Skriptum gemäß schulinternem Lehrplan

Grundkurs Informatik – EF

Stand: 29. September 2016

Inhaltsverzeichnis

Vorwort

Informatik hat – als Schulfach – in Nordrhein-Westfalen eine lange Tradition – seit 1969 wird Informatik in der Oberstufe unterrichtet. Weil es immer wieder in der Umsetzung der Arbeitsergebnisse für die Informatiksysteme Änderungen gibt, hat sich gezeigt, dass Materialien, die erstellt werden, oft nach kurzer Zeit nur noch teilweise genutzt werden können.

Mit diesem Skriptum legen wir ein Dokument für unsere Schülerinnen und Schüler vor, das es ermöglichen soll, den Unterricht vor- und nachzubereiten.

Die Arbeitsmaterialien, die verwendet werden, wurden in den vergangenen Jahren von Informatikreferendarinnen und Informatikreferendaren der Fachseminare Informatik an den Zentren für schulpraktische Lehrerausbildung (ZfsL) Hamm, Arnsberg und Solingen entwickelt und im Unterricht erprobt.

Die Materialien stehen unter einer freien Lizenz ( – Erläuterung siehe unten) und sind zum größten Teil über die Webseite `http://ddi.uni-wuppertal.de/material/` öffentlich zugänglich (vgl. (Pieper und Müller 2014)).

Vorbereitung

Was brauchen Sie?

In unserem Unterricht werden Sie schriftliche Notizen anfertigen – Sie erhalten Informationsblätter und Arbeitsblätter. Für den Fließtext, den Sie schreiben, benötigen Sie regelmäßig Kugelschreiber oder Füller. Ihre Aufzeichnungen müssen Sie auf kariertem DIN-A4-Papier vornehmen, da häufig auch kleine Skizzen anzufertigen sind. Dazu benötigen Sie zwingend einen angespitzten Bleistift (HB) und zwei farbige Stifte (grün und rot). Damit Skizzen vernünftig aussehen, benötigen Sie ein Lineal und ein Geodreieck. Sammeln Sie Ihre Unterlagen in einem Hefter, den Sie im Unterricht immer dabei haben und den Sie jederzeit abgeben können. Wir sammeln Ihren Hefter gelegentlich ein und ziehen Ihre Aufzeichnungen zur Bewertung Ihrer sonstigen Mitarbeit heran.

! Versehen Sie jedes Arbeitsblatt, jedes Informationsblatt und jede Kompetenzüberprüfung, die wir Ihnen geben, unverzüglich mit Ihrem Namen und heften Sie diese Materialien in Ihren Hefter, der ebenfalls mit Ihrem Namen gekennzeichnet werden muss.

Gegebenenfalls erhalten Sie durch uns einen geschützten Zugang zu Informatiksystemen. Das Passwort für solche Zugänge dürfen **keinesfalls** an andere (auch nicht im Kurs) weitergegeben werden.

Was haben Sie?

Aufgaben 0.1

a) Sie verfügen über ein mobiles Informatiksystem (Smartphone oder Tablet)?
b) Geben Sie die Bezeichnung für das Betriebssystem dieses Informatiksystems an:

c) Geben Sie die genaue Versionsnummer des Betriebssystems an: H

Vorhaben 1

Informatik – Fachgebiete, typische Problemstellungen, Geschichte

1.1 Welche Kompetenzen[1] sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (IF1, M).
- entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (IF2, M).
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (IF4, K).
- beschreiben und erläutern den strukturellen Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der Von-Neumann-Architektur (IF4, A).
- bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (IF5, A).
- erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (IF5, A).
- nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (IF5, K).
- stellen ganze Zahlen und Zeichen in Binärcodes dar (IF4, D).
- interpretieren Binärcodes als Zahlen und Zeichen (IF4, D).

[1] Alle in diesem Skriptum ausgewiesenen Kompetenzen orientieren sich an dem KLP-Informatik, der ab dem Schuljahr 2014/2015 die verpflichtende Grundlage für das Schulfach Informatik in der gymnasialen Oberstufe ist (vgl. (MSW-NW 2013)).

1.2 Informatik ist . . .

Die Bezeichnung Informatik ist ein sogenanntes Kompositum aus den beiden Begriffen **Infor**mation und Auto**matik**. Der Begriff wurde von dem deutschen Ingenieur Karl Steinbuch (1956/57) geprägt. Die seinerzeitige Definition, nämlich **Automatische Informationsverarbeitung**, ist bis heute gültig.

