

EINI LogWing/WiMa/MP

Einführung in die Informatik für
Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 25/26

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@tu-dortmund.de
<http://ls14-www.cs.tu-dortmund.de>

Thema

Kapitel 7

Objektorientierte Programmierung – Vererbung

EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

Unterlagen

- ▶ Dißmann, Stefan und Ernst-Erich Doberkat: *Einführung in die objektorientierte Programmierung mit Java*, 2. Auflage. München [u.a.]: Oldenbourg, 2002.
(→ ZB oder Volltext aus Uninetz)
- ▶ Echtle, Klaus und Michael Goedicke: *Lehrbuch der Programmierung mit Java*. Heidelberg: dpunkt-Verl, 2000. (→ ZB)

Übersicht

Begriffe

- ✓ Spezifikationen, Algorithmen, formale Sprachen
- ✓ Programmiersprachenkonzepte
- ✓ Grundlagen der imperativen Programmierung

✓ Algorithmen und Datenstrukturen

- ✓ Felder
- ✓ Sortieren
- ✓ Rekursive Datenstrukturen (Baum, binärer Baum, Heap)
- ✓ Heapsort

In diesem Kapitel:

- Prolog
 - Vererbung
 - Begriffe
 - Vererbung in Java
 - Attribute und Methoden
 - Polymorphie
- Objektorientierung
- ✓ Einführung
 - Vererbung
 - ▶ Anwendung

Gliederung

- Vererbung (anschaulich)
 - Transportmittel
 - Konto

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

- Begriffe
- Vererbung in Java
- Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- Polymorphie

Vererbung (anschaulich) I

Klassen können zueinander in einer "**ist ein**"-Beziehung stehen.

