

---

# 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 Student {  
  
    int key;  
    int matrikelnr;  
    boolean male;  
    String vorname, nachname;  
  
    public String toString () {  
        String anrede;  
        if (male) anrede = "Herr ";  
        else      anrede = "Frau ";  
  
        return anrede + vorname +  
            " " + nachname; }  
  
    ... }  
}
```

```
public class Angestellter {  
  
    String stellung;  
    int key;  
    boolean male;  
    String vorname, nachname;  
  
    public String toString () {  
        String anrede;  
        if (male) anrede = "Herr ";  
        else      anrede = "Frau ";  
  
        return anrede + vorname +  
            " " + nachname; }  
  
    ... }  
}
```

# Beziehungen zwischen Klassen

```
public class Student {
```

```
    int key;
```

```
    int matrikelnr;
```

```
    boolean male;
```

```
    String vorname, nachname;
```

```
public class Angestellter {
```

```
    String stellung;
```

```
    int key;
```

```
    boolean male;
```

```
    String vorname, nachname;
```

```
public class Person {
```

```
    int key;
```

```
    boolean male;
```

```
    String vorname, nachname;
```

```
public
```

```
    String
```

```
    if
```

```
    else
```

```
    return
```

```
...}
```

```
...}
```

```
    () {
```

```
        "Herr ";
```

```
        "Frau ";
```

```
        me +
```

```
    }; }
```

# Beziehungen zwischen Klassen

```
public class Student  
extends Person {
```

```
    int matrikelnr;
```

```
public class Angestellter  
extends Person {
```

```
    String stellung;
```

```
public class Person {  
    int key;  
    boolean male;  
    String vorname, nachname;  
    public String toString () {  
        String anrede;  
        if (male) anrede = "Herr ";  
        else      anrede = "Frau ";  
        return anrede + vorname + " " + nachname; }  
    ... }  
}
```

# Beziehungen zwischen Klassen

```
public class Student
extends Person {

    int matrikelnr;
    ...}
```

```
public class Angestellter
extends Person {

    String stellung;

    ...}
```

```
public class Person {
    int key;
    boolean male;
    String vorname, nachname;

    public String toString () {
        String anrede;
        if (male) anrede = "Herr ";
        else      anrede = "Frau ";

        return anrede + vorname + " " + nachname; }

    ...}
```

# Datentypanpassung und Zugriff

```
Student s = new Student ();
```

```
Angestellter a = new Angestellter ();
```

```
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!

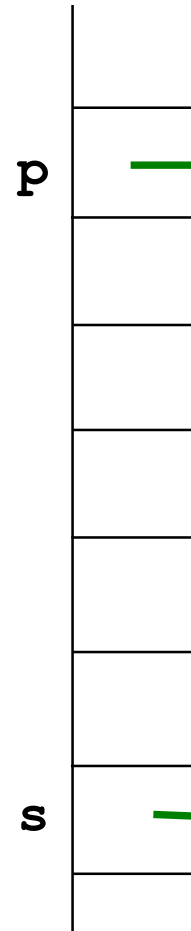
```
s = (Student) p;
```

explizite Datentypanpassung

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

# Objekte in Klassenhierarchien

```
Person p = new Person ();  
  
Student s = new Student ();  
  
p = s;  
  
System.out.println (s.key +  
    ", " + s.matrikelnr);  
  
System.out.println (p.key +  
    ", " + p.matrikelnr);  
  
s = (Student) p;
```