Informatik ist die Wissenschaft, die sich mit den Fragen der automatischen Verarbeitung von Information (besser: Daten) beschäftigt.

1.2.1 Fachgebiete der Informatik

1976 wurde vom Fakultätentag Informatik eine fachliche Gliederung der Wissenschaft Informatik in Form von sechs Fachgebieten (vgl. Abbildung 1.1) beschlossen.

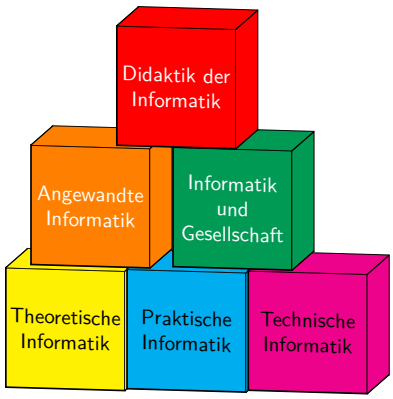


Abbildung 1.1: Fachgebiete – Informatikturm

Danach gibt es drei Fachgebiete, die zusammenfassend als **Kerninformatik** bezeichnet werden: *Theoretische Informatik*, *technische Informatik* und *praktische Informatik* – außerhalb der Kerninformatik liegen die Fachgebiete: *angewandte Informatik*, *Informatik und Gesellschaft* und *Didaktik der Informatik*.

Aufgabe 1.1

Begründen Sie Ihre vermutete Zuordnung der folgenden Elemente aus der Informatik zu den sechs Fachgebieten der Informatik. H

1. Ausspionieren von Informatiksystemen L
2. Warten auf die Antwort einer Suchmaschine L
3. Ein Dokument wird ausgedruckt L

4. Jeder Mensch soll programmieren können L
5. Die Geschwindigkeit eines Prozessors hat zugenommen L
6. Soziale Netzwerke L

Als Ergebnis der weiteren Arbeit sollten Sie anschließend für konkrete Fragestellungen sicher entscheiden können, welches Fachgebiet der Informatik primär für die Bearbeitung zuständig ist. Eine Übersicht finden Sie in der Abbildung 1.2.

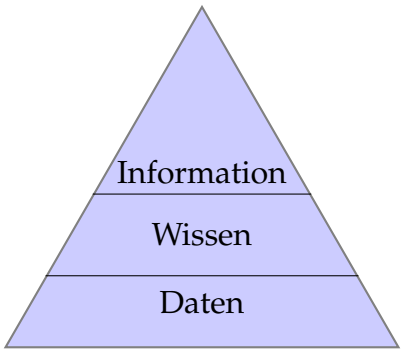
1.2.2 Daten – Wisssen – Information

Definition von Informatik

Informatik kann aus den beiden Bestandteilen Information und Automatik gebildet werden.

> *Definition*: **Informatik** ist die Wissenschaft, die sich mit der **automatischen Verarbeitung** von Information (besser: **Daten**) beschäftigt

Dann muss noch geklärt werden, was »automatische Verarbeitung« und »Information« bedeuten.



Für den Begriff **Information** gibt es verschiedene Erklärungen. Wir müssen uns auf eine Erklärung einigen.

