Informatik an Beruflichen Gymnaien

Jahrgangsstufe 1

- **Relationale Datenbanken**
- **Objektorientierte Systementwicklung**

5. Auflage, 2014

von
Dipl.-Hdl. Wolfgang Braun

Leseprobe (Ausschnitte)

1 Datenbanken als Informationssysteme

Fast jeder von uns wird heutzutage mit einer lawinenartig anwachsenden Flut von Informationen geradezu überschwemmt. Parallel dazu werden die Möglichkeiten an Informationen zu gelangen, diese auszuwerten und eventl. zu speichern immer umfangreicher.

Damit sind wir bereits mitten im Thema, nämlich zu analysieren, inwieweit Datenbanken ein Mittel des Informationsmanagements sind.

Somit stellen sich folgende Fragen:

- ***Welche*** Informationen werden benötigt?
- ***Wo*** sind die relevanten Informationen zu finden?
- ***Wie*** gelangt man an relevante Information?
- ***Wieviel*** *Zeit investiert man zur Informationsbeschaffung und –auswahl?*

Willkommen im Informationszeitalter!

In vielen historischen Bibliotheken[1] „ruhen“ Wissensschätze, die über Jahrhunderte hinweg von Generation zu Generation zusammengetragen und weitergegeben wurden.

Datenverwaltungssysteme sind nicht zwingend an Computer gebunden was aber bedeutet, dass man „herkömmliche“ Systeme persönlich aufsuchen muss, um nach bestimmten Daten zu recherchieren, wovon ganze Generationen von Studenten ein Lied davon singen können.

Kann man solche Datensammlungen, die von vielen fleißigen Händen – oft in mühsamer Handarbeit erstellt wurden – als **Datenbank(en)** bezeichnen?

[1] Kloster Schussenried, Bibliothek, Oberschwaben, Baden- Württemberg

Hier ein Definitionsversuch des Begriffes **Datenbank:**

Eine **Datenbank** ist eine elektronisch gespeicherte Sammlung unterschiedlicher Daten, die thematisch (logisch) in irgendeiner Form zusammengehören und auf die über verschiedene Suchkriterien zugegriffen werden kann, z. B. Datenbank über vorhandene Artikel in einer Filiale, Datenbank über aktive Vulkane, Datenbank über die Schüler einer Schule, Verkehrssünder-Datenbank in Flensburg, Datenbank über das Weltwissen (Wikipedia?) u. v. m.).

In diesen Datensammlungen sind Auszüge der realen Welt abgebildet, wobei nur diejenigen Eigenschaften berücksichtigt sind, die für die Arbeit mit der Datenbank von Bedeutung sind. So enthält z. B. eine Datenbank über Sportartikel Informationen zu Artikelnummern, Preise, Lieferanten, Mengen, nicht aber (zwangsläufig) zu Farbe des Verpackungsmaterials, Familienstand des Vertreters, Anzahl der weiblichen Mitarbeiter des Lieferanten usw. In der Datenbank befindet sich also nur ein Ausschnitt der Objekte aus der realen Welt.

Auch dann, wenn man die Bücher in der oben abgebildeter historischen Bibliothek Seite für Seite einscannen und abspeichern würde, könnte man noch nicht von einer Datenbank sprechen, denn der Begriff bezeichnet zwei verschiedene Dinge:
Zum einen die **Datensammlung** selbst, zum anderen das **Programm**, das diese Daten verwaltet.

Bei den Daten handelt es sich um eine nach bestimmten Regeln strukturierte Ansammlung von Informationen zu verschiedenen Themengruppen.

Das Anwendungsprogramm, mit dem diese Daten verwaltet werden können, enthält mehr oder weniger mächtige Funktionen zum Suchen, Sortieren, Filtern und benutzerfreundlicher Ausgabe dieser Daten. Ein solches System wird als **Database Management System (DBMS),** also als Datenbankverwaltungssystem[2] bezeichnet.

1.1 Kampf dem Datenchaos

Salopp konnte man sagen:

Mit Datenbanken wird der Versuch unternommen Teile der Wirklichkeit abzubilden.

Da die Wirklichkeit aber sehr komplex ist, um sie vollends digital abzubilden, reduziert man die Informationen meist auf einen sehr kleinen, aber ausschlag-

[2] vgl. S. 10 ff.

gebenden Bestandteil. Eine Kundendatenbank enthält z. B. meist Adressen, Telefonnummern und andere zur Abwicklung der Geschäfte wichtige Informationen. Dies ist nur ein kleiner Ausschnitt über die Realität der Kunden, da z. B. Informationen über das Privatleben und das Umfeld der Kunden nicht gespeichert werden. Hier sind Aspekte des Datenschutzes zu beachten, weshalb diesem wichtigen Punkt bereits im Buch für die Eingangsklasse dieser Buchreihe ein eigenes Kapitel gewidmet ist.[3]

Durch unkontrolliert wachsende Datenbestände ist jedoch in bestimmten Bereichen ein **Datenchaos** entstanden, wovon Herr Berger, Eigentümer des Sport- und Freizeithauses ***BergerSports*****,** ein Lied singen kann. Oft wird dieses Datenchaos durch die eigenständige isolierte Datenhaltung für einzelne Anwendungen verursacht. So kommt es häufig vor, dass Kundendaten für unterschiedliche Anwendungen jeweils neu gespeichert werden, z. B. für Abrechnungen, für die Zusendung von Werbematerial, für Service und Beratung.

Liebe Grüße vom **Ordnungsteufel**

Vielleicht etwas übertrieben, aber deutlich ausgedrückt:

> "Das Jahrhundertproblem der Informatik besteht in der Bewältigung des Datenchaos, das infolge historisch, mitunter auch hysterisch und archaisch, sicher aber unkontrolliert gewachsener Datenbestände fast überall entstanden ist." [4]

Somit lautet eine Hauptaufgabe der Datenbankerstellung, die Komplexität der Daten zu reduzieren, um sie so in strukturierter Form bearbeiten zu können. Wird mit mehreren Datenbanken (DB) gearbeitet, ist ein Verwaltungsprogramm erforderlich, das das Ganze steuert. Dieses Verwaltungsprogramm wird als **Datenbankmanagementsystem** (DBMS) bezeichnet.

Ein Datenbanksystem (DBS) besteht aus einem Datenbankmanagementsystem und einer bzw. mehreren Datenbanken; kurz:

DBS = DBMS + nDB (n >=1)

Ein Datenbankmanagementsystem übernimmt somit die zentrale Verwaltungs-, Steuerungs- und Kontrollfunktion der Datenbank(en), das auch für die Kommunikations mit seiner Umwelt verantwortlich ist. Es hat folgende grundlegende Aufgaben zu erfüllen:

- Durchführung der Datenspeicherung
- Durchführung der Datenmanipulation (Arbeiten mit der Datenbank)
- Gewährleistung des Zugriffs auf die Daten je nach Berechtigung
- Optimale Ausnutzung des vorhandenen Speichers
- Gewährleistung der Datensicherheit und des Datenschutzes

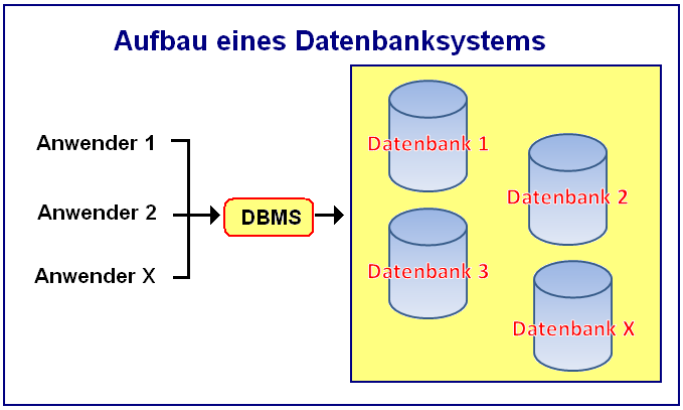
---

[3] Vgl. Wolfgang Braun, Informatik für berufliche Gymnasien, Eingangsklasse

[4] Vetter, M.: Das Jahrhundertproblem der Informatik, in: Müller-Ettrich (Hrsg.): Effektives Datenbankdesign, Köln 1989, S. 11-31.

Zur Erfüllung dieser genannten Aufgaben muss das Datenbanksystem mit seiner Umwelt kommunizieren können[5]. Dazu gehört die Kommunikation mit

- dem DV-System über das Betriebssystem,
- Anwenderprogrammen, insbesondere mit Programmsystemen, die auf die Datenbank zugreifen,
- dem Datenbankbenutzer, der als Endbenutzer die Datenbank nutzt, d. h. auf die Daten zugreift und mit ihnen arbeitet,



- dem Anwendungsprogrammierer, der eine Datenbank nutzt,
- dem Systemverwalter, der die Endanwender und Anwendungsprogrammierer bei ihrer Arbeit unterstützt.

1.2 Informationsverarbeitung beim Sport- und Freizeithaus *Berger Sports*

PROBLEMSTELLUNG

Herr Berger formuliert folgenes Ziel:

„Wir müssen weg vom Datenchaos, hin zur geordneten Datenverwaltung!“

Das Sport- und Freizeithaus im Wandel der Zeit

Herr Berger erkennt die Zeichen der Zeit. Seine Geschäftspolitik trägt dem gesellschaftlichen Trend, einer zunehmenden Kommerzialisierung von Freizeitaktivitäten Rechnung. Hierzu gehört die Ausweitung des bestehenden Sportartikelgeschäftes um Geschäftsbereiche wie Fitnesssport, Touristik, Erlebnissport, Sportrehabilitation, aber auch Mode, Sportartikelverleih und Entertainment. In allen Abteilungen sind bereits mit Erfolg Textverarbeitungs-, Tabellenkalkulations-, Kommunikationsprogramme und diverse Tools im Einsatz. Gerade im Bereich der Datenbankanwendungen besteht jedoch noch ein großer Bedarf, die bisherigen Anwendungen zu optimieren. Auch wurde bisher z. B. gänzlich darauf verzichtet, die Möglichkeit der Programme zu nutzen, Vorgänge zu automatisieren.

Entsprechend werden zuerst die Datenbankanwendungen und danach die Möglichkeiten der Objektorientierten Programmierung in diesem Buch vorgestellt.

[5] Weitergehende Informationen finden Sie unter:
www.sql-und-xml.de/sql-tutorial/datenbank-grundbegriffe.html

Merke

- Unter einem **Datenbanksystem** versteht man die Zusammenfassung der Komponenten **Datenbank (DB)** und **Datenbankmanagementsystem** (DBMS).
- Unter einer **Datenbank** versteht man eine Menge von logisch zusammengehörenden Daten, die zur Deckung des Informationsbedarfes verschiedener Benutzer durch ein automatisiertes Datenverarbeitungssystem unter Berücksichtigung der logischen Beziehungen verwaltet werden.
- Unter einem **Datenbankmanagementsystem** versteht man ein Softwaresystem zur Verwaltung von Datenbanken und Datenbankanwendungen.

1.3 Datenmodellierung: Von der realen Welt zum Modell

Folgendes Beispiel versucht die oben bereits beschriebene Problematik, Teile der Realität abzubilden, zu verdeutlichen.

Realsphäre

Yvonne Hauser
Hauptstraße 89
12345 Irgendwo

„Die lustigen Drei“
Marienplatz
54321 Anderswo

Der Radgeber
Service rund ums Rad
Wiesenweg 4
73549 Mauer

Zielgerichtete Abstraktion durch reales Ausgrenzen von Elementen der Realsphäre (konzeptionelles Schema).

Kunde

KdNr	Name	Straße	PLZ	Ort

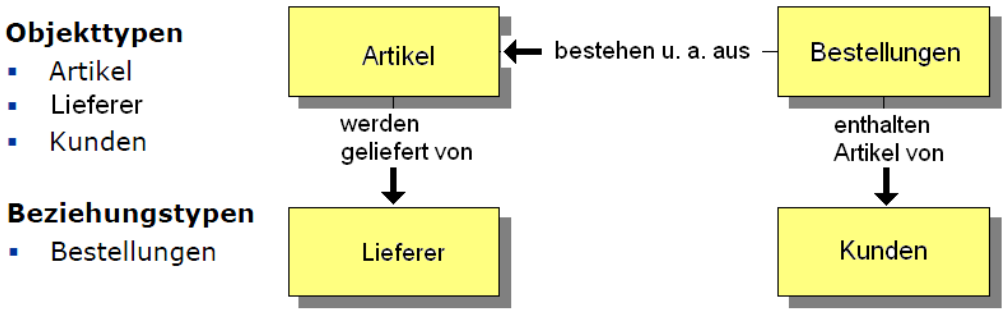
Zusammenstellung logisch zusammengehörender Daten

Aussschnitt einer Datenbank

KdNr	Name	Straße	PLZ	Ort
1	Hauser, Yvonne	Hauptstraße 89	12345	Irgendwo
2	Die lustigen Drei	Marienplatz 7	54321	Anderswo
3	Der Radgeber	Wiesenweg 4	73549	Mauer

- Um eine Datenbank aufzubauen, muss zunächst ein konzeptionelles Schema definiert werden, das ein möglichst exaktes Bild der realen Welt präsentiert.
- Zunächst wird durch Analyse der Realwelt ein zu modellierender Ausschnitt (**Miniwelt**) definiert.
- Beim Betrachten der Miniwelt erkennt man gewisse **Objekte** (**Entities**) wie Artikelnamen, Kundennamen, Kundenadressen usw. Zwischen diesen Objekten bestehen **Beziehungen** (**Relationships**).
- Um zu einem Datenmodell zu kommen, fasst man gleichartige Objekte und gleichartige Beziehungen zu **Typen** zusammen.

Untenstehende Abbildung zeigt ein vereinfachtes Objekttypen- und Beziehungsmodell zwischen Kunden und Lieferern.



Merke

Das Datenmodell ist eine sprachliche Beschreibung der Datenelemente einer real existierenden Welt innerhalb eines Informationssystems. Man unterscheidet zwischen der

a) **konzeptionellen Datenmodellierung**
als formale Beschreibung der Datenstrukturen,

b) **logischen Datenmodellierung** als formale Beschreibung der Datenstrukturen unter Verwendung des jeweiligen Datenbankmodells,

c) **physischen Datenmodellierung** als Überführung des logischen Datenmodells in das jeweilige Datenbanksystem.

