

EINI LW

**Einführung in die Informatik für
Naturwissenschaftler und
Ingenieure**

Vorlesung 2 SWS WS 15/16

Dr. Lars Hildebrand

Fakultät für Informatik – Technische Universität Dortmund

lars.hildebrand@tu-dortmund.de

<http://ls1-www.cs.tu-dortmund.de>

Thema

▶ Kapitel 7

Objektorientierte Programmierung - Vererbung

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

▶ Unterlagen

- ▶ Echte, Goedicke: Einführung in die objektorientierte Programmierung mit Java, dpunkt-Verlag.
- ▶ Doberkat, Dissmann: Einführung in die objektorientierte Programmierung mit Java, Oldenbourg-Verlag, 2. Auflage.

In diesem Kapitel:

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

Übersicht

Begriffe

- ▶ Spezifikationen, Algorithmen, formale Sprachen, Grammatik
- ▶ Programmiersprachenkonzepte
- ▶ Grundlagen der Programmierung

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

- ▶ 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

Eini LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

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

- ▶ Vererbung (anschaulich)
 - ▶ Transportmittel
 - ▶ Konto
- ▶ Begriffe
- ▶ Vererbung in Java
- ▶ Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- ▶ Polymorphie

In diesem Kapitel:

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

Vererbung (anschaulich)

