

Programmieren I

Dr. Klaus Höppner

Hochschule Darmstadt – Wintersemester 2009/2010

1/22

Design von Klassen

Beispiel: Bibliothek

2/22

Strategie zum Entwurf von Klassen

- *Objektorientierte Sichtweise:*
Mit welchen Objekten habe ich es in einem Programm zu tun?
- Suche nach gemeinsamen *Eigenschaften* und *Methoden*
- Davon ausgehend Entwurf der Klassen mit ihren Attributen und Methoden
- *Empfehlenswert:* Schematische Übersicht

3/22

Beispiele

- Kontenverwaltung
- Klasse: **Konto**
- Attribute:

Attribut	Typ
Inhaber	String
Adresse	String
Kontostand	double
Kreditrahmen	double
...	

4/22

- Lagerverwaltung
- Klasse: **Artikel**
- Attribute:

Attribut	Typ
Beschreibung	String
EAN (Barcode)	String
Bestand	int
Einkaufspreis	double
Verkaufspreis	double
Lagerort	String
...	

5/22

- Kundenverwaltung
- Klasse: **Kunde**
- Attribute:

Attribut	Typ
Name	String
Adresse	String
Zahlungsart	enum
Kredit	double
Rabatt	double
Kundenklasse	enum
...	

6/22

- Bibliothek
- Klasse: **Buch**
- Attribute:

Attribut	Typ
Autor	String
Titel	String
ISBN	String
Verlag	String
Erscheinungsjahr	int
Auflage	int
ausgeliehen	boolean
Ausleiher	String
...	

7/22

Grundgerüst der Klasse

- Deklaration der Klasse erfolgt in der Datei „*KlassenName.java*“.
- Grundgerüst:


```
public class Klassenname {
    ...
}
```
- *Konvention*: Namen von Klassen beginnen immer mit Großbuchstaben (wortweise, z. B. *MeineKlasse*).

8/22

Hinzufügen der Attribute

- Anschließend werden die Attribute deklariert.
- *Faustregel*: Attribute sind in der Regel *privat*:


```
public class Klassenname {
    private typ attribut;
    ...
}
```
- *Konvention*: Attribute und Methoden beginnen mit Kleinbuchstaben.
- *Ausnahme*: öffentliche konstante Attribute (**public final static**) komplett in Großbuchstaben.

9/22

Der Konstruktor

- Konstruktoren heißen genauso wie die Klasse und haben keinen Rückgabewert.
- Wird *kein* Konstruktor definiert, existiert implizit ein Standardkonstruktor:

```
class Klassenname {  
    public Klassenname() {  
    }  
}
```

- In der Regel wird man eigene Konstruktoren definieren.
- Falls benötigt, muss dann auch der Standardkonstruktor (d. h. ohne Parameter) definiert werden.
- Konstruktoren sind in aller Regel **public**.

10/22

Zugriffsmethoden (getter/setter)

- Ein Teil der Methoden ist notwendig, um auf *private* Attribute *lesend* (getter) oder *schreibend* (setter) zuzugreifen.
- Getter und Setter tragen die Namen `getAttributeName` bzw. `setAttributeName`.
- *Ausnahme* für Attribute vom Typ **boolean**: Getter heißt `isAttributname`.
- *Grundüberlegung*: Auf welche Attribute ist ein direkter Lese- oder Schreibzugriff nötig?
- *Faustregel*: Zugriffsmethoden werden in der Regel als **public** definiert.

11/22

Weitere Methoden

- Die *problemspezifische* Funktionalität steckt in weiteren Methoden.
- *Grundfrage*: Welche Fähigkeiten haben alle Objekte des gleichen Typs?
- Die Methoden können sowohl **private** als auch **public** sein.
- Die öffentlichen Methoden repräsentieren nach außen die *Schnittstellen* für die Kommunikation mit den Objekten.
- Neben dem Grunddesign der Klasse liegt hier die *eigentliche Kreativität* beim Programmdesign.

