

Programmierung 2

Kapitel 5 – Vererbung

Motivation

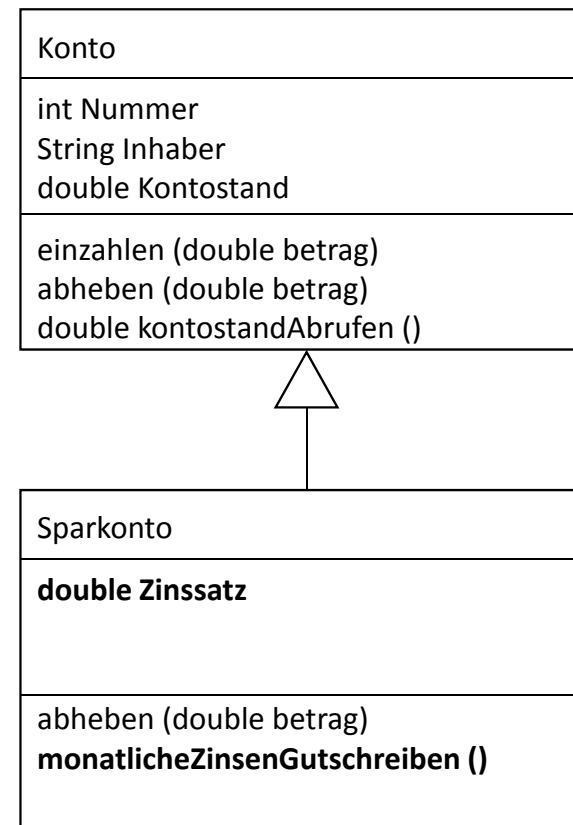
- Verfeinerung einer Klasse durch Spezialisierung
 - Unterklasse hat alle Merkmale der Oberklasse
 - Unterklasse hat zusätzliche Merkmale

Konto
int Nummer String Inhaber double Kontostand
einzahlen (double betrag) abheben (double betrag) double kontostandAbrufen ()

Sparkonto
int Nummer String Inhaber double Kontostand double Zinssatz
einzahlen (double betrag) abheben (double betrag) double kontostandAbrufen () monatlicheZinsenGutschreiben()

Motivation

- Sparkonto „erbt“ Merkmale von Konto
 - Objekte der Klasse Sparkonto haben alle Methoden und Attribute der Klasse Konto
- Sparkonto überschreibt die Methode **abheben()**
 - Auszahlungen nur mit Vorlage des Sparbuchs
- Sparkonto hat zusätzliche Merkmale
 - Attribute **Zinssatz**, z.B. 3% pro Jahr
 - Methode **monatlicheZinsenGutschreiben()**, für die monatliche Berechnung der Zinsen

