**Ministerium für
Schule und Weiterbildung
des Landes Nordrhein-Westfalen**

Handreichung zu den Bildungsplänen zur Erprobung Teil III

für die Bildungsgänge, die zu einem Berufsabschluss nach Landesrecht und zur allgemeinen Hochschulreife oder zu beruflichen Kenntnissen und zur allgemeinen Hochschulreife führen

Wirtschaftsinformatik

Fachbereich Wirtschaft und Verwaltung

Grundkurs

**Ministerium für
Schule und Weiterbildung
des Landes Nordrhein-Westfalen**

Herausgegeben vom Ministerium für Schule und Weiterbildung
des Landes Nordrhein-Westfalen
Völklinger Straße 49, 40221 Düsseldorf
08/2010

Inhalt Seite

1 Einführung

Die Vorgaben für das Fach Wirtschaftsinformatik gelten für folgende Bildungsgänge:

Kaufmännische Assistentin (Akzentuierung Betriebsorganisation)/AHR Kaufmännischer Assistent (Akzentuierung Betriebsorganisation)/AHR	APO-BK, Anlage D 12
Technische Assistentin für Betriebsinformatik/AHR Technischer Assistent für Betriebsinformatik/AHR[1]	APO-BK, Anlage D 13
Allgemeine Hochschulreife (Betriebswirtschaftslehre mit Rechnungswesen und Controlling)	APO-BK, Anlage D 27
Allgemeine Hochschulreife (Fremdsprachenkorrespondentin/Fremdsprachenkorrespondent) (Betriebswirtschaftslehre mit Rechnungswesen und Controlling, Sprachen)	APO-BK, Anlage D 28

Diese Bildungsgänge sind im Fachbereich Wirtschaft und Verwaltung dem fachlichen Schwerpunkt Wirtschaftswissenschaften zugeordnet.

Die vorliegende Handreichung dient in erster Linie dem besseren Verständnis des Fachlehrplans Wirtschaftsinformatik.

Anhand der folgenden Beispielaufgaben soll der Detaillierungsgrad exemplarisch dargestellt werden. In den Beispielaufgaben werden i. d. R. die in Kapitel 3 dieser Handreichungen definierten Operatoren verwandt, um deren Anwendung aufzuzeigen.

Für die Bearbeitung und mögliche Lösung der Beispielaufgaben stehen gezippte Projektordner zum Download zur Verfügung[2].

Weitere geeignete Beispielaufgaben finden sich auch in den **E**inheitlichen **P**rüfungsanforderungen in der **A**biturprüfung Berufliche Informatik (Beschluss der Kultusministerkonferenz vom 01.06.1979 i. d. F. vom 10.05.2007)[3].

2 Beispielaufgaben

2.1 Jahrgangsstufe 11.1

Hier bieten sich zusammenhängende Aufgaben an, wie z. B. Lohn- und Gehaltsabrechnung.

[1] Anmerkung: Die Betriebsinformatik basiert auf der Konzeption des Lehrplans für Wirtschaftsinformatik. Aufgrund anderer Stundentafel bedarf es einer schulinternen Lehrplananpassung in der didaktischen Jahresplanung.
[2] http://www.berufsbildung.nrw.de/handreichungen-berufliches-gymnasium/
[3] http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1979/1979_06_01-EPA-berufliche-Informatik.pdf

2.2 Jahrgangsstufe 11.2

Beispielaufgabe zu Objektinteraktionen

Titel	**Kornspeicher**
Vorbemerkung	Die folgende Aufgabe steht als Beispiel für eine umfangreiche ganzheitliche Übungsaufgabe, die Objektinteraktionen zwischen den Objekten einer Klasse thematisiert.
Voraussetzungen gemäß Lehrplan	Grundelemente der Objektorientierung (ohne GUI lösbar) Erweiterung der OOP Botschaften zwischen Objekten einer Klasse
Anwendungsbezug	Lagerlogistik
Fachlicher Schwerpunkt	Interaktion zwischen Objekten einer Klasse
Eingesetzte Hilfsmittel	Programmiersprache Java, Programmierumgebung BlueJ (alternative Lösung mit C# und GUI)
Anmerkungen	keine

Situation

Die Kornhaus Eisen GmbH betreibt in der Warburger Börde einen Getreidegroßhandel. Die vielen landwirtschaftlichen Betriebe der Börde liefern ihr Getreide direkt nach der Ernte dort an. Das Getreide wird in einer Anzahl von verschieden großen Silos gelagert und zu einem späteren Zeitpunkt an Großmühlen verkauft.

Für die Verwaltung der Silos ist eine entsprechende Software zu entwickeln. Die Anforderungen an diese Software sind in dem folgenden Pflichtheft zusammengefasst.

Pflichtenheft

Programmdaten

/D10/ SiloNr
/D20/ Kapazität // als gesamtes Fassungsvermögen des Silos in t
/D30/ Bestand // aktueller Getreidebestand im Silo in t

Programmfunktionen

/F10/ Bei der Erzeugung eines Siloobjektes müssen mindestens die SiloNr und die Kapazität sofort in das Objekt geschrieben werden.

/F15/ Sollte bei der Erzeugung eines Siloobjektes auch der Bestand bekannt sein, so ist dieser sofort beim Erzeugungsvorgang mit in das Objekt zu schreiben.

/F20/ Die Kapazität eines Siloobjektes kann sich nach seiner Erzeugung nicht mehr ändern. Das ist bei der Methode zum Schreiben der Silokapazität zu berücksichtigen.

/F30/ Alle anderen Daten müssen einzeln geschrieben und alle Daten einzeln gelesen werden können.

/F40/ Bei der Programmfunktion Silo **einlagern** ist zu berücksichtigen, dass beim Füllen des Silos das Silo nicht überläuft.

/F50/ Bei der Funktion **auslagern** kann nicht mehr als der vorhandene Bestand aus einem Silo entnommen werden.

/F60/ Die Funktion **umlagern** muss die bestehenden Restriktionen aus /F40/ und /F50/ beachten.

/F70/ Durch die Funktion **trocknen** wird der gesamte Bestand eines Silos über einen Trockner geschickt und mit einem prozentualen Gewichtsverlust wieder in dasselbe Silo zurückgeführt.

Hinweis: Die Software ist so zu konstruieren, dass vor dem Ein- oder Auslagern jeweils die zu bewegende Menge anzugeben ist. Widerspricht diese Menge den bestehenden Restriktionen, wird der gesamte Vorgang des Ein- oder Auslagerns mit einer entsprechenden Meldung nicht durchgeführt.

Testfälle

Folgende Objekte sind zu erzeugen und die anschließend aufgeführten Fälle zu testen:

Silo Nr.	1	2	3	4
Kapazität in t	2000	600	1500	900
Bestand in t	215	588,5	1200	850

(1) Aus Silo Nr. 1 werden 25 t auf einen LKW verladen.

(2) Silo Nr. 2 wird mit 50 t beschickt.

(3) Von Silo Nr. 3 nach Silo Nr. 4 werden 60 t umgelagert.

(4) Der Inhalt von Silo Nr. 4 wird getrocknet mit einem Gewichtsverlust von 5 %.

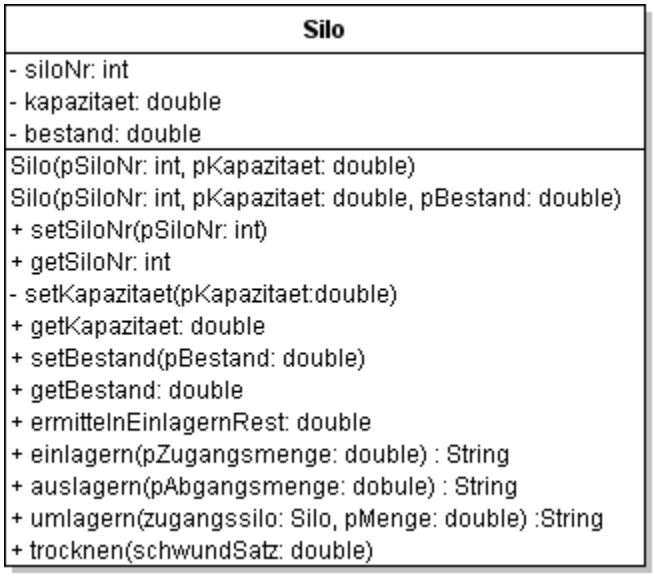
Arbeitsaufträge

A1 Erstellen Sie ein Klassendiagramm für die Siloverwaltung.

A2 Implementieren Sie die Klasse Silo in einem neuen Projekt namens „SiloLoesung“ in BlueJ und dokumentieren Sie dabei Ihren Java-Code.

A3 Testen Sie mit BlueJ die Klasse unter Verwendung der vorgegebenen Objekte und der Testfälle (1) bis (4). Dokumentieren Sie Ihre Testergebnisse.

Lösung zu A1



Lösung zu A2
Anlage des Projektes und der Klasse in BlueJ

Java-Code mit Kommentaren

siehe Folgeseite

```
/**Aufgabe1SiloLoesung04 <br>
 * <hr>
 * Fachklasse: <b> Silo </b>
 *
 * <hr>
 * Aufgabe: Software für Siloverwaltung
 * <hr>
 * @author
 * @ version 1.1 vom 10.07.09
 */

public class Silo
{
    /**
     * Instanzvariablen
     */
    private int siloNr;
    private double kapazitaet; // in t
    private double bestand;    // in t

    /**<b> Konstruktur mit Parametern </b> <br>
     * Jede Siloobjekt verlangt mindestens die Angabe der
     * siloNr und der Kapazität.  Die Kapazität kann
     * sich während der Lebensdauer eines Objektes nicht
     * ändern. Ein Wechsel der SiloNr grundsätzlich möglich.
     * @param pSiloNr für Instanzenvariable kontonummer
     * @param pKapazitaet für Instanzenvariable kapazitaet
     */

    public Silo(int pSiloNr, double pKapazitaet)
    {
        siloNr = pSiloNr;
        kapazitaet =pKapazitaet;

    }

     /**<b> Konstruktur mit Parametern </b> <br>
     * Jede Siloobjekt verlangt mindestens die Angabe der
     * siloNr und der kapazität.  Die Kapazität kann
     * sich während der Lebensdauer eines Objektes nicht
     * ändern. Ein Wechsel der SiloNr grundsätzlich möglich.
     * Ist bei der Instanzierung eines Siloobjektes der akutelle Bestand
     * bekannt, kann dieser gleich über den Parameter pBestand mit in
     * das Objekt geschrieben werden.
     * @param pSiloNr für Instanzenvariable kontonummer
     * @param pKapazitaet für Instanzenvariable kapazitaet
     */

     public Silo(int pSiloNr, double pKapazitaet, double pBestand)
    {
        siloNr = pSiloNr;
        kapazitaet =pKapazitaet;
        bestand = pBestand;

    }

    /**<b> Schreibende Methode </b> <br>
     * Die Nummer des Silos wird mit dieser Methode in das Siloobjekt geschrieben.
     * @param pSiloNr für Instanzenvariable siloNr
     */

    public void setSiloNr(int pSiloNr){
      siloNr = pSiloNr;
    }

    /** <b> Lesende Methode </b> <br>
```

```
67          *   Die Silonummer aus dem Siloobjekt wird an die aufrufenden Instanz überge-
68      ben.
69          *  @return siloNr
70          */
71          public int getSiloNr (){
72          return siloNr;
73          }
74
75         /**<b> Schreibende Methode </b> <br>
76          * Die Kapazität des Silos wird mit dieser Methode in das Siloobjekt geschrie-
77      ben.
78          * @param pKapaziät für Instanzenvariable Kapazität.
79          * Diese private Methode kann nur innerhalb dieser Klasse Silo Verwendung fin-
80      den.
81          */
82         private void setKapazitaet (double pKapazitaet) {
83             kapazitaet = pKapazitaet;
84         }
85
86         /** <b> Lesende Methode </b> <br>
87          *   Die maximale Lagerkapaziät des Silos aus dem Siloobjekt wird an die aufru-
88      fenden Instanz übergeben.
89          *  @return siloNr
90          */
91         public double getKapzitaet(){
92             return kapazitaet;
93            }
94
95          /**<b> Schreibende Methode </b> <br>
96          * Der akutelle im Silo lagernde Bestand wird mit dieser Methode in das Siloob-
97      jekt geschrieben.
98          * @param pKapaziät für Instanzenvariable Kapazität.
99          */
100        public void setBestand (double pBestand) {
101            bestand = pBestand;
102        }
103
104        /** <b> Lesende Methode </b> <br>
105         *   Der akutelle Lagerbestand des Silos aus dem Siloobjekt wird an die aufru-
106     fenden Instanz übergeben.
107         *  @return bestand
108         */
109
110         public double getBestand(){
111             return bestand;
112            }
113
114        /**<b> Funktionsmethode </b> <br>
115         * Die Methode berechnet wie viel Tonnen noch in das Silo des Siloobjektes ein-
116     gelagert werden können.
117         * @retrun restmenge
118         */
119        public double ermittelnEinlagernRest(){
120        double restmenge;
121        restmenge = kapazitaet - bestand;
122        return restmenge;
123        }
124
125        /**<b> Funktionsmethode </b> <br>
126         * Die Methode ermittelt den Bestand im Silo, wenn neue Mengen eingelagert wer-
127     den.
128         * @param pZugangsmenge für die Instanzenvariable bestand.
129         * @retrun meldung (als Information über Erfolg oder Mißerfolg der Einlage-
130     rungsaktion)
131         */
132           public String einlagern(double pZugangsmenge) {
```

```
133         String meldung =„„;
134         double differenzmenge;
135         differenzmenge = ermittelnEinlagernRest();
136 
137         if (differenzmenge>= pZugangsmenge){
138             bestand = bestand + pZugangsmenge;
139             meldung = „Einlagerung war erfolgreich“;
140 
141          }
142          else{
143              meldung = „Einlagerung wegen Platzmangel im Silo nicht möglich.“;
144          }
145 
146         return meldung;
147       }
148 
149       /**<b> Funktionsmethode </b> <br>
150        * Die Methode ermittelt den Bestand im Silo, wenn eine gegebene Menge ausgela-
151    gert wird.
152        * @param pAbgangsmenge für die Instanzenvariable bestand.
153        * @return meldung (als Information über Erfolg oder Misserfolg der Auslage-
154    rungsaktion)
155        */
156 
157       public String auslagern(double pAbgangsmenge){
158       String meldung = „„;
159 
160 
161        if (bestand >= pAbgangsmenge){
162           bestand = bestand - pAbgangsmenge;
163           meldung =„ Auslagerung war erfolgreich.“;
164 
165           }
166           else{
167           meldung = „Angeforderte Menge kann wegen eines zu geringen Bestandes im Si-
168    lo nicht abgegeben werden.“;
169 
170           }
171 
172        return meldung;
173       }
174 
175       /**<b> Funktionsmethode </b> <br>
176        * Die Methode ist stets an dem Siloobjekt aufzurufen, von dem das Getreide ab-
177    geht;
178        * sie ermittelt die neuen Bestandsmengen nach dem Umlagern.
179        * @param Zugangssiloobjekt
180        * @param pMenge
181        * @return meldung (als Information über Erfolg oder Mißerfolg der Umlagerungs-
182    aktion)
183        */
184 
185       public String umlagern( Silo zugangssilo, double pMenge){
186       String meldung =„„;
187       double maxZugang;
188       maxZugang = zugangssilo.ermittelnEinlagernRest();
189 
190       if (this.bestand >= pMenge && maxZugang >= pMenge  ) {
191           this.auslagern(pMenge);
192           zugangssilo.einlagern(pMenge);
193           meldung = „Umlagern wurde erfolgreich abgeschlossen.“;
194 
195       }
196       else{
197        meldung = „Umlagern nicht möglich, wegen Beschränkungen im Abgangs- oder Zu-
198    gangssilo.“;
```

```
199         }
200         return meldung;
201         }
202 
203     /** <b> Funktionsmethode </b> <br>
204          * Die Methode errechnet nach Übergabe des Schwundsatzes beim Trocken den neuen
205     Bestand im Silo
206          * nach einem Trocknungsvorgang.
207          * @param schwundSatz als Prozentsatz
208          */
209         public void trocknen(double schwundSatz){
210             bestand = bestand * (100-schwundSatz)/100;
211         }
212 
213     }
214 
```

Anmerkung:
JDoc bietet die Möglichkeit die Dokumentation als HTML-Seite aufzubereiten:

Method Summary

String	**auslagern**(double pAbgangsmenge) **Funktionsmethode** Die Methode ermittelt den Bestand im Silo, wenn eine gegebene Menge ausgelagert wird.
String	**einlagern**(double pZugangsmenge) **Funktionsmethode** Die Methode ermittelt den Bestand im Silo, wenn neue Mengen eingelagert werden.
double	**ermittelnEinlagernRest**() **Funktionsmethode** Die Methode berechnet wie viel Tonnen noch in das Silo des Siloobjektes eingelagert werden können.
double	**getBestand**() **Lesende Methode** Der akutelle Lagerbestand des Silos aus dem Siloobjekt wird an die aufrufenden Instanz übergeben.
double	**getKapzitaet**() **Lesende Methode** Die maximale Lagerkapaziät des Silos aus dem Siloobjekt wird an die aufrufenden Instanz übergeben.
int	**getSiloNr**() **Lesende Methode** Die Silonummer aus dem Siloobjekt wird an die aufrufenden Instanz übergeben.
void	**setBestand**(double pBestand) **Schreibende Methode** Der akutelle im Silo lagernde Bestand wird mit dieser Methode in das Siloobjekt geschrieben.
void	**setSiloNr**(int pSiloNr) **Schreibende Methode** Die Nummer des Silos wird mit dieser Methode in das Siloobjekt geschrieben.
String	**umlagern**(Silo zugangssilo, double pMenge) **Funktionsmethode** Die Methode ist stets an dem Siloobjekt aufzurufen, von dem das Getreide abgeht; sie ermittelt die neuen Bestandsmengen nach dem Umlagern.

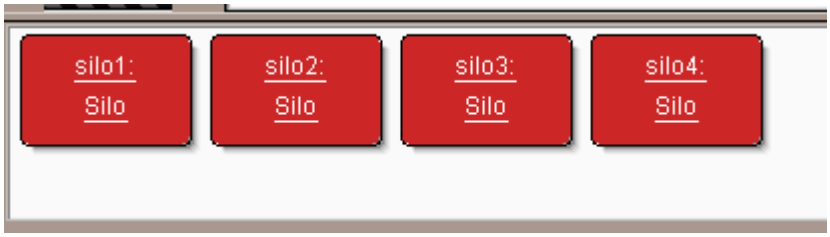
void	**trocknen**(double schwundSatz) **Funktionsmethode** Die Methode errechnet nach Übergabe des Schwundsatzes beim Trocken den neuen Bestand im Silo nach einem Trocknungsvorgang.

Constructor Summary

Silo(int pSiloNr, double pKapazitaet) **Konstruktur mit Parametern** Jede Siloobjekt verlangt mindestens die Angabe der siloNr und der kapazität.
Silo(int pSiloNr, double pKapazitaet, double pBestand) **Konstruktur mit Parametern** Jede Siloobjekt verlangt mindestens die Angabe der siloNr und der kapazitä; ist bei der Instanzierung eines Silobjekts der aktuelle Bestand bekannt, kann dieser mit in das Objekt geschrieben werden.

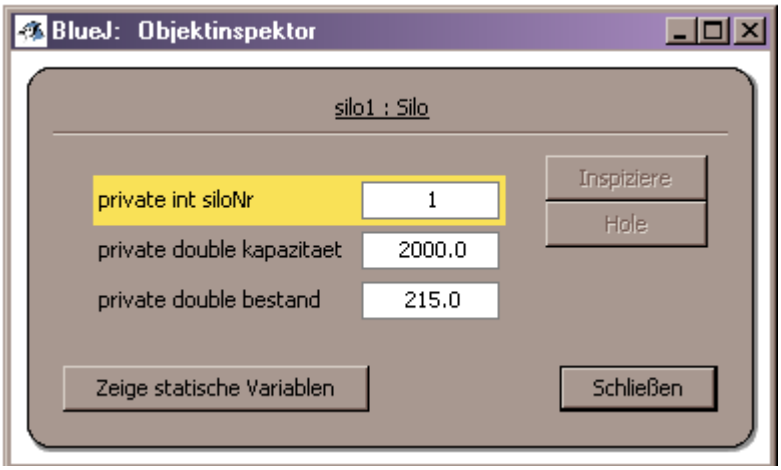
Lösung zu A3

Generierte Objekte für den Test

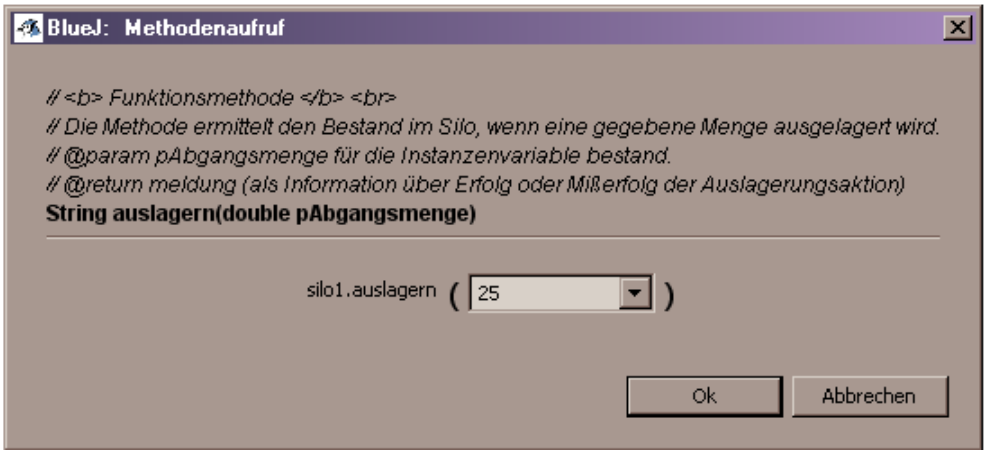


(1) Aus Silo Nr. 1 werden 25 t auf einen LKW verladen.

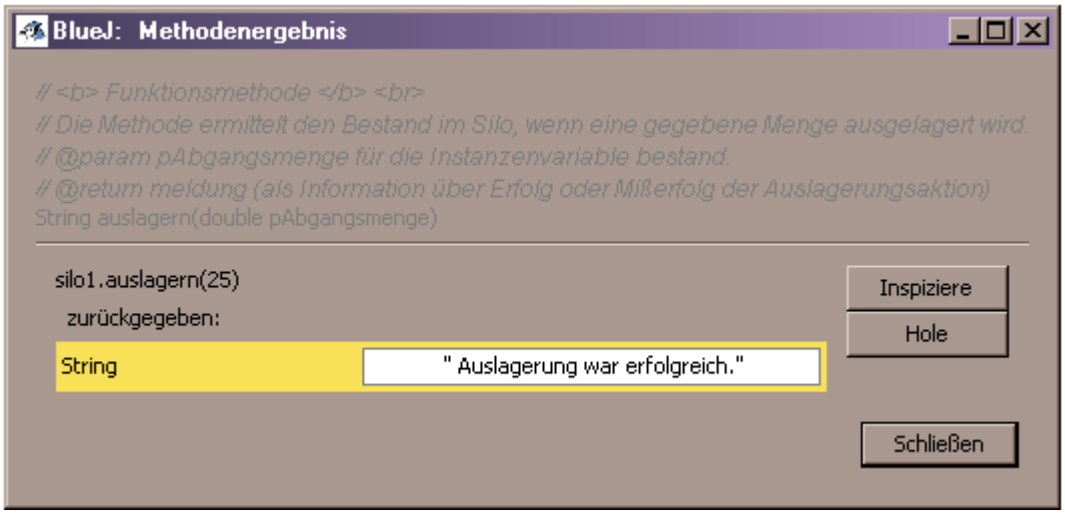
SiloObjekt vor Ausführen von Testfall 1



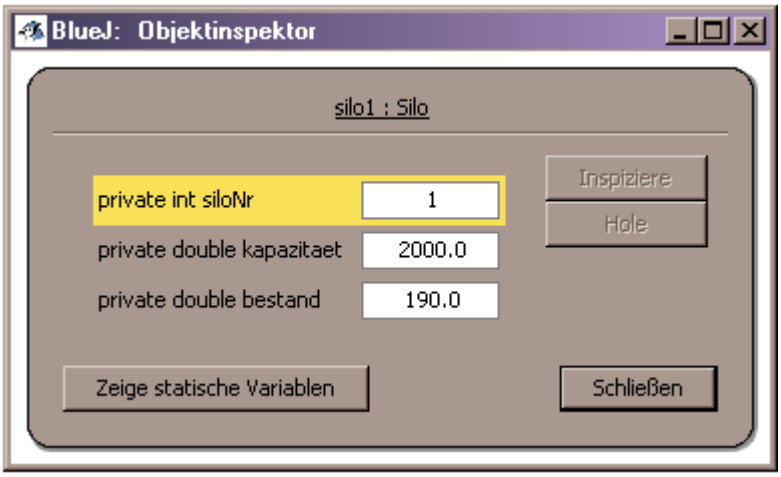
Durchführung von Testfall 1



Ergebnis

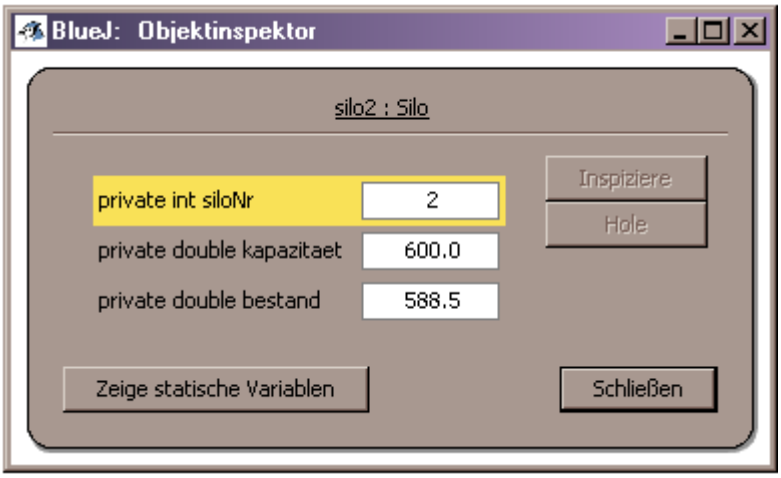


Objektzustand nach Testfall 1, also korrekter Testverlauf

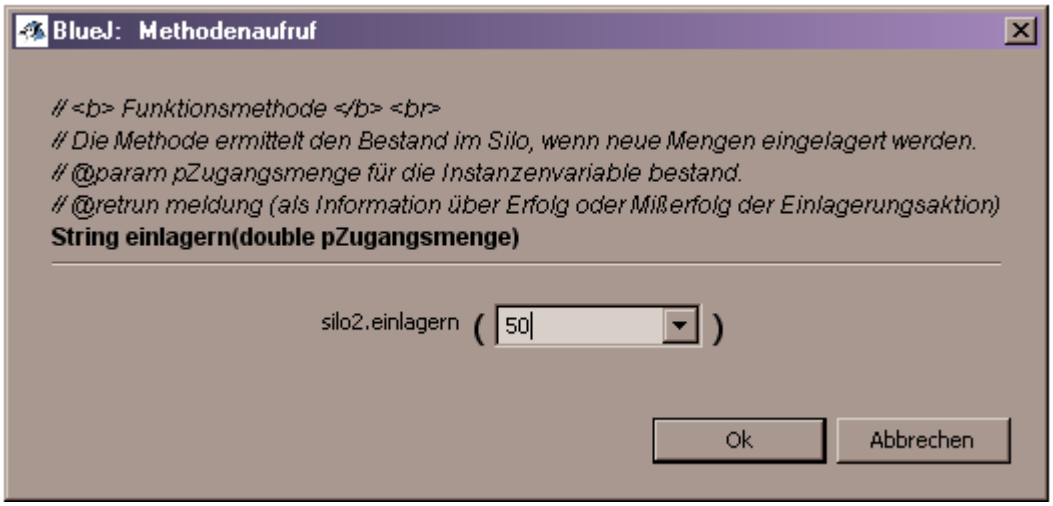


(2) Silo Nr. 2 wird mit 50 t beschickt.

