Prof. Dr. Verena Majuntke - Prof. Dr. Ulrich Meissen

Grundlagen der Programmierung

Objektorientierung – Teil II

14.12.2020

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Themen

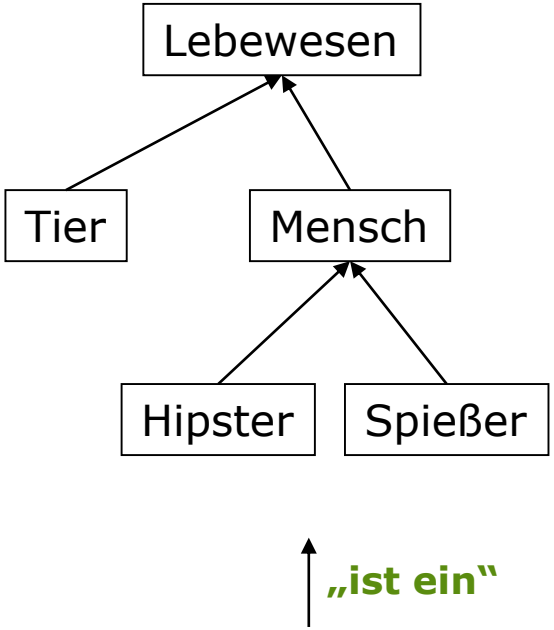
- Vererbung
- Schlüsselwort *static*
- Klasseneigenschaften
- Objekteigenschaften

Lernziele

- Was bedeutet Vererbung in der Programmierung?
- Was ist eine Oberklasse/Unterklasse?
- Was haben Ober- und Unterklassen gemeinsam und was unterscheidet Sie?
- Bedeutung und Verwendung des Schlüsselworts static?
- Unterschied zwischen Klasseneigenschaften und Objekteigenschaften?
- Wie rufe ich Klassenvariablen/-funktionen im Gegensatz zu Objektvariablen/Methoden auf?

Vererbung – Reale Welt

- Elemente kommen in verschiedenen Varianten vor, die sich hierarchisch klassifizieren lassen
- Elemente, die hierarchisch tiefer stehen, sind speziellere Varianten der übergeordneten, generelleren Elemente
- Speziellere Elemente besitzen die Eigenschaften der generelleren Elemente plus weitere, spezifischere Eigenschaften



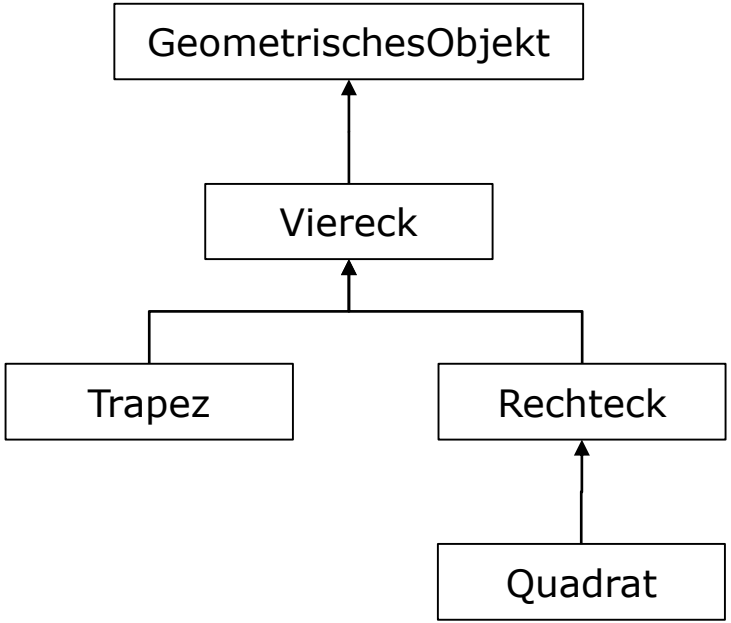
Vererbung – Programmierung

- Unterklassen übernehmen (erben) Eigenschaften (Attribute) und Operationen der Oberklasse
- Unterklassen können zusätzlich weitere Eigenschaften (Attribute) und Operationen enthalten
- Unterklassen können andere Initialwerte für Attribute haben
- Die Funktionalität von Operationen in Ober- und Unterklasse kann sich unterscheiden

