
    }
    public String lesePersonname(){
        String ausgabe;

        ausgabe = this.anrede + " " + this.vorname;
        ausgabe = ausgabe.trim();
        ausgabe = ausgabe + " " + this.nachname;
        ausgabe = ausgabe.trim();
        return (ausgabe);
    }
    public static String getAnzahlPersonen(){
        return (clsBasis_Person.anzahlPerson + " Personen");
    }
}
```

Methoden einer Subklasse

```
public class  clsSub_Mitarbeiter {
    private void setzeGehalt(String gehalt)
    {
        this.gehalt = new BigDecimal(MINDESTGEHALT);
    }

    public void aendereGehalt(String gehalt)
    {
        setzeGehalt(gehalt);
    }

    public String leseGehalt()
    {
        return(this.gehalt.toString());
    }
}
```

Aufbau von Methoden

```
private void setzeGehalt(String gehalt)
```

Methoden-kopf

```
{
  if (Double.parseDouble(gehalt) >= MINGEHALT)
  {
    this.gehalt = new BigDecimal(gehalt);
  }
  else
  {
    this.gehalt = new BigDecimal(MINGEHALT);
  }
}
```

Methoden-rumpf

Methodenkopf

`private`	`void`	`setzeGehalt`	`(`		`)`
`zugriff`	`datentyp`	`methodenname`	`(`		`)`

- Jede Methode beginnt mit dem Zugriffsmodifikator. Methoden sind häufig öffentlich (`public`).
- Dem Zugriffsmodifikator folgt der Datentyp der Methode.
- Dem Datentyp folgt der Methodenname. Der Methodenname ist eindeutig in einer Klasse.
- Dem Methodennamen folgt in runden Klammern die Parameterliste. Die Parameterliste kann leer sein.

Methodenrumpf

- Beginn und Ende mit den geschweiften Klammern.
- Zusammenfassung von Anweisungen für eine bestimmte Aktion.
- Die Aktion wird in dem Methodennamen beschrieben.

Öffentliche Methoden

```
public void aendereGehalt(String gehalt)
```

- Instanzen der Klasse können öffentliche Methoden aufrufen.
- Mit Hilfe von öffentlichen Methode können Instanzen Attribute lesen und verändern.
- Die Methode kann aus allen anderen Klassen aufgerufen werden.

Private Methode

```
private void setzeGehalt(String gehalt)
```

- Die Methode kann nur in der Klasse aufgerufen werden, in der sie definiert ist.
- Berechnungen, die alle Objekte gemeinsam betreffen, werden intern in einer Klasse durchgeführt.

Methoden einer Basisklasse

```
public class clsBasis_Person {
    protected void aendereNachname(String nachname){
         this.nachname = nachname;
    }

    public String lesePersonname(){
         String ausgabe;

         ausgabe = this.anrede + " " + this.vorname;
         ausgabe = ausgabe.trim();
         ausgabe = ausgabe + " " + this.nachname;

         return (ausgabe);
     }
```

Geschützte Methode

```
protected void aendereNachname(String nachname){
```

- Die Methode kann von aus allen Klassen innerhalb desselben Pakets aufgerufen werden.
- Die Methode kann von den Subklassen der Basisklasse aufgerufen werden.
- Instanzen der Klasse können die Methode aufrufen.
- Öffentliche und geschützte Methoden können von der Subklasse überschrieben werden.

… aus der Subklasse aufrufen

```
ausgabe = super.lesePersonname();
```

- Öffentliche und geschützte Methoden der Basisklasse können aus der Subklasse heraus aufgerufen werden.
- Das Schlüsselwort `super` ist ein Platzhalter für die Basisklasse. Das Schlüsselwort verweist auf die Basisklasse der Subklasse.
- Der Punktoperator verbindet den Platzhalter mit der aufzurufenden Methode aus der Basisklasse.

Klassenmethoden

```
public class clsBasis_Person {

    public static String getAnzahlPersonen(){
         return (clsBasis_Person.anzahlPerson);
    }

}
```

Erläuterung

- Das Schlüsselwort `static` kennzeichnet Klassenmethoden.
- Statische Attribute können nur in Konstruktoren oder statischen Methoden genutzt werden.
- Subklassen erben keine Klassenmethoden.

Die Methode ... aus der Basisklasse

```
public class clsBasis_Person {

    protected void aendereNachname(String nachname){
         this.nachname = nachname;
     }

}
```

… wird in der Subklasse überschrieben

```
public class clsBasis_Mitarbeiter {

     @Override
    public void aendereNachname(String nachname){
        this.geburtsname = this.nachname;

        this.nachname = nachname;
    }

}
```

Regeln

- Der Methodenkopf der zu überschreibenden Methode wird nicht durch die Subklasse verändert.
- Der Name der Methode wird in der Subklasse nicht verändert.
- Die Parameterliste in der Subklasse entspricht der Parameterliste in der Basisklasse.
- Der Datentyp der Methode ist in der Basisklasse und in der Subklasse gleich.
- Die Zugriffsmodifikatoren können in der Basisklasse und in der Subklasse unterschiedlich sein.
- Der Methodenrumpf in der Subklasse wurde entsprechend der Klasse erweitert und angepasst.

Annotationen

- Einführung mit Java 5.0.
- Beginn mit dem Add-Zeichen.
- Annotationen werden beim Kompilieren ausgewertet. Die Annotation `@Override` gibt einen Fehler aus, wenn die zu überschreibende Methode nicht in der Basisklasse vorhanden ist.
- Tutorial: https://docs.oracle.com/javase/tutorial/java/annotations/

Aufruf der überschriebenen Methode

- Instanzen der Subklasse rufen die passende Methode in der Subklasse auf. Mit Hilfe des Schlüsselwortes `super.methode(param01, … paramN)` kann die Methode aus der Basisklasse aufgerufen werden.
- Instanzen der Basisklasse rufen die passende Methode in der Basisklasse auf.

Schutz vor Überschreibung