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

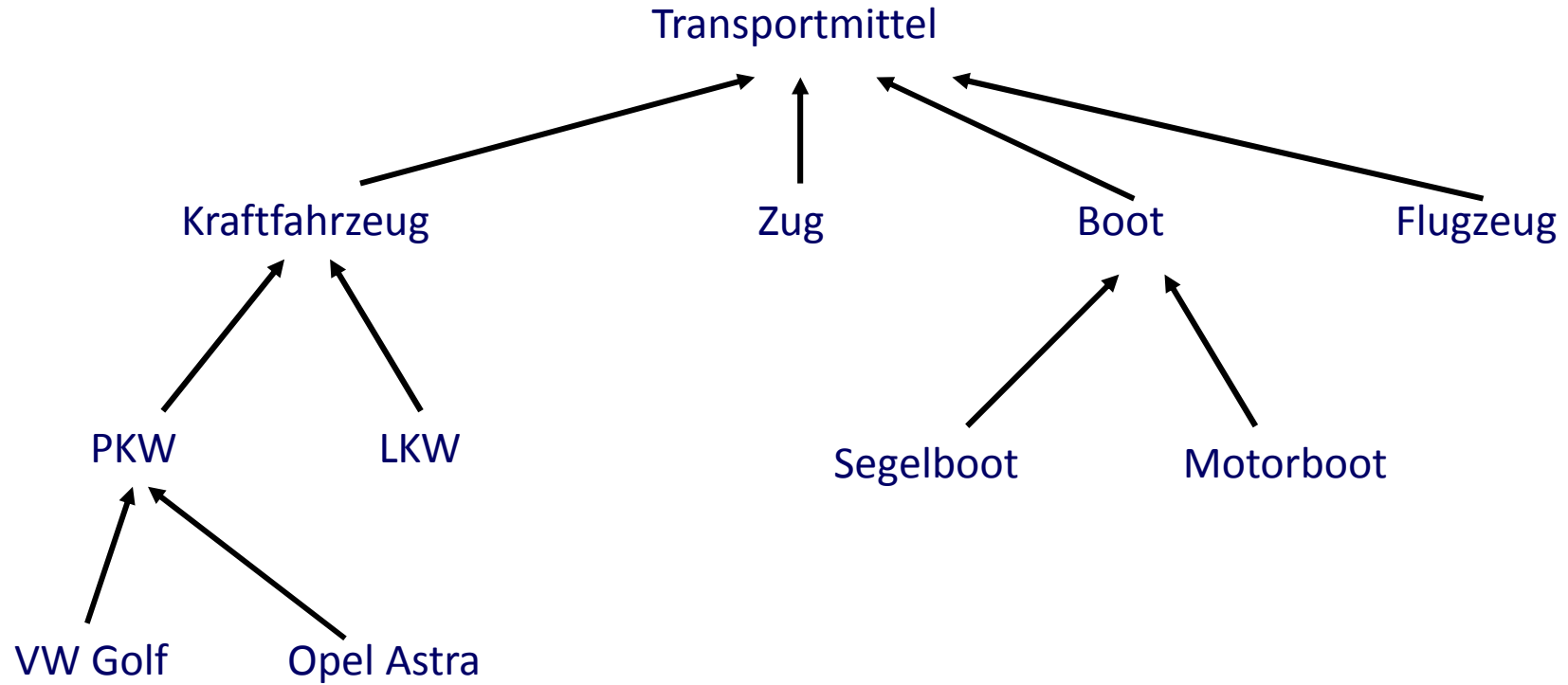
Beispiel

- ▶ 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)



In diesem Kapitel:

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

Vererbung (anschaulich)

- ▶ Ein **PKW** besitzt
 - ▶ **Fahrsitz** und **Fahrtür**
 - ▶ Funktion, den **Sitz zu verstellen**
 - ▶ Funktion, die **Fahrtür zu schließen**
 - ▶ Funktion, **zu fahren**

PKW
Fahrsitz Fahrtür
Sitz_verstellen() Tür_schließen() Fahren()

- ▶ Ein **LKW** besitzt
 - ▶ **Fahrsitz** und **Fahrtür**
 - ▶ Funktion, den **Sitz zu verstellen**
 - ▶ Funktion, die **Fahrtür zu schließen**
 - ▶ Funktion, **zu fahren**

LKW
Fahrsitz Fahrtür
Sitz_verstellen() Tür_schließen() Fahren()

Vererbung (anschaulich)

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

- ▶ PKWs haben jedoch mit der
 - ▶ Rückbank und dem Kofferraum eigene Attribute
 - ▶ und mit "hinten einsteigen" eigene Methoden
- ▶ LKWs haben mit der
 - ▶ Ladefläche und dem Anhänger auch eigene Attribute
 - ▶ und "beladen" ist eine eigene Methode

→ PKW und LKW haben Gemeinsamkeiten

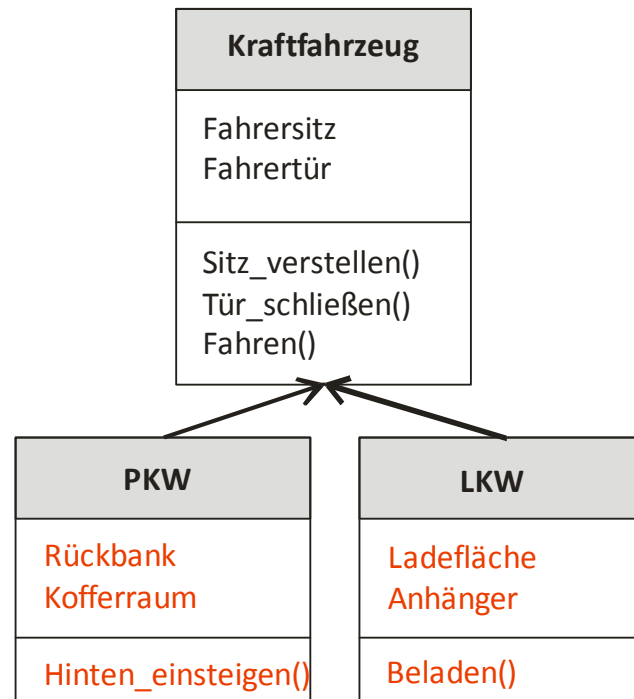
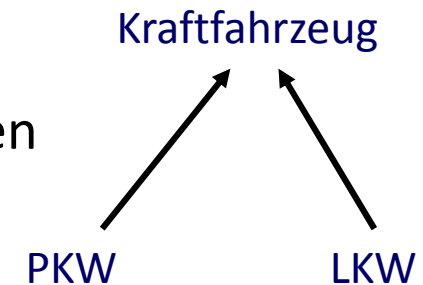
→ PKW und LKW haben Unterschiede

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

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

Vererbung (anschaulich)

- ▶ Wir nutzen nun die Fahrzeughierarchie, die wir beschrieben haben
 - ▶ 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