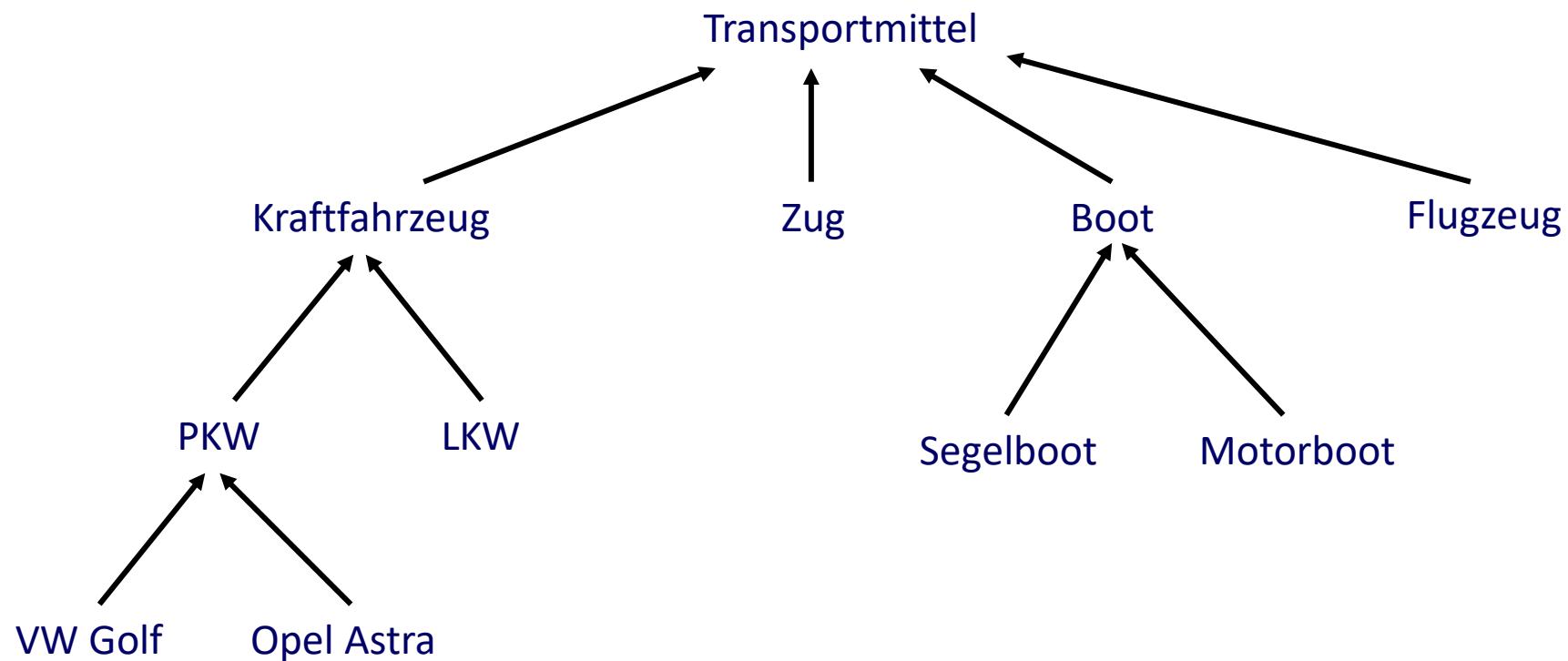
Beispiel

- ▶ Jeder PKW ist ein Kraftfahrzeug.
- ▶ Jedes Kraftfahrzeug ist ein Transportmittel.

Aber auch

- ▶ Jeder LKW ist ein Kraftfahrzeug.
- ▶ Jeder Zug,
- ▶ jedes Schiff und
- ▶ jedes Flugzeug ist ein **Transportmittel**.

Vererbung (anschaulich) II



Vererbung (anschaulich) III

EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- **Vererbung**
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

- ▶ Ein PKW besitzt
 - ▶ Fahrersitz und Fahrertür
 - ▶ die Funktion, den Sitz zu verstellen
 - ▶ die Funktion, die Fahrertür zu schließen
 - ▶ die Funktion, zu fahren

PKW
Fahrersitz Fahrertür Sitz_verstellen() Tür_schließen() Fahren()

- ▶ Ein LKW besitzt
 - ▶ Fahrersitz und Fahrertür
 - ▶ die Funktion, den Sitz zu verstellen
 - ▶ die Funktion, die Fahrertür zu schließen
 - ▶ die Funktion, zu fahren

LKW
Fahrersitz Fahrertür Sitz_verstellen() Tür_schließen() Fahren()

Vererbung (anschaulich) III

- ▶ PKWs haben jedoch
 - ▶ eigene Attribute: Rückbank und Kofferraum
 - ▶ und mit "hinten einsteigen" eigene Methoden.

- ▶ LKWs haben auch
 - ▶ eigene Attribute: Ladefläche und Anhänger
 - ▶ und "beladen" ist eine eigene Methode.

PKW
Fahrersitz
Fahrertür
Rückbank
Kofferraum
Sitz_verstellen()
Tür_schließen()
Fahren()
Hinten_einstiegen()

LKW
Fahrersitz
Fahrertür
Ladefläche
Anhänger
Sitz_verstellen()
Tür_schließen()
Fahren()
Beladen()

- PKWs und LKWs haben Gemeinsamkeiten.
- PKWs und LKWs haben Unterschiede.

EINI LogWing /
WiMa

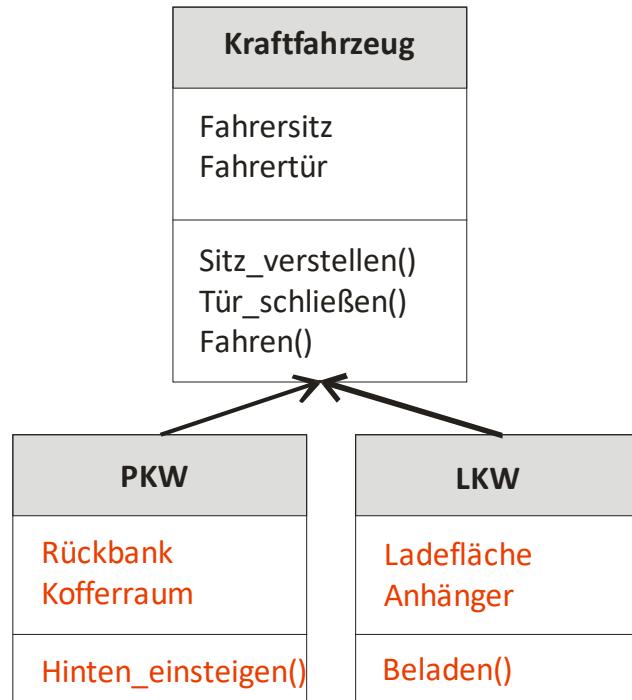
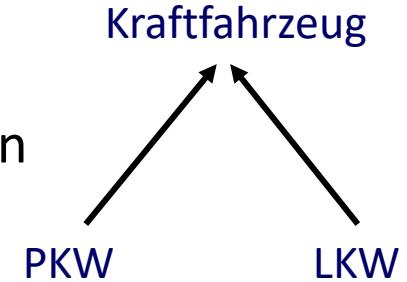
Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- **Vererbung**
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