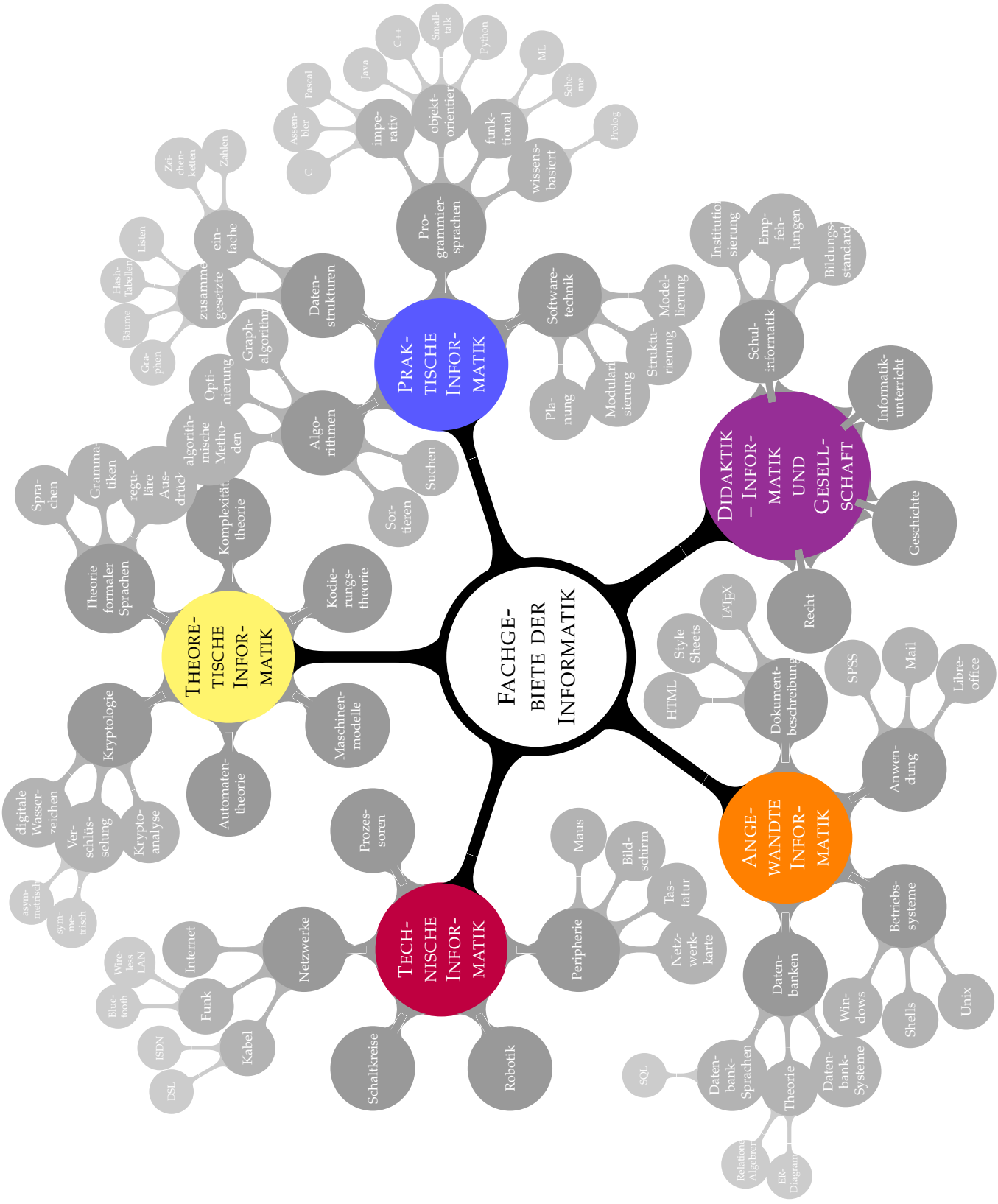
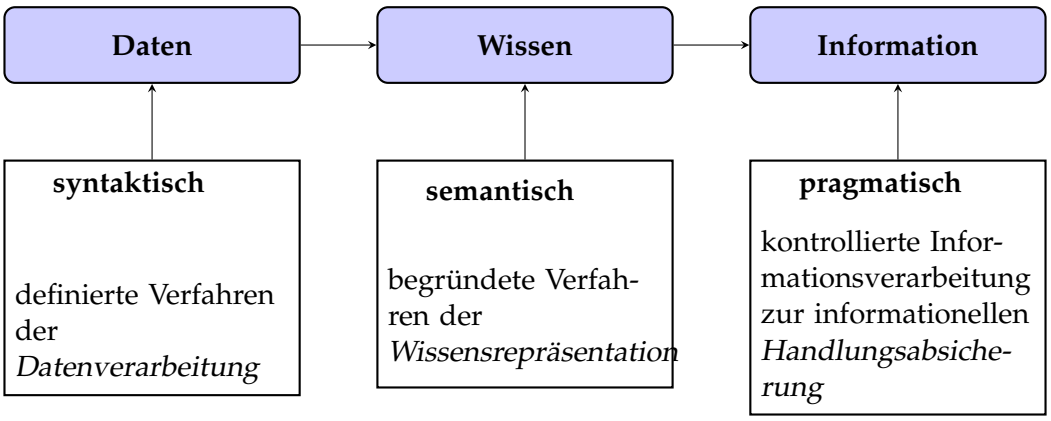


Abbildung 1.2: Fachgebiete der Informatik



Diese Variante erklärt die Begriffe Daten, Wissen und Information durch Fachbegriffe aus der Informatik, die gebräuchlich sind: **Syntax**, **Semantik** und **Pragmatik**.

Daten – Syntax

Sie kennen den Syntaxfehler, wenn Sie einen Taschenrechner »falsch« bedienen. Er tritt nicht auf, wenn Sie die Wurzel aus -3 ziehen, wohl aber, wenn Sie einen zu berechnenden Ausdruck mit einer schließenden Klammer beginnen.

