Objektorientierte Programmierung mit Java

Fallbeispiele und Anwendungsfälle mit betriebswirtschaftlichem Hintergrund

0. Einführung
1. Autovermietung
2. Kassenumsätze (BALDI)
3. Firmenumsatz (OPAL)
4. Kunde und Betreuer
5. Kontenverwaltung in der Bank (Vererbung, Konstruktoren mit super())
6. Geldautomat (DOS)
7. Geldautomat (Windows)
8. Auftragsstatistik
9. Wer liefert was? (Verweis auf Objekte in einem Array)
10. Platzreservierung in der Bahn
11. Telefonbucheinträge (Array mit Objekten)
12. Photovoltaikanlagen
13. Pizza-Service
14. Platzreservierung
15. Das Spiel: Hol's der Geier

Legende
Standardschrift: Arial 11

Klassen	Schriftart 12, fett
Attribute	fett
Methoden	fett, kursiv
Objektnamen	kursiv
Attributsausprägungen	kursiv
Applikation	Courier new, fett

verwendet Software.

Java-Editor
Visio
Sequenzdiagramm Editor
Strukted

www.pellatz.de

1. Fallbeispiel: **Autovermietung**

Inhalte Teil A: Klassen, Attribute, Methoden, Klassendiagramme, Konstruktor, Vererbung
Teil B:

Eine Autovermietung vermietet Fahrzeuge an Kunden. Wenn ein Kunde ein Auto mietet, hat dies eine Buchung zur Folge. Eine Buchung kann sich nur auf ein Fahrzeug beziehen.

Aufgaben:

Teil A:

1. Erstellen Sie für die dargestellte Situation ein Klassendiagramm mit Assoziationen und Kardinalitäten.

2. Programmieren Sie die Klasse Fahrzeug mit folgenden Attributen.
   Kennzeichen, Marke, Farbe, Verbrauch, kw, kmStand, maxTankinhalt, aktTankinhalt und **Mietsatz**.
   Alle Attribute sind als *private* und mit einem passenden Datentyp zu deklarieren.
   Außerdem ist für jedes Attribut eine ***set***- und eine ***get-Methode*** zu erstellen.

3. Legen Sie einen Konstruktor an, für die Initialisierung aller Attribute.

4. Erstellen Sie eine neue Applikation **`auto1`**, erzeugen Sie zwei Instanzen *a1* und *a2* und lassen Sie die Attributwerte der beiden Objekte wieder ausgeben.
   Die Ausgabe der Attributwerte kann mit einer Methode gelöst werden.

5. Die Klasse **Auto** soll zwei zusätzliche Methoden erhalten. Die Methode ***tanken*** mit dem Übergabeparameter *liter* und die Methode ***fahren*** mit dem Übergabeparameter *km*.

   Die Methode ***tanken*** verändert den aktuellen Tankinhalt. Dabei ist darauf zu achten, dass der maximale Tankinhalt nicht überschritten werden darf.
   Die Methode ***fahren*** verändert den kmStand sowie den aktuellen Tankinhalt.

6. Es soll eine neue Klasse **LKW** erzeugt werden, die eine von der Klasse Auto abgeleitete Klasse ist. Die Klasse LKW erhält als zusätzliches Attribute das **maxLadegewicht** und das **aktLadegewicht** (in kg). Die Methode ***beladen*** erhält als Übergabeparameter die Zuladung (in kg) und soll ein Beladen bis zur Höhe des maximalen Ladegewichts ermöglichen.
   Erzeugen Sie eine Instanz von dieser Klasse und lassen Sie die initialisierten Werte in der Applikation **`auto2`** wieder ausgeben. Testen Sie die Methode ***beladen*** aus.

Teil B:

In stark vereinfachter Form sollen nun die Anwendungsfälle **Kunde bucht Fahrzeug**, **Kunde gibt Fahrzeug zurück** und **Anzeige aller Buchungen** realisiert werden.

1. Erzeugen Sie die beiden Klassen **Kunde** und **Buchung**. Die Klasse Kunden hat die Attribute **kdnr** und **kName**, die Klasse Buchung die Attribute **bnr**, **Kennzeichen** und **Dauer**.

2. In der Applikation **`auto3`** wird zunächst der Anwendungsfall **Kunde bucht Fahrzeug** realisiert. Dabei werden 2 Instanzen der Klasse Auto und eine Instanz der Klasse Kunden erzeugt. In einem Bildschirmdialog wird der Kunde mit Namen begrüßt, die zur Verfügung stehenden Fahrzeuge angezeigt und der Kunde wird aufgefordert, das Kennzeichen des zu buchenden Fahrzeuges sowie die Mietdauer einzugeben. Der Kundenmethode ***buchen*** werden die Parameter Buchungsnummer, Kennzeichen und Mietdauer übergeben. Die Methode erzeugt sodann eine Instanz der Klasse **Buchung.**

2. Fallbeispiel**: Kassenumsätze (BALDI)**

Inhalte: Klassen, Attribute, Methoden, Klassendiagramme, Konstruktor

Die Zentrale des Lebensmittelunternehmens **BALDI** verwaltet die beiden Filialen **Neustadt** und **Burghausen**. Die Filiale Neustadt hat 3 Kassen (**K1**, **K2** und **K3**), die Filiale Burghausen hat die beiden Kassen **K4** und **K5**. In den Kassen werden die Kassenbezeichnung, die morgendliche Bargeldeinlage und der aktuelle Kassenbestand digital gespeichert.
Die Filialen können die Daten ihrer Kassen per EDV verarbeiten. Die Zentrale kann auf die Daten der Filialen zugreifen, nicht aber auf die Daten der Kassen vor Ort. Der Firmeninhaber wünscht, dass er von der Zentrale aus jeden Abend den gesamten Tagesumsatz des Unternehmens feststellen kann.
Erstellen Sie eine objektorientierte Software für die Zentrale.

Aufgaben

2. Erstellen Sie ein Klassendiagramm mit Assoziationen und Kardinalitäten. Attribute und Methoden müssen nicht dargestellt werden.

3. Legen Sie die Klasse **Kasse** anhand des vorliegenden Klassendiagramms an und erstellen Sie zusätzlich ein Objektdiagramm für eine bestimmte Kasse.

4. Die Klassen **Zentrale** und **Filiale** liegen im Ordner /Baldi in codierter Form vor. Erstellen Sie anhand des Quellcodes ein detailliertes Klassendiagramm für jede der beiden Klassen.

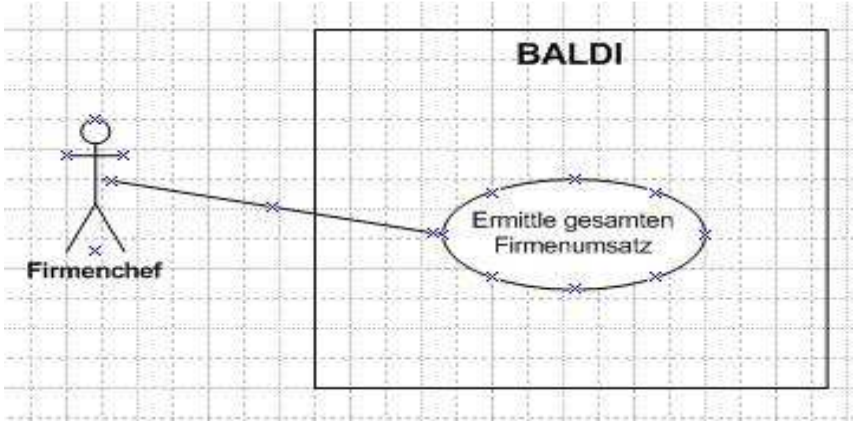
5. Die Attribute in der Klasse Kasse sollen mit einem **Konstruktor** initialisiert werden. Realisieren Sie diesen Konstruktor.

6. Erstellen Sie die Applikation **`Kassenumsatz1`**, in der alle Objekte erzeugt werden und die notwendigen Initialisierungen vorgenommen werden. Lassen Sie die Attributswerte am Bildschirm anzeigen.

7. In der Klasse Kasse soll eine Methode ***getTagesumsatz*** realisiert werden, die den Tagesumsatz berechnet und das Ergebnis als Rückgabewert liefert.

8. Berechnen Sie den Umsatz einer Filiale in der Applikation **`Kassenumsatz2`**. Übergabewert für die Methode ***setFilialumsatz*** ist der Tagesumsatz einer Kasse. Der Gesamtumsatz einer Filiale soll am Bildschirm angezeigt werden.

9. Realisieren Sie nun den Anwendungsfall „gesamten Firmenumsatz ermitteln“.



3. Fallbeispiel: **Firmenumsatz (OPAL)**

Inhalte: Klassen, Attribute, Methoden, Klassendiagramm, Vererbung, Verweisattribute

Der Autohersteller OPAL hat zwei Werke in *Dortmund* und in *Essen*, die zentral von der Firma in Köln verwaltet werden. Jedes Werk stellt einen bestimmten Fahrzeugtyp her. In Dortmund wird der *Saphir* hergestellt, in Essen der *Smaragd*. Für jeden Fahrzeugtyp wird der Stückpreis und die Verkaufsmenge gespeichert.

Aufgaben

1. Erstellen Sie ein Klassendiagramm mit Assoziationen und Kardinalitäten

2. Legen Sie die notwendigen Klassen in Java mit folgenden Attributen an:
   Firma: **Name** (String)
   Werk: **Standort** (String),
   Produkt: **Bezeichnung** (String), **Preis** (double), **Verkaufsmenge** (int)
   (zunächst werden keine Verweisattribute benötigt)

   Für jedes Attribut soll eine get- und eine set-Methode existieren.