Vererbung (anschaulich) IV

- ▶ Verwendung der Fahrzeughierarchie
 - ▶ Gemeinsamkeiten werden in dem übergeordneten Transportmittel beschrieben → **Allgemein**
 - ▶ Unterschiede in den untergeordneten Transportmitteln → **Speziell**



EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- **Vererbung**
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

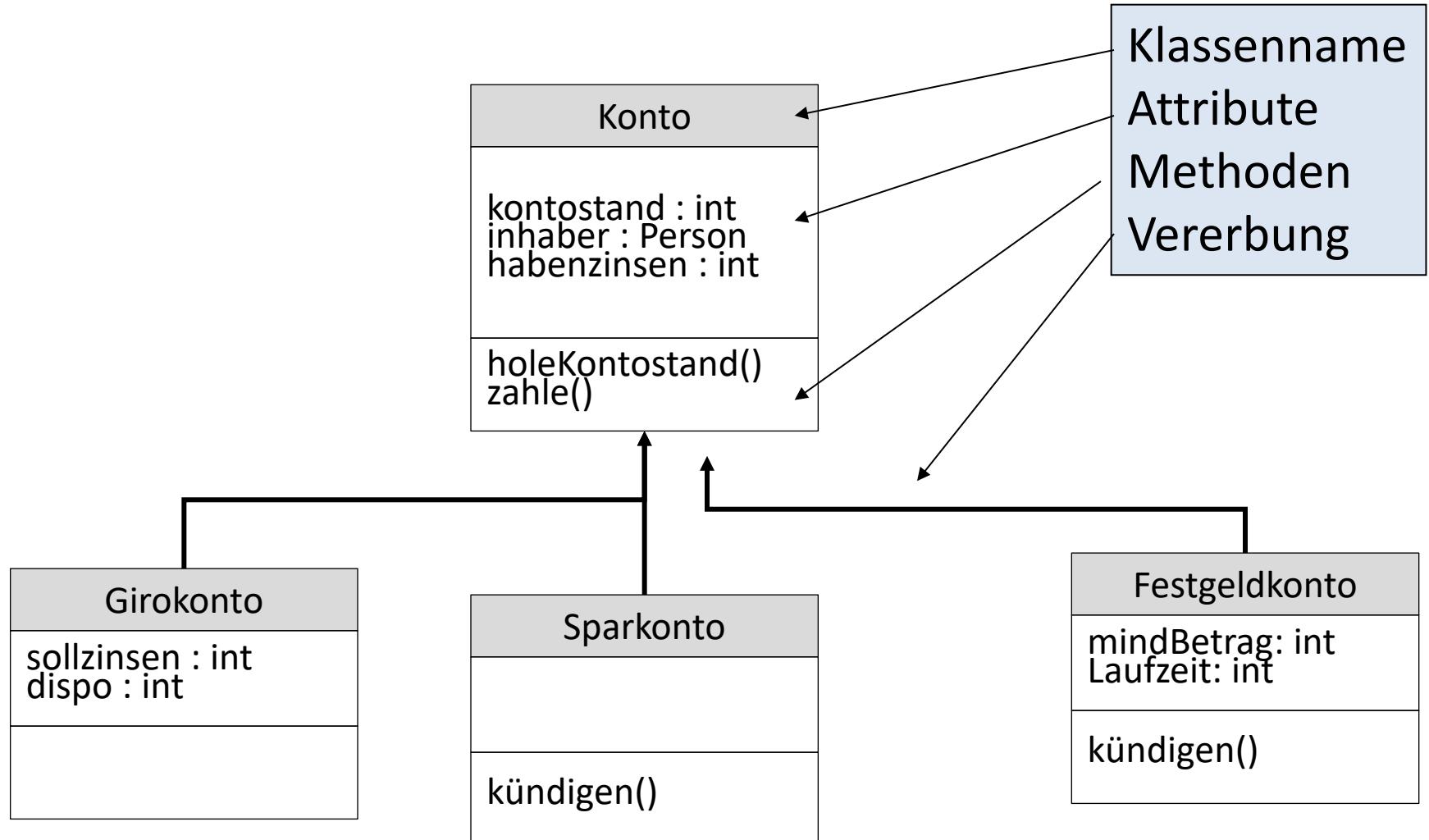
Vererbung I

Ähnlichkeiten bei Objekten – Beispiel Bankkonten

- ▶ Identifizieren von
 - Gemeinsamkeiten
 - Unterschieden

Girokonto	Sparkonto	Festgeld
kontostand: int inhaber: Person habenzinsen: int sollzinsen: int dispo: int	kontostand: int inhaber: Person habenzinsen: int	kontostand: int inhaber: Person habenzinsen: int mindBetrag: int laufzeit: int
holeKontostand() zahle()	holeKontostand() zahle() kündigen()	holeKontostand() zahle() kündigen()

