



Vererbung, Tipps & Fragen

Sebastian Lehnerer

Info-Tutorium der Fachschaft
Bioinformatik

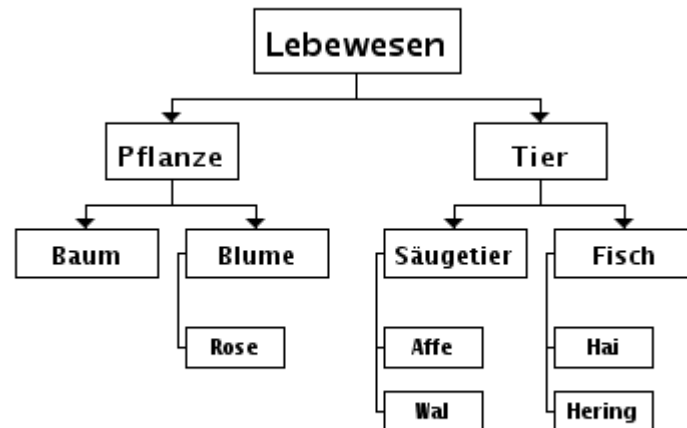
18.01.2011

Wiederholung

- Java ist eine Objektorientierte Programmiersprache (OOP)
 - Abstraktion
 - Kapselung
 - **Vererbung**
 - Polymorphie

Vererbung

- Objekte haben oft gleiche Eigenschaften, manchmal sogar gemeinsame



Vererbung

- **Subklassen** erben von **Superklassen**
- Eigenschaften und Fähigkeiten werden vererbt – in informatisch:
- Methoden und Attribute (methods, fields)

Vererbung

- Methoden und Attribute müssen *sichtbar* sein
- Getter und Setter Methoden

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Vererbung

- Methoden können überschrieben werden

```
public class Ape extends Mammal {  
    public static void speak() {  
        System.out.println("UHUHUH");  
    }  
}
```

```
public class Mammal extends Animal {  
    public static void speak() {  
        System.out.println(„not discovered speech yet“);  
    }  
}
```

Ape.speak() ??

Vererbung

- Methoden können überschrieben werden

```
public class Ape extends Mammal {  
    String myWords= "UHHUHUH";  
}
```

```
public class Mammal extends Animal {  
    String myWords= "not discovered speech yet";  
    public static void speak() {  
        System.out.println(myWords);  
    }  
}
```

Ape.speak() ??

Abstrakte Klassen und Methoden


- Klassen und Methoden können als abstrakt markiert werden
- Abstrakte Methoden haben keinen Rumpf – sie bestehen nur aus der **Signatur**
- Abstrakte Methoden müssen von der Subklasse überschrieben werden
- Abstrakte Klassen können nicht instanziiert werden

Abstrakte Klassen und Methoden

```
public abstract class Mammal extends Animal {  
  
    public abstract void speak();  
  
}
```

Beispiel: Katze

```
1 public class Animal {
2     String habitat;
3     int age;
4     Animal prey;
5
6     public Animal (String habitat, int age, Animal prey) {
7         this.habitat = habitat;
8         this.age = age;
9         this.prey = prey;
10    }
11 }
12
13 public class Cat extends Animal {
14     String furColour;
15     int micePerDay;
16
17     public Cat(String habitat, int age, Animal prey, String fur, int mpd) {
18         super(habitat, age, prey);
19         this.furColour = fur;
20         this.micePerDay = mpd;
21     }
22 }
```



Übung: Basisklasse Person

- Person hat einen Namen
- Instanzvariablen sollen private sein
- Implementiere getter und setter Methoden
- Implementiere eine einfache Methode print, die Person auf der Konsole ausgibt
- Implementiere eine Methode **isEqual**, die Person mit einer anderen Person vergleicht und einen boolean zurück gibt
- Implementiere eine abstrakte Methode **iAmA()**
- Initialisiere ein Person Object in der main Methode und benutze die implementierten Methode

Übung: Erweiterung Student

- Ein Student ist eine Person und erbt damit Attribute und Methoden von Person
- Student hat eine Matrikelnummer
- Implementiere nötige getter und setter Methoden
- Überlagere die Methode **isEqual**s, die dann Student mit einem anderen Student vergleicht
- Implementiere eine Methode **iAmA()** die zurück gibt das die Person ein Student ist
- Implementiere eine main Methode und benutze sowohl geerbte als auch neue Methoden von Student

Übung: Erweiterung Professor

- Ein Professor ist eine Person und erbt damit Attribute und Methoden von Person
- Professor hat eine *Mitarbeiternummer*
- Implementiere nötige getter und setter Methoden
- Überlagere die Methode **isEqual**s, die dann Professor mit einem anderen Professor vergleicht
- Implementiere eine Methode **iAmA()** die zurück gibt das die Person ein Professor ist
- Implementiere eine main Methode und benutze sowohl geerbte als auch neue Methoden von Professor

Generische Klassen

```
public class GenericItem <T> {  
    public T value ;  
    public GenericItem <T> previous ;  
    public GenericItem <T> next ;  
  
    public GenericItem (T item) {  
        value = item;  
    }  
}
```

Generische Klassen

```
public class GenericSingleLinkedList {  
  
    private GenericItem head = null ;  
  
    public void add ( String v) {  
        GenericItem item = new GenericItem (v);  
        item . next = head ;  
        head = item ;  
    }  
}
```

Generische Klassen

```
public class GenericSingleLinkedList <T> {  
  
    private GenericItem<T> head = null ;  
  
    public void add ( T v) {  
        GenericItem<T> item = new GenericItem<T> (v);  
        item . next = head ;  
        head = item ;  
    }  
}
```

Generische Klassen

```
public static void main ( String [] args ) {  
    GenericSingleLinkedList<String> list  
        = new GenericSingleLinkedList<String> ();  
    list .add ("String 1");  
    list .add ("String 2");  
    list .add ("String 3");  
}
```

Generische Klassen

```
public static void main ( String [] args ) {  
    GenericSingleLinkedList<Integer> list  
        = new GenericSingleLinkedList<Integer> ();  
    list .add (1);  
    list .add (2);  
    list .add (3);  
}
```