## Objekt Person

Methode
toString

## Attribute

int key
String vorname
String nachname
boolean male

## Objekt Student

### Attribut

int matrikelnr
----------------

## Objekt Person

Methode
toString

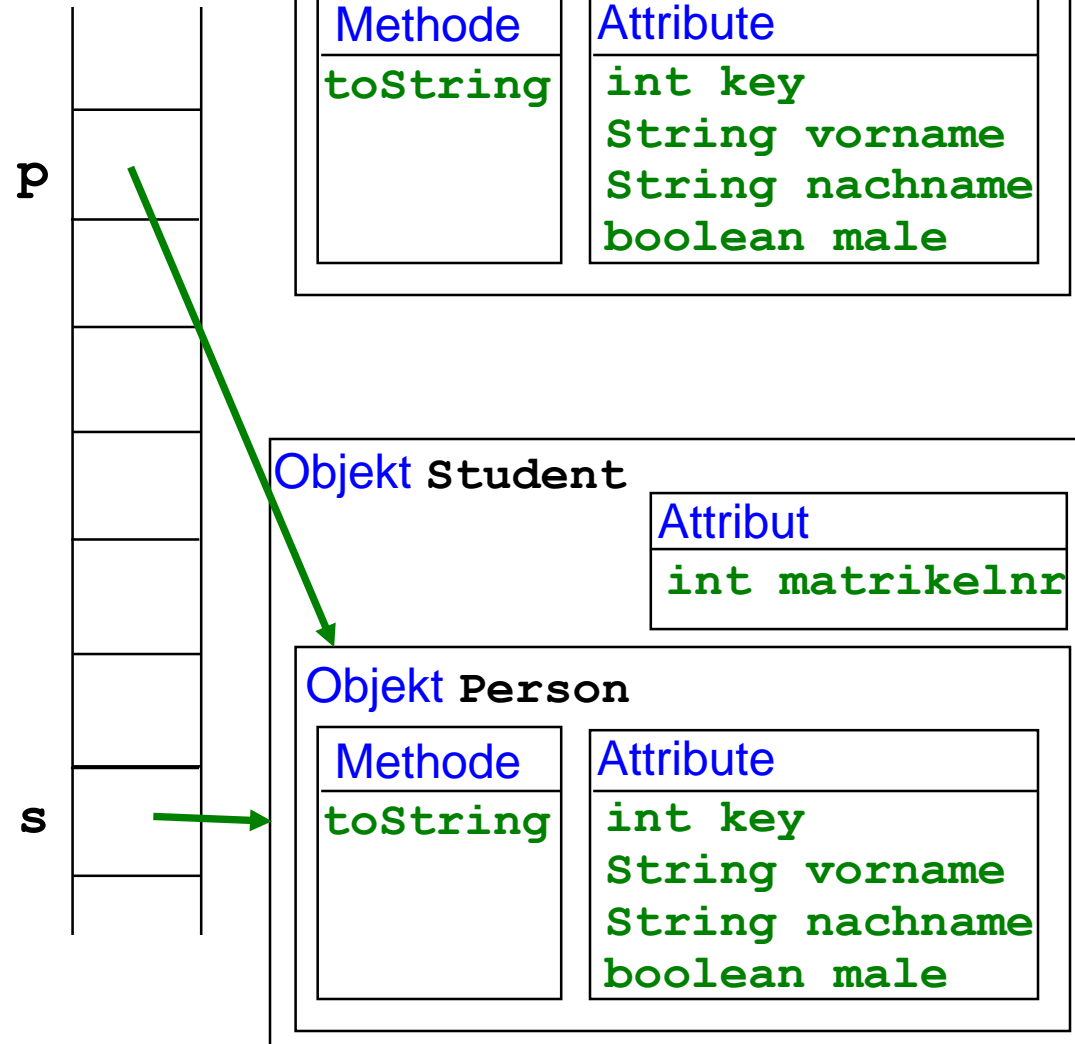
## Attribute

int key
String vorname
String nachname
boolean male



# Objekte in Klassenhierarchien

```
Person p = new Person ();  
  
Student s = new Student ();  
  
p = s;  
  
System.out.println (s.key +  
    ", " + s.matrikelnr);  
  
System.out.println (p.key +  
    ", " + p.matrikelnr);  
  
s = (Student) p;
```



# Konstrukturen in Klassenhierarchien

```
public class Person {  
  
    int key;    boolean male;  
    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");  
        male = SimpleIO.getBoolean("Person  
            männlich?", "ja", "nein");  
    }  
  
    ...  
}
```

