Java:
Vererbung

Teil 3: `super()`

Konstruktor und Vererbung

Kindklasse ruft SELBSTSTÄNDIG und IMMER zuerst den Konstruktor der Elternklasse auf!

Konstruktor und Vererbung

Kindklasse ruft SELBSTSTÄNDIG und IMMER zuerst den Konstruktor der Elternklasse auf!

Zur Erinnerung:
-Alle Klassen BRAUCHEN einen Konstruktor.
-Wenn wir keinen manuell schreiben, fügt der Compiler einen ein.

```
public class B extends A
{

}
```

```
public class B extends A
{
    public B()
    { }
}
```

X = vom Compiler eingefügt

Konstruktor/Vererbung: Beispiel

Startklasse:

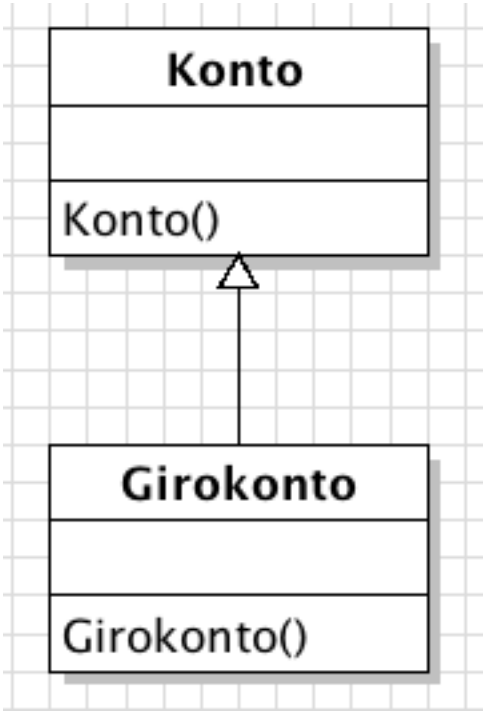
```
public static void main(String[] args){
 System.out.println("Erzeuge jetzt Konto:");
 Konto k = new Konto();
 System.out.println("Erzeuge jetzt Girokonto");
 Girokonto g = new Girokonto(); }
```

Konto:

```
public class Konto{
// Konstruktor
public Konto(){
System.out.println("KONSTRUKTOR von Konto"); }
```

Girokonto:

```
public class Girokonto extends Konto {
// Konstruktor
public Girokonto()  {
 System.out.println("KONSTRUKTOR GIROKONTO");}
```



Konstruktor/Vererbung: Beispiel

Startklasse:

```
public stat
 System.out
 Konto k =
 System.out
 Girokonto
```

```
run:
Erzeuge jetzt Konto:
KONSTRUKTOR von Konto
Erzeuge jetzt Girokonto
KONSTRUKTOR von Konto
KONSTRUKTOR GIROKONTO
ERSTELLEN ERFOLGREICH (Gesamtzeit: 0 Min
```

Konto:

```
public class Konto{
// Konstruktor
public Konto(){
System.out.println("KONSTRUKTOR von Konto"); }
```

Girokonto:

```
public class Girokonto extends Konto {
// Konstruktor
public Girokonto()  {
 System.out.println("KONSTRUKTOR GIROKONTO");}
```

Aufgabe 1: Konstruktor/Vererbung

Aufgabe:

Programmieren Sie zwei Klassen (Eltern- und Kindklasse) + Startklasse.

Lassen Sie in beiden Konstruktoren eine eindeutige Meldung ausgeben ("Ich bin der Konstruktor der Klasse x").

Instanziieren Sie jeweils ein Objekt und beachten Sie die Ausgaben.

`super()`

Mit dem Aufruf `super()` wird der Standard-Konstruktor der Elternklasse aufgerufen.

