
II. Imperative und objektorientierte Programmierung

- 1. Grundelemente der Programmierung
- 2. Objekte, Klassen und Methoden
- 3. Rekursion und dynamische Datenstrukturen
- 4. Erweiterung von Klassen und fortgeschrittene Konzepte

II.4. Erweiterungen von Klassen und fortgeschrittene Konzepte

- 1. Unterklassen und Vererbung
- 2. Abstrakte Klassen und Interfaces
- 3. Modularität und Pakete
- 4. Ausnahmen (Exceptions)
- 5. Generische Datentypen
- 6. Collections

Beziehungen zwischen Klassen

```
public class Stud {
```

```
    int key;  
    int matrikelnr;  
    Gender gend;  
    String vorname, nachname;
```

```
public class Angestellt {
```

```
    String stellung;  
    int key;  
    Gender gend;  
    String vorname, nachname;
```

```
public enum Gender {  
  
    m, f, d  
  
}
```

Beziehungen zwischen Klassen

```
public class Stud {  
  
    int key;  
    int matrikelnr;  
    Gender gend;  
    String vorname, nachname;  
  
    public String toString () {  
        String anrede = "";  
        if (gend == Gender.m)  
            anrede = "Herr ";  
        else if (gend == Gender.f)  
            anrede = "Frau ";  
  
        return anrede + vorname +  
            " " + nachname; }  
  
    ... }
```

```
public class Angestellt {  
  
    String stellung;  
    int key;  
    Gender gend;  
    String vorname, nachname;  
  
    public String toString () {  
        String anrede = "";  
        if (gend == Gender.m)  
            anrede = "Herr ";  
        else if (gend == Gender.f)  
            anrede = "Frau ";  
  
        return anrede + vorname +  
            " " + nachname; }  
  
    ... }
```

Beziehungen zwischen Klassen

```
public class Stud {
```

```
    int key;  
    int matrikelnr;  
    Gender gend;  
    String vorname, nachname;
```

```
public class Angestellt {
```

```
    String stellung;  
    int key;  
    Gender gend;  
    String vorname, nachname;
```

```
public class Person {  
    int key;  
    Gender gend;  
    String vorname, nachname;
```

```
...}
```

Beziehungen zwischen Klassen

```
public class Stud {
```

```
    int key;  
    int matrikelnr;  
    Gender gend;  
    String vorname, nachname;
```

```
public class Angestellt {
```

```
    String stellung;  
    int key;  
    Gender gend;  
    String vorname, nachname;
```

```
public class Person {
```

```
    int key;  
    Gender gend;  
    String vorname, nachname;
```

```
    public String toString () {
```

```
        String anrede = "";
```

```
        if (gend == Gender.m) anrede = "Herr ";
```

```
        else if (gend == Gender.f) anrede = "Frau ";
```

```
        return anrede + vorname + " " + nachname; }
```

```
    ... }
```

Beziehungen zwischen Klassen

```
public class Stud
extends Person {

    int matrikelnr;
    ...}
```

```
public class Angestellt
extends Person {

    String stellung;

    ...}
```

```
public class Person {
    int key;
    Gender gend;
    String vorname, nachname;

    public String toString () {
        String anrede = "";

        if (gend == Gender.m) anrede = "Herr ";
        else if (gend == Gender.f) anrede = "Frau ";

        return anrede + vorname + " " + nachname; }
    ...}
```

Datentypanpassung und Zugriff

```
Stud s = new Stud ();
```

```
Angestellt a = new Angestellt ();
```

```
Person p;
```

implizite Datentypanpassung

```
p = s;
```

Verboten!

```
s = a;
```

```
System.out.println (s.key + ", " + s.matrikelnr);
```

```
System.out.println (p.key + ", " + p.matrikelnr);
```

Verboten!

```
s = p;
```

Verboten!

```
s = (Stud) p;
```

explizite Datentypanpassung

```
if (p instanceof Stud) s = (Stud) p;
```


Objekte in Klassenhierarchien

```
Person p = new Person ();
```

```
Stud s = new Stud ();
```

```
p = s;
```

```
System.out.println (s.key +  
    ", " + s.matrikelnr);
```

```
System.out.println (p.key +  
    ", " + p.matrikelnr);
```

```
s = (Stud) p;
```

p

s

Objekt Person

Methode
toString

Attribute
int key
String vorname
String nachname
Gender gend

Objekt Stud

Attribut
int matrikelnr

Objekt Person

Methode
toString

Attribute
int key
String vorname
String nachname
Gender gend

Objekte in Klassenhierarchien

```
Person p = new Person ();
```

```
Stud s = new Stud ();
```

```
p = s;
```

```
System.out.println (s.key +  
    ", " + s.matrikelnr);
```

```
System.out.println (p.key +  
    ", " + p.matrikelnr);
```

```
s = (Stud) p;
```