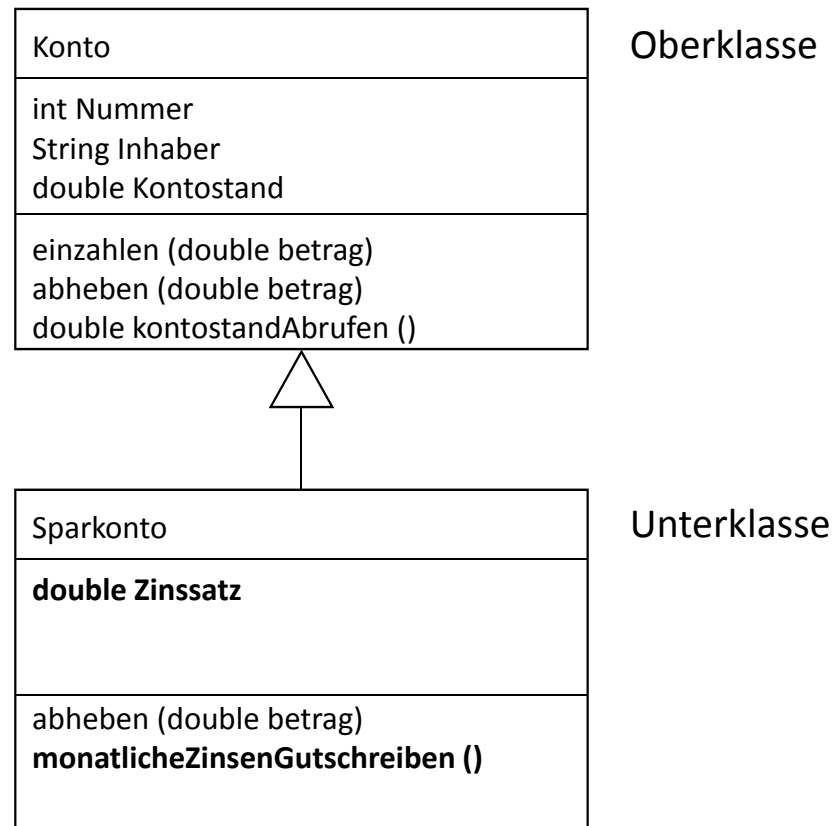


*Alle Methoden
und Attribute?*

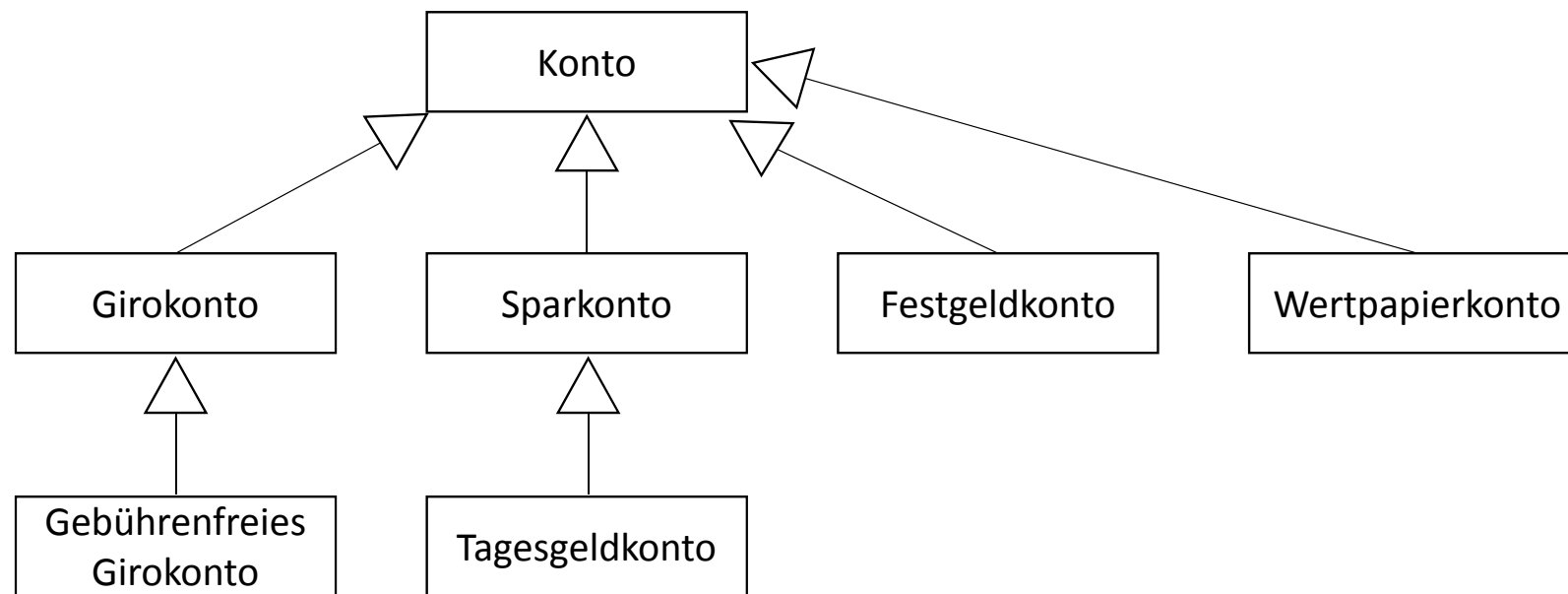
Definitionen und Begriffe

- **Oberklasse:** Klasse, von der eine andere abgeleitet wurde
- **Unterklasse:** Klasse, die von einer anderen abgeleitet wurde
- **Vererbung:** Strukturierungsprinzip bei der Klassendefinition
 - Unterklassen durch Erweiterung bzw. Modifikation einer bereits existierenden Oberklassen definieren
 - Unterklassen erben Attribute und Methoden der Oberklasse
- **Ableitung:** Prozess der Klassenbildung durch Vererbung
- **Klassenhierarchie:** mehrfach fortgesetzte Definition von Klassen durch Vererbung führt zu einer Klassenhierarchie
- **Signatur:** Menge der von außen sichtbaren Methoden und Attribute

Beispiel: Konto

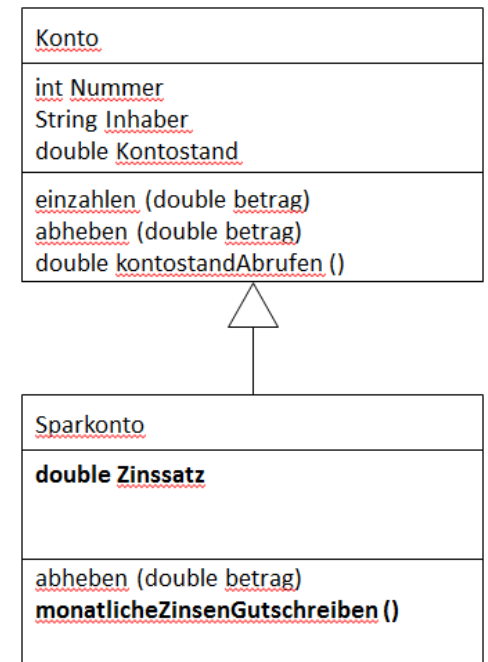


Klassenhierarchie Konto



Vorteile Vererbung

- Methoden der Oberklasse erhalten eine neue Implementierung
- Unterklasse erhält zusätzliche Methoden
- Unterklasse nutzt Attribute der Oberklasse anders
- Unterklasse erhält zusätzliche Attribute
- Warum ist das nützlich?
 - Code Wiederverwendung ohne Copy&Paste
 - Reduzierung Fehlerquellen, vereinfachte Wartung
 - Kapselung von Eigenschaften der Unterklasse



Vererbung in Java

- Vererbung mit Schlüsselwort **extends**
public class Unterklasse **extends** Oberklasse {}

Beispiel Sparkonto


Sparkonto ist abgeleitet von Klasse Konto

```
public class Sparkonto extends Konto {  
    static double zinssatz = 3.0;  
  
    public Sparkonto( int nummer, String name, double betrag ) {  
        // Aufruf Konstruktor der Oberklasse  
        super(nummer, name, betrag);  
    }  
  
    public void abheben( double betrag ) {  
        ...  
    }  
  
    void monatlicheZinsenGutschreiben() {  
        ...  
    }  
}
```

