

*Georgische Technische Universität  
Humboldt Universität zu Berlin*

*Klaus Bothe, Gia Surguladze*

*OBJEKTORIENTIERTE MODELLIERUNG  
UND PROJEKTIERUNG  
(Lehrmaterialien-1)*

*unterstützt durch DAAD  
(Deutschland)*



*Berlin - Tbilissi - 2006*

*bothe@informatik.hu-berlin.de*

**Prof.Dr. Klaus Bothe**



- 1979 Promotion (A) zur Thematik "Spezifikation und Verifikation abstrakter Datentypen";
- 1986 Promotion (B) zur Thematik "Ein algorithmisches Interface fuer Pascal-Compiler: Compiler-Portabilität durch Modularisierung";
- 09.1986-07.1987 Forschungsaufenthalt am ungarischen Forschungszentrum SZKI in Budapest;
- 09.1991 - 02.1992 Sonderforschungsstipendiat der Humboldt- Stiftung in Erlangen bei Prof. Stoyan;
- Seit 12.1993 Professur fuer Softwaretechnik und Theorie der Programmierung an der Humboldt-Universität zu Berlin.
- Bisherige Arbeitsgebiete: Theorie der Programmierung (Algebraische Spezifikation abstrakter Datentypen); Implementation prozeduraler Programmiersprachen (Compilertechnik); Logische Programmierung und Implementation von Prolog Wissensverarbeitung und Expertensysteme.

**Prof.Dr. Gia Surguladze**

*gsurg@gmx.net*

- 1980 Promotion (A), Elektrotechnisches Institut, St-Peterburg (Leningrad);
- 1991 Sonderforschungsstipendiat DAAD in Erlangen bei Prof. Stoyan (Deutschland);
- 1993 Habilitation (B) an der GTU, Tbilissi;
- 1995,96, 99, 2003 Sonderforschungsstipendiat DAAD in Erlangen bei den Professoren: H. Wedekind, F. Hofmann, G. Bolch;
- 2000, 03 Sonderforschungsstipendiat DAAD an der Humboldt Universität zu Berlin, bei Prof. K.Bothe, Prof. W.Reisig;
- 1994 - bis h/Z Professor an dem Lehrstuhl „Management Information Systems“, GTU.
- 2001 – bis h/Z Direktor des Deutsch-Georgischen wissenschaftlichen Gemeinschaftslehrzentrum GTU.
- Forschungsinteressen: UML, OO-Programmierung; Relationale verteilte Datenbanken; Petrinetzen, PNML; .NET Technologie, C#, ADO, ASP.



---

© "Georgian Technical University", 2006

ISBN 99940-56-01-8

## ***Inhaltsverzeichnis***

<b><i>1. Einführung in die objektorientierte Modellierung</i></b>	<b>4</b>
<b><i>2. Klassen und Objekte</i></b>	<b>5</b>
<b><i>3. Analysemethoden zur Ermittlung von Klassen</i></b>	<b>9</b>
<b><i>3.1. Die Textanalyse</i></b>	<b>9</b>
<b><i>3.2. Realitätsbeobachtungen</i></b>	<b>11</b>
<b><i>3.3. Die Generalisierung und Spezialisierung</i></b>	<b>13</b>
<b><i>3.4. Die Ereignisanalyse</i></b>	<b>14</b>
<b><i>4. Vererbungshierarchie</i></b>	<b>17</b>
<b><i>5. Polymorphismus</i></b>	<b>19</b>
<b><i>6. Beziehungsstrukturen</i></b>	<b>21</b>
<b><i>6.1. Beziehungstypen</i></b>	<b>21</b>
<b><i>6.2. Abbildungen</i></b>	<b>23</b>
<b><i>7. Aggregationsstrukturen</i></b>	<b>24</b>
<b><i>8. Verhaltensmodellierung</i></b>	<b>25</b>
<b><i>9. Glossar</i></b>	<b>28</b>
<b><i>10. Fragen und Übungen</i></b>	<b>30</b>

## **1. EINFÜHRUNG IN DIE OBJEKTORIENTIERTE MODELLIERUNG**

*Mit dem objektorientierten Ansatz sind nach Auffassung zahlreicher Fachleute signifikante Verbesserungen bei Entwurf, Realisierung, Test und Wartung von Anwendungen zu erzielen.*

*Bei der datenorientierten Vorgehensweise konzentriert sich das Interesse zunächst auf Objekte der Realität. Die Ermittlung von Funktionen (Tätigkeiten) wird erst dann in Angriff genommen, nachdem besagte Objekte bestimmt sind und im Sinne eines Leitbildes modellmässig zur Verfügung stehen /1,2/.*

*In der Vergangenheit war die Betrachtung eines Datenverarbeitungssystems sehr stark durch das Daten in Funktionen geprägt. Das rührt sicher auch daher, dass das Entwickeln von Software als Schreiben von Programmen gesehen wird und ein Programm realisiert eben eine oder auch mehrere Funktionen /3/.*

*In den 80-er Jahren haben die Daten zunehmend stärkere Beachtung erfahren. Indiz dafür ist die hohe Bedeutung, die man Datenmodellen, relationalen Datenbanken, Data Dictionaries, der Datenadministration u.ä.m. beimisst /5/ .*

*Beide Sichten – die funktions- wie die datenorientierte – haben natürlich ihren Sinn, aber sie vermitteln jeweils nur einen einseitigen und somit unvollständigen Blick auf ein System. Von daher ist eine Methode, deren Wesen in einer organischen Verbindung von Daten und Funktionen liegt, genau das Richtige. Und so ist es bei der objektorientierten Methodik.*

*Die Methodik der objektorientierten Modellierung umfasst die statistische Objektmodellierung (Zustandsmodellierung) und die dynamische Objektmodellierung (Verhaltensmodellierung).*

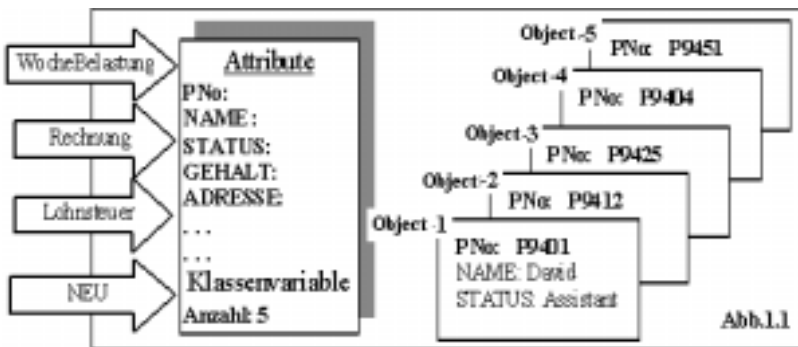
*Objekte repräsentieren individuelle und identifizierbare Exemplare von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt. Ein Objekt entspricht also einem Einzelfall und ist als solcher wenig geeignet, die Realität modellhaft abzubilden.*

Methoden (auch Services) bestimmen ja bekanntlich die Reaktion von Objekten auf einen äusseren Einfluss und sind damit für das Verhalten von Objekten zuständig. Zunächst soll aber von Nachrichten die Rede sein. Mit letzteren sind Methoden zu aktivieren und Objekte zum Agieren aufzufordern.

## **2. KLASSEN und OBJEKTE**



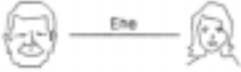


Eine Klasse repräsentiert das Muster für die Kreierung von Objekten gleichen Typs. In einer Klasse sind somit Methoden und Variablen vorzufinden, welche an die zu kreierenden Objekte weiterzugeben sind.