P

S

Objekt Person

Methode
toString

Attribute
int key
String vorname
String nachname
Gender gend

Objekt Stud

Attribut
int matrikelnr

Objekt Person

Methode
toString

Attribute
int key
String vorname
String nachname
Gender gend

Konstrukturen in Klassenhierarchien

```
public class Person {

    int key;    Gender gend;
    String vorname, nachname;

    public Person () {

        key = SimpleIO.getInt("Key
                               der Person");
        vorname = SimpleIO.getString("Vor-
                                       name der Person");
        nachname = SimpleIO.getString("Nach-
                                       name der Person");
        gend = ... ;

    }

    ...
}
```

```
public class Stud
extends Person {

    int matrikelnr;

    public Stud () {

        super ();
        matrikelnr = SimpleIO.getInt(
                               "Matrikelnr");
    }

    ...
}
```

Konstrukturen in Klassenhierarchien

```
public class Person {  
  
    ...  
  
    public Person (int key) {  
        this.key = key;  
    }  
  
    public Person (int key,  
                   String vorname, String  
                   nachname, Gender gend) {  
  
        this.key = key;  
        this.vorname = vorname;  
        this.nachname = nachname;  
        this.gend = ... ;  
    }  
    ...  
}
```

```
public class Stud  
extends Person {  
  
    ...  
  
    public Stud (int key,  
                 String vorname, String  
                 nachname, Gender gend,  
                 int matrikelnr) {  
  
        super (key, vorname,  
                nachname, gend);  
  
        this.matrikelnr =  
                                matrikelnr;  
    }  
    ...  
}
```

Stattdessen möglich:
this (key);

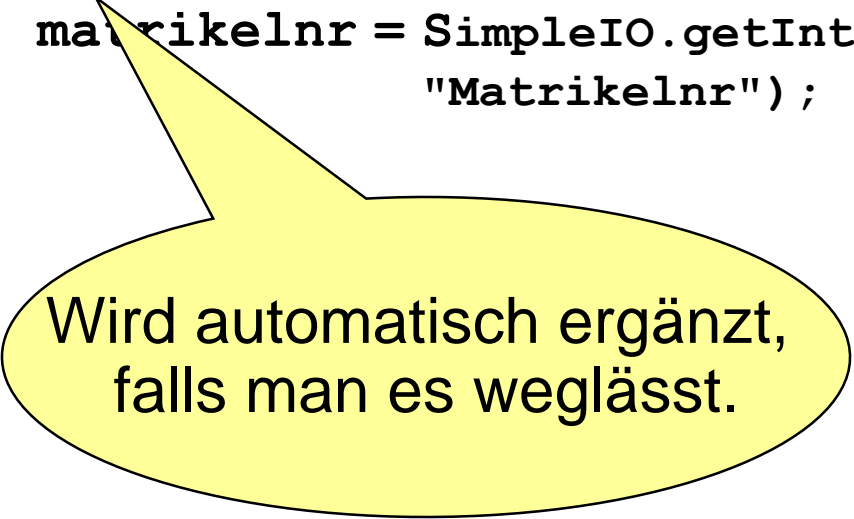
Konstrukturen in Klassenhierarchien

```
public class Person {  
  
    ...  
  
    public Person (String vorname, String nachname) {  
        this (0, vorname, nachname, Gender.m);  
    }  
  
    public Person (int key,  
        String vorname, String  
        nachname, Gender gend) {  
  
        this.key = key;  
        this.vorname = vorname;  
        this.nachname = nachname;  
        this.gend = ... ;  
    }  
    ...  
}
```