Je nach dem verwendeten Datenbankverwaltungssystem kommen dafür zz. in Betracht:

1. **das hierarchische Modell,**
2. **das Netzwerkmodell,**
3. **das relationale Modell.**

Die ersten beiden Modelle sind nicht Gegenstand dieses Buches.

Doch jetzt zurück zum Sport- und Freizeithaus *BergerSports.*

PROBLEMSTELLUNG

Eine Vertriebsmitarbeiterin des Sport- und Freizeithauses ***BergerSports*** hat u. a. den Auftrag, die Bestellung der Firma Intersport GmbH zu bearbeiten.

Intersport GmbH
Murgweg 3
76437 Rastatt

12.10.20..

BergerSports
Rödernweg 1
76437 Rastatt

Bestellung

Sehr geehrte Damen und Herren,

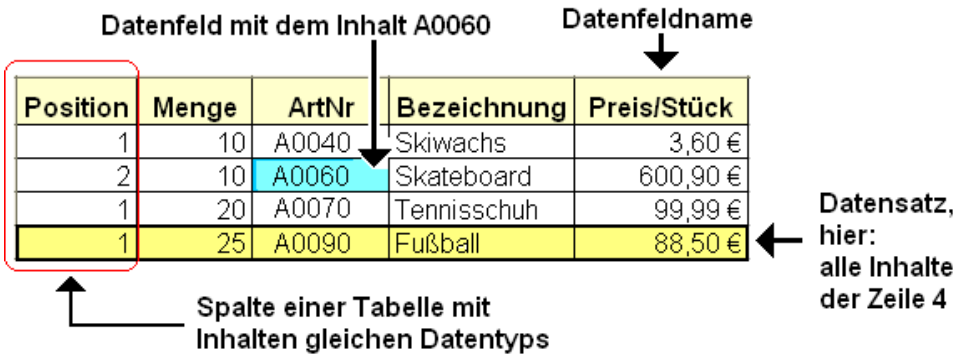
gemäß Ihrem Katalog bestellen wir:

Position	Menge	ArtNr	Bezeichnung	Preis/Stück
1	10	A00040	Skiwachs	3,60 €
2	10	A00060	Skateboard	600,90 €

Bitte liefern Sie die Artikel bis spätestens 22.10.20..

Mit freundlichen Grüßen

Werden die wesentlichen Elemente des Kundenauftrags in tabellarischer Form zusammen mit anderen Aufträgen angeordnet, ergibt sich folgende Darstellung.



Hieraus lassen sich wichtige Gesetzmäßigkeiten ableiten:

1. Unter den Spaltenüberschriften stehen immer gleichartige Daten. So stehen z. B. unter der Spaltenüberschrift “Menge” ausschließlich ganzzahlige Stückzahlen. Allgemein gilt: Spaltenweise dargestellte Daten haben immer denselben **Datentyp**. Grundlegende Datentypen sind Zahlen und Zeichenketten, doch davon später mehr.
2. Jedes Element einer Tabelle bezeichnet man als **Datenfeld**.
3. Die Spaltenüberschriften werden als **Datenfeldnamen** (Attributnamen) bezeichnet.
4. Jede Zeile einer Tabelle enthält unterschiedliche Daten zu einem gegebenen Sachverhalt. Man bezeichnet eine solche Zeile als **Datensatz** (Tupel bzw. Row).
5. Alle Schnittpunkte von Zeilen und Spalten haben Werte (Daten, Merkmalsausprägungen, Inhalte).

Wichtig

Um jeden Datensatz eindeutig zu bestimmen, sind **Primärschlüssel** (Primary Key, Identifikationsschlüssel) zu vereinbaren. Dabei handelt es sich um ein Feld oder auch mehrere Felder, die jeden Datensatz innerhalb der Tabelle eindeutig bestimmen. Informationen in Feldern, für die ein Primärschlüssel vereinbart wurde, dürfen sich nicht wiederholen. Deshalb legt man spezielle Primärschlüsselfelder an, beispielsweise eine Artikelnummer, eine Personalnummer oder eine Kundennummer. Diese Nummern werden nur einmal vergeben, sodass man eine eindeutige Zuordnung von Primärschlüssel und zugehörigem Datensatz erhält.

1.3.1 Das relationale Datenbankmodell

1.3.1.1 Entity-Relationship-Modellierung

Zur Modellierung der in einer Datenbank zu speichernden Daten wird in der Definitionsphase i. d. R. die Entity-Relationship-Modellierung (ER-Modellierung) eingesetzt, die bereits 1976 von P. Chen-entwickelt wurde. Das Modell der elationalen Datenbank basiert auf E. F. Codd. Er entwickelte ab 1970 ein Datenbankmodell, dessen Inhalte in Tabellenform vorliegen und untereinander in Beziehung stehen. Codd stellte u. a. die Regeln der Normalisierung auf, die bei der Erstellung von Datenbanken zu beachten sind. So ist es ein Zeil, redundante (überflüssige) Informationen und spätere Integritätsverletzungen zu vermeiden, was noch ausführlich erklärt wird.

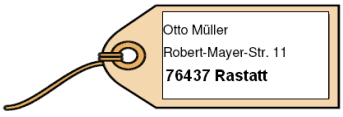
Das ER-Modell besteht aus den folgenden Elementen:

1. Entitäten (Entities)	2. Attribute	3. Beziehungen

Merke

Die **Entität** ist ein konkretes, eindeutig identifizierbares Objekt bzw. Exemplar von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt, das in Beziehung zu anderen Entitäten steht, für das einzelne ausgewählte oder alle relevanten Merkmale bzw. **Attribute** festzuhalten sind.

Der Kofferanhänger von Herrn Otto Müller beschreiben Attribute der Entität „Person“.



Neben den rein beschreibenden Attributen sind **identifizierende Attribute** (z. B. *Kundennummer, Personalnummer, Kursnummer, …)* von entscheidender Bedeutung. Jeder Wert eines identifizierenden Attributs darf genau einmal auftreten. Durch diese Attribute kann eine Entität innerhalb eines **Entitätstyps** eindeutig identifiziert werden. Solche Attribute übernehmen die Aufgabe eines **Schlüssels** (Key).

Die Entität ist Mitglied einer Gruppe, deren Mitglieder alle strukturell gleich sind, d. h. die gleichen Merkmale (Attribute) haben, sich aber in der konkreten Ausprägung unterscheiden. Diese Gruppe wird als **Entitätstyp** bezeichnet.

Die relativ abstrakte Sprache der Begriffe soll an einem Beispiel veranschaulicht werden.

Entities sind **Objekte** wie zum Beispiel ein Auto, das durch seine **Eigenschaften** beschrieben wird: Dieselmotor, fünf Türen, 180 km/h Höchstgeschwindigkeit. Um nicht nur ein bestimmtes Auto, sondern mehrere Autotypen mit verschiedenen Motoren und Geschwindigkeiten zu modellieren, werden die Entities typisiert. Der **Entity-Typ** Auto lässt sich z. B. durch die **Attribute** *Hersteller, Modell-Nr, Modell* und *Motortyp* beschreiben. Der Entity-Typ „Hersteller“ lässt sich z. B. durch die Attribute *Auto*, *Name* und *Standort* beschreiben. Der **Wertebereich** des Attributs *Hersteller* reicht vom Audi, BMW, Ford, Mercedes, Opel, ... Für das Attribut *Motortyp* sind als Wertbereich Motortypen wie Diesel-, Benzin- oder Elektromotoren denkbar. Die Felder *Hersteller* (Hersteller-Nr.) und *Auto* (Modell-Nr.) sind die jeweiligen **Schlüssel** der beiden möglichen Tabellen.

Beziehungstypen (Relationships)

Dass Beziehungen im Alltags- und Berufsleben eine wichtige Rolle spielen, „pfeifen die Spatzen von den Dächern.“ In der Datenbanktheorie gilt:

Eine Beziehung beschreibt mögliche Zusammenhänge zwischen Entitätstypen.

Doch vorab gilt es, dem Wirrwarr der Begriffe ein Ende zu erteilen: Es gelten folgende Synonyme:

Entität	entity	Objekt
Entitätsmenge	entity set	Entitätstyp
Beziehung	relationship	
Assoziation	relationship set	Beziehungstyp
Attribut	Spalte einer Tabelle	property

1.3.1.2 Grafische Darstellung in der Entity-Relationship-Modellierung

Das **Entity-Relationship-Modell** ist ein grafisches Hilfsmittel für den Entwurf eines Datenmodells. Es beinhaltet grafische Elemente zur Darstellung von Objekten (Objekttypen) und deren Objektbeziehungen (Relationship). Die Grundbausteine eines ER-Modells sind Objekttypen, die bestimmte Eigenschaften (Attribute) besitzen.

Ein Objekttyp kann in der Realwelt dabei alles Mögliche repräsentieren: Dinge, Prozesse oder Funktionseinheiten. Zwischen Objekten können Beziehungen bestehen, die einen unterschiedlichen Komplexitätsgrad aufweisen können. Ein **Entitytyp** wird durch ein Rechteck dargestellt. Darin steht der Name der Entität. Die Wechselwirkungen und Abhängigkeiten zwischen Entitäten werden durch Beziehungen verdeutlicht, in der Fachsprache auch als **Assoziation** bezeichnet. Beziehungen werden durch Rauten dargestellt. In einer Raute steht der Name der Beziehung. Als Name sollte ein Verb gewählt werden.

Folgende Beziehungskonstellationen (Assoziationstypen) zwischen zwei Objekten sind von Bedeutung:

Die 1:1-Beziehung

Eins zu eins: Jedem Objekt A wird genau ein Objekt B zugeordnet und umgekehrt.
Hier gilt: Ein Händler bildet einen Azubi aus. Ein Azubi lernt genau bei diesem Händler. Es gibt auch Entitäten, zwischen denen eine Beziehung stehen **kann, aber nicht muss.** Wo die Beziehung optional (conditional) ist, wird ein „o“ oder ein „c“ geschrieben.

> **Hinweis**
>
> Eine Beziehung ist auch optional, wenn sie den Wert „null“ haben kann.

Die 1:C-Beziehung

Eins zu keinem oder einem

((entity3.TIFF))

Hier gilt: Entweder bildet der Händler einen oder keinen Azubi aus.

Die 1:N-Beziehung

Eins zu einem oder mehreren.
Die 1:N-Beziehung ist der häufigste Beziehungstyp.

((entity4.TIFF))

Hier gilt: Ein Händler kann einen oder mehrere Azubis ausbilden. Umgekehrt gilt: Ein oder mehrere Azubis werden genau von einem Händler ausgebildet.

Unterbrechnung der Leseprobe

1.4 Datenmodellierung einer Auftragsverwaltung

Die **Normalisierung** ist ein mehrstufiger Prozess, bei dem Tabellen einer Datenbank organisiert, zerlegt und die Daten geordnet werden. Er dient in erster Linie der **redundanzfreien Speicherung** von Informationen innerhalb der Tabellen. Redundanzfreie Speicherung bedeutet, dass kein Teil eines Datenbestandes weggelassen werden kann, ohne dass dies zu Informationsverlusten führt. Die einzelnen Stufen der Normalisierung bauen aufeinander auf. Obwohl es mehrere Stufen dieses Optimierungsprozesses gibt, sollte eine Normalisierung bis zur 3. Normalform ausreichen. Somit ist gewährleistet, dass Datenredundanz (z. B. überflüssige Mehrfachnennungen) in den Datenbeständen vermieden wird.

AUSGANGSLAGE

Organisation und Bearbeitung von Kundenaufträgen

Das Sport- und Freizeithaus ***BergerSports*** besteht zz. aus den Abteilungen Vertrieb (Auftragsbearbeitung, Prüfung von Liefermöglichkeiten, Terminierung, Angebotserstellung), Lager (Prüfung des Wareneins- und –ausgangs, Bedarfsmeldung), Einkauf (Bestellung, Angebotspüfung, Beschaffungspolitik) und Verwaltung (Personal- und Anlageverwaltung).

Aus der Vielzahl dieser Aktivitäten wird im Folgenden der Teilprozess „Bearbeitung von Kundenaufträgen“ in Form einer zu erstellenden Datenbank besonders bearbeitet. Eine Tabelle, die Bestellungen von Kunden aufnimmt, könnte z. B. folgendes Aussehen haben.

Tabelle AUFTRAGSVERWALTUNG als „chaotische“ Tabelle

AuftrDat	Artikel	Menge	Preis	Umsatz	Kunde
12.10.20..	A0040 Skiwachs	10	3,60	36,00	Intersport, 76437 Rastatt, Murgweg 3
12.10.20..	A0060 Surfbrett	10	600,90	6.009,00	Intersport, 76436 Rastatt, Murgweg 3
18.10.20..	A0070 Skateboard	30	99,90	2.997,00	Froh OHG, 12345 Irgendwo, Bachstr. 6
usw.					

Aus der oben stehenden Tabelle ergeben sich vielfältige Probleme:

1. Mehrere gleiche Einträge führen zu unnötigen Wiederholungen **(Datenredundanzen).** So ist z. B. die Adresse der Firma Intersport mehrfach gespeichert, was weitere Probleme zur Folge hat.
2. Welche PLZ stimmt? 76437 oder 76436? Diese unterschiedlichen Angaben führen zu **Dateninkonsistenzen**.
3. Was ist, wenn ein Kunde hinzugefügt werden soll? In unserer Tabelle kann dies nur dann geschehen, wenn er einen Sportartikel kauft. Ansonsten entsteht eine unvollständige Zeile = **Einfügeanomalie**.

4. Im Gegensatz zu Punkt 3 gehen dann Informationen über die Kundenanschrift verloren, wenn der Auftrag eines Kunden gelöscht wird = **Löschanomalie.**
5. Ferner ergeben sich Probleme, wenn ein Kunde seinen Firmensitz in einen anderen Ort verlegt. Hier sind Änderungen in Datensätzen vorzunehmen, was u. U. zu **Änderungsanomalien** führen kann.

Folge: Die Tabelle muss optimiert werden. Es gilt also, die „chaotische“ Tabelle in die jeweiligen **Normalformen** zu überführen.