Vererbung II



Gliederung

✓ Vererbung (anschaulich)

✓ Transportmittel

✓ Konto

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- **Begriffe**
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

➤ Begriffe

➤ Vererbung in Java

➤ Attribute & Methoden

➤ Zugriffsrechte

➤ Überschreiben

➤ abstrakte Methoden / Klassen

➤ Polymorphie

Begrifflichkeiten

- ▶ Die **vererbende** Klasse heißt Super- oder **Oberklasse**.
- ▶ Die **erbenden** Klassen sind Sub- oder **Unterklassen**.
 - ▶ Konto ist also die Super-/Oberklasse der Klassen Girokonto, Festgeldkonto und Sparkonto.
 - ▶ Diese sind wiederum die Sub-/Unterklassen der Klasse Konto.

Welche Möglichkeiten entstehen durch diese Konstruktion?

- ▶ Abstraktion und Spezialisierung
 - ▶ Attribute und Methoden werden möglichst problemadäquat zugeordnet.
 - ▶ Allgemeine Lösungen sind von allgemeinem Nutzen!

Gliederung

- ✓ Vererbung (anschaulich)
 - ✓ Transportmittel
 - ✓ Konto

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- **Vererbung in Java**
- Attribute und Methoden
- Polymorphie

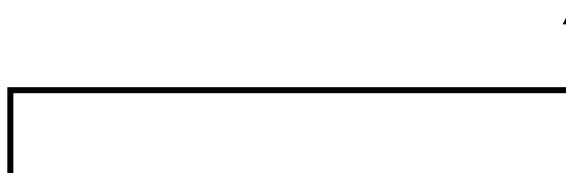
- ✓ Begriffe
- Vererbung in Java
- Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- Polymorphie

Beispiel: Die Klasse Konto

```
01 public class Konto {  
02     private String inhaber;  
03     private int habenZinsen;  
04     private int kontoStand;  
05  
06     public Konto(String inhaber) {  
07         this.inhaber = inhaber;  
08         this.kontoStand = 0;  
09         this.habenZinsen = 1;  
10     }  
11     public void zahle (int cent) {  
12         kontoStand += cent;  
13     }  
14     public int holeKontostand() {  
15         return (this.kontoStand);  
16     }  
17 } // Ende der Klasse Konto
```

Beispiel: Die Klasse Girokonto

```
public class Girokonto extends Konto {  
  
    private int sollZinsen;  
    private int dispo;  
  
} // Ende der Klasse Girokonto
```



EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- **Vererbung in Java**
- Attribute und Methoden
- Polymorphie

Vererbung in Java (technische Details) I

- Vererbung wird über das Schüsselwort **extends** realisiert:

```
class Unterkasse extends Oberkasse {  
    ... // Hier zusätzliche Attribute und Methoden  
}
```

- Die **neu definierte Unterkasse** erweitert also die **anderswo definierte Oberkasse** um
 - neue Attribute und
 - Methoden.
- Alle Methoden und Attribute der Oberklasse werden übernommen, wenn sie nicht als **private** deklariert sind.
 - Zu welchem Problem führt das?

Vererbung in Java (technische Details) II

- ▶ Ist **keine** Oberklasse definiert (kein **extends**), so ist die **Systemklasse**

Object

die Oberklasse.

- ▶ **Object** ist eine Oberklasse für alle Klassen (bis auf **Object** selbst).
- ▶ Aus wie vielen Oberklassen kann geerbt werden?
 - ▶ Java: Jede Klasse hat **genau eine Oberklasse**, nicht mehr und nicht weniger.

Vererbung in Java (technische Details) III

- ▶ Konstruktoren werden nicht vererbt.
- ▶ Konstruktoren der abgeleiteten Klasse müssen **neu definiert** werden.
- ▶ Über das Schlüsselwort **super** kann am Anfang eines Konstruktors der abgeleiteten Klasse ein Konstruktor der Oberklasse aufgerufen werden.