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

# Konstruktor in Klassenhierarchien

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

```
public class Student  
extends Person {  
  
    ...  
  
    public Student (int key,  
                   String vorname, String  
                   nachname, boolean male,  
                   int matrikelnr) {  
  
        super (key, vorname,  
              nachname, male);  
  
        this.matrikelnr =  
                               matrikelnr;  
    }  
  
    ...  
}
```

Stattdessen möglich:  
**this (key);**

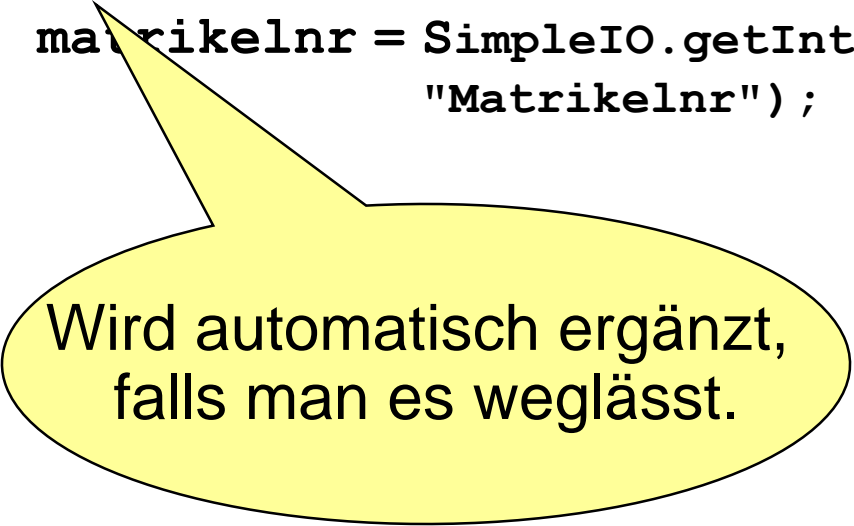
# Konstruktor in Klassenhierarchien

```
public class Person {  
  
    ...  
  
    public Person (String vorname, String nachname) {  
        this (0, vorname, nachname, true);  
    }  
  
    public Person (int key,  
                  String vorname, String  
                  nachname, boolean male) {  
  
        this.key = key;  
        this.vorname = vorname;  
        this.nachname = nachname;  
        this.male = male;  
    }  
  
    ...  
}
```

# Konstrukturen in Klassenhierarchien

```
public class Person {  
  
    int key;    boolean male;  
    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");  
        male = SimpleIO.getBoolean("Person  
            männlich?", "ja", "nein");  
        hochschule = SimpleIO.getBoolean("Hat  
            Person Hochschulabschluss?",  
            "ja", "nein");  
    } ...
```