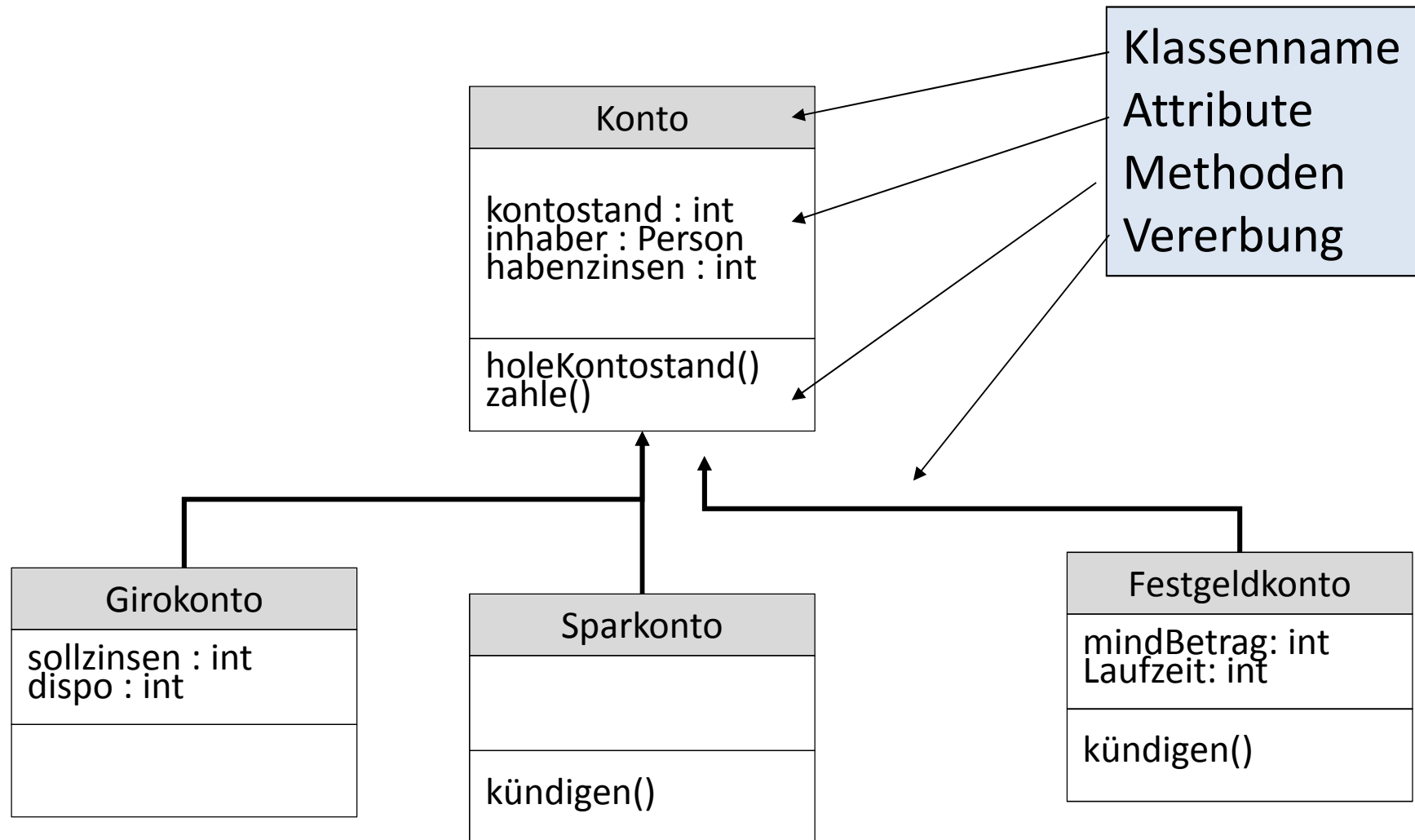
Vererbung

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
holeKontostand() zahle()	holeKontostand() zahle() kündigen()	mindBetrag: int laufzeit: int
		holeKontostand() zahle() kündigen()

Vererbung



In diesem Kapitel:

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

Gliederung

- ▶ Vererbung (anschaulich)
 - ▶ Transportmittel
 - ▶ Konto
- ▶ **Begriffe**
- ▶ Vererbung in Java
- ▶ Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- ▶ Polymorphie

In diesem Kapitel:

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

Begrifflichkeiten

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

Welche Möglichkeiten entstehen durch diese Konstruktion?