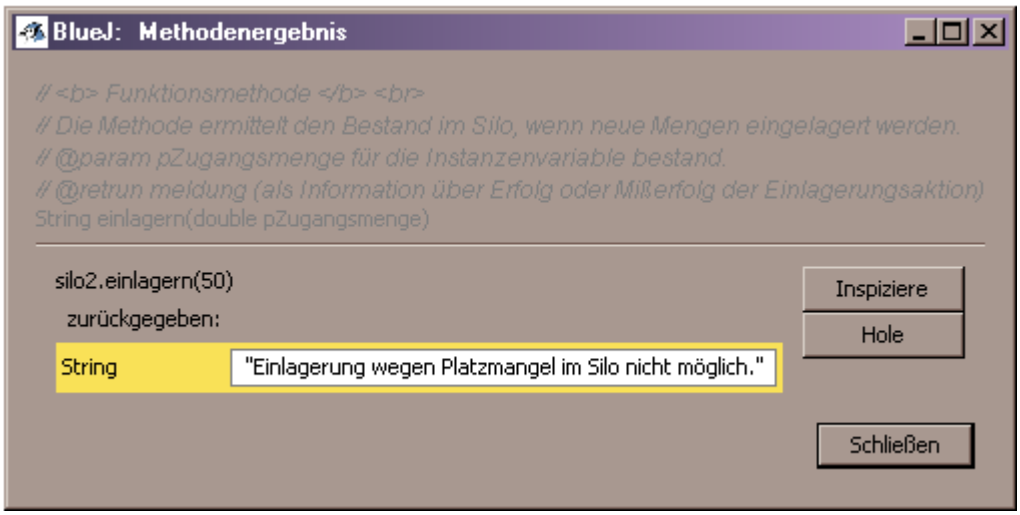
SiloObjekt vor Ausführen von Testfall 2



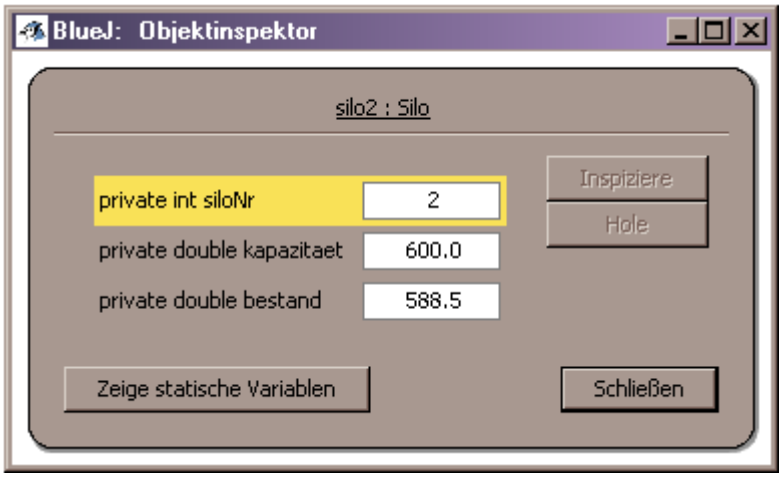
Durchführung von Testfall 2



Ergebnis

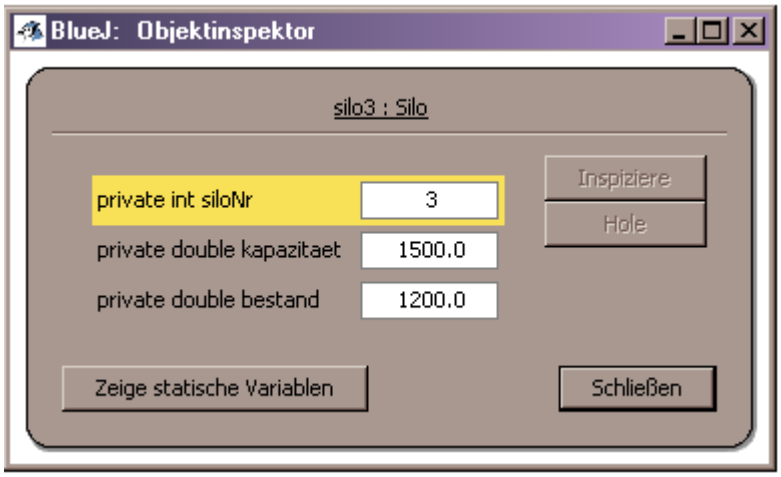


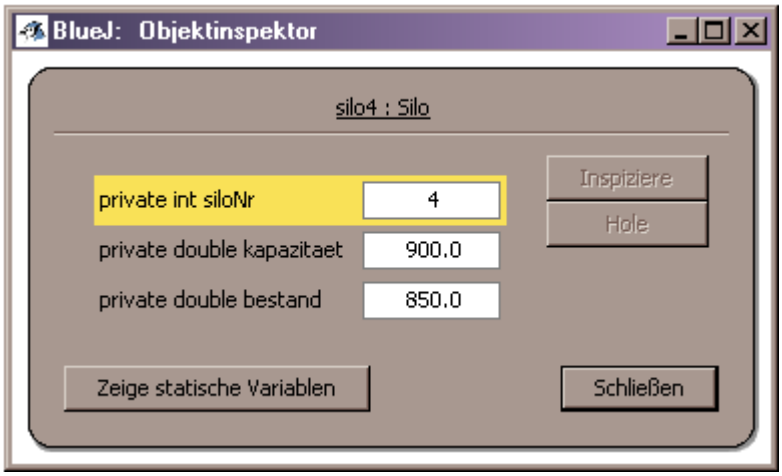
Objektzustand nach Testfall 2, keine Einlagerung, also korrekter Testverlauf



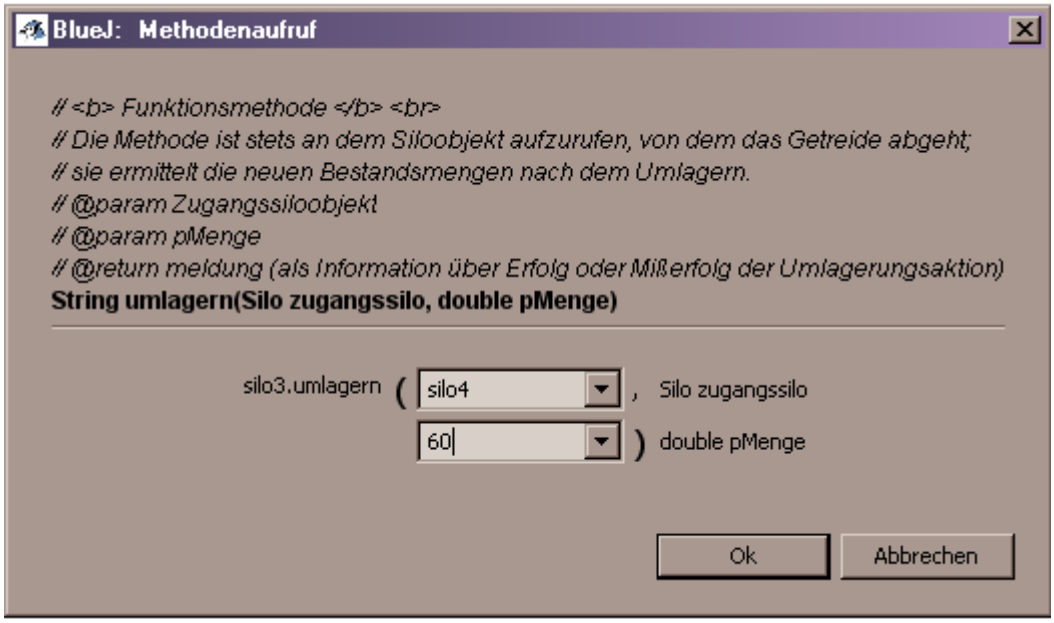
(3) Von Silo Nr. 3 nach Silo Nr. 4 werden 60 t umgelagert.

Objektzustände vor Ausführung von Testfall 3

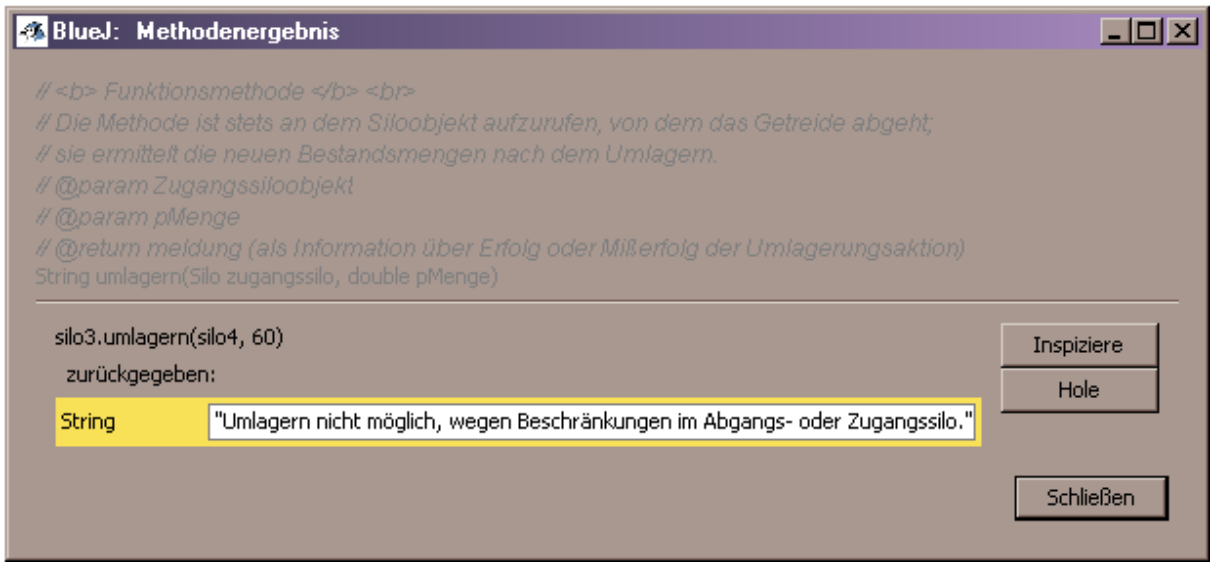




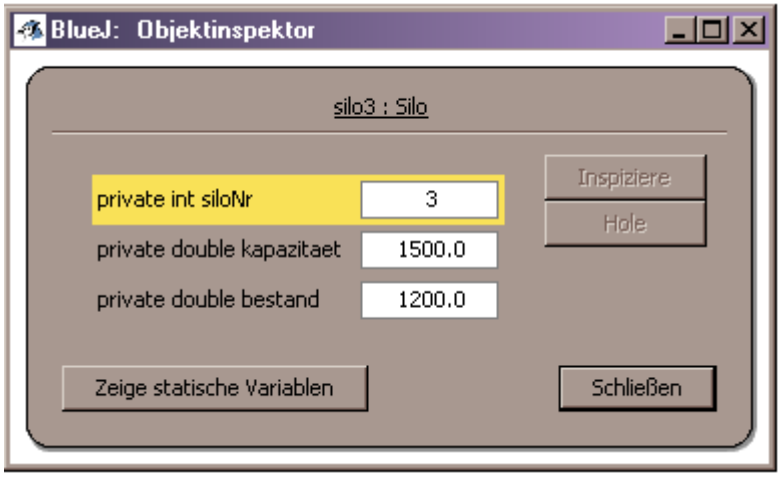
Durchführen von Testfall 3

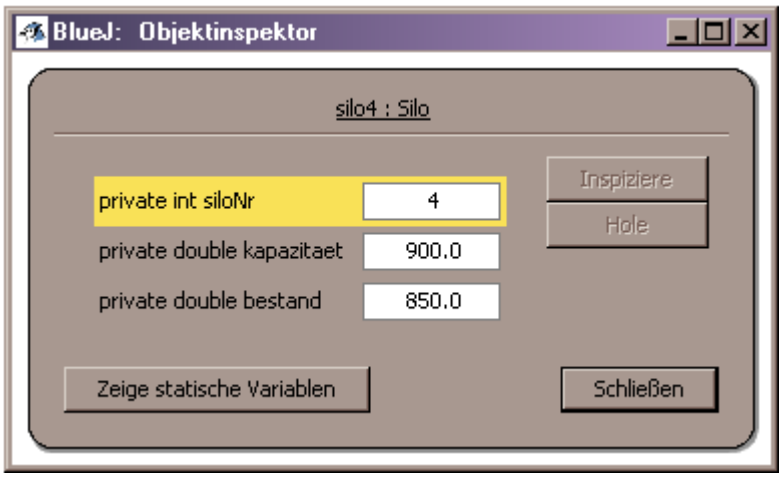


Ergebnis



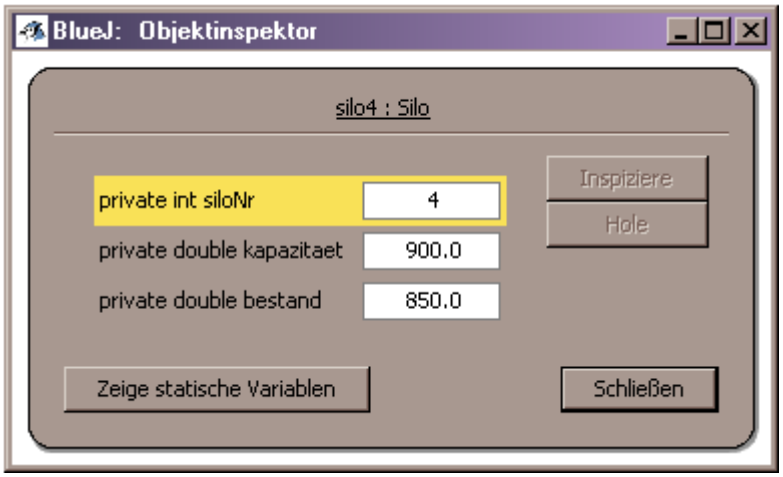
Objektzustände nach Testfall 2, keine Umlagerung, also korrekter Testverlauf



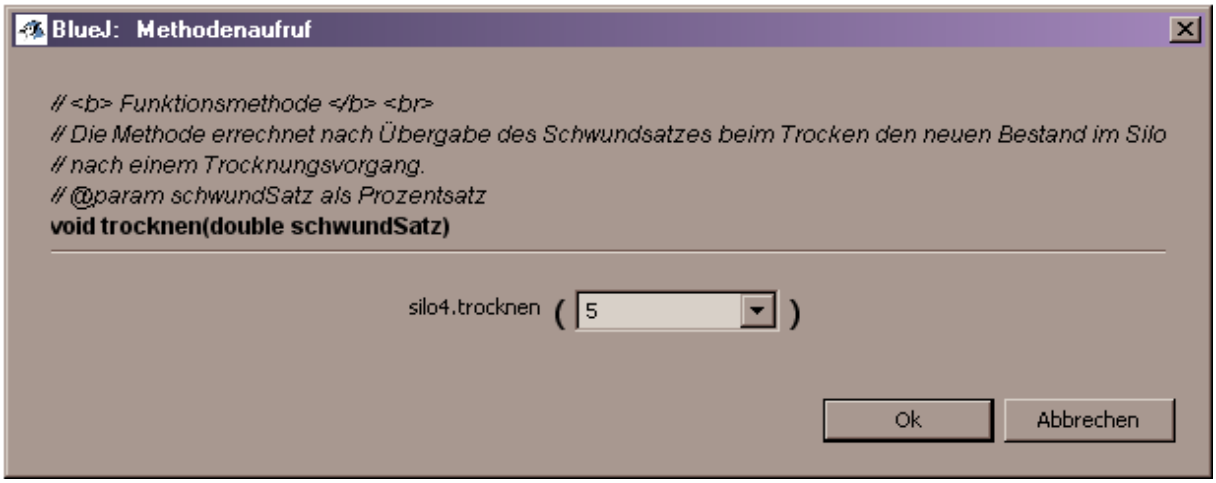


(4) Der Inhalt von Silo Nr. 4 wird getrocknet mit einem Gewichtsverlust von 5 %.

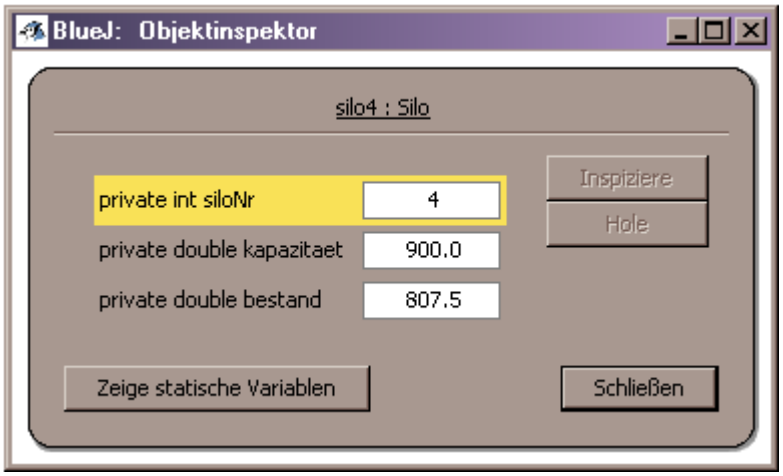
Objektzustand vor Ausführung von Testfall 4



Durchführen von Testfall 4

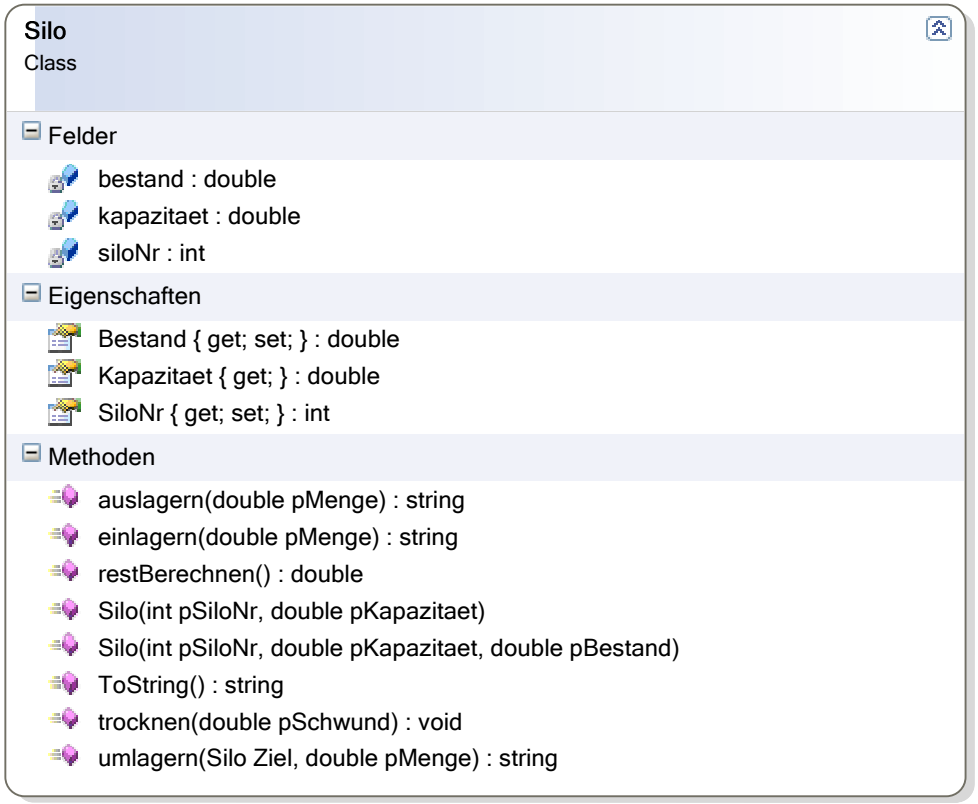


Objektzustand nach Ausführung von Testfall 4, also korrekter Testverlauf



Alternativer Lösungsansatz in C# (VS 2008)

Lösung zu A1



Alternativ zum Klassendiagrammeditor kann auch ein Tool wie Jude oder Visio verwendet werden.

Lösung zu A2

Fügen Sie zum Projekt eine neue Klasse Silo hinzu (z. B.: über den Projektmappenexplorer)

Hier folgt der Code incl. der integrierten Kommentare:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace SiloCSharp
{
    class Silo
    {
        private int siloNr;

        public int SiloNr
        {
            get { return siloNr; }
            set { siloNr = value; }
```

```
        }
        private double kapazitaet;

        /// <summary>
        /// Wert darf nur gelesen, und nach Initialisierung nicht
mehr geändert werden.
        /// </summary>
        /// <value>Menge, die ein Silo fasst</value>
        public double Kapazitaet
        {
            get { return kapazitaet; }

        }
        private double bestand;

        /// <summary>
        /// Aktuelle Füllmenge des Silos in Tonnen.
        /// </summary>
        public double Bestand
        {
            get { return bestand; }
            set { bestand = value; }
        }

        /// <summary>
        /// Konstruktor in dem alle Felder gesetzt werden können.
        /// </summary>
        /// <remarks>leerer Konstruktor laut Pflichtenheft nicht
zulässig</remarks>
        /// <returns>keine</returns>
        public Silo(int pSiloNr, double pKapazitaet, double
pBestand):this(pSiloNr,pKapazitaet)
        {
            //Verkettung von Konstruktoren
            bestand = Math.Max(pBestand, 0);
        }

        /// <summary>
        /// /F15/ Konstruktor sieht mindestens die Festlegung der
SiloNr und der Kapazität vor.
        /// </summary>
        /// <remarks>einen leeren Konstruktor darf es laut
Pflichtenheft nicht geben.</remarks>
        /// <param name=„pSiloNr“>ganzzahliger Wert</param>
        /// <param name=„pKapazitaet“>positiver Wert</param>
        /// <returns>keine</returns>
        public Silo(int pSiloNr, double pKapazitaet)
        {
            siloNr = Math.Max(pSiloNr,0);
            kapazitaet = Math.Max(pKapazitaet,0);
        }

        /// <summary>
```

```
        /// /F40/ In ein Silo können mit Hilfe dieser Methode
zusätzliche Mengen eingelagert werden. Würde durch das Einlagern die
Kapazität überschritten, wird der Vorgang nicht durchgeführt und
entsprechend protokolliert.
        /// </summary>
        /// <remarks>Teileinlagerung ist nicht vorgesehen. Ein
Auftrag ist ganz oder gar nicht abzuwickeln.</remarks>
        /// <param name=„pMenge“>positiver Wert</param>
        /// <returns>Text mit „Aktion erfolgreich durchgeführt“ bzw.
„Aktion nicht ausgeführt, da Kapazität nicht ausreichend“</returns>
        public string einlagern(double pMenge)
        {
            pMenge = Math.Max(pMenge, 0);
            string Ausgabe = „Aktion erfolgreich“;
            if (pMenge <= this.restBerechnen())
            {
                bestand += pMenge;
            }
            else
            {
                Ausgabe = „Aktion nicht ausgeführt, da Kapazität
nicht ausreichend“;

            }
            return Ausgabe;


        }

        /// <summary>
        /// /F50/ Bei dieser Methode reduziert sich der Bestand um
die übergebene Menge, es sei denn es soll mehr ausgelagert werden,
als vorrätig ist.
        /// </summary>
        /// <remarks>Aktion wird vollständig oder gar nicht
ausgeführt!</remarks>
        /// <param name=„pMenge“>positiver Wert</param>
        /// <returns>Text „Aktion erfolgreich“ bzw. „Aktion wurde
nicht durchgeführt, es sollte mehr als der vorhandene Bestand
ausgelagert werden“.</returns>
        public string auslagern(double pMenge)
        {
            pMenge = Math.Max(pMenge, 0);
            string Ausgabe = „Aktion erfolgreich“;
            if (pMenge <= bestand)
            {
                bestand -= pMenge;
            }
            else
            {
                Ausgabe = „Aktion wurde nicht durchgeführt, es
sollte mehr als der vorhandene Bestand ausgelagert werden“;
            }
            return Ausgabe;
```

```
        }

        /// <summary>
        /// Aus dem angesprochenen Silo wird die übergeben Menge in
das ebenfalls angegeben Zielsilo übertragen, sofern dessen Kapazität
ausreicht und das abgebende Silo hinreichend Bestand aufweist.
        /// </summary>
        /// <returns>Text „Aktion erfolgreich“ bzw. „Aktion nicht
erfolgt , da Bestand im abgebenden Silo nicht ausreichend“ oder
„Aktion nicht erfolgt, da Aufnahmekapazität des Zielsilos
unzureichend.</returns>
        public string umlagern(Silo Ziel, double pMenge)
        {
            string Ausgabe = „Ausgabe erfolgreich“;
            if ((pMenge <= Ziel.restBerechnen()) && pMenge <=
bestand)
            {
                this.auslagern(pMenge);
                Ziel.einlagern(pMenge);
            }
            else
            {
                Ausgabe = „Aktion nicht durchgeführt, Restriktionen
verletzt“;
            }
            return Ausgabe;
        }

        /// <summary>
        /// /F70/
        /// Reduktion des Lagerbestandes durch Trocknen um einen
übergebenen Prozentsatz.
        /// </summary>
        /// <param name=„pSchwund“>Prozentsatz zwischen 0 und
100</param>
        public void trocknen(double pSchwund)
        {
            pSchwund = Math.Min(Math.Max(0, pSchwund), 100);
            /* Hier wird gesichert, dass nur Prozentsätze zwischen
             * 0 und 100 verwendet werden.  */
            this.auslagern((pSchwund / 100) * bestand);
        }

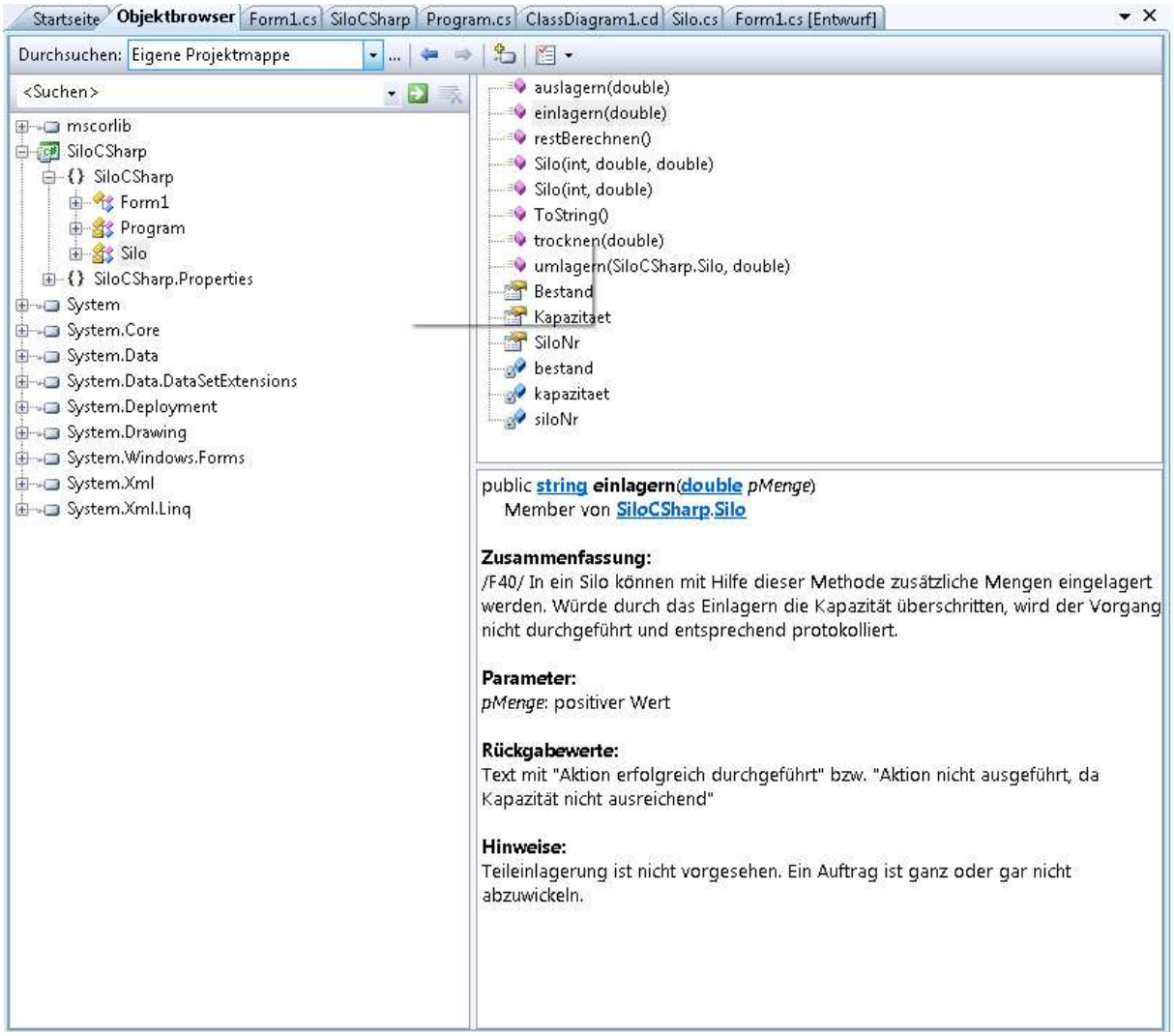
        /// <summary>
        /// Zur Prüfung der Restriktionen, wird hier die noch
mögliche Nachfüllmenge des Silos berechnet.
        /// </summary>
        /// <returns>mögliche Restmenge= Kapazität-
Bestand</returns>
        public double restBerechnen()
        {
            return kapazitaet - bestand;
        }
```

```
        public override string ToString()
        {
            string Ausgabe="";
            Ausgabe = „SiloNr: „ + siloNr.ToString() + „ \n\r“;
            Ausgabe = Ausgabe + „Kapazität: „ +
kapazitaet.ToString() + „ \n\r“;
            Ausgabe = Ausgabe + „Bestand: „ + bestand.ToString() ;
            return Ausgabe;


        }
    }
}
```

Anmerkung:

Im Objektbrowser lassen sich alle Kommentare in aufbereiteter Form nachschlagen



Alle Kommentare werden in einer XML-Datei ausgelagert. Diese kann mit entsprechendem Schema oder beispielsweise dem kostenlosen Tool NDoc aufbereitet werden.

Lösung zu A3

Dokumentation der Testfälle

I: Mit Oberfläche

```
private void Form1_Load(object sender, EventArgs e)
        {

            string neueZeile = „\n\r“;
            string test = „Testfallprotokoll „ + neueZeile +
neueZeile; ;
            Silo silo1 = new Silo(1, 2000, 215);
            Silo silo2 = new Silo(2, 600, 588.5);
            Silo silo3 = new Silo(3, 1500, 1200);
            Silo silo4 = new Silo(4, 900, 850);
            // (1) aus Silo Nr 1 werden 25 t verladen

            test = „(1) 25t auslagern: „ + silo1.auslagern(25) ;
            test = test + neueZeile + silo1.ToString();

            // (2) Silo Nr 2 50t einfüllen:

            test = test + neueZeile + neueZeile;
            test = test + „(2) 50t in Silo2 einlagern: „
+silo2.einlagern(50);
            test = test + neueZeile + silo2.ToString();


            // (3) Von Silo 3 60t nach Silo4 umlagern

            test = test + neueZeile + neueZeile;
            test = test + „(3) 60t von Silo 3 nach Silo4 umlagern: „
+ silo3.umlagern(silo4,60);
            test = test + neueZeile + silo3.ToString();
            test = test + neueZeile + silo4.ToString();

            // (4) Inhalt Silo 4 um 5% eintrocknen

            test = test + neueZeile + neueZeile;
            test = test + „(4) Silo 4 um 5% eintrocknen „;
            silo4.trocknen(5);
            test = test +  neueZeile + silo4.ToString();
            //

            lblTest.Text = test;

        }
```

Form1

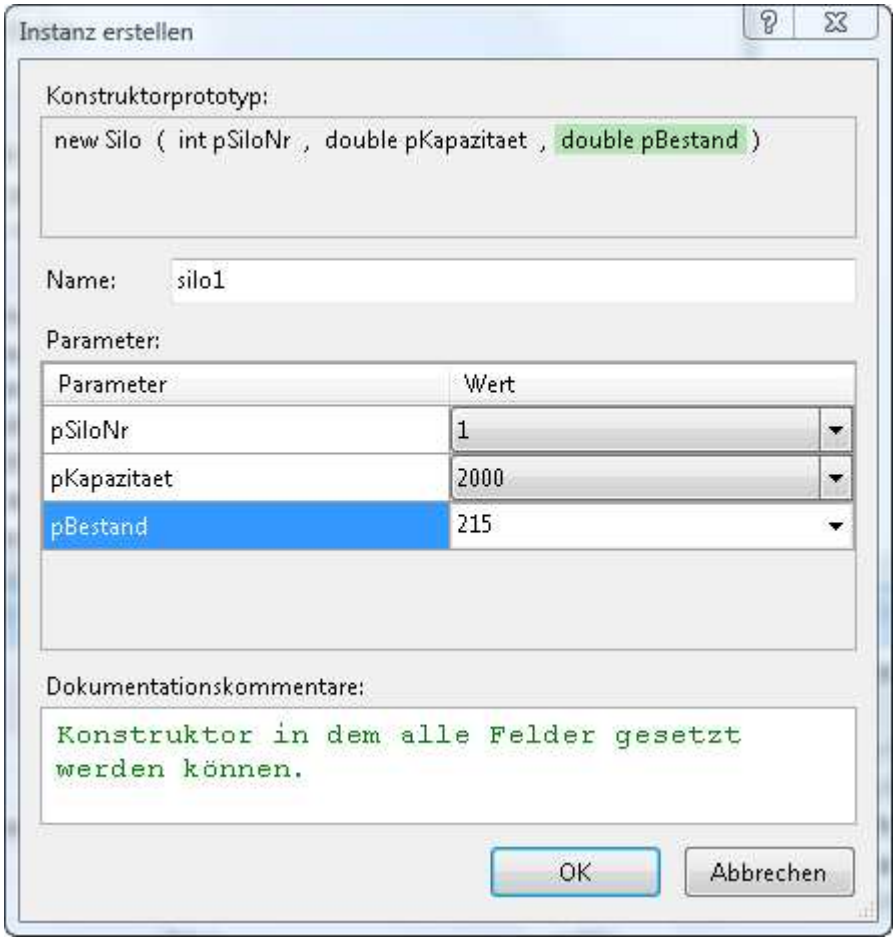
```
(1) 25t auslagern: Aktion erfolgreich
SiloNr: 1
Kapazität: 2000
Bestand: 190