Klasse Object

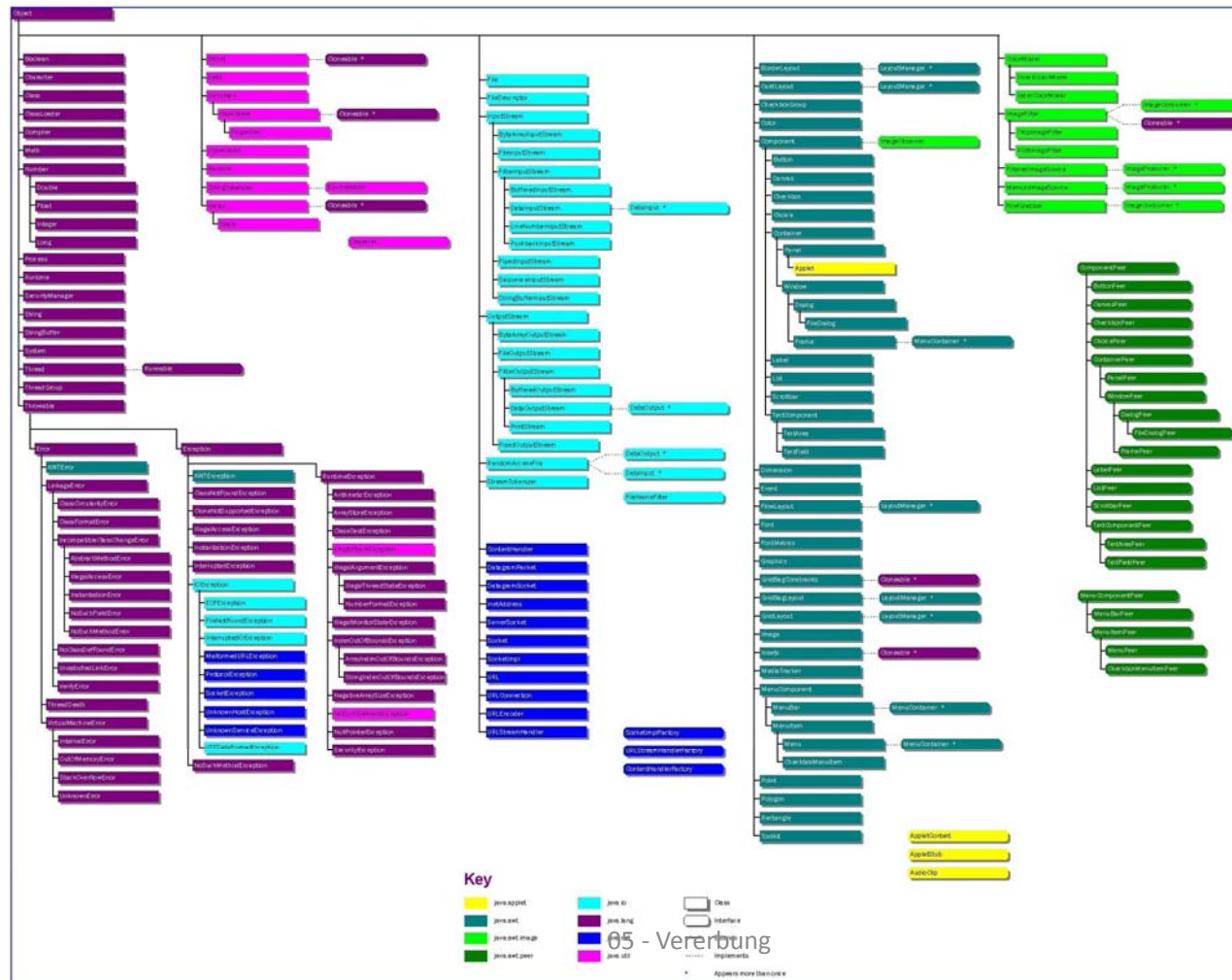
- Implizite Vererbung von Klasse Object
 - Schlüsselwort **extends** nicht notwendig

```
public class Object {  
  
    public Object() {}  
  
    public boolean equals(Object obj) {  
        return (this == obj);  
    }  
  
    public String toString() {  
        return getClass().getName() +  
            "@" + Integer.toHexString(hashCode());  
    }  
    ...  
}
```