Beispiel:

```
class A {  
    A(String name) { ...  
}  
}  
}  
  
class B extends A {  
    B(String name, int a) {  
        super(name);  
        ...  
    }  
}
```

The diagram illustrates the inheritance relationship between Class A and Class B. It shows the constructor definitions in each class and the call to the superclass constructor in Class B. The code is annotated with arrows pointing to callout boxes:

- An arrow points from the constructor definition in Class A to the box "Konstruktor Klasse A".
- An arrow points from the constructor definition in Class B to the box "Konstruktor Klasse B".
- An arrow points from the `super(name);` call in Class B to the box "Aufruf des Oberklassenkonstruktors".

Vererbung in Java (technische Details) IV

- ▶ Wenn in der ersten Anweisung des Unterklassen-Konstruktors
 - ▶ keiner der Konstruktoren der Oberklasse aufgerufen wird,
 - ▶ dann wird der parameterlose Oberklassen-Konstruktor (**Standardkonstruktor**) automatisch aufgerufen,
 - ▶ bevor irgendeine andere Anweisung des Unterklassen-Konstruktors aufgerufen wird.

Weitere Fragestellungen

- ▶ Wie lassen sich die Variationen von Attributen und Methoden innerhalb der Hierarchie kontrollieren?

Gliederung

✓ Vererbung (anschaulich)

✓ Transportmittel

✓ Konto

EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

✓ Begriffe

✓ Vererbung in Java

➤ Attribute & Methoden

➤ Zugriffsrechte

➤ Überschreiben

➤ abstrakte Methoden / Klassen

➤ Polymorphie

Attribute und Methoden

Aufgrund der Beziehung in der Vererbung sind **Attribute und Methoden von Oberklassen** noch sinnvoll **nutzbar**.

- ▶ Folgefragen
 - ▶ Wie lassen sich bestehende Methoden anpassen?
 - ▶ Lässt sich diese Möglichkeit auch von der Oberklasse aus verhindern?
- ▶ Zugriffsrechte bisher
 - ▶ **private**: Zugriff nur innerhalb der Klasse (keine Vererbung)
 - ▶ **public**: Zugriff auch von außerhalb der Klasse (Vererbung, aber gleichzeitig völlig uneingeschränkter Zugriff)
- ▶ Gibt es auch Regelungen für die Zugriffsrechte innerhalb der Vererbungshierarchie?

Attribute und Methoden: Zugriffsrechte

protected (in Java)

- ▶ **private** Methoden und Attribute sind nur in der Klasse verwendbar, in der sie definiert sind. Sie sind **nicht in den erbenden Klassen verwendbar**.
- ▶ Oft sollen Methoden und Attribute nicht von außen verfügbar sein, aber gleichzeitig vererbt werden.
 - ▶ → Schlüsselwort **protected**
- ▶ **protected** Methoden und Attribute sind **in der Klasse selbst und in allen Unterklassen** sichtbar und verwendbar.

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- **Attribute und Methoden**
- Polymorphie

Attribute und Methoden: Zugriffsrechte (Beispiel)

```
01 public class Konto {  
02     protected String inhaber;  
03     protected int habenZinsen;  
04     private int kontoStand;  
05  
06     public Konto(String inhaber) {  
07         this.inhaber = inhaber;  
08         this.kontoStand = 0;  
09         this.habenZinsen = 1;  
10     }  
11     public void zahle (int cent) {  
12         kontoStand += cent;  
13     }  
14     public int holeKontostand() {  
15         return (this.kontoStand);  
16     }  
17 } // Ende der Klasse Konto
```

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:
• Prolog
• Vererbung
• Begriffe
• Vererbung in Java
• **Attribute und Methoden**
• Polymorphie

Attribute und Methoden: Überschreiben I

Überschreiben von Methoden in Vererbungshierarchien

- ▶ Aufgabenstellung: Berechnung von Zinsen
- ▶ Methode: **berechneZinsen (int tage)**
 - ▶ gleiche Implementierung in Sparkonto und Festgeld
 - → Sollzinsen existieren nicht.
 - ▶ aber: Berechnung aus Sollzinsen und Habenzinsen in **Girokonto**
- ▶ Lösung unter Nutzung der Vererbungshierarchie:
 - ▶ Standard-Implementierung in Konto
 - ▶ **Überschreiben der Methode in Girokonto für den Spezialfall**

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- **Attribute und Methoden**
- Polymorphie

Attribute und Methoden: Überschreiben II

Allgemeiner Fall wird in der Oberklasse implementiert.

```
public class Konto {
```

```
    ...
```