Regel zur Stufe 1 der Normalform

Eine Relation befindet sich in der 1. Normalform, wenn alle Attribute nur einfache Attributwerte aufweisen.

Tabelle AUFTRAG in der 1. Normalform

AuftrNr	AuftrPos	AuftrDat	ArtNr	ArtBez	AuftrPreis	AuftrMenge
A01-20	1	12.10.20..	A0040	Skiwachs	3,60	10
A01-20	2	12.10.20..	A0060	Surfbrett	600,90	10
A02-20	1	18.10.20..	A0070	Skateboard	99,90	30

Umsatz	KdNr	Name	Straße	PLZ	Ort
36,00	K1030	Intersport	Murgweg 3	76437	Rastatt
6.009,00	K1030	Intersport	Murgweg 3	76437	Rastatt
2.997,00	K1080	Froh OHG	Am Bach 5	12345	Irgendwo

In der 1. Normalform gibt es nur noch einen einzigen Wert pro Datenfeld. Das Feld *AuftrNr* wurde u. a. deshalb eingefügt, um alle Datensätze eindeutig zu identifizieren. Wenn sich die Elemente in einem solchen Feld nicht wiederholen dürfen (dies trifft hier noch nicht zu), spricht man von einem **Primärschlüsselfeld.**

Jetzt ist die Frage zu beantworten, ob jedes Merkmal unmittelbar vom Schlüssel abhängt.

Da die Artikelnummer sicher nicht von der Auftragsnummer abhängt, kann obige Lösung nicht akzeptiert werden. Ein weiterer Nachteil dieser Zwischenlösung liegt darin, dass immer dann, wenn z. B. ein anderer Kunde das gleiche Surfbrett bestellt, dieser Artikel jeweils mit Bezeichnung und Preis usw. neu eingegeben werden müsste. Dies kostet Platz, Zeit und der "Tippfehlerteufel“ wartet bereits. Somit gilt es, die 2. Normalform herzustellen.

Regel zur Stufe 2 der Normalform

Eine Relation befindet sich in der 2. Normalform, wenn sie schon in der 1. Normalform vorliegt. Hierbei müssen alle nicht zum Schlüssel gehörenden Attribute von diesem voll funktional abhängig sein. Besteht ein Schlüssel aus mehreren Teilschlüsseln, so ist das Element aus dem Datensatz herauszuziehen, das nur von einem Teilschlüssel abhängt. Anders ausgedrückt:

Jedes Nicht-Schlüsselfeld muss durch ein Schlüsselfeld identifizierbar sein.

Entfernt man die obigen unkorrekten Abhängigkeiten, so ergibt sich folgende **Zwischenlösung:**

Tabellen in der 2. Normalform

Tabelle: AUFTRAG

AuftrNr	KdNr	AuftrDat	Name	Straße	PLZ	Ort
A01-20	K1030	12.10.20..	Intersport	Murgweg 3	76437	Rastatt
A01-20	K1030	12.10.20..	Intersport	Murgweg 3	76437	Rastatt
A02-20	K1080	18.10.20..	Froh OHG	Am Bach 5	12345	Irgendwo

Tabelle: AUFTRAGSPOS

AuftrNr	AuftrPos	ArtNr	ArtBez	AuftrMen-ge	AuftrPreis	Umsatz
A01-20	1	A0040	Skiwachs	10	3,60	36,00
A01-20	2	A0060	Surfbrett	10	600,90	6.090,00
A02-20	1	A0070	Skateboard	30	99,90	2.997,00

Alle Informationen, die nicht direkt mit den einzelnen Bestellungen zu tun haben, wurden in separate Tabellen ausgegliedert. Es können aber immer noch Anomalien auftreten, da z. B. Kundennamen in der Tabelle AUFTRAG mehrfach zu nennen sind. Dasselbe gilt u. a. für die Postleitzahlen und Orte. Müssten also z. B. Kundennamen oder Artikelbezeichnungen geändert werden, so müsste die Änderung in mehreren Datensätzen erfolgen, woraus folgt, dass das Redundanz- und Anomalienproblem noch nicht vollständig gelöst ist. Ferner sind z. B. in der Tabelle AUFTRAG Kundenadressen nicht direkt abhängig von der Auftragsnummer. Es liegen noch immer **transitive Abhängigkeiten** vor. Letztendlich zu falschen Ergebnissen führen solche Abhängigkeiten in der Tabelle Auftragsposition. Der Umsatz ist nicht von der Auftragsnummer abhängig, sondern von der Multiplikation aus *AuftrPreis * AuftrMenge*. Solche transitiven Abhängigkeiten sind zu eliminieren.

Regel zur Stufe 3 der Normalform

Eine Relation befindet sich in der 3. Normalform, wenn sie schon in der 2. Normalform ist. Zwischen Spalten, die nicht den Schlüssel bilden, besteht keine Abhängigkeit (transitive Abhängigkeit).

Man könnte obige Regel auch wie folgt lesen:

Alle Datenfelder sind nur vom gemeinsamen Schlüssel abhängig; untereinander treten keine Abhängigkeiten auf.

1.5 Eine neue Datenbank entsteht

1.5.1 Ziele und Aufgaben

- Verwaltung wichtiger Geschäftsprozesse des Sport- und Freizeithauses ***BergerSports*** und ihrer Geschäftspartner
- Verwaltung von Kundenaufträgen
- Verwaltung aller Aktivitäten der Außendienstpartner (Vertreter) inkl. der Berechnung der umsatzabhängigen Vertreterprovisionen

1.5.2 Datenbankobjekte näher betrachtet

Egal welches Datenbankprogramm Sie verwenden, eines haben diese Programme gemeinsam:

Sie verwalten **Datenbankobjekte**.

Nebenstehende Abbildung zeigt die wichtigsten Datenbankobjekte.

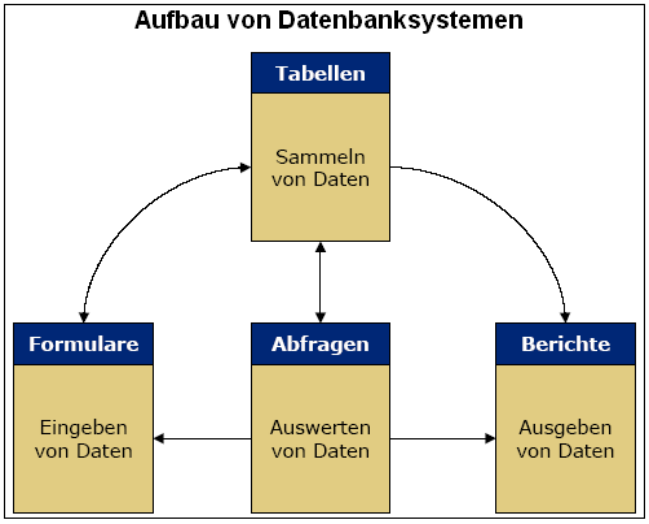


Tabelle Eine Tabelle ist eine Zusammenstellung von Daten zu einem bestimmten Thema. Sie enthält eine zeilen- und spaltenweise angeordnete Sammlung von zusammengehörenden Informationen, wobei jede Zeile genau einen, aus mehreren Datenfeldern bestehenden Datensatz enthält.

Formular Ein Formular ist für Eingabe, Änderung und Einsicht von Datensätzen in der Datenbank vorgesehen. Beim Öffnen eines Formulars werden die Daten von Tabellen oder Abfragen aufgerufen und in einem speziellen Layout ausgegeben.

Abfrage Eine Abfrage in einer Datenbank kann sich auf Datensätze einer oder mehrerer Tabellen beziehen. Eine Abfrage stellt angeforderte Informationen nach bestimmten Kriterien zusammen.

Bericht Ein Bericht ist eine Zusammenfassung von Werten mehrerer Datensätze. Er dient u. a. einer aussagekräftigen Präsentation dieser Daten. Im Bericht können aber nicht nur Datensätze zusammengestellt werden, sondern auch Gruppierungsfunktionen für einzelne Datensatzgruppen. Es kann auch eine Gesamtsumme für Zahlen berechnet werden. In einem Bericht können - im Gegensatz zu einem Formular - keine Daten eingegeben werden.

Makro[6] Mithilfe von Makros können grundlegende Funktionen automatisiert werden. Ein Makro kann eine Liste von Aktionen oder ein ablauffähiges Programm sein.

Modul[8] Ein Modul enthält in Microsoft Access Basic geschriebene Anweisungen. Access Basic ist eine integrierte Datenbanksprache, die sich an Visual Basic anlehnt.

Arbeitsauftrag

Erstellen Sie mit einem Datenbankprogramm, z. B. *Access* oder *LibreOffice,* eine Datenbank mit dem Ziel der Optimierung der Abwicklung von Kundenaufträgen.

Tabellen: *Artikel, Eindeut_PLZ, Auftrag, AuftragsPos, Kunden, Bestellungen, Vertreter, ArtLief, Rabstaffel, Lieferer* und *BestellPos.*

Hinweis

Standardmäßig beziehen sich die Bildschirmabbildungen auf den folgenden Seiten auf **MS Access.** Immer dann, wenn sich wesentliche Abweichungen zu **LibreOffice** ergeben, ist dies extra vermerkt.

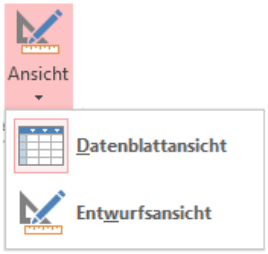
Beginnen wir zunächst mit den Tabellen *Eindeut_PLZ* und *Kunden*.

Arbeitsaufträge

1. Legen Sie auf Ihrem Datenträger einen neuen Ordner an.
2. Starten Sie ein Datenbankprogramm.

[6] Makros und Module werden in diesem Buch nicht behandelt.

Bei MS Access gehen Sie folgendermaßen vor:



- ► leere Desktopdatenbank
- ► in die Entwurfsansicht wechseln (siehe Abbildung)

Unter LibreOffice gilt:

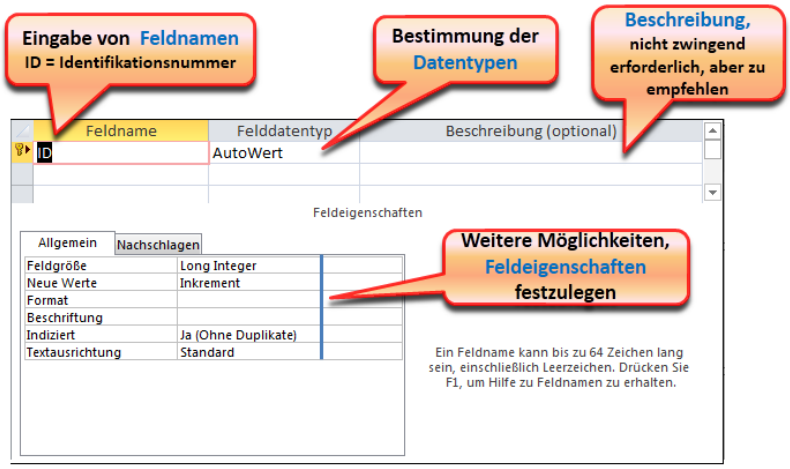
- ► neue Datenbank erstellen

3. Bestimmen Sie den Speicherort und wählen Sie den Datenbanknamen, in unserem Fall: *Auftragsverwaltung_Berger_1*

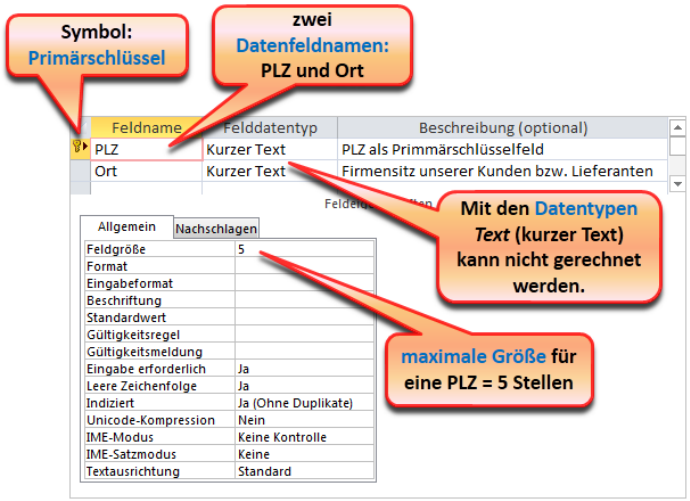
Wichtig

In der **Entwurfsansicht** erstellen Sie die Strukturen von Datenbankobjekten, hier die erste Tabelle. Die **Datenblattansicht** liefert die Tabellenansicht, also die eigentliche Tabelle.

Wichtige Aktionen im Entwurfsfenster



Entwurfsfsansicht der Tabelle
Eindeut_PLZ



Hinweis zu LibreOffice

Wollen Sie die Struktur einer bestehenden Tabelle ändern, so klicken Sie im Datenbankfenster mit der rechten Maustaste auf die zu Ändernde Tabelle und wählen den Menüpunkt *Bearbeiten*. Jetzt sind Sie in der Entwurfsansicht.

Das **Primärschlüsselfeld** ist zu bestimmen. Sie erkennen dies an dem Schlüsselsymbol, links neben der Spalte „Feldname“. Bei LibreOffice empfiehlt es sich mit der rechten Maustaste im Tabellenentwurfsfenster auf den Zeilenkopf vor der Zeile *PLZ* zu klicken. Jetzt können Sie die Option *Primärschlüssel* wählen.

