LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Grundlagen der OOP

Bioinformatik Bachelor

Propädeutikum WS 2019/2020

03. Dezember 2019

Christian Hoffmann

Outline

- Generalisierung / Spezialisierung
- Vererbung
- Polymorphismus

Generalisierung / Spezialisierung

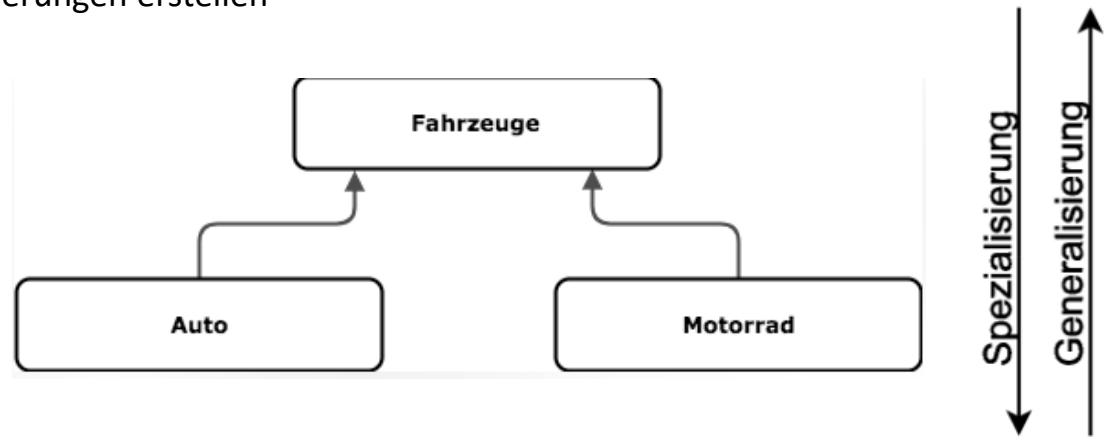
Def: Eines der Kernkonzepte der objektorientierten Programmierung. Hier werden generalisierte Klassen und Objekte genutzt, um gemeinsame Attribute und Methoden in logische Einheiten zu bündeln.

Generalisierung:

- Vom Besonderen zum Allgemeinen
- Gemeinsamkeiten in genereller Klasse

Spezialisierung

- Vom Allgemeinen zum Besonderen
- Spezialisierungen erstellen



Vererbung

Def: Vererbung -> „Kind-Objekt“ übernimmt alle Eigenschaften und Methoden eines „Eltern-Objekts“.

Idee: Neue Klasse auf Basis von bereits bestehender Klasse erstellen & erweitern.

Begriffe:

- **subclass** (child) – Klasse, welche von einer anderen erbt. Wird auch derived class, extended class, oder child class genannt
- **superclass** (parent) – Klasse, von der eine Subklasse Eigenschaften erbt. Wird auch base class oder parent class genannt

Syntax

```
class Subclass-name extends Superclass-name
{
  // Attribute und Methoden
}
```

Erklärung:

extends keyword: Klasse erbt von einer bereits existierenden Klasse. Objekte dieser Klasse haben Zugriff Attribute und Methoden der Superklasse.


```java
class Tier  {
  int alter; // Attribut
  public Tier(int alter){ // Konstruktor
    this.alter = alter;
  }

  public void essen(){
    System.out.println("eat");
  }
}
```

```java
class Vogel extends Tier {
  public Vogel(int alter){
    super(alter); // superclass
  }

  public void fliegen(){
    System.out.println("fly");
  }
}
```

```java
class Fisch extends Tier {
   public Fisch(int alter){
     super(alter);
   }

   public void tauchen(){
    System.out.println("dive");
  }
}
```

Beispiel

```
public static void main(String[] args) {
    Vogel papagei = new Vogel(3);
    Fisch nemo = new Fisch(1);
    Tier tier = new Tier(2)

    papagei.essen();
    nemo.essen();
    tier.essen();

    nemo.tauchen();
    papagei.fliegen();
}
```

```
eat
eat
eat
dive
fly
```

“final” Keyword

„final“: Verhindert Vererbung

```
final class Tier  {
  // Attribute und Methoden
}
```