3. Erzeugen Sie alle genannten Objekte und initialisieren Sie die Attribute mit den genannten Daten sowie mit folgenden Werten:
   Saphir: **Preis** *20 000* €, **Verkaufsmenge** *15 0*
   Smaragd: **Preis** *16 500* €, **Verkaufsmenge** *220*
   Die Initialisierungen können auch mit einem Konstruktor erfolgen. Lassen Sie alle Daten am Bildschirm ausgeben (verwenden Sie dazu evtl. die toString-Methode).
   Die Applikation soll die Bezeichnung **`Umsatz1`** tragen.

4. Es wird ein drittes Werk in *Siegen* eröffnet, in dem LKWs vom Typ *Mammut* hergestellt werden. Die Klasse **LKW** ist von der Klasse Produkt abgeleitet und erhält ein zusätzliches Attribut mit der Bezeichnung **Zuladung** (int)**.** Erzeugen Sie diese Klasse und ein Objekt von dieser Klasse sowie ein neues Werk in der Applikation **`Umsatz2`**.

5. Wir erzeugen die Assoziationen zwischen den Klassen Werk und Produkt. Die Klasse Werk bekommt ein zusätzliches Attribut **meinProdukt** vom Typ **Produkt**, Codieren Sie dieses Attribut und erstellen Sie dazu eine get- und set-Methode. Initialisieren Sie die Attribute in der Applikation **`Umsatz3`**.

6. Die Klasse Werk soll eine Methode bekommen, mit der der Umsatz des Werkes berechnet wird. Das Ergebnis wird in dem neuen Attribut **WerksUmsatz** (double) gespeichert. Realisieren Sie diese Methode ***bWerksUmsatz*** und lassen sie alle Werksumsätze für die drei Werke in der Applikation **`Umsatz4`** berechnen.

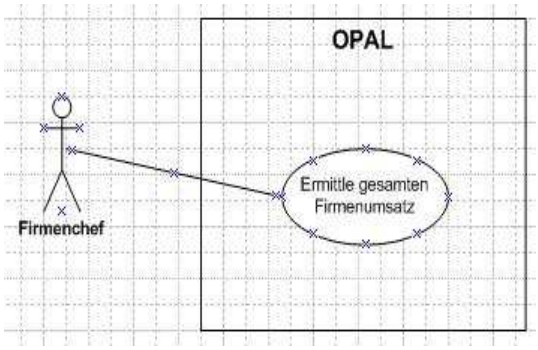
7. Der Firmeninhaber möchte eine Übersicht über den Umsatz der drei Werke.

   Realisieren Sie den Anwendungsfall ***BerechneFirmenumsatz*** als Methode der Klasse **Firma** anhand des vorliegenden **Sequenzdiagrammes** in der Applikation **`Umsatz5`**. Es soll dabei folgende Ausgabe (je nach Zahlen) am Bildschirm erzeugt werden:

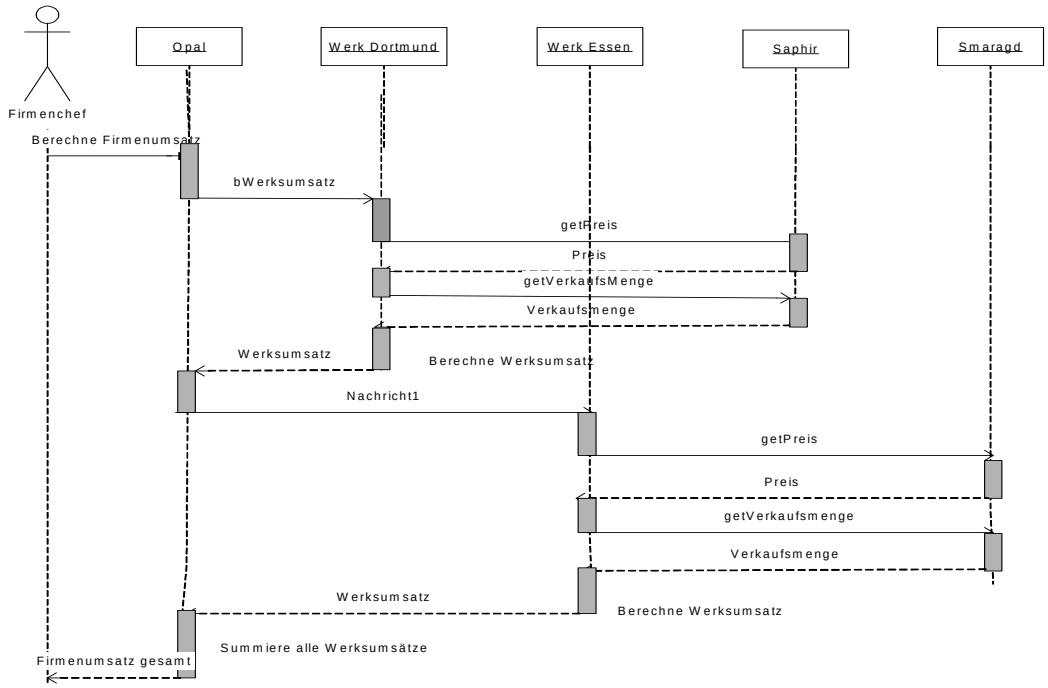
```
Werk Dortmund: Produkt Saphir, Umsatz 3000000 Euro
Werk Essen: Produkt Smaragd, Umsatz 36300000 Euro
Werk Siegen: Produkt Mammut, Umsatz 2400000 Euro

Gesamtumsatz der Firma OPAL  9030000 Euro
```

   (Sie können die Methode bWerksUmsatz in der Klasse Firma ohne entsprechendes Verweisattribut direkt mit der Objektbezeichnung aufrufen. Preis und Verkaufsmenge werden aber mit dem entsprechenden Verweisattribut (siehe Aufg. 5) angesprochen.)

Anwendungsfalldiagramm für den Anwendungsfall **„Berechne Firmenumsatz“**



Sequenzdiagramm für die Methode ***berechneFirmenumsatz***
(zur Vereinfachung ist hier das Werk Siegen nicht mit aufgeführt



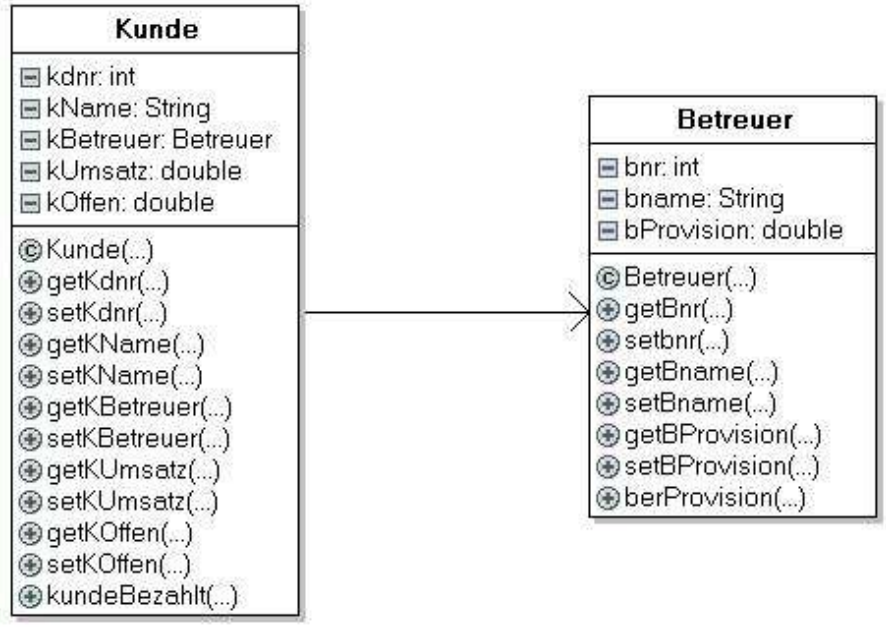
4. Fallbeispiel: **Kunde und Betreuer**

Inhalte: Klassen, Attribute, Methoden, Verweisattribut, Konstruktor, Tastatureingabe, Array mit Objekten, switch..case

Jeder Kunde eines Unternehmens wird von einem Kundenbetreuer betreut. Wenn der Kunde einen Auftrag erteilt, bekommt der Betreuer eine Provision von 5% der vom Kunden bezahlten Auftragssumme. Bei Autragserteilung wird eine Rechnung erstellt und das Attribut Offene Posten beim Kunden um den Rechnungsbetrag erhöht. Jeder Zahlungseingang des Kunden vermindert die Offenen Posten und führt zu einer Provisionsberechnung beim Betreuer. Eine Software soll die Anwendungsfälle **Kunde erteilt Auftrag**, **Kunde bezahlt Rechnung** und **Betreuerprovision anzeigen** realisieren.