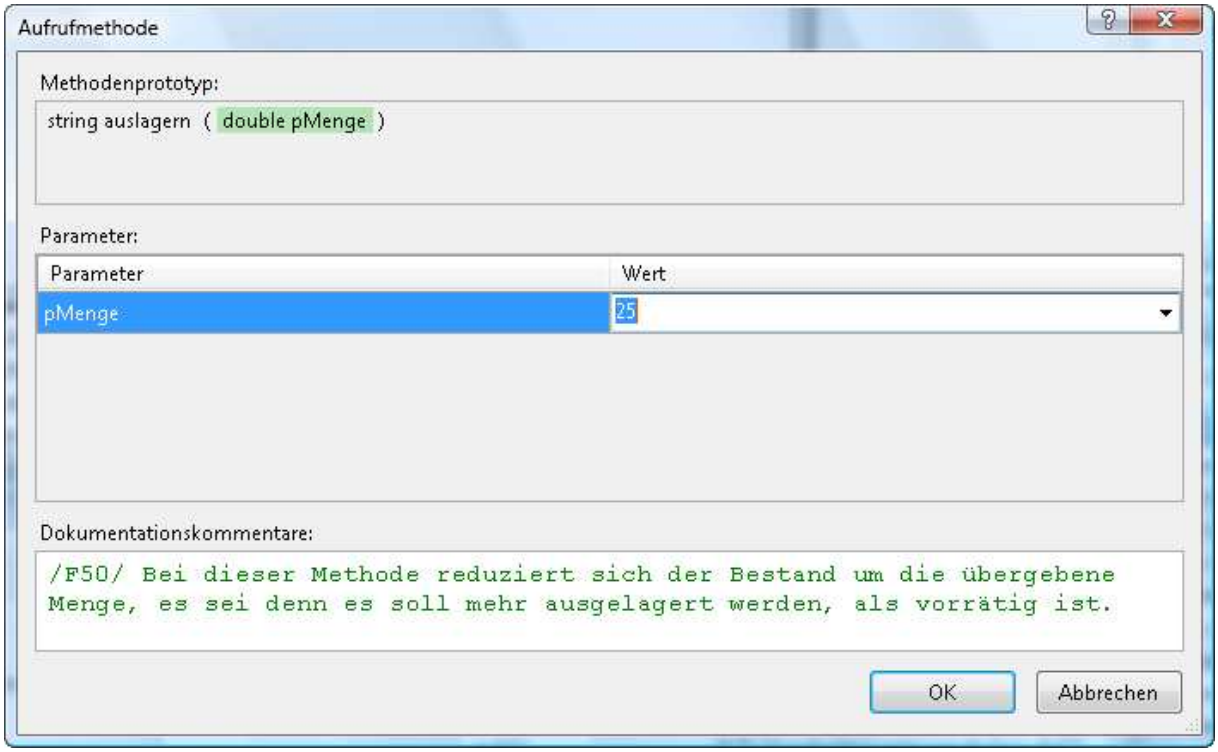
(2) 50t in Silo2 einlagern: Aktion nicht ausgeführt, da Kapazität nicht ausreichend
SiloNr: 2
Kapazität: 600
Bestand: 588,5

(3) 60t von Silo 3 nach Silo4 umlagern: Aktion nicht durchgeführt, Restriktionen verletzt
SiloNr: 3
Kapazität: 1500
Bestand: 1200
SiloNr: 4
Kapazität: 900
Bestand: 850

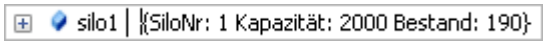
(4) Silo 4 um 5% eintrocknen
SiloNr: 4
Kapazität: 900
Bestand: 807,5
```

II. Oder im Objekttestcenter:




Aufrufmethode
Methodenprototyp:
string auslagern (double pMenge)
Parameter:
Parameter
Wert
pMenge
25
Dokumentationskommentare:
/F50/ Bei dieser Methode reduziert sich der Bestand um die übergebene Menge, es sei denn es soll mehr ausgelagert werden, als vorrätig ist.
OK
Abbrechen


Ergebnis des Methodenaufrufs
Der folgende Methodenaufruf wurde erfolgreich ausgeführt:
string auslagern (25)
Rückgabewert:
"Aktion erfolgreich"
Rückgabewert speichern:
string1
Wiederholen
OK


silo1 | {SiloNr: 1 Kapazität: 2000 Bestand: 190}

usw.

2.3 Jahrgangsstufe 12.1

Aufgabe

Titel	**Portorechner**
Vorbemerkung	Die folgende Aufgabe steht als Beispiel für eine mögliche Klausuraufgabe, bei der Vererbungsstrukturen modelliert und implementiert werden.
Voraussetzungen gemäß Lehrplan	Generalisierung/Spezialisierung GUI Objektsammlungen (für die Zusatztaufgabe)
Anwendungsbezug	Portokostenberechnung eines Dienstleistungsanbieters (Portorechner)
Fachlicher Schwerpunkt	Vererbung
Eingesetzte Hilfsmittel	Programmiersprache C#, Programmierumgebung Visual Studio Alternativlösung mit Java, Programmierumgebung NetBeans
Anmerkungen	Zeitbedarf: ca. 2 Zeitstunden (ohne Zusatzaufgabe)

Situation

Sie arbeiten beim Softwareentwicklungsunternehmen SysDev in Gescher und werden von dem Unternehmen L&P beauftragt einen Preisrechner für Kundenabfragen zu entwickeln.

L&P ist ein privates Dienstleistungsunternehmen für die Zustellung von Briefen und Paketen. Dieses Unternehmen L&P möchte ein neuartiges Konzept zur Berechnung des Portos einführen.

Es werden zwei Postprodukte unterschieden, der Brief und das Paket. Für den Brief gibt es nur drei feste Formate *C4*, *C5* und *C6*, und für Pakete gibt es die Formate *Standard* (=Quader) und *Sperrgut*.

Gemeinsam mit L&P haben Sie bereits Vorgaben für die Portoberechnung ermittelt. Einen Auszug des entstandenen Pflichtenhefts finden Sie im Folgenden.

Produktfunktionen

/F10/ Portoberechnung für alle Postprodukte

Im Allgemeinen berechnet sich das Porto durch folgende Vorschrift:

$$Porto = 30Cent + Entfernungskm \cdot 0{,}05\frac{Cent}{km}$$

/F20/ Portoberechnung für Briefe

Für das Format C6 wird die Portoberechnung für alle Postprodukte übernommen, für C5 gibt es einen 20 %-igen und für C4 einen 40 %-igen Aufschlag. Für Briefe, die schwerer als 100 g sind, verdoppelt sich der Preis.

/F30/ Portoberechnung für Pakete

Für Pakete ergibt sich das Porto wie folgt:

2 Euro	Grundgebühr
+n * 50 Cent	Anzahl der angefangenen 100 km (Entfernung) n berechnet sich durch Aufrundung des Quotienten Entfernung/100 auf die nächste ganze Zahl
+m * 30 Cent	m= Anzahl der angefangenen 1000 g m berechnet sich durch Aufrundung des Quotienten Gewicht/1000 auf die nächste ganze Zahl
+4 Cent/cm³ * 0,1 % des Volumens	

Für das Format „*Sperrgut*“ **verdreifacht** sich die **Grundgebühr**.

Beispiel:

Folgende Sendung wird aufgegeben:

Produktart:	Paket
Entfernungskm:	280 km
Länge:	50 cm
Breite:	70 cm
Höhe:	35 cm
Gewicht:	4800 g
Format:	Standard

Dann beträgt das Volumen 122500 cm³.

Das Porto berechnet sich durch:

Grundgebühr	200 Cent
Je angefangene 100 km Entfernung (n=3)	150 Cent
Je angefangene 1000 g (m=5)	150 Cent
0,1 % des Volumen * 4 Cent/cm³	490 Cent
Summe:	990 Cent = 9,90 Euro

/F40/ Ausgabe des berechneten Portos

Das Ergebnis wird als Text aufbereitet, z. B.:

„Produktart: *Paket*, Porto: *9,90* EUR“
(Hierbei hängen die kursiven Angaben von den Daten des jeweiligen Objekts ab).

Produktdaten

/D10/ Eigenschaften aller Postprodukte:

- Produktart (Brief, Paket)
- Entfernungskm (in km)
- Gewicht (in Gramm)

/D20/ Zusätzliche Eigenschaften für Briefe

- Briefformat (C4, C5, C6)

/D30/ Zusätzliche Eigenschaften für Pakete

- Länge
- Breite
- Höhe
- Paketformat (Standard, Sperrgut)

Anmerkung: Beim Format Sperrgut, also Sonderformate wie z. B. Prisma oder Zylinder, werden die Maße des kleinsten Quaders genommen, in die das Sonderformat ganz hineinpasst.

Arbeitsaufträge

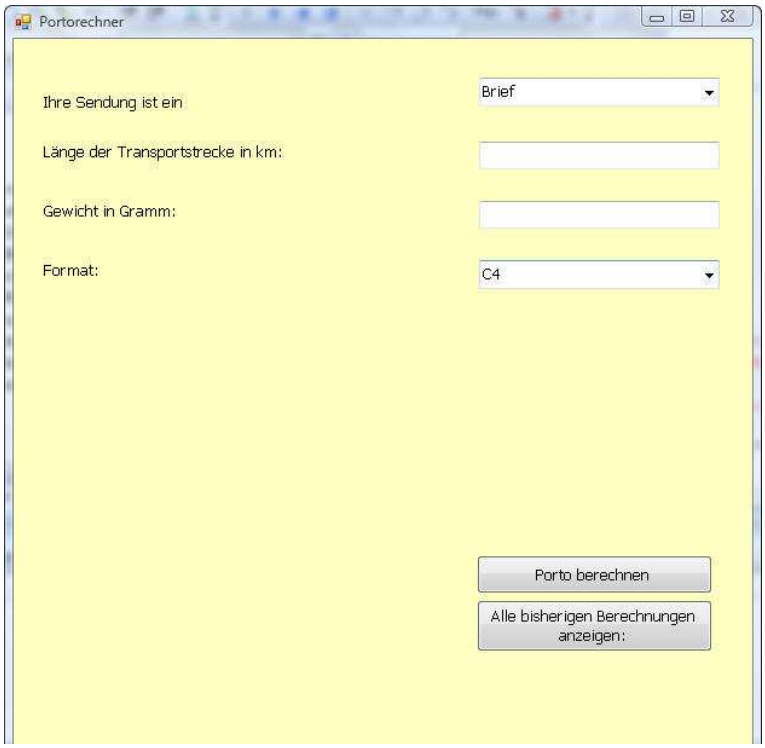
Für beide Postprodukte soll eine Anwendung erstellt werden, die die Berechnung des Portos ermöglicht.

A1 Da die Pakete und Briefe Gemeinsamkeiten besitzen, sollen gemeinsame Eigenschaften und Methoden in einer Elternklasse zusammengefasst werden. Entwerfen Sie unter Berücksichtigung der vorliegenden Informationen ein geeignetes Klassendiagramm mit den Klassen Postprodukt, Brief und Paket. Nutzen Sie für die Produktfunktion /F40/ die ToString()-Methode.

A2 Implementieren Sie die modellierten Klassen gemäß des erstellten Klassendiagramms und den oben gegebenen Beschreibungen. Denken Sie hierbei an alle durch die Vererbung verursachten Besonderheiten.

A3 Implementieren Sie nun zum Testen den Portorechner mit einer Oberfläche nach folgendem Muster:

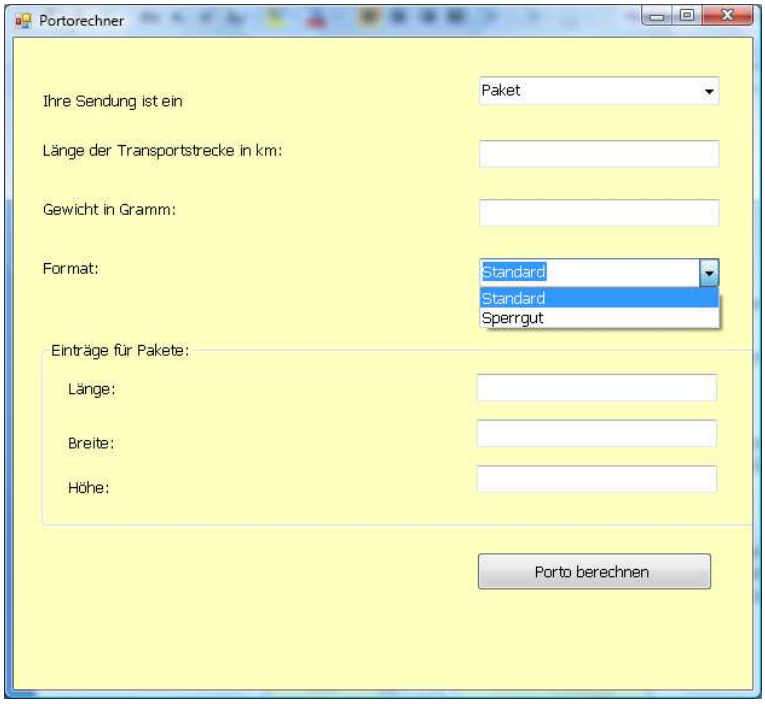
Beim ersten Aufruf sieht die Oberfläche wie folgt aus:



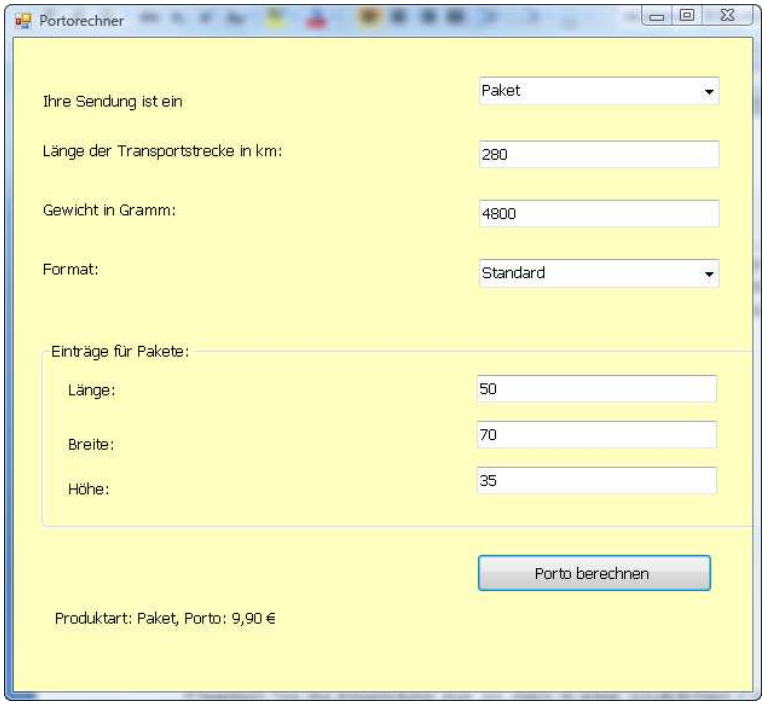
In der Auswahlbox zur Produktart steht die Auswahl *Brief* bzw. *Paket* zur Auswahl.

Solange *Brief* ausgewählt ist, kann in der Combobox das Format für Briefe (C4, C5, C6) ausgewählt werden.

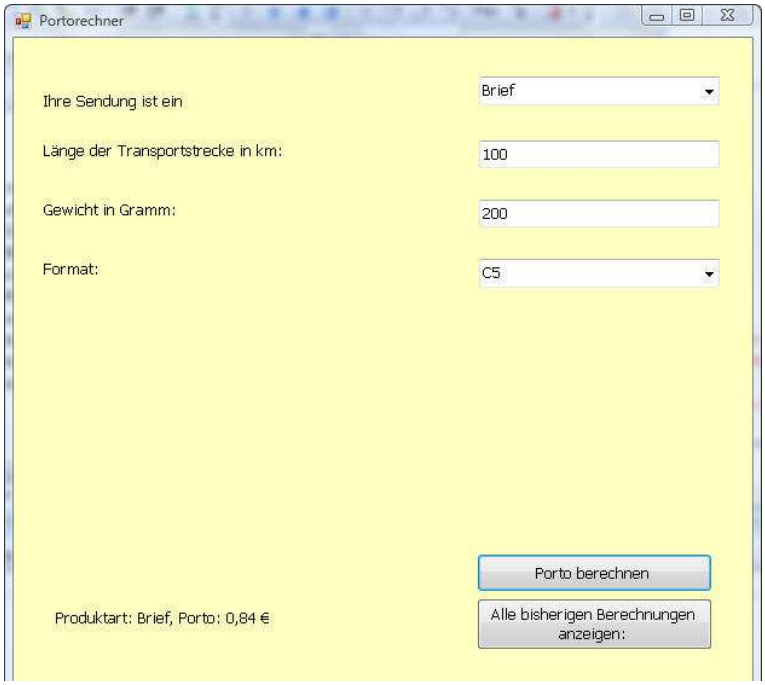
Wird *Paket* ausgewählt, werden die zusätzlichen Eigenschaften der Unterklasse *Paket* sichtbar und es werden nur noch die Formate *Standard* und *Sperrgut* angeboten.



Je nachdem ob *Paket* oder *Brief* ausgewählt wird, wird ein Objekt der Klasse *Paket* oder *Brief* instanziiert und das Porto berechnet und ausgegeben. Für ein Paket sieht das Ergebnis dann so aus:



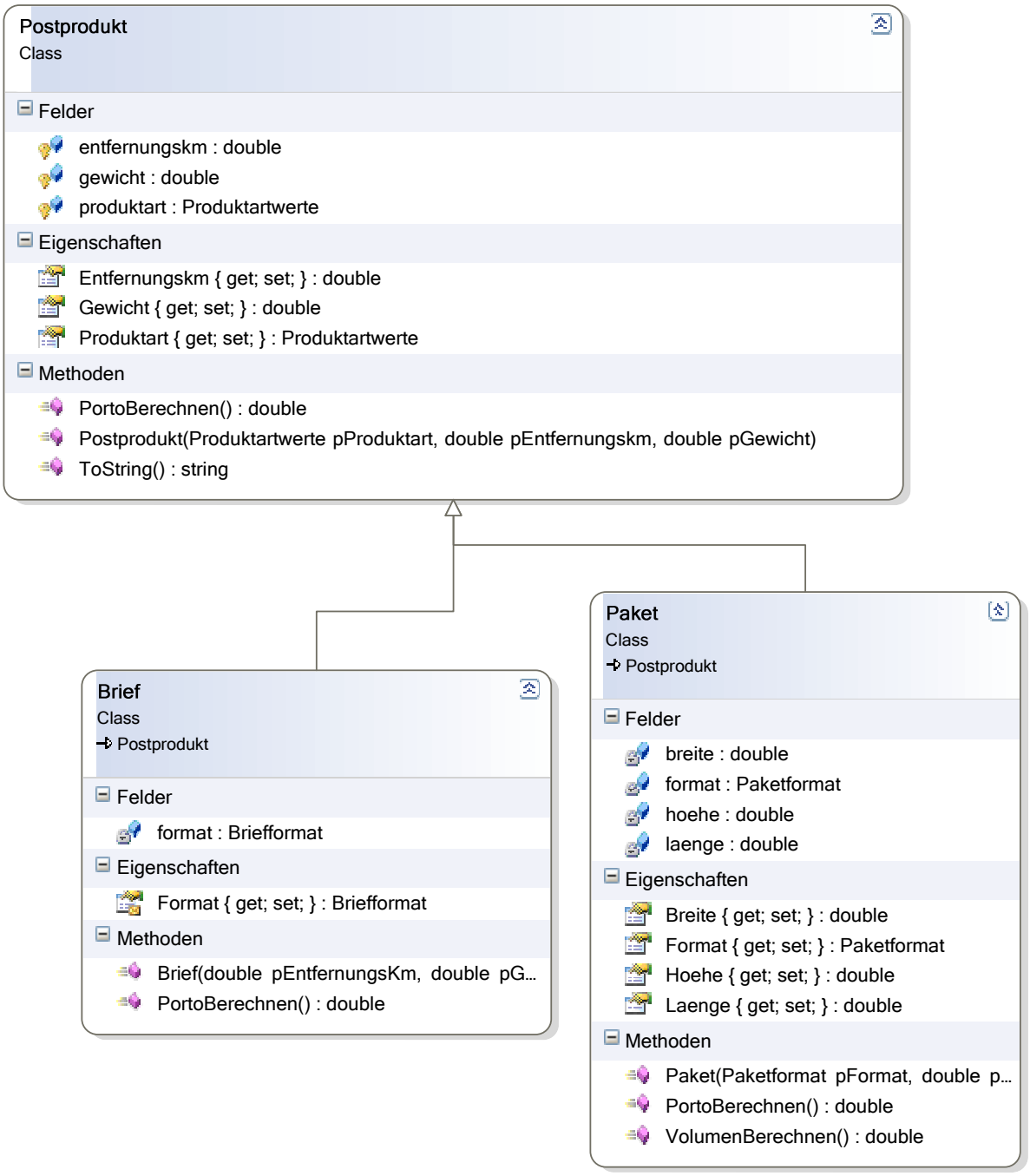
Für einen Brief z. B.:



A4 (Zusatzaufgabe)
Erweitern Sie die Anwendung nun so, dass in einem zusätzlichen Label oder in einer Listbox alle bereits durchgeführten Berechnungen angezeigt werden. Nutzen Sie hierzu eine Arraylist.

Lösung (für C#)

Zu A1



Zu A2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Portorechner
{
    public enum Produktartwerte { Brief, Paket};
    public enum Briefformat { C4, C5, C6 };
    public enum Paketformat { Standard, Sperrgut };


    public class  Postprodukt
    {

        protected Produktartwerte produktart;

        public Produktartwerte Produktart
        {
            get { return produktart; }
            set { produktart = value; }
        }
        protected double entfernungskm;

        public double Entfernungskm
        {
            get { return entfernungskm; }
            set { entfernungskm = value; }
        }
        protected double gewicht;

        public double Gewicht
        {
            get { return gewicht; }
            set { gewicht = value; }
        }

        public Postprodukt(Produktartwerte pProduktart, double
pEntfernungskm, double pGewicht)
        {
            Produktart=pProduktart;
            Entfernungskm = pEntfernungskm;
            Gewicht = pGewicht;
        }
        public override string ToString()
        {
            return String.Format(„Produktart: {0}, Porto: {1:n} €“
                        , produktart, (this.PortoBerechnen()/100));

        }
        public virtual double PortoBerechnen()
            // Anmerkung: Während in Java jede automatisch als
            // „virtual“ deklariert, muss dies in C# explizit
            // erfolgen.
        {
            return 30 + entfernungskm * 0.05;
        }
```

```
    }

    public class Brief : Postprodukt
    {
        public Brief(double pEntfernungsKm, double pGewicht, Briefformat
pFormat):base(Produktartwerte.Brief,pEntfernungsKm,pGewicht)
        {
            format = pFormat;
        }
        private Briefformat format;

        internal Briefformat Format
        {
            get { return format; }
            set { format = value; }
        }
        public override double PortoBerechnen()
        {
            double porto = base.PortoBerechnen();
            switch (format)
            {
                case Briefformat.C4: porto = porto * 1.4;
                    break;
                case Briefformat.C5: porto = porto * 1.2;
                    break;
                case Briefformat.C6: porto = porto * 1;
                    break;
            }
            if (gewicht > 100) porto = porto * 2;
            return porto;
        }
    }

    public class Paket : Postprodukt
    {
        private Paketformat format;

        public Paketformat Format
        {
            get { return format; }
            set { format = value; }
        }
        private double breite;

        public double Breite
        {
            get { return breite; }
            set { breite = value; }
        }
        private double laenge;

        public double Laenge
        {
            get { return laenge; }
            set { laenge = value; }
        }
        private double hoehe;
```

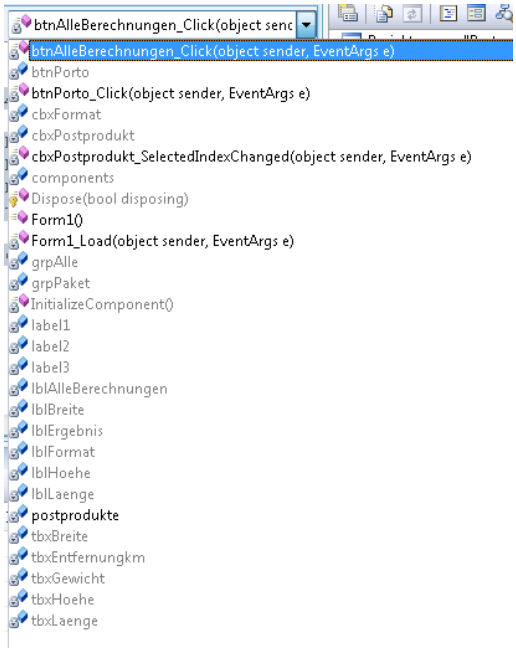
```
        public double Hoehe
        {
            get { return hoehe; }
            set { hoehe = value; }
        }

         public Paket(Paketformat pFormat, double pEntfernungsKm, double
pGewicht, double pLaenge, double pBreite, double
pHoehe):base(Produktartwerte.Paket,pEntfernungsKm,pGewicht)
        {
            format = pFormat;
            laenge = pLaenge;
            breite = pBreite;
            hoehe = pHoehe;
        }
         public double VolumenBerechnen()
         {
             return laenge*breite*hoehe;
         }

         public override double PortoBerechnen()
         {
             double porto=200;
             if (format == Paketformat.Sperrgut) porto = porto * 3;
             int n = (int)(Entfernungskm / 100);
              if (Entfernungskm>(n*100)) {n=n+1;}
             int m = (int)(Gewicht / 1000);
              if (Gewicht>(m*1000)) {m=m+1;}
             porto = porto + n * 50 + m * 30 + 4 * 0.001 *
VolumenBerechnen();
             return porto;
         }
    }
}
```

Zu A3

Die verwendeten Namen für die verwendeten Steuerelemente sind der folgenden Liste zu entnehmen:




```
private void Form1_Load(object sender, EventArgs e)
        {
            cbxPostprodukt.DataSource =
Enum.GetValues(typeof(Produktartwerte));
        }

        private void cbxPostprodukt_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (cbxPostprodukt.SelectedIndex == 0)  // Brief aus gewählt
            {
                cbxFormat.DataSource = Enum.GetValues(typeof(Briefformat));
                grpPaket.Visible = false;

            }
            else
            {
                cbxFormat.DataSource = Enum.GetValues(typeof(Paketformat));
                grpPaket.Visible = true;
            }
            lblErgebnis.Text = „„;
        }

        private void btnPorto_Click(object sender, EventArgs e)
        {


            Brief myBrief;
            Paket myPaket;
                double enfernungskm;
                double.TryParse(tbxEntfernungkm.Text, out enfernungskm);
                double gewicht;
                double.TryParse(tbxGewicht.Text, out gewicht);
                Briefformat format = (Briefformat)cbxFormat.SelectedValue;


                if (cbxPostprodukt.SelectedIndex == 0) //Brief
                {
```

```
                        myBrief = new Brief(enfernungskm, gewicht, format);

                        lblErgebnis.Text = myBrief.ToString();
                        myBrief = null;

                }
                else
                {

                        double laenge;
                        double.TryParse(tbxLaenge.Text, out laenge);
                        double breite;
                        double.TryParse(tbxBreite.Text, out breite);
                        double hoehe;
                        double.TryParse(tbxHoehe.Text, out hoehe);
                        Paketformat pformat =
(Paketformat)cbxFormat.SelectedValue;

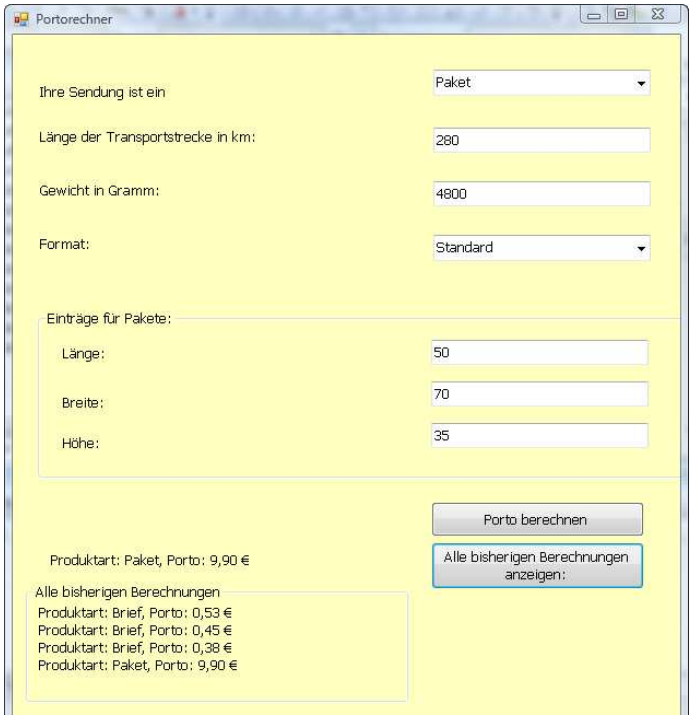
                        myPaket = new Paket(pformat, enfernungskm, gewicht,
laenge, breite, hoehe);

                    lblErgebnis.Text = myPaket.ToString();
                        myPaket = null;

                }
            }


    }
}
```

Zu A4 (Zusatzaufgabe)



- Um Arraylists nutzen zu können, muss der Namensraum System.Collections eingebunden werden.

```
using System.Collections;
```

- Der Label lblAlleBerechnungen wird (nur zum optischen Hervorheben) in eine Groupbox grpAlle eingebettet. Diese steht zu Beginn auf „nicht sichtbar“ (visible=false).