Von Objekten gleichen Tzps spricht man, wenn ihr Status aufgrund identischer Variablen (Attributen) festzuhalten ist (jeder Lektor hat PNo, NAME, STATUS, GEHALT, ADRESSE u.s.w.) und ein identisches Verhalten vorliegt (identische Methoden wie WOCHEBELASTUNG, RECHNUNG, LOHNSTEUER u.s.w.). Abb. 1.1 illustriert das Prinzip einer Klasse.



Zu unterschieden ist zwischen konkreten (instanzierbaren) Klassen und abstrakten (nicht instanzierbaren) Klassen. Eine konkrete Klasse enthält im Normalfall Objekte, während in einer abstrakten Klasse niemals Objekte vorzufinden sind. Der Zweck einer abstrakten Klasse liegt darin, Attribute und Methoden an konkrete Klasse zu vererben.

*Zu beachten ist, dass ein Objekt grundsätzlich irgend ein individuelles und identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt zu repräsentieren vermag (Abb.1.2) /2/.*

Ein Objekt kann sein:	Beispiele
Ein Individuum	 Einwohner    Mitarbeiter    Student    Dozent
Ein reales Objekt	 Maschine    Gebäude    Produkt
Ein abstraktes Konzept	Fachgebiet "Informatik" Vorlesung "Mathematik"
Ein Ereignis	Immatrikulation eines Studenten Rechnungsverbuchung
Eine Beziehung	 Ehe
Ein Software-Konstrukt	 Linked List
Ein Ergebnis	

**Abb.1.2. Beispiele des Objekts**

*Ein Objekt besitzt individuelle Eigenschaften, die sich in Daten (Attributen) sowie Verhalten (Methoden) gliedern lassen. Die Kommunikation zwischen Objekten erfolgt ausschließlich durch das Senden von Nachrichten. Das Verhalten eines Objekts wird durch seine Methoden bestimmt. Erhält ein Objekt eine Nachricht, so wählt es eine geeignete Methode aus, welche ein Ergebnis ermittelt und dieses an Sender-Objekt zurückschickt.*

*Ein Objekt kann mehrere Zustände einnehmen und in jedem Zustand anders reagieren. Der Zustand eines Objekts wird*

*mittels Daten (Attributen) festgehalten. Ein Objekt verkapselt sowohl seine Daten (Attribute) wie auch seine Methoden und verkehrt mit der Aussenwelt über eine wohldefinierte Schnittstelle.*

*Von besonderer Bedeutung ist die auch unter dem Begriff Geheimnisprinzip (information hiding) bekannte Verkapselung von Daten und Methoden. Damit sind nämlich Implementationsdetails zu verstecken und damit Benutzer von Implementationsdetails zu entlasten.*

*In der Literatur gibt es zahlreiche Vorschläge für die Notation von Klassen und Objekte (Abb.1.3) /4/.*

Element Autor	Class, Object	Assotiation	Subsystem	Inheritance	Message
G. Booch		N M 			
P. Coad E. Yordon		NM NM 			
D. Martin					
S. Shleer S. Melor					
I. Jacobson		[N,M] 			

**Abb.1.3 Notation verschiedener OO-Methoden**

Abb.1.4 zeigt z.B., die Notation Gemäss Coad-Zordon (a) und gemäss Booch (b) für konkrete bzw. abstrakte Klassen.

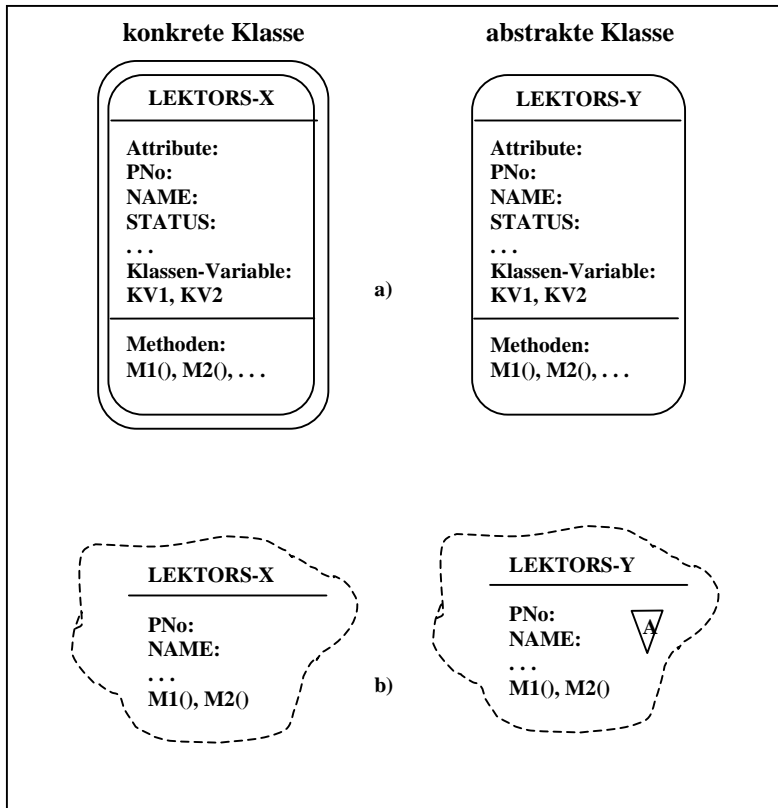


Abb.1.4 Notation der Klasse

### **3. ANALYSMETHODEN ZUR ERMITTLUNG VON KLASSEN**

*In der klassischen Datenmodellierung betrachtet man folgende Techniken zur Ermittlung von Klassen: die Textanalyse, Realitätsbeobachtungen, die Benützersichtanalyse, die Generalisierung und Spezialisierung, die Ereignisanalyse.*

#### **3.1. Die Textanalyse**

*Für die textuelle Beschreibung eines Problems verwendet man die grammatische Kategorien der Sprache, und zwar die Substantive als Kandidaten für Klassen. In analoger Weise sind Variable anhand von Adjektiven und Methoden anhand von Verben ausfindig zu machen. Dieses Verfahren ist nicht perfekt, führt aber sehr schnell zu ersten Hinweisen. Selbstverständlich steht und fällt das Verfahren mit der Qualität des verfügbaren Textes.*

*Beispiel: Eine Unternehmung möchte die Administration ihrer internen Kurse automatisieren. Hinsichtlich der Kursorganisation liegen die im nachfolgenden Rahmen festgehaltenen Hinweise vor. Eine Textanalyse liefert folgende Kandidaten für Klassen (im Text-1 durch unterstrichene Substantive markiert):*

*MITARBEITER  
DOZENT  
STUDENT  
KURSTYP  
KURS  
LOKAL  
SCHLAFRAUM*

*Zudem sind folgende Methoden in Betracht zu ziehen (im Text kursiv markiert):*

*Kursanmeldungen pro Student durchführen  
Kursabschlüsse pro Student durchführen  
Teilnehmerliste pro Kurs erstellen*

*Atteste pro Kurs erstellen  
Kursliste pro Dozent erstellen  
Belegungsplan pro Lokal bzw. Schlafraum ermitteln*

*Offensichtlich sind mittels einer Textanalyse neben Klassen, Variablen sowie Methoden auch Beziehungen zwischen Klassen ausfindig zu machen. Beispielsweise deutet die Aussage „Jeder Kurs erfordert einen Dozenten“ darauf hin, dass die Klassen KURS und DOZENT miteinander in Beziehung stehen.*

*Zu beachten ist, dass Variable nicht nur anhand von Adjektiven, sondern auch von Methoden her abzuleiten sind. So ist beispielsweise die Methode Belegungsplan pro Lokal ermitteln nur dann zu realisieren, wenn bekannt ist, welche Kurse von wann bis wann in welchen Lokalen stattfinden. Hierfür sind aber die Variablen AB.DATUM und BIS.DATUM für die Klasse KURS erforderlich.*