Aufgaben

1. Erstellen Sie die Klassen Kunde und Betreuer nach folgendem Klassendiagramm:

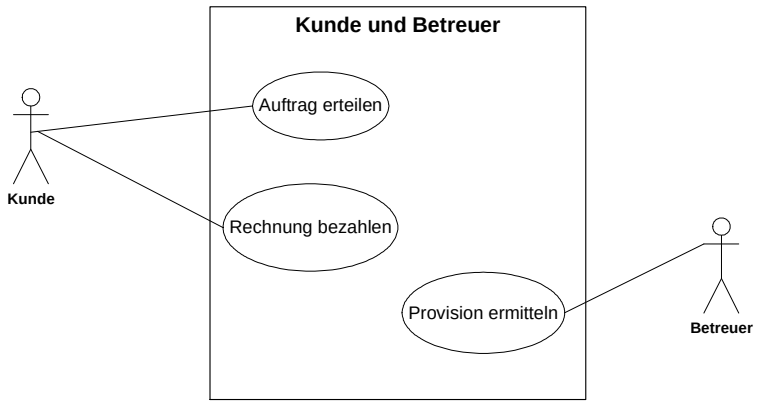


Die Methoden ***kundeBezahlt*** und ***berProvision*** können zunächst noch weggelassen werden. Erzeugen Sie mindestens jeweils drei Kunden und Betreuer über einen Konstruktor. Die Attributwerte für **kUmsatz**, **kOffen** und **bProvision** werden mit dem Wert 0 initialisiert. Damit Kunden oder Betreuer ausgewählt werden können, sollen die Objekte der beiden Klassen in einem Array gespeichert werden. Die Syntax dafür lautet:

```
//Arrays für Klassenobjekte erzeugen
Kunde[] k = new Kunde[5];
Betreuer [] b = new Betreuer[5];
//Objekte erzeugen und intialiseren
k[0]= new Kunde(101,"Meier");
b[0]= new Betreuer(1001,"Hans");
k[0].setKBetreuer(b[0]);
```

2. Realisieren Sie in die Applikation **`Kundenbetreuung1`** den Anwendungsfall **KundeErteiltAuftrag**.

3. Erstellen Sie die Applikation **`Kundenbetreuung2`**, die die folgenden Anwendungsfälle realisiert.
   Die Anwendungsfälle sollen mit Hilfe eines Auswahlmenüs über ein switch..case-Konstrukt angesteuert werden.



Sofern die einzelnen Anwendungsfälle über eine eigen Prozedur aufgerufen werden, müssen die Arrays für die Klassen Kunden und Betreuer übergeben werden.

Aufruf: auftrag(k, b) → **Prozedurkopf**: public static void auftrag(Kunde[ ] k, Betreuer [ ] b)

5. Fallbeispiel: **Kontenverwaltung in der Bank**

Inhalte: Klassen, Attribute, Methoden, Verweisattribut, Konstruktor, Tastatureingabe, Array mit Objekten, switch..case, Vererbung

Eine Bank hat genau drei Kunden. Jeder Kunde besitzt ein Konto. Es sollen folgende Anwwendungsfälle realisiert werden: **Akteur Kunde**: Kontostand abfragen, Geld einzahlen, Geld abheben **Akteur Bank:** Dispo für einen Kunden ändern

Gegeben ist das detaillierte Diagramm für die einzelnen Klassen.

1. Erstellen Sie ein Klassendiagramm mit allen Klassen sowie Assoziationen und Kardinalitäten.
2. Codieren Sie die Klassen anhand der Klassendiagramme
3. Zeichnen Sie ein Anwendungsfalldiagramm
4. Realisieren Sie die Anwendungsfälle (use-cases) durch Ergänzen von Methoden in den Klassen und testen sie diese in einer Applikation

5. Der Anwendungsfall xx wird durch das folgende Sequenzdiagramm dargestellt. Realisieren Sie diese Methode und testen sie diese in einer Applikation.

6. Fallbeispiel: **Geldautomat (DOS)**

Inhalte: Klassen, Attribute, Methoden, Verweisattribut, Konstruktor, Tastatureingabe, Array mit Objekten, switch..case

Phase: Design und Codierung

Es soll die Funktionsweise für einen Geldautomat für die MWS-Bank programmiert werden. Der Automat kann ohne Magnetkarte bedient werden. Zur Sicherheit muss sich der Kunde mit zwei Zahlen identifizieren. Zum einen mit der Kundennummer und danach mit der Geheimzahl seines Kontos.

Zum Testen der Funktionsweise soll zunächst ein Prototyp des Geldautomaten auf der DOS-Ebene erstellt werden, dessen Eingangsbildschirm nach korrekter Eingabe von Kundennummer und Geheimzahl folgendes Aussehen haben soll.

Prototyp des Geldautomaten (DOS-Fenster)

```
B A N K O M A T                MWS-Bank          Standort: Max-Weber-Schule


Bitte Kundennummer eingeben:   1234

Guten Tag Frau Wagner !


Bitte Geheimzahl eingeben:     222

Sie haben Zugriff auf Ihr Konto mit der Nummer 100200

Wählen Sie

1 – Kontostand abfragen
2 – Einzahlen
3 – Abheben
4 – System verlassen

Eingabe:
```

1. Erstellen Sie diesen Bildschirm zunächst noch ohne Funktionalität.
   Kundennummer und Geheimzahl soll über die Tastatur eingegeben werden.
2.
3.

7. Fallbeispiel: **Geldautomat (Windows)**

8. Fallbeispiel: **Auftragsstatistik**

In einem Unternehmen soll die Auftragsbearbeitung mit einem neuen Programm erledigt werden. In einem Klassendiagramm soll zunächst die Struktur des Systems modelliert werden. Die Grundlage dazu liefern folgende Informationen:

Wenn ein Kunde einen Auftrag erteilt, werden die einzelnen Auftragspositionen erfasst.
Bei jeder Auftragsposition wird anhand der Produktdaten die Lieferfähigkeit überprüft.
Nach Abschluss der Auftragserfassung wird eine Rechnung generiert.

Aufträge werden mit Auftragsnummer, Datum und Auftragssumme erfasst.
Für Kunden werden die Kundennummer, Name und Anschrift gespeichert. Auftragspositionen Beinhalten die Positionsnummer, Artikelnummer und Menge.
Die Produktdaten speichern Artikelnummer, Artikelbezeichnung, Bestand und Preis.
Rechnungen beinhalten die Rechnungsnummer und das Rechnungsdatum.

a) Erstellen Sie zunächst das **Klassendiagramm.**
b) Wie werden die **Assoziationen** zwischen den Klassen realisiert?
c) Stellen Sie den Anwendungsfall „Auftragsstatistik erstellen“ in einem **Sequenzdiagramm** dar.
   Als Ergebnis der Auftragsstatistik soll täglich eine Liste mit den Aufträgen des Tages erstellt werden nach folgendem Muster:

Auftragsnummer	**Kundennummer**	**Kundenname**	**Auftragssumme**
080703001	2007	Meier KG	559,50
080703002	2003	Rinn und Keil	1300,00
080703003	2013	Loose GmbH	712,50
080703004	2003	Rinn und Keil	79,95

9. Fallbeispiel: **Wer liefert was?**

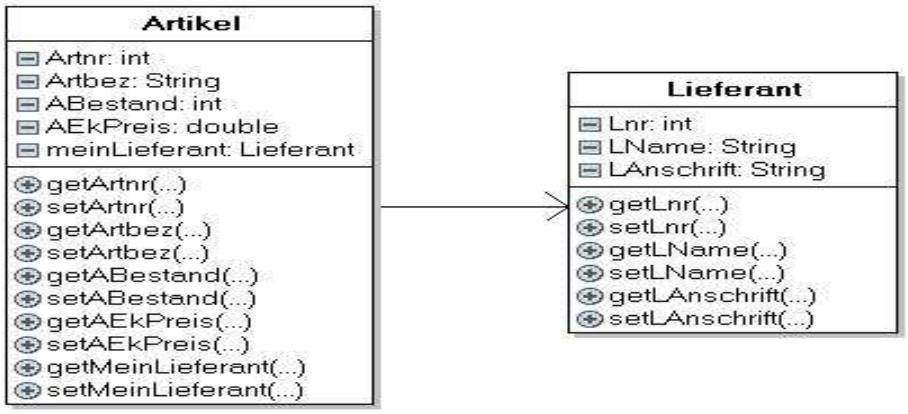
Inhalte: Klassen, Attribute, Methoden, Verweisattribut, Konstruktor, Tastatureingabe, Array mit Objekten

Jeder Artikel eines Handelsunternehmens hat genau einen Lieferanten.
Die Geschäftsleitung möchte folgende Anwendungsfälle realisieren:

- Ausgabe einer tabellarischen Artikelliste und einer tabellarischen Lieferantenliste.
- Ausgabe des gesamten Lagerwertes aller Artikel.
- Anzeige des Lieferanten anhand einer eingegebenen Artikelnummer

Aufgaben

1. Erstellen Sie ein Klassendiagramm
2. Erstellen Sie ein Use-case-Diagramm für die genannten Anwendungsfälle
3. Codieren Sie die beiden Klassen anhand des vorliegenden Klassendiagramms.



4. Erstellen Sie die Applikationen **Artikelliste** und **Lieferantenliste**.
   Erzeugen Sie die Instanzen für die Klasse Artikel und die Klasse Lieferanten über ein Array. Es sollen mindestens 5 Artikel und 3 Lieferanten über einen Konstruktor erzeugt werden.
5. Erstellen Sie die Applikation **LagerwertErmitteln.** Die Auswertung der Objekte erfolgt mit einer for-Schleife. Zeichnen Sie ein **Sequenzdiagramm** für diese Applikation.
6. Erstellen Sie die Applikation **werLiefertWas**. Damit sollen anhand der Artikelnummer, die über die Tastatur eingegeben wird, die Daten des Lieferanten angezeigt werden. Die Eingabe soll so lange wiederholt werden, bis als Artikelnummer 0 eingegeben wird. Falls eine Artikelnummer nicht vorhanden ist, soll die Meldung „Artikel nicht vorhanden“ angezeigt werden.
7. Fortführung: Speichern der Objekte in einem Container der Form ArrayList. Verarbeiten der Artikeldaten und Lieferantendaten in einer serialisierten Datei.

10. Fallbeispiel: **Platzreservierung in der Bahn**

Ein Zug besteht aus Waggons. Jeder Waggon hat genau 40 Sitzplätze. Ein Sitzplatz kann durch eine Buchung reserviert sein.