Definition in der Oberklasse

```
protected int berechneZinsen(int tage) {
```

```
    int zinsen =
```

```
        kontoStand*(habenZinsen/100)*(tage/365);
```

```
    return (zinsen);
```

```
}
```

```
    ...
```

```
}
```

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- **Attribute und Methoden**
- Polymorphie

Attribute und Methoden: Überschreiben III

Spezieller Fall überschreibt Methode der Oberklasse.

```
public class Girokonto extends Konto {
```

```
    ...
```

Überschreiben der
Definition

```
protected int berechneZinsen(int tage) {  
  
    int guthaben = holeKontostand();  
    int zinsen;  
  
    if (guthaben > 0) {  
        zinsen = guthaben * (habenzinsen/100)*(tage/365);  
    } else {  
        zinsen = -guthaben * (sollzinsen/100)*(tage/365);  
    }  
  
    return (zinsen);  
}
```

```
}
```

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- **Attribute und Methoden**
- Polymorphie

Attribute und Methoden: Überschreiben IV

Zugriff auf überschriebene Attribute / Methoden

- ▶ In einem Objekt einer abgeleiteten Klasse ist **super** eine Referenz auf das **Teilobjekt der Oberklasse**.
- ▶ Attribute und Methoden der Oberklasse lassen sich so ansprechen (auch überschriebene Attribute und Methoden).
- ▶ Beispiel

```
class A {           class B extends A {  
    int variable;      int variable;  
    void methode() {  void methode() {  
        ...             ...  
    }                 }  
}                   void methode2() {           // Zugriff auf  
                                super.variable = 3; // überschriebene  
                                super.methode(); // Attribute und  
}                         // Methoden der  
                        // Oberklasse
```

Attributen und Methoden: Überschreiben V

Schlüsselwort: **final**

- ▶ Verhindert, dass eine **Methode** überschrieben wird

```
public final int holeKontostand() { ... }
```

- ▶ Verbietet erben von einer **Klasse**

```
public final class Girokonto extends Konto  
{
```

 • • •

```
}
```

- ▶ Alle Methoden und Attribute einer finalen Klasse sind implizit auch final.

Attributen und Methoden: Überschreiben VI

Schlüsselwort: **final**

- ▶ Finale Klassen und Methoden sind zuweilen aus Sicherheitsgründen erforderlich
 - ▶ Aufgabe ist festgelegt.
 - ▶ Manipulation ist nicht möglich.
- ▶ Typische Anweisung: eine Methode zur Passwort-Prüfung
- ▶ **final- Attribute** sind Konstanten.
 - ▶ Sie dürfen nicht verändert werden!
 - ▶ Beispiel

```
public final int mwst;
```

Abstrakte Methoden/Klassen

► Situation

Jede Unterklasse hat die gleiche Methode aber unterschiedliche Implementierung.

► Beispiel: **auszahlen(int betrag)**

- ▶ Girokonto: beliebige Auszahlung bis Limit
- ▶ Sparkonto: Restguthaben von € 5,- nötig (außer nach Kündigung)
- ▶ Festgeld: Auszahlung erst nach Ende der Laufzeit

► Lösung: **abstrakte Methode** in der Oberklasse.

- ▶ Eine abstrakte Methode ist eine **Methode, die nicht realisiert ist.**
- ▶ Die **abstrakte Methode** der Oberklasse gibt **nur die Signatur** der Methode an, nicht aber ihre Realisierung.

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- **Attribute und Methoden**
- Polymorphie

Abstrakte Methoden/Klassen: Beispiel I

```
public abstract class Konto {
```

...

Einzahlen() für alle
Unterklassen gleich

```
    public void einzahlen(int betrag) {  
        zahle(betrag);  
    }
```

...

```
    public abstract int auszahlen(int betrag);
```

}



Auszahlen() für alle
Unterklassen
unterschiedlich

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

Abstrakte Methoden/Klassen: Beispiel II

```
class Girokonto extends Konto {
```

...

Hier die konkrete
Realisierung für die
Unterklasse

```
public int auszahlen(int betrag) {  
  
    if (kontostand-betrag > dispo) {  
        zahle(-betrag);  
        return (betrag);  
    } else {  
        System.out.println("Auszahlen nicht möglich");  
        return (0);  
    }  
}
```

}

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

Abstrakte Methoden/Klassen

- ▶ Enthält eine Klasse eine abstrakte Methode, so ist die ganze Klasse **abstract**.

- ▶ Eine **abstrakte Klasse kann nicht instanziert werden.**
 - ▶ D.h.: Es können keine Objekte zu dieser Klasse erzeugt werden.
 - ▶ Es kann nur Objekte zu den nicht abstrakten Unterklassen geben.

