Programmierung – Nachklausurtutorium

Laryssa Horn, Tim Engelhardt

20 März 2018

Klassen

Wofür wir Klassen brauchen:

- Definieren ein „Bauplan“ eines Objektes
- Bauplan enthält Attribute und Methoden

Klasse Beispiel

```
public class Koordinate{

   private int x;
   private int y;

   public Koordinate(int pX, int pY){
       x = pX;
       y = pY;
   }

   public int getX(){
       return x;
   }
}
```

Attribute

Konstruktor

Methoden

Konstruktor

Mithilfe eines Konstruktors kann man eine Klasse instanziieren (Objekt einer Klasse erstellen)

Bsp. Erstellung eines neuen Koordinaten-Objektes:

```
Koordinate punkt = new Koordinate(4, 7);
```

(*Konstruktoraufruf*)

- Sollte kein Konstruktor definiert sein, existiert immer ein Default-Konstruktor, der keine Parameter erwartet.

Zugriff der Variablen

```
public class Fahrrad{

   private int zoll;

   private static final int RAEDER = 2;

   public Fahrrad(int zoll){
      this.zoll = zoll;
   }

   public int zaehleSpeichen(){
      int anzahlSpeichen = 0;
      …
   }
}
```

Instanzvariable

Klassenvariable (konstante)

lokale Variable

Weitere Definitionen

static

- Methoden: können ohne Objekt aufgerufen werden
- Variablen: Klassenvariable, alle Objekte dieser Klasse benutzen die „gleiche“ Variable

this

- Liefert das aktuell referenzierte Objekt der Klasse

Live Coding

Koordinaten

(weitere Funktionalitäten implementieren,
Beispiel Objekte Funktionen ausführen lassen)

Vererbung

Vererbung wird in Java verwendet, um eine bereits vorhanden Klasse zu erweitern bzw. zu spezialisieren

- es existiert immer eine Ober- und Unterklasse
- Die Unterklasse erbt alle Methoden und Attribute der Oberklasse (außer private)
- Unterklasse kann zudem um weitere Attribute/ Methoden erweitert werden (Spezialisierung)

Vererbung weitere Informationen

- Konstruktor und Funktionen der Oberklasse können mit super() aufgerufen werden
- Methoden können in der Unterklasse überschrieben werden
- Eine Klasse kann maximal von einer Oberklasse erben

Vererbung

```
public class Hund extends Tier{
…
}
```

Hier erbt Hund von Tier mit Hilfe von „extends“

Live Coding

Vererbungsstrukturen

Abstrakte Klassen Definition

- spezielle Klasse, die sich nicht instanziieren lässt
- Dient lediglich als „Strukturelement“ in der Vererbung
- Beinhaltet nur Methodensignaturen (keine vollständigen Methoden)
- Alle abstrakten Funktionen müssen(!) in der geerbten Klasse Implementiert werden

Abstrakte Klassen Definition

- Sinnvoll wenn man sicherstellen möchte, dass eine Unterklasse gewisse Methoden implementiert
- Sobald eine Methode abstrakt ist, muss die gesamte Klasse abstrakt sein

Beispiel

```
public abstract class Tier{

  protected int anzahlBeine;

  public abstract int beschaffeNahrung();
}
```

Live Coding

Beispiel Tier

Polymorphie Definition

Polymorphie bedeutet wortwörtlich „Vielgestaltigkeit“ und findet Anwendung in der Vererbung von Klassen

Beispiel Polymorphie

Beispiel

Live Coding

Interface

- Ähnlich wie in der Vererbung, bestimmt man einen „Bauplan“ für eine Klasse
- Im Gegensatz zur Vererbung kann eine Klasse jedoch mehrere Interfaces implementieren
- Ein Interface besitzt nur Methodensignaturen und Konstanten als Variablen
- Ein Interface darf niemals private sein

Interface

- Wie bei abstrakten Klassen müssen auch hier alle Funktionen des Interfaces in den Klassen welche das Interface benutzen implementiert werden

Beispiel Interface

```
public interface BspInterface {
   // Beispiel Variablen
   variablenTyp konstante1 = wert1;
   variablenTyp konstante2 = wert2;
   ...

   // Beispiel Methoden
   returnTyp methodenName1(ParameterListe1);
   void methodenName2(ParameterListe2);
   ...
 }
```

(Quelle: http://www.javakurs-online.de/javakurs08.html)

Beispiel Interface Implementierung

```
public class Beispiel implements BspInterface{
   …
}
```

- Beispiel für die Implementierung eines Interfaces
- In der Klasse Beispiel müssen nun alle Methoden von BspInterface implementiert werden

Live Coding

Interface