Die Syntax für mathematische Ausdrücke (engl. expression) legt u. a. fest, dass einer schließenden Klammer immer eine öffnende vorausgegangen sein muss.

Daten liegen in strukturierter Form vor, d. h. sie sind nach bestimmten Regeln geformt. Diese Regeln legen die »Sprache« hinsichtlich ihrer Struktur fest – aus den Sprachfächern kennen Sie die Bezeichnung Grammatik – die Grammatik einer Sprache ist deren Syntax.

Die Grammatik macht keine Aussage über den Sinn, d. h. ob eine Aussage, die grammatikalisch richtig ist, eine Bedeutung hat oder Nonsens ist, hat nichts mit der Grammatik zu tun.

Wissen – Semantik

Der Sinn oder besser die Bedeutung der Daten wird erst durch die Semantik festgelegt. Wir kommen mit der Semantik von der Strukturebene auf die Wissensebene. Die Voraussetzung für die Wissensebene sind dabei syntaktisch korrekte Ausdrücke.

Information – Pragmatik

Die Aktivitäten, die ein biologisches System entfaltet, wenn es mit Daten und Wissen ausgestattet wird. Einfaches Beispiel ist der Börsenmakler, der aus dem Fenster springt, nachdem er verstanden hat, dass seine Beratung zum finanziellen Ruin führen wird.

Informatiksysteme sind nicht in der Lage, Wissen so zu interpretieren, dass diese Ebene je erreicht wird – allerdings kann mit Verfahren »so getan werden, als ob«, dies ist z. B. der Fall, wenn jemand mit seinem Informatiksystem schimpft, weil die Aktion, die das System durchführt, nicht dem entspricht, was der Mensch erwartet.

1.2.3 Freihandversuche: Sprache, Wegenetz, CD-Kratzer

mi2mu – Beispiel für ein Problem in einer künstlichen Sprache

Die Bezeichnung »Wort« ist nicht identisch mit dem, was wir in »lebenden« Sprachen unter »Wort« verstehen – in künstlichen Sprachen entspricht ein »Wort« eher dem, was Sie aus lebenden Sprachen als »Satz« kannen. Dennoch gibt es in natürlichen Sprachen auch Regeln zur Wortbildung, z. B. wird festgelegt, welche Vorsilbe (Präfix)[2] oder welches Ende (Postfix)[3] ein Wort in einem bestimmten Zusammenhang hat.

Das folgende Beispiel wurde nach (Hofstadter 1985, S. 37–45) gestaltet.

$mi \overset{*}{\vdash} mu$ – Sprachbeschreibung

Symbole m i u

Startwort mi

Regeln

1. Ist Xi ein Wort, dann auch Xiu
2. Ist mX ein Wort, dann auch mXX
3. In jedem Wort kann iii durch u ersetzt werden
4. uu kann aus jedem Wort weggelassen werden

X in Regel 1 und 2 steht für eine beliebige Folge von Symbolen
XX bezeichnet die Verdoppelung der Symbolfolge X

Von mi nach mu

Aufgabe 1.2

Geben Sie die Folge von Umformungen an, die vom Startwort mi durch Anwendung der Regeln zu mu führt. H

[2] Ein Beispiel für eine Vorsilbe: binden – verbinden, hier ist »ver« die Vorsilbe.
[3] Beispiel für ein Wortende: Informatiksystem – Informatiksysteme, hier ist das »e« die Endung, die aus dem Singular einen Plural erzeugt (wird in den natürlichen Sprachen auch Suffix genannt).

Nachdem Sie etwas probiert haben, stellen Sie fest, dass die Lösung vielleicht doch schwieriger ist, als gedacht. Daher empfiehlt sich ein systematischer Aufbau: beginnend mit dem Anfangswort werden systematisch jeweils alle Regeln angewendet.

Von Bäumen, Knoten, Kanten und Kindern Man erhält einen »Baum«, der auf dem Kopf steht – oben ist die Wurzel des Baumes; die jeweils nächste Ebene wird durch die Anwendung einer Regel erreicht. Informatikkundige zeichnen Bäume umgedreht – also: oben ist die **Wurzel** und der Baum **wächst nach unten** – die Stellen mit den Verzweigungen heißen **Knoten** und die Verbindungen werden **Kanten** genannt. Außerdem wird von Kindern gesprochen, wenn die durch Kanten verbundene Knoten in der nächsten Ebene angesprochen werden.