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



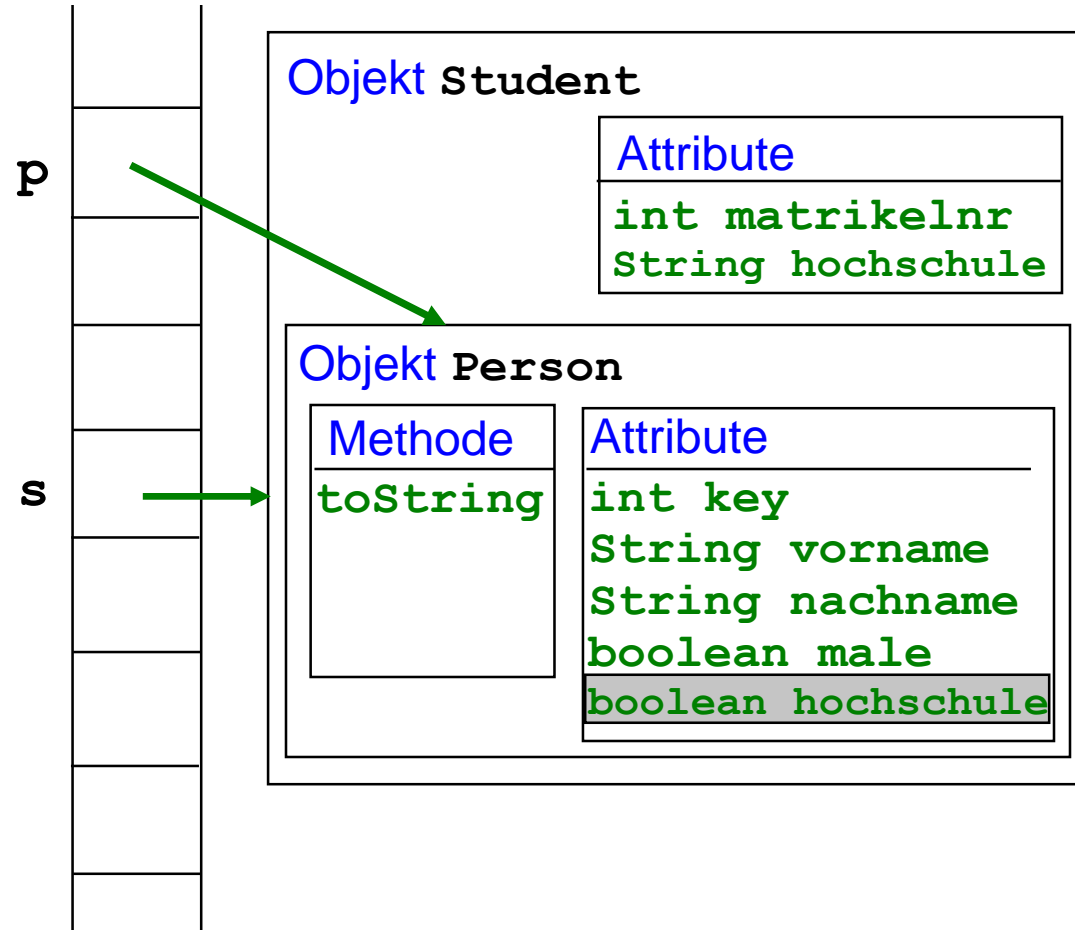
Wird automatisch ergänzt,  
falls man es weglässt.

# Verdecken von Attributen

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

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