Konstrukturen in Klassenhierarchien

```
public class Person {  
  
    int key;    Gender gend;  
    String vorname, nachname;  
  
    public Person () {  
  
        key = SimpleIO.getInt("Key  
            der Person");  
        vorname = SimpleIO.getString("Vor-  
            name der Person");  
        nachname = SimpleIO.getString("Nach-  
            name der Person");  
        this.gend = ... ;  
    }  
  
    ...  
}
```

```
public class Stud  
extends Person {  
  
    int matrikelnr;  
  
    public Stud () {  
  
        super ();  
        matrikelnr = SimpleIO.getInt(  
            "Matrikelnr");  
    }  
  
    ... }  
}
```



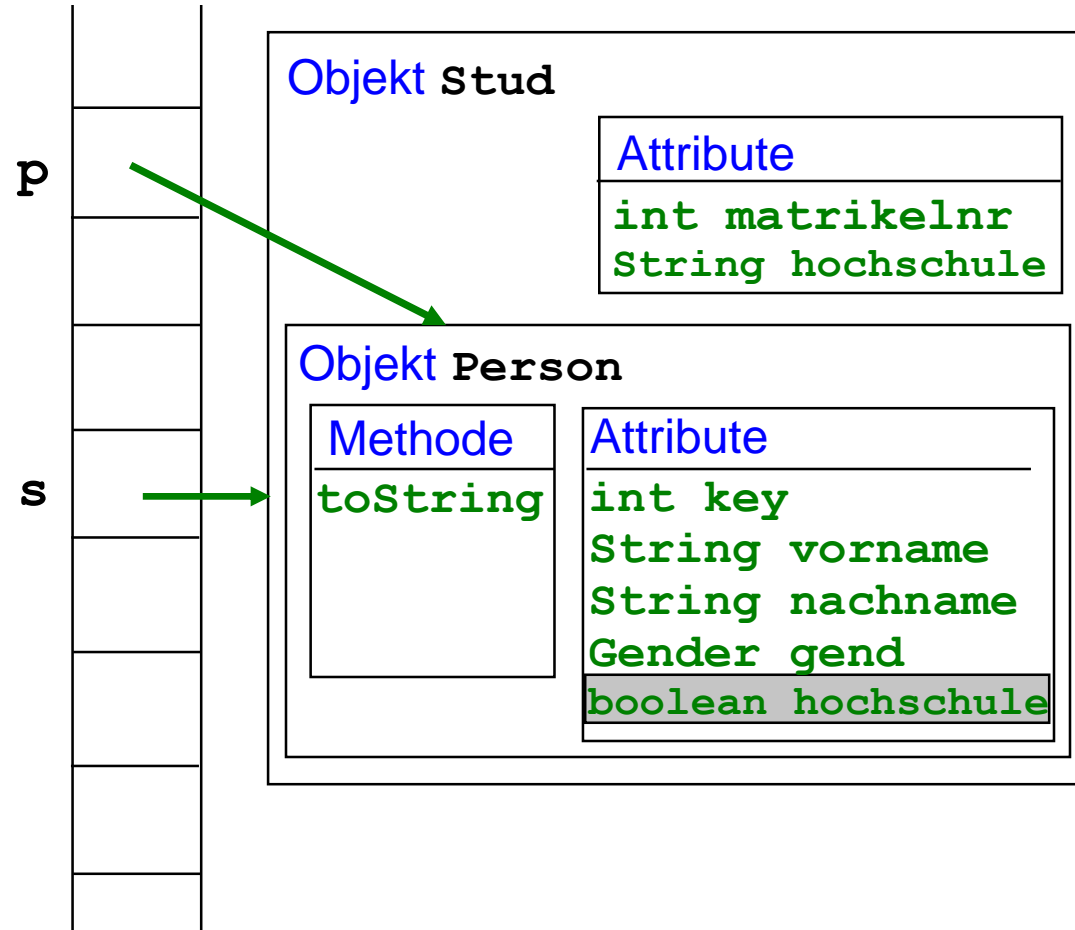
Wird automatisch ergänzt,
falls man es weglässt.