Wenn wir super() nicht manuell schreiben, macht der Compiler das für uns als ERSTE ANWEISUNG im Konstruktor der Kindklasse.

```
public class B extends A{

}
```

```
public class B extends A{
    public B()
    { }
}
```

```
public class B extends A{
    public B(){
        super();
    }
}
```

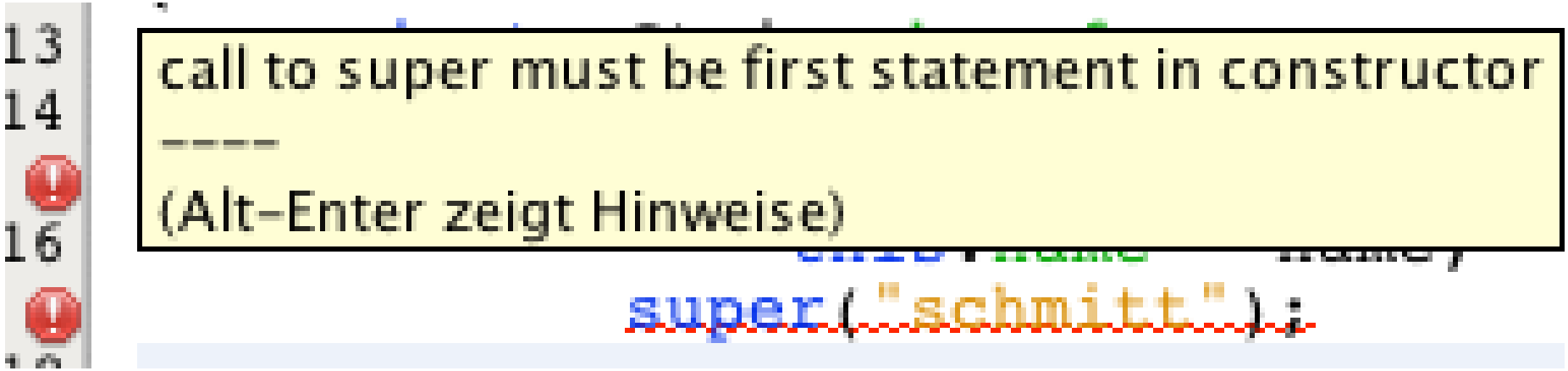
Gleiche Funktion bei allen drei Varianten!

X = vom Compiler eingefügt

`super()` ist die

ERSTE ANWEISUNG

im Konstruktor der Kindklasse.



`SUPER()`

SCHREIBEN WIR NUR DANN, WENN ES SEIN MUSS.

Aufgabe 2: super() verstehen

Aufgabe:

Programmieren Sie zwei Klassen ("Fahrzeug", Elternklasse und "Auto", Kindklasse) + Startklasse.

Spielen Sie mit dem super()-Aufruf, z.B:

- lassen Sie den Konstruktor der Kindklasse weg
- benutzen Sie den Konstruktor der Kindklasse, einmal mit und einmal ohne super()

Instanziieren Sie zwischendurch Objekte und beachten Sie die Ausgaben.

`super()` und Konstruktor mit Parameter(n)

Situation:

Konstruktor der Elternklasse hat Parameter

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein){
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

`super()` und Konstruktor mit Parameter(n)

Situation:

Konstruktor der Elternklasse hat Parameter

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

Zur Erinnerung:
In der Regel werden die dem Konstruktor übergebenen Parameter entsprechenden Attributen zugewiesen (wie Setter)!

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein){
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

Aufgabe 3a: Welche Attribute hat

- der Vater?
- der Sohn?

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein){
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

Aufgabe 3b: Was passiert, wenn ein Objekt der Klasse Sohn erzeugt wird?

```
public static void main(String[] args)
{
   Sohn s = new Sohn(true);
}
```

```
 8  constructor Vater in class vererbung11.Vater cannot be applied to given types;
 9    required: java.lang.String
10    found: no arguments
11    reason: actual and formal argument lists differ in length
12  ----
13  (Alt-Enter zeigt Hinweise)
14      public Sohn(boolean mopedfuehrerschein)
15      {
16          this.mopedfuehr
public class Vater {
    protected String nachname
    public Vater(String nachn
        this.nachname = nachn
} ...
```

das heißt:
Parameter für Konstruktor "Vater"
fehlen beim Aufruf des Konstruktors
"Sohn"

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein){
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

Lösung: `super()` parametrisieren

Dem Aufruf super() werden die Parameter für den Konstruktor der Elternklasse übergeben.