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



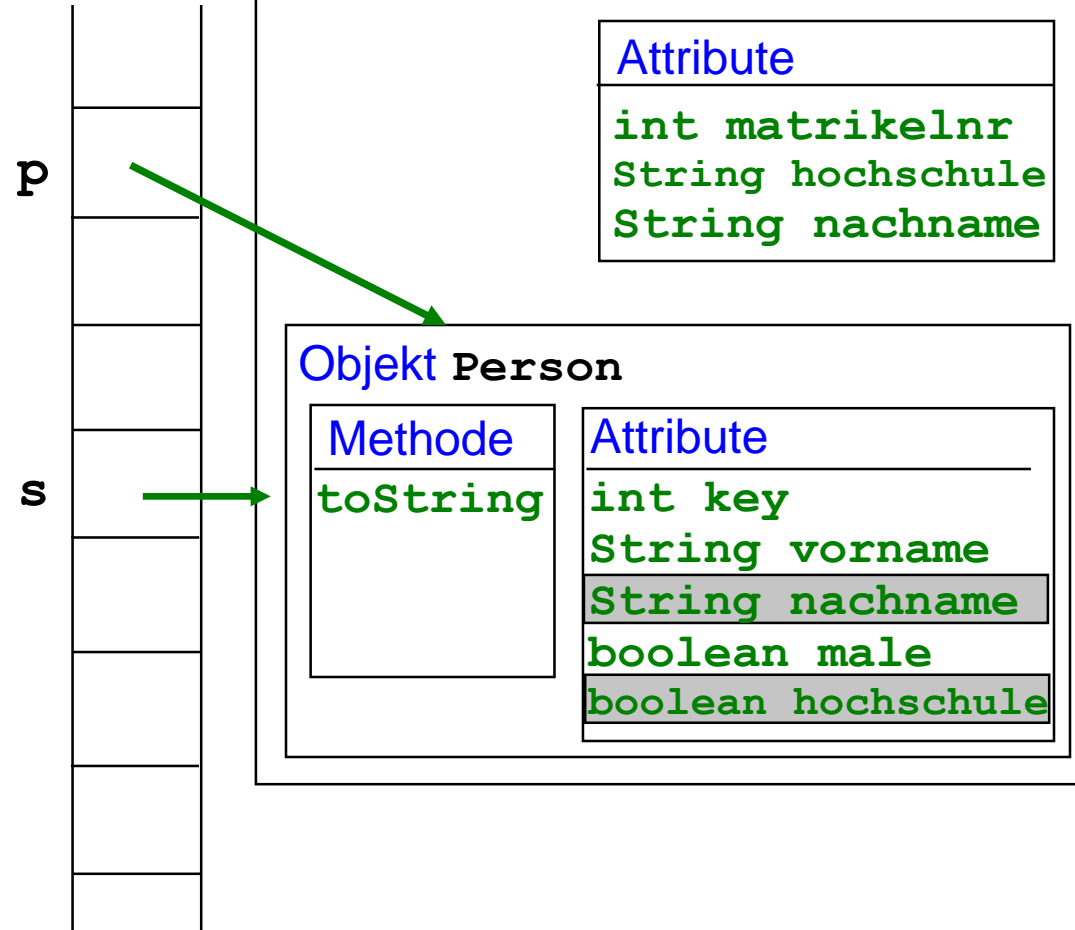
# Verdecken von Attributen

```
public class Student
extends Person {

    int matrikelnr;
    String hochschule;
    String nachname;

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