*Text-1: „KURSOrganization“*

- 1. Es werden **Kurse** unterschiedlichen **Typs** (beispielsweise Informatik, Betriebswirtschaftslehre, Branchenkunde, etc.) angeboten. Für jeden **Kurstyp** gibt es mehrere **Kurse** (beispielsweise Informatik im Frühjahr, Sommer und Herbst).*
- 2. Jeder **Kurs** erfordert einen **Dozenten**. Ein **Dozent** ist in der Regel für mehrere **Kurse** zuständig.*
- 3. Ein **Kurs** kann von mehreren **Studenten** besucht werden. Umgekehrt kann ein **Student** mehrere **Kurse** besuchen.*
- 4. **Dozenten** und **Studenten** sind **Mitarbeiter** ein und derselben **Firma**.*
- 5. Jeder **Kurs** findet in einem **Lokal** statt. Ein **Lokal** kann im Verlaufe der Zeit von verschiedenen **Kursen** belegt sein.*
- 6. Jeder **Dozent** und jeder **Student** braucht normalerweise einen **Schlafraum**. Ein **Schlafraum** kann im Verlaufe der Zeit von verschiedenen **Mitarbeitern** belegt werden.*
- 7. Pro **Student** sind **Kursanmeldungen** sowie **Kursabschlüsse durchzuführen**.*
- 8. Pro **Kurs** ist eine **Teilnehmerliste** und sind **Kursatteste zu erstellen**.*
- 9. Pro **Dozent** ist eine **Kursliste zu erstellen**.*
- 10. Pro **Lokal** ist ein **Belegungsplan zu erstellen**.*
- 11. Pro **Schlafraum** ist ein **Belegungsplan zu erstellen**.*

### 3.2. Realitätsbeobachtungen

Bei diesem Verfahren werden die für einen Problembereich relevanten Objekttypen und damit Klassen aufgrund von Realitätsbeobachtungen ermittelt. Auch dem vorstehend analysierten Text liegen Realitätsbeobachtungen zugrunde. Start diese Beobachtungen nun aber mühsam in textueller Form festzuhalten, dokumentiert man sie gleich in objektgerechter Form. Wie man sich dies für das Beispiel Kursorganisation vorzustellen hat, ist Abb. 1.5 (OO-Diagramm nach Coad/Yourdon) zu entnehmen.

Empfehlenswert ist, bei der Definition einer Klasse gleich auch die zur Identifikation tatsächlicher Objekte erforderliche Variable - die identifizierende Variable also - festzulegen. Für die Klasse MITARBEITER kommt hierfür beispielsweise die Variable M# in Frage.

Realitätsbeobachtungen eignen sich hervorragend, um möglichst rasch ein, wenn auch grobes, OO-Modell zu ermitteln.

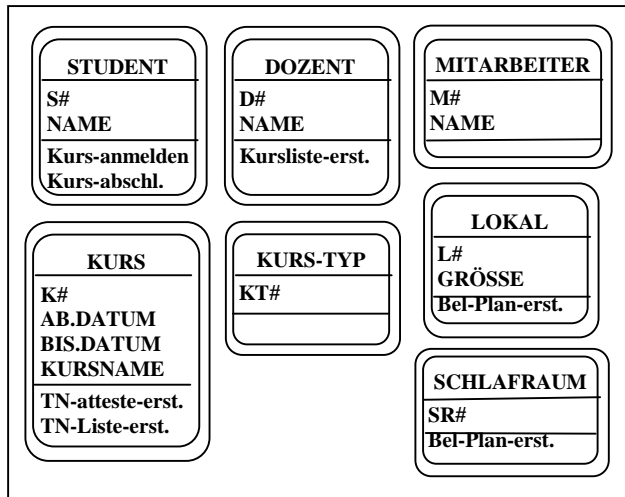


Abb. 1.5. OO-Diagramm nach Coad/Yourdon

## *Die Benützersichtanalyse*

*Bei der Bearbeitung von Wirtschafts- und Verwaltungsproblemen geht man zweckmassigerweise ergebnisorientiert vor. Zu diesem Zwecke ermittelt man kooperativ - also mit Beteiligung von Führungskräften, Sachbearbeitern und Informatikern - die für eine Anwendung zu produzierenden Ergebnisse in Form von Benützersichten für Bildschirmausgaben, Listen, Formulare, Belege etc.). Heute setzt man hierfür geeignete Tools ein, mit denen die angesprochenen Ergebnisse prototypmässig zu erzeugen sind (das heisst: ohne dass die gezeigten Daten in gespeicherter Form vorliegen oder die sie verarbeitenden Programme existieren müssen). Wir illustrieren die Benützersichtanalyse im folgenden an einem Beispiel und unterstellen zu diesem Zweck, dass für den Problembereich Kursorganisation die in Abb. 1.6 gezeigte Kursbeschreibung zu erstellen ist.*

STUDENTEN					LEITER		BELEGTE KURSE		
St N	Stud. Name	Geb.- Datum	Sprache	Kentus	Dozent-Numm.	Doz-Name	Kurs-Num.	Kurs-Name	Evalu-ation
15	Dvali	05.25.85	Engl.	gut	D05	Kio	k1	Inf.	gut
			Deut.	no			k2	Math.	gut
			Fran.	schl			k3	Deut.	gut
							k4	Alg.	gut
22	Guhua	12.03.80	Engl.	schl	D05	Kio	k1	Inf.	mittel
			Deut.	gut			k3	Engl.	gut
			Fran.	no			k5	Log.	schlecht

**Abb. 1.6. Beispiel für die Benützersichtanalyse**

## **KURSBESCHREIBUNG**

KURSNUMMER: K1

DOZENTENNUMMER: D5

KURSNAME: Informatik

DOZENTENNAME: Shonia

LOKALNUMMER: L50

LOKALGROESSE: 55

EINGESCHRIEBENE STUDENTEN:

*Pro Kurs ist also das Lokal, der Dozent, die Studenten, deren Studienleiter sowie die pro Student besuchten Kurse auszuweisen. Damit sind aber auch schon die wesentlichen Objekttypen und damit Klassen genannt, die von der in Abb.1.6 gezeigten Benützersicht abzuleiten sind.*

*Mit der Benützersichtanalyse sind auch Beziehungen zwischen Klassen, sowie Variable (Attribute) zu ermitteln. So ist beispielsweise zu erkennen, daß für die Klasse KURS eine Variable KURSNAME, für LOKAL eine Variable GRÖSSE, für STUDENT die Variablen NAME sowie GEBURTSDATUM und für DOZENT die Variable NAME erforderlich sind.*

*Eine Benützersichtanalyse eignet sich hervorragend, wenn ein grobes, beispielsweise mittels Realitätsbeobachtungen ermitteltes OO-Modell zu verfeinern ist.*

### **3.3. Die Generalisierung und Spezialisierung**

*Bei dieser Technik überlegt man sich:*

*1. Ist für eine (oder mehrere) Klasse(n) eine generellere, also allgemeinere Klasse denkbar ?*

*Beispiel: Hätte man für den Problembereich Kursorganisation entsprechend Abb. 1.7 zunächst nur die Klassen STUDENT und DOZENT festgelegt, dann hätte eine Generalisierung die Klasse MITARBEITER ergeben.*

*2. Sind für eine Klasse Spezialisierungen denkbar, oder mit andern Worten: Ist eine Klasse nach innen zu gliedern ?*