– **Übergeordnet wird die Arraylist deklariert und instanziiert:**

```
private void Form1_Load(object sender, EventArgs e)
{
cbxPostprodukt.DataSource =
Enum.GetValues(typeof(Produktartwerte));
}

ArrayList postprodukte = new ArrayList();
 private void cbxPostpr.....
```

Die Clickroutine des Buttons btnAlle sorgt nun für die Ausgabe im Label:

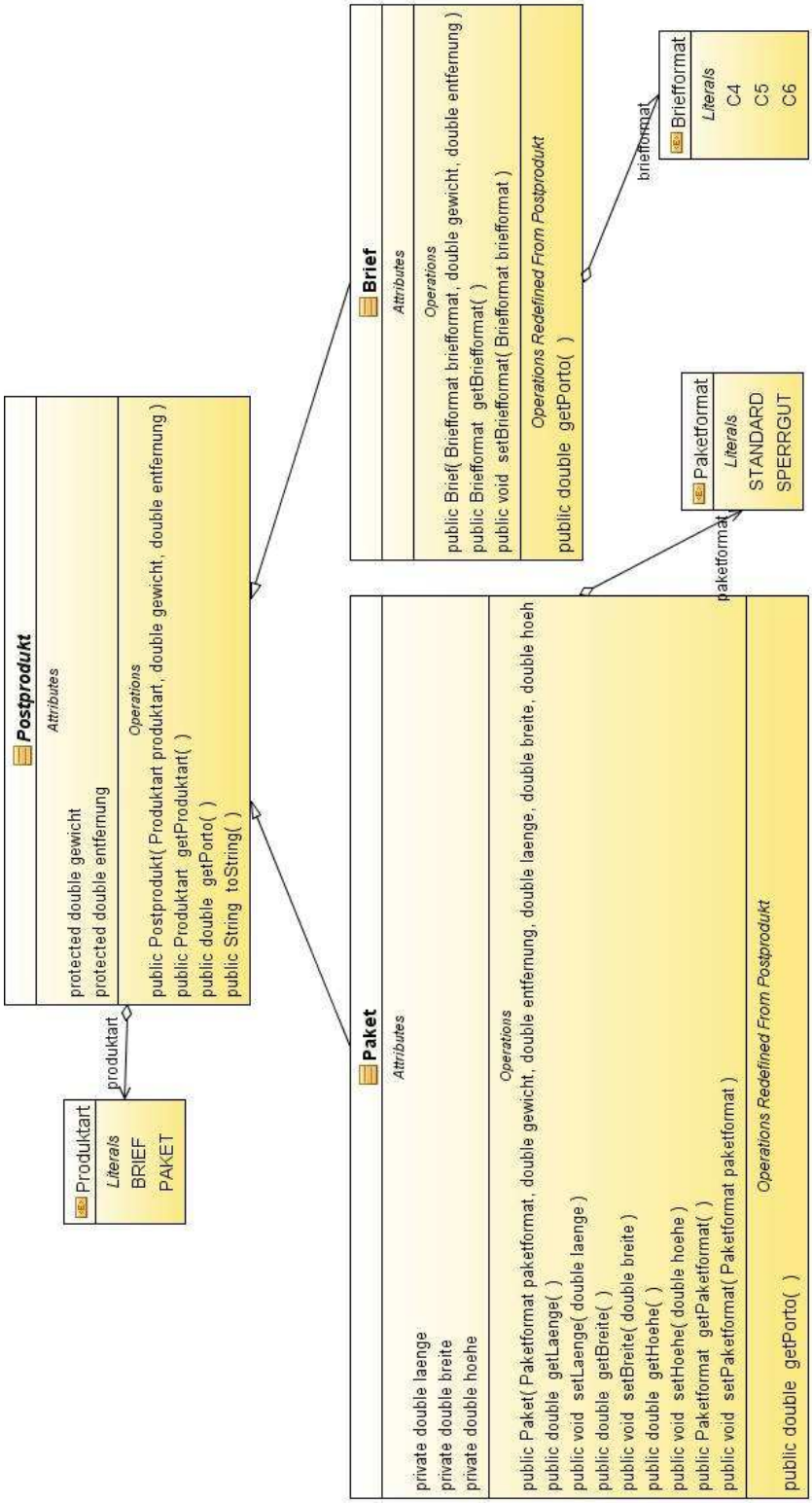
```
private void btnAlleBerechnungen_Click(object sender, EventArgs
e)
        {
            grpAlle.Visible = true;
            lblAlleBerechnungen.Text = "";
            const char LF = (char)10;
            foreach (Postprodukt p in postprodukte)
            {
                lblAlleBerechnungen.Text =
lblAlleBerechnungen.Text+p.ToString() + LF;

            }
        }
```

Wird erneut die Clickroutine des Buttons btnPortoBerechnen aufgerufen, wird die Groupbox grpAlle wieder auf „nicht sichtbar“ umgestellt.

```
    grpAlle.Visible = false;
```

Lösung (für Java/NetBeans)



Quellcode der Portoapplikation

Enumerationen

```
public enum Produktart {BRIEF, PAKET};
public enum Briefformat {C4, C5, C6};
public enum Paketformat {STANDARD, SPERRGUT};
```

Klasse Postprodukt

```
public abstract class Postprodukt {
    private Produktart produktart;
    protected double gewicht;
    protected double entfernung;

    public Postprodukt(Produktart produktart, double gewicht, double ent-
fernung){
      this.produktart = produktart;
      this.gewicht = gewicht;
      this.entfernung = entfernung;
    }

    public Produktart getProduktart(){
      return produktart;
    }

    /* /F10/ Portoberechnung für alle Postprodukte */
    public double getPorto(){
      return 30 + entfernung * 0.05;
    }

   /* /F40 Ausgabe des berechneten Portos */
    @Override
    public String toString(){
        return „Produktart: „ + getProduktart() +
               „, Porto: „ + String.format(„%.2f EUR“, getPorto()/100);
    }
}
```

Klasse Brief erbt von Postprodukt

```
public class Brief extends Postprodukt {
  // Anfang Attribute
private Briefformat briefformat;
  // Ende Attribute

//Konstruktor
public Brief(Briefformat briefformat, double gewicht, double entfernung){
  super(Produktart.BRIEF,gewicht,entfernung);
  this.briefformat = briefformat;
}

  // Anfang Methoden
  public Briefformat getBriefformat() {
    return briefformat;
  }

  public void setBriefformat(Briefformat briefformat) {
    this.briefformat = briefformat;
  }

/* /F20/ Überschriebene Methode Portoberechnung für Briefe */
 @Override public double getPorto() {
      double porto = super.getPorto();
      if (briefformat == Briefformat.C5)
          porto *= 1.2;
      else if (briefformat == Briefformat.C4)
          porto *=  1.4;
      if (gewicht > 100)
          porto *= 2;
      return porto;
  }
  // Ende Methoden
}
```

Klasse Paket erbt von Postprodukt

```
public class Paket extends Postprodukt{
  // Anfang Attribute
  private Paketformat paketformat;
  private double laenge, breite, hoehe;
  // Ende Attribute

public Paket(Paketformat paketformat, double gewicht, double entfernung,
       double laenge, double breite, double hoehe){
super(Produktart.PAKET,gewicht,entfernung);
  this.paketformat = paketformat;
  this.laenge = laenge;
  this.breite = breite;
  this.hoehe = hoehe;
}
    public double getLaenge() {
    return laenge;
  }
  public void setLaenge(double laenge) {
    this.laenge = laenge;
  }

  public double getBreite() {
    return breite;
  }
  public void setBreite(double breite) {
    this.breite = breite;
  }

  public double getHoehe() {
    return hoehe;
  }
  public void setHoehe(double hoehe) {
    this.hoehe = hoehe;
  }

  public Paketformat getPaketformat() {
    return paketformat;
  }
  public void setPaketformat(Paketformat paketformat) {
    this.paketformat = paketformat;
  }
```

```
 @Override public double getPorto() {
      double porto = 200;
      if (paketformat == Paketformat.SPERRGUT)
          porto *= 3;
      porto += Math.ceil(entfernung / 100) * 50 + Math.ceil(gewicht /1000)
* 30;
      porto += 4 * 0.1/100 * laenge*breite*hoehe;
      return porto;
  }
  }
```

GUI-Applikation

```
public class FormPortoapplikation extends javax.swing.JFrame {

    public FormPortoapplikation() {
        initComponents();
    }
…     Ohne Generierung der einzelnen GUI-Elemente
private void formWindowOpened(java.awt.event.WindowEvent evt) {
       this.pnlPortoListe.setVisible(false);
        // Das Panel mit den Textfeldern Länge, Breite und Höhe wird ver-
steckt
        this.pnlVolumen.setVisible(false);
       //Die Einträge der Combobox für die Produktart werden gelöscht
        this.cboProduktart.removeAllItems();
        //und hier mit allen Werten aus der Enumeration Produktart gefüllt
       for (Produktart pf : Produktart.values())
           this.cboProduktart.addItem(pf);


    }

    private   void   cboProduktartActionPerformed(java.awt.event.ActionEvent
evt) {
        this.lblAusgabe.setText(„„);
        //Die Einträge der Combobox für die Formate wird gelöscht
        this.cboFormat.removeAllItems();
        //Je nach Produktart wird die Combobox mit den Formaten gefüllt
        if (this.cboProduktart.getSelectedItem() == Produktart.PAKET){
            // Combobox wird mit Paketformaten gefüllt
            for (Paketformat pf : Paketformat.values())
            this.cboFormat.addItem(pf);
            this.pnlVolumen.setVisible(true);
```

```
        }
        else {
             this.pnlVolumen.setVisible(false);
             //Combobox wird mit BRiefformaten gefüllt
             for (Briefformat pf : Briefformat.values())
             this.cboFormat.addItem(pf);
        }

            // TODO add your handling code here:
    }

    private   void   cmdBerechnenActionPerformed(java.awt.event.ActionEvent
evt) {
    try {
        ausgabezeilen.clear();
        this.pnlPortoListe.setVisible(false);
        //Gewicht einlesen
        double gewicht = Double.parseDouble(txtGewicht.getText());
        //Entfernung einlesen
        double entfernung = Double.parseDouble(txtEntfernung.getText());
        //Auslesen der Produktart aus der Combobox
        Produktart art = (Produktart)this.cboProduktart.getSelectedItem();
        //Eine Referenzvariable für ein Postprodukt deklarieren
        Postprodukt produkt;
        if (art == Produktart.PAKET){
            //Falls die Produktart ein Paket müssen Länge, Breite und Höhe
            // eingelesen werden.
            double laenge = Double.parseDouble(txtLaenge.getText());
            double breite = Double.parseDouble(txtBreite.getText());
            double hoehe = Double.parseDouble(txtHoehe.getText());
            //Auslesen des Paketformats aus der Combobox
            Paketformat           pf           =           (Paketformat)
this.cboFormat.getSelectedItem();
            //Erzeugen des Paketes
            produkt = new Paket(pf,gewicht,entfernung,laenge,breite,hoehe);
        }
        else {
            //Auslesen des Briefformats aus der Combobox
            Briefformat           bf           =           (Briefformat)
this.cboFormat.getSelectedItem();
            //Erzeugen des Briefes
            produkt = new Brief(bf, gewicht, entfernung);
        }
        //  Ausgabe des berechneten Portos in ein Label
        this.lblAusgabe.setText(produkt.toString() );
```

```
        //Einfügebn des berechneten Portos in ein Liste
        portoListe.add(produkt.toString());
        //ausgabezeilen.addElement(portoListe);
        //ausgabezeilen.addElement(produkt.toString());
        //this.lstBerechnungen.setModel(ausgabezeilen);
   }
       catch (Exception e){
       this.lblAusgabe.setText(„Fehlende oder falsche Eingaben!“);
     }
    }

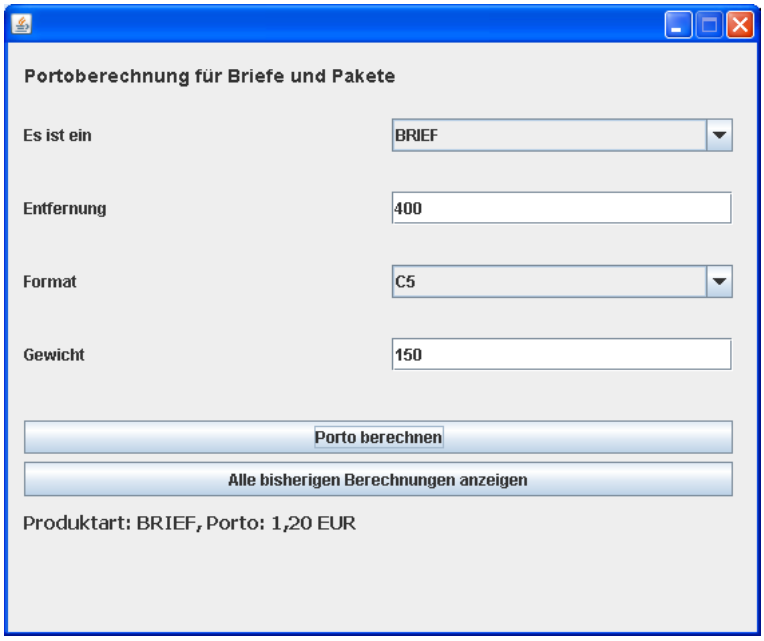
    private  void  cmdListeLöschenActionPerformed(java.awt.event.ActionEvent
evt) {
        // Liste wird gelöscht
        ausgabezeilen.removeAllElements();
        this.lstBerechnungen.setModel(ausgabezeilen);
        portoListe.clear();
    }
    private          void          cmdAlleBerechnungenAnzeigenActionPerfor-
med(java.awt.event.ActionEvent evt) {
        this.pnlPortoListe.setVisible(true);
       ausgabezeilen.removeAllElements();
      for (String text: this.portoListe){
            ausgabezeilen.addElement(text);
       }
       this.lstBerechnungen.setModel(ausgabezeilen);
           }

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FormPortoapplikation().setVisible(true);
            }
        });
    }
```

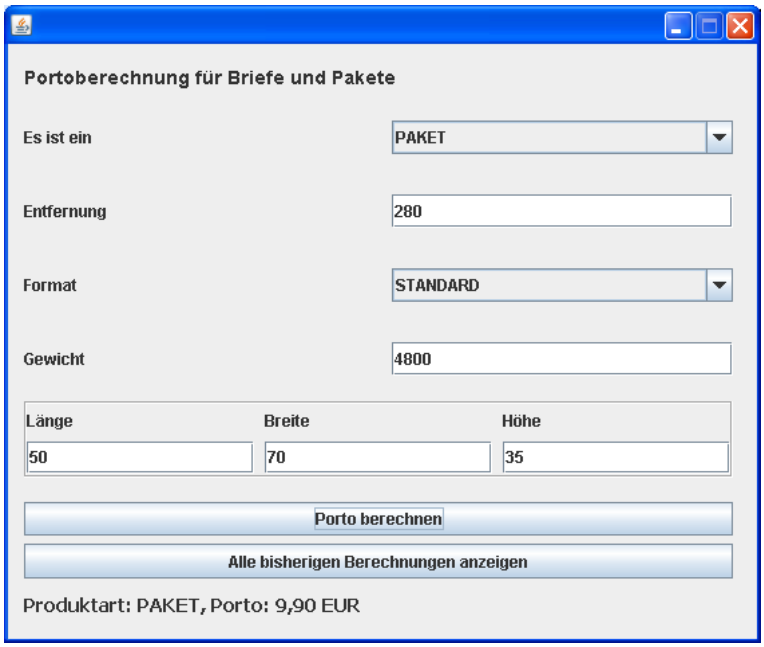
Programmablauf

Je nachdem, ob ein Paket oder Brief ausgewählt wird, wird ein Objekt der Klasse Paket oder Brief instanziiert und das Porto berechnet und ausgegeben. Das Ergebnis wird in einer ArrayList gespeichert.

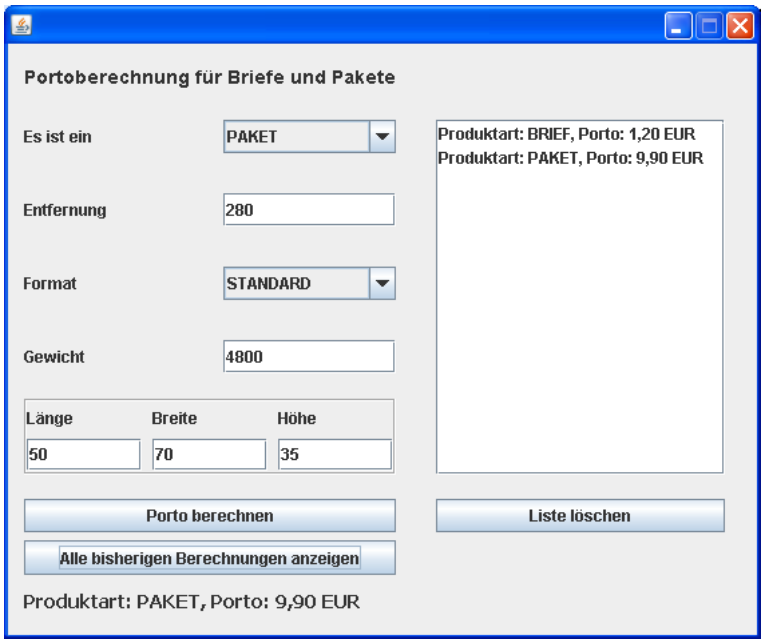
Für einen Brief sieht das Ergebnis folgendermaßen aus:



Für ein Paket sieht das Ergebnis folgendermaßen aus:



Beim Klicken auf die Schaltfläche „AlleBisherigen…“ wird die gefüllte ArrayList in einer jList ausgegeben.



2.4 Jahrgangsstufe 12.2

Beispielaufgabe zu Datenbankentwurf und SQL

Titel	**Videoverleih**
Vorbemerkung	Das folgende Beispiel steht für eine Entwicklungsaufgabe, die einen Datenbankentwurf über die Schritte ER-Modell, Relationenmodell und Nutzung mittels SQL erfordern.
Voraussetzungen gemäß Lehrplan	Datenbankentwurf DQL DML
Anwendungsbezug	Kundenauftrag für eine Videothek in einer Datenbank abbilden
Fachlicher Schwerpunkt	ER-Modellierung und SQL-Programmierung
Eingesetzte Hilfsmittel	DIA zum Zeichnen des ER-Modells
Anmerkungen	

Situation

Als Mitarbeiter der „bkal-Software-consulting“ werden Sie damit beauftragt, für eine Videothek eine Datenbank zu konzipieren. Im Pflichtenheft wurde festgelegt, dass die Datenhaltung mittels einer „relationalen Datenbank“ erfolgen soll. Hierzu steht das relationale Datenbanksytem MySQL zur Verfügung, mit dessen Hilfe die Datenhaltung erfolgen kann.

Im Pflichtenheft wird im Bereich Produktdaten genau beschrieben, welche Daten gespeichert werden müssen, um den Betriebsablauf gewährleisten zu können. Hierzu werden folgende Angaben gemacht:

Produktdaten

/D010/ Kundendaten: Name, Geburtsdatum, Anschrift, Telefonnummer, Geschlecht

/D020/ Film: Titel, Erscheinungsjahr, FSK, Genre, Dauer, Preis

/D030/ Medien: Art (DVD, VHS, BlueRay), Kapazität, Standort

/D040/ Schauspieler: Name

Sachverhalt

Um die einzelnen gespeicherten Daten in einen Zusammenhang zu bringen, erhalten Sie darüber hinaus eine Zusammenfassung des Gesprächs, das mit dem Eigentümer der Videothek geführt wurde:

„Die Kundendaten benötigen wir, um den Videoverleih nachvollziehen zu können. Die Filme befinden sich u. U. auf verschiedenen Medien (z. B. DVD oder BlueRay-Disc), um den unterschiedlichen Wünschen der Kunden gerecht zu werden. Damit

Kunden die Möglichkeit haben, Filme nicht nur über den Titel, sondern auch über die mitspielenden Schauspieler suchen zu können, werden auch diese Daten gespeichert.“

Arbeitsaufträge

A1 Bearbeiten Sie zunächst die folgenden Teilaufgaben, indem Sie im Internet auf der Seite *http://www.tinohempel.de/info/info/datenbank/index.htm* recherchieren.

A1.1 Nennen Sie die Bestandteile eines relationalen Datenbanksystems.
A1.2 Erklären Sie, welche Funktionen diese Bestandteile übernehmen.
A1.3 Erläutern Sie die Anforderungen an ein relationales Datenbanksystem.

A2 Modellieren Sie ein ER-Modell für die Videothek, das die oben beschriebenen Vorgaben korrekt abbildet.

A3 Erstellen Sie zu dem ER-Modell der Videothek ein relationales Datenbankmodell.

A4 Implementieren Sie das relationale Datenbankmodell unter MySQL mit Hilfe eines passenden Tools unter dem Namen „Videothek“.

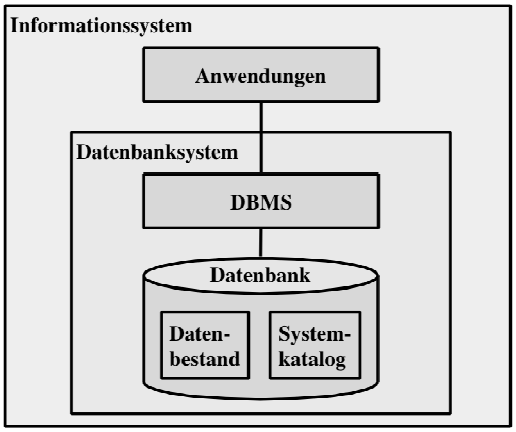
A5 Implementieren Sie auf der Basis des Relationenmodells und ihrer erstellten Datenbank für folgende Aufträge die SQL-Statements.

A5.1 Der Kunde Sebastian Krumbein, wohnhaft Ölweg 23, in 40123 Düsseldorf wird als neuer Kunde erfasst.
A5.2 Der Kunde Harald Schmidt ist in den Mühlenweg 12, in 50432 Bergheim gezogen.
A5.3 Der Film Crazy wird aufgrund mangelnder Nachfrage aus dem Sortiment genommen.
A5.4 Suchen Sie alle Kunden (Nachname, Vorname, Geburtsdatum) absteigend sortiert, deren Nachname mit B oder K beginnt.
A5.5 Suchen Sie alle Filme (Titel, FSK, Genre, Erscheinungsjahr) mit FSK 12 und Genre Drama.
A5.6 Suchen Sie alle Medien (Mediennr, MedienArt) mit den entsprechenden Filmen (FilmNr, Titel) die als VHS-Kassette vorrätig sind.
A5.7 Suchen Sie alle Filme (FilmNr, FSK, Titel), in denen Ove Sproge mitgespielt hat. In der Ergebnisanzeige soll die Spalte Titel mit „Filmtitel“ und die Spalte FSK mit „frei ab“ überschrieben sein.
A5.8 Geben Sie die Anzahl der Kunden aufgegliedert nach Geschlecht aus.
A5.9 Geben Sie eine Liste der Schauspieler aus, die in mehr als 2 Filmen mitgespielt haben.

Lösung zu A1

A1.1

Das Datenbanksystem besteht aus einem Datenspeicher (Datenbank), dem Data Dictionary und einem Datenbankmanagementsystem (DBMS).



Modell eines Datenbanksystems

A1.2

Die Aufgabe des Datenspeichers besteht in der physikalischen Datenspeicherung und der Bereitstellung eines Systemkatalogs (Data Dictionary), in dem alle Informationen zur physischen Speicherung der Daten und dem logischen Aufbau der Datensammlung enthalten sind.

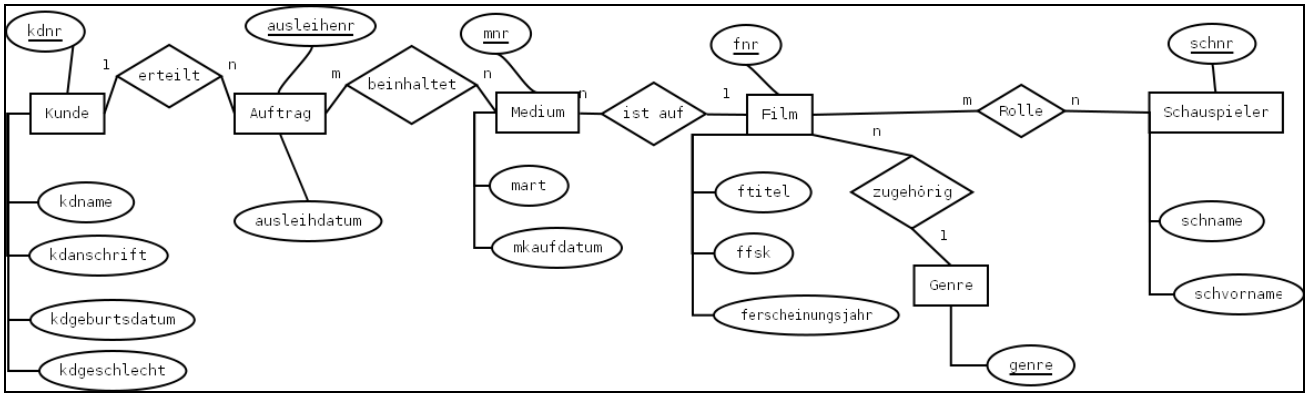
Das Datenbankmanagementsystem beinhaltet die Funktionen zur Erzeugung, Verwaltung und Manipulation des Datenbestandes. Es bietet Schnittstellen, mit deren Hilfe auf den Datenbestand zugegriffen werden kann. Ziel ist die Realisierung logischer und physischer Datenunabhängigkeit.

A1.3

Integritätssicherung:	Daten werden auf Korrektheit (bereits während der Eingabe) überprüft und Anomalien verhindert.
Redundanzarmut:	Es gibt keine ungeordnete Mehrfachspeicherung von Datenwerten.
Datensicherheit:	Ungewollter Datenverlust wird durch interne Backup- und Prüfmechanismen verhindert.
Datenschutz:	Zugriffskontrolle und spezifische Sichten sorgen für einen Zugang gemäß der Rechte des Nutzers.
Mehrbenutzerbetrieb:	Viele Benutzer können parallel auf der Datenbank arbeiten.
Datenunabhängigkeit:	Das DBMS ist nicht an die Daten der Datenbank gekoppelt, es kann unabhängig von den Daten weiterentwickelt werden.
Zentrale Kontrolle:	Ein Administrator ist in der Lage, das gesamte System von einem Rechner aus zu verwalten.

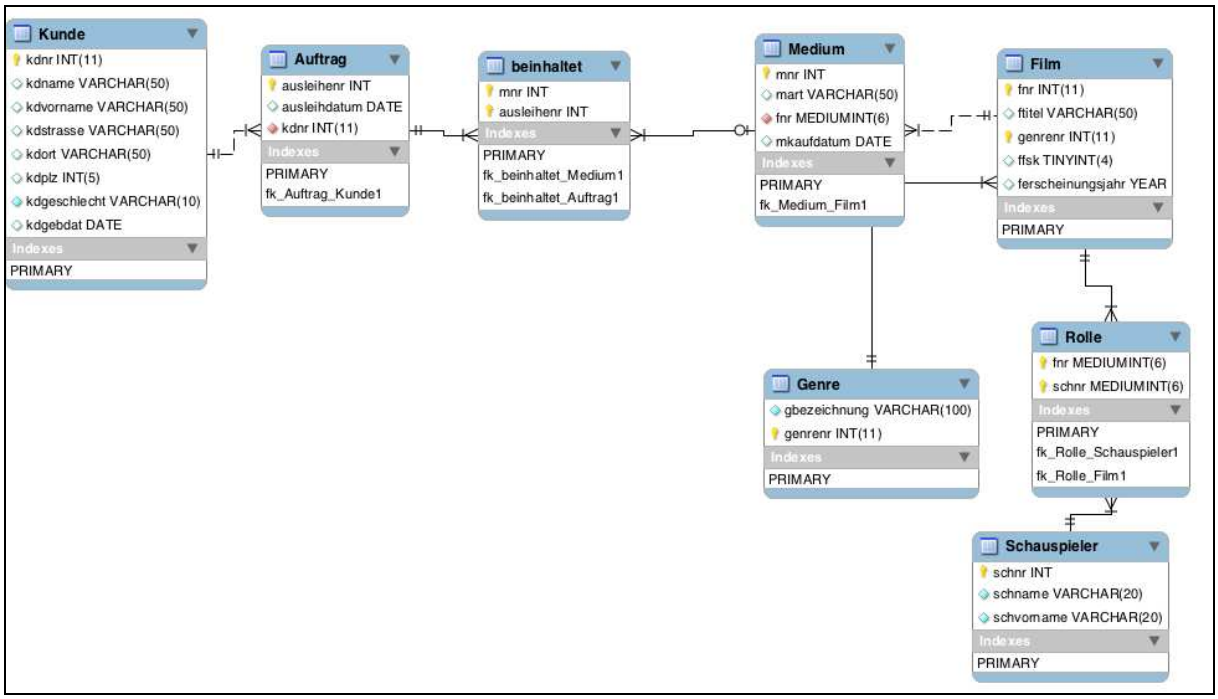
Quelle: *http://www.tinohempel.de/info/info/datenbank/index.htm*