12/22

Beispiel

- Klasse **Konto**
- Methoden

Methode

einzahlung
 auszahlung
 ueberweisung
 lastschrift

13/22

- Klasse **GeldAutomat**
- Methoden

Methode

Art

Geld ausgeben	privat
Kontostand ändern	privat

Auszahlung	öffentlich
------------	------------

benutzt intern die Methoden
 »Geld ausgeben« und
 »Kontostand ändern«

14/22

Klasse Buch

Buch
-String autor
-String titel
-String ausleiher
+String getAutor()
+String getTitel()
+boolean isAusgeliehen()
+String getAusleiher()
+boolean ausleihe(string)
+void rueckgabe()

Aus Gründen der Übersichtlichkeit fehlen Attribute wie isbn, verlag, ...

15/22

Code der Klasse Buch

```
public class Buch {
    private String autor;
    private String titel;
    private String ausleiher;

    public Buch(String autor, String titel) {
        this.autor = autor;
        this.titel = titel;
        this.ausleiher = null;
    }
    public String getAusleiher() {
        return ausleiher;
    }
    public String getAutor() {
        return autor;
    }
    public String getTitel() {
        return titel;
    }
}
```

16/22

Code der Klasse Buch (Forts.)

```
public boolean isAusgeliehen() {
    return ausleiher != null;
}
public void ausleihe(String ausleiher) {
    this.ausleiher = ausleiher;
}
public void rueckgabe() {
    ausleiher = null;
}
public String toString() {
    String result = this.autor+": "+titel;
    if (isAusgeliehen()) {
        result+=" (ausgeliehen an "+ausleiher+");"
    } else {
        result+=" (nicht ausgeliehen)";
    }
    return result;
}
}
```

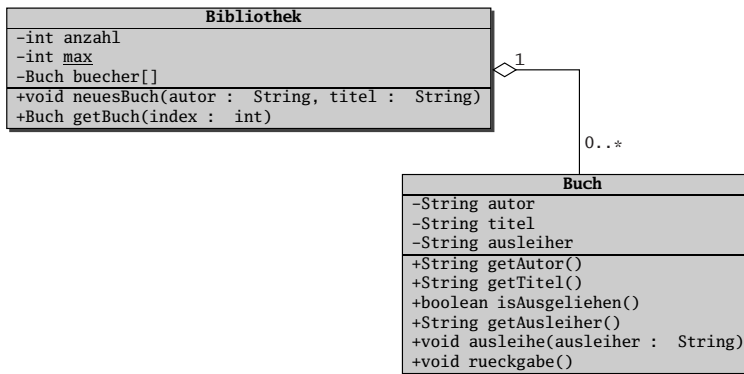
17/22

Erweiterung des Beispiels

- Für die Bibliothek soll eine gleichnamige Klasse **Bibliothek** geschaffen werden.
- Diese soll in einem Feld die in der Bibliothek vorhandenen Bücher speichern (also die Instanzen der Klasse Buch).
- Über den Index des Buches soll man auf die einzelnen Instanzen der Klasse Buch zugreifen können.
- Eine solche Klasse wird häufig als *Containerklasse* bezeichnet.

18/22

Klassendiagramm



19/22

Code der Klasse Bibliothek

```

public class Bibliothek {
    private int anzahl;
    private Buch[] buecher;
    private static final int max = 100;

    public Bibliothek() {
        anzahl = 0;
        buecher = new Buch[max];
    }
  
```

20/22

Code der Klasse Bibliothek (Forts.)

```

    public void neuesBuch(String autor, String titel) {
        if (anzahl < max) {
            Buch b = new Buch(autor, titel);
            buecher[anzahl++] = b;
        }
    }

    public Buch getBuch(int index) {
        if (index >= 0 && index < anzahl) {
            return buecher[index];
        } else {
            return null;
        }
    }
  
```

21/22

Code der Klasse Bibliothek (Forts.)

In diesem Beispiel enthält die Klasse `Bibliothek` auch die `main`-Methode, das demonstriert, wie ein Buch in die Bibliothek aufgenommen wird und ausgeliehen bzw. zurück gegeben werden kann.

```
public static void main(String[] args) {
    Bibliothek bib = new Bibliothek();

    bib.neuesBuch("Dan Brown", "Sakrileg");
    bib.getBuch(0).ausleihe("Thomas Schmidt");
    System.out.println(bib.getBuch(0));
    bib.getBuch(0).rueckgabe();
    System.out.println(bib.getBuch(0));
}
}
```