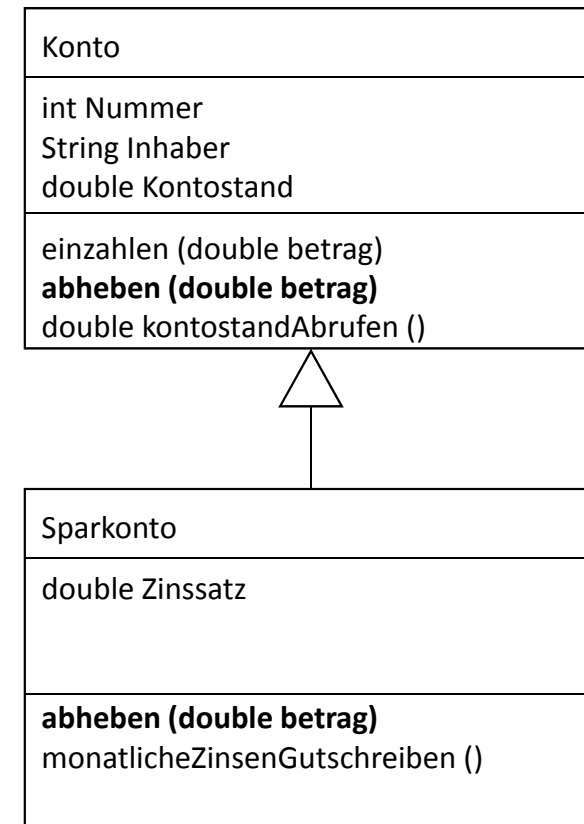
```
public class Konto {  
  
    private double kontostand;  
    private int nummer;  
    private String name;  
    ...  
}
```

Java Klassenhierarchie (Auszug)



Vererbung in Java

- Überschreiben von Methoden
 - Eine Unterklasse implementiert eine Methode der Oberklasse neu
 - Die Signaturen beider Methoden sind identisch



Beispiel Sparkonto

```
public class Sparkonto extends Konto {  
    static double zinssatz = 3.0;  
  
    public Sparkonto( int nummer, String name, double betrag ) {  
        // Aufruf Konstruktor der Oberklasse  
        super(nummer, name, betrag);  
    }  
  
    public void abheben( double betrag ) {  
        ...  
    }  
  
    void monatlicheZinsenGutschreiben() {  
        ...  
    }  
}
```

Erweiterung der Klasse Konto
durch Überschreiben einer Methode

Vererbung in Java

- Zugriff auf Elemente der Oberklasse mit **super**
 - Attribute
 - **super.oberklassenattribute**
 - Methode
 - super.oberklassenmethode()**
 - Konstruktor
 - super()**

Beispiel Sparkonto

```
public class Sparkonto extends Konto {  
    static double zinssatz = 3.0;  
  
    public Sparkonto( int nummer, String name, double betrag ) {  
        // Aufruf Konstruktor der Oberklasse  
        super(nummer, name, betrag);  
    }  
  
    public void abheben( double betrag ) {  
        ...  
    }  
  
    void monatlicheZinsenGutschreiben() {  
        ...  
    }  
}
```

Aufruf Konstruktor der Klasse Konto

Schlüsselwort super

- In allen Instanz-Methoden einer Klasse verfügbar
- Referenz zum aktuellen Objekt als eine Instanz seiner Oberklasse
 - Für Aufruf eines Konstruktors der Oberklasse
 - Zum Zugriff auf geerbte (überschriebene) Methoden/Attribute

Super in Konstruktoren

- Unterklassen müssen einen der Konstruktoren der Oberklasse aufrufen
 - Ausnahme: der leere Default-Konstruktor
- Muss immer die erste Anweisung in einem Konstruktor sein
- Aufruf nur von in der Oberklasse definierten Konstruktoren möglich

```
public Sparkonto( int nummer, String name, double betrag ) {  
    // Aufruf Konstruktor der Oberklasse  
    super(nummer, name, betrag);  
}
```