Lösung zu A2



ERM zur Videothek

Lösung zu A3



Modell der Datenbank mittels „MySQL workbench“ erzeugt

Skript zur Erzeugung der Datenbank:

```
-- phpMyAdmin SQL Dump
-- version 2.11.7
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Erstellungszeit: 22. Juni 2010 um 08:44
-- Server Version: 5.0.51
-- PHP-Version: 4.4.8

SET SQL_MODE=„NO_AUTO_VALUE_ON_ZERO“;

--
-- Datenbank: ‘videothek’
--

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Auftrag’
```

```
--

CREATE TABLE IF NOT EXISTS ‘Auftrag’ (
  ‘ausleihenr’ int(11) NOT NULL auto_increment,
  ‘ausleihdatum’ date default NULL,
  ‘kdnr’ int(11) NOT NULL,
  PRIMARY KEY  (‘ausleihenr’),
  KEY ‘fk_Auftrag_Kunde1’ (‘kdnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=47 ;

--
-- Daten für Tabelle ‘Auftrag’
--

INSERT INTO ‘Auftrag’ (‘ausleihenr’, ‘ausleihdatum’, ‘kdnr’) VALUES
(1, '2008-10-26', 3334),
(2, '2008-11-15', 3669),
(3, '2007-10-24', 1202),
(4, '2007-06-14', 1210),
(5, '2007-12-16', 1224),
(6, '2008-01-26', 3269),
(7, '2008-03-23', 1303),
(8, '2008-07-17', 1300),
(9, '2008-08-26', 1210),
(10, '2008-05-02', 1301),
(11, '2008-04-27', 3843),
(12, '2008-06-24', 3200),
(13, '2008-02-27', 1233),
(14, '2008-11-17', 1225),
(15, '2008-09-23', 3270),
(16, '2008-06-14', 3474),
(17, '2007-12-23', 3690),
(18, '2008-10-17', 3720),
(19, '2008-04-17', 3041),
(20, '2008-03-06', 1318),
(21, '2008-10-21', 1202),
(22, '2009-01-24', 4008),
(24, '2007-11-25', 3622),
(25, '2008-03-27', 1215),
(26, '2008-08-28', 1224),
(27, '2007-02-24', 1303),
(28, '2008-04-27', 3532),
(29, '2008-02-26', 3271),
(30, '2008-05-14', 3980),
(31, '2007-07-03', 1225),
(32, '2008-03-19', 3222),
(33, '2008-06-17', 3221),
(34, '2007-02-27', 1202),
(35, '2008-03-07', 1303),
(38, '2008-09-03', 3624),
(39, '2007-12-07', 3269),
(40, '2007-12-13', 1233),
(43, '2012-04-01', 4012),
(44, '2012-04-02', 1215),
(45, '2012-02-02', 1225),
(46, '2012-02-23', 4013);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘beinhaltet’
--

CREATE TABLE IF NOT EXISTS ‘beinhaltet’ (
  ‘mnr’ int(11) NOT NULL,
  ‘ausleihenr’ int(11) NOT NULL,
  PRIMARY KEY  (‘mnr’,’ausleihenr’),
  KEY ‘fk_beinhaltet_Medium1’ (‘mnr’),
  KEY ‘fk_beinhaltet_Auftrag1’ (‘ausleihenr’)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

--
-- Daten für Tabelle ‘beinhaltet’
--

INSERT INTO ‘beinhaltet’ (‘mnr’, ‘ausleihenr’) VALUES
(3, 1),
(4, 17),
(6, 16),
(7, 4),
(8, 2),
(8, 18),
(8, 43),
(10, 19),
```

```
(11, 6),
(11, 20),
(11, 24),
(11, 39),
(12, 35),
(12, 45),
(13, 3),
(14, 10),
(16, 8),
(17, 29),
(18, 12),
(18, 13),
(21, 15),
(22, 31),
(23, 7),
(24, 5),
(24, 11),
(24, 14),
(24, 27),
(26, 9),
(32, 28),
(33, 34),
(35, 32),
(38, 38),
(50, 22),
(51, 21),
(59, 33),
(61, 30),
(66, 26),
(77, 40),
(83, 46),
(85, 44),
(92, 25);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Film’
--

CREATE TABLE IF NOT EXISTS ‘Film’ (
  ‘fnr’ int(11) NOT NULL auto_increment,
  ‘ftitel’ varchar(50) default NULL,
  ‘genrenr’ int(11) default NULL,
  ‘ffsk’ tinyint(4) default NULL,
  ‘erscheinungsjahr’ year(4) default NULL,
  PRIMARY KEY  (‘fnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=56 ;

--
-- Daten für Tabelle ‘Film’
--

INSERT INTO ‘Film’ (‘fnr’, ‘ftitel’, ‘genrenr’, ‘ffsk’, ‘erscheinungsjahr’) VALUES
(1, 'Monty Pythons - Das Leben des Brian', 13, 16, 1979),
(12, 'Das Schweigen der Lämmer', 21, 16, 1990),
(2, 'Braveheart', 1, 16, 1994),
(3, 'Butterfly Effect', 16, 16, 2004),
(4, 'Cliffhanger - Nur die Starken überleben', 7, 18, 1993),
(6, 'Das geheime Fenster', 11, 18, 2004),
(7, 'Das Leben des David Gale', 8, 12, 2002),
(8, 'Der Pianist', 8, 12, 2002),
(9, 'Der Staatsfeind Nr.1', 21, 12, 1998),
(10, 'Der Untergang', 8, 12, 2004),
(11, 'Die Olsenbande f?§hrt nach J?ºtland', 13, 6, 1971),
(13, 'Die Olsenbande: Der (voraussichtlich) letzte Strei', 13, 6, 1974),
(14, 'Dirty Dancing', 20, 12, 1987),
(15, 'Eiskalte Engel', 8, 12, 1998),
(16, 'Es war einmal in Amerika', 4, 16, 1982),
(17, 'Extreme Rage - A Man apart', 21, 16, 2002),
(30, 'Pearl Harbor', 8, 12, 2001),
(31, 'Pearl Harbour', 2, 18, 1991),
(33, 'Richi Rich', 13, 6, 1994),
(34, '2 Fast 2 Forious', 2, 12, 2003),
(35, 'Riddick', 18, 16, 2004),
(36, 'Save the last Dance', 15, 12, 2000),
(37, 'Terminator', 2, 16, 1988),
(38, 'Terminator 2', 2, 16, 1990),
(39, 'Terminator 3', 2, 16, 2003),
(40, 'The beach', 1, 16, 1999),
(41, 'The Boondock Saints', 2, 12, 1999),
(42, 'The Fast and the Furious', 2, 16, 2001),
(43, 'The Game', 21, 12, 1997),
(45, 'Armageddon', 18, 12, 1998),
```

```
(46, 'The Ring', 11, 16, 2002),
(47, 'The Sixth Sense', 16, 16, 1999),
(48, 'The Village', 21, 12, 2004),
(32, 'Resident Evil', 5, 18, 2002),
(18, 'Final Destination', 11, 18, 2000),
(19, 'From Dusk Till Dawn', 10, 18, 1995),
(20, 'Gladiator', 6, 12, 2000),
(21, 'Good Will Hunting', 8, 12, 1997),
(22, 'Grease', 14, 6, 1978),
(23, 'Dumm und d?ºmmer', 13, 12, 1994),
(24, 'Hip Hop Hood', 13, 12, 1996),
(25, 'Honey', 8, 0, 2003),
(26, 'So High', 13, 16, 2001),
(27, 'Lammbock', 13, 16, 2001),
(28, 'Last Samurai', 8, 16, 2003),
(29, 'Nicht auflegen!', 9, 16, 2002),
(49, 'Titanic', 8, 16, 1997),
(50, 'Troja', 3, 12, 2004),
(51, 'BAD TASTE', 19, 18, 1987),
(52, 'Biker Boyz', 6, 12, 2003),
(53, 'Blade 2', 12, 18, 2002),
(54, 'Braindead', 19, 18, 1991),
(44, 'The Presidental Election', 17, 16, 2000),
(55, 'Rio Bravo', 22, 6, 1959);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Genre’
--

CREATE TABLE IF NOT EXISTS ‘Genre’ (
  ‘gbezeichnung’ varchar(100) character set utf8 collate utf8_bin NOT NULL,
  ‘genrenr’ int(11) NOT NULL auto_increment,
  PRIMARY KEY  USING BTREE (‘genrenr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=23 ;

--
-- Daten für Tabelle ‘Genre’
--

INSERT INTO ‘Genre’ (‘gbezeichnung’, ‘genrenr’) VALUES
('Abenteuer Epos', 1),
('Action', 2),
('Action-lovestory', 3),
('Action-thriller', 4),
('Action/Science-fiction', 5),
('Actiondrama', 6),
('Bergsteigerdrama', 7),
('Drama', 8),
('Drama/Thriller', 9),
('Horror', 10),
('Horror- Thriller', 11),
('Horror/Action', 12),
('KomÔæödie', 13),
('Musical', 14),
('Musikfilm/ USA 2000', 15),
('Mystery- Thriller', 16),
('Report', 17),
('Science-Fiction', 18),
('Splatter', 19),
('Tanzfilm', 20),
('Thriller', 21),
('Western', 22);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Kunde’
--

CREATE TABLE IF NOT EXISTS ‘Kunde’ (
  ‘kdnr’ int(11) NOT NULL auto_increment,
  ‘kdname’ varchar(50) default NULL,
  ‘kdvorname’ varchar(50) default NULL,
  ‘kdstrasse’ varchar(50) default NULL,
  ‘kdort’ varchar(50) default NULL,
  ‘kdplz’ char(5) default NULL,
  ‘kdgeschlecht’ varchar(10) NOT NULL,
  ‘kdgebdat’ date default NULL,
  PRIMARY KEY  (‘kdnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=4015 ;

--
```

```
-- Daten für Tabelle ‘Kunde’
--

INSERT INTO ‘Kunde’ (‘kdnr’, ‘kdname’, ‘kdvorname’, ‘kdstrasse’, ‘kdort’, ‘kdplz’,
‘kdgeschlecht’, ‘kdgebdat’) VALUES
(1001, 'Pflaume', 'Reiner', 'Pallasstr. 5', 'Köln', '50345', 'männlich', '1980-07-24'),
(1202, 'Dietz', 'Anja', 'Kalkscheunenstr. 22', 'Köln', '50935', 'weiblich', '1983-06-30'),
(1210, 'Dorfmann', 'Yvonne', 'Magazinstr. 12', 'Oldenburg', '26133', 'weiblich', '1981-08-
07'),
(1213, 'Krokowski', 'Timo', 'Wilsnacker Str. 61', 'Köln', '50858', 'männlich', '1980-03-23'),
(1215, 'Hristov', 'Karsten', 'Am Fischtal 62', 'Köln', '50876', 'männlich', '1955-12-12'),
(1218, 'Hellwig', 'Nathan', 'Scholtz Allee 62', 'Bonn', '53125', 'männlich', '1983-01-11'),
(1224, 'Pucciarelli', 'Piero', 'Prinzenstr. 25', 'Berlin', '10559', 'männlich', '1983-12-04'),
(1225, 'Weber', 'Falco', 'Heilbronner Str. 27', 'Bergheim', '50129', 'männlich', '1983-07-
08'),
(1228, 'Prokurat', 'Sonja', 'Siemensstr. 1', 'Köln', '50858', 'weiblich', '1982-07-09'),
(1229, 'Luft', 'Isabel', 'Takustr. 4', 'Köln', '50345', 'weiblich', '1960-04-08'),
(1231, 'Khalil', 'Hanna', 'Apostel-Paulus-Str. 12', 'Frechen', '50226', 'weiblich', '1983-04-
30'),
(1232, 'Toblach', 'Pinar', 'Mansteinstr. 14', 'Köln', '50858', 'weiblich', '1982-08-07'),
(1233, 'Kalle', 'Horst', 'Reeperbahn 12', 'Hamburg', '20359', 'männlich', '1966-12-09'),
(1300, 'Ruff', 'Fabian', 'Gotzkowskystr. 100', 'Platsdorf', '12329', 'männlich', '1981-03-
23'),
(1301, 'Gout', 'Julien', 'Motzstr. 53', 'Köln', '50859', 'männlich', '1983-01-30'),
(1303, 'Kidal', 'Sevcan', 'Alvenslebenstr. 33', 'Bergheim', '50123', 'weiblich', '1982-12-
03'),
(1318, 'Knipper', 'Volker', 'Richthofen-Str. 10', 'Köln', '50987', 'männlich', '1978-12-23'),
(2944, 'Flantz', 'Jaromir', 'Cordes Allee 45', 'Kerpen', '50171', 'männlich', '1982-12-15'),
(2945, 'He', 'Goran', 'Eberth Strasse 80', 'Berlin', '11535', 'männlich', '1996-10-06'),
(3041, 'Speider', 'Jacques', 'Junck Strasse 8', 'Frechen', '50226', 'männlich', '1977-02-14'),
(3200, 'Kock', 'Xenia', 'Haase Allee 53', 'Bergheim', '50129', 'weiblich', '1987-04-22'),
(3220, 'Junken', 'Hektor', 'Eigenwillig Gasse 3', 'Hheim', '19007', 'männlich', '1986-01-15'),
(3221, 'Dobes', 'Eleonora', 'Zhao Gasse 44', 'Köln', '50234', 'weiblich', '1978-12-30'),
(3222, 'Silbereisen', 'Therese', 'Sager Allee 86', 'Berlin', '10497', 'weiblich', '1970-11-
01'),
(3269, 'Becker', 'Muriel', 'Everts Weg 48', 'Frechen', '50226', 'weiblich', '1992-07-01'),
(3270, 'Unfried', 'Nico', 'Hettner Allee 44', 'Köln', '50987', 'männlich', '1961-04-24'),
(3271, 'Jättner', 'Sabine', 'Cramer Allee 62', 'Bergheim', '50129', 'weiblich', '1984-12-05'),
(3334, 'Girschner', 'Damian', 'Schmidt Allee 65', 'Bergheim', '50126', 'männlich', '1979-03-
20'),
(3335, 'Stiebitz', 'Otmar', 'Franze Str. 66', 'Eimersdorf', '18847', 'männlich', '1972-11-
11'),
(3336, 'Pruschke', 'Uwe', 'Hühnerbein - Strasse 19', 'Berlin', '10334', 'männlich', '1978-07-
19'),
(3452, 'Friesemann-Kraus', 'Emanuela', 'Laubs Acker 13', 'Frechen', '50226', 'weiblich',
'1982-01-14'),
(3474, 'Bruder', 'Walburga', 'Schmiedt Acker 37', 'Köln', '50935', 'weiblich', '1976-08-29'),
(3532, 'Schomber', 'Daniela', 'Khalil Str. 45', 'Duisburg', '47055', 'weiblich', '1967-06-
24'),
(3622, 'Wachtendorf', 'Yara', 'Meister Weg 57', 'Kerpen', '50171', 'weiblich', '1980-05-17'),
(3623, 'Sonnenburg', 'Rabea', 'Al-Bagikni Gasse 13', 'Bergheim', '50126', 'weiblich', '1985-
12-26'),
(3624, 'Wende', 'Gloria', 'Vater - Strasse 10', 'Köln', '50858', 'weiblich', '1967-07-16'),
(3669, 'Ullmann', 'Angelika', 'Santosa Str. 69', 'Bergheim', '50129', 'weiblich', '1979-06-
24'),
(3689, 'Jonson', 'Dominik', 'Rohrbach Platz 92', 'Frechen', '50226', 'männlich', '1981-06-
04'),
(3690, 'Lindau', 'Oskar', 'Seidensticker Platz 44', 'Bergheim', '50129', 'männlich', '1984-04-
10'),
(3720, 'von Aswege', 'Carol', 'Nordburg Str. 24', 'Kerpen', '50170', 'männlich', '1978-08-
20'),
(3760, 'Santosa', 'Orson', 'Armbruster Strasse 36', 'Eimersdorf', '16840', 'männlich', '1969-
09-24'),
(3841, 'Maelzer', 'Irene', 'Seifert Strasse 10', 'Kerpen', '50170', 'weiblich', '1983-01-09'),
(3842, 'Becker', 'Lester', 'Jancauskaite Str. 25', 'Frechen', '50226', 'weiblich', '1988-06-
04'),
(3843, 'Kok', 'Lydia', 'Dowerg Allee 27', 'Bergheim', '50129', 'weiblich', '1972-01-28'),
(3844, 'Thanel', 'Dominik', 'Silbweisen Platz 4', 'Arnoldstadt', '18289', 'männlich', '1980-
08-17'),
(3845, 'Wu', 'Casimir', 'von Aswege Strasse 50', 'Kluthbach', '12661', 'männlich', '1976-09-
18'),
(3938, 'Schalkenhauer', 'Wilfried', 'Han - Strasse 69', 'Bonn', '52175', 'männlich', '1976-06-
17'),
(3959, 'Everts', 'Jan', 'Jung Allee 5', 'Frechen', '50226', 'männlich', '1968-04-24'),
(3980, 'Bolnbach', 'Claus', 'Käster Str. 20', 'Bergheim', '50128', 'männlich', '1963-10-22'),
(4008, 'Hummelheber', 'Wolfgang', 'Juncken Allee 99', 'Frechen', '50226', 'männlich', '1979-
11-13'),
(4009, 'Maffay', 'Peter', 'Caucescu-Allee 20', 'Banat', '11111', 'männlich', '1940-08-18'),
(4010, 'Müller', 'Katja', 'Kiefernweg 15', 'Treuenbrietzen', '14929', 'weiblich', '1972-11-
15'),
(4011, 'Müller', 'Hans', 'Froschweg 88', 'Bergheim', '50129', 'männlich', '1978-08-27'),
(4012, 'Schmidt', 'Harald', 'Mühlenweg 12', 'Bergheim', '51432', 'männlich', '1955-06-15'),
(4013, 'Schmidt', 'Hannelore', 'Bergstr. 16', 'Kerpen', '50171', 'männlich', '1956-06-16'),
(4014, 'Krumbein', 'Sebastian', 'Ölweg 23', 'Düsseldorf', '40123', 'männlich', '1983-10-27');
```

```
-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Medium’
--

CREATE TABLE IF NOT EXISTS ‘Medium’ (
  ‘mnr’ int(11) NOT NULL auto_increment,
  ‘mart’ varchar(50) character set utf8 collate utf8_bin default NULL,
  ‘fnr’ mediumint(6) NOT NULL,
  ‘mkaufdatum’ date default NULL,
  PRIMARY KEY  (‘mnr’),
  KEY ‘fk_Medium_Film1’ (‘fnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COMMENT='Datentraeger uebersicht' AUTO_INCREMENT=101 ;

--
-- Daten für Tabelle ‘Medium’
--

INSERT INTO ‘Medium’ (‘mnr’, ‘mart’, ‘fnr’, ‘mkaufdatum’) VALUES
(86, 'blu ray', 50, NULL),
(82, 'blu ray', 49, NULL),
(81, 'blu ray', 48, NULL),
(80, 'blu ray', 48, NULL),
(85, 'blu ray', 49, NULL),
(84, 'blu ray', 51, NULL),
(83, 'blu ray', 50, NULL),
(79, 'blu ray', 47, NULL),
(75, 'blu ray', 45, NULL),
(74, 'blu ray', 44, NULL),
(73, 'blu ray', 43, NULL),
(78, 'blu ray', 46, NULL),
(77, 'blu ray', 45, NULL),
(76, 'blu ray', 44, NULL),
(100, 'blu ray', 49, NULL),
(96, 'blu ray', 25, NULL),
(95, 'blu ray', 12, NULL),
(94, 'blu ray', 54, NULL),
(99, 'blu ray', 3, NULL),
(98, 'blu ray', 10, NULL),
(97, 'blu ray', 1, NULL),
(93, 'blu ray', 54, NULL),
(89, 'blu ray', 53, NULL),
(88, 'blu ray', 52, NULL),
(87, 'blu ray', 51, NULL),
(92, 'blu ray', 53, NULL),
(91, 'blu ray', 53, NULL),
(90, 'blu ray', 47, NULL),
(3, 'blu ray', 54, NULL),
(25, 'DVD', 15, NULL),
(61, 'DVD', 35, NULL),
(60, 'DVD', 34, NULL),
(16, 'DVD', 10, NULL),
(15, 'DVD', 8, NULL),
(62, 'DVD', 36, NULL),
(12, 'DVD', 7, NULL),
(11, 'DVD', 6, NULL),
(53, 'DVD', 31, NULL),
(59, 'DVD', 33, NULL),
(58, 'DVD', 32, NULL),
(13, 'DVD', 7, NULL),
(26, 'DVD', 16, NULL),
(71, 'DVD', 42, NULL),
(22, 'DVD', 13, NULL),
(23, 'DVD', 13, NULL),
(24, 'DVD', 14, NULL),
(67, 'DVD', 38, NULL),
(18, 'DVD', 11, NULL),
(17, 'DVD', 9, NULL),
(70, 'DVD', 41, NULL),
(69, 'DVD', 40, NULL),
(68, 'DVD', 39, NULL),
(40, 'DVD', 24, NULL),
(5, 'DVD', 2, NULL),
(32, 'DVD', 19, NULL),
(43, 'DVD', 25, NULL),
(42, 'DVD', 25, NULL),
(41, 'DVD', 24, NULL),
(31, 'DVD', 19, NULL),
(2, 'DVD', 5, NULL),
(35, 'DVD', 21, NULL),
(34, 'DVD', 20, NULL),
(52, 'DVD', 30, NULL),
```

```
(49, 'DVD', 28, NULL),
(10, 'DVD', 5, NULL),
(51, 'DVD', 30, NULL),
(50, 'DVD', 29, NULL),
(33, 'DVD', 20, NULL),
(7, 'DVD', 3, NULL),
(8, 'DVD', 4, NULL),
(44, 'DVD', 26, NULL),
(21, 'VHS', 12, NULL),
(1, 'VHS', 1, NULL),
(6, 'VHS', 3, NULL),
(4, 'VHS', 53, NULL),
(9, 'VHS', 5, NULL),
(20, 'VHS', 12, NULL),
(19, 'VHS', 11, NULL),
(14, 'VHS', 8, NULL),
(46, 'VHS', 24, NULL),
(45, 'VHS', 25, NULL),
(36, 'VHS', 20, NULL),
(66, 'VHS', 39, NULL),
(48, 'VHS', 27, NULL),
(47, 'VHS', 26, NULL),
(30, 'VHS', 18, NULL),
(29, 'VHS', 17, NULL),
(39, 'VHS', 23, NULL),
(38, 'VHS', 22, NULL),
(37, 'VHS', 21, NULL),
(65, 'VHS', 38, NULL),
(56, 'VHS', 31, NULL),
(27, 'VHS', 16, NULL),
(28, 'VHS', 17, NULL),
(72, 'VHS', 42, NULL),
(55, 'VHS', 30, NULL),
(54, 'VHS', 29, NULL),
(64, 'VHS', 37, NULL),
(63, 'VHS', 36, NULL),
(57, 'VHS', 32, NULL);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Rolle’
--

CREATE TABLE IF NOT EXISTS ‘Rolle’ (
  ‘fnr’ mediumint(6) NOT NULL,
  ‘schnr’ mediumint(6) NOT NULL,
  PRIMARY KEY  (‘fnr’,’schnr’),
  KEY ‘fk_Rolle_Schauspieler1’ (‘schnr’),
  KEY ‘fk_Rolle_Film1’ (‘fnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8;

--
-- Daten für Tabelle ‘Rolle’
--

INSERT INTO ‘Rolle’ (‘fnr’, ‘schnr’) VALUES
(1, 23),
(1, 52),
(2, 22),
(3, 26),
(4, 43),
(5, 36),
(5, 42),
(6, 40),
(7, 40),
(8, 5),
(9, 38),
(10, 21),
(11, 41),
(12, 20),
(13, 41),
(14, 44),
(15, 30),
(15, 32),
(15, 50),
(16, 13),
(17, 16),
(18, 27),
(19, 25),
(19, 28),
(19, 45),
(20, 8),
(21, 12),
```

```
(22, 46),
(23, 6),
(24, 49),
(25, 2),
(26, 29),
(27, 4),
(28, 4),
(29, 18),
(30, 1),
(30, 53),
(30, 54),
(31, 45),
(32, 24),
(33, 10),
(34, 47),
(35, 16),
(37, 37),
(38, 37),
(39, 37),
(40, 15),
(41, 11),
(42, 16),
(43, 17),
(44, 7),
(45, 51),
(46, 48),
(47, 31),
(48, 33),
(49, 15),
(49, 55),
(50, 34),
(51, 35),
(52, 19),
(53, 39),
(54, 3),
(55, 56);

-- --------------------------------------------------------

--
-- Tabellenstruktur für Tabelle ‘Schauspieler’
--

CREATE TABLE IF NOT EXISTS ‘Schauspieler’ (
  ‘schnr’ int(11) NOT NULL auto_increment,
  ‘schname’ varchar(20) collate utf8_bin NOT NULL,
  ‘schvorname’ varchar(20) collate utf8_bin NOT NULL,
  PRIMARY KEY  (‘schnr’)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=57 ;

--
-- Daten für Tabelle ‘Schauspieler’
--

INSERT INTO ‘Schauspieler’ (‘schnr’, ‘schname’, ‘schvorname’) VALUES
(1, 'Affleck', 'Ben'),
(2, 'Alba', 'Jessica'),
(3, 'Balme', 'Timothy'),
(4, 'Bleibtreu', 'Moritz'),
(5, 'Brody', 'Adrien'),
(6, 'Carrey', 'Jim'),
(7, 'Chiney', 'William'),
(8, 'Crowe', 'Russel'),
(9, 'Cruise', 'Tom'),
(10, 'Culkin', 'Macaulay'),
(11, 'Dafoe', 'Willem'),
(12, 'Damon', 'Matt'),
(13, 'deNiro', 'Robert'),
(14, 'Depp', 'Johnny'),
(15, 'Di Caprio', 'Leonardo'),
(16, 'Diesel', 'Vin'),
(17, 'Douglas', 'Michael'),
(18, 'Farrell', 'Colin'),
(19, 'Fishburne', 'Laurence'),
(20, 'Foster', 'Jodie'),
(21, 'Ganz', 'Bruno'),
(22, 'Gibson', 'Mel'),
(23, 'Idle', 'Eric'),
(24, 'Jovovich', 'Milla'),
(25, 'Keitel', 'Harvey'),
(26, 'Kutcher', 'Ashton'),
(27, 'Larter', 'Ali'),
(28, 'Lewis', 'Juliette'),
(29, 'Man', 'Method'),
```

```
(30, 'Michelle-Gellar', 'Sarah'),
(31, 'Osment', 'Haley- Joel'),
(32, 'Phillip', 'Ryan'),
(33, 'Phonix', 'Joaquin'),
(34, 'Pitt', 'Brad'),
(35, 'Potter', 'Terry'),
(36, 'Schilling', 'Tom'),
(37, 'Schwarzenegger', 'Arnold'),
(38, 'Smith', 'Will'),
(39, 'Snipes', 'Wesley'),
(40, 'Spacey', 'Kevin'),
(41, 'Sproge', 'Ove'),
(42, 'Stadtlober', 'Robert'),
(43, 'Stallone', 'Sylvester'),
(44, 'Swayze', 'Patrick'),
(45, 'Tarantino', 'Quentin'),
(46, 'Travolta', 'John'),
(47, 'Walker', 'Paul'),
(48, 'Watts', 'Naomi'),
(49, 'Wayans', 'Marlon'),
(50, 'Whitherspoon', 'Reese'),
(51, 'Willis', 'Bruce'),
(52, 'Cleese', 'John'),
(53, 'Hartnett', 'John'),
(54, 'Beckinsale', 'Kate'),
(55, 'Winslet', 'Kate'),
(56, 'Wayne', 'John');
```

Lösung zu A4

Kunde:

`#kdnr, kdname, kdvorname, kdstrasse, kdplz, kdort, kdgebdat, kdgeschlecht`

Auftrag:

`#ausleihenr, ausleihdatum, kdnr`

beinhaltet

`#ausleihenr, #mnr`

Medium:

`#mnr, mart, mkaufdatum, fnr`

Film:

`#fnr, ftitel, ffsk, erscheinungsjahr, genre`

Genre:

`#genrenr, gbezeichnung`

Rolle:

`#fnr, #schnr`

Schauspieler:

`#schnr, schname, schvorname`

Anmerkung:

- # kennzeichnet einen Primärschlüssel
- _ kennzeichnet Fremdschlüssel
- Die Auflösung des Attributs Kundenname erfolgt durch die Spalten `kname` und `kvorname`, und die Auflösung des Attributs Kundenanschrift erfolgt durch die Spalten `kdstrasse, kdplz, kdort.`
- Auf die Ausgliederung einer Tabelle Kundenort mit den Spalten `kdort` und `kdplz` gemäß den Regeln zur dritten Normalform wurde hier verzichtet (Denormalisierung).

Lösung zu A5

A5.1

```
INSERT INTO videothek.kunde (kdname, kdvorname, kdstrasse, kdort, kdplz,
kdgeschlecht, kdgebdat)
VALUES ('Krumbein', 'Sebastian', 'Ölweg 23', 'Düsseldorf', '40123', 'männ-
lich', '1983-10-27')
```

A5.2

```
UPDATE kunde SET kdstrasse='Mühlenweg 12', kdort='Bergheim', kdplz ='51432'
WHERE kdname='Schmidt' and kdvorname='Harald'
```

A5.3

Film:

```
DELETE FROM film
WHERE ftitel=’Crazy’
```

Medium:

```
DELETE FROM medium
WHERE filmnr=5
```

A5.4

```
SELECT kdname, kdvorname, kdgebdat
FROM Kunde
WHERE kdname Like 'b%' Or kdname Like 'k%'
ORDER BY kdname;
```

A5.5

```
SELECT ftitel, erscheinungsjahr, Genre.gbezeichnung, ffsk
FROM Film INNER JOIN Genre ON Film.genrenr=genre.genrenr
WHERE ffsk=12 AND gbezeichnung='Drama';
```

A5.6

```
SELECT medium.mnr, medium.mart, film.fnr, film.ftitel
FROM Film INNER JOIN Medium ON film.fnr = medium.fnr
WHERE mart='VHS';
```

A5.7

```
SELECT Film.fnr, Film.ftitel AS 'Filmtitel' , Film.ffsk 'frei ab' , Schau-
spieler.schname, Schauspieler.schvorname
FROM Schauspieler INNER JOIN(Rolle INNER JOIN Film ON Rolle.fnr=Film.fnr)ON
Rolle.schnr=Schauspieler.schnr
WHERE schname='Sproge' AND schvorname='Ove';
```

A5.8

```
SELECT count(kdnr) AS 'Anzahl Kunden', kdgeschlecht
FROM Kunde
GROUP BY kdgeschlecht
```

A5.9

```
SELECT COUNT(film.fnr),Schauspieler.schname, Schauspieler.schvorname
FROM Film INNER JOIN (Rolle INNER JOIN Schauspieler ON Schaus-
pieler.schnr=rolle.schnr) ON Rolle.fnr=Film.fnr
GROUP BY Schauspieler.schname, Schauspieler.schvorname
HAVING COUNT(film.fnr)>2
```

2.5 Jahrgangsstufe 13.1

Beispielaufgabe für eine Datenbankanwendung

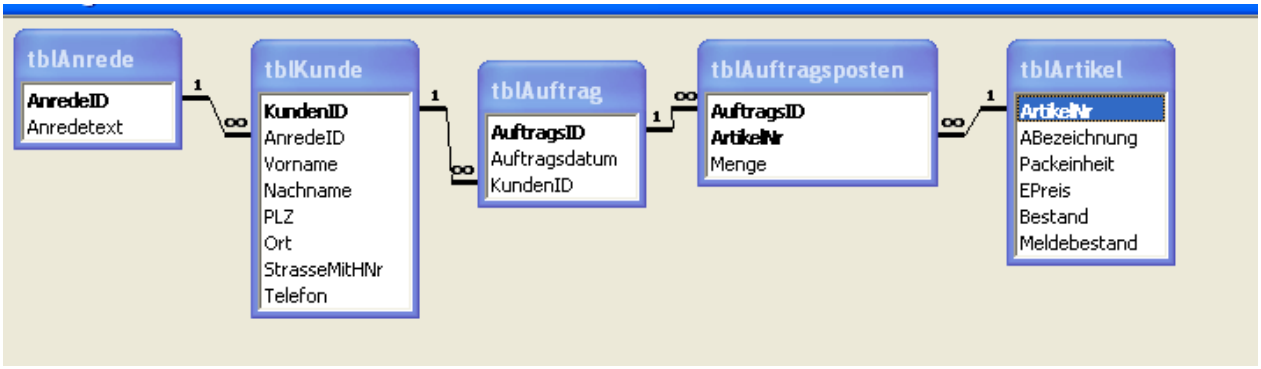
Titel	**Artikelverwaltung für die Paperback GmbH**
Vorbemerkung	Die folgende Aufgabe steht als Beispiel für die Implementation von Funktionen, die für eine Datenbankanwendung notwendig sind. Es handelt sich um eine formularbasierte Anwendung mit jeweils eigenen Klassen für die Benutzungsoberfläche, die Fachschicht und den Datenbankzugriff.
Voraussetzungen gemäß Lehrplan	Objektorientierter Datenbankzugriff Integrierte SQL-Anweisungen GUI
Anwendungsbezug	Stammdatenverwaltung für die Artikel eines Handelsunternehmens
Fachlicher Schwerpunkt	Objektorientierter Datenbankzugriff
Eingesetzte Hilfsmittel	Programmiersprache Java, Programmierumgebung NetBeans 6.5, Access-Datenbank PapArtikel.mdb
Anmerkungen	Die verwendeten Klassen lassen sich im gängigen Schichtenmodell den Schichten Präsentation, Fachkonzept und Datenbankzugriff zuordnen. Interaktion findet jeweils nur zwischen der Präsentations- und der Fachkonzept- sowie der Fachkonzept- und der Datenbankzugriffsschicht statt. Allerdings sind zu Gunsten didaktischer Reduktion die Schichten nicht streng entkoppelt (hierfür müssten sonst die Konzepte „Interface“ und evtl. geeignete Entwurfsmuster zur Verfügung stehen).

Situation

Sie sind bei der Papierwarengroßhandlung Paperback GmbH als neuer Mitarbeiter ins IT-Entwicklungsteam eingestellt. Ihr Gruppenleiter ist gerade damit befasst, den Zugriff auf die Accessdatenbank **PapArtikel.mdb** mit Hilfe der Programmiersprache Java zu automatisieren. Diese Datenbank dient u. a. dazu, die Daten für die Artikelverwaltung dauerhaft zu speichern und zu pflegen.