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein){
        super("Schmitt");
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

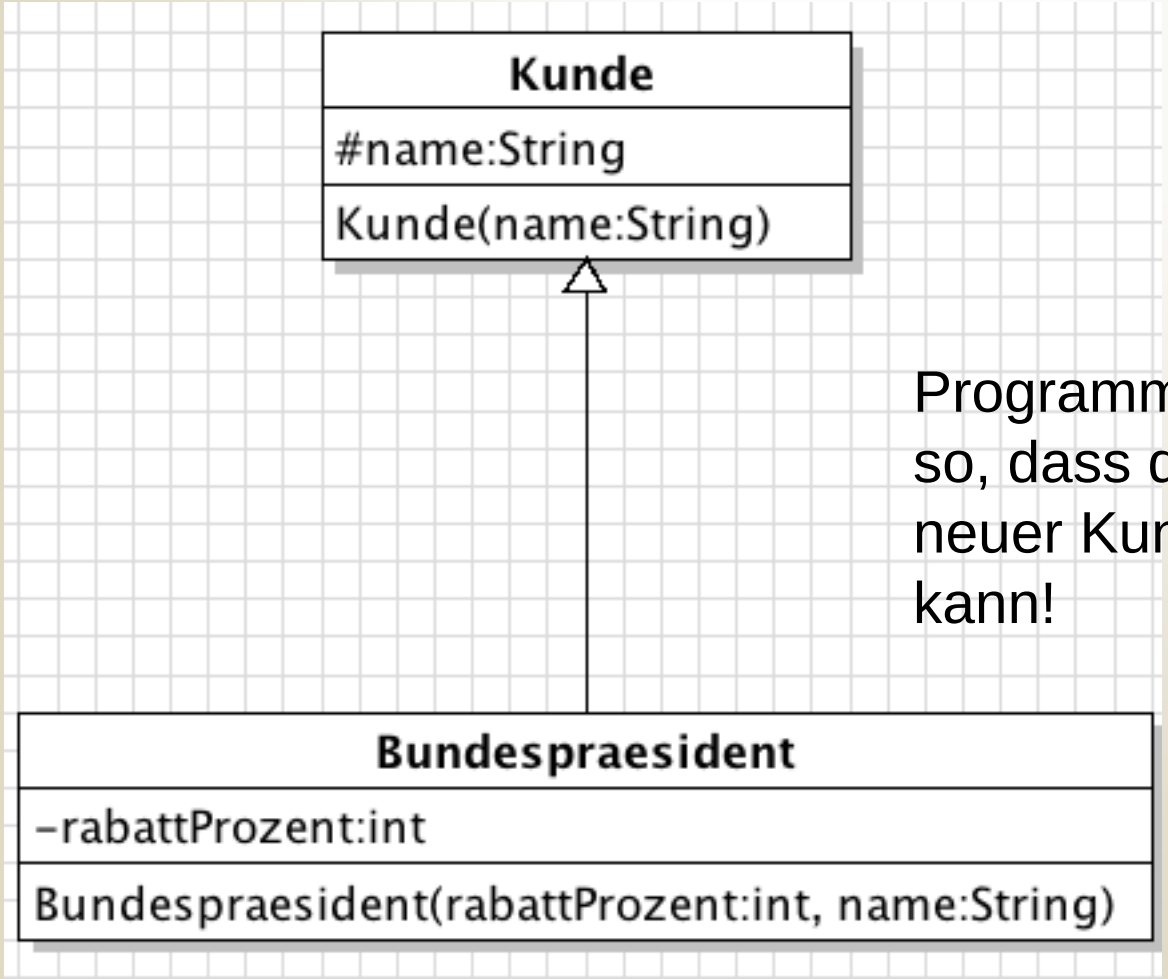
Übliches Vorgehen: Konstruktorwerte an Oberklasse übergeben

Der Elternklasse werden alle Parameter übergeben; diejenigen, die der Konstruktor der Elternklasse benötigt, werden mit dem super()-Aufruf "nach oben" weitergeleitet:

```
public class Vater {
    protected String nachname
    public Vater(String nachname){
        this.nachname = nachname;
} ...
```

```
public class Sohn extends Vater{
    protected boolean mopedfuehrerschein;
    public Sohn(boolean mopedfuehrerschein, String nachname){
        super(nachname);
        this.mopedfuehrerschein = mopedfuehrerschein;
    }
}
```

Aufgabe 4: parametrisiertes super() verwenden



Programmieren Sie diese Klassen so, dass der Bundespräsident als neuer Kunde instanziiert werden kann!

Um zu testen, ob alles funktioniert, gibt der Bundespraesident am Ende des Konstruktors aus "Ich heiße <name> und erhalte überall <rabattProzent> Prozent Rabatt."