Arbeitsaufträge

1. Erstellen Sie die abgebildete Tabellenstruktur.
2. Geben Sie einige Daten ein.

Tipp

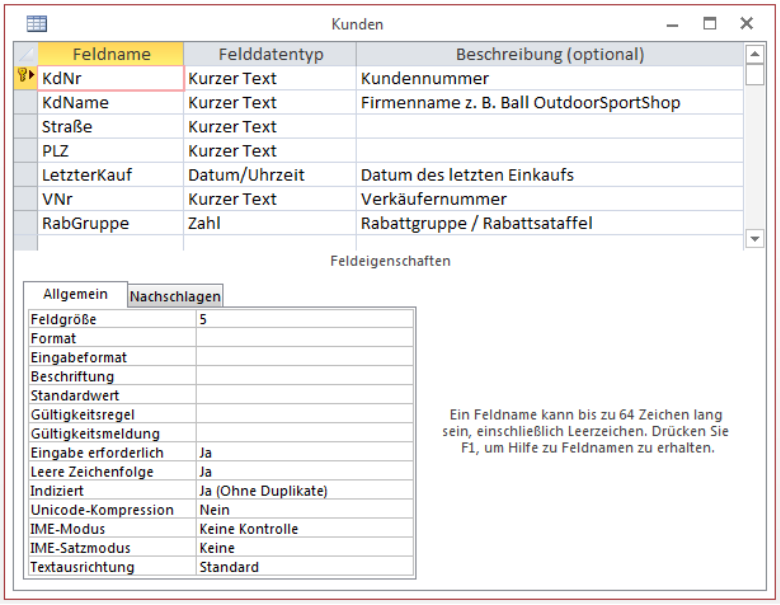
Wenn Sie Postleitzahlen suchen, finden Sie diese unter folgender Adresse: www.postdirekt.de/plzserver/

PLZ	Ort
10319	Berlin
10589	Berlin
12161	Berlin
12345	Irgendwo
13595	Berlin
14004	Berlin
22323	Freistätten
24345	Sonnleiten
32451	Hausen
41224	Weisenbach
42001	Rheinau
43532	Steinsfeld
43654	Bergried
45387	Bonndorf
69115	Heidelberg
69117	Heidelberg
69126	Heidelberg
76412	Murghausen
76437	Rastatt
76530	Baden-Baden
86356	Lechtal
98348	Berghaupten

3. Erstellen Sie in einem weiteren Schritt die Tabelle *Kunden* wie unten abgebildet.

Abkürzungen: *KdNr* = Kundenummer,
VNr = Vertreternummer,
RabGruppe = Rabattgruppe

Entwurfsansicht der Tabelle *Kunden*



Datenblattansicht (Ausschnitt) der Tabelle *Kunden*

KdNr	KdName	Straße	PLZ	LetzterKauf	VNr	RabGruppe
K1000	Freizeit AG	Kaiserstraße 6	76437	15.01.2014	V2000	1
K1010	Weinelt Hans	Flotoweg 5	76437	01.02.2014	V2000	1
K1020	Fraas Helga	Murgstraße 5	76530	21.01.2014	V2000	2
K1030	Intersport GmbH	Murgweg 2	12345	10.01.2014	V2000	2
K1040	Campinghaus Fun KG	Bergstraße 4	32451	15.01.2014	V1000	2
K1050	Maier & Co	Rheinstraße 4	43532	12.01.2014	V1000	2
K1060	Mayer KG	Mainstraße 4	12345	12.01.2014	V1000	1
K1070	Meier OHG	Eisenstraße 8	86356	21.01.2014	V3000	1
K1080	Froh OHG	Ringstraße 1	43654	14.01.2014	V3000	1
K1090	Zanupio Franz	Am Bach 3	45387	12.01.2014	V3000	3
K1100	Sportring OHG	Haller Straße 1	42001	15.01.2014	V3000	1
K1110	Strunk KG	Hauptstraße 55	41224	29.01.2014	V3000	2

usw.

1.5.3 Felddatentypen und Feldeigenschaften sorgen für Ordnung

Bisher wurde noch nicht begründet, weshalb der Datenbankanwender überhaupt Felddatentypen und Feldeigenschaften zu definieren hat.

Stellen Sie sich vor, Sie übertragen dem Computer die Aufgabe, folgende Addition vorzunehmen:

007 + 001 = ?
Der Computer als „nichtdenkende" Maschine kommt zu dem Ergebnis 008. Der PC hat keine Chance zu erkennen, dass 007 keine Zahl, sondern eine Bezeichnung für einen Geheimagenten ist. Die Addition zweier Geheimagenten ist jedoch sicher sinnlos, weshalb der Computer angewiesen werden muss, solche Sinnlosigkeiten zu unterlassen. Genau hierzu dienen Datentypen.

Haben Sie den Felddatentyp *Text* gewählt, passiert oben beschriebenes Missverständnis nicht. **Mit Texten kann nicht gerechnet werden!**

Sind Postleitzahlen wirklich Zahlen?

Die PLZ besteht aus 5 Stellen ohne Nachkommastellen. Somit läge es nahe, den Datentyp *Zahl* mit der Feldgröße *Integer* (ganze Zahl) zu verwenden. Diese Feldgröße (vgl. folgende Übersicht) kann zwar fünfstellig vereinbart werden, sie reicht aber nur bis zur Zahl 32 767, sodass überlegt werden muss, die Feldgröße *Long Integer* (große ganze Zahl) zu vereinbaren.

Aber! Wie speichert man die Postleitzahlen der neuen Bundesländer?
Sie beginnen mit einer 0, aber (leider) arbeitet die Feldgröße *Long Integer* mit der Vornullen-Unterdrückung.

Ist es wirklich nötig, Postleitzahlen mit dem Datentyp *Zahl* zu vereinbaren?

Nötig wäre dies nur, wenn mit Postleitzahlen Berechnungen durchgeführt werden sollen, was in hier Fall sicher nicht zutrifft.

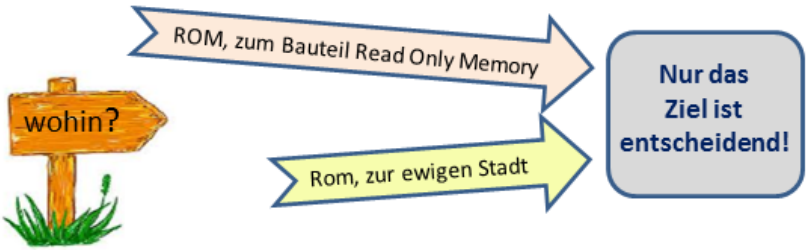
Unterbrechnung der Leseprobe

2 Objektorientierte Systementwicklung

2.1 Von der strukturierten Programmierung zum objektorientierten Softwaredesign

Vorbemerkung zum methodischen Vorgehen.

Viele Wege führen nach Rom (ROM)



Der Autor dieses Buches hat sich entschieden, in die Thematik der „echten“ Objektorientierung erst dann einzusteigen, wenn der Anwender die Grundlagen der strukturierten Programmierung beherrscht.

Wohl wissend, dass es andere methodische Ansätze gibt (Viele Wege führen nach Rom), wird in diesem Kapitel so vorgegangen, dass grundlegende **Programmstrukturen** (lineare Strukturen, Auswahlanweisungen und Wiederholungsanweisungen) mit **Jave/Eclipse**[7] erarbeitet werden, und erst in einem weiteren Schritt Objekte aus deren Klassen „entstehen“.

So erklärt sich die Überschrift dieses Abschnitts, in dem versucht wird, softwareübergreifende Programmstrukturen zu erarbeiten, objektorientiert zu programmieren und erst in einem weiteren Schritt mit der grafischen Benutzeroberfläche von Eclipse zum Softwaredesign überzugehen.

Vorab eine wichtige Adresse die Ihnen vor allem dann hilft, wenn Sie am PC sitzen und sich vor Verzweiflung die Haare raufen. Es ist eine lexikalische Auflistung, zusammengestellt vom Max-Planck-Institut.

www.mpi-inf.mpg.de/departments/d5//teaching/ss05/is05/java/GoToJava2.2/html/index_i.html#ixb100574

2.1.1 Von der Problemstellung zum Algorithmus

Bereits in Band 1 dieser Buchreihe wurde ein Computer als ein technisches Gerät bezeichnet, das Informationen automatisch verarbeiten und speichern kann. Im Unterschied z. B. zu einem normalen Automaten, wie einem Getränkeautomaten, welcher nur

[7] Eclipse ist nicht nur ein Softwareprodukt. Es war in den 30er Jahren ein PKW-Statussymbol von Peugeot. Weltweit ist noch die Existenz von 34 dieser Peugeot-401-Eclipse-Modellen bekannt.

festgelegte Aktionen ausführt, kann dem Computer die Vorschrift nach der er arbeiten soll, jeweils neu vorgegeben werden.

Eine solche Handlungsvorschrift heißt **Algorithmus.**

Bevor aber überhaupt ein Computer zum Einsatz kommt, beginnt die Arbeit mit einer **Problemstellung**. Hier wird in allgemeinen Formulierungen beschrieben, was ein Computer überhaupt tun soll. Danach wird in einem weiteren Schritt die Problemstellung in einer **Problemanalyse** konkretisiert. Erst danach erfolgt die eigentliche Arbeit am Gerät, nämlich die Übertragung der Ergebnisse der Problemanaylse in ein maschinenverständliches Konzept, die **Programmierung**, moderner ausgedrückt: **Softwaredesign.**

Zu Beginn ein Ausflug in die Geschichte.

Wussten Sie schon, dass das Wort **Algorithmus** auf den persisch-arabischen Mathematiker und Astronom Muhammad Ibn Musa Al-Hwârizmî nach seinem Werk „Kitab al jabr w.almuqabala“ („Regeln der Wiedereinsetzung und Reduktion“) um 825 n. Chr. zurückzuführen ist. Das Werk – es ist im Original leider nicht mehr vorhanden – behandelt algebraische Gleichungen, das indische Zahlensystem und das Rechnen in diesem. Eine lateinische Übersetzung aus dem 12. Jahrhundert „Algorithmi de numero Indorum“ existiert noch. [8]

Seit Jahrhunderten versteht man unter einem Algorithmus eine präzise, d. h. in einer festgelegten Sprache abgefasste, endliche Beschreibung eines allgemeinen Verfahrens unter Verwendung ausführbarer Verarbeitungsschritte zur Lösung einer Aufgabe.

Wie setzt man mit Hilfe eines Computers einen Algorithmus um?

Ein Computer ist mittels eines Rechenwerks in der Lage, einfache Operationen in hoher Geschwindigkeit auszuführen. Die Reihenfolge der auszuführenden Operationen wird mittels des Steuerwerks organisiert, wobei folgende beiden Voraussetzungen erfüllt sein müssen:

- Das zu lösende Problem muss durch Arbeitsanweisungen (Operationen) beschreibbar sein.
- Die Reihenfolge der auszuführenden Operationen muss bekannt sein.

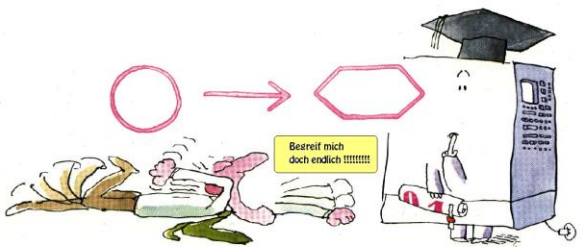
Merke

Alle Rechenanlagen erfordern zu ihrer Handhabung ein System von Anweisungen, deren Anwendung festen Regeln unterliegt. Eine solche Folge von präzise formulierten Anweisungen, die einen eindeutigen und vollständigen Lösungsweg aufzeigen, bezeichnet man als **Algorithmus**. Den in eine Programmiersprache übersetzten Algorithmus nennt man ein **Programm.** Ein Programm besteht somit aus einer Folge von logisch angeordneten Programmzeilen zur Lösung eines konkreten Problems.

[8] Wenn Sie mehr erfahren wollen, hier eine interessante Adresse: www.kk.s.bw.schule.de/mathge/alhwariz.htm

Algorithmen werden mittels Programmiersprachen auf Computer übertragen. Die Darstellung eines Algorithmus´ kann u. a. in umgangssprachlicher Form

- im **Pseudo-Code**,
- in grafischer Form (**Struktogramm** oder **Programmablaufplan** [PAP]) und/oder
- in einer **Programmiersprache** erfolgen.



2.1.2 Vom Algorithmus zum Programm

Tipp

Nur echten Genies ist es vorbehalten eine gegebene Problemstellung sofort in eine Programmiersprache zu übertragen. Allen anderen wird empfohlen, die Lösung in Teilschritten vorzunehmen, wobei sich die folgenden Schritte der Problemanalyse bewährt haben.

1. Schritt: Problemstellung

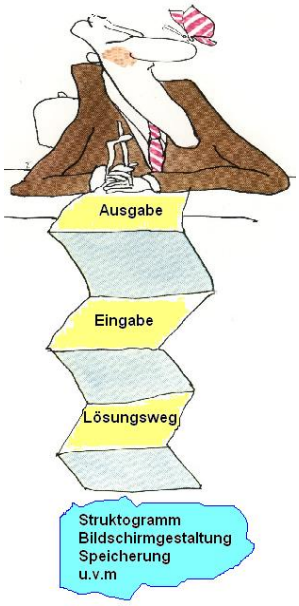
Soll ein Programm entwickelt werden, so muss zunächst eine konkrete Problemstellung vorliegen.

2. Schritt: Problemanalyse

Um eine vorgegebene Problemstellung programmtechnisch lösen zu können, ist es erforderlich, die gewünschten Ergebnisse (Ausgabedaten) und die dazu vorhandenen Ausgangsdaten (Eingagedaten) genau zu analysieren.

Dazu dient die **Problemanaylse**, welche u. a. auf folgende Fragen eine Antwort geben soll.

- Was soll ausgegeben werden?
  Zusammenstellung der **Ausgabedaten**.
- Welche Daten müssen eingegeben werden?
  Zusammenstellung der **Eingabedaten**.
- Wie sollen die Eingabedaten verarbeitet werden?
  Eindeutige Darstellung des gewählten **Lösungswegs**.