Sie sollen das begonnene kleine Projekt eigenständig weiterführen. Dazu wird Ihnen die folgende Dokumentation der bisher geleisteten Arbeit übergeben. Ergänzt wird die Dokumentation durch Aufträge über die noch von Ihnen durchzuführenden Arbeiten.

Datenbank



GUI-Artikelverwaltung und geforderte Funktionalitäten

Auf der Grundlage des Pflichtenhefts zum Projekt wurde bereits die nachfolgende GUI unter **Java** in der Entwicklungsumgebung NetBeans im Projekt **Aufgabe131** implementiert, wobei die dokumentierten Funktionalitäten zum Teil noch von Ihnen im Projekt implementiert werden müssen.

Nach einem Klick auf die Kommandoschaltfläche

(1) *cmdAlle* sollen alle Artikel, die sich zu diesem Zeitpunkt in der Datenbank befinden, unten in dem Listenfeld angezeigt werden.

(2) *cmdAnlegen* sollen neue Artikel in der Tabelle tblArtikel der Datenbank PapArtikel.mdb dauerhaft erfasst werden.

(3) *cmdSuchen* soll der Artikel, dessen Artikelnummer eingegeben wurde, gesucht und angezeigt werden.

(4) *cmdAendern* sollen Änderungen, die für einen angezeigten Artikel vorgenommen werden, in der Datenbank aktualisiert werden.

(5) *cmdLoeschen* soll der Artikel, dessen Daten gerade angezeigt werden, aus der Tabelle tblArtikel gelöscht werden. Anschließend sind auch die Felder in der GUI zu leeren.

(6) *cmdLeeren* sollen die angezeigten Daten in den Feldern der GUI (und nur hier) gelöscht werden.

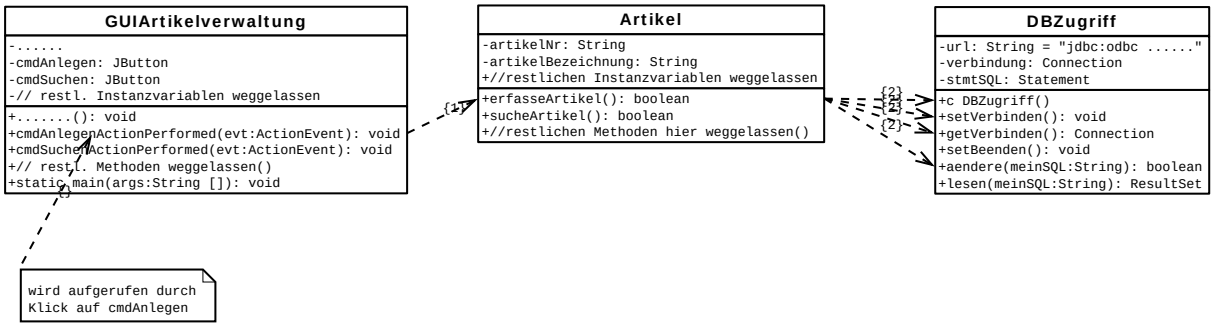
(7) *cmdEnde* soll die Anwendung beendet werden.

Die Funktionen für die Kommandoschaltflächen (1), (2), (6) und (7) sind bereits in dem NetBeans-Projekt namens **Aufgabe131** in Java implementiert.

Verfahrensdarstellung bei der Ersterfassung eines Artikel

Ein Artikel wird über die Maske erfasst und in der Tabelle tblArtikel der Datenbank gespeichert. Es wird eine 3-Schichten-Architektur angewandt. Aus Gründen der Einfachheit und Übersichtlichkeit sind die Schichten nicht vollständig entkoppelt.

GUI-Schicht Fachschicht DB-Zugriffsschicht



(1) Durch einen Klick auf den `JButton cmdAnlegen` wird der Methode `cmdAnlegenActionPerformed()` des Ereignisbehandlers in der GUI-Schicht für diesen Button aktiviert.

GUI-Schicht

```
private void cmdAnlegenActionPerformed(java.awt.event.ActionEvent evt) {
 boolean okAnlegen;
 leerenNachricht();
 String aNr =txtArtikelnummer.getText();
 String aBez = txtArtikelbezeichnung.getText();
 String aPack = txtPackeinheit.getText();
 double aPreis  = Double.parseDouble(txtEinkaufspreis.getText());
 int aBest = Integer.parseInt(txtBestand.getText());
 int aMeld = Integer.parseInt(txtMeldebestand.getText());
 akArtikel = new Artikel(aNr, aBez, aPack, aPreis, aBest, aMeld);
 okAnlegen = akArtikel.erfassenArtikel();
 if (okAnlegen == true) {
   lblNachricht.setText(„Artikel wurde erfolgreich erfasst; Alle anzeigen erneut aktivieren.“);
 }
 else{
    lblNachricht.setText(„Fehler bei der Artikelerfassung; Alle anzeigen erneut aktivieren.“);
 }
```

(2) Aus der Ereignisbehandlermethode wird an einem Objekt der Fachschicht die Methode `erfasseArtikel()` aus der Fachschicht aufgerufen. Die GUI-Schicht greift nur auf die Fachschicht zu.

```
public boolean erfassenArtikel(){
    boolean okErfassenArtikel;
    try{
        DBZugriff dbPapAuftrag = new DBZugriff();
        dbPapAuftrag.setVerbinden();
        String mSQL = „ INSERT INTO tblArtikel ( ArtikelNr, ABezeichnung, Packeinheit, EPreis,
        Bestand, Meldebestand )“;
        mSQL = mSQL + „Values ('„ + this.getArtikelNr() + „', '„ + this.artikelBezeichnung + „', '„ +
        this.getPackeinheit() + „', „ + this.getPreis() + „, „ + this.getBestand() + „, „ +
        this.getMeldebestand() + „)“;
        dbPapAuftrag.aenderen(mSQL);
        dbPapAuftrag.setBeenden();
        okErfassenArtikel = true;    }
    catch(Exception e){
        okErfassenArtikel = false;    }
    return okErfassenArtikel;
    }
```

1 2 3

siehe Folgeseite

1 2 3

DB-Zugriffsschicht

```
public void setVerbinden(){
     try {
         Class.forName(„sun.jdbc.odbc.JdbcOdbcDriver“);
         verbindung =DriverManager.getConnection(url);
         stmtSQL = verbindung.createStatement();

     }
     catch (Exception e){
         System.out.println(„Datenbankanbindung fehlgeschlagen“);
         System.out.println(e.toString());
     }
 }
public boolean aenderen(String meinSQL){
    boolean okAendernInDB;
    try{
        stmtSQL.execute(meinSQL);
        okAendernInDB = true;
    }
    catch(Exception e){
        okAendernInDB = false;
     }
    return okAendernInDB;
}

public void setBeenden(){
    try{
        verbindung.close();
    }
    catch(Exception e){
      System.out.println(„Fehler beim Schließen der Verbindung zur DB Pa-
pAuftrag.“);
 }

}
```

Die Methode `anderen(String meinSQL)` schließlich schreibt durch Aufruf der Methode `execute(meinSQL)` an einem Objekt der vorher geöffneten (1) Datenhaltungsschicht (nämlich Papauftrag.mdb) gemäß dem übergebenen SQL-Befehl die Artikeldaten in die Tabelle tblArtikel.

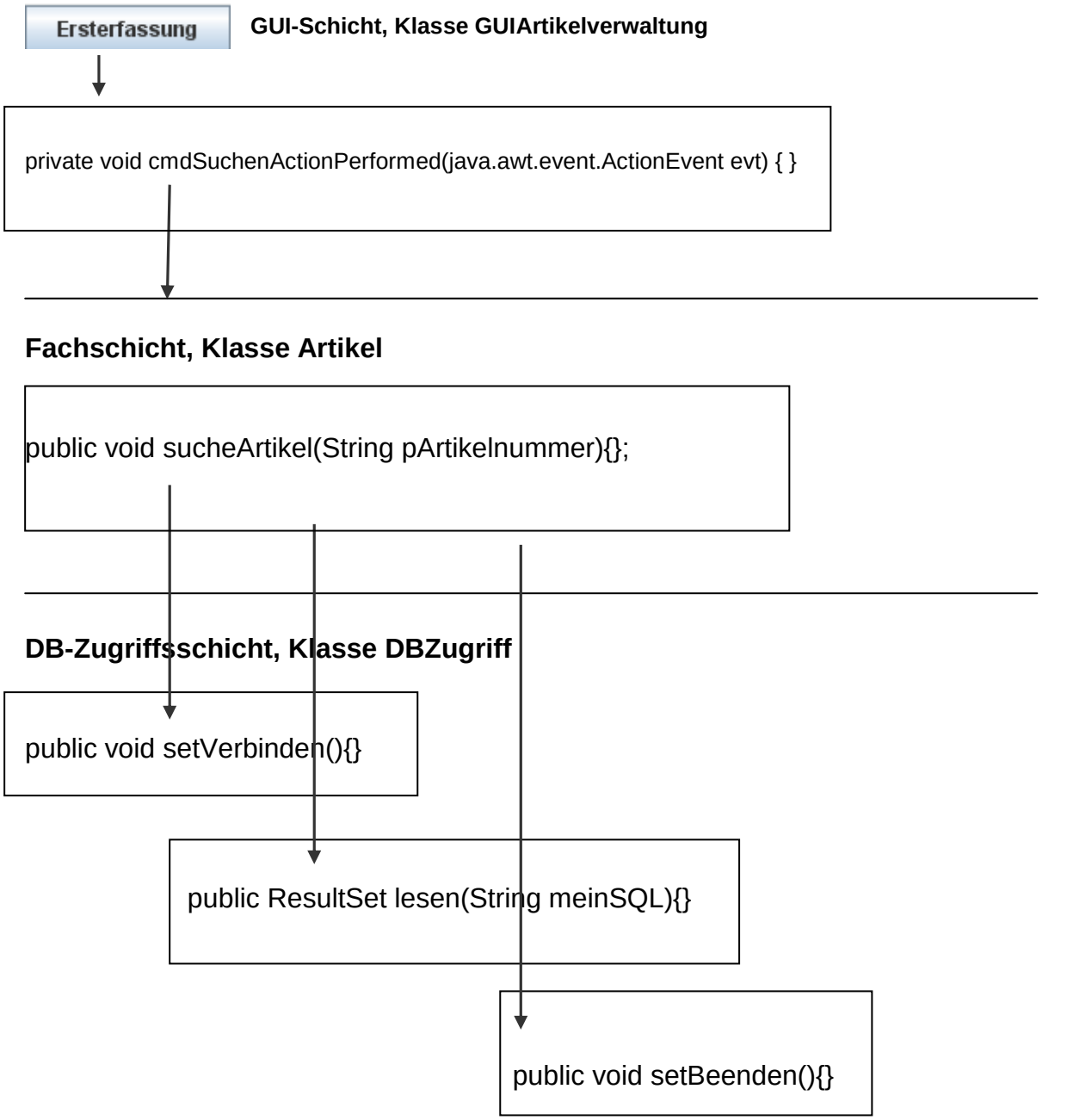
tblArtikel : Tabelle

ArtikelNr	ABezeichnung	Packeinheit	EPreis	Bestand	Meldebestand
P001	Briefpapier L1	50 Blatt	3,44	150	50
P002	Schreibmaschinenpapier S2	100 Blatt	4,55	200	50
P003	Kopierpapier K33	500 Blatt	10,99	400	100
P004	Neues Zeitungspapier	100 Blatt	100	60	10
P005	Blaues Briefpapier	50 Blatt	8,45	0	0
P006	Briefpapier Classic	50 Blatt	6,78	21	5
P007	Werkdruckpapier	1000 Bogen	234,34	100	50
P008	Kopierpapier rosa	200 Blatt	12,56	80	20
P009	Kopierpapier grün	200 Blatt	12,56	80	20
				0	0

Arbeitsaufträge

A1 Durch Klick auf den JButton *cmdSuchen* in der GUI soll derjenige Artikel, dessen Artikelnummer eingegeben wurde, in der Datenbank gesucht und angezeigt werden. Falls für eine eingegebene Artikelnummer kein Artikel in der Datenbank gefunden wurde, ist dies als Nachricht in der GUI anzuzeigen.

Die Funktionalität ist über die folgende Aufrufkette in Analogie zur bereits implementierten Funktionalität für den JButton *cmdAnlegen* im Projekt Aufgabe 131 zu implementieren. Bereits vorhandene Implementierungen speziell in Klasse DBZugriff können und sollten genutzt werden.



A2 Ändern und Löschen

Für die JButtons *cmdAendern* und *cmdLoeschen* sind ebenfalls die Funktionalitäten **analog** zur zuvor dargestellten Aufrufkette im NetBeans Projekt Aufgabe131 zu implementieren. Geforderte Funktionalitäten:

A2.1 Durch Klick auf *cmdAendern* sollen Änderungen, die für einen angezeigten Artikel vorgenommen werden, in der Datenbank aktualisiert werden. Falls eine Änderung nicht möglich war, ist dieses als Nachricht in der GUI anzuzeigen.

A2.2 Durch Klick auf *cmdLoeschen* soll der Artikel, dessen Daten gerade angezeigt werden, aus der Tabelle tblArtikel gelöscht werden. Anschließend sind auch die Felder in der GUI zu leeren. Falls ein Löschen in der Datenbank nicht möglich war, ist dies als Nachricht auf der GUI anzuzeigen.

Lösung

Lösung zu A1 (**cmdSuchen**)

Lösungsschritt 1 **(GUI-Ebene, Klasse GUIArtikelverwaltung)**

private void cmdSuchenActionPerformed(java.awt.event.ActionEvent evt) {
`akArtikel = new Artikel(); //(1)`
 	
`String sArtikelNr = txtArtikelnummer.getText(); //(2)`
 	
akArtikel.sucheArtikel(sArtikelNr); ***//(3)***
	(1) Es wird ein neues Artikelobjekt erzeugt
	
	(2) Die Artikelnummer wird aus dem Textfeld der GUI ausgelesen.
	
	(3) Anschließend wird am Artikelobjekt die Methode **sucheArtikel()** mit der ausgelesenen Artikelnummer als aktueller Parameter aufgerufen.
	Damit wird eine Methode aus der Fachebene, nämlich hier aus der Klassse Artikel aktiviert.

// **(4)**
txtArtikelnummer.setText(akArtikel.getArtikelNr());
txtArtikelbezeichnung.setText(akArtikel.getArtikelBezeichnung());
txtPackeinheit.setText(akArtikel.getPackeinheit());
txtEinkaufspreis.setText(Double.toString(akArtikel.getPreis()));
txtBestand.setText(Integer.toString(akArtikel.getBestand()));
txtMeldbestand.setText(Integer.toString(akArtikel.getMeldebestand()));
String sArtNr = akArtikel.getArtikelNr();
	(4) Nachdem die Methode sucheArtikel() das Artikelobjekt mit den gesuchten Werten aus der Datenbank gefüllt hat (siehe Lösungsschritt2), werden die gefundenen Werte aus dem Artikelobjekt ausgelesen und in den Textfeldern der GUI angezeigt.

//(5)
boolean ergebnis = sArtNr.equals(„nicht gefunden“);
if (ergebnis){
lblNachricht.setText(„Artikel ist in der Datenbank nicht vorhanden.“);
}
 	
}
	(5) Sollte sich der Wert „nicht gefunden“ in der Instanzvariable im Artikelobjekt befinden (siehe Lösungsschritt2), so war die Suche in der Datenbank nicht erfolgreich, was durch eine entsprechende Nachricht im GUI-Element lblNachricht angezeigt wird.

Lösungsschritt 2 (**Fachebene, Klasse Artikel**)

`public void sucheArtikel(String pArtikelnummer){ //(1)`	(1) Die Methode übernimmt als Parameter die beim Aufruf übergebenen Artikelnummer
`String mSQL; //(2)`	(2) und vereinbart eine Stringvariable für den SQL-Befehl.
`dbPapAuftrag = new DBZugriff(); //(3)`	(3) Es wird für den Datenbankzugriff ein neues DB-Objekt der Klasse DBZugriff instanziiert.
`dbPapAuftrag.setVerbinden(); //(4)`	(4) An dem erzeugten Objekt wird die Methode setVerbinden() aus der Zugriffsschicht (siehe Lösungsschritt3) für dasErzeugen einer Verbindung zur Datenbank aufgerufen.

`//(5)`
`mSQL = „Select ArtikelNr, ABezeichnung, Packeinheit, Epreis, Bestand, Meldebestand From tblArtikel“;`
`mSQL = mSQL + „ WHERE ArtikelNr = '„ + pArtikelnummer + „';“;`
	(5) Es wird der notwendige SQL-Befehl in der Stringvariablen mSQL erzeugt.

`ResultSet rsA = dbPapAuftrag.lesen(mSQL);//(6)`	(6) Am DB-Objekt wird die Methode lesen() mit dem erzeugten SQL-Befehl als Übergabeparameter aufgerufen. Sie gibt den empfangenen Datensatz als ResultSet durch die Variablen rsA zurück.

`//(7)`
`try{`
`rsA.next();`
`String nr = rsA.getString(1);`
`setArtikelNr(nr);`
`String bez = rsA.getString(2);`
`setArtikelBezeichnung(bez);`
`String pack = rsA.getString(3);`
`setPackeinheit(pack);`
`double pr = rsA.getDouble(4);`
`setPreis(pr);`
`int best = rsA.getInt(5);`
`setBestand(best);`
`int melde = rsA.getInt(6);`
`setMeldebestand(melde);`
`}`
`catch(Exception e){`
`System.out.println(„Kein Datensatz mit der Artikelnummer gefunden.“ + e.toString());`
`setArtikelNr(„nicht gefunden“);`
`}`
`finally{`
`dbPapAuftrag.setBeenden();`
`}`
	(7) In einer try-catch-Anweisung werden die einzelnen Felder des im ResultSet stehenden Datensatzes ausgelesen und den Instanzvariablen des aktuellen Artikelobjektes (siehe Lösungsschritt2, Schritte (1) bis (3)) übergeben. Im catch-Block wird im Fehlerfall die Instanzvariable ArtikelNr mit der entsprechenden Set-Methode mit einem Fehlerwert aufgefüllt.
	In jedem Fall wird im finally-Block der try-catch-Anweisung am DB-Objekte die Methode der Zugriffsschicht zum Schließen der DB aufgerufen.

`}`	

Lösungsschritt 3 **(Datenzugriffsschicht, Klasse DBZugriff)**

Hinweis: Die gesamte Klasse DBZugriff ist in dem vorgegebenen Projekt **Aufgabe131** vollständig in Java implementiert. Sie muss vom Lernenden verstanden und ihre Methoden im Lösungsschritt 2 korrekt genutzt werden. Aus Gründen der Vollständigkeit wird hier die gesamte Klasse DBZugriff dokumentiert.

`package AVEinfach;`
`import java.sql.*;`
`public class DBZugriff`
`{private final String url = „jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)}; DBQ=C:\\PapAuftrag.mdb“;`
`private Connection verbindung;//(1)`
`private Statement stmtSQL; //(1)`
`public DBZugriff(){`
`verbindung = null; //(2)`
`}`
	(1) Zunächst werden zwei Instanzvariablen für den Verweis auf zwei verschiedene Objekte deklariert:
	- ein Connectionobjekt (verbindung) für das Halten einer Verbindung zur Datenbank und
	- ein Statementobjekt für das Halten von auszuführenden SQL-Anweisungen.
	(2) Im Konstruktor wird das Objekt verbindung zuächst auf null (leer) gesetzt. Das Füllen der Instanzvariablen verbindung soll durch Methodenaufruf an einem DB-Objekt geschehen.

`public void setVerbinden(){ //(3)`
`try {`
`Class.forName(„sun.jdbc.odbc.JdbcOdbcDriver“);`
`verbindung =DriverManager.getConnection(url);`
`stmtSQL = verbindung.createStatement();`
`}`
`catch (Exception e){`
`System.out.println(„Datenbankanbindung fehlgeschlagen“);`
`System.out.println(e.toString());`
`}`
`}`
	(3) Die Methode setVerbinden( ) dient zum Aufbau einer Verbindung zu Datenbank PapAuftag.mdb, wobei die Datenbank im Wurzelverzeichnis von Laufwerk C: stehen muss.

`public Connection getVerbinden(){ // (4)`
`return verbindung;`
`}`
	(4) Die Methode getVerbinden() gibt eine Verbindung, die zuvor geschaffen wurde, an die aufrufende Instanz zurück.

`public void setBeenden(){//(5)`
`try {verbindung.close();}`
`catch(Exception e){`
`System.out.println(„Fehler beim Schließen der Verbindung zur DB PapAuftrag.“);`
`}`
`}`
	(5) Die Methode setBeenden( ) schließt eine bestehende Verbindung zu einer Datenbank, wenn sie aufgerufen wird.

```
public boolean aendern(String meinSQL){
     boolean okAendern2DB = false;
     try {
        int zeilen =  stmtSQL.executeUpdate(meinSQL);
        if (zeilen == 0){
            okAendern2DB = false;
        }
        else{
            okAendern2DB = true;
        }
     }
     catch(Exception e){
            okAendern2DB = false;
     }
     return okAendern2DB;
 }


public ResultSet lesen(String meinSQL){ //(6)
     ResultSet ergebnisTabelle = null;

     try{
         ergebnisTabelle = stmtSQL.executeQuery(meinSQL);
         return ergebnisTabelle;

     }
     catch( Exception e){
         ergebnisTabelle = null;
         return ergebnisTabelle;
     }
}
}
```

(6) Die Methode lesen() übernimmt aus der Fachklasse Artikel die dort formulierte SQL-Anweisung als Parameter (siehe Klasse Artikel und dort die Methode sucheArtikel im Lösungsschritt2). Eine Variable vom Typ ResultSet wird deklariert. Im try-Block wird an dem Objekt stmtSQL die übernommene SQL-Anweisung ausgeführt und das Ergebnis der Ausführung als Objekt vom Typ ResultSet zurückgegeben an die Variable ergebnisTabelle.

Ein Objekt vom Typ ResultSet kann man als eine Tabelle auffassen, die gemäß der übergebenen SQL-Anweisung hier einen Datensatz (falls Artikel gefunden) oder keinen Datensatz (falls in den catch-Block verzweigt wurde) enthält.

Lösung zu A2

A2.1 *cmdAendern*

Die Lösung zu dieser Teilaufgabe wird hier unkommentiert dokumentiert.

Lösungsschritt 1 **(GUI-Ebene, Klasse GUIArtikelverwaltung)**

```
private void cmdAendernActionPerformed(java.awt.event.ActionEvent evt) {
    boolean okAendern;
    try{
        akArtikel = new Artikel();
        akArtikel.setArtikelNr(txtArtikelnummer.getText());
         akArtikel.setArtikelBezeichnung(txtArtikelbezeichnung.getText());
        akArtikel.setPackeinheit(txtPackeinheit.getText());
        akArtikel.setPreis(Double.parseDouble(txtEinkaufspreis.getText()));
        akArtikel.setBestand(Integer.parseInt(txtBestand.getText()));
        akArtikel.setMeldebestand(Integer.parseInt(txtMeldebestand.getText()));
        okAendern = akArtikel.aendereArtikel();

        if (okAendern == true){
           lblNachricht.setText(„Artikeländerung erfolgreich; Alle anzeigen erneut
aktivieren.“);
            }
            else{
            lblNachricht.setText(„Artikeländerung fehlerhaft; Alle anzeigen erneut
aktivieren.“);
            }
        }
        catch(Exception e){
            lblNachricht.setText(„Weil nicht alle Felder müssen ausgefüllt sind,
kann eine Änderung nicht durchgeführt werden.“);
        }
    }
```

Lösungsschritt 2 (**Fachebene, Klasse Artikel**)

```
public boolean aendereArtikel(){
        boolean okAendernArtikel;
        try{
        DBZugriff dbPapAuftrag = new DBZugriff();
        dbPapAuftrag.setVerbinden();
        String sql = „UPDATE tblArtikel SET tblArtikel.ABezeichnung = '„ + artikel-
Bezeichnung + „', Packeinheit = ' „ + packeinheit + „' , EPreis = „ + preis + „,
Bestand = „ + bestand + „, Meldebestand = „ + meldebestand ;
        sql = sql + „ WHERE ArtikelNr = '„ + artikelNr + „';“;
        System.out.println(sql);
        dbPapAuftrag.aenderen(sql);
        dbPapAuftrag.setBeenden();
        okAendernArtikel = true;
        }
        catch(Exception e){
            okAendernArtikel = false;  }
        return okAendernArtikel;
    }
```

Lösungsschritt 3 **(DB-Zugriffsschicht**, **Klasse DBZugriff)**

Vergleiche hierzu die Lösung zu A1 Lösungsschritt 3, insbesondere die vorgegebene Methode aendern().

A2.2 *cmdLoeschen*

Lösungsschritt 1 **(GUI-Ebene, Klasse GUIArtikelverwaltung)**

```
private void cmdLoeschenActionPerformed(java.awt.event.ActionEvent evt) {
        akArtikel = new Artikel();
        akArtikel.setArtikelNr(txtArtikelnummer.getText());

        boolean okLoeschen = akArtikel.loescheArtikel();
        if (okLoeschen) {
        txtArtikelnummer.setText(„„);
        txtArtikelbezeichnung.setText(„„);
        txtPackeinheit.setText(„„);
        txtEinkaufspreis.setText(„„);
        txtBestand.setText(„„);
        txtMeldebestand.setText(„„);
        lblNachricht.setText(„Löschbefehl wurde akzeptiert; Alle anzeigen erneut
aktivieren.“);
        }
        else{
        lblNachricht.setText(„Es konnte kein Datensatz gelöscht werden; Alle anzei-
gen erneut aktivieren.“);
        }

    }
```

Lösungsschritt 2 (**Fachebene, Klasse Artikel**)

```
public boolean loescheArtikel(){

        boolean okLoeschenArtikel;
        try {
        DBZugriff dbPapAuftrag = new DBZugriff();
        dbPapAuftrag.setVerbinden();
        String sql = „Delete * from tblArtikel  WHERE ArtikelNr = '„ + artikelNr +
„';“;
        okLoeschenArtikel = dbPapAuftrag.aendern(sql);
        dbPapAuftrag.setBeenden();
        }
        catch(Exception e){
            okLoeschenArtikel = false;
        }

        return okLoeschenArtikel;

    }
```

Lösungsschritt 3 **(DB-Zugriffsschicht**, **Klasse DBZugriff)**

Vergleiche hierzu die Lösung zu A1 Lösungsschritt 3, insbesondere die vorgegebene Methode aendern().

2.6 Jahrgangsstufe 13.2

Titel	**Autoteilezuliefererbetrieb mit Hochregallager**
Vorbemerkung	Die folgende Aufgabe steht als Beispiel für eine umfangreiche ganzheitliche Aufgabe, die die Themen der Halbjahre 12.1 - 13.1 vernetzt.
Voraussetzungen gemäß Lehrplan	Grundelemente der Objektorientierung Erweiterung der OOP Datenbankentwurf UML SQL Objektorientierter Datenbankzugriff
Anwendungsbezug	Auftragsbearbeitung
Fachlicher Schwerpunkt	Komplexe Anwendungsaufgabe
Eingesetzte Hilfsmittel	Programmiersprache C#, Programmierumgebung Visual Studio, SQL Server 2005 (Lösung übertragbar auf SQL Server Express und Visual Studio Express)
Anmerkungen	Zeitbedarf: längere Unterrichtssequenz (ca. 4 Doppelstunden) auch als Projektaufgabe geeignet

Situation

Die Firma Kormann GmbH, mit Sitz in Borken, ist ein führender Lieferant von Autozubehör für Kunden im westlichen Münsterland. Das Sortiment der Firma umfasst ca. 8000 verschiedene Artikel aus dem Bereich Ersatzteile, Zubehör, Verschleißteile; alles für Automobile, Personen- wie Nutzfahrzeuge.

Logistik des Unternehmens

Kormann betreibt in Billerbeck (Westfalen) ein Hochregallager (HRL) als Basis seiner Logistik.

Täglich werden bis zu 400 Kommissionen zusammengestellt mit im Mittel 1700 „Picks“, also Einzelpositionen, die einen Zugriff auf das Lager erfordern.

Die Firma Kormann beliefert überregional ca. 2000 Automobilwerkstätten.

Lagerverwaltung

Das HRL in Billerbeck ist gekennzeichnet durch folgende Merkmale

- Automatisches Hochregallager
- Dimension: 64 m x 30 m x 20 m (HBT)
- 6 Lagergassen

- Lager für Palettengröße
  1,2 m x 0,8 m x 1,4 m
- Anzahl Lagerplätze: 4000
- Feste Regale
- Automatische Regalbediengeräte (Stöcklin)

Vernetzung von Kormann

Die drei Standorte von Kormann (Borken, Coesfeld und Billerbeck) sind über ein firmeneigenes Intranet vernetzt.

Ihre Rolle bei Kormann

Sie arbeiten als Mitarbeiterin bzw. Mitarbeiter der IT-Abteilung, die EDV-Systeme entwickelt, um sämtliche Geschäftsprozesse des Unternehmens automatisiert zu unterstützen. Sie persönlich beschäftigen sich mit einem Projekt zur Artikelverwaltung.

Pflichtenheft

Ein Pflichtenheft ist bereits in Ansätzen erstellt. Konkretisiert werden soll im ersten Ansatz die Bearbeitung eines Kundenauftrags. Zu diesem Zweck führen Sie ein Gespräch mit den Anwendern in ihrem Unternehmen, welches auszugsweise mitprotokolliert wurde und dem zur näheren Erläuterung einige Belege beigefügt wurden.

Frau Sosti (Vertrieb): „Zunächst geht bei mir eine telefonische oder schriftliche Bestellung ein, zumeist in Folge eines angefragten und erstellten Angebots (s. Anhang). Ich überprüfe zunächst, ob der Kunde bei uns offene Rechnungen hat und als zuverlässig gilt. Weiter wird die Bestellung selbst geprüft. Hierzu geht eine Anfrage über die Verfügbarkeit der bestellten Artikel an einen Lagermitarbeiter."

Herr Karsten (Lager): „Wir prüfen die zugehörigen Lagerbestände. Sind ausreichend Artikel vorhanden, benachrichtigen wir den Vertrieb und die Inkassoabteilung und verpacken die bestellten Waren und stellen diese bereit. In der Inkasso-Abteilung wird die zugehörige Rechnung erstellt. Diese wird im Warenausgang der bereitgestellten Ware hinzugefügt und der Warenausgang veranlasst gegebenenfalls den Transport."

IT-Abteilung: „Und was passiert, wenn der Lagerbestand zu gering ist?"

Herr Karsten (Lager): „In diesem Fall benachrichtigen wir ebenfalls den Vertrieb, der dann eine entsprechende Auftragsbestätigung versendet. Weiter wenden wir uns an den Einkauf, damit entsprechende Nachbestellungen veranlasst werden."