Vererbung

- Die Vererbung ist eine elementare Möglichkeit der Wiederverwendbarkeit von Klassen
- Vererbung basiert auf der Idee, dass „Elternklassen“ ihren „Kindern“ Variablen und Methoden vererben
- Durch Vererbung entsteht eine Klassenhierarchie

Beispiel



Eine Klasse *Viereck*

```
public class Viereck
{
      //Seiten
      int a,b,c,d;

      int umfang()
          {
              return a+b+c+d;
          }
}
```

Anwendung der Klasse *Viereck*

```
public class TestKlasse
{

     public static void main(String[] args)
     {
          Viereck v1 = new Viereck();
          v1.a = 10;
          v1.b = 20;
          v1.c = 30;
          v1.d = 40;
          System.out.println("Umfang des Vierecks: " + v1.umfang());
     }
}
```

Das Schlüsselwort *extends*

```
public class Rechteck extends Viereck
{

}
```

- Rechteck erbt alle Eigenschaften von Viereck (die Objekt-variablen und die Objektmethoden)
- jedes Rechteck ist ein Viereck („is-a"-Relation) – Spezialisierung

Begriffe

Oberklasse (Basisklasse, Elternklasse, Superklasse):

Klasse, die Variablen und Methoden an andere Klassen vererbt

Unterklasse (Subklasse, Kindklasse):

Klasse, die von einer Oberklasse erbt (von dieser abgeleitet wird)

Beispiel:

- die Klasse Rechteck wird aus der Klasse Viereck abgeleitet
- Syntax:

```
class Unterklasse extends Oberklasse
{

}
```

```
class Rechteck extends Viereck
{

}
```

Anwendung der Klasse *Rechteck*

```
public class TestKlasse
{

     public static void main(String[] args)
     {
          Rechteck r1 = new Rechteck();
          r1.a = 15;
          r1.b = 35;
          r1.c = 15;
          r1.d = 35;
          System.out.println("Umfang des Rechtecks: " + r1.umfang());
     }
}
```

Erweiterung der Klasse *Rechteck*

```
public class Rechteck extends Viereck
{
        int flaecheninhalt()
        {
                return a * b;
        }
}
```

- Rechteck besitzt nun auch die Operation *flaecheninhalt()*
- Viereck besitzt diese Operation nicht!

Anwendung der Klassen

```
public class TestKlasse
{

      public static void main(String[] args)
      {
            Viereck v1 = new Viereck();
            Rechteck r1 = new Rechteck();
            v1.a = 10; v1.b = 20; v1.c = 30; v1.d = 40;
            r1.a = 15; r1.b = 35; r1.c = 15; r1.d = 35;
            System.out.println("Umfang des Vierecks: " + v1.umfang());
            System.out.println("Umfang des Rechtecks: " + r1.umfang());
            System.out.println(" Flächeninhalt des Vierecks: " +
                                                v1.flaecheninhalt());// Fehler
            System.out.println(" Flächeninhalt des Rechtecks: " +
                                                r1.flaecheninhalt());

            }
}
```

Das Schlüsselwort *final*

Wird eine Variable oder eine Methode als *final* deklariert, so kann diese nicht mehr überschrieben werden

final Variable: Der Variablenwert ist nicht mehr veränderbar

```
final double PI = 3.14;

final int MWST = 19;
```

final Methode: Methode kann in abgeleiteten Klassen nicht überschrieben werden

final Klasse: Die Klasse kann keine Kindklasse/Subklasse haben

Methoden überschreiben

Idee des Überschreibens:

verschiedene Funktionalität durch gleichnamige Methoden in Ober- und Unterklasse bereitstellen

Beispiel: Rückgabe des Flächeninhalts (Rechteck/Quadrat)

Bedingung: gleicher Methodenkopf (Signatur inkl. Rückgabetyp)

Methoden überladen

Idee des Überladens:

gleiche Funktionalität durch gleichnamige Methoden mit verschiedenen Parameterlisten (verschiedenen Signaturen) bereitstellen

(Kennen wir schon von Konstruktoren!)

Beispiel: Addieren von zwei oder mehr Zahlen vom Typ int

Bedingung: unterschiedliche Methoden-Signatur

```
public class BeispielUeberladen {

	public static int addiere(int s1, int s2){
		return s1+s2;
	}

	public static int addiere(int s1, int s2, int s3){
		return s1+s2+s3;
	}

	public static double addiere(double s1, double s2){
		return s1+s2;
	}


	public static void main(String[] args) {
		System.out.println(addiere(2,3));
		System.out.println(addiere(2,3,4));
		System.out.println(addiere(12.0,13.0));
	}
}
```


Signatur von Methoden (I)

Zur *Signatur* einer Methode gehören:

- Name der Methode
- Anzahl, Typ und Typreihenfolge der Parameter.