*Beispiel: Hätte man für den Problembereich Kursorganisation entsprechend Abb. 1.7 zunächst nur die Klasse MITARBEITER festgelegt, dann hätte eine Spezialisierung die Klassen STUDENT und DOZENT ergeben.*

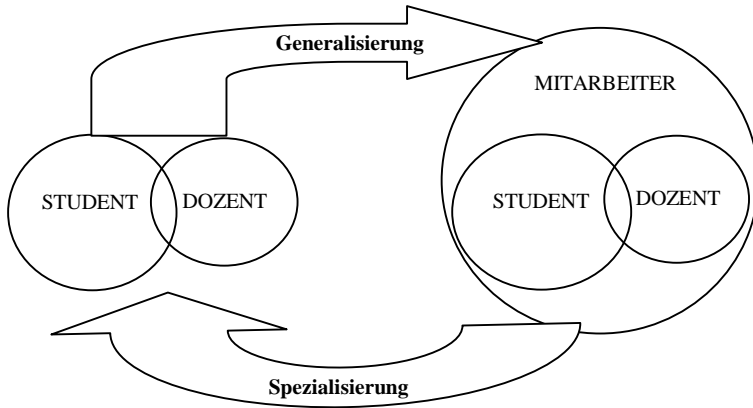


Abb.1.7. Generalisierung und Spezialisierung für Klassen

### 3.4. Die Ereignisanalyse

*Sind über bereits erfolgte oder zu erwartende Ereignisse Informationen festzuhalten, so sind hierfür Ereignisklassen vorzusehen. Mit Ereignisklassen ist also das Geschehen auf der Zeitachse zu modellieren.*

*Beispiel: Im Problembereich Kursorganisation erfordert die Reservierung von Schlafräumen eine die Variablen AB.DATUM sowie BIS.DATUM aufweisende Ereignisklasse RESERVATION, denn nur so ist die zukünftige Belegung von Schlafräumen aufzuzeichnen. Analog erfordert die Belegung von Kursen durch Studenten eine Ereignisklasse BELEGUNG, denn nur so sind die von den Studenten im Verlaufe der Zeit erbrachten Kursleistungen festzuhalten. Letztere sind in Abb. 1.6 mit EVALUATION bezeichnet und müssten im OO-Modell als Variable der Klasse BELEGUNG berücksichtigt werden.*

*Eine Ereignisklasse ist auch erforderlich, wenn die zeitliche Abfolge von Kursen zu modellieren ist (einem Kurs für Anfänger folgen normalerweise mehrere Kurse für Fortgeschrittene, oder*

*umgekehrt: einem Kurs für Fortgeschrittene gehen normalerweise mehrere einfachere Kurse voraus). In unserem OO-Modell soll die zeitliche Abfolge von Kursen mit der Ereignisklasse ABFOLGE zum Ausdruck kommen.*

*Wie später darzulegen sein wird, steht die Ereignisklasse RESERVATION mit den Klassen SCHLAFRAUM und MITARBEITER in Beziehung. Analog steht die Ereignisklasse BELEGUNG mit den Klassen STUDENT und KURS in Beziehung, während die Ereignisklasse ABFOLGE zweckmässigerweise mit der Klasse KURS-TYP in Beziehung zu setzen ist. In der klassischen Datenmodellierung würde man bezüglich der genannten Ereignisklassen von Beziehungsmengen sprechen.*

*Abb. 1.8 von Coad/Yourdon zeigt das mit den Ereignisklassen RESERVATION, BELEGUNG und ABFOLGE ergänzte OO-Diagramm für das Beispiel Kursorganisation.*

*Neben den klassischen, der Datenmodellierung entlehnten Techniken zur Bestimmung von Klassen, gelangen im objektorientierten Umfeld zunehmend auch die im nachfolgenden Rahmen aufgeführten Techniken zum Einsatz, z.B. Verhaltensanalyse (Behavior Analysis), CRC-Karten (Class-Responsibility-Collaboration) u.s.w.*

### Anmerkungen:

#### **Objektklassen (und -typen):**

- Gleichartige Objekte werden in einer Klasse zusammengefaßt.
- Jedes Objekt ist Instanz genau einer Klasse.
- Ein Objekt ändert seine Klasse während seiner Lebenszeit nicht.
- Die Klasse legt für ihre Instanzen fest:
  - ❖ alle Attribute mit zulässigen Wertebereichen.
  - ❖ alle Operationen mit ihren Implementierungen.
  - ❖ die Trennung von Schnittstelle und Rumpf.
- Objekttyp = Schnittstelle einer Klasse ganz ohne Implementierung.
- Eine Klasse kann mehrere Schnittstellen (Typen) implementieren.
- Abstrakte Klasse = Klasse ohne Instanzen mit unvollständiger Implementierung

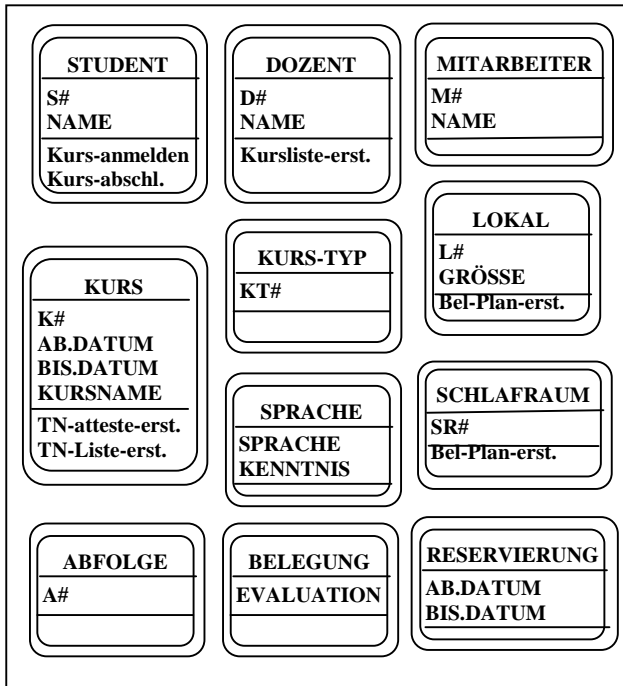


Abb. 1.8. Mit Ereignisklassen ergänztes OO-Diagramm nach Coad/Yourdon

### Anmerkungen:

#### **Objekte:**

- Konkretes Objekt der modellierten Welt, abstraktes Abbild im Programm.
- Kleinste (gekapselte) Einheit in einem objektorientierten System.
- Eindeutig identifizierbar.
- Hat einen internen Zustand:
  - \* repräsentiert durch **Attribute** (Instanzvariablen) = Datenelement.
  - Hat ein dynamisches Verhalten:
    - \* repräsentiert durch **Operationen** (Methoden) = Prozeduren.
- Objekte besitzen eine unveränderliche Identität.
- Objekte besitzen einen veränderlichen Zustand.
- Objekte sind Instanzen von Klassen, die ihr Verhalten festlegen.
- Objekte kommunizieren über Nachrichtenaustausch.
- Klassen bilden eine Vererbungshierarchie und
- auf Programmiersprachen ebene: Polymorphie (dynamisches Binden).

## 4. VERERBUNGSHIERARCHIE

Vererbungsstrukturen resultieren, wenn man die Objekte einer Klasse nach bestimmten Kriterien gruppiert. Der Vorgang läuft darauf hinaus, daß man die Objekte einer sogenannten Superklasse als Menge auffast und diese unter Bildung von Untermengen - sogenannten Subklassen - nach innen gliedert.