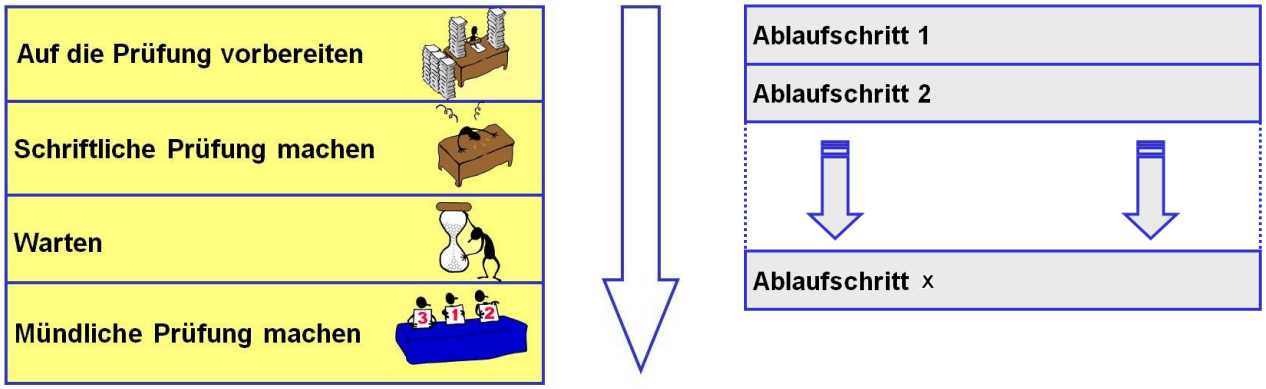


3. Schritt: Grafische Darstellung im Struktogramm

Merke

Ein Struktogramm zeigt in grafischer Form die logische Reihenfolge der einzelnen Arbeitsschritte. Es ist somit ein nach DIN 66261 genormtes Hilfsmittel zur Planung, Entwicklung und Dokumentation von Programmstrukturen.

Ein linearer Verlauf wird z. B. durch eine untereinander stehende Folge von Vierecken dargestellt wird nach jeder Anweisung mit einer horizontalen Linie abgeschlossen.



4. Schritt: Codierung, d. h. Übertragung in eine Programmiersprache.

In diesem Buch wird die objektorientierte Programmiersprache **Java** unter der Oberfläche von **Eclipse** eingesetzt.

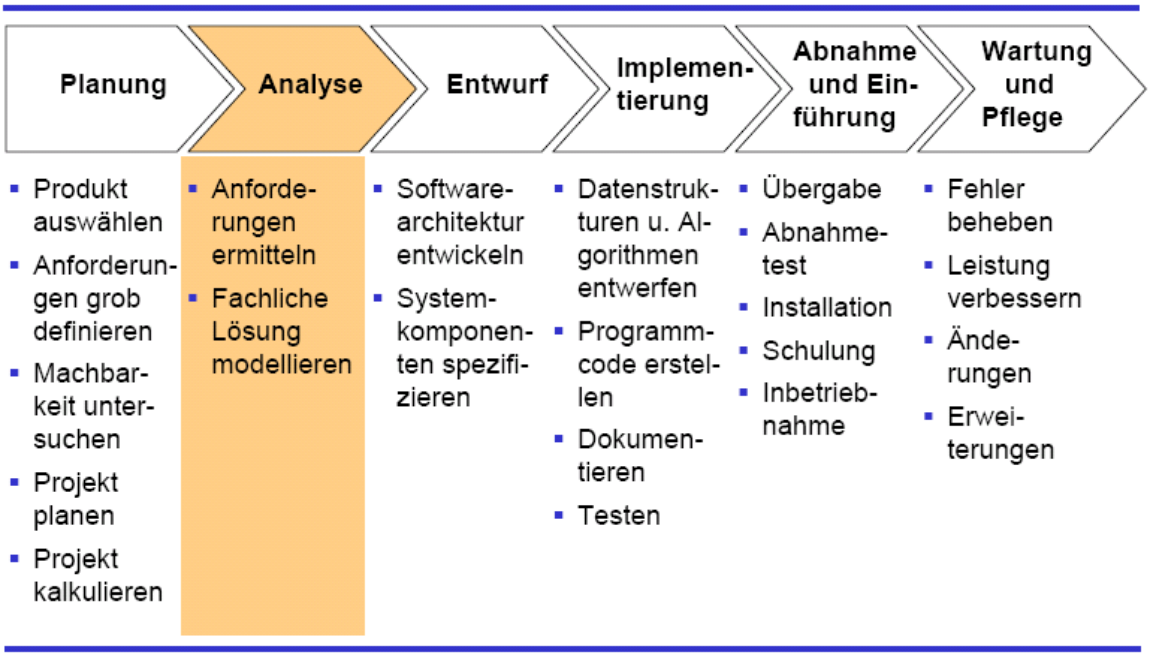
5. Schritt: Programmtest

Der so erstellte Programmcode wird vom Computer und/oder in Form eines Schreibtischtestes auf syntaktische (formale) und/oder semantische (logische) Fehler getestet.

6. Schritt: Dokumentation

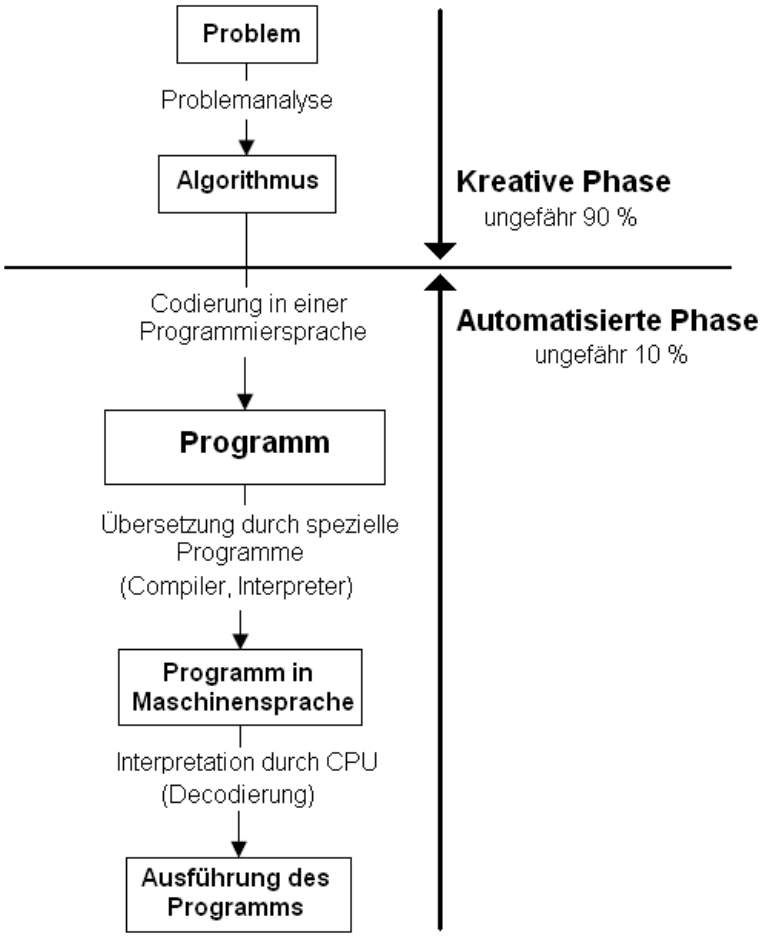
Während der gesamten Softwareentwicklung wird jede einzelne Entwicklungsstufe dokumentiert.

2.1.3 Phasen bei der Erstellung von Softwareprojekten



2.1.4 Vom Programmcode zur Maschinensprache

Ablaufsteuerung der beschriebenen Arbeitsschritte



Folgender Klassiker der Programmierung soll verdeutlichen, wie man von einer konkreten Problemstellung zu einem Algorithmus gelangt.

Ein spektakuläres Landgeschäft

PROBLEMSTELLUNG/Sachverhalt

Im Jahre 1627 verkauften die Indianer die Insel Manhattan für den Spottpreis von 24,00 $ an einen Niederländer namens Pieter Minnewit. Dieser gründete darauf das Fort Neu-Amsterdam und jeder weiß, was seitdem daraus geworden ist.
Aber einmal abgesehen davon, dass die 24,00 $ für Minnewit und seine weißen Nachkommen ausgezeichnet investiert waren, hätten auch die Indianer eine gute Verzinsung erzielt, wenn sie die 24,00 $ auf einem Sparkonto angelegt hätten (was 1627 leider nicht möglich war). Unterstellen wir einmal, die Indianer hätten das Geld zu einem Zinssatz von nur 2 % angelegt und die Zinsen und Zinseszinsen immer liegen gelassen. Auf wie viel Dollar wäre das Kapital bis heute angewachsen?

Ausgabedaten: Endkapital im aktuellen Jahr.

Eingabedaten: Anfangskapital, Zinssatz, Anfangsjahr (hier 1627), aktuelles Jahr

Verarbeitung: Rechnerisch gesehen, erreicht ein Kapital *K* in *n* Jahren - wenn die Zinsen alljährlich zum Kapital hinzuaddiert werden durch Zinseszinsen zu *p %* - den Endwert $Kn = Kq^n$. Dabei ist der jährliche Diskontfaktor $q = 1 + p/100$. Die Formel unserer Aufgabe lautet daher für das Jahr 2009:

$K_{(2014-1627)} = 24 \times 1{,}02^{382}$

Bitte schätzen Sie, wie viele Dollar die Indianer für ihren Kapitaleinsatz von 24 $ vor 382 Jahren erhalten hätten. Wetten auch heute bleibt den roten Brüdern als Nachfahren nichts anderes übrig, als sich über die entgangenen Möglichkeiten grün und blau zu ärgern.

2.2 Grundlegende Eigenschaften von Java

Unterbrechnung der Leseprobe

1.4 Erstellung linearer Programmstrukturen

Eine lineare Folge ist dadurch gekennzeichnet, dass die Anweisungen immer in eine Richtung, nämlich von oben nach unten ausgeführt werden. Die oberste Anweisung wird als erste, die unterste Anweisung als letzte bearbeitet. Dabei wird jede Anweisung genau einmal ausgeführt.

1.4.1 Variablen, Konstanten, Datentypen

Variablen bezeichnen Speicherbereiche im Hauptspeicher, in denen ein Programm Werte ablegen kann. Für den Compiler ist eine Variable nichts anderes als ein Verweis auf den Anfang eines Speicherbereichs. Da nun sehr unterschiedliche Werte (Zahlen, Buchstaben u. v. m.) unter einer Variablen abgespeichert werden können, muss der Compiler die Größe des zu reservierenden Speicherbereiches kennen. Deshalb wird bei der Deklaration nicht nur der Name der Variablen, sondern auch deren **Datentyp** definiert. Der Datentyp gibt dem Compiler an, wie der Speicherinhalt der Variablen zu interpretieren ist.

Wetten, dass Sie – auch wenn sie noch nie mit einem Computer gearbeitet haben – bereits mit Variablen zu tun hatten!

Sicher wissen Sie, wie groß ihr Arbeits- oder Schlafzimmer ist. Sie können dies schnell nachprüfen, indem Sie die Quadratmeterzahl nach der Formel

```
Fläche = Länge x Breite
```

ermitteln.
Die Variablen zur Berechnung der Zimmergröße sind somit `Länge` und `Breite`, für die Sie jeweils die konkreten Werte einsetzen.

Zur Berechnung der Größe Ihres Zimmers wurden Variablen verwendet, die ausschließlich Zahlenwerte beinhalten dürfen. Sollten Sie aus Versehen keine Zahl, sondern einen Text eingeben, muss das System „rebellieren“. Somit ist nicht nur der Name einer Variablen dem System mitzuteilen, sondern auch der jeweilige **Datentyp**.

Manchmal hilft ein Vergleich.

Stellen Sie sich eine Variable ruhig wie einen Behälter vor, in den aber immer nur ähnliche Gegenstände abgelegt werden. Wenn Sie Ihr Auto waschen, verwenden Sie vielleicht mehrere Wassereimer mit verschiedenen Reinigungsflüssigkeiten. Genau so verhält es sich mit Variablen. Eine Variable nimmt vielleicht nur ganze Zahlen auf, eine andere nur Zahlen mit Nachkommastellen, wieder eine andere nur Texte, usw.[9]

Allgemein gilt

- Variablen müssen immer vor dem ersten Aufruf deklariert werden.
- Bei der Deklaration wird der Variablen einen Datentyp zugewiesen.
- Es können (durch Komma getrennt) mehrere Variablen gleichzeitig deklariert werden.

2.4.2 Variablen und deren Datentypen[10]

Den Namen der Variablen können Sie selbst festlegen. Erlaubt ist fast alles, außer:

1. Er darf kein reserviertes Java-Wort sein.
2. Er darf nur aus alphanumerischen Zeichen und dem Zeichen _ (Unterstrich) bestehen.
3. Er darf nicht mit einer Ziffer beginnen.

[9] Die Idee zu dem Schaubild wurde entnommen: „Programmieren mit Java: Vorlesung, Grundlagen und Beispiele, Institut für Softwaretechnik, Universität Koblenz/Landau“.

[10] Weitere Informationen finden Sie im Anhang.

Allgemein gilt

`variablenTyp variablenName;`	Groß- und Kleinschreibung und das Setzen des Semikolons beachten
z. B.: `double kreisUmfang;`	// Deklaration der Variablen
`kreisUmfang = 3;`	// Wertzuweisung an die Variable

Hier ist `double` der **Datentyp** zur Vereinbarung einer Gleitkommazahl.

Achtung

Der Datentyp wird bei der Deklaration dem Variablennamen vorangestellt. Er bestimmt die Wertemenge und die zulässigen Operationen.

2.4.2.1 Elementare Datentypen: Ganzzahlen

Es existieren vier Ganzzahltypen:

byte	8-Bit-Zahl	-2^7 .. 2^7-1 (-128 bis 127)
short	16-Bit-Zahl	-2^{15} .. $2^{15}-1$ (-32 768 ... 32 767)
int	32-Bit-Zahl	-2^{31} ..$2^{31}-1$ (-2 147 483 648 .. 2 147 483 647)
long	64-Bit-Zahl	-2^{63} .. $2^{63}-1$

2.4.2.2 Elementare Datentypen: Fließkommazahlen

Dies sind Zahlen mit gebrochenem Anteil (reelle Zahlen). Mit den Datentypen *float* und *double* können reelle Zahlen repräsentiert werden. Sie sind vorzeichenbehaftet und unterscheiden sich nur in ihrer Größe und Genauigkeit.

float	4 Byte (32-Bit-Zahl)	10^{-46} bis 10^{38}
double	8 Byte (64-Bit-Zahl)	10^{-324} bis 10^{308}

2.4.2.3 Elementare Datentypen: Zeichen und Wahrheitswerte

Bei **char** handelt es sich um einen Datentyp, der beliebige Zeichen aufnehmen kann. Entsprechende Zeichenkonstanten sind dafür in einfache Anführungszeichen zu setzen, z. B. `'c'`. Dieser Typ erlaubt die Repräsentation von Zeichen im sogenannten **Unicode-Zeichensatz**, der u. a. auch chinesische und arabische Schriftzeichen unterstützt. Unicode ist ein internationaler Standard, in dem langfristig für jedes sinntragende Schriftzeichen bzw. Textelement aller bekannten Schriftkulturen und Zeichensysteme ein digitaler **Code** festgelegt wird. Ziel ist es, die Verwendung unterschiedlicher und inkompatibler Kodierungen in verschiedenen Ländern oder Kulturkreisen zu beseitigen. Dieser Code wird laufend um Zeichen weiterer Schriftsysteme ergänzt.