Verdecken von Attributen

```
public class Person {  
  
    int key;    Gender gend;  
    String vorname, nachname;  
    boolean hochschule;  
    ...  
}
```

```
public class Stud  
extends Person {  
  
    int matrikelnr;  
    String hochschule;  
    ...  
}
```

```
Stud s = new Stud ();  
Person p = s;  
p.hochschule = true;  
s.hochschule = "RWTH";
```



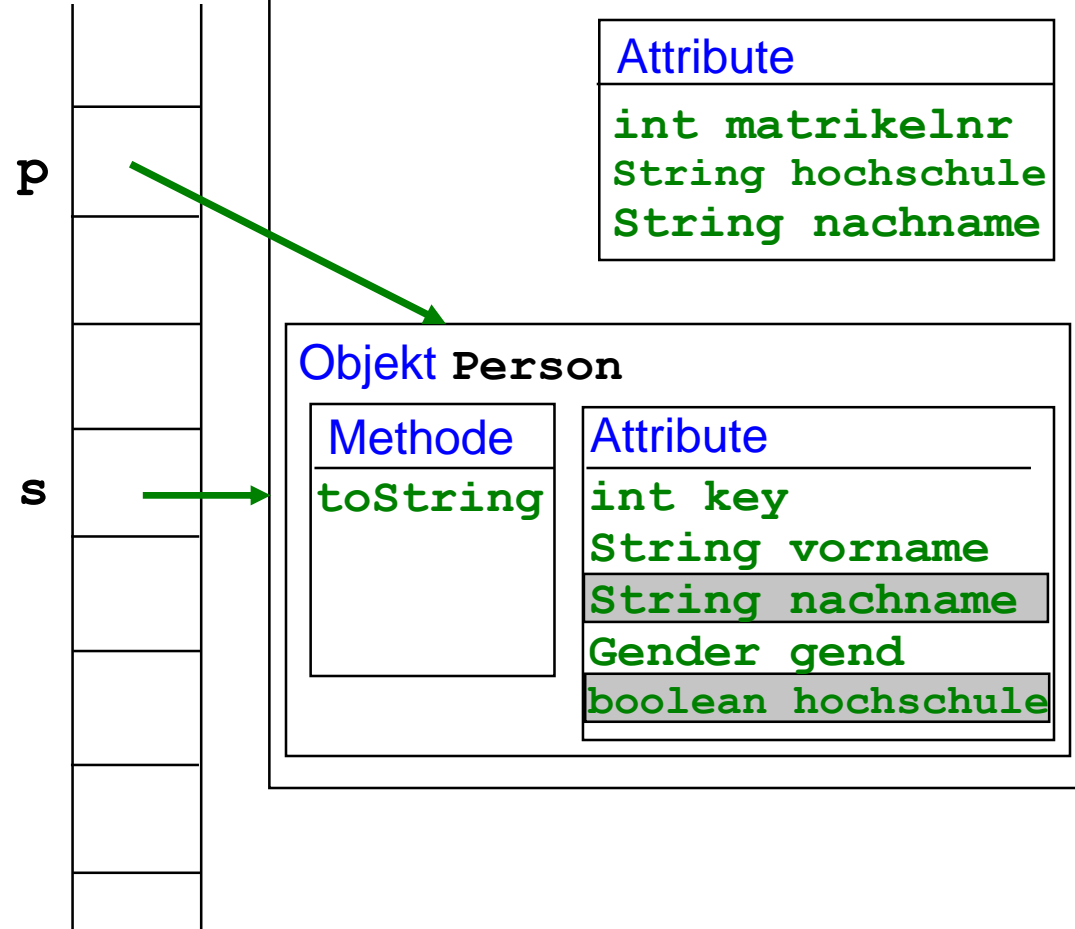
Verdecken von Attributen

```
public class Stud
extends Person {

    int matrikelnr;
    String hochschule;
    String nachname;

    public Stud () {
        super ();
        matrikelnr = SimpleIO.getInt(
            "Matrikelnr");
    }
    ...
}
```

```
Stud s = new Stud ();
Person p = s;
```



```
p.nachname == "Meier"
```

```
s.nachname == null
```


Überschreiben von Methoden

```
public class Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahngebuehr:" + geb, "Mahnung");  
    }  
}
```

```
public class Stud extends Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahngebuehr:" + geb, "Mahnung");  
  
        SimpleIO.output("Mitteilung an " +  
            "Studierendensekretariat:" +  
            this + " noch nicht " +  
            "exmatrikulieren", "Mahnung");  
    }  
}
```