Abb.1.7 illustriert ein Beispiel für Vererbungs- bzw. Generalisierungs-Spezialisierungsstruktur. Zu erkennen ist eine nach innen gegliederte Menge (Superklasse) MITARBEITER. Als Untermengen (Subklassen) treten zunächst LEKTOR und STUDENT in Erscheinung. Als Superklasse ist möglich eine Menge von Personen (PERSON) aufzufassen, die alle Mitarbeiters der Institution beinhaltet Abb.1.9.

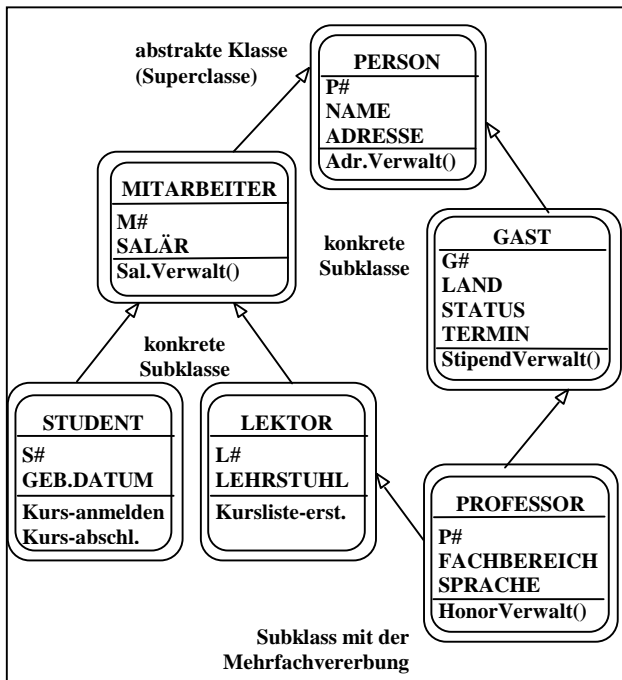


Abb. 1.9. Vererbungsstrukturen

Mit einer abstrakten Klasse sind Variable (Attribute) sowie Methoden an konkrete Klassen zu vererben. Also, mit Vererbung bezeichnet man die Funktionalität, mit welcher Variablen und Methoden einer Superklase an Subklasse weiterzugeben sind.

Falls eine vererbte Variable oder Methode in einer Spezialisierung nicht passt, so kann sie überschrieben werden.

Einfache Vererbung liegt vor, wenn jede Spezialisierung nur eine Generalisierung aufweist.

Mehrfache Vererbung liegt vor, wenn eine Spezialisierung mehrere Generalisierungen aufweist.

Vererbungsstrukturen nutzt und damit insofern einen hohen Grad an Wiederverwendbarkeit bietet, als in Superklassen definierte Variable (Attribute) und Methoden in Subklassen anzusprechen sind. Ein objektorientiertes Vorgehen nutzt das Prinzip der Daten- und Funktionskapselung sowie jenes von Klassen (Abstraktion), Abb.1.10.

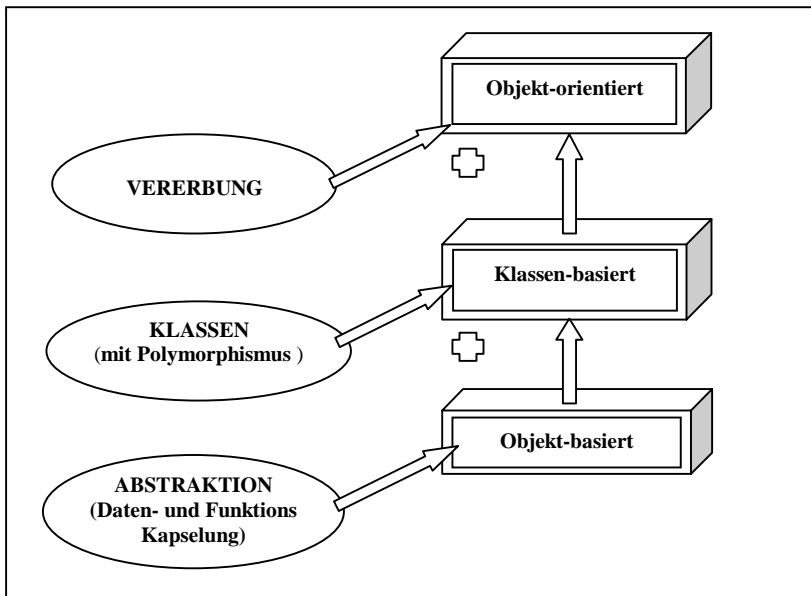


Abb. 1.10. Hauptprinzip der OO-Ansatzes

## 5. POLYMORPHISMUS

Polymorphismus gewährleistet, daß unterschiedlichen Klassen angehörende Objekte aufgrund ein und derselben Nachricht verschieden reagieren können.

Anzumerken ist, daß Polymorphismus natürlich auch bei Klassen funktioniert, die nicht an Vererbungsstrukturen beteiligt sind. Mit Polymorphismus ist nicht nur eine erstaunliche Flexibilität, sondern auch eine signifikante Vereinfachung der Logik von Programmen und damit eine Reduktion des Wartungsaufwandes zu erzielen (Abb.1.11). Kommt dazu, daß für jeden neuen Objekttyp Anpassungen in der Programmlogik

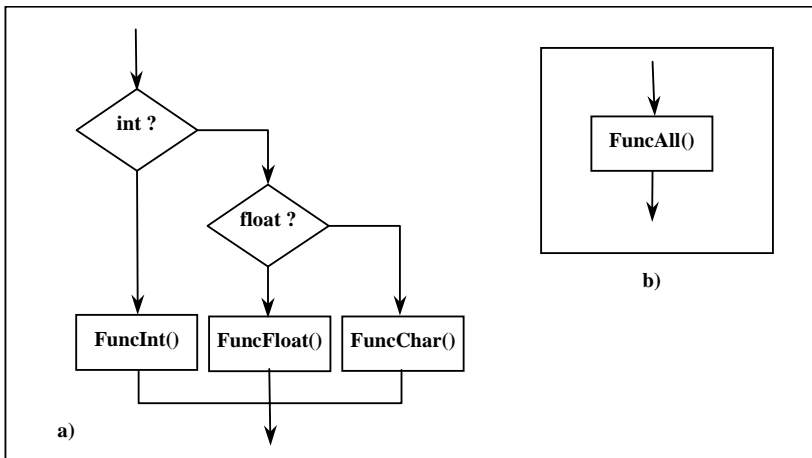
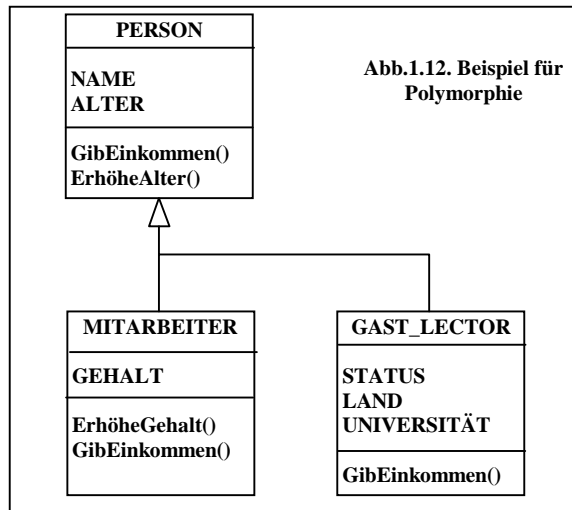


Abb.1.11. Das Programmfragment ohne Polymorphismus (a) und mit Polymorphismus (b)

erforderlich wären. All dies entfällt mit Polymorphismus, kennt doch jedes zu einer Aktion aufgeforderte Objekt die auszuführende Methode.