Bei *char* gilt:

Pro Variable kann nur ein einziges Zeichen gespeichert werden.

Der Datentyp **boolean** wurde nach George Boole, einem Mathematiker des 19. Jahrhunderts benannt. Dieser Datentyp kennt zwei Zustände, nämlich die Wahrheitswerte „wahr“ und „falsch“ (true bzw. false).

2.4.2.4 Beispiele für Initialisierungen - Wertzuweisungen

`int x`	vereinbart eine Variable *x* des Typs int (integer).
`int a, b`	vereinbart 2 Variablen *a* und *b* des Typs int (integer).
`int x = 100;`	vereinbart die Variable *x* des Typs int (integer) und weist ihr den Anfangswert 100 zu.
`int a = 0, b = 1;`	vereinbart 2 ganzzahlige Variablen a und b und weist die Anfangswerte 0 bzw. 1 den Variablen zu.

`int a, b;`
`a = 3;`
`b = 5;`
	Der Wert einer Variablen wird durch eine Zuweisung (durch den Zuweisungsoperator „=“) verändert. Der Variablen a wird der Wert 3 zugewiesen, der Variablen b der Wert 5.

`int a, b;`
`a = 3;`
`b = a + 5;`
	In der Variable *b* wird die Zahl 8 gespeichert.

`string name = „Maier“;`	Hier wird der Variablen *name* die Zeichenkette *Maier* zugeordnet.
`double d = 1.123;`	vereinbart eine Variable d als Gleitkommazahl und weist den Wert *1.123* zu.
`char c = 'a';`	vereinbart die Variable c als Zeichen aus dem Unicode-Zeichensatz und weist ihr den Wert *a* zu

Merke

Im Zusammenhang mit Variablen hat das Gleichheitszeichen nicht die aus der Mathematik bekannte Funktion der Gleichheit. Es ist vielmehr ein **Wertzuweisungsoperator**. Bei der Wertzuweisung x = 13 wird der aktuelle Wert einer Variablen auf den neuen Wert 13 gesetzt.

Die erste Wertzuweisung zu einer Variablen wird auch als **Initialisierung** bezeichnet.

2.5 Methoden mit Kontrollstrukturen

Merke

Verzweigungen (Auswahlstrukturen) bilden mit den **Schleifen** (Wiederholungsanweisungen) die **Kontrollstrukturen.**

Bei Auswahlstrukturen wird aufgrund einer oder mehrerer vorab gestellter Bedingung(en) der Programmfluss (die Abfolge der Ausführung der Befehle) gesteuert. Im Gegensatz zu Schleifen, die den Programmablauf nach oben zurückführen können, geht der Ablauf bei einer Verzweigung immer über einen von mehreren Wegen weiter nach unten. Synonyme für Verzweigungen sind **Auswahl** und **Selektion**.

Prinzipiell unterscheidet man drei Arten von Auswahlstrukturen:

- **einseitige Auswahl**
- **zweiseitige Auswahl**
- **mehrseitige Auswahl** (Fallunterscheidung oder CASE-Struktur)

2.5.1 Auswahlstrukturen näher analysiert

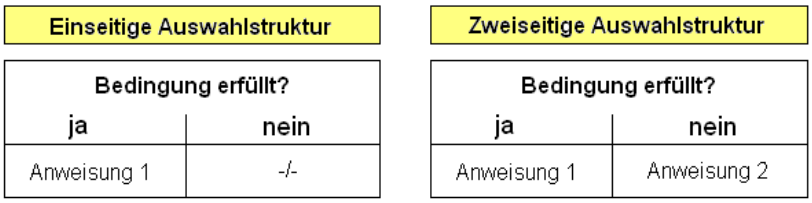
Die bislang behandelten Anwendungen waren dadurch gekennzeichnet, dass die Anweisungen **linear**, d. h. Programmzeile für Programmzeile von oben nach unten ausgeführt wurden. Die oberste Anweisung wurde als erste, die unterste Anweisung als letzte bearbeitet. Dabei wurde jede Anweisung genau einmal ausgeführt. Diese Vorgehensweise würde z. B. zu folgenden Fragestellungen keine befriedigenden Ergebnisse liefern können, da die Anweisungen in Abhängigkeit einer (oder mehrerer) Bedingungen ausgeführt bzw. auch nicht ausgeführt werden.

WENN Aktienkurs <= 500,00 €	**DANN** verkaufe	**SONST** behalte das Papier
WENN Umsatz >= 1.000,00 €	**DANN** Provision = 10 %	**SONST** keine Zahlung
WENN Bestellmenge > 10 Stück	**DANN** gewähre 5 % Rabatt	**SONST** 3 %

Merke

Eine Auswahl wird getroffen, indem eine Frage mit *Ja* oder *Nein* beantwortet wird. Führt nur **ein** Fall zu einer Aktivität, so handelt es sich um eine **einfache** bzw. **einseitige Auswahl.** Liegen zwei oder mehrere Alternativen vor, spricht man von einer **zweiseitigen- bzw. mehrseitigen Auswahlstruktur.**

Darstellungen in Struktogrammen[11]



Allgemein lässt sich die Syntax einer **if-Anweisung** wie folgt darstellen:

```
if(Bedingung)
{
     // Code, der ausgeführt wird,
     // wenn die Bedingung erfüllt ist.
}
else
{
     // Code, der ausgeführt wird,
     // wenn die Bedingung nicht erfüllt ist.
}
```

Wichtig sind hier wieder einmal die Klammern.

Nach dem Schlüsselwort ***if*** muss zunächst in runden Klammern die Bedingung formuliert werden. Anschließend wird mit Hilfe von geschweiften Klammern der Code-Block definiert. Danach folgt das Schlüsselwort ***else***, dessen Code-Block ebenfalls in geschweiften Klammern einzuschließen ist.

PROBLEMSTELLUNG

Mithilfe einer Java-Applikation soll – nach Eingabe des Lebensalters – entschieden werden, ob der Anwender über 18 Jahre alt ist und somit Auto fahren darf, oder ob er unter 18 Jahre ist und somit nicht selbst per PKW am Straßenverkehr teilnehmen kann.
Java Applikationsname = `Altersvergleich_1`

Problemanalyse

Ausgabedaten (über Konsole)

entweder: „Sie dürfen noch nicht Auto fahren!“
oder: „Klar: Sie dürfen Auto fahren!"

Eingabedaten

[11] Da einseitige Auswahlstrukturen äußert selten vorkommen, werden sie in diesem Buch nicht weiter behandelt.

Lebensalter über die Tastatur

Verarbeitung

WENN Lebensalter < 18
Textausgabe: „Sie dürfen noch nicht Auto fahren!“
SONST

Textausgabe: "Klar, Sie dürfen Auto fahren!"

Darstellung im Struktogramm (nur Auswahlstruktur)

Lebensalter < 18	
ja	nein
Textausgabe: Sie dürfen noch nicht Auto fahren!	**Textausgabe:** Endlich: Sie dürfen Auto fahren!

Codierung

```
package packageAltersvergleich_1;
import java.io.*;
public class Altersvergleich_1
{
public static void main(String[] args)throws IOException
   {
      BufferedReader eingabe;
      int alter;
      eingabe = new BufferedReader(new InputStreamReader(System.in));
      System.out.print("Wie alt sind Sie? ");
      Alter = Integer.parseInt(eingabe.readLine());
      //Beginn der Auswahlstruktur
      if (alter < 18)
      {
      System.out.println ("Sie dürfen noch nicht Auto fahren!");
      }
      else
      {
      System.out.println ("Klar: Sie dürfen Auto fahren!");
      }
  }
}
```

Erklärungen

`if (alter < 18)`	Zuerst wird der Ausdruck (alter < 18) ausgewertet. Falls *true* wird Anweisung 1 ausgeführt, falls *false* wird Anweisung 1 übergangen und Anweisung 2 ausgeführt.
`else`	Der else-Teil einer if-Anweisung kann fehlen, was aber selten vorkommt.

Unterbrechnung der Leseprobe

2.7 Objektorientierte Analyse und objektorientiertes Design

„Wenn eine Idee nicht zuerst absurd erscheint, taugt sie nichts."

Albert Einstein

2.7.1 Objektorientierung im Überblick

Will ein Bauherr ein Haus bauen, fängt er nicht an sofort die Baugrube auszuheben, sondern beauftragt einen Architekten Pläne und Modelle, Kostenvoranschlägen u. v. m.

zu unterbreiten. Aber auch der Architekt erfindet das berühmte Rad nicht neu, sondern greift auf bestehende Modelle, Pläne, Schablonen zurück.

In der Sprache der objektorientierten Programmierung (im Folgenden als OOP abgekürzt) entspricht die Schablone einer **Klasse** und die gebauten Häuser den **Objekten**. Während es mehrere Häuser geben kann, die mit Hilfe des gleichen Plans gebaut werden, so kann es mehrere Objekte vom Typ einer Klasse geben. So wie ein Plan lediglich die Beschreibung eines Hauses ist, ist die Klasse lediglich eine Beschreibung eines Objekts. Abhängig von der jeweils verwendeten Klasse besitzen Objekte unterschiedliche Eigenschaften und Fähigkeiten. Je nachdem, was Sie für Eigenschaften und Fähigkeiten benötigen, müssen Sie daher beim Erstellen eines Objekts auf die richtige Klasse zugreifen.

Die Entwicklungsarbeit im Rahmen einer OOP besteht also zum Teil darin herauszufinden, welche Klasse für eine bestimmte Funktionalität benötigt wird. Steht eine Klasse nicht zur Verfügung, müssen Sie diese unter Umständen selber programmieren.

Bitte lassen sie sich durch folgendes Zitat nicht entmutigen.

"When I wrote the software only God and I knew how the software worked.
Now a year later only God knows how it works." [12]

Hier die wichtigsten Grundsätze

1. Jedes Objekt gehört zu einer **Objektklasse.**

   Das Erstellen von Objekten geschieht immer in zwei Schritten: Zuerst benötigen Sie eine Klasse, die Sie entweder einer Klassenhierarchie entnehmen oder selber programmieren. Dann erstellen Sie ein Objekt vom Typ dieser Klasse. Somit versteht man unter einer Klasse **die Zusammenfassung von Objekten gleicher Struktur und gleichen Verhaltens**. Eine Klasse kann als Schablone gesehen werden, die beschreibt, wie Objekte aufgebaut sind, wie man Objekte erzeugt, initialisiert und zerstört, mit welchen **Methoden** Objekte bearbeitet werden können und welche Beziehungen zu anderen Klassen existieren.

   Unter Wikipedia finden Sie folgende Definition:

   „**Klasse** ist in der **Objektorientierung** ein abstrakter Oberbegriff für die Beschreibung der gemeinsamen Struktur und des gemeinsamen Verhaltens von **Objekten** (*Klassifizierung*). Sie dient dazu, Objekte zu abstrahieren. Im Zusammenspiel mit anderen Klassen ermöglichen sie die Modellierung eines abgegrenzten Systems in der **objektorientierten Analyse** und dem **objektorientierten Design**."

   Ein konkret existierendes Objekt heißt auch **Instanz** der Klasse. David Flanagan[13] drückt dies so aus: ***„An Object is an Instance of a Class."*** In Java wird eine Klassendefinition durch das Schlüsselwort `class` eingeleitet, doch davon später mehr.

[12] Quelle: http://www.wi.hs-wismar.de/~laemmel/Lehre/JAVA/EiP0-Einfuehrung.pdf

[13] David Flanagan ist ein bedeutender Buchautor zu den Themen der Objektorientierung.

2. Die Struktur einer Klasse bilden die **Attribute** (*Eigenschaften*).

   Bereits im ersten Kapitel dieses Buches wurden Tabellenelemente wie NAME, PLZ, ORT … im Zusammenhang mit Datenbanken als **Datenfeldnamen** bezeichnet. In der Sprache der Objektorientierung sind dies **Attribute** (genauer: Attributnamen), wobei gerade hier die Wahl des „richtigen“ Datentpys von Bedeutung ist.

   Beispiele für Datentypen hierzu sind:

   String: Menge aller Zeichenketten
   Integer: Menge aller ganzen Zahlen
   Boolean: Menge der Wahrheitswserte (Wahr / Falsch)
   usw.

Die Attributwerte eines Objekts können sich ändern. Die Attribute selbst stellen unveränderliche Merkmale eines Objekts dar.

3. Verarbeitende Elemente einer Klasse werden als **Methoden** bezeichnet. Sie definieren das Verhalten von Objekten und werden innerhalb einer Klassendefinition angelegt. Somit haben sie Zugriff auf alle Variablen des Objekts.

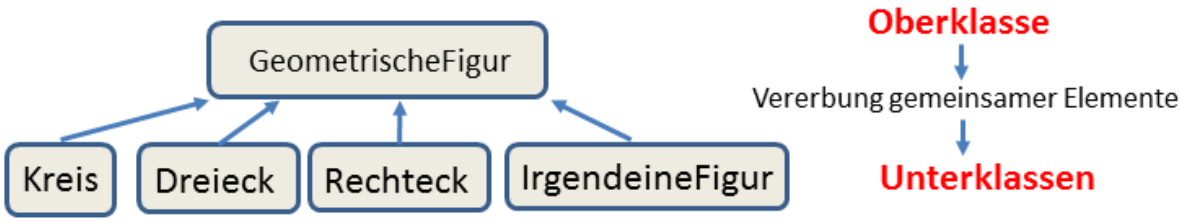
   Methoden sind damit in objektorientierten Sprachen das Mittel, um **Algorithmen** zu formulieren.