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

In diesem Kapitel:

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

Gliederung

- ▶ Vererbung (anschaulich)
 - ▶ Transportmittel
 - ▶ Konto
- ▶ 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
```

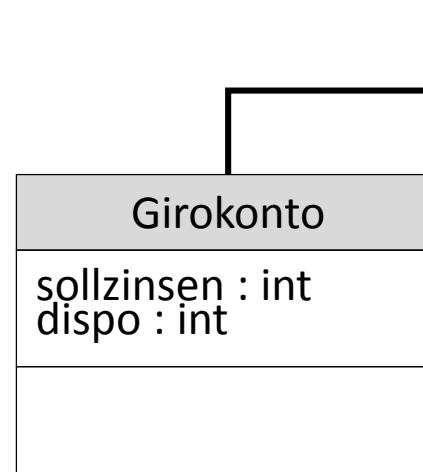
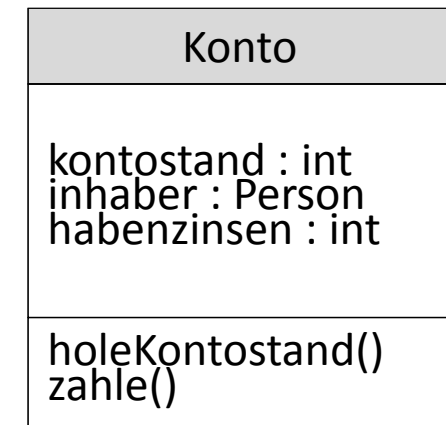
Beispiel: Die Klasse Girokonto

```
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)

- ▶ Vererbung wird über Schlüsselwort **extends** realisiert:

```
class Unterklasse extends Oberklasse {  
    ... // Hier zusätzliche Attribute und Methoden  
}
```

- ▶ Die **neu definierte Unterklasse** erweitert also die **anderswo definierte Oberklasse** um
 - ▶ neue Attribute und
 - ▶ Methoden
- ▶ Alle Methoden und Attribute der Oberklasse werden übernommen, wenn sie nicht **private** deklariert sind
 - ▶ oh, haben wir nun ein Problem?

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)

- ▶ 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 wievielen Oberklassen kann geerbt werden?
 - ▶ Java: jede Klasse hat genau eine Oberklasse.
 - ▶ Nicht mehr.
 - ▶ Nicht weniger.

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)

- ▶ Konstruktoren werden nicht vererbt, Konstruktoren der abgeleiteten Klasse müssen neu definiert werden!
- ▶ Über 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);  
        ...  
    }  
}
```

Konstruktor Klasse A

Konstruktor Klasse B

Aufruf des
Oberklassen-
konstruktors

In diesem Kapitel:

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

Vererbung in Java (technische Details)

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

Weitere Fragestellungen:

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

Gliederung

Eini LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

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

- ▶ Vererbung (anschaulich)
 - ▶ Transportmittel
 - ▶ Konto
- ▶ Begriffe
- ▶ Vererbung in Java
- ▶ Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- ▶ Polymorphie

In diesem Kapitel:

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

Attribute und Methoden

- ▶ Aufgrund der Beziehung in der Vererbung sind Attribute & 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 Regelungen auch für die Zugriffsrechte innerhalb der Vererbungshierarchie?

Attribute und Methoden - Zugriffsrechte

Zugriffsrecht: protected (in Java)

Eini LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

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

▶ **private** Methoden und Attribute sind nur in der Klasse zugreifbar, in der sie definiert sind. Sie sind **nicht in den erbenden Klassen zugreifbar**

▶ Oft ist es so, dass Methoden und Attribute nicht von außen zugreifbar sein sollen, dass sie aber schon vererbt werden sollen. Genau dies wird durch das Schlüsselwort **protected** vereinbart

▶ **protected** Methoden und Attribute sind in der Klasse selbst und in allen Unterklassen sichtbar und zugreifbar

Attribute und Methoden - Zugriffsrechte