- ▶ Abstrakte Methoden **müssen** in den Unterklassen implementiert werden (oder die Unterklassen sind wieder abstrakt).

Gliederung

✓ Vererbung (anschaulich)

✓ Transportmittel

✓ Konto

EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- Polymorphie

✓ Begriffe

✓ Vererbung in Java

✓ Attribute & Methode

✓ Zugriffsrechte

✓ Überschreiben

✓ abstrakte Methoden / Klassen

➤ Polymorphie

Polymorphie I

Darstellung aus mengentheoretischer Sicht

► Alle Objekte sind Konten.

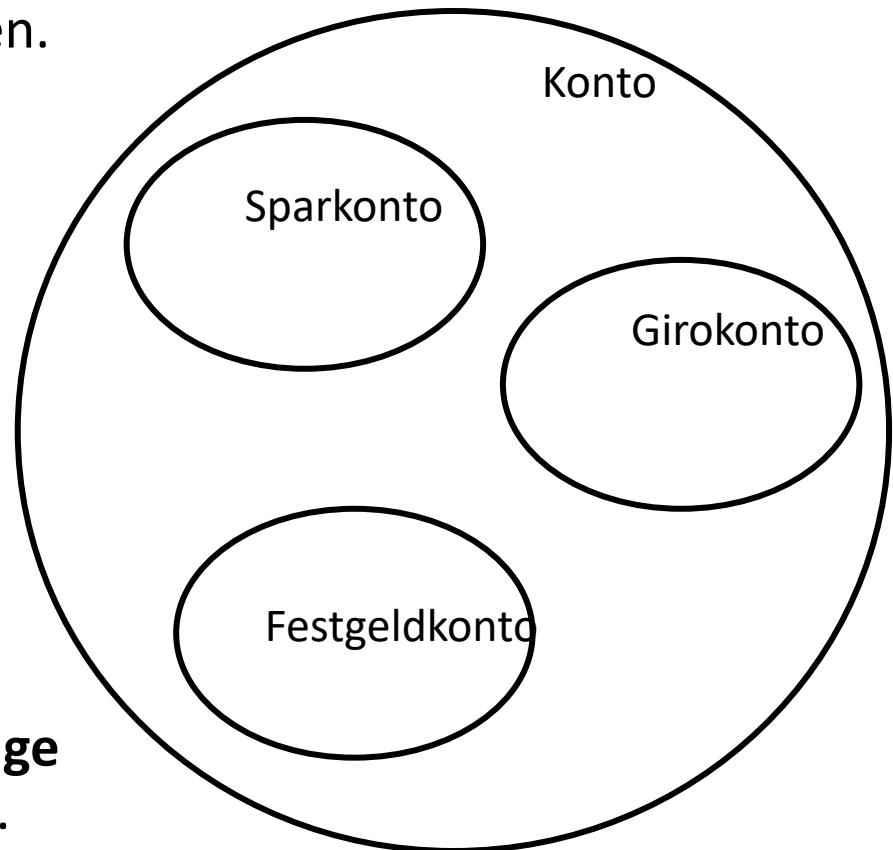
► Einige sind besondere Arten von Konten.

► Die **Menge** der

- Sparkonten,
- Girokonten und
- Festgeldkonten

► ist jeweils eine **Teilmenge** der **Menge der Konten**.

► Die Teilmengen sind **disjunkt**.



EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

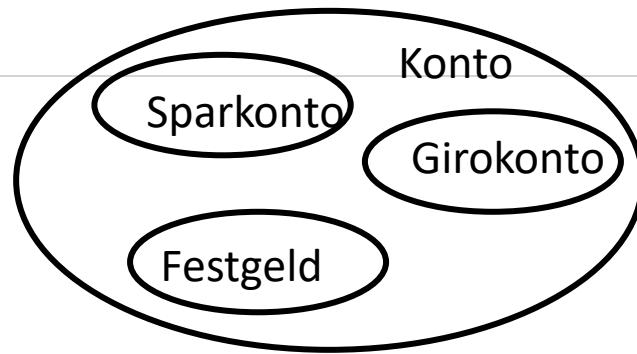
In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Polymorphie II

Wir nehmen folgende **Deklarationen** an:

- ▶ Girokonto einGirokonto;
- ▶ Sparkonto einSparkonto;
- ▶ Konto einKonto, einAnderesKonto;



Legale Zuweisungen:

- ▶ einGirokonto = new Girokonto();
- ▶ einSparkonto = new Sparkonto();
- ▶ einGirokonto.sollzinsen = 12;
- ▶ einKonto = einGirokonto;
- ▶ einAnderesKonto = new Sparkonto();

Illegalen Zuweisungen:

- ▶ einSparkonto = einGirokonto;
- ▶ einGirokonto = new Sparkonto();

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Polymorphie III

Jedes Sparkonto oder Girokonto ist auch ein Konto, deshalb ist
einKonto = einSparkonto;
legal.