Objekt Stud

Methode

mahnung

Attribute

int matrikelnr
String hochschule

Objekt Person

Methoden

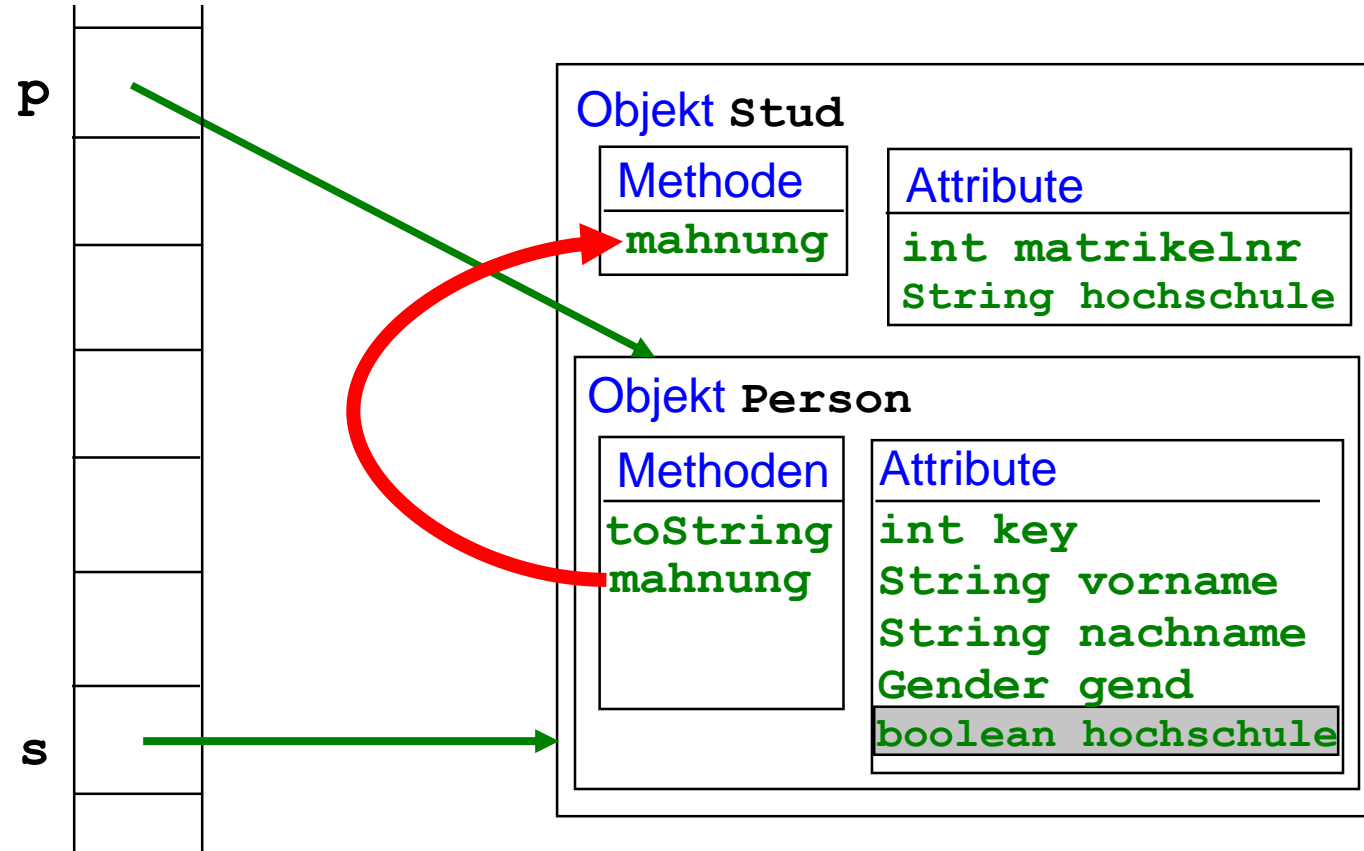
toString
mahnung

Attribute

int key
String vorname
String nachname
Gender gend
boolean hochschule

Überschreiben von Methoden

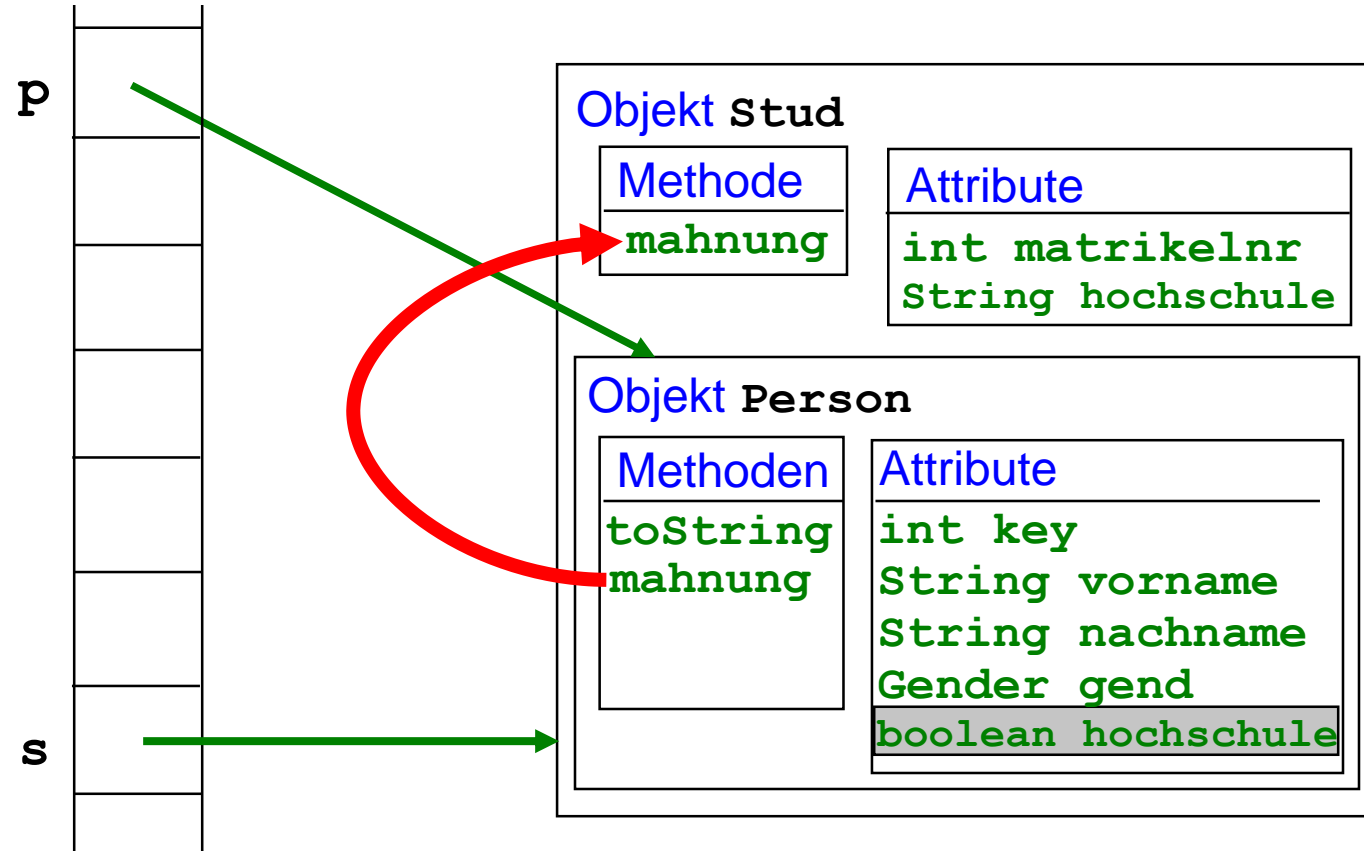
```
Stud s =  
    new Stud ();  
  
Person p = s;  
  
s.mahnung (10);  
p.mahnung (20);
```



```
Mitteilung an Frau Anna Meier  
Mahngebuehr: 10  
Mitteilung an Studierendensekretariat  
Frau Meier noch nicht exmatrikulieren
```

Überschreiben von Methoden

```
Stud s =  
    new Stud ();  
  
Person p = s;  
  
s.mahnung (10);  
p.mahnung (20);
```



```
Mitteilung an Frau Anna Meier  
Mahngebuehr: 20  
Mitteilung an Studierendensekretariat  
Frau Meier noch nicht exmatrikulieren
```

Verwendung überschriebener Methoden

```
public class Person {  
  
    void mahnung (int geb) {...}  
  
    static void sendeMahnungen (Person [] ausleiher, int geb) {  
  
        for (Person p : ausleiher) {  
            p.mahnung (geb);  
        }  
    }  
}
```

```
public class Stud extends Person {  
  
    void mahnung (int geb) {...}  
}
```

```
public class Angestellt extends Person {  
  
    void mahnung (int geb) {...}  
}
```