```
Student s = new Student ();
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 Student extends Person {  
  
    void mahnung (int geb) {  
  
        SimpleIO.output("Mitteilung an " + this +  
            "Mahngebuehr:" + geb,"Mahnung");  
  
        SimpleIO.output("Mitteilung an " +  
            Studentensekretariat:" +  
            this + " noch nicht " +  
            "exmatrikulieren", "Mahnung");  
  
    }  
  
}
```

## Objekt Student

### Methode

mahnung

### Attribute

int matrikelnr  
String hochschule

## Objekt Person

### Methoden

toString  
mahnung

### Attribute

int key  
String vorname  
String nachname  
boolean male  
boolean hochschule



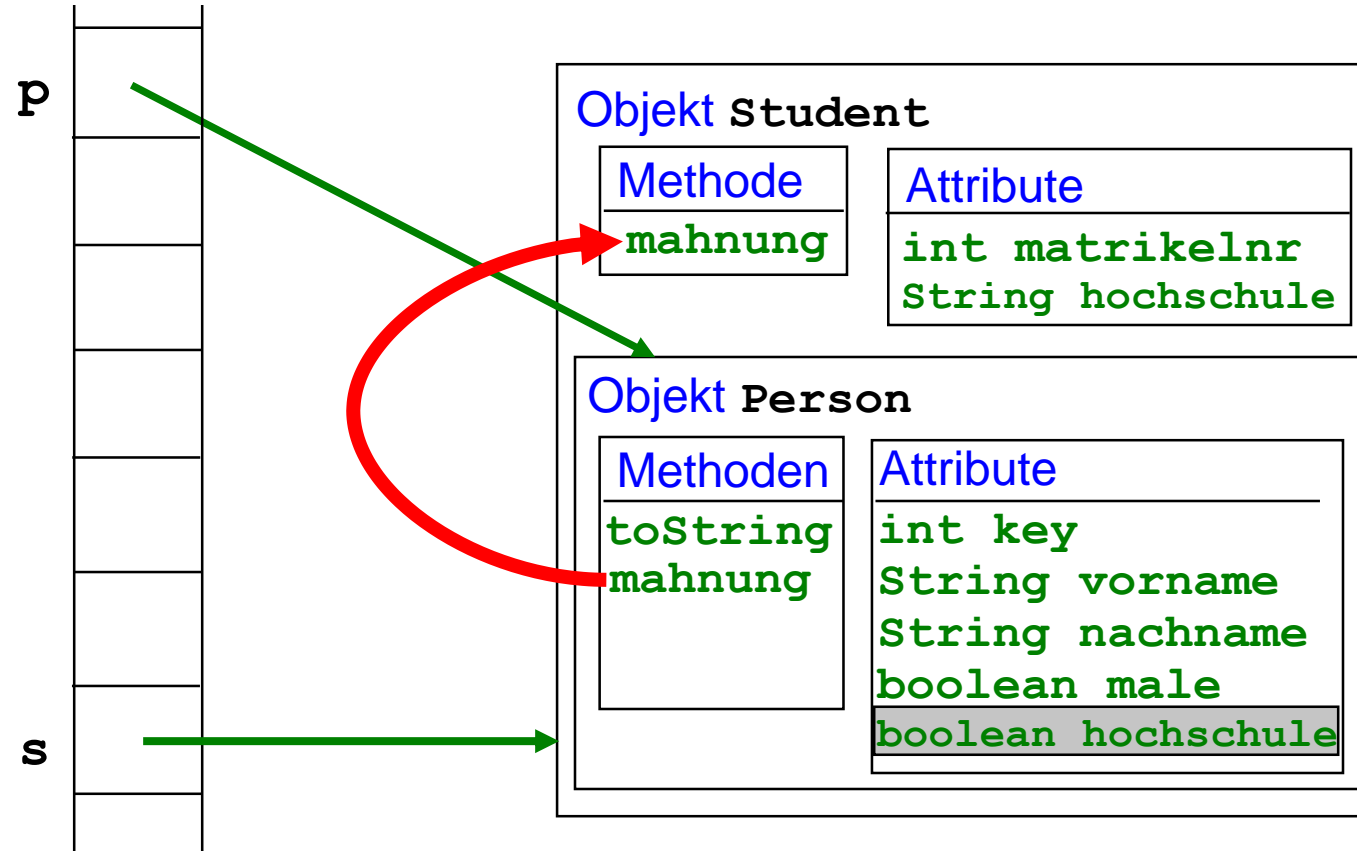
# Überschreiben von Methoden