```
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

Überschreiben von Methoden in Vererbungshierarchien

- ▶ Aufgabenstellung: Berechnung von Zinsen
- ▶ Methode: `berechneZinsen (int tage)`
 - ▶ gleiche Implementierung in `Sparkonto` und `Festgeld`
 - ▶ da keine Sollzinsen existieren
 - ▶ aber: in `Girokonto` Berechnung aus Sollzinsen und Habenzinsen
- ▶ 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

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);  
}
```

```
...
```

```
}
```

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

Attribute und Methoden - Überschreiben

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

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

Attributen & Methoden - Überschreiben

Schlüsselwort: final

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

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

- ▶ Erben von einer **Klasse** verbieten:

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

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

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

Attributen & Methoden - Überschreiben

Schlüsselwort: final

- ▶ Finale **Klassen** und **Methoden** sind aus Sicherheitsgründen zuweilen erforderlich.
 - ▶ Sie tun das, was sie tun sollen und
 - ▶ können nicht manipuliert werden.
- ▶ Typische Anweisung: eine Methode zur Passwort-Prüfung.
- ▶ **final** - **Attribute** sind Konstanten, sie dürfen nicht verändert werden.
 - ▶ Beispiel:

```
public final int mwst;
```

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

In diesem Kapitel:

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

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** dann **nur** die **Signatur** der Methode **an**, nicht aber ihre Realisierung.

Abstrakte Methoden/Klassen

```
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

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

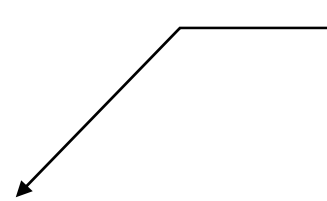
In diesem Kapitel:

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

Abstrakte Methoden/Klassen

```
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("Kein Auszahle möglich");  
        return (0);  
    }  
}
```

```
}
```

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

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 instanziiert 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)

In diesem Kapitel:

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

Gliederung

- ▶ Vererbung (anschaulich)
 - ▶ Transportmittel
 - ▶ Konto
- ▶ Begriffe
- ▶ Vererbung in Java
- ▶ Attribute & Methoden
 - ▶ Zugriffsrechte
 - ▶ Überschreiben
 - ▶ abstrakte Methoden / Klassen
- ▶ **Polymorphie**

Polymorphie

Darstellung aus mengentheoretischer Sicht

▶ Alle Objekte sind Konten!

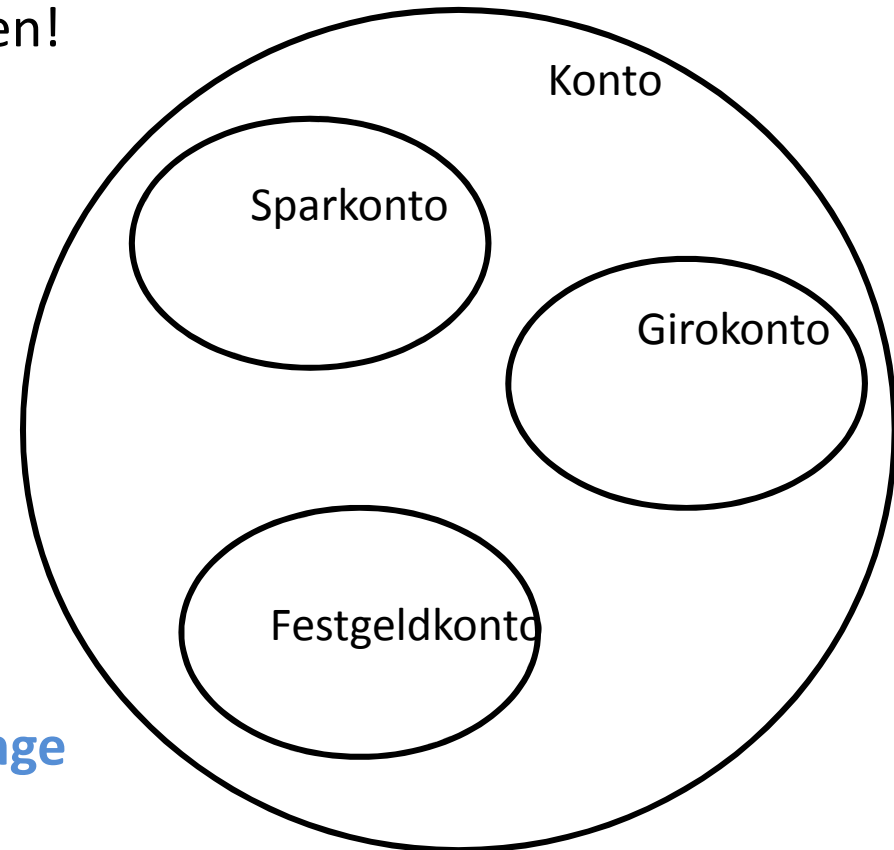
▶ Einige sind besondere Arten von Konten.