Aufgaben

1. Erstellen Sie für die dargestellte Situation ein Klassendiagramm mit Assoziationen und Kardinalitäten

2. Folgende Attribute werden benötigt:

   Zug: **Zugnummer**
   Waggon: **Waggonnummer**
   Sitzplatz: **Platznummer**, **Belegvermerk**
   Buchung: **Name**, **Zugnumme**r, **Waggonnummer**, **Platznummer, Startort, Zielort**

3. Erstellen Sie alle Klassen mit den genannten Attributen. Verwenden Sie Konstruktoren, für die Klassen **Zug**, **Waggon** und **Sitzplatz**. Im Konstruktor für die Klasse **Waggon** werden automatisch 40 Sitzplatzobjekte erzeugt, die von 1 bis 40 nummeriert werden. Das Attribut **Belegvermerk** hat den Datentyp boolean und wird im Konstruktor auf *false* gesetzt.

4. Es müssen die notwendigen Verweisattribute ergänzt werden.

5. Das Buchungssystem soll folgende Anwendungsfälle realisieren:

   - Zug konfigurieren: Ein neuer Zug wird erzeugt, es wird eine bestimmte Anzahl Waggons angehängt:
   - Platz buchen: Ein neues Buchungsobjekt wird erzeugt, der Kundenname eingetragen, Waggonnummer und Platznummer vermerkt. Es wird geprüft, ob der Platz frei ist. Wenn ja, wird der Belegvermerk auf *true* gesetzt.
   - Leere Plätze in einem Waggon anzeigen:

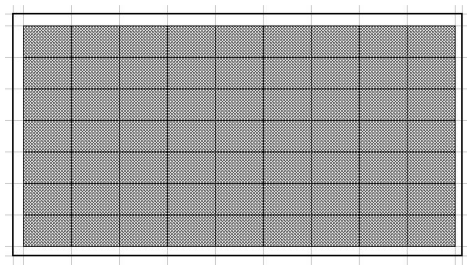
6. Erstellen Sie ein Anwendungsfalldiagramm. Überlegen Sie, wer für die genannten Anwendungsfälle jeweils der Akteuer ist.

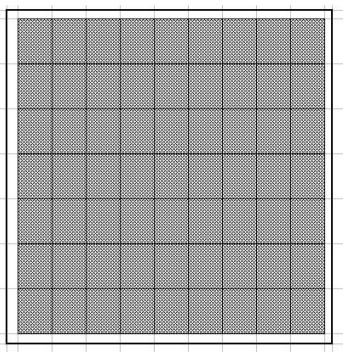
7. Ergänzen Sie das System um weitere, sinnvolle Anwendungsfälle.

12. Fallbeispiel: **Installation von Photovoltaikanlagen**

Peter Cornelius ist Mitarbeiter der Firma ***Sunpower***, die Photovoltaikanlagen installiert. Eine Photovoltaikanlage besteht aus einzelnen Solarmodulen, die auf das Dach montiert werden. Durch die Sonneneinstrahlung wird Strom erzeugt, der in das Stromnetz eingespeist wird und von den Stromgesellschaften nach dem Energieeinspeisungsgesetz in bestimmter Höhe vergütet werden muss. Je größer die Dachfläche, desto mehr Module können installiert werden und um so höher ist die Stromerzeugung.
Es gibt Module verschiedener Hersteller, die unterschiedliche Abmessungen haben.
Bei der Auswahl der geeigneten Module geht es in erster Linie darum, die Dachfläche optimal auszunutzen, so dass die Nutzfläche am größten ist.
Diese Frage kann Peter Cornelius im Kundengespräch nie sofort beantworten, da er für die Berechnung der optimalen Modulanordnung eine gewisse Zeit benötigt.

Ihre Aufgabe ist es, eine Software zu entwickeln, die je nach Dachabmessungen die geeigneten Module so auswählt, dass die nutzbare Dachfläche möglichst groß wird.
(Die Module können horizontal oder vertikal installiert werden)





Beispiele für horizontale und vertikale Installation der Solarmodule

<u>Aufgaben</u>

Wir gehen davon aus, dass die Dachfläche rechteckig ist und keine Besonderheiten (Fenster, Gauben usw.) aufweist.
Der Benutzer gibt die Maße des Daches ein und erhält als Antwort den geeigneten Modultyp sowie die Anzahl der Module und die maximal belegbare Dachfläche. Es ist zu berücksichtigen, dass am Rand des Daches immer 10 cm frei bleiben müssen.

Es stehen folgende Module zur Auswahl:

1. Alfasolar	1475 mm x 986 mm,
2. Sunpower	1560 mm x 800 mm
3. Conerergy	1490 mm x 1000 mm
4. Mitsubishi	1250 mm x 800 mm
5. Kyocera	1225 mm x 1225 mm
6. Solarworld	1675 mm x 1000 mm

1. Erstellen Sie eine Klasse **Modul** und schreiben Sie eine Applikation, die diese Aufgabe löst.
2. Peter Cornelius möchte die Software als **Java-Applet** auf der Homepage der Firma Sunpower installieren, damit der Kunde die geeigneten Module schon vorher auswählen kann. Lösen Sie das Problem.

13. Fallbeispiel: **Pizza-Service Bella Italia**

Der Pizza-Service Bella Italia bietet seinen Kunden Pizzen in zwei Größen mit 12 unterschiedlichen Belägen an, von denen der Kunde beliebig viele für eine Pizza auswählen kann. Es gibt drei Preiskategorien für Beläge:

Preiskategorie 1	Paprika
	Peperoni
	Knoblauch
	Oliven
	Zwiebeln
Preiskategorie 2	Salami
	Schinken
	extra Käse
	Ananas
Preiskategorie 3	Gorgonzola
	Muscheln
	Shrimps

	28 cm	**32 cm**
Grundpizza	4,00	4,50
Preiskategorie 1	0,50	0,70
Preiskategorie 2	1,00	1,30
Preiskategorie 3	1,20	1,60

Die Grundpizza enthält immer Tomaten und Käse

Ihre Aufgabe ist es, ein Softwaresystem zu entwickeln, das einen Pizzamitarbeiter bei der Annahme der Bestellungen gut unterstützt. Dazu soll es alle für diesen Zweck notwendigen Daten verarbeiten können. Am Ende der Bestellung soll der Preis für die Pizza angezeigt werden.
(Diese Aufgabe war eine von 5 Aufgaben des Bundeswettebewerbs für Informatik 2009.)

---

1. Beschreiben Sie, wie der Vorgang der Bestellannahme mit Hilfe des Systems ablaufen soll. Wählen Sie dazu eine geeignete graphische Darstellung.

2. Entwerfen Sie für das System eine Benutzerschnittstelle und stellen Sie diese graphisch dar.

3. Entwickeln Sie ein Datenmodell für das System in Form eines Klassendiagramms

4. Realisieren Sie die Kernfunktionalität des Systems

14. Fallbeispiel: **Platzreservierung (Wer sitzt wo?)**

In einer Sportarena gibt es Blöcke, Reihen und Sitzplätze. Besucher können freie Plätze reservieren. Es soll ein System entwickelt werden, welches die freien Plätze anzeigt und namentliche Platzreservierungen vornehmen kann.
Realisieren Sie diese Anwendung in einem Beispiel mit mindestens 30 Plätzen (2 Blöcke mit je 3 Sitzreihen und 5 Plätzen pro Sitzreihe.

Weitere Überlegungen

Es können die Klassen **Platz** und **Besucher** angelegt werden.
Attribute für die Klasse Platz sind die Platzid, Block, Reihe und Platznummer sowie ein Verweisattribut für den Besucher.
So lange der Platz nicht belegt ist, ist das Verweisattribut mit frei zu bezeichnen.

Für den Besucher können Name, Vorname und Anschriftdaten erfasst werden.
Der Name genügt aber für die Funktionalität. Außerdem ein Verweisattribut auf den gebuchten Platz. Dabei kann ein Besucher auch mehrere Plätze gebucht haben (Erweiterung)

Die Applilakation kann in Form eines Menüs dargestellt werden.
1. Alle Plätze anzeigen
2. Freie Plätze anzeigen
3. Platz buchen
4. Besucher und seine gebuchten Plätze anzeigen.
5. Buchung stornieren.