   Gegenüberstellung von Attributen (Attributwerte) und Methoden am Beispiel von Mitarbeitern einer Behörde:

Attribute und Attributwerte	**Methoden**
Personalnummer: *8815* Name: *Müller* Vorname: *Eva* Gehalt: *4500,00 €*	einstellen () entlassen () druckeAusweis () erhöheGehalt ()

4. Eine **Klasse** kann durch **Vererbung** aus einer anderen Klasse hervorgehen. Bei der Betrachtung von Klassenhierarchien werden alle Vorgänger einer Klasse als **Oberklassen** (super classes, Superklassen) und alle Nachfolger als **Unterklassen** (sub classes, Subklassen) bezeichnet. Eine Klasse erbt alle Eigenschaften der unmittelbaren Oberklasse und gibt all ihre Eigenschaften an die unmittelbare Unterklasse weiter. Geerbte Eigenschaften werden dabei ebenfalls weitergegeben. In Java besitzt eine Klasse genau eine unmittelbare Oberklasse (mit Ausnahme der Klasse `Object`, die den Kopf jeder Klassenhierarchie in Java darstellt). Ist keine konkrete Oberklasse angegeben, wird standardmäßig `Object` als Oberklasse festgelegt.

   Somit bedeutet Vererbung, dass Klassen ihre Eigenschaften, also Methoden und Datenelemente, auf andere Klassen übertragen.



5. Jedes Objekt arbeitet in sich abgeschlossen und kommuniziert über definierte Schnittstellen mit der Umwelt. Der innere Aufbau eines Objekts bleibt von außen unsichtbar. Durch diese sogenannte **Kapselung** können daher kaum ungewollte Auswirkungen auf andere Programmteile auftreten (Zugriffsschutzmechanismus). Diese Form der **Modularisierung** hilft dabei, Problemlösungen durch Lösung von Teilproblemen zu erzielen, was eine Vereinfachung des Lösungsverfahrens bedeutet.

   In Java geschieht dies mit Hilfe der Schlüsselwörter *private* und *protected*, die die Sichtbarkeit der damit notierten Komponenten einschränken. „Private“ bedeutet *nur für Objekte der gleichen Klasse sichtbar*. „Protected“ bedeutet *nur für Objekte der gleichen Klasse oder von Unterklassen sichtbar.*

Manchmal helfen Vergleiche

Die bereits verwendeten Begriffe haben im Alltagsleben oft andere Bedeutungen, weshalb sie am Beispiel anhand der Klasse „Bauplan einer Wohnanlage“ erklät werden sollen. [14]

Objekt: Wohnanlage am Waldrand

Klasse: Grundriss

Attribute: z. B. Stärke der Mauersteine (30 cm), Heizungssystem (Solar), Farbe des Außenputzes (Gelb)

Methoden: z. B. Stärke der Außenwände um 3 cm vergrößern, Gartenboden um 5 cm höher legen, Kinderzimmer um 1 m² verkleinern, neue Farbe für Aussenputz anbringen

[14] vgl.: www.bauplan-haus.de/images/bild4.jpg

Vererbung: Grundriss soll für Wohnanlage "Am Waldrand" soll auch für die Wohnanlage "Am See" gelten.

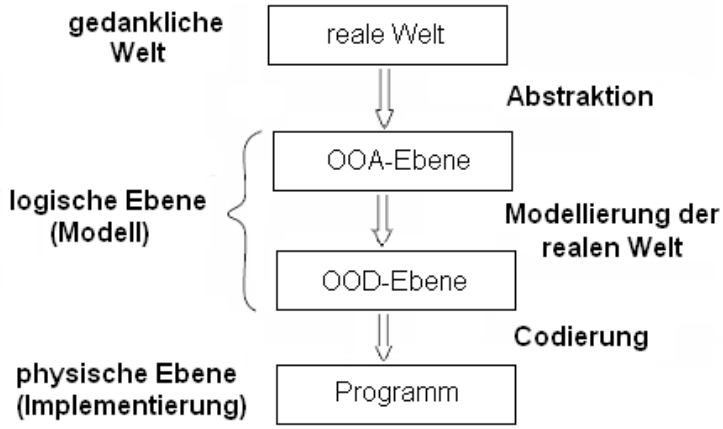
Kapselung: Kostenvoranschlag nicht öffentlich

Wichtiges auf einen Blick[15]

Klassen sind Baupläne für Objekte, vergleichbar mit dem Bauplan für ein Haus. Im wirklichen Leben kann man in Bauplänen nicht leben. Es muss erst eine konkrete Instanz des Plans erzeugt werden (also ein Haus). Erst dadurch entsteht ein **Objekt** mit all den Eigenschaften (Attributen) eines Hauses (Küche, Schlafzimmer, Wohnbereich usw.). Mit einem Bauplan können beliebig viele Häuser gebaut werden. Jedes Haus ist ein **Objekt,** das sich durch den Wert der Attribute von den anderen Häusern unterscheiden kann (unterschiedliche Wandfarben, Bodenbeläge, Fenster usw.). Eine Klasse enthält also Attribute (Eigenschaften), die die einzelnen Objekte charakterisieren. Zusätzlich zu den Attributen werden in Klassen auch **Methoden** (Operationen) deklariert. Während die Attribute die Eigenschaften eines Objektes repräsentieren, stellen die Methoden das Verhalten dar. Die Methoden beschreiben die Aktionen, die ein Objekt ausführen kann bzw. die auf einem Objekt ausgeführt werden können.

2.7.2 Objektorientierte Systementwicklung

2.7.2.1 Von der realen Welt zum Programm



((OO_Systementwicklung_1.TIFF)

Nach der Analyse von Teilen der realen Welt werden die Erkenntnisse in einem von Programmiersprachen unabhängigen Modell analysiert, der Objektorientierten Analyse (OOA).

Die Fragestellung der **OOA** ist das WAS und nicht das WIE.

Stufen der objektorientierten Softwareentwicklung

Unterbrechnung der Leseprobe

[15] Sie finden unter: https://staff.ti.bfh.ch/fup1/Contents/Scripts/OO-Grundlagen_Java.pdf einen guten Vergleich zu der Problematik der Klassenbildung.

2.8 Softwaredesign mit der grafischen Benutzeroberfläche

Kurz und bündig

Dateneingaben und -ausgaben waren bei den bisher erstellten Projekten alles andere als benutzerfreundlich organisiert. Im Zeitalter der bunten Welt der Bildschirmgrafiken ist eine reine Konsolenbenutzung durch den Einsatz von grafischen Objekten zu ersetzen. Solche Objekte wie z. B. Befehlsschaltflächen, Textfelder, Bezeichnungsfelder, Optionsschaltflächen u.v.m. sind in vordefinierten Klassen nach logischen Gruppen (Packages / Klassenbibliotheken) zusammengefasst und können in den Java-Code importiert bzw. eingebunden werden. Häufig verwendete Packages sind das AWT- und das SWING-Package. AWT steht für ***A***bstract ***W***indowing ***T***oolkit, **SWING** ist fester Bestandteil des Java-Development-Kit, weshalb es hier im Einsatz ist. Hierzu ist der VisualEditor bzw. WindowBuilder erforderlich.[16]

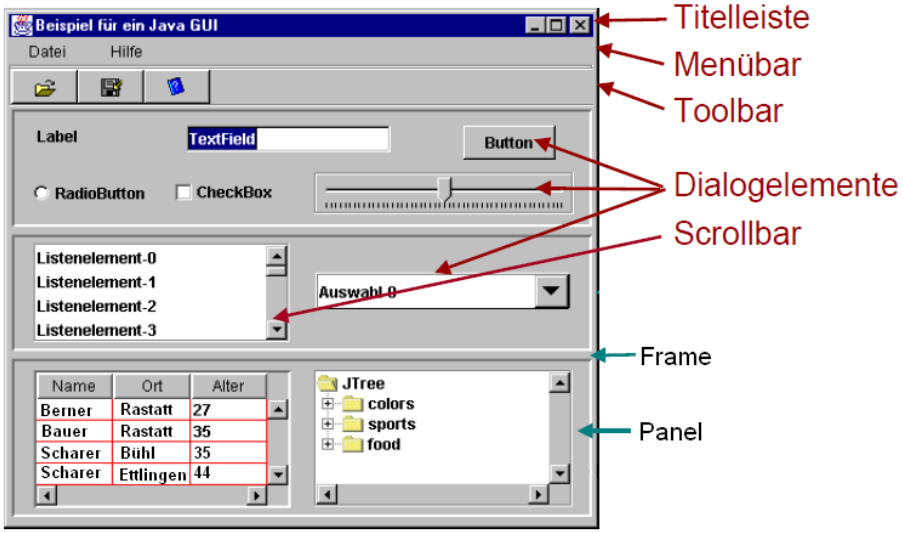
2.8.1 Übersicht über häufig benötige GUI-Komponenten

GUI: **engl.**
„**G**raphical **U**ser **I**nterface“ = grafische Benutzeroberfläche

Merke

Das Swing-Paket dient dazu, unter Java grafische Benutzeroberflächen zu erstellen. Dazu bietet es Komponenten wie **Fenster**, **Dialogboxen**, **Buttons**, **Textfelder** usw., aus denen eine Oberfläche zusammengestellt werden kann. Ferner steuert es die auslösenden Ereignisse, auf welche das System (z. B. durch das Anklicken eines Buttons) zu reagieren hat.

Beispiel wichtiger GUI-Elemente



[16] Hinweise zum Arbeiten mit dem WindowsBuilder entnehmen Sie folgenden Adressen:
http://download.eclipse.org/windowbuilder/WB/release/R201306261200/4.3/
https://www.fbi.h-da.de/fileadmin/personal/b.kreling/dscriptdata/Ena/Praktiku/ErsteSchritteEclipse.pdf

2.8.2 Eine eigene Oberfläche erstellen

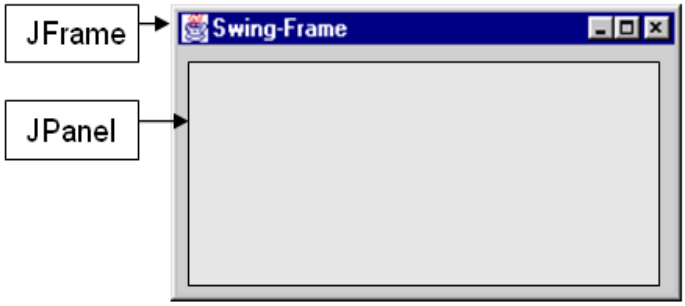
Zur Erzeugung einer grafischen Benutzerschnittstelle bietet Swing eine Hierarchie verschiedener Fenstertypen. Auf der obersten Ebene wird ein Fensterrahmen (*JFrame*) benötigt, der die Grundfläche des GUI-Fensters bestimmt. Es weist alle Grundbestandteile und Funktionalitäten eines Windows-Fensters auf (Titelleiste, Icons, Grössen-Änderungs-Icons …).

2.8.2.1 Erzeugung einer Komponentenfläche (JPanel)

Ein Swing-Fenster besteht meistens aus mindestens zwei Bestandteilen, einem Grundrahmen (JFrame) und einer darin befindlichen Inhaltsfläche, die zur Aufnahme der geplanten Komponenten dient.

Der Rahmen (Frame) dient nicht dazu, die vorgesehenen graphischen Komponenten direkt aufzunehmen. Hierfür sind besondere Inhaltsbereiche (Container) zuständig.

Der wichtigste unter den verschiedenen Containern ist die Zeichenfläche (JPanel).[17]
Das **Panel** dient uns zur Aufnahme der für die Anwendung benötigen GUI-Komponenten (Textfelder, Buttons, Labels …).



Merke

Kontrollelemente sind die Elemente für die Ein-/Ausgabe über eine GUI. Sie müssen zu Containern hinzugefügt werden, damit sie sichtbar werden. Bei den meisten Kontrollelementen wird ein **Event** generiert, wenn sie mit der Maus angeklickt werden oder eine entsprechende Taste aktiviert wird. Grundsätzlich kann zwischen Kontrollelementen für die Eingabe und für die Ausgabe unterschieden werden. Einige Elemente können für beides verwendet werden.

In `main()` wird ein Objekt erzeugt, das von der Swing-Klasse *JFrame* abgeleitet ist. Deshalb muss `javax.swing.*` importiert werden.

[17] Da in diesem Buch keine Erstellung eigener Zeichnungen vorgenommen wird, wird auf eine Beschreibung der umfangreichen graphischen Befehle verzichtet.

2.8.2.2 Datenumwandlungen von Textelementen

Textfelder dienen zur editierbaren Ein- und Ausgabe von Textdaten (Zeichenketten, Strings, Buchstaben, Ziffern).

Daten, die vom Benutzer eingegeben wurden, können mit der Methode …

```
<Komponentenname>.getText();
```

… ausgelesen werden. Diese Methode gibt den Text im Textfeld immer als **String** zurück. Jetzt entsteht das Problem, dass man bekanntlich mit Texten nicht rechnen kann!

> **Hinweis**
>
> Wenn eine Zahleneingabe bearbeitet werden soll, muss diese erst in eine Zahl des entsprechenden Datentyps (integer oder double) konvertiert werden. Dies geschieht durch entsprechende Java-Methoden, die einen String in eine Zahl umwandeln.