```
Student s =  
  new Student ();
```

```
Person p = s;
```

```
s.mahnung (10);
```

```
p.mahnung (20);
```



```
Mitteilung an Frau Anna Meier
```

```
Mahngebuehr: 10
```

```
Mitteilung an Studentensekretariat
```

```
Frau Meier noch nicht exmatrikulieren
```

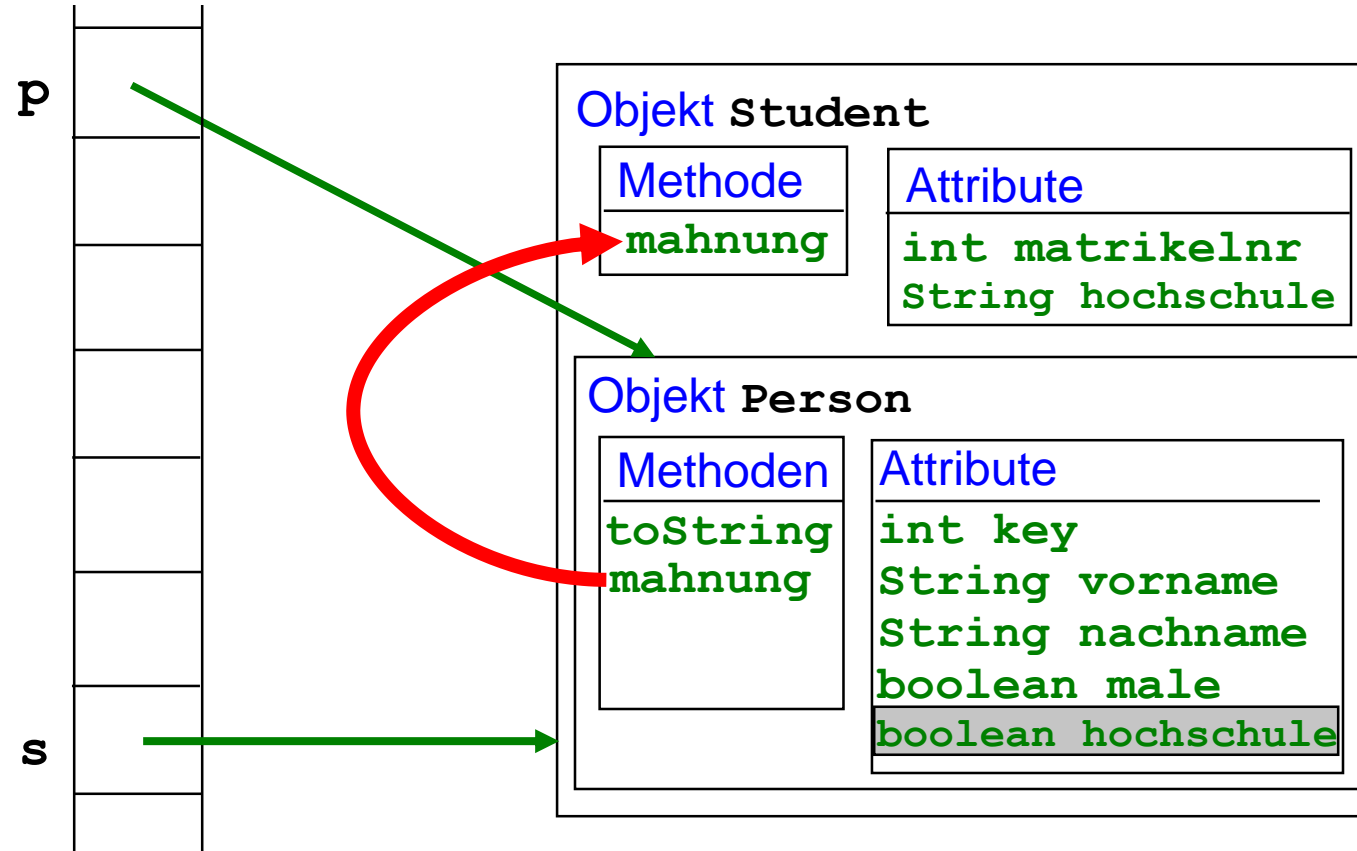
# Überschreiben von Methoden

```
Student s =  
  new Student ();
```

```
Person p = s;
```

```
s.mahnung (10);
```

```
p.mahnung (20);
```



```
Mitteilung an Frau Anna Meier
```

```
Mahngebuehr: 20
```

```
Mitteilung an Studentensekretariat
```

```
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 Student extends Person {  
  
    void mahnung (int geb) {...}
```

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

## Objekt Student

### Methode

mahnung

### Attribute

int matrikelnr  
String hochschule

## Objekt Person

### Methoden

toString  
mahnung

### Attribute

int key  
String vorname  
String nachname  
boolean male  
boolean hochschule

# Finale Methoden

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

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

## Objekt Student

### Attribute

```
int matrikelnr  
String hochschule
```

## Objekt Person

### Methoden

```
toString  
mahnung
```

### Attribute

```
int key  
String vorname  
String nachname  
boolean male  
boolean hochschule
```

# Zugriff auf überschriebene Methoden

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

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

## Objekt Student

### Methode

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

## Objekt Person

### Methoden

```
toString  
mahnung
```

### Attribute

```
int key  
String vorname  
String nachname  
boolean male  
boolean hochschule
```

# Zugriff auf verdeckte Attribute

```
public class Person {  
  
    int key;    boolean male;  
    String vorname, nachname;  
    boolean hochschule;
```

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

## Objekt Student

### Methode

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

## Objekt Person

### Methoden

```
toString  
mahnung
```

### Attribute

```
int key  
String vorname  
String nachname  
boolean male  
boolean hochschule
```

# Überladen von Methoden

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

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

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

Objekt Student

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 boolean male;  
    protected String vorname, nachname;  
    protected boolean hochschule;  
  
    public void mahnung (int geb) {...}  
  
    ...  
}
```

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