Finale Methoden

```
public class Person {  
  
    final void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahnggebuehr:" + geb,"Mahnung");  
    }  
}
```

```
public class Stud extends Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahnggebuehr:" + geb,"Mahnung");  
  
        SimpleIO.output("Mitteilung an " +  
            "Studierendensekretariat:" +  
            this + " noch nicht " +  
            "exmatrikulieren", "Mahnung");  
    }  
}
```

Objekt Stud

Methode

mahnung

Attribute

int matrikelnr
String hochschule

Objekt Person

Methoden

toString
mahnung

Attribute

int key
String vorname
String nachname
Gender gend
boolean hochschule

Finale Methoden

```
public class Person {  
  
    final void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahnguebuehr:" + geb, "Mahnung");  
  
    }  
  
}
```

```
public class Stud extends Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahnguebuehr:" + geb, "Mahnung");  
  
        SimpleIO.output("Mitteilung an " +  
            "Studierendensekretariat:" +  
            this + " noch nicht " +  
            "exmatrikulieren", "Mahnung");  
  
    }  
  
}
```

Objekt Stud

Attribute

```
int matrikelnr  
String hochschule
```

Objekt Person

Methoden

```
toString  
mahnung
```

Attribute

```
int key  
String vorname  
String nachname  
Gender gend  
boolean hochschule
```

Zugriff auf überschriebene Methoden

```
public class Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahngebuehr:" + geb,"Mahnung");  
    }  
}
```

```
public class Stud extends Person {  
  
    void mahnung (int geb) {  
  
        super.mahnung (geb);  
  
        SimpleIO.output("Mitteilung an " +  
            "Studierendensekretariat:" +  
            this + " noch nicht " +  
            "exmatrikulieren","Mahnung");  
    }  
}
```

Objekt Stud

Methode

```
void mahnung (int geb) {  
    super.mahnung (geb); ...  
}
```

Objekt Person

Methoden

```
toString  
mahnung
```

Attribute

```
int key  
String vorname  
String nachname  
Gender gend  
boolean hochschule
```

Zugriff auf verdeckte Attribute

```
public class Person {  
  
    int key;    Gender gend;  
    String vorname, nachname;  
    boolean hochschule;  
  
}
```

```
public class Stud extends Person {  
  
    int matrikelnr;  
    String hochschule;  
  
    boolean hatHochschulabschluss () {  
  
        return super.hochschule;  
  
    }  
  
}
```

Objekt Stud

Methode

```
boolean hatHochschulabschluss {  
    return super.hochschule ...}
```

Objekt Person

Methoden

```
toString  
mahnung
```

Attribute

```
int key  
String vorname  
String nachname  
Gender gend  
boolean hochschule
```


Überladen von Methoden

```
public class Person {  
    void mahnung (int geb) { ... }  
  
    int mahnung (int g, int h) {  
        return g + h;  
    }  
}
```

```
public class Stud extends Person {  
    void mahnung (int geb) { ... }  
}
```

```
Stud s = new Stud ();  
Person p = s;  
gebuehr = p.mahnung(10, 5);  
  
s.mahnung (gebuehr);  
p.mahnung (gebuehr);
```

Objekt Stud

Methode

void mahnung (int geb)

Objekt Person

Methoden

String toString ()

void mahnung (int geb)

int mahnung (int g, int h)

Zugriffsspezifikationen

Einschränkung des Zugriffs auf Attribute und Methoden:

- **private:**

Komponente nur innerhalb der Klasse bekannt

- **kein Schlüsselwort:**

Komponente nur innerhalb des Pakets bekannt

- **public:**

Komponente überall bekannt

Zugriffsspezifikationen

Einschränkung des Zugriffs auf Attribute und Methoden:

- **private:**

Komponente nur innerhalb der Klasse bekannt

- **kein Schlüsselwort:**

Komponente nur innerhalb des Pakets bekannt

- **protected:**

Komponente innerhalb des Pakets und in allen Unterklassen bekannt

- **public:**

Komponente überall bekannt

Zugriffsspezifikation

```
public class Person {  
  
    protected int key;  
    protected Gender gend;  
    protected String vorname, nachname;  
    protected boolean hochschule;  
  
    public void mahnung (int geb) {...}  
  
    ...  
}
```

```
public class Stud extends Person {  
  
    protected int matrikelnr;  
    protected String hochschule;  
  
    public void mahnung (int geb) {...}  
  
    ...  
}
```