Umwandlungsmethoden

```
double d = Double.parseDouble(<Name>.getText());
```

beziehungsweise

```
int i = Integer.parseInt(<Name>.getText());
```

Zahlenformate müssen in das geforderte Stringformat überführt werden. Dazu benutzt man die Tatsache, dass Zahlen, die mit „+“-Zeichen an einen String angehängt werden, automatisch in das Stringformat konvertiert werden.

```
double zahl = 4711;
<Name>.setText("  " + zahl);
```

2.8.2.3 Konventionen zur Namensvergabe

Zur exakten Definition der grafischen Objekte werden folgende Abkürzungen (Vorsilben) dem jeweiligen Objekt mit Kleinbuchstaben vorangestellt:

Label (Bezeichnungsfeld) aus der Klasse **Jlabel** Diese Objekte stehen meist bei Textfeldern und dienen der Erläuterung/Bezeichnung anderer Objekte.	***lbl***Labelname, z. B. ***lbl***Beschriftung
TextField (Textfeld) aus der Klasse **JtextField** In Textfeldern werden meist Daten ein- und ausgegeben.	***tf***TextFeldName, z. B. ***tf***Eingabe
Button (Befehlsschaltflächen) aus der Klasse **Jbutton**	***btn***Name_der_Schaltfläche, z. B. *btn*Rechnen

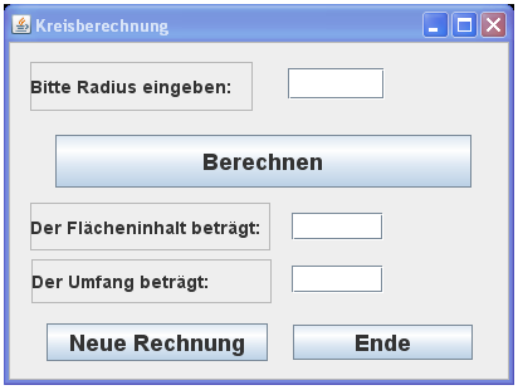
2.8.3 Problemlösungen mit der grafischen Benutzeroberfläche

2.8.3.1 Realisierung linearer Strukturen mit der GUI

PROBLEMSTELLUNG

Bereits im Kapitel „2.4.3.3 Dateneingabe am Beispiel geometrischer Berechnungen“, wurden Fläche und Umfang von Kreisen berechnet.
Genau die selbe Problemstellung soll jetzt mittels Einsatz einer grafischen Benutzeroberfläche realisiert werden.

Gestaltungsvorschlag einer GUI



Und hier die Lösungsschritte im Einzelnen:

Lösung mit dem VisualEditor

- ► File / New
- ► Java Projekt
- ► Projektname = ***Kreis_2***
- ► File / New
- ► Visual Class = ***FrmKreis_2***
- ► Paketname: ***packageKreis_2***

Lösung mit dem WindowBuilder

wie links, aber

…

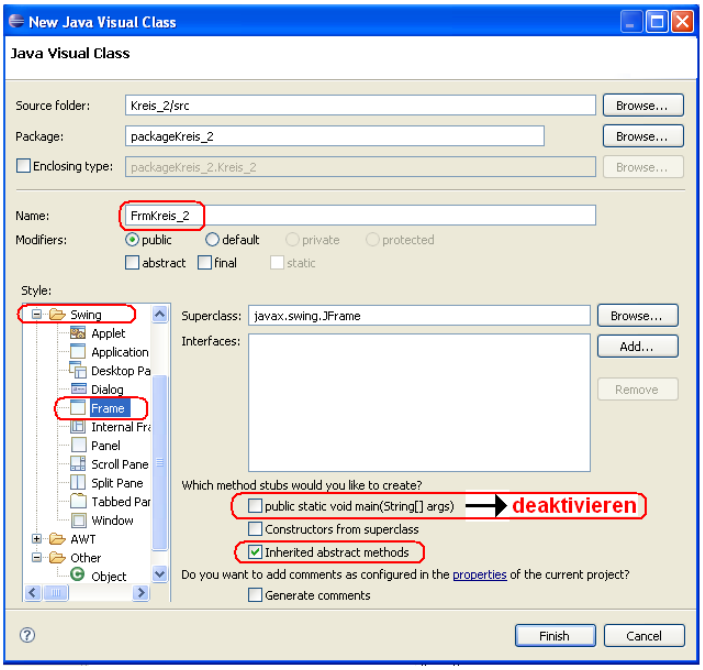
…

- ► File / New / other
- ► WindowBuilder / Swing Designer
- ► jFrame ***(FrmKreis_2)***
- ► Registerkarte ***Design***

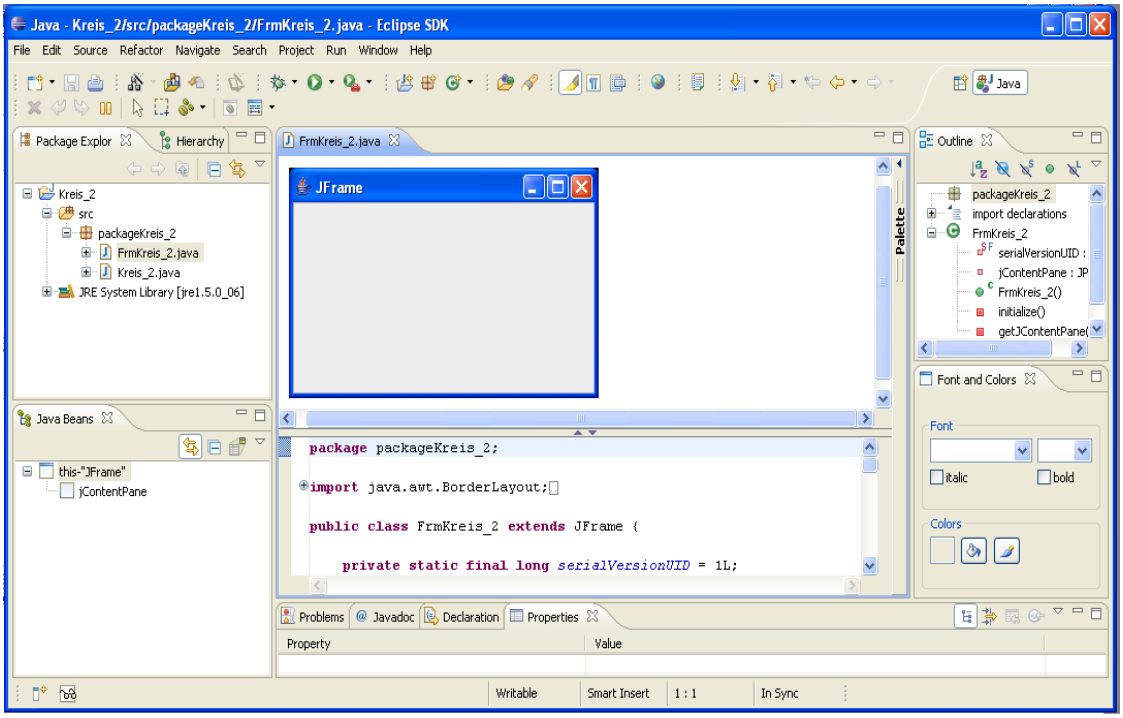
Hinweis

Deaktivieren Sie die Option *public static void(String[]args)*.

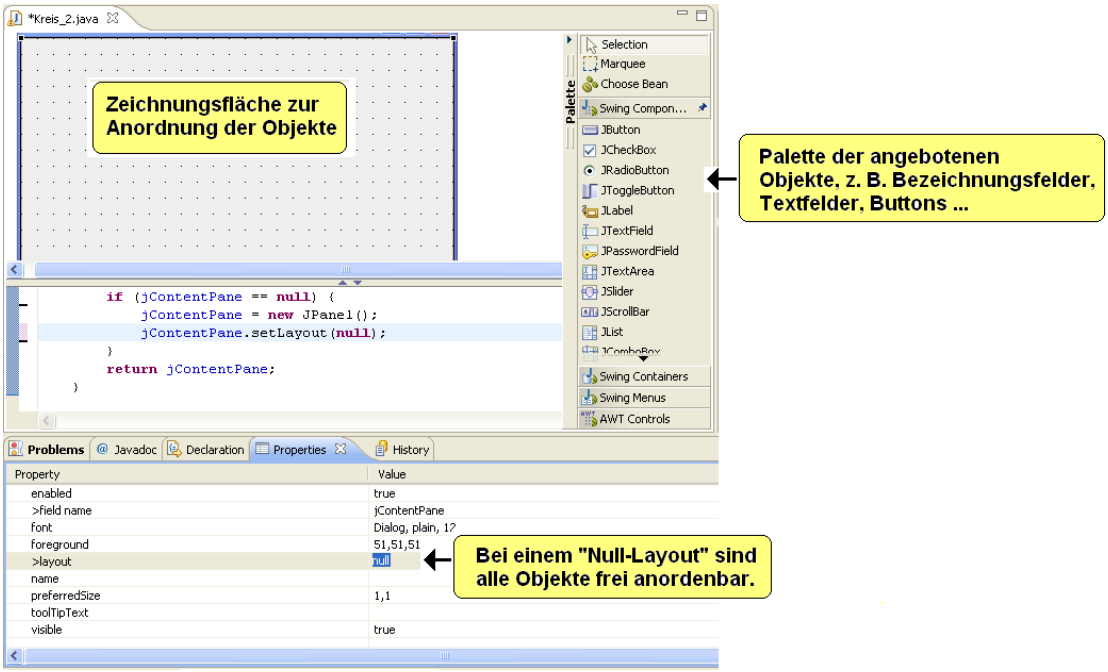
- Im Fenster ***Style*** den Ordner ***Swing*** und die Anwendung ***Frame*** aktivieren. Hierdurch können die grafischen Komponenten des Package SWING übernommen werden.
- Schaltfläche ***Finish*** anklicken.



Im oberen Teil des Bildschirms erscheint jetzt das *frame (JFrame),* also das Formular zur Erstellung der grafischen Benutzeroberfläche mit dem im unteren Teil des Bildschirms erscheinenden Java-Quellcode.

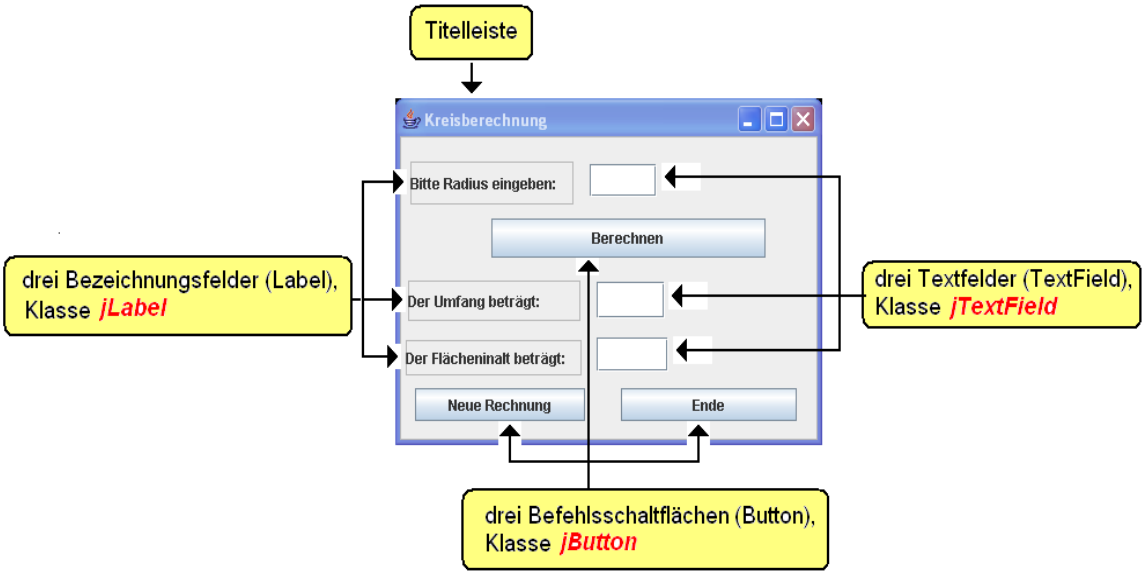


Damit alle Objekte frei angeordnet werden können, sollten Sie im Fenster ***Java-Beans*** die Komponente *jContentPane* anklicken und im Eigenschaftsfenster für “Layout“ (BorderLayout) den Wert *null* auswählen. Die Eigenschaft beim WindowBuilder lautet: ***GroupLayout.***



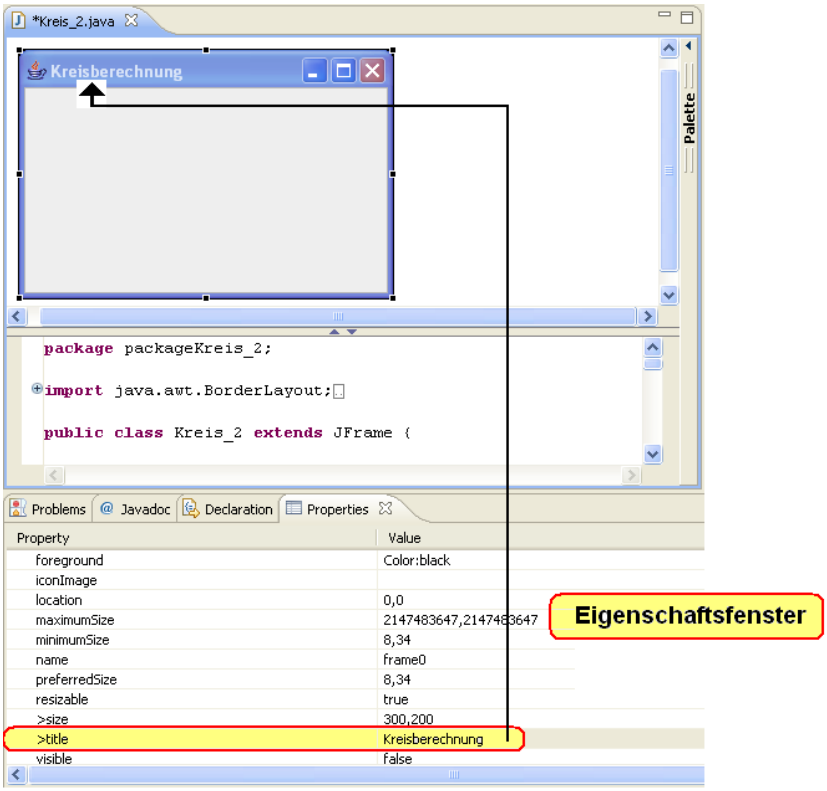
Nun können die Objekte der GUI mit Drag & Drop aus der Palette in die Zeichnungsfläche gezogen und deren Eigenschaften festgelegt werden.

Beachten Sie folgende Bildschirmmaske:



Lösungsschritt 1: Beschriftung der Titelleiste

- ► Mausklick auf der Titelleiste *(JFrame)*.
- ► Im Eigenschaftsfenster die Eigenschaft ändern, z. B. Eingabe des Values *Kreisberechnung*.



Ende der Leseprobe