Evtl. weitere

15. Fallbeispiel: **Das Spiel: Hol’s der Geier**

4. Lösungsteil

1. Fallbeispiel: Autovermietung

Teil A:

```
public class auto {
  private String Kennzeichen;
  private String Marke;
  private String Farbe;
  private double Verbrauch;
  private double kw;
  private int kmStand;
  private double maxTankinhalt;
  private double aktTankinhalt;

  // Anfang Ereignisprozeduren
  public auto(String k, String m, String f, double v, double kw, int km, double mt, double at)
  {
    this.Kennzeichen=k;
    this.Marke=m;
    this.Farbe=f;
    this.Verbrauch=v;
    this.kw=kw;
    this.kmStand=km;
    this.maxTankinhalt=mt;
    this.aktTankinhalt=at;
  }
  public String getKennzeichen() {
    return Kennzeichen;
  }

  public void setKennzeichen(String Kennzeichen) {
    this.Kennzeichen = Kennzeichen;
  }

  public String getMarke() {
    return Marke;
  }

  public void setMarke(String Marke) {
    this.Marke = Marke;
  }

  public String getFarbe() {
    return Farbe;
  }

  public void setFarbe(String Farbe) {
    this.Farbe = Farbe;
  }

  public double getVerbrauch() {
    return Verbrauch;
  }

  public void setVerbrauch(double Verbrauch) {
    this.Verbrauch = Verbrauch;
  }
```

```
 public double getKw() {
  return kw;
 }

 public void setKw(double kw) {
  this.kw = kw;
 }

 public int getKmStand() {
  return kmStand;
 }

 public void setKmStand(int kmStand) {
  this.kmStand = kmStand;
 }

 public double getMaxTankinhalt() {
  return maxTankinhalt;
 }

 public void setMaxTankinhalt(double maxTankinhalt) {
  this.maxTankinhalt = maxTankinhalt;
 }

 public double getAktTankinhalt() {
  return aktTankinhalt;
 }

 public void setAktTankinhalt(double aktTankinhalt) {
  this.aktTankinhalt = aktTankinhalt;
 }
 public void ausgabe()
 {
  System.out.println("Kennzeichen:    "+this.Kennzeichen);
  System.out.println("Marke:          "+this.Marke);
  System.out.println("Farbe:          "+this.Farbe);
  System.out.println("Verbrauch:      "+this.Verbrauch);
  System.out.println("kw:             "+this.kw);
  System.out.println("Kilometerstand: "+this.kw);
  System.out.println("Tankinhalt max: "+this.maxTankinhalt);
  System.out.println("Tankinhalt akt: "+this.aktTankinhalt);

 }
  // Ende Ereignisprozeduren
}


public class auto1
{
 public static void main(String[] args)
 {
        //2 Instanzen der Klasse auto erzeugen
        auto a1 = new auto("GI-JP 111","Renault","rot",6.2,90,32500,60,32);
        auto a2 = new auto("GI-VH-200","VW","schwarz",7.5,75,65000,50,40);
        //Verändern einiger Attribute
        //Werte anzeigen
        System.out.println("\nAusgabe a1\n");
        a1.ausgabe();
        System.out.println("\nAusgabe a2\n");
        a2.ausgabe();
 }
}
```

```
public class lkw extends auto {

  // zusätzliche Attribute
  private int maxLadegewicht;      //maximal zulässige Zuladung
  private int aktLadegewicht;      //aktuelles Ladegewicht

  //Methoden
  public lkw(int max,int akt)       //Konstruktor
  {
    this.maxLadegewicht=max;
    this.aktLadegewicht=akt;
  }
  public void setMaxLadegewicht(int kg) {
    this.maxLadegewicht = kg;
  }
  public int getMaxLadegewicht()   {
    return this.maxLadegewicht;
  }

  public void setAktLadegewicht(int kg) {
    this.aktLadegewicht = kg;
  }

  public int getAktLadegewicht() {
    return this.aktLadegewicht;
  }

  public void beladen(int kg) {
    if(this.aktLadegewicht+kg>this.maxLadegewicht)
    {
       this.aktLadegewicht=this.maxLadegewicht;
    }
    else
    {
      this.aktLadegewicht = this.aktLadegewicht+kg;
    }
  }
  public void ausgabe()
  {
        System.out.println("Maximales Gewicht: "+this.maxLadegewicht);
        System.out.println("Aktuelles Gewicht: "+this.aktLadegewicht);
  }
}

public class auto2
{
  public static void main(String[] args)
  {
          //1 Instanzen der Klasse lkw erzeugen
          lkw l1 = new lkw(30000,12000);
          //Werte anzeigen
          System.out.println("\nAusgabe l1\n");
          l1.ausgabe();
  }
}
```

Teil B:

```
public class Kunde {

  // Anfang Variablen
  private int Kdnr;
  private String KName;
  private Buchung meineBuchung;
  // Ende Variablen

  // Anfang Ereignisprozeduren
  public Kunde(int k, String n)
  {
    this.Kdnr=k;
    this.KName=n;
  }

  public int getKdnr() {
    return Kdnr;
  }

  public void setKdnr(int Kdnr) {
    this.Kdnr = Kdnr;
  }

  public String getKName() {
    return KName;
  }

  public void setKName(String KName) {
    this.KName = KName;
  }
  public void buchen(int bnr, String kz,int d)
  {
    Buchung b1=new Buchung(bnr,kz,10);
    this.meineBuchung=b1;
  }
  public Buchung getMeineBuchung() {
    return meineBuchung;
  }

  public void setMeineBuchung(Buchung meineBuchung) {
    this.meineBuchung = meineBuchung;
  }

  // Ende Ereignisprozeduren
}
```

```
public class Buchung {

  // Anfang Variablen
  private int bnr;
  private String kennz;
  private int Dauer;
  // Ende Variablen

  // Anfang Ereignisprozeduren
  public Buchung(int b, String k, int d)
  {
    this.bnr=b;
    this.kennz=k;
    this.Dauer=d;
  }
  public int getBnr() {
    return bnr;
  }

  public void setBnr(int bnr) {
    this.bnr = bnr;
  }

  public String getKennz() {
    return kennz;
  }

  public void setKennz(String kennz) {
    this.kennz = kennz;
  }

  public int getDauer() {
    return Dauer;
  }

  public void setDauer(int Dauer) {
    this.Dauer = Dauer;
  }

  // Ende Ereignisprozeduren
}

import java.io.*;
public class auto3   //Anwendungsfall Kunde bucht Fahrzeug
{
  public static void main(String[] args) throws IOException
  {
     //2 Instanzen der Klasse auto erzeugen
     auto a1 = new auto("GI-JP 111","Renault","rot",6.2,90,32500,60,32,40);
     auto a2 = new auto("GI-VH 200","VW","schwarz",7.5,75,65000,50,40,35);
     //1 Kunden erzeugen
     Kunde k1 = new Kunde(100,"Meier");
     System.out.println("Guten Tag, Herr "+k1.getKName()+"! Es stehen folgende Fahrzeuge zur
Verfügung:\n ");
     System.out.println(a1.getMarke()+", Kennzeichen: "+a1.getKennzeichen()+" "+a1.getKw()+"
kw");
```

```
    System.out.println(a2.getMarke()+", Kennzeichen: "+a2.getKennzeichen()+" "+a2.getKw()+"
kw");
    System.out.print("\nWelches Fahrzeug wollen Sie buchen? (Kennzeichen eingeben): ");
    String kz=Console.s();
    System.out.print("\nWie lange wollen Sie buchen? (in Tagen): ");
    int d=Console.i();
    k1.buchen(1,kz,d);    //Kunde bucht Fahrzeug und erzeugt Buchungsobjekt
    //Werte anzeigen
    System.out.println("\nKundennummer: "+k1.getKdnr()+" Name: "+k1.getKName());
    System.out.println("Hat gebucht: "+k1.getMeineBuchung().getKennz()+" für
"+k1.getMeineBuchung().getDauer()+" Tage");
  }
}
```

2. Fallbeispiel: Kassenumsätze (BALDI)

```
public class zentrale {

  // Anfang Variablen
  private String zBez;
  private double firmenumsatz;
  // Ende Variablen

  // Anfang Ereignisprozeduren
  public void berFirmenumsatz(double f)
  {

this.firmenumsatz=this.firmenumsatz+f;
  }
  public void setZBez(String z)
  {
    this.zBez=z;
  }
  public String getZBez()
  {
     return zBez;
  }
  public double getFirmenumsatz()
  {
    return this.firmenumsatz;
  }
  // Ende Ereignisprozeduren
}
```

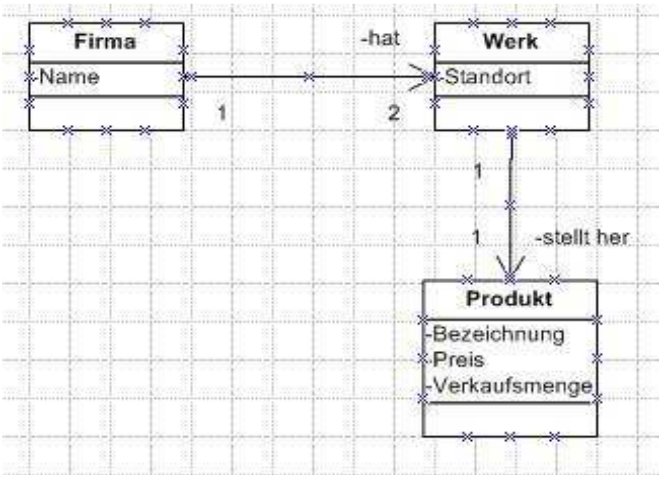
```
public class filiale {

  // Anfang Variablen
  private String fBez;
  private double filialumsatz;
  // Ende Variablen

  // Anfang Ereignisprozeduren

  public void berechneFilialumsatz(double k)
  {
    this.filialumsatz=filialumsatz+k;
  }
  public void setFBez(String f)
  {
    this.fBez=f;
  }
  public String getFBez()
  {
     return fBez;
  }
  public void setFilialumsatz(double f)
  {
    this.filialumsatz=f;
  }
  public double getFilialumsatz()
  {
     return filialumsatz;
  }
  // Ende Ereignisprozeduren
}
```

```
public class kasse {

  // Anfang Variablen
  private double anfangsbestand;
  private double endbestand;
  private String kBez;
  // Ende Variablen

  // Anfang Ereignisprozeduren

  public kasse(double a, double e, String b)
  {
    this.anfangsbestand=a;
    this.endbestand=e;
    this.kBez=b;
  }

  public double getAnfangsbestand() {
    return anfangsbestand;
  }

  public void setAnfangsbestand(double anfangsbestand)
{
    this.anfangsbestand = anfangsbestand;
  }

  public double getEndbestand() {
    return endbestand;
  }

  public void setEndbestand(double endbestand) {
    this.endbestand = endbestand;
  }

  public String getKBez() {
    return kBez;
  }

  public void setKBez(String kBez) {
    this.kBez = kBez;
  }
  public double getTagesumsatz()
  {
     return this.endbestand-this.anfangsbestand;
  }
  // Ende Ereignisprozeduren
}
```

```
public class kassenumsatz
{
  public static void main(String argv[])
  {
    //Objekte erzeugen und initialisieren
    zentrale z = new zentrale();
    z.setZBez("BALDI");
    filiale f1 = new filiale();
    f1.setFBez("BALDI Nord");
    f1.setFilialumsatz(0);
    filiale f2 = new filiale();
    f2.setFBez("BALDI SUED");
    f2.setFilialumsatz(0);
    kasse k1 = new kasse(100,500,"Kasse 1");
    kasse k2 = new kasse(150,600,"Kasse 2");
    kasse k3 = new kasse(80,450,"Kasse 3");
    kasse k4 = new kasse(100,1500,"Kasse 4");
    kasse k5 = new kasse(50,600,"Kasse 5");

    //Filialumsätze ermitteln

    f1.berechneFilialumsatz(k1.getTagesumsatz());
    f1.berechneFilialumsatz(k2.getTagesumsatz());
    System.out.println("Tagesumsatz Filiale
"+f1.getFBez()+": "+f1.getFilialumsatz());
    z.berFirmenumsatz(f1.getFilialumsatz());
    f2.berechneFilialumsatz(k3.getTagesumsatz());
    f2.berechneFilialumsatz(k4.getTagesumsatz());
    f2.berechneFilialumsatz(k5.getTagesumsatz());
    System.out.println("Tagesumsatz Filiale
"+f2.getFBez()+": "+f2.getFilialumsatz());
    z.berFirmenumsatz(f2.getFilialumsatz());
    //Gesamtumsatz ermitteln
    System.out.println("\nGesamtumsatz:
"+z.getFirmenumsatz());
  }
}
```