**Autowerkstatt Bruns
Reifenweg 33
46325 Borken
Tel.: 02861 2233**

Kormann Autoteile GmbH
Frau Sosti, Vertrieb
Autostraße 155
46325 Borken

Borken, 21.04.2010

Sehr geehrte Frau Sosti,

bezugnehmend auf Ihr Angebot vom 23.03.2010 bestellen wir bei Ihnen

ArtikelNr	Beschreibung	Menge	Einheit	Einzelpreis	Gesamtpreis
9	Anlasser, Starter für BMW	3	Stück	305,90 EUR	917,70 EUR
			Zuzüglich 19 % MwSt.:		174,36 EUR
				Total:	1092,06 EUR

zur Lieferung zum 24.04.2010.

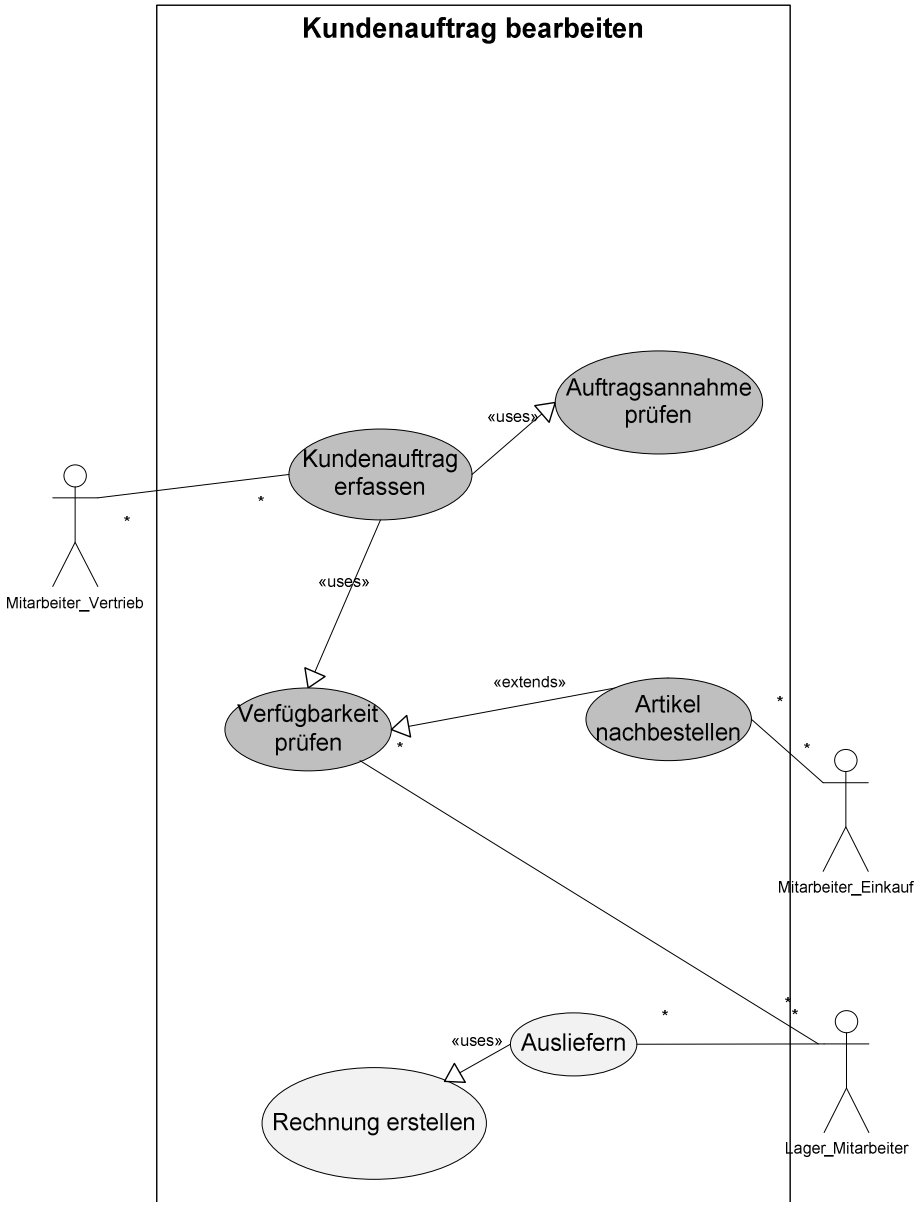
Mit freundlichen Grüßen

Peter Walter
Geschäftsführung

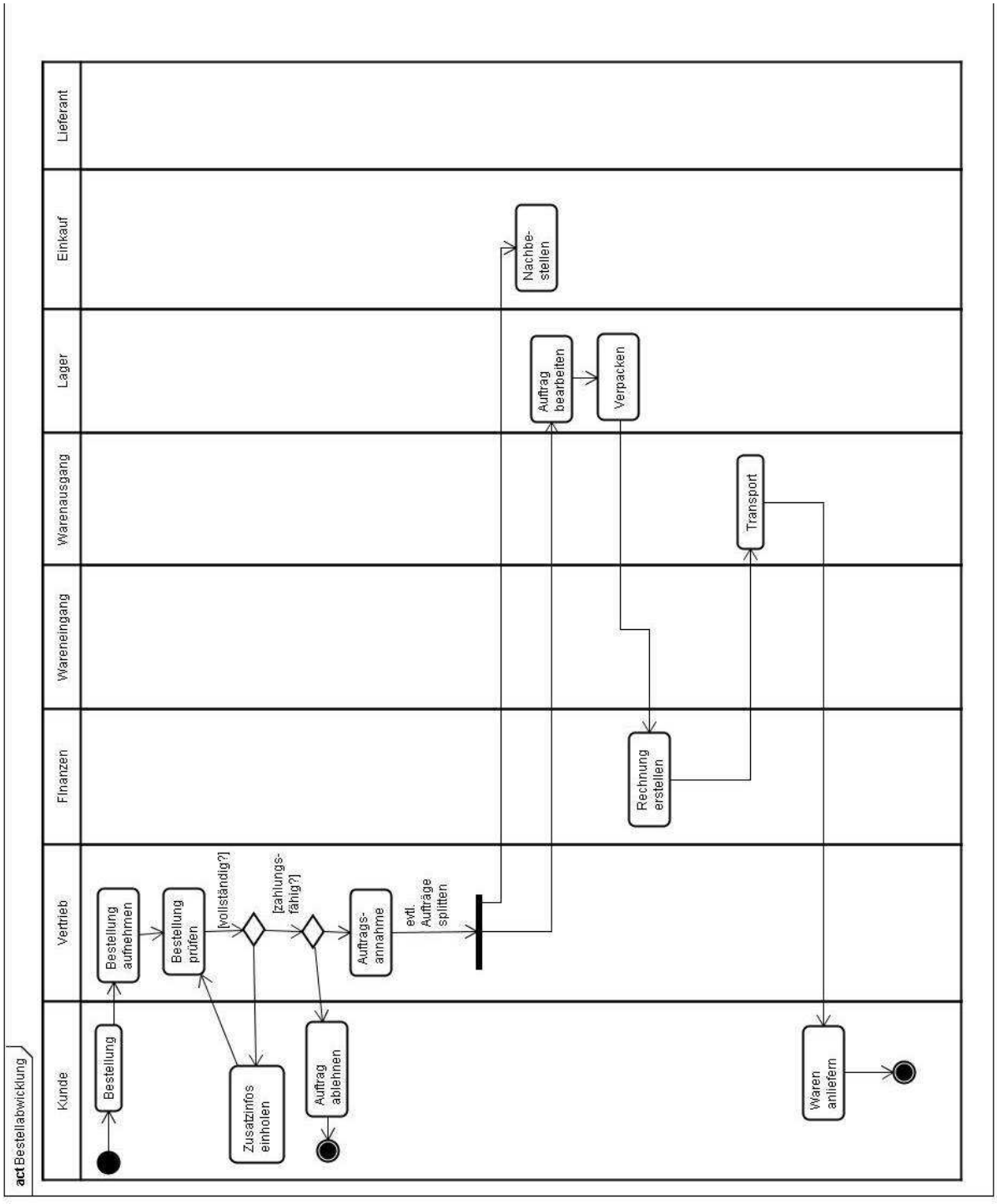
Arbeitsauftrag:

Zum besseren Verständnis der Gesprächsnotizen **erstellen** Sie zur Auftragsbearbeitung eine Anwendungsfallbeschreibung, ein detaillierteres Unteranwendungsfalldiagramm und ein Aktivitätsdiagramm, um zu visualisieren, wer mit welchem Zwischenschritt betraut ist. Bauen Sie diese an passender Stelle in das unvollständige Pflichtenheft ein.

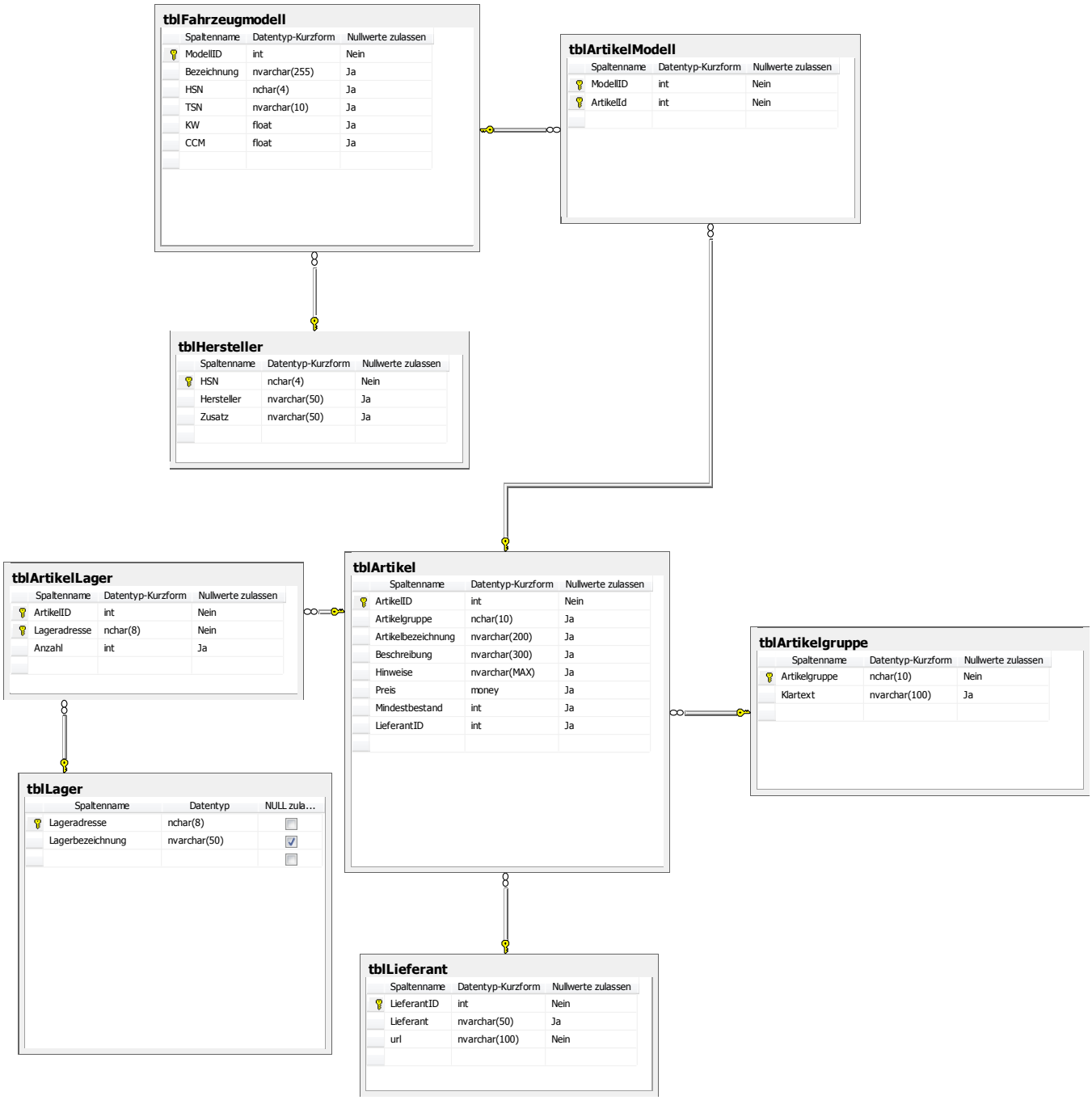
Mögliche Lösung:



Name	**Bestellabwicklung**
Kurzbeschreibung	Bestellung eines Kunden prüfen, bearbeiten, verpacken und liefern
Akteure	Kunde, Vertrieb, Finanzen, Wareneingang, Warenausgang, Lager, Einkauf, Lieferant
Auslöser	Bestellung des Kunden geht ein
Vorbedingung	Bestellung des Kunden kann angenommen werden
Ergebnis	Bestellte Waren werden beim Kunden angeliefert
Nachbedingung	-
Fehlerfälle	F1: Kunde als zahlungsunfähig bekannt F2: Fehler in der Bestellung (Artikel unbekannt etc.) F3: Bestellte Artikel nicht in ausreichender Menge auf Lager
Standardablauf	1. Prüfung der Bestellung 2. Auftragsannahme durch Vertrieb 3. Senden einer Auftragsbestätigung (Vertrieb) 4. Auftrag bearbeiten (Bereitstellen und Verpacken der bestellten Waren im Lager), zeitgleich können bei Verletzung der Mindestbestände Nachbestellungen bei den Lieferanten veranlasst werden (Einkauf) 5. Rechnung wird erstellt (Finanzen). 6. Transport veranlasst (Warenausgang) 7. Anlieferung beim Kunden
Alternativabläufe	2a: Ablehnung des Auftrags. 2b: Zusätzliche Informationen beim Kunden einholen (Vertrieb) 2c: Splitten in mehrere Aufträge, falls Artikel nur teilweise am Lager


act Bestellabwicklung
Kunde
Vertrieb
Finanzen
Wareneingang
Warenausgang
Lager
Einkauf
Lieferant
Bestellung
Bestellung aufnehmen
Bestellung prüfen
[vollständig?]
[zahlungs-fähig?]
Zusatzinfos einholen
Auftrag ablehnen
Auftrags-annahme
evtl. Aufträge splitten
Nachbe-stellen
Auftrag bearbeiten
Verpacken
Transport
Rechnung erstellen
Waren anliefern

Für die Lagerverwaltung muss eine komplexe Datenbank erstellt werden. Teile dieser Datenbank sind bereits schon geplant und wurden, um eine schnelle Umsetzung eines ersten Prototyps zu ermöglichen, bereits mit Teildatenbeständen als SQL Server-Datenbankdatei zur Verfügung gestellt. Das zugehörige Datenbankdiagramm ist nachfolgend abgebildet:



Ihnen ist nach der Konkretisierung des Kundenauftrages aufgefallen, dass Ihnen noch einige Informationen in der Datenbank fehlen, um die geforderten Leistungen der Anwendung erfüllen zu können. Sie machen sich folgende Stichpunkte:

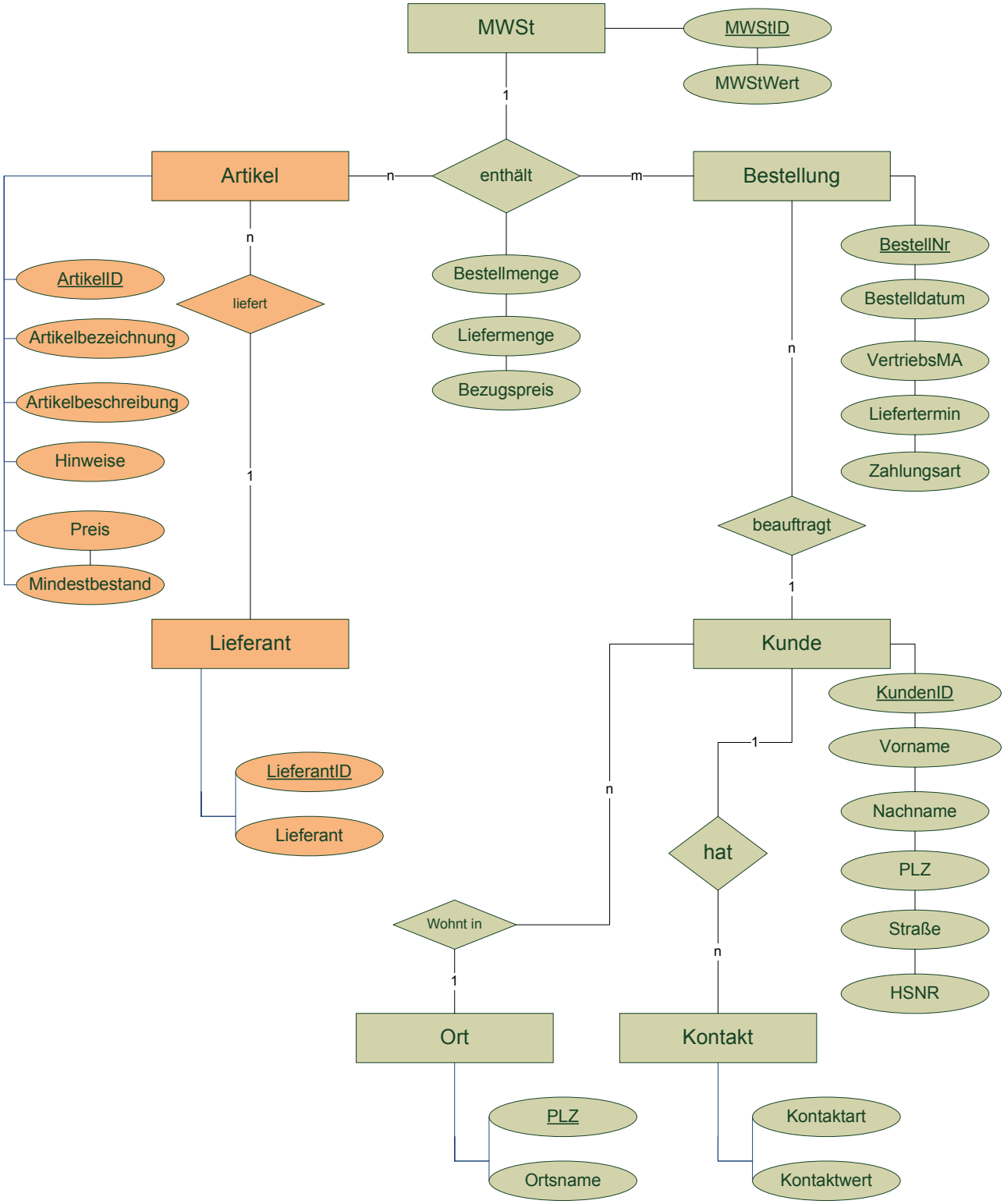
Ein Kunde gibt eine Bestellung auf. Zum Kunden werden Vor- und Nachname, Adressdaten und Kontaktdaten (E-Mail, Telefon geschäftlich, Telefon privat) gespeichert. In der Bestellung können mehrere Artikel bestellt werden. Zur Bestellung werden das Bestelldatum, der Vertriebsmitarbeiter, der den Auftrag angenommen hat, sowie der voraussichtliche Liefertermin und die Zahlungsweise angegeben. Zum bestellten Artikel wird der Bezugspreis je Mengeneinheit und die Bestellmenge sowie die Liefermenge eingetragen.

Arbeitsauftrag:

Vervollständigen Sie die Stichworte auf der Basis ihrer bisherigen Überlegungen und Ergebnisse und **entwerfen** Sie ein ER-Modell, das die Daten für Bestellvorgänge hinreichend berücksichtigt. Sofern Entitätstypen aus dem bereits bestehenden Modell relevant sind, sollen diese mit einbezogen werden und können falls erforderlich auch um weitere Attribute ergänzt werden.

Bisher wurde in der Datenbank die Mehrwertsteuer noch nicht berücksichtigt. Ergänzen Sie diese Information sinnvoll im Datenmodell und **begründen** Sie ihr Vorgehen.

Mögliche Lösung:



Es ist Freitagnachmittag – gedanklich sind Sie schon fast im Wochenende – da kommt ein Telefonanruf von der Geschäftsführung herein:

„Hallo, ich wollte Sie nur bitten für mich und die anderen Geschäftsführerkollegen eine kleine Präsentation vorzubereiten. Wir hatten doch mal über einige Auswertungen gesprochen, die wir zur Unterstützung von Entscheidungen aus dem neuen System benötigen. Wir würden uns gerne informieren, ob dies nach dem gegenwärtigen Entwicklungsstand schon möglich ist!“

Sie: „Brauchen Sie das noch heute?“

„Nein, das hat Zeit. Ich treffe mich mit den Kollegen erst am Montagmorgen um 10:00 Uhr.“

„Ich werde schauen, was sich machen lässt. Ein schönes Wochenende!“ ...

Fieberhaft suchen Sie in Ihren Aufzeichnungen nach den Anforderungen der Geschäftsführung und finden folgende Gesprächsnotiz:

- Für statistische Auswertungen wird eine Übersicht benötigt, die für **jeden** Lieferanten die Felder „LieferantID“ und „Lieferant“ anzeigt, sowie **die Anzahl der im Lager vorhandenen Artikel** dieses Lieferanten ausgibt**.**
- Des Weiteren soll der folgende Artikel ebenfalls in die Artikel-Tabelle eingefügt werden:

  Wärmetauscher, Innenraumheizung für Audi A2

  Artikelgruppe: 130000

  Beschreibung: Breite: 161 mm, Länge: 161 mm, Nennleistung [W]: 900 , Tiefe: 13 mm

  Hinweise: keine

  Preis: 364 EUR

  Mindestbestand: 5

  LieferantenID: 6 (Bosch)

- Es sollen diejenigen Artikelgruppen (Klartext) und deren Durchschnittspreis angezeigt werden, deren durchschnittlicher Artikelpreis aller zugeordneten Artikel mehr als 100 EUR beträgt. In der Ausgabe sollen die Spalten sinnvoll beschriftet sein und die Ausgabe absteigend nach dem Durchschnittspreis sortiert sein.

Arbeitsauftrag:

Programmieren Sie die erforderlichen SQL-Anweisungen und bereiten Sie eine kurze Präsentation für die Geschäftsführung vor. Überlegen Sie auch bereits im Vorfeld, wie Sie darauf reagieren, falls die Geschäftsführung die Sorge äußert, dass Sie nicht schnell und flexibel genug auf neue Anforderungen reagieren können.

Mögliche Lösung (mit der SQL Server Datenbank HRLKormann):

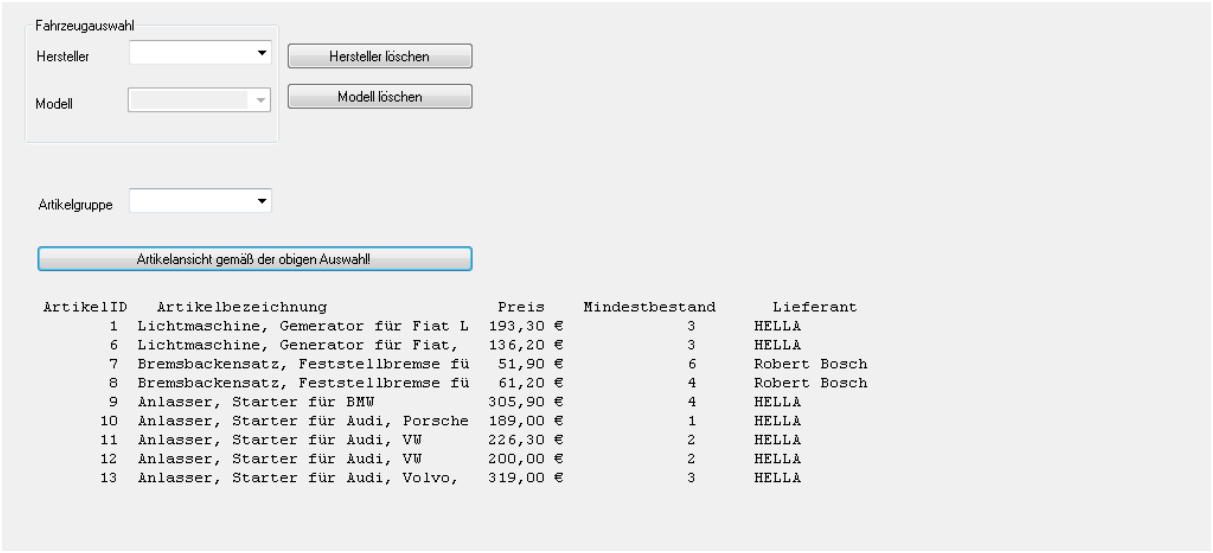
- ```
  use HRLKormann;
  --
  SELECT       tblLieferant.LieferantID,
             , tblLieferant.Lieferant
             ,  SUM(ISNULL(Anzahl, 0)) AS Lagermenge
  FROM
             tblLieferant LEFT OUTER JOIN
             tblArtikel
               ON  tblLieferant.LieferantID =
                           tblArtikel.LieferantID
             LEFT OUTER JOIN
             tblArtikelLager
               ON  tblArtikel.ArtikelID =
                           tblArtikelLager.ArtikelID
  GROUP BY  tblLieferant.LieferantID
          ,  tblLieferant.Lieferant
  ORDER BY Lagermenge DESC
  ```

- ```
  INSERT INTO tblArtikel
             (Artikelgruppe
             ,Artikelbezeichnung
             ,Artikelbeschreibung
             ,Hinweise
             ,Preis
             ,Mindestbestand
             ,LieferantID)
        SELECT
             '130000'
             ,'Wärmeaustauscher, Innenraumheizung für Audi A2'
             , 'Breite: 161 mm, Länge: 161 mm, Nennleistung [W]: 900 , Tiefe: 13 mm'
             ,NULL
             ,364
             ,5
             ,6
  ```

- ```
  SELECT       tblArtikelgruppe.Klartext
             , AVG(tblArtikel.Preis) AS Durchschnittspreis
  FROM          tblArtikel INNER JOIN
                          tblArtikelgruppe ON
  tblArtikel.Artikelgruppe =  tblArtikelgruppe.Artikelgruppe
  GROUP BY tblArtikelgruppe.Klartext
  HAVING   AVG(tblArtikel.Preis) > 100
  ORDER BY  AVG(tblArtikel.Preis)  DESC
  ```
  ```

Für das Aussehen der graphischen Oberfläche haben Sie im Projekteam ebenfalls bereits Vorstellungen entwickelt. Insbesondere ein Formular zur Artikelansicht wurde bereits detailliert geplant. Folgende Vorgaben sind im Pflichtenheft festgehalten:

Direkt nach dem Laden des Formulars „frmAutozubehör“ erscheint die folgende Ansicht:



Hierbei soll das Formular folgendermaßen auf die Interaktionsmöglichkeiten reagieren:

Klappbox für die Hersteller:

- Beim Laden des Formulars wird in der Klappbox nichts ausgewählt.
- Beim Aufklappen erscheint die Liste aller Hersteller in der Datenbank.
- Der Anwender kann einen Hersteller auswählen. Der ausgewählte Hersteller wird im zugehörigen Textfeld angezeigt.

Klappbox für die Modelle:

- Diese Klappbox ist zunächst inaktiv.
- Erst nach Wahl eines Herstellers wird die Klappbox aktiviert, aber zunächst ist nichts ausgewählt.
- Beim Aufklappen wird die Liste aller Modelle angezeigt, die zum gewählten Hersteller gehören.
- Der Anwender kann ein Modell wählen. Das ausgewählte Modell wird im zugehörigen Textfeld angezeigt.

Schaltfläche „Modell löschen“

- Nach Klicken dieser Schaltfläche wird die Klappbox für die Modelle erneut in den Ausgangszustand versetzt (bleibt aber aktiv).

Schaltfläche „Hersteller löschen“

- Nach Klicken dieser Schaltfläche wird die Klappbox für die Modelle und die Hersteller in den Ausgangszustand versetzt, die Klappbox für die Modelle ist nun inaktiv.

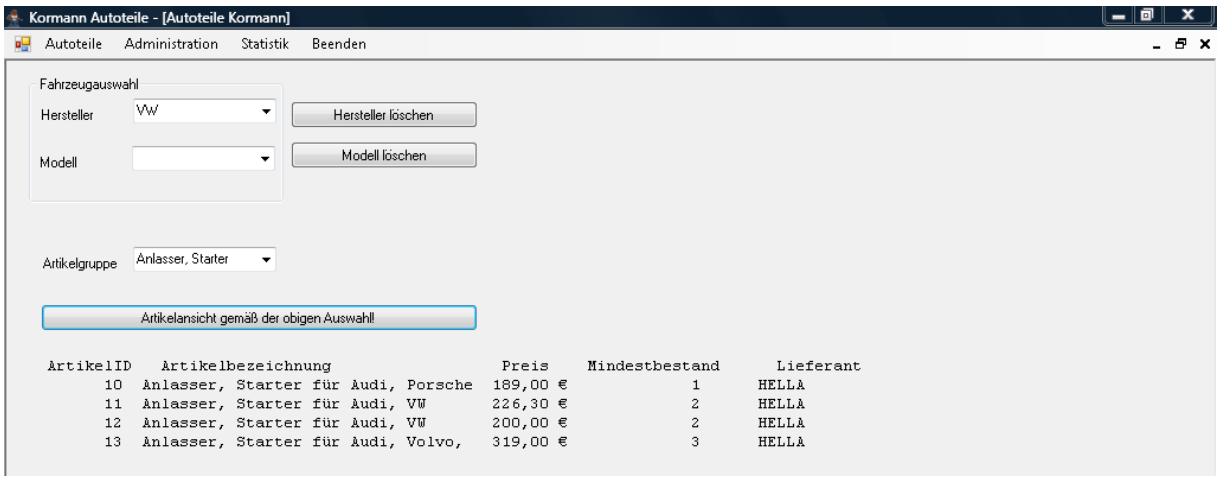
Klappliste „Artikelgruppe“

- In der Klappliste Artikelgruppe ist ebenfalls zu Beginn nichts ausgewählt.
- Trägt der Anwender nichts in das Textfeld ein und klappt die Liste auf, so bekommt er alle vorhandenen Artikelgruppen angezeigt.
- Trägt er in das Textfeld eine Zeichenkette ein und öffnet dann die Klappliste so werden nur noch die Artikelgruppen angezeigt, die mit dieser Zeichenkette beginnen (Kann-Feature)

Schaltfläche „Artikelliste gemäß der getroffenen Auswahl anzeigen“

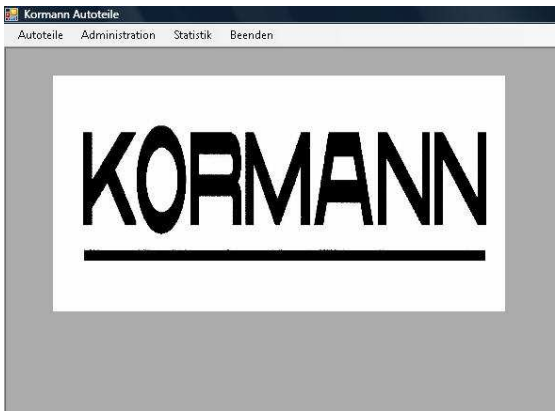
- Nach Klicken dieser Schaltfläche werden alle Artikel, die die vom Anwender vorgegebenen Kriterien erfüllen, aus der Datenbank ausgelesen und in einem Label angezeigt.

Im Folgenden ein Beispiel mit getroffenen Auswahlen:

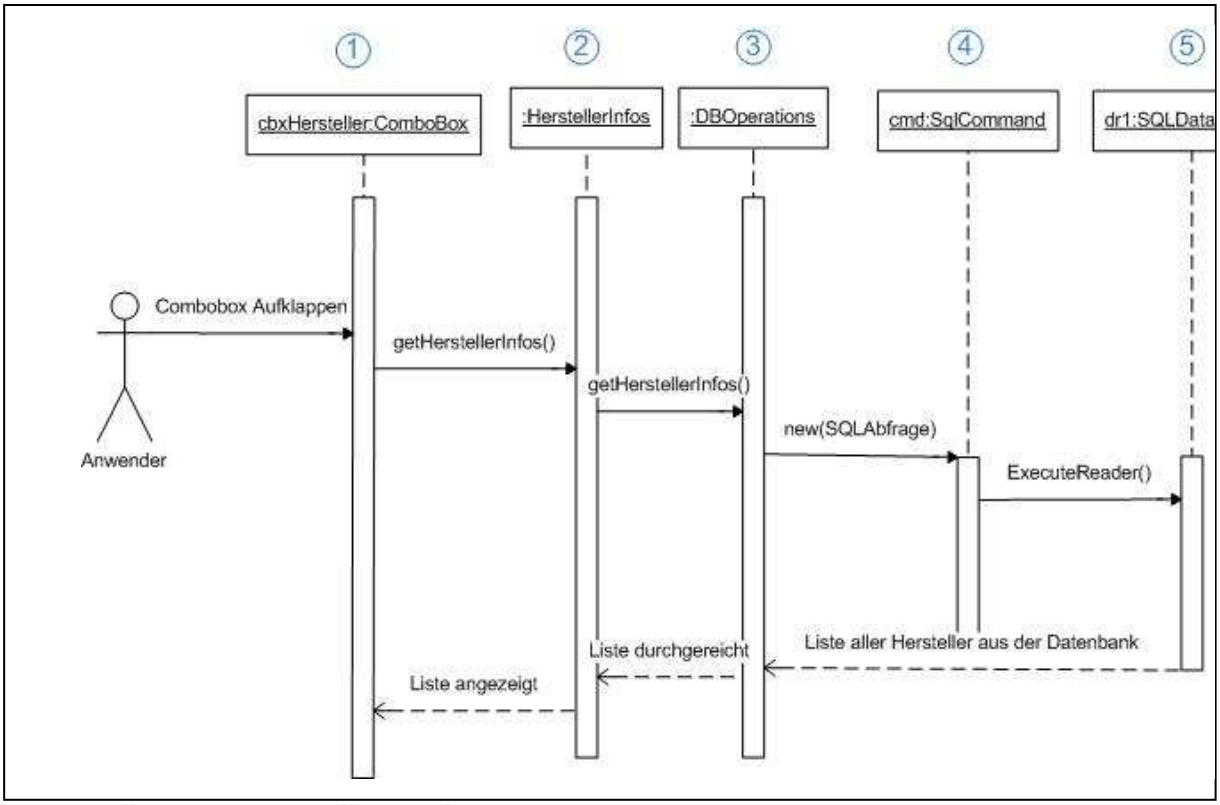


Arbeitsauftrag:

A1 Erstellen Sie eine MDI-Anwendung, in der das Formular zur Anzeige von Artikeln eingebettet wird. Hierbei soll das Menü gemäß der folgenden Abbildung eingerichtet werden. Beim Klicken auf den Menüeintrag „beenden“ wird die Anwendung geschlossen, beim Klicken auf den Menüeintrag „Autoteile“ wird ein zunächst leeres Formular angezeigt.



A2 In der Vorlagendatei Aufgabe13_2_Vorlage_Csharp finden Sie einen Lösungsvorschlag zu A1 sowie das Formular Autoteile. Die Funktionalität für die Klappbox „cbxHersteller“ ist bereits realisiert. Sie wird durch das folgende Sequenzdiagramm „HerstellerAnzeigen“ und die zugeordneten Codeabschnitte dokumentiert. Erstellen Sie für die Klappboxen cbxArtikelgruppe und cbxModell ebenfalls ein dokumentierendes Sequenzdiagramm. Implementieren Sie anschließend alle geforderten Funktionalitäten für diese beiden Klappboxen.



Sequenzdiagramm: HerstellerAnzeigen

Die im Sequenzdiagramm gezeigte Kommunikation zwischen anonymen Objekten in den Bearbeitungsstufen 2 und 3 wird aus Vereinfachungsgründen in der Lösung durch den Aufruf statischer Klassenmethoden realisiert.

Präsentationsschicht (Schritt 1 im Sequenzdiagramm)

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Linq;

namespace Kormann
{
    public partial class Autoteile : Form
    {
        // formularweite Variablen


        public Autoteile()
        {
            InitializeComponent();
            cbxModell.Enabled = false;
            FillComboBoxes();
        }
        private void FillComboBoxes()
        {
            //
            // Beginn Schritt 1 im Sequenzdiagramm HerstellerAnzeige
            //
            ArrayList lst = Herstellerinfos.GetHerstellerinfos();
            cbxHersteller.DataSource = lst;
            //
            // Ende Schritt 1 im Sequenzdiagramm HerstellerAnzeige
            //
            // Ab hier ergänzen Sie den Code zum Füllen der Artikelgruppen
         }
      }
```