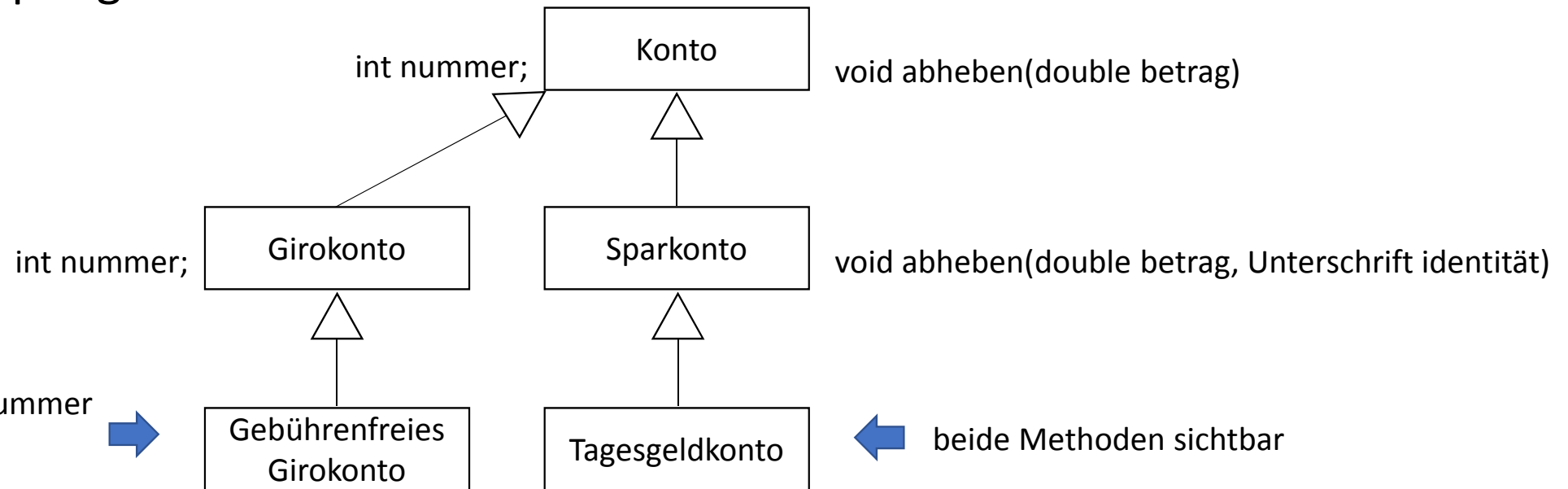
Super in Methoden

- Super kann an jeder beliebigen Stelle einer Methode eingefügt werden
- Methoden der Oberklasse können in jeder beliebigen Methode der Unterklasse aufgerufen werden
- Attribute der Oberklasse können in jeder beliebigen Methode der Unterklasse zugegriffen werden

```
public void abheben( double betrag ) {  
    // Gebühren berechnen und prüfen ob ausreichend Deckung existiert  
  
    // Aufruf der Methode der Oberklasse zum Abheben des gegebenen Betrags  
    super.abheben(betrag);  
}
```

Schlüsselwort super

- Super kann nicht verkettet werden
super.super.kontostand geht nicht
- Super greift auf das erste sichtbare Element in der Hierarchie zu



Girokonto.nummer
sichtbar



beide Methoden sichtbar

Typumwandlungen

```
// Instanz der Unterklasse in Variable vom Typ der Oberklasse speichern: ok
Konto konto1 = new Konto(nummer, "Fred", 0.0);
Konto konto2 = new Sparkonto(nummer, "Lisa", 20.0);

// Fehler!
Sparkonto konto3 = new Konto(nummer, "Mike", 25.0);

// Fehler, da Methode nicht in der Klasse Konto existiert
konto2.monatlicheZinsenGutschreiben();

// Typecast von Rechner -> LimitsRechner: ok
((Sparkonto)konto2).monatlicheZinsenGutschreiben();

// Typecast von Konto -> Sparkonto: schlägt zur Laufzeit fehl
((Sparkonto)konto1).monatlicheZinsenGutschreiben();
```

instanceof Operator

- Test ob ein Objekt vom Typ einer gegebenen Klasse ist

```
Konto konto2 = new Sparkonto(nummer, "Lisa", 20.0);  
if (konto2 instanceof Sparkonto)  
    System.out.println("Sparkonto gefunden");
```

- Wo sinnvoll?
 - Auffinden aller Sparkonten im Array kontos der Klasse Bank
 - Berechnung der monatlichen Zinsen
 - In Kombination mit Typumwandlungen

Vererbung in Java



- Keine Mehrfachvererbung
 - Eine Klasse kann immer nur genau von einer Oberklasse erben
 - Auf **extends** folgt immer nur genau ein Klassenname
- Keine Auswahl
 - Alle sichtbaren Attribute und Methoden der Oberklasse werden vererbt
- Klassen-Attribute werden nicht vererbt
- Konstruktoren werden nicht vererbt