3. Fallbeispiel: Firmenumsatz (OPAL)

zu 1. Klassendiagramm



zu 2-7. Fertige Klassen (endgültige Fassung)

```
public class Firma {
  // Attribute
  private String Name;
  private double gUmsatz;
//Aufgabenteil 7
  // Ende Attribute

  // Anfang Ereignisprozeduren

  public String getName() {
    return Name;
  }

  public void setName(String Name) {
    this.Name = Name;
  }
  public double getGUmsatz()
  {
    return this.gUmsatz;
  }
  public void
BerechneFirmenumsatz(double wu)
  {
     this.gUmsatz=this.gUmsatz+wu;
  }

  // Ende Ereignisprozeduren
}
```

```
public class Werk {
  // Attribute
  private String Standort;
  private Produkt meinProdukt;
//Aufgabenteil 5
  private double WerksUmsatz;
//Aufgabenteil 6

  // Methoden
  public void setStandort(String
n)
  {
    this.Standort=n;
  }
  public String getStandort()
  {
    return this.Standort;
  }
  public void
setMeinProdukt(Produkt p)
  {
    this.meinProdukt=p;
  }
  public Produkt
getMeinProdukt()
  {
    return meinProdukt;
  }
  public double
bWerksUmsatz()
  {

this.WerksUmsatz=this.getMein
Produkt().getPreis()*this.getMe
inProdukt().getVerkaufsmenge(
);
    return this.WerksUmsatz;
  }
}
```

```
public class Produkt {
  // Attribute

  String Bezeichnung;
  float Preis;
  int Verkaufsmenge;

  // Methoden

  public void setBezeichnung(String n)
  {
    this.Bezeichnung=n;
  }
  public String getBezeichnung()
  {
    return this.Bezeichnung;
  }
  public void setPreis(float p)
  {
    this.Preis=p;
  }
  public float getPPreis()
  {
    return this.Preis;
  }
  public void setVerkaufsmenge(int a)
  {
    this.Verkaufsmenge=a;
  }
  public int getVerkaufsmenge()
  {
    return this.Verkaufsmenge;
  }

}
```

```
public class Umsatz5
{
 public static void main(String[] args)
 {
  //Aufgabenteil 3
  Firma f = new Firma();
  f.setName("Opal");
  Werk w1 = new Werk();
  Werk w2 = new Werk();
  w1.setStandort("Dortmund");
  w2.setStandort("Essen");
  Produkt p1 = new Produkt();
  Produkt p2=new Produkt();
  p1.setBezeichnung("Smaragd");
  p1.setPreis(20000);
  p1.setVerkaufsmenge(150);
  p2.setBezeichnung("Saphir");
  p2.setPreis(16500);
  p2.setVerkaufsmenge(220);
  //Neues Werk in Siegen   Aufgabenteil 4
  Werk w3 = new Werk();
  w3.setStandort("Siegen");
  LKW l1 = new LKW();
  l1.setBezeichnung("Mammut");
  l1.setPreis(80000);
  l1.setVerkaufsmenge(30);
  l1.setZuladung(18000);
  //Aufgabenteil 5
  w1.setMeinProdukt(p1);
  w2.setMeinProdukt(p2);
  w3.setMeinProdukt(l1);
  //Aufgabenteil 6
  System.out.println("Werk: "+w1.getStandort()+" Produkt: "+w1.getMeinProdukt().getBezeichnung()+",
Umsatz: "+w1.bWerksUmsatz());
  System.out.println("Werk: "+w2.getStandort()+" Produkt: "+w2.getMeinProdukt().getBezeichnung()+",
Umsatz: "+w2.bWerksUmsatz());
  System.out.println("Werk: "+w3.getStandort()+" Produkt: "+w3.getMeinProdukt().getBezeichnung()+",
Umsatz: "+w3.bWerksUmsatz());
  //Aufabenteil 7
  f.BerechneFirmenumsatz(w1.bWerksUmsatz());
  f.BerechneFirmenumsatz(w2.bWerksUmsatz());
  f.BerechneFirmenumsatz(w3.bWerksUmsatz());
  System.out.println("Gesamtumsatz der Firma "+f.getName()+" "+f.getGUmsatz());
  }
}
```

```
public class LKW extends Produkt {

 // Anfang Variablen
 private int Zuladung;
 // Ende Variablen

 // Anfang Ereignisprozeduren
 public int getZuladung() {
  return Zuladung;
 }

 public void setZuladung(int Zuladung) {
  this.Zuladung = Zuladung;
 }

 // Ende Ereignisprozeduren
}
```