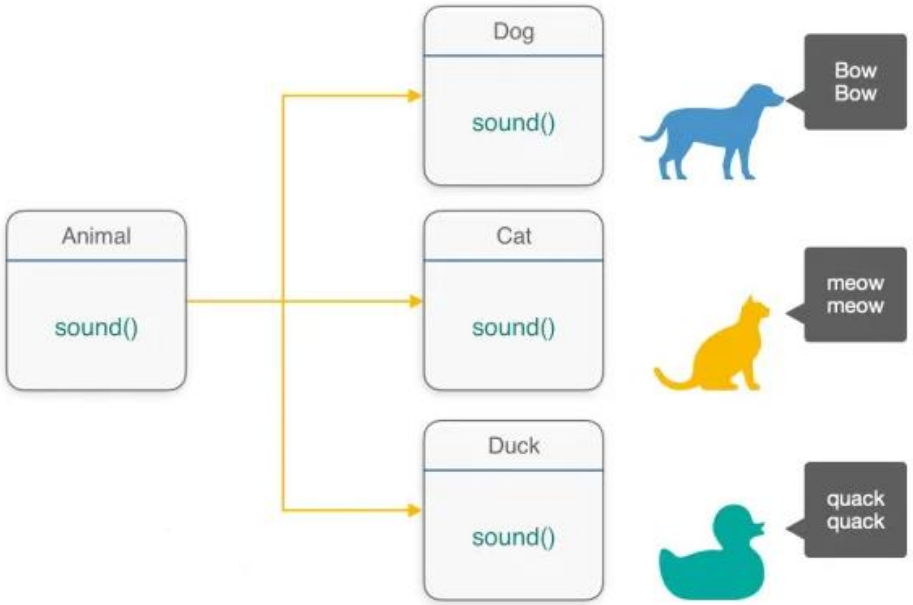
```
class Affe extends Tier
{
  // Attribute und Methoden
}
```

```
error: cannot inherit from final Tier
```

Polymorphismus

Def: heißt „viele Formen“, tritt auf wenn wir viele Klassen haben, die miteinander durch Vererbung verbunden sind.

Idee: Vererbung lässt Eigenschaften einer anderen Klasse erben. Polymorphismus nutzt die geerbten Methode und spezifiziert sie, um verschiedene Aufgaben zu erledigen.



```java
class Tier  {
   .
   .
   .
   public void lautGeben(){
     System.out.println("tierlaut");
   }
}
```

```java
class Vogel extends Tier {
   .
   .
   .
   @Override
   public void lautGeben(){
     System.out.println("zwitscher");
   }
}
```

```java
class Fisch extends Tier {
   .
   .
   .
   @Override
   public void lautGeben(){
     super.lautGeben();
     System.out.println("blub");
   }
}
```

Wichtig: `@Override` Keyword: Zeigt der Klasse, dass wir eine geerbte Methode überschreiben / spezialisieren!

Beispiel

```
public static void main(String[] args) {
    Vogel papagei = new Vogel(3);
    Fisch nemo = new Fisch(1);
    Tier tier = new Tier(2)

    tier.lautGeben();
    papagei.lautGeben();
    nemo.lautGeben();

}
```

```
tierlaut
zwitscher
tierlaut
blub
```

Override toString

```
class Complex  {
  private double re, im;

  public Complex(double re, double im ){
    this.re = re;
    this.im = im;
  }
}
```

```
class Main  {
  public static void main(String[] args){
    Complex c1 = new Complex(10, 15);
    System.out.println(c1);
  }
}
```

```
Complex@19821f    (interne Objektid)
```

```java
class Complex  {
  private double re, im;

  public Complex(double re, double im ){
    this.re = re;
    this.im = im;
  }

  @Override
  public String toString(){
     return String.format(re + " i" + im);
  }
}
```

```java
class Main  {
  public static void main(String[] args){
    Complex c1 = new Complex(10, 15);
    System.out.println(c1);
  }
}
```

```
10.0 + i15.0
```

Infos fürs Blatt:

- „obj instanceof class“ gibt einen boolean zurück, der zeigt ob ein gegebenes Objekt einer Klasse angehört
- Override equals Methoden: https://www.geeksforgeeks.org/overriding-equals-method-in-java/
- System.out.println(obj) gibt die toString-Methode des Objektes aus (wichtig: Bei overrides keine prints einfügen!)