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Ein Objekt einer Klasse kann also mehrere Erscheinungsformen haben:

- ▶ Es kann
 - ▶ ein Objekt der Klasse selbst oder
 - ▶ ein Objekt einer der Unterklassen dieser Klasse oder
 - ▶ ein Objekt einer der Oberklassen dieser Klasse sein.
- ▶ Das Objekt bewegt sich in der **Vererbungshierarchie**.

→ Es ist **polymorph**.

Polymorphie IV

Nicht jedes Konto ist ein Sparkonto!

► Ist dann

einSparkonto = (Sparkonto)einKonto;
legal?

EINI LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

► Ja!

- Denn Objekte der Klasse Sparkonto sind wandelbar zu Objekten der Klasse Konto.
- Allerdings ist **kein Zugriff auf alle Attribute** möglich:
 - **einKonto** hat ja nicht die **Sparkonto**-Attribute.

Polymorphie V

Was passiert bei folgender Anweisung?

```
if (x == 1)
    einKonto = einSparkonto;
else
    einKonto = einGirokonto;
```

- ▶ Der Compiler ist **nicht** in der Lage, die Klasse von **einKonto** zu ermitteln.
- ▶ Die Klasse von **einKonto** ist nach dieser Zuweisung **nicht vorhersehbar**.
- ▶ **einKonto** kann also nach dieser Anweisung eine von mehreren Klassen haben (= **polymorph**).

Polymorphie VI

Wunsch

- ▶ Alle Objekte aus der Oberklasse “Konto” sollen in der gleichen Weise behandelt werden können.

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Lösung: Polymorphie

- ▶ Eine Oberklassen-Referenz kann auch auf Objekte der Unterklassen verweisen.
 - ▶ Methoden der Oberklasse können so aufgerufen werden.
 - ▶ Wurde eine **Methode** von einer **Unterklasse** überschrieben,
 - so wird nicht die Methodenimplementierung der Oberklasse aufgerufen,
 - sondern die **Implementierung der Unterklasse**.

Polymorphie VII

- Methoden können so mit allen möglichen Konten arbeiten:

```
public int berechneVermoege(Konto[ ] konten) {  
    int vermoegen = 0;  
  
    for (int i=0; i<konten.length; i++) {  
        Konto k = konten[i];  
        vermoegen += k.holeKontostand();  
    }  
  
    return (vermoegen);  
}
```

- Der Methodenaufruf wird an die entsprechende Subklasse weitergeleitet.

Vererbung: Zusammenfassung I

► Vererbung

- ▶ Klassen können als **Unterklasse** von einer Klasse definiert werden.
- ▶ Java: Vererbungshierarchie mit **1 Oberklasse** je Klasse.

► Folgen

- ▶ Behandlung namens-/signaturgleicher Methoden in Ober-/Unterklassen, Zugriffsmöglichkeiten auf verdeckte Attribute und Methoden
- ▶ Erweiterung der Definition von Zugriffsrechten: **private, public, protected**
- ▶ Behandlung von **abstrakten** („noch zu implementierenden“) Methoden.
- ▶ Begrenzung der Möglichkeit des Überschreibens: **final**

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Vererbung: Zusammenfassung II

► Nutzen

- ▶ Erlaubt **allgemeine Lösungen in Spezialfällen** ohne redundanten Code zu nutzen.
- ▶ Erlaubt Anforderungen zu spezifizieren: **abstrakte Klassen**
- ▶ Erlaubt Abwandlung von Methoden: **Überschreiben** (bei gleicher Signatur)
 - ❖ Achtung! Nicht mit **Überladen** verwechseln (ungleiche Signatur).

EINI LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**



Vererbung

Artikel im EINI-Wiki:

- **Vererbung**
- **Konstruktor**
- **Sichtbarkeit**
- **Final**
- **Signatur**

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

Übersicht



Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- ▶ Nächste Vorlesung – WiMa 22.1.2026, 08:15
 - ▶ Nächste Vorlesung – LogWing 23.1.2026, 08:15