▶ Die **Menge** der

- ▶ Sparkonten,
- ▶ Girokonten,
- ▶ Festgeldkonten

▶ ist jeweils eine **Teilmenge** der **Menge der Konten**

▶ Die Teilmengen sind **disjunkt**



Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

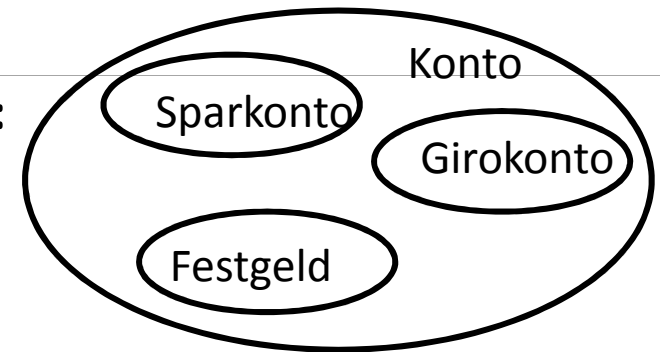
- Prolog
- Vererbung
- Begriffe
- Vererbung in Java
- Attribute und Methoden
- **Polymorphie**

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

Polymorphie

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();`

Illegale Zuweisungen:

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

Polymorphie

Jedes Sparkonto / Girokonto ist auch ein Konto, deshalb ist

einKonto = einSparkonto;

legal.

Ein Objekt einer Klasse kann also mehrere Erscheinungsformen haben:

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

Eini LogWing /
WiMa

Kapitel 7
Objektorientierung
- Vererbung

In diesem Kapitel:

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

Polymorphie

Nicht jedes Konto ist ein Sparkonto.

▶ Ist dann

```
einSparkonto = (Sparkonto) einKonto;
```

legal?

▶ Ja, denn Objekte der Klasse Sparkonto sind wandelbar zu Objekten der Klasse Konto.

- ▶ Allerdings ist der Zugriff auf **alle Attribute nicht möglich**,
- ▶ denn einKonto hat ja nicht die Sparkonto-Attribute.

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

Polymorphie

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** nach dieser Zuweisung ist **nicht vorhersehbar**.
- ▶ **einKonto** kann also nach der Anweisung eine von mehreren Klassen haben, es ist halt **polymorph**.

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

In diesem Kapitel:

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

Polymorphie

Wunsch:

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

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

- ▶ Methoden können so mit allen möglichen Konten arbeiten

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

- ▶ Methodenaufruf wird an die entsprechende Subklasse weitergeleitet

Eini LogWing /
WiMa

Kapitel 7

Objektorientierung
- Vererbung

In diesem Kapitel:

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

In diesem Kapitel:

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

Vererbung - Zusammenfassung

▶ 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 & Methoden
- ▶ Erweiterung der Definition von Zugriffsrechten: (private, public, protected)
- ▶ Behandlung von abstrakten („noch zu implementierenden“) Methoden
- ▶ Begrenzung der Möglichkeit des Überschreibens: final

Vererbung - Zusammenfassung

▶ 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**

Übersicht



Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- ▶ Nächste Vorlesung – WiMa 21.1.2016, 08:15
- ▶ Nächste Vorlesung – LogWIng 22.1.2016, 08:15