Polymorphie kommt aus dem Griechischen und heißt „Vielgestaltung“. Als polymorph werden alle Programmiersprachen bezeichnet, die die Definition von Prozeduren (Operationen) erlauben, die auf einer bestimmten Eingabeparameterposition Aktualparameter unterschiedlicher

Typen (Klassen) erlauben. Beispiel für Polymorphie wird in Abb.1.12 illustriert.



Die Operation *GibEinkommen* ist polymorph, da sie auf Objekten der Klassen *PERSON*, *MITARBEITER* und *GAST\_LEKTOR* aufgerufen werden kann.

In polymorphen funktionalen Sprachen wird derselbe Programmcode für Eingabe-parameter unterschiedlicher Typen abgearbeitet (kein dynamisches Binden). Bei objektorientierten Sprachen kann ein bestimmter Operationsaufruf in jeder erfindenden Unterklasse durch eine völlig andere Methode implementiert sein. Die klassenabhängige Auswahl einer bestimmten Implementierungsmethode zu einem Operationsaufruf (zu einer empfangenen Nachricht) zur Programmlaufzeit nennt man dynamisches Binden. Im Gegensatz dazu wird die Auswahl einer bestimmten Prozedur aus einer Menge gleichbenannter Prozeduren zur Übersetzungszeit *Overloading* genannt (anhand der Aktualparameter Typen). Beispiel: der Operator „+“ kann auf *integer*, *float* u.s.w. realisiert sein. Der Übersetzer entscheidet anhand der Aktualparameter welchen Maschinencode er für das „+“ erzeugen muß.

## 6. BEZIEHUNGSSTRUKTUREN

### 6.1. Beziehungstypen

Mit Beziehungsstrukturen sind die Objekte einer Klasse oder verschiedener Klassen miteinander in Beziehung zu setzen. Je nach Anzahl der Objekte, mit denen ein bestimmtes Objekt in Beziehung stehen kann, unterscheidet man in Analogie zur Datenmodellierung folgende vier Fälle:

- Einfache (man sagt auch: Typ 1) Beziehungen.
- Konditionelle (Typ 0,1 oder auch Typ C genannte) Beziehungen.
- Komplexe (Typ 1,M oder auch Typ M genannte) Beziehungen.
- Komplex-konditionelle (Typ 0,M oder auch Typ MC genannte) Beziehungen.

Anstelle von Beziehungen ist mitunter auch von Instance Connections (Coad/Yourdon) oder Associations (Booch) die Rede, während man bezüglich der Beziehungsarten (also einfach, konditionell, komplex, komplex-konditionell) auch von Zuordnungskardinalitäten spricht.

#### **Einfache (Typ 1) Beziehung:**

Wenn jedes Objekt einer Klasse A jederzeit mit einem Objekt einer Klasse B in Beziehung steht, so liegt von A nach B eine einfache (Typ 1) Beziehung vor (im Spezialfall steht jedes Objekt der Klasse A mit einem Objekt der gleichen Klasse in Beziehung). Es ist üblich, eine einfache Beziehung von einer Klasse A zu einer Klasse B wie folgt darzustellen (Abb. 1.13):  $A \rightarrow B$ , z.B.:  $STUDENT\_ID \rightarrow GROUP\_ID$ .

#### **Konditionelle (Typ 0,1 oder C) Beziehung:**

Wenn jedes Objekt einer Klasse A jederzeit mit einem oder keinem Objekt einer Klasse B in Beziehung steht, so liegt von A nach B eine konditionelle (Typ 0,1 oder C) Beziehung vor (im Spezialfall steht jedes Objekt der Klasse A mit einem oder keinem Objekt der gleichen Klasse in Beziehung). Es ist üblich, eine konditionelle Beziehung von einer Klasse A zu einer Klasse

*B* wie folgt darzustellen (Abb. 1.13):  $A \rightarrow B$ .

Z.B.: LEKTOR\_NAME  $\rightarrow$  EHEPAAR\_NAME.

**Komplexe (Typ M oder 1,M) Beziehung:**

Wenn jedes Objekt einer Klasse *A* jederzeit mit einem oder mehreren Objekten einer Klasse *B* in Beziehung steht, so liegt von *A* nach *B* eine komplexe (Typ M oder 1,M) Beziehung vor (im Spezialfall steht jedes Objekt der Klasse *A* mit einem oder mehreren Objekten der gleichen Klasse in Beziehung). Es ist üblich, eine komplexe Beziehung von einer Klasse *A* zu einer Klasse *B* wie folgt darzustellen (Abb. 1.13):  $A \rightarrow\!\!\rightarrow B$ .

Z.B.: STUD\_Id  $\rightarrow\!\!\rightarrow$  LEKTOR\_Id.

**Komplex-konditionelle (Typ 0,M oder MC) Beziehung:**

Wenn jedes Objekt einer Klasse *A* jederzeit mit beliebig vielen Objekten einer Klasse *B* in Beziehung steht, so liegt von *A* nach *B* eine komplex-konditionelle (Typ 0,M oder MC) Beziehung vor (im Spezialfall steht jedes Objekt der Klasse *A* mit beliebig vielen Objekten der gleichen Klasse in Beziehung). Es ist üblich, eine komplex-konditionelle Beziehung von einer Klasse *A* zu einer Klasse *B* wie folgt darzustellen (Abb. 1.13):

$A \rightarrow\!\!\rightarrow B$ , z.B.: STUD\_Id  $\rightarrow\!\!\rightarrow$  FACH\_ID.

1 genau ein	0, 1 kein oder ein	M ein oder mehrere	0, M kein, ein oder mehrere
<u>1</u>	<u>C</u>	<u>M</u>	<u>MC</u>
$\rightarrow$	$\rightarrow)$	$\rightarrow\!\!\rightarrow$	$\rightarrow\!\!\rightarrow)$
$\rightarrow$	$\rightarrow)$	$\rightarrow\!\!\rightarrow$	$\rightarrow\!\!\rightarrow)$
$\rightarrow$	$\rightarrow\bigcirc$	$\rightarrow\!\!\rightarrow$	$\rightarrow\!\!\rightarrow\bigcirc$
u. s. w.			

**Abb. 1.13. Gebräuchlichste Notationen für Beziehungsarten**

## 6.2. Abbildungen

*Mit einer Abbildung sind Beziehungsarten einer Beziehung in beiden Richtungen auszuweisen.*

*Eine Abbildung macht eine Aussage über die Beziehungsarten, die einer Beziehung zwischen den Objekten einer Klasse A und jenen einer Klasse B zugrunde liegen (im Spezialfall betrifft die Abbildung Beziehungen von Objekten ein und derselben Klasse).*

*Ist die Beziehungsart von den Objekten einer Klasse A zu jenen einer Klasse B vom Typ T (also einfach, konditionell, komplex oder komplex-konditionell) und ist die dazu inverse Beziehungsart vom Typ T', so ist eine Abbildung formal wie folgt festzuhalten: ( $T' : T$ ).*

*Dabei sind T und T' durch I (für einfache Beziehungen) bzw. C oder 0,1 (für konditionelle Beziehungen) bzw. M oder 1,M (für komplexe Beziehungen) bzw. MC oder 0,M (für komplex-konditionelle Beziehungen) zu ersetzen.*

*Gebräuchlich ist auch eine Notation, die neben den Beziehungsarten T und T' die Klassen A und B der an der Beziehung beteiligte Objekte wie folgt zum Ausdruck bringt:*

$$A \ T' - T \ B.$$

*Dabei sind T und T' mit geeigneten Symbolen aus Abb. 1.14 (verschiedene Abbildungstypen) zu ersetzen.*