Zur *Signatur* einer Methode *gehören nicht*:

- Rückgabetyp,
- Bezeichner der Parameter,
- Modifizierer.

Signatur von Methoden (II)

Die Signatur der Methoden einer Klasse **muss eindeutig** sein!

Eindeutig: die Beispiele aus vorheriger Folie

Nicht eindeutig:

```
public double addiere(int s1, int s2){}
public int addiere(int s1, int s2){}
public int addiere(int summand1, int summand2){}
```

Das Schlüsselwort *static*

Recap: Klasse und Objekt

- Klasse – benutzerdefinierter abstrakter Datentyp
- Objekt – konkrete Instanz einer Klasse
- Objekt-/Instanzvariable – Daten des Objektes = Attribut
- Methode – Funktion, die an Objekt gebunden ist

Wie werden Daten und Funktionen von Klassen abgebildet?

- Klassenvariable – (globale) Daten der Klasse, gehören nicht den Instanzen an
- Klassenfunktion – (globale) Funktion der Klasse, keine Bindung an Instanzen

Das Schlüsselwort `static` (I)

- Zugriff auf Objektvariablen und –methoden nur über ein Objekt
  - `referenzvariable.objektvariable`
  - `referenzvariable.objektmethode`

- In einigen Fällen möchte man kein Objekt erzeugen, um Variablen und Methoden zu nutzen
  - `main()` – Methode einer Klasse
  - `cos()`, `sin()`, `tan()` sind Methoden der Klasse `Math`
  - Konstante `PI` der Klasse `Math`

Das Schlüsselwort `static` (II)

- Um auf Methoden und Variablen einer Klasse zugreifen zu können, ohne vorher von dieser Klasse Objekte zu erzeugen, gibt es das Schlüsselwort `static`

- Sind Variablen `static` deklariert, heißen diese Variablen Klassenvariablen
- Sind Methoden als `static` deklariert, heißen diese Methoden Klassenmethoden

Klasseneigenschaften

`static` kennzeichnet Klassen**eigenschaften**

- Methoden (Klassenmethoden):

```
public static void methode(){ ... }
```

- Variablen (Klassenvariablen):

```
private static int counter;
```

Objekteigenschaften

Ohne `static` kennzeichnet Objekteigenschaften

- Methoden (Objektmethoden):

```
public static void methode(){ ... }
```

- Variablen (Objektvariablen):

```
private static int counter;
```


Unterschied Objektvariable/Klassenvariable

```
public class Konto
{
     double          guthaben;
     int             kontonr;

     static int      anzahlKonten=0;

     Konto()
     {
          guthaben=0.0;
          anzahlKonten++;
          kontonr = anzahlKonten;
     }
}
```

Objektvariablen

Klassenvariable

Zugriff auf Klassenvariablen

```
public class MeineProgrammKlasse
{
    public static void main(String[] args)
    {
        Konto ko1 = new Konto();
        Konto ko2 = new Konto();
        Konto ko3 = new Konto();
        int anzahl = Konto.anzahlKonten
        System.out.printf("Anzahl der Konten : %d %n",anzahl);
    }
}
```

Klasseneigenschaften vs. Objekteigenschaften

Klasse

- Werden mit `static` gekennzeichnet

```
static int variablenname;
```

- Werden über den Klassennamen angesprochen

```
Klasse.variablenname;
```

- Werden von allen Instanzen/Objekten gemeinsam genutzt

Objekt

- Werden NICHT mit `static` gekennzeichnet

```
int attribut;
```

- Werden über den Objektnamen (= Referenzvariable) angesprochen

```
Objekt.attribut;
```

- Werden nur von der Instanz/Objekt genutzt

Lernziele

- Was bedeutet Vererbung in der Programmierung?
- Was ist eine Oberklasse/Unterklasse?
- Was haben Ober- und Unterklassen gemeinsam und was unterscheidet Sie?
- Bedeutung und Verwendung des Schlüsselworts static?
- Unterschied zwischen Klasseneigenschaften und Objekteigenschaften?
- Wie rufe ich Klassenvariablen/-funktionen im Gegensatz zu Objektvariablen/Methoden auf?

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de