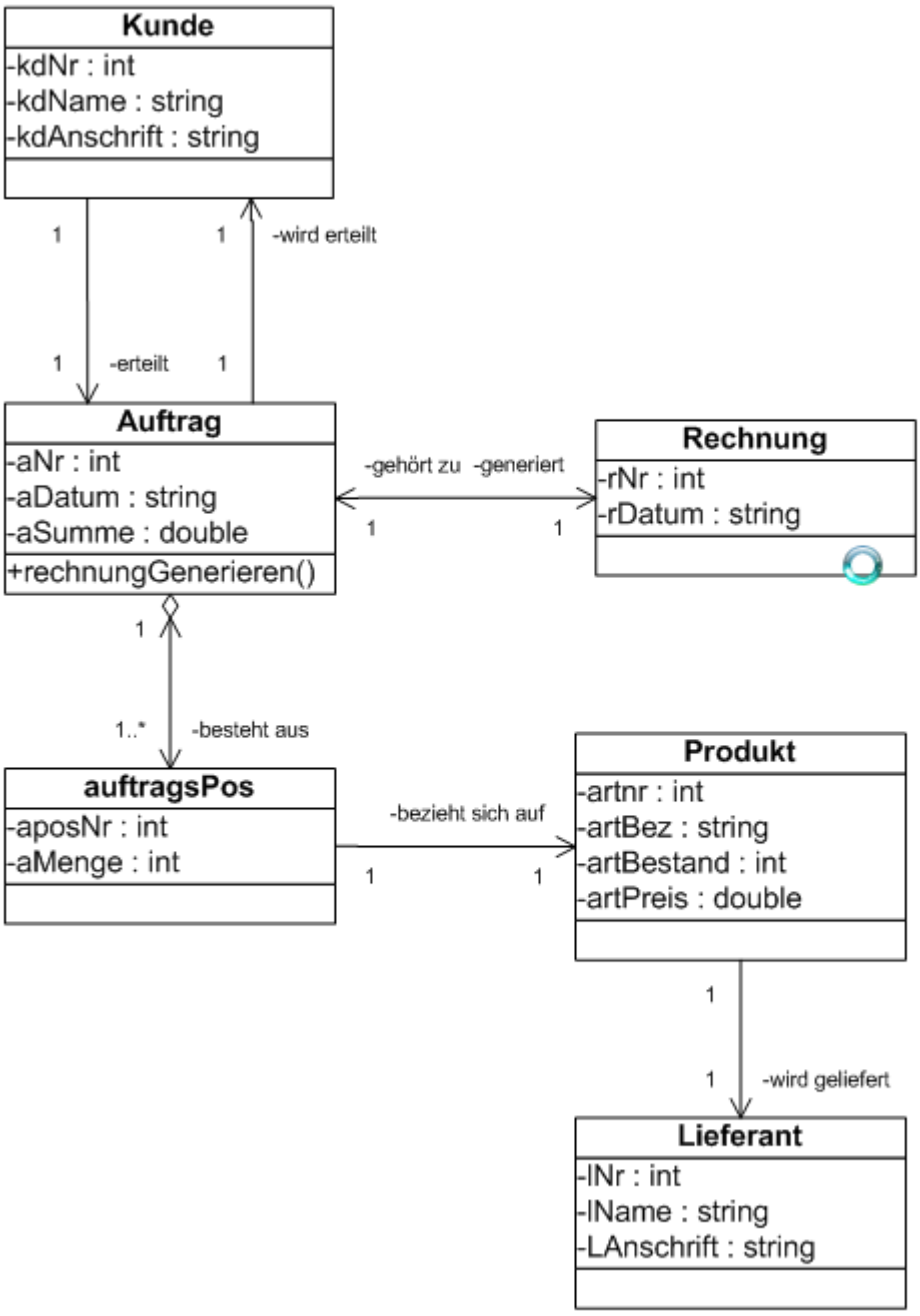
4. Fallbeispiel: Kunde und Betreuer

Lösung für alle drei Anwendungsfälle

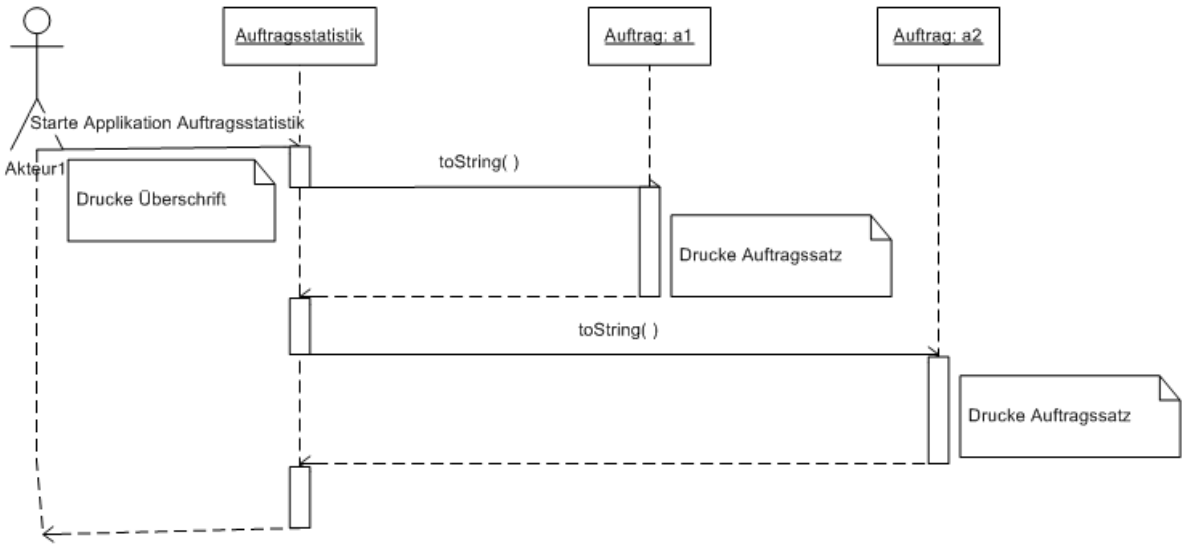
```
import java.io.*;
public class Kundenbetreuung2
{
  public static void main(String[] args) throws IOException
  {
    int wahl=9;
    //Arrays für Klassenobjekte erzeugen
    Kunde[] k = new Kunde[5];
    Betreuer [] b = new Betreuer[5];
    //Objekte erzeugen und intialiseren
    k[0]= new Kunde(101,"Meier");
    b[0]= new Betreuer(1001,"Hans");
    k[0].setKBetreuer(b[0]);
    k[1]= new Kunde(102,"Wagner");
    b[1]= new Betreuer(1002,"Alma");
    k[1].setKBetreuer(b[1]);
    k[2]= new Kunde(103,"Klee & Munk");
    b[2]= new Betreuer(1003,"Fritz");
    k[2].setKBetreuer(b[2]);
    k[3]= new Kunde(104,"Koch GmbH");
    b[3]= new Betreuer(1004,"Ali");
    k[3].setKBetreuer(b[3]);
    k[4]= new Kunde(105,"Weber OHG");
    b[4]= new Betreuer(1005,"Hassan");
    k[4].setKBetreuer(b[4]);

    do
    {
       System.out.println("\nAnwendungsfälle");
       System.out.println("\n1 - Kunde erteilt Auftrag");
       System.out.println("2 - Kunde bezahlt");
       System.out.println("3 - Provision anzeigen");
       System.out.println("0 - Beenden");
       System.out.print("\n\nEingabe ");
       wahl=Console.i();
       switch(wahl)
       {
            case 1:    auftrag(k,b); break;
            case 2:    bezahlen(k,b); break;
            case 3:    provision(b);  break;
            case 0:    break;
            default:   System.out.println("\nUngültige Eingabe");
       }
    }
    while(wahl>0);

  }
  public static void auftrag(Kunde [] k,Betreuer [] b) throws IOException
  {
        int kd;
        int merk=-1;
        double s;
        System.out.print("\nKundennummer: ");
        kd=Console.i();
        for (int i=0;i<5; i++)
        {
```

```
          if (k[i].getKdnr()==kd)
          {
            merk=i;
          }
        }
        if(merk>=0)
        {
           System.out.print("Auftragssumme: ");
           s=Console.d();
           k[merk].setKUmsatz(s);
           k[merk].setKOffen(s);
           System.out.println("\nKunde "+k[merk].getKName()+" hat noch "+k[merk].getKOffen()+" zu
bezahlen");
        }
        else
        {
          System.out.println("Kunde nicht gefunden");
        }
  }
  public static void bezahlen(Kunde [] k,Betreuer [] b) throws IOException
  {
        int kd;
        int merk=-1;
        double s;
        System.out.print("\nKundennummer: ");
        kd=Console.i();
        for (int i=0;i<5; i++)
        {
          if (k[i].getKdnr()==kd)
          {
            merk=i;
          }
        }
        if(merk>=0)
        {
           System.out.print("Bezahlter Betrag: ");
           s=Console.d();
           k[merk].kundeBezahlt(s);
        }
        else
        {
          System.out.println("Kunde nicht gefunden");
        }
    }

  public static void provision(Betreuer [] b)  throws IOException
  {
        int bd;
        int merk=-1;
        double s;
        System.out.print("\nBetreuer: ");
        bd=Console.i();
        for (int i=0;i<5; i++)
        {
          if (b[i].getBnr()==bd)
          {
            merk=i;
          }
        }
        System.out.println("Provision: "+b[merk].getBProvision());
  }
}
```

8. Fallbeispiel: Auftragsstatistik



Fallbeispiel: Auftragsstatistik

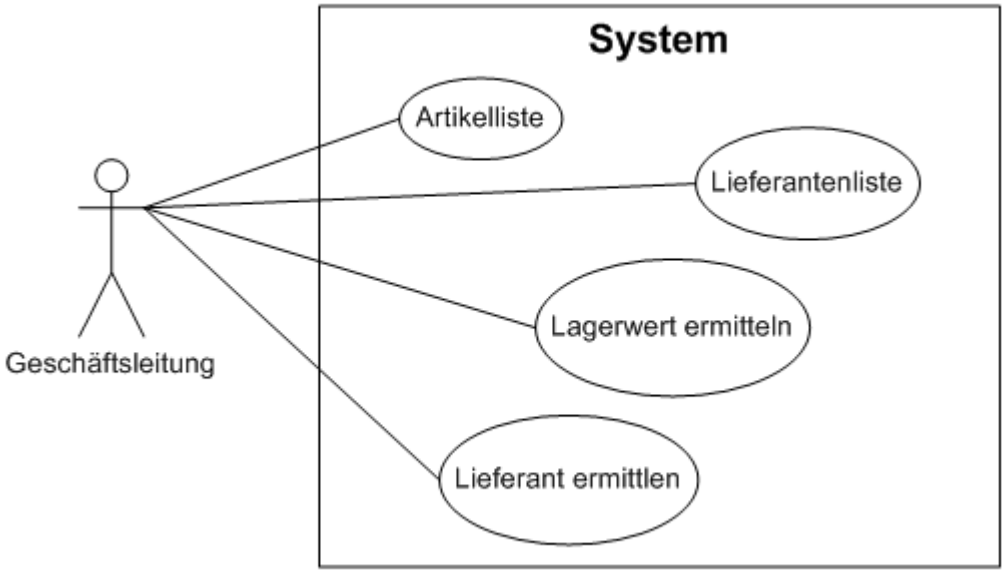


Im Sequenzdiagramm wird davon ausgegangen, dass es eine toString-Methode gibt, die die Werte des Auftrags in der Form, wie in der Aufgabe beschrieben, ausgibt.