*Nachdem die Beziehungsart der Objekte einer Klasse A zu jenen einer Klasse B einfach, konditionell, komplex sowie komplex-konditionell sein kann, und nachdem der gleiche Sachverhalt auch für die inverse Beziehungsart zutrifft, unterscheidet man  $4 \times 4 = 16$  verschiedene Abbildungstypen.*

*Zu beachten ist, dass beispielsweise eine (1:M)-Abbildung durch eine Vertauschung der Klassen A und B in eine (M:1)-Abbildung umzufunktionieren ist. Entsprechend sagt man, dass die Abbildungen (1:M) und (M:1) zueinander symmetrisch seien. Von den 16 Abbildungstypen in Abb. 1.14 sind insgesamt 6 zueinander symmetrisch, nämlich:*

T' \ T	1	0,1	M	0,M
1	(1 : 1) ↔	(1 : 0,1) ↔	(1 : M) ↔	(1 : 0,M) ↔
0,1	(0,1 : 1) ↔	(0,1 : 0,1) ↔	(0,1 : M) ↔	(0,1 : 0,M) ↔
M	(M : 1) ↔	(M : 0,1) ↔	(M : M) ↔	(M : 0,M) ↔
0,M	(0,M : 1) ↔	(0,M : 0,1) ↔	(0,M : M) ↔	(0,M : 0,M) ↔

Abb.1.14. Notation für Abbildungstypen

## 7. AGGREGATIONSSTRUKTUREN

Mit einer Aggregationsstruktur ist Zusammensetzung eines Ganzen aus mehreren Teilen zum Ausdruck zu bringen.

(1:M)	und	(M:1)
(1:C)	und	(C:1)
(1:MC)	und	(MC:1)
(C:M)	und	(M:C)
(C:MC)	und	(MC:C)
(M:MC)	und	(MC:M)

Zu unterscheiden sind also grundsätzlich 10 verschiedene Abbildungstypen.

G. Booch spricht bezüglich Aggregationsstrukturen von *Containing Relationship* und verwendet dafür die in Abb.1.15 gezeigte Notation /4/.

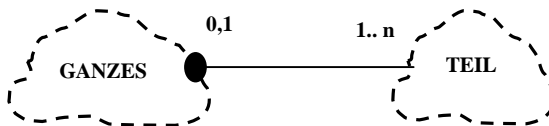


Abb.1.15. Aggregationsstrukturen

## 8. VERHALTENSMODELLIERUNG

Unter der Verhaltensmodellierung versteht man eine dynamische Objektmodellierung. Methoden (oder Services) bestimmen die Reaktion von Objekten auf einen äusseren Einfluss und sind damit für das Verhalten von Objekten zuständig. Die Methoden werden aufgrund von Nachrichten aktiviert.

Als Nachrichtenquellen kommen Objekte, Klassen sowie externe Ereignisse in Frage. Der Austausch von Nachrichten wird in einem OO-Diagramm mittels Nachrichten-Verbindungen (Message Connections) dargestellt. Abb.1.16. illustriert anhand der Booch Notation daß Nachrichten grundsätzlich wie folgt zu versenden sind. Ein Senderobjekt (Client) schickt die Nachricht an das Empfängerobjekt (Server).

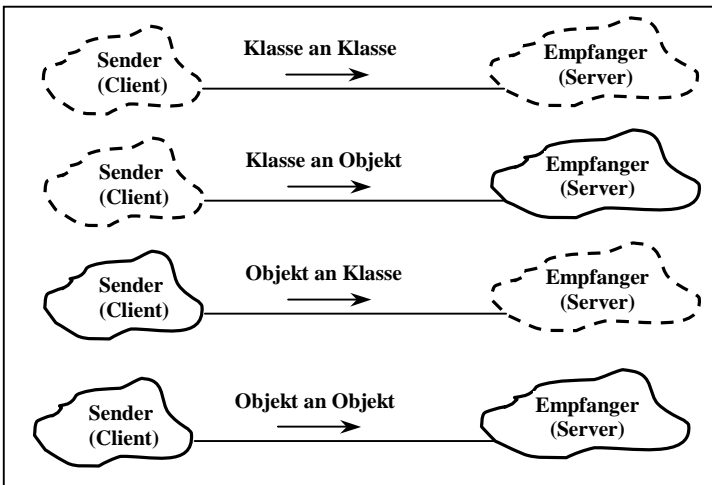


Abb.1.16. Austausch von Nachrichten

Das Empfängerobjekt führt die Methode aus und schickt das Ergebnis an das Senderobjekt zurück.

Mit Objekt-Diagrammen sind Ablaufszenarien - also exakte Abläufe von Systemfunktionen - festzuhalten. Die folgende Abbildungen 1.17 - 1.20 zeigen die Ablaufszenarien (OO-Diagramme) für eine Bibliothek.

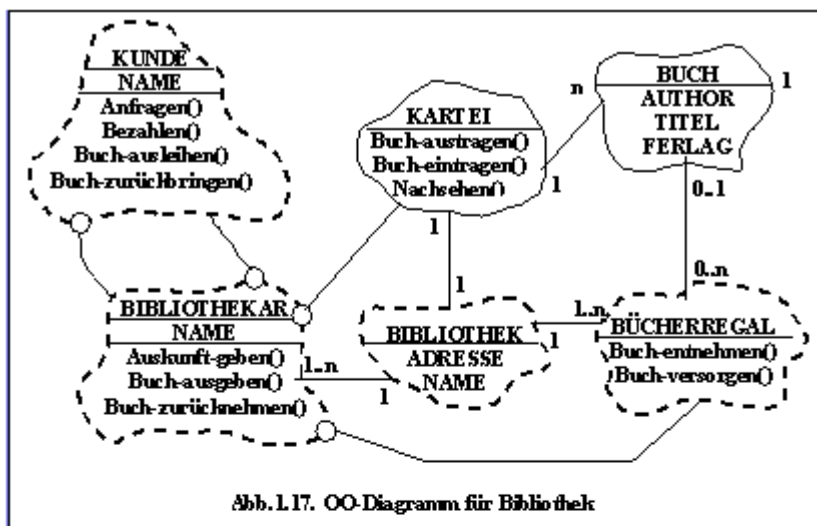


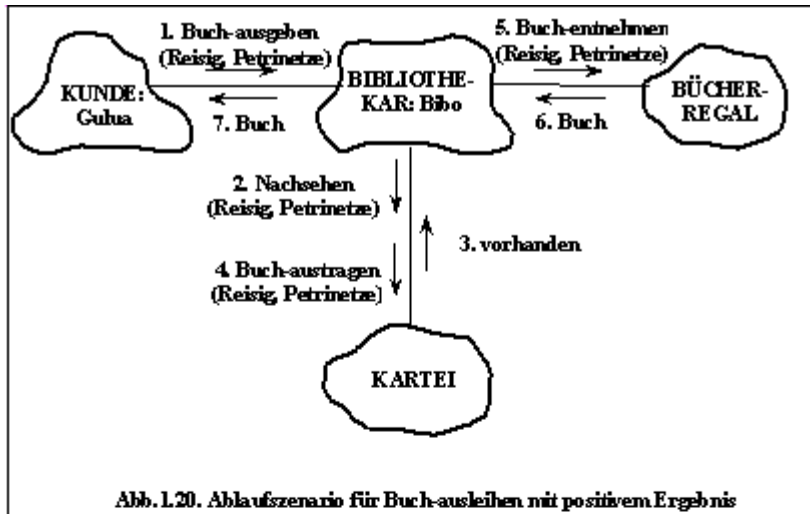
Abb. 1.17. OO-Diagramm für Bibliothek



Abb. 1.18. Ablaufscenario für Anfragen



Abb. 1.19. Ablaufscenario für Buch-ausleihen mit negativem Ergebnis



*Textuelle Beschreibung des Ablaufszenarios Buchausleihen:*

1. Bibliothekar.Buch-ausgeben(Reisig, Petrinetze)
2. Kartei.Nachsehen(Reisig, Petrinetze) → ... → finden
3. Kartei.Buch-austragen(Reisig, Petrinetze)
4. Bücherregal.Buch-entnehmen(Reisig, Petrinetze) → Buch
5. Ende der Methode → Buch.