```
public class clsBasis_Mitarbeiter {

    public final String lesePersonname(){
         String ausgabe;

         ausgabe = this.anrede + " " + this.vorname;
         ausgabe = ausgabe.trim();
         ausgabe = ausgabe + " " + this.nachname;
         ausgabe = ausgabe.trim();

         return (ausgabe);
     }
}
```

Erläuterung

- Dem Zugriffsmodifikator folgt das Schlüsselwort `final`.
- Eine Methode, die mit `final` gekennzeichnet ist, kann von der Subklasse nicht überschrieben werden.

Konstrukturen

- Methoden zur Erzeugung von Instanzen.
- Zum Bau eines Objekts werden Anfangswerte übergeben.
- Der Name eines Konstruktoren ist immer gleich der Klasse, von der ein Objekt konstruiert werden soll.
- Automatischer Aufruf durch das Schlüsselwort `new()`.
- Konstruktoren der Basisklasse werden nicht an die Subklasse vererbt.

Konstruktoren in der Subklasse

```
public class clsBasis_Mitarbeiter {


    public clsSub_Mitarbeiter(String anrede, String vorname,
            String nachname, String angestelltAm, String gehalt)
    {
        super(anrede, vorname, nachname);
        this.angestelltAm = angestelltAm;

        setzeGehalt(gehalt) ;

        clsSub_Mitarbeiter.anzahlMitarbeiter++;
    }

}
```

Standardkonstruktor

```
public clsSub_Mitarbeiter()
{
   this("", "", "", "", "0");
}
```

- Der Standardkonstruktor hat eine leere Parameterliste.
- Die Attribute werden mit Hilfe von Standardwerten initialisiert.
- In diesem Beispiel wird mit Hilfe des Schlüsselwortes `this` ein Konstruktor in der Klasse aufgerufen.

Schlüsselwort this

- Das Schlüsselwort `this` ist ein Platzhalter für die Instanz, welches die Methode aufgerufen hat. Attribute für die Instanz werden gesetzt. Methoden in Abhängigkeit der Instanz werden in einer Klasse aufgerufen.
- Dem Schlüsselwort `this` folgen die runden Klammern. Entsprechend der Parameter in den Klammern wird der passende Konstruktor für `this` aufgerufen.

Schlüsselwort this

```
public clsSub_Mitarbeiter()
{
   this("", "", "", "", "0");
}
```

- Dem Schlüsselwort `this` folgen die runden Klammern.
- Entsprechend der Parameter in den Klammern wird der passende Konstruktor für das Schlüsselwort `this` aufgerufen.
- Das Schlüsselwort `this` symbolisiert die zu erzeugende Instanz.

Sonstige Konstruktoren

```
public clsSub_Mitarbeiter(String anrede,
          String vorname, String nachname,
          String angestelltAm, String gehalt)
{
        super(anrede, vorname, nachname);
        this.angestelltAm = angestelltAm;

        setzeGehalt(gehalt) ;

        clsSub_Mitarbeiter.anzahlMitarbeiter++;
}
```

Schlüsselwort super

```
public clsSub_Mitarbeiter(String anrede,
          String vorname, String nachname,
          String angestelltAm, String gehalt)
{
        super(anrede, vorname, nachname);
```

- Das Schlüsselwort `super` symbolisiert die Basisklasse der Subklasse.
- Dem Schlüsselwort `super` folgen die runden Klammern.
- Entsprechend der Parameter in den Klammern wird der passende Konstruktor für das Schlüsselwort `super` aufgerufen.

Reihenfolge

- Zuerst wird der Konstruktor der Basisklasse aufgerufen. Die, in der Basisklasse definierten Attribute werden initialisiert.
- Falls erforderlich, wird der Konstruktor der Subklasse aufgerufen.
- Anschließend werden weitere Attribute der Subklasse gesetzt.

Schlüsselwort this

```
public clsSub_Mitarbeiter(String anrede,
          String vorname, String nachname,
          String angestelltAm, String gehalt)
{
        this.angestelltAm = angestelltAm;

        this.setzeGehalt(gehalt);
```

- Das Schlüsselwort `this` ist ein Platzhalter für die Instanz, welches die Methode oder den Konstruktor aufgerufen hat.
- Attribute für die Instanz werden gesetzt.
- Methoden in Abhängigkeit der Instanz werden innerhalb der Klasse aufgerufen.

Aufruf der Konstruktoren

```
clsBasis_Person person;
clsSub_Mitarbeiter mitarbeiterB;
clsSub_Mitarbeiter mitarbeiterC;

person = new clsBasis_Person("Herr", "A");
mitarbeiterB = new clsSub_Mitarbeiter("Frau", "B",
                                      "21.03.2017");
```

Erläuterung

- Mit Hilfe des Schlüsselwortes `new Klasse()` wird der passende Konstruktor in der Klasse aufgerufen.
- Anhand der Parameterliste wird ein passender Konstruktor gewählt.

Finale Klassen

```
final public class clsSub_Kunde
          extends clsBasis_Person {
    private final Date aktuellZeit;
    private String titel;
    private Calendar aufgenommenAm;

}
```

Erläuterungen

- Klassen, die mit dem Schlüsselwort `final` gekennzeichnet, besitzen keine Subklassen.
- Finale Klassen vererben keine Attribute und Methoden.