Vererbung und Klassen/Instanz-Methoden

- Instanz-Methoden können keine Klassen-Methoden überschreiben
- Klassen-Methoden können keine Instanz-Methoden überschreiben

```
class Konto {  
    public void abheben( double betrag ) {  
        ...  
    }  
  
    static double getZinssatz() {  
        ...  
    }  
}
```

```
class Sparkonto extends Konto {  
    public static void abheben( double betrag ) {  
        ...  
    }  
  
    double getZinssatz() {  
        ...  
    }  
}
```



Zugriffsrechte in der Vererbungshierarchie


- **public:** Methode / Attribut ist für alle anderen Klassen sichtbar
- **protected:** Methode / Attribut ist in den Unterklassen sichtbar
- **<keine Angabe>:** Methode / Attribut ist im selben Paket sichtbar
- **private:** Methode / Attribut ist nur in der Klasse selbst sichtbar
 - Kann in Unterklassen nicht zugegriffen/überschrieben werden
- **final:** Methode / Attribut darf nicht in Unterklassen geändert werden

Richtlinie zur Verwendung der Zugriffsrechte

- Überschriebene Methoden in Unterklassen können die Berechtigungen nur „großzügiger“ machen
→ private → <keine Angabe> → protected → public

```
class Konto {  
    public void abheben( double betrag ) { }  
    void einzahlen( double betrag ) { }  
}
```

```
class Sparkonto extends Konto {  
    private void abheben( double betrag ) { }  
    public void einzahlen( double betrag ) { }  
}
```



Warum muss das so sein?

Schlüsselwort final

```
public final class Sparkonto {...}
```

→ von *Sparkonto* A kann keine Unterklasse abgeleitet werden

```
public class Tagesgeldkonto {  
    final void zinsenBerechnen() {...}  
}
```

→ in Unterklasse von Tagesgeldkonto darf die Methode *zinsenBerechnen()* nicht überschrieben werden

```
public class Tagesgeldkonto {  
    final static double zinssatz = 1.5;  
}
```

→ Wert der Variable *zinssatz* kann nicht verändert werden (Definition von Konstanten)

Polymorphismus

- Überschreiben von Methoden
 - Eine Unterklasse definiert eine Methode der Oberklasse
 - Beide Methoden haben dieselbe Signatur
- Überladen von Methoden
 - Derselbe Methodenname kann innerhalb einer Klasse mit verschiedenen Parametern wiederverwendet werden

Überladen von Methoden

- Überladene Methoden einer Klasse unterscheiden sich in ihren Eingabeparametern

- Anzahl
- Type
- Reihenfolge

```
void einzahlen(double betrag) {}  
  
void einzahlen(String zweck, double betrag) {}  
  
void einzahlen(double betrag, String zweck) {}  
  
void einzahlen(double betrag, String zweck) {}  
  
void einzahlen(double betrag, String zweck, String einzahler) {}
```



Wozu ist das gut?

Zusammenfassung

- Vererbung
 - Spezialisierungsbeziehungen durch Ableitung von Unterklassen aus Oberklassen
 - Überschreiben, hinzufügen und überladen von Methoden
 - Aufruf von Methoden der Oberklasse mit **super**
- Vorteile der Vererbung
 - Wiederverwendbarkeit
 - keine Quellcode-Duplizierung notwendig
 - Änderungen an Oberklasse automatisch in allen Unterklassen
- Schlüsselwort **final** knüpft die Vererbung an Bedingungen

Kontrollfragen

- Erläutern Sie das Konzept der Vererbung mit einer Klassenhierarchie.
- Wie kann in einer Unterklasse auf Methoden und Attribute der Oberklasse zugegriffen werden?
- Wann ist der Konstruktor der Oberklasse in einer Unterklasse aufzurufen?
- Was ist der Unterschied zwischen überladen und überschreiben von Methoden?
- Warum sollten Sie nur Konstanten aber nie Attribute mittels public öffentlich sichtbar machen, sondern lieber über setter/getter Methoden zugegriffen werden?