#### Anmerkungen:

- Zustandsmodellierung wird als statische Objektmodellierung betrachtet. Für die Realitätsmodellierung verwendet man bekannte Begriffen der Modellierungskonstrukte: Klasse, Objekt, Vererbung, Beziehung, Relation, Aggregation.
- Zustandsmodellierung hat viele Ähnlichkeit mit der Entitäten-Relations Modellierung (ER-M).
- Verhaltensmodellierung wird als dynamische Objektmodellierung betrachtet. Hier verwendet man die Begriffe: Nachrichten, Methoden, Senderobjekt (Client), Empfängerobjekt (Server), Ereignisse, Ablaufszenario.
- Verhaltensmodellierung ist als Basis für die Use-Case Werkzeuge der Unified Modeling Language (UML).

## 9. GLOSSAR

**Abstraction.** *Abstraction ist eine Methode, bei der unter einem bestimmten Gesichtspunkt die wesentlichen Merkmale eines Gegenstandes oder Begriffes herausgesondert werden.*

**Abstrakte Klasse.** *Klassen, von denen keine konkreten Exemplare erzeugt werden koennen, d.h. von denen es grundsaeztlich keine Objekte geben wird.*

**Aggregation.** *Eine Aggregation ist eine Sonderform der Assotiation, bei der die beteiligten Klassen keine gleichwertige Beziehung fuhren, sondern eine Ganzes-Teile-Hierarchie darstellen. Eine Aggregation beschreibt, wie sich etwas Ganzes aus seinen Teilen zusammensetzt.*

**Analyse.** *Mit objektorientierter Analyse werden alle Procedures im Rahmen des Softwareentwicklungsprocesses bezeichnet, die der Ermittlung, Klaerung und Beshreibung der Anforderungen an das System dienen (d.h. die Klaerung was das System leisten soll).*

**Attribut.** *Eine benannte Eigenschaft eines Typs. Ein Attribut ist ein Datenelement, das in jedem Objekt einer Klasse gleichermassen enthalten ist und von jedem Objekt mit einem individuellen Wert repraesentiert wird.*

**Design.** *Mit objektorientiertem Design werden alle Procedures im Rahmen des Softwareentwicklungsprocesses bezeichnet, mit denen ein Modell logisch und physisch strukturiert wird und die dazu dienen zu beschreiben, wie das System die in der Analyse beschriebenen Anforderungen erfuehlt.*

**Ereignis (event)** ist ein Geschehen, das in einem gegebenen Kontext eine Bedeutung hat und sich räumlich und zeitlich lokalisieren lässt.

**Exemplar (Objekt, Instanz)**.

**Generalisierung** - (Ant. von **Spezialisierung** - konkretisierung).

**Geschäftsprozess (Workflow)** ist eine Zusammenfassung von organisatorisch verteilten, fachlich jedoch zusammenhängenden Prozeduren, die notwendig sind, um einen Geschäftsvorfall (z.B. einen konkreten Antrag) ergebnisorientiert zu bearbeiten.

**Geschäftsvorfall (Vorgang)**. Ein Geschäftsvorfall ist ein (Geschäfts-) Objekt (z.B. ein konkreter Vertrag), das durch ein Ereignis ausgelöst (z.B. Antragseingang) durch innerhalb eines Geschäftsprozesses beschriebenen Prozeduren bearbeitet wird..

**GUI (Graphical User Interface)** ist grafische Benutzeroberfläche.

**Interface - Schnittstelle**.

**Kardinalität - Multiplizität**. Anzahl der Elemente.

**Klasse**. Eine Klasse beschreibt die Struktur und das Verhalten einer Menge gleichartiger Objekte.

**Mehrfachvererbung**. Eine Klasse hat mehrere direkte Oberklassen.

**Merkmal - Eigenschaftswert**.

**Methode**. Das ist eine **Operation** (oder der Implementierung der Operation).

**Nachricht**. Nachrichten sind ein Mechanismus, mit dem Objekte untereinander kommunizieren können. Ein Nachricht fordert ein Objekt zur Ausführung einer Operation auf.

**Objekt**. Ein Objekt ist eine konkret vorhandene und agierende Einheit mit eigener Identität und definierten Grenzen des Zustands

und Verhalten kapselt. Der Zustand wird repräsentiert durch die Attribute und Beziehungen, das Verhalten durch Operationen bzw. Methoden.

**OO** ist die Abkürzung für Objektorientierung.

**Polymorphismus.** Vielgestaltigkeit heißt, dass gleichlautende Nachrichten an kompatible Objekte unterschiedliches Verhalten bewirken können.

**Sichtbarkeitskennzeichen** schränken die Zugriffbarkeit von Attributen und Methoden (private, protected, public).

**Vererbung** (inheritance). Unterklassen dürfen die Eigenschaften ihrer Oberklassen mitbenutzen.

**Virtuelle Klasse** (Operation) ist die Abstrakte Klasse (Operation).

**Workflow** ist die computergestützte Automatisierung des Geschäftsprozesses.

## **10. FRAGEN UND ÜBUNGEN**

1. Womit ist der Zustand eines Objekts festzuhalten ?
2. Was bestimmt das Verhalten eines Objekts ?
3. Was bedeutet der Begriff Datenkapsel ?
4. Was repräsentieren Klassen und Objekte ?
5. Wie erfolgt die Kommunikation zwischen Objekten ?
6. Was bedeutet Generalisierung und Spezialisierung ?
7. Man generalisiere die Klasse MITARBEITER.
8. Man spezialisere die Klasse MITARBEITER.
9. Was bedeutet Klassifikation und Aggregation ?

10. Was versteht man unter Vererbung, Mehrfachvererbung ?
11. Was ist Polymorphismus ?
12. Wie sind objektbasiertes, klassenbasiertes und objektorientiertes Vorgehen zu charakterisieren ?
13. Worin unterscheiden sich Beziehungsstrukturen von Vererbungsstrukturen ?
14. Welche Beziehungsarten unterscheidet man ?
15. Was ist Abbildung und wie klassiert man sie ?
16. Was wird von Superklassen an Subklassen vererbt ?
17. Was sind Methoden und womit sind sie aktivieren ?
18. Was unterscheidet ein Objektdiagramm von einem Klassendiagramm ?
19. Was ist ein Ablaufszenario ?
20. Kostruiere ein Ablaufszenario für PRÜFUNG im Institut.
21. Kostruiere ein Objektdiagramm für EINKAUF im Gescheft.

### **LITERATUR**

1. Bothe K., Surguladze G. *Modern Platforms and Languages of Programming*. Tbilisi, 2003.
2. Vetter M. *Objektmodellierung. Eine Einfuehrung in die objektorientierte analyse und das objektorientierte Design*. Zuerich, Stuttgart, 1995.
3. Gogichaishvili G., Surguladze G., Shonia. *Methods of Programming: C & C++*, GTU, Tbilisi, 1997..
4. Oestereich B. *Objektorientierte Softwareentwicklung*. München, 1998.
5. Bothe K., Surguladze G. Inheritance in the Programming of MIS: from the Databases to the UML Technology. GTU, 4(437),2001