Fachschicht (Schritt 2 im Sequenzdiagramm)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace Kormann
{
    //
    // Beginn Schritt 2 im Sequenzdiagramm HerstellerAnzeige
    //
    public class HerstellerInfo
    {
        private string hsn;

        public string Hsn
        {
            get { return hsn; }
        }
        private string hersteller;

        public string Hersteller
        {
            get { return hersteller; }
        }

        public HerstellerInfo(string hsn, string hersteller)
        {
            this.hsn = hsn;
            this.hersteller = hersteller;
        }
        public override string ToString()
        {
            return Hersteller;
        }
    }

    public class Herstellerinfos
    {
        public static ArrayList GetHerstellerinfos()
        {
            return DBOperations.GetHersteller();
        }

    }
    //
    // Ende Schritt 2 im Sequenzdiagramm HerstellerAnzeige
    //
      // Ergänzen Sie hier bitte Ihre Fachklasssen Modellinfo, Modellinfos,
Artikelgruppeinfo, Artikelgruppeninfos
}
```

Datenzugriffsschicht (Schritte 3 bis 5 im Sequenzdiagramm)

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace Kormann
{
    public static class DBOperations
    {
        // Beginn Schritt 3 im Sequenzdiagramm HerstellerAnzeige
        //
        public static ArrayList GetHersteller()
        {
            ArrayList lst = new ArrayList();
            lst.Add(new HerstellerInfo(„„, „„));
            //
            // Beginn Schritt 4 im Sequenzdiagramm HerstellerAnzeige
            //
            SqlConnection conn = new  SqlConnection(Properties.Settings.Default.conHRL);
            string query = @“SELECT
                                tblHersteller.HSN,
                                tblHersteller.Hersteller
                              FROM tblHersteller
                              ORDER BY Hersteller ASC“;
            SqlCommand cmd = new SqlCommand(query, conn);
            try
            {
                conn.Open();
                //
                // Beginn Schritt 5 im Sequenzdiagramm HerstellerAnzeige
                //
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    string hsn = dr.GetString(0);
                    string hersteller = dr.GetString(1);

                    lst.Add(new HerstellerInfo(hsn, hersteller));
                }
                dr.Close();
                //
                // Ende Schritt 5 im Sequenzdiagramm HerstellerAnzeige
                //
            }
            catch (Exception ex)
            {
                throw ex;
            }
            finally
            {
                conn.Close();
                //
                // Ende Schritt 4 im Sequenzdiagramm HerstellerAnzeige
            }

            return lst;
            //
            // Ende Schritt 3 im Sequenzdiagramm HerstellerAnzeige
        }
         // Implementieren Sie hier entsprechend die Methoden
         // GetModelle(HerstellerInfo h) und GetArtikelgruppen()
}
```

A3 (Zusatzaufgabe)
Implementieren Sie die Funktionalität zum Button „Artikelansicht gemäß der obigen Auswahl“, die zur Artikelanzeige in einem Label führt, wobei die in den Klappboxen ausgewählten Einschränkungen bei der Anzeige zu berücksichtigen sind.

Anmerkung: Hier bietet sich an, Schülerinnen und Schüler nur Teile ergänzen zu lassen und nicht die Entwicklung des gesamten Codes zu fordern.

Mögliche Lösung:

Zu A1:

```
private void autoteileToolStripMenuItem_Click(object sender, EventArgs e)
        {
                pictureBox1.Visible=false;
                pictureBox2.Visible = false;
            Autoteile a1 = new Autoteile();
            a1.MdiParent = this;
            a1.BringToFront();

            a1.Show();

        }

        private void beendenToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            this.Close();
        }
```

Gesamtlösung zu A2 und A3:

Präsentationsschicht (Datei Autoteile.cs im Projektordner):

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Linq;

namespace Kormann
{
    public partial class Autoteile : Form
    {
        // formularweite Variablen


        public Autoteile()
        {
            InitializeComponent();
            cbxModell.Enabled = false;
            FillComboBoxes();

        }
```

```
        private void FillComboBoxes()
        {
            //
            // Beginn Schritt 1 im Sequenzdiagramm HerstellerAnzeige
            //
            ArrayList lst = Herstellerinfos.GetHerstellerinfos();
            cbxHersteller.DataSource = lst;
            //
            // Ende Schritt 1 im Sequenzdiagramm HerstellerAnzeige
            //

            // Artikelgruppen zur Auswahl laden
            string pText=„ '%' „;
            ArrayList lstAGr = Artikelgruppeninfos.GetArtikelgruppeninfos(pText);
            cbxArtikelgruppe.DataSource = lstAGr;
            cbxArtikelgruppe.SelectedIndex = -1;

        }

        private void Autoteile_Load(object sender, EventArgs e)
        {

        }


        private void cbxHersteller_SelectedIndexChanged_1(object sender, EventArgs e)
        {
            if (cbxHersteller.SelectedIndex > 0)
            {
                cbxModell.Enabled = true;
                cbxModell.DataSource = null;

                HerstellerInfo h = (HerstellerInfo)cbxHersteller.SelectedItem;

                ArrayList lst = Modellinfos.GetModellinfos(h);
                cbxModell.DataSource = lst;
            }
            else
            {
                cbxModell.Enabled = false;
            }


        }

        private void cmdHerstellerLoeschen_Click_1(object sender, EventArgs e)
        {
            cbxHersteller.SelectedIndex = -1;
            cbxModell.Enabled = false;
        }

        private void cmdModellLoeschen_Click_1(object sender, EventArgs e)
        {
            cbxModell.SelectedIndex = -1;
        }

        private void cbxArtikelgruppe_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void cbxArtikelgruppe_TextChanged(object sender, EventArgs e)
        {

        }

        private void cbxArtikelgruppe_DropDown(object sender, EventArgs e)
        {
            string pText = „'„ + cbxArtikelgruppe.Text.Trim() + „%'„;
            ArrayList lstAGr = Artikelgruppeninfos.GetArtikelgruppeninfos(pText);
            cbxArtikelgruppe.DataSource = lstAGr;
        }

        private void btnArtikelansicht_Click(object sender, EventArgs e)
```

```
        {
            try
            {
            HerstellerInfo h;
            ModellInfo m;
            ArtikelgruppenInfo a;
            if (cbxHersteller.SelectedIndex >= 0)
            {
                h = (HerstellerInfo)cbxHersteller.SelectedItem;
            }
            else
            {
                h = new HerstellerInfo(„„, „„);
            }
            if (cbxModell.Enabled == true && cbxModell.SelectedIndex >= 0)
            {
                m = (ModellInfo)cbxModell.SelectedItem;
            }
            else
            {
                m = new ModellInfo(0, „„, „„);
            }
            if (cbxArtikelgruppe.SelectedIndex >= 0)
            {
                 a = (ArtikelgruppenInfo)cbxArtikelgruppe.SelectedItem;

            }
            else
            {
                 a = new ArtikelgruppenInfo(„„, „„);
            }
                // Aufruf der Methode der Fachschicht mit
                // weitergabe aller ausgewählten Objekte
                // für Hersteller, Modell und Artikelgruppe
                // Hat eine Wahl nicht stattgefunden kann dies an der Art
                // der Feldbelegung festgelegt werden.
            ArrayList lstArt = Artikelinfos.GetArtikelinfos(h,m,a);
            const char LF = (char)10;
            lblAktualisiert.Text = Artikelinfos.Kopf() + LF;
                foreach (ArtikelInfo art in lstArt)
                {
                    lblAktualisiert.Text = lblAktualisiert.Text + art.ToString() + LF;
                }
        }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
                }
            }
    }
}
```

Fachschicht (Datei DataObjects.cs im Projektordner)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace Kormann
{
    //
    // Beginn Schritt 2 im Sequenzdiagramm HerstellerAnzeige
    //
    public class HerstellerInfo
    {
        private string hsn;

        public string Hsn
        {
            get { return hsn; }
        }
        private string hersteller;

        public string Hersteller
        {
            get { return hersteller; }
        }

        public HerstellerInfo(string hsn, string hersteller)
        {
            this.hsn = hsn;
            this.hersteller = hersteller;
        }

        public override string ToString()
        {
            return Hersteller;
        }
    }

    public class Herstellerinfos
    {
        public static ArrayList GetHerstellerinfos()
        {
            return DBOperations.GetHersteller();
        }

    }
    //
    // Ende Schritt 2 im Sequenzdiagramm HerstellerAnzeige
    //

    public class ModellInfo
    {
        private int modellID;

        public int ModellID
        {
            get { return modellID; }
        }
        private string bezeichnung;

        public string Bezeichnung
        {
            get { return bezeichnung; }
        }
        private string hsn;

        public string Hsn
        {
            get { return hsn; }
        }
```

```
        public ModellInfo(int modellID, string bezeichnung, string hsn)
        {
            this.modellID = modellID;
            this.bezeichnung = bezeichnung;
            this.hsn = hsn;
        }

        public override string ToString()
        {
            return Bezeichnung;
        }
    }
    public class Modellinfos
    {
        public static ArrayList GetModellinfos(HerstellerInfo h)
        {
            return DBOperations.GetModelle(h);
        }

    }
    public class ArtikelgruppenInfo
    {
        private string artikelgruppe;

        public string Artikelgruppe
        {
            get { return artikelgruppe; }
            set { artikelgruppe = value; }
        }
        private string agrBezeichnung;

        public string AgrBezeichnung
        {
            get { return agrBezeichnung; }
            set { agrBezeichnung = value; }
        }
        public ArtikelgruppenInfo(string pArtikelgruppe, string pAgrBezeichnung)
        {
            this.artikelgruppe = pArtikelgruppe;
            this.agrBezeichnung = pAgrBezeichnung;
        }
        public override string ToString()
        {
            return agrBezeichnung;
        }
    }
    public class Artikelgruppeninfos
    {
        public static ArrayList GetArtikelgruppeninfos(string pText)
        {
            return DBOperations.GetArtikelgruppen(pText);
        }

    }
    public class ArtikelInfo
    {
        private int artikelID;

        public int ArtikelID
        {
            get { return artikelID; }
            set { artikelID = value; }
        }
        private string artikelbezeichnung;

        public string Artikelbezeichnung
        {
            get { return artikelbezeichnung; }
            set { artikelbezeichnung = value; }
        }
        private double preis;

        public double Preis
        {
```

```
            get { return preis; }
            set { preis = value; }
        }
        private int mindestbestand;

        public int Mindestbestand
        {
            get { return mindestbestand; }
            set { mindestbestand = value; }
        }
        private string lieferant;

        public string Lieferant
        {
            get { return lieferant; }
            set { lieferant = value; }
        }
        public ArtikelInfo(int pArtikelID, string pArtikelbezeichnung, double pPreis, int
pMindestbestand, string pLieferant)
        {
            this.artikelID = pArtikelID;
            this.artikelbezeichnung = pArtikelbezeichnung;
            this.preis = pPreis;
            this.mindestbestand = pMindestbestand;
            this.lieferant = pLieferant;

        }
        public override string ToString()
        {
            string s =„{0, 8:G}  {1,-30}  {2,8:C}  {3,12:G}      {4,-15}“;
            return String.Format(s, artikelID,
                artikelbezeichnung.Substring(0, Math.Min(artikelbezeichnung.Length,
35)).PadRight(35),
                preis, mindestbestand,
                lieferant.Substring(0, Math.Min(lieferant.Length, 15)).PadRight(15));

        }
    }
    public class Artikelinfos
    {
        public static ArrayList GetArtikelinfos(HerstellerInfo h, ModellInfo m,
ArtikelgruppenInfo a)
        {
            try
            {
            return DBOperations.GetArtikel(h,m,a);
                }
            catch (Exception ex)
            {
                throw ex;
                }
        }
        public static string Kopf()
        {
            return „ArtikelID   Artikelbezeichnung     Preis    Mindestbestand    Lieferant“;
        }
    }
}
```

Datenzugriffsschicht (Datei DBOperations.cs im Projektordner)

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.Windows;
using System.Windows.Forms;
namespace Kormann
{
    public static class DBOperations
    {
        //
        // Beginn Schritt 3 im Sequenzdiagramm HerstellerAnzeige
        //
        public static ArrayList GetHersteller()
        {
            ArrayList lst = new ArrayList();
            lst.Add(new HerstellerInfo(„„, „„));
            //
            // Beginn Schritt 4 im Sequenzdiagramm HerstellerAnzeige
            //
            SqlConnection conn = new SqlConnection(Properties.Settings.Default.conHRL);
            string query = @“SELECT
                                tblHersteller.HSN,
                                tblHersteller.Hersteller
                            FROM tblHersteller
                            ORDER BY Hersteller ASC“;

            SqlCommand cmd = new SqlCommand(query, conn);
            try
            {
                conn.Open();
                //
                // Beginn Schritt 5 im Sequenzdiagramm HerstellerAnzeige
                //
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    string hsn = dr.GetString(0);
                    string hersteller = dr.GetString(1);

                    lst.Add(new HerstellerInfo(hsn, hersteller));
                }
                dr.Close();
                //
                // Ende Schritt 5 im Sequenzdiagramm HerstellerAnzeige
                //
            }
            catch (Exception ex)
            {
                throw ex;
            }
            finally
            {
                conn.Close();
                //
                // Ende Schritt 4 im Sequenzdiagramm HerstellerAnzeige
                //
            }

            return lst;
            //
            // Ende Schritt 3 im Sequenzdiagramm HerstellerAnzeige
            //
        }

        public static ArrayList GetModelle(HerstellerInfo hersteller)
        {
            ArrayList lstModelle = new ArrayList();
            lstModelle.Add(new ModellInfo(0, „„, „„));
```

```
        SqlConnection conn = new SqlConnection(Properties.Settings.Default.conHRL);
        string query = @"SELECT
                          m.ModellID,
                          m.Bezeichnung
                        FROM tblFahrzeugmodell m
                        WHERE m.HSN = @HSN";

        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.Add("@HSN", SqlDbType.NChar, 4);
        cmd.Parameters["@HSN"].Value = hersteller.Hsn;
        try
        {
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                int id = dr.GetInt32(0);
                string bez = dr.GetString(1);

                lstModelle.Add(new ModellInfo(id, bez, hersteller.Hsn));
            }
            dr.Close();
        }
        catch (Exception ex)
        {
            throw ex;

        }
        finally
        {
            conn.Close();
        }

        return lstModelle;
    }
    public static ArrayList GetArtikelgruppen(string pText)
    {
        ArrayList lstAGr = new ArrayList();
        lstAGr.Add(new ArtikelgruppenInfo("", ""));

        SqlConnection conn = new SqlConnection(Properties.Settings.Default.conHRL);
        string query = @"SELECT
                           tblArtikelgruppe.Artikelgruppe,
                           tblArtikelgruppe.Klartext
                         FROM tblArtikelgruppe
                         WHERE tblArtikelgruppe.Klartext LIKE ";
        query=query+pText + " ORDER BY tblArtikelgruppe.Klartext ASC ";

        SqlCommand cmd = new SqlCommand(query, conn);

     //

        try
        {
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                string agr = dr.GetString(0);
                string artikelgruppe = dr.GetString(1);

                lstAGr.Add(new ArtikelgruppenInfo(agr, artikelgruppe));
            }
            dr.Close();
            return lstAGr;
        }
        catch (Exception ex)
        {
            throw ex;
        }
        finally
        {
            conn.Close();
```

```
            }


        }
        public static ArrayList GetArtikel(HerstellerInfo h, ModellInfo m, ArtikelgruppenInfo
a)
        {
            ArrayList lstArt = new ArrayList();
           // lstArt.Add(new ArtikelInfo(0,“„,0.0,0,“„));
            SqlConnection con = new SqlConnection(Properties.Settings.Default.conHRL);

            string sql = @“
                                SELECT DISTINCT [tblArtikel].[ArtikelID]
                                      ,[tblArtikel].[Artikelbezeichnung]
                                      ,[tblArtikel].[Preis]
                                      ,[tblArtikel].[Mindestbestand]
                                      ,tblLieferant.[Lieferant]

                                  FROM (([HRL].[dbo].[tblArtikel] INNER JOIN tblLieferant
                                       ON tblArtikel.LieferantID=tblLieferant.LieferantId)
                                       INNER JOIN tblArtikelModell ON
                                           tblArtikelModell.ArtikelID=tblArtikel.ArtikelID)
                                        INNER JOIN tblFahrzeugmodell ON
                                tblFahrzeugmodell.ModellID =tblArtikelModell.ModellID
                                 WHERE „;
            string Bedingung = „ 1=1 „;

                //HerstellerInfo h = (HerstellerInfo)cbxHersteller.SelectedItem;
                //string HSN = cbxHersteller.SelectedValue.ToString();
                string HSN = h.Hsn;
                if (HSN.Trim().Length > 0)
                {
                    Bedingung = Bedingung + „ AND tblFahrzeugmodell.HSN = '„ + HSN + „'„;
                }


                //ModellInfo m = (ModellInfo)cbxModell.SelectedItem;
                //string MId = cbxModell.SelectedValue.ToString();
                if (m.ModellID > 0)
                {
                    string MId = m.ModellID.ToString();
                    if (MId.Trim().Length > 0)
                    {
                        Bedingung = Bedingung + „ AND tblFahrzeugmodell.ModellId = „ + MId;
                    }
                }


            string agr = a.Artikelgruppe;
            if (agr.Trim().Length>0)
            {
                Bedingung = Bedingung + „ AND tblArtikel.Artikelgruppe = '„+agr.Trim()+“'„;
            }
            sql = sql + Bedingung;
            SqlCommand cmd = new SqlCommand(sql, con);

            try
            {
                con.Open();
                SqlDataReader dr1 = cmd.ExecuteReader();
                while (dr1.Read())
                {

                    int artikelID = 0;
                    string artikelbezeichnung = „„;
                    double preis = 0.0;
                    int mindestbestand = 0;
                    string lieferant = „„;
                    if (dr1.GetValue(0) != DBNull.Value)
                    {
                        artikelID = (int)dr1.GetInt32(0);
                    }
                    if (dr1.GetValue(1) != DBNull.Value)
                    {
```

```
                        artikelbezeichnung = dr1.GetValue(1).ToString();
                    }
                    if (dr1.GetValue(2) != DBNull.Value)
                    {
                        preis = Convert.ToDouble(dr1.GetValue(2));
                    }
                    if (dr1.GetValue(3) != DBNull.Value)
                    {
                        mindestbestand = (int)dr1.GetInt32(3);
                    }
                    if (dr1.GetValue(4) != DBNull.Value)
                    {
                        lieferant = dr1.GetValue(4).ToString();
                    }


                    lstArt.Add(new ArtikelInfo(artikelID, artikelbezeichnung, preis,
mindestbestand,lieferant));
                }
                dr1.Close();

            }
            catch (Exception ex)
            {
                throw ex;
            }
            finally
            {
                con.Close();
            }
            return lstArt;
        }
    }
}
```

3 Fachspezifische Operatoren

Operatoren dienen dazu, die Kommunikation zwischen Aufgabenerstellern und Prüflingen auf eine gemeinsame Verständnisebene zu stellen. Sie sind Handlungsaufforderungen, die Schülertätigkeiten initiieren, lenken und strukturieren. Sie bestimmen die Mittel und Methoden, die ein Schüler wählt, um eine Aufgabenstellung zu bearbeiten. Operatoren machen dem Aufgabenersteller deutlich, ob alle Anforderungsbereiche berücksichtigt wurden.

Die folgende Liste erhebt keinen Anspruch auf Vollständigkeit. Die Operatoren wurden in Anlehnung an die EPA 2007 formuliert. Im Hinblick auf zukünftige Entwicklungen in der Wirtschaftsinformatik sind auch die Operatoren entsprechend erweiterbar.

Operator	**Anforderungs-bereich**	**Definition**	**Beispiel**
anwenden	I – II	einen bekannten Sachverhalt, eine bekannte Methode auf eine neue Problemstellung beziehen	Wenden Sie die Regeln zur Umwandlung eines ER-Diagramm in ein logisches Datenmodell auf das vorgegebene ER-Modell an, um hieraus ein konkretes physikalisches Datenmodell herzuleiten.
begründen	II – III	für einen gegebenen Sachverhalt einen folgerichtigen Zusammenhang zwischen Ursache und Wirkung herstellen	Begründen Sie, warum ein m:n-Beziehungstyp in einem ER-Diagramm zu einer eigenen Relation aufgelöst werden muss.
benennen/ nennen	I	Sachverhalte, Strukturen und Prozesse begrifflich aufführen	Nennen Sie die im abgebildeten Klassendiagramm vorkommenden Assoziationstypen.
berechnen	I – II	mittels charakteristischer Merkmale einen Sachverhalt genau feststellen und beschreiben	Berechnen Sie gemäß dem dargestellten Struktogramm (Aktivitätsdiagramm), für die Eingabe i=1 und j=2 den Ausgabewert x.
beschreiben	I – II	Strukturen, Sachverhalte oder Zusammenhänge strukturiert und fachsprachlich richtig mit eigenen Worten wiedergeben	Beschreiben Sie mit eigenen Worten den Anwendungsfall, der durch das folgende Sequenzdiagramm dargestellt wird.

Operator	**Anforderungs-bereich**	**Definition**	**Beispiel**
bestimmen, ermitteln	II	einen Zusammenhang oder einen möglichen Lösungsweg aufzeigen und das Ergebnis formulieren	Bestimmen Sie, im gegebenen ER-Diagramm die fehlenden Kardinalitäten.
beurteilen	III	den Stellenwert von Sachverhalten oder Prozessen in einem Zusammenhang bestimmen, um kriterienorientiert zu einem begründeten Sachurteil zu gelangen	Beurteilen Sie, ob das beschriebene Datensicherungsverfahren für die angestrebten Ziele ausreichend ist.
bewerten	II – III	zu einem Sachverhalt, zu einem Konzept oder zu einer Problemstellung eine Argumentation entwickeln, die zu einer begründeten Bewertung führt	Bewerten Sie das vorliegende Pflichtenheft hinsichtlich der Vollständigkeit und der Verwendbarkeit als Vertragsgrundlage.
darstellen	II	Zusammenhänge, Sachverhalte, Methoden etc. in strukturierter Form graphisch oder gegebenenfalls fachsprachlich wiedergeben	Stellen Sie den folgenden Anwendungsfall… in einem Sequenzdiagramm dar.
definieren und abgrenzen	II – III	einen Begriff exakt bestimmen, um ihn von anderen abzugrenzen	Definieren Sie die Begriffe Datenschutz und Datensicherung und grenzen Sie sie voneinander ab.
dokumentieren	II	alle notwendigen Erklärungen, Herleitungen und Skizzen darstellen	Dokumentieren Sie den angestrebten Soll-Zustand in einem Anwendungsfalldiagramm mit zugehörigen Anwendungsfallbeschreibungen.
entwerfen/ planen	II – III	Zusammenstellen von Funktionalitäten unter Berücksichtigung vorgegebener Daten	Entwerfen Sie aufgrund der Beschreibung in der Ausgangssituation ein Klassendiagramm für die neu zu entwickelnde Datenbankanwendung zur Auftragsverwaltung.

Operator	Anforderungs-bereich	Definition	Beispiel
entwickeln	II – III	zu einem Sachverhalt oder zu einer Problem-stellung ein konkretes Lösungsmodell oder ein Lösungskonzept nach-vollziehbar skizzieren	Entwickeln Sie zu den gewünschten Statistiken die erforderlichen SQL-Anweisungen.
erklären/ erläutern	I – II	Strukturen, Prozesse und Zusammenhänge erfassen, in Einzelheiten verdeutlichen und durch zusätzliche Informatio-nen verständlich machen	Erläutern Sie, wozu ein try-catch-Befehl genutzt werden kann.
erstellen	II – III	darstellen von Sachver-halten gemäß vorgege-bener Syntax	Erstellen Sie ein Se-quenzdiagramm, das oben geschildertes Szenario abbildet.
erweitern	II – III	eine vorgegebene Struk-tur um Bestandteile er-gänzen	Erweitern Sie das Klas-sendiagramm entspre-chend der dargestellten Situation.
identifizieren/ kennzeichnen	II	das Wesentliche und Typische benennen	Identifizieren Sie die Fremdschlüssel und kenn-zeichnen Sie sie durch …
implementie-ren	II	Algorithmen und Daten-strukturen in eine Pro-grammiersprache um-setzen	Implementieren Sie die Klassen „Kunde“, „Konto“ und „Girokonto“ auf der Basis des gegebenen Klassendiagramms.
modellieren	II – III	zu einem Ausschnitt der Realität ein informati-sches Modell anfertigen	Modellieren Sie den von der Fachkraft beschriebe-nen Anwendungsfall „Auf-tragseingang“ als Se-quenzdiagramm.
Stellung neh-men	III	unter Heranziehung von Kenntnissen differenziert eine eigene begründete Position beziehen	Nehmen Sie Stellung zu den Vor- und Nachteilen zur Nutzung von sozialen Netzwerken wie z. B. „XING“ …
überprüfen/ testen	II	Sachverhalte, Probleme, Fragestellungen nach bestimmten fachlich übli-chen Kriterien untersu-chen	Testen Sie die implemen-tierte Methode mit Hilfe einer geeigneten GUI-Anwendung.
vergleichen	II – III	Gemeinsamkeiten, Ähn-lichkeiten und Unter-schiede ermitteln	Vergleichen Sie die unter-schiedlichen Implementie-rungen der Methode „aus-zahlen“.

4 Ergänzende Hinweise

Literatur

- David J. Barnes, Michael Kölling: Java lernen mit BlueJ. Eine Einführung in die objektorientierte Programmierung. München u. a. 2006.
- Prof. Hermann Meyer, Karsten Schleider: Objektorientierung leicht gemacht. Troisdorf 2007.
- Christoph Kecher: UML 2. Das umfassende Handbuch. Bonn 2009.
- Peter Loos: Objektorientiertes Programmieren in Visual C#. Eine methodische Einführung für Einsteiger und Fortgeschrittene. Unterschleißheim 2006.
- Helmut Balzert: Java 5: Objektorientiert programmieren. Vom objektorientierten Analysemodell bis zum objektorientierten Programm. Herdecke, Bochum 2006.