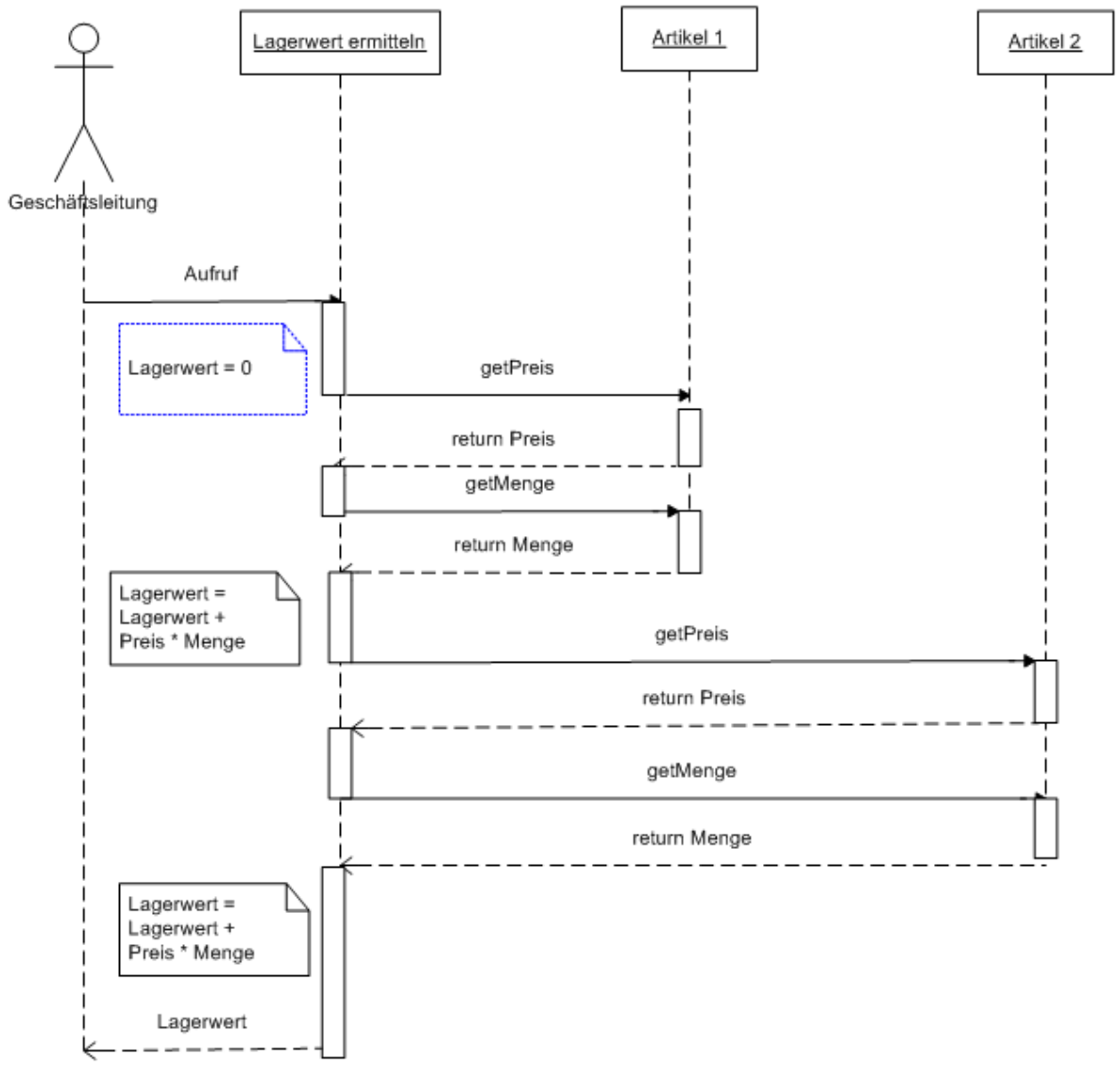
9. Fallbeispiel: Wer liefert was?

1. Klassendiagramm (ohne Attribute und Methoden)

2. Anwendungsfalldiagramm (Use-Case-Diagramm)



3. Sequenzdiagramm (eine von mehreren Lösungsmöglichkeiten)



Quellcode

1. Klassen

```
public class Artikel {

  // Anfang Variablen
  private int Artnr;
  private String Artbez;
  private int ABestand;
  private double AEkPreis;
  private Lieferant meinLieferant;
  // Ende Variablen

  // Anfang Ereignisprozeduren
  public Artikel(int a, String b, int m, double p)
  {
    this.Artnr=a;
    this.Artbez=b;
    this.ABestand=m;
    this.AEkPreis=p;
  }
  public int getArtnr() {
    return Artnr;
  }

  public void setArtnr(int Artnr) {
    this.Artnr = Artnr;
  }

  public String getArtbez() {
    return Artbez;
  }

  public void setArtbez(String Artbez) {
    this.Artbez = Artbez;
  }

  public int getABestand() {
    return ABestand;
  }

  public void setABestand(int ABestand) {
    this.ABestand = ABestand;
  }

  public double getAEkPreis() {
    return AEkPreis;
  }

  public void setAEkPreis(double AEkPreis) {
    this.AEkPreis = AEkPreis;
  }

  public Lieferant getMeinLieferant() {
    return meinLieferant;
  }

  public void setMeinLieferant(Lieferant meinLieferant) {
    this.meinLieferant = meinLieferant;
  }

  // Ende Ereignisprozeduren
}
```

```
public class Lieferant {

  // Anfang Variablen
  private int Lnr;
  private String LName;
  private String LAnschrift;
  // Ende Variablen

  // Anfang Ereignisprozeduren

  public Lieferant(int l, String n, String a)
  {
    this.Lnr=l;
    this.LName=n;
    this.LAnschrift=a;
  }
    public int getLnr() {
    return Lnr;
  }

  public void setLnr(int Lnr) {
    this.Lnr = Lnr;
  }

  public String getLName() {
    return LName;
  }

  public void setLName(String LName) {
    this.LName = LName;
  }

  public String getLAnschrift() {
    return LAnschrift;
  }

  public void setLAnschrift(String LAnschrift) {
    this.LAnschrift = LAnschrift;

  }
  // Ende Ereignisprozeduren
}
```

2. Applikationen

```
import java.io.*;

public class Artikelliste
{
  public static void main(String argv[]) throws IOException
  {
    int artikelnr;
    Artikel [] a = new Artikel[5];
    a[0] = new Artikel(100,"Hose",100,49.5);
    a[1] = new Artikel(101,"Hemd",85,27.5);
    a[2]= new Artikel(102,"Schuh",42,99.50);
    a[3]= new Artikel(103,"Strumpf",50,9.50);
    a[4]= new Artikel(104,"Jacke",30,75.00);
    Lieferant [] l = new Lieferant[3];
    l[0] = new Lieferant(200,"Huber AG","Frankfurt");
    l[1] = new Lieferant(201,"Meier","Gießen");
    l[2] = new Lieferant(202,"Allkauf GmbH","Hanau");
    a[0].setMeinLieferant(l[0]);
    a[1].setMeinLieferant(l[1]);
    a[2].setMeinLieferant(l[2]);
    a[3].setMeinLieferant(l[2]);
    a[4].setMeinLieferant(l[2]);
    System.out.println("Artikelliste\n");
    for(int i=0;i<5;i++)
    {

System.out.println(a[i].getArtnr()+"\t"+a[i].getArtbez()+"\t"+a[i].getABestand()+"\t"+a[i].get
AEkPreis());
    }
  }
```

```
import java.io.*;

public class Lieferantenliste
{
  public static void main(String argv[]) throws IOException
  {
    int artikelnr;
    Artikel [] a = new Artikel[5];
    a[0] = new Artikel(100,"Hose",100,49.5);
    a[1] = new Artikel(101,"Hemd",85,27.5);
    a[2]= new Artikel(102,"Schuh",42,99.50);
    a[3]= new Artikel(103,"Strumpf",50,9.50);
    a[4]= new Artikel(104,"Jacke",30,75.00);
    Lieferant [] l = new Lieferant[3];
    l[0] = new Lieferant(200,"Huber AG","Frankfurt");
    l[1] = new Lieferant(201,"Meier","Gießen");
    l[2] = new Lieferant(202,"Allkauf GmbH","Hanau");
    a[0].setMeinLieferant(l[0]);
    a[1].setMeinLieferant(l[1]);
    a[2].setMeinLieferant(l[2]);
    a[3].setMeinLieferant(l[2]);
    a[4].setMeinLieferant(l[2]);

    System.out.println("Lieferantenliste\n");
    for(int i=0;i<3;i++)
    {
      System.out.println(l[i].getLnr()+"\t"+l[i].getLName()+"\t"+l[i].getLAnschrift());
    }
  }
}


import java.io.*;

public class Lagerwert
{
  public static void main(String argv[]) throws IOException
  {
    int artikelnr;
    Artikel [] a = new Artikel[5];
    a[0] = new Artikel(100,"Hose",100,49.5);
    a[1] = new Artikel(101,"Hemd",85,27.5);
    a[2]= new Artikel(102,"Schuh",42,99.50);
    a[3]= new Artikel(103,"Strumpf",50,9.50);
    a[4]= new Artikel(104,"Jacke",30,75.00);

    double lw=0;
    System.out.println("Lagerwert\n");
    for(int i=0;i<5;i++)
    {
       lw=lw+a[i].getABestand()*a[i].getAEkPreis();
    }
    System.out.println("Gesamtwert aller Artikel zum EK: "+lw);
  }
}
```

```
import java.io.*;

public class werLiefertWas
{
  public static void main(String argv[]) throws IOException
  {
    int artikelnr;
    Artikel [] a = new Artikel[3];
    a[0] = new Artikel(100,"Hose",100,49.5);
    a[1] = new Artikel(101,"Hemd",85,27.5);
    a[2]= new Artikel(102,"Schuh",42,99.50);
    Lieferant l1=new Lieferant(200,"Huber AG","Frankfurt");
    Lieferant l2=new Lieferant(201,"Meier","Gießen");
    Lieferant l3=new Lieferant(202,"Allkauf GmbH","Hanau");
    a[0].setMeinLieferant(l1);
    a[1].setMeinLieferant(l2);
    a[2].setMeinLieferant(l3);
    System.out.print("Artikelnummer eingeben: ");
    artikelnr=Console.i();                          //siehe Anmerkung
    int z,merk=0;
    boolean g=false;
    for(z=0; z<3; z++)
    {
      if(artikelnr==a[z].getArtnr())
      {
         g=true;
         merk=z;
      }
    }
    if(g)
      System.out.println("Artikel "+a[merk].getArtnr()+" "+a[merk].getArtbez()+", Lieferant:
"+a[merk].getMeinLieferant().getLName());
    else
      System.out.println("Artikel nicht gefunden!");
  }
}
```

Anmerkung: Die Tastatureingabe wird über eine die Klasse Console realisert, die sich in diesem Fall im gleichen Verzeichnis befinden muss, wie die Applikation. Hier wird die Methode ***i( )*** der Klasse **Console** aufgerufen, die einen Integerwert einliest.