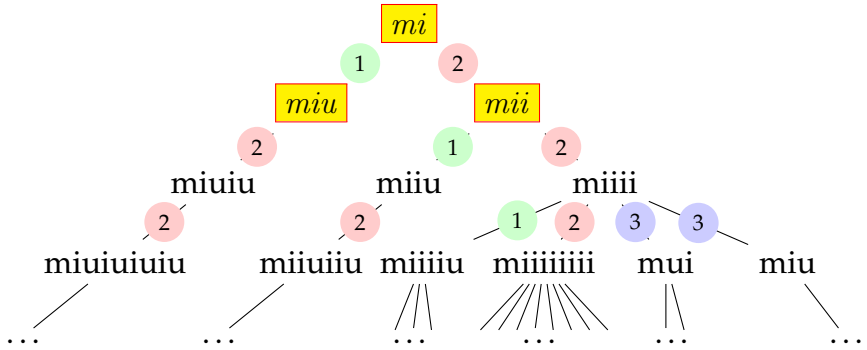


Abbildung 1.3: mi2mu – alle Möglichkeiten in einer Baumstruktur angeordnet

Wegauswahl: Problemstellung – Abstraktion – erste Stufe

Es ist die kürzeste Verbindung zwischen zwei Orten zu ermitteln.

Um die Problemstellung zu konkretisieren, zeichnen wir das zur konkreten Situation gehörende Netz[4] (vgl. Abbildung 1.4). Die Punkte (= Orte) werden hier mit Kleinbuchstaben bezeichnet – die Abstände durch einheitenfreie Zahlen oberhalb der jeweiligen Verbindung.

[4] Eine ähnliche Abbildung und die LaTeX -Quelle wurden von Kjell Magne Fauske auf der Webseite http://www.texample.net/tikz/examples/prims-algorithm/ veröffentlicht.

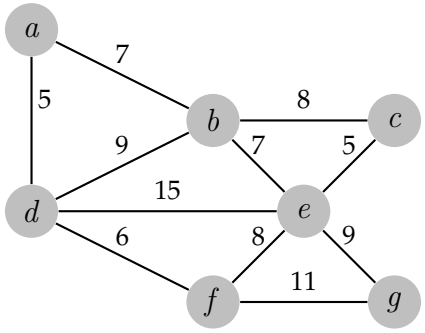


Abbildung 1.4: Schema eines Wegenetzes

Aufgaben 1.3

a) Bestimmen Sie den kürzesten Weg von a nach g
b) Beschreiben Sie stichwortartig, wie Sie vorgegangen sind
c) Verallgemeinern Sie das Verfahren – beschreiben Sie, wie man vorgehen kann, um den kürzesten Weg von einem Punkt zu einem beliebigen anderen in einem Netzwerk zu finden

Damit Sie verstehen, wie ein Navigationsgerät den kürzesten Weg zwischen zwei Orten bestimmen kann, hilft es, einen Ameisenstaat zur Hilfe zu nehmen: Alle Ameisen verteilen sich im Startknoten des Netzes auf die möglichen Wege (Kanten) und laufen (alle gleich schnell) los. Sobald sie den ersten Knoten auf ihrer Kante finden, verteilen sie sich auf alle Wege, die von diesem Knoten wegführen. Nun kommt es vor, dass sich irgendwo auf dem Wegenetz Ameisen entgegen kommen (treffen) – dann kann diese Kante (Wegstück) aus dem Netz entfernt werden. Auf diese Art wird das Wegenetz immer einfacher, bis am Ende nur ein Weg vom Anfangsknoten zum Ziel übrigbleibt.

Dieser Weg ist der kürzeste.

CD hat Kratzer – na und?

Vielleicht kennen Sie das Problem: wird eine Vinyl-Schallplatte unsachgemäß behandelt, so beeinträchtigt das Ergebnis den Hörgenuss. Haben Sie schon mal eine (leicht) verkratzte CD gehört?

Nun – bei einer CD beeinträchtigen kleinere Schäden offenbar nicht den Hörgenuss.

Wir werden sehen, wie die Daten der CD rekonstruiert werden können, obwohl sich Kratzer auf der CD befinden.

Freihandversuch Für eine $n \times m$ Tabelle, die mit 0 und $\times$ gefüllt ist, werden Teilnehmende gebeten, *eines* der Felder zu ändern. Nach einem kurzen Blick kann die Vortragende herausfinden, welches Feld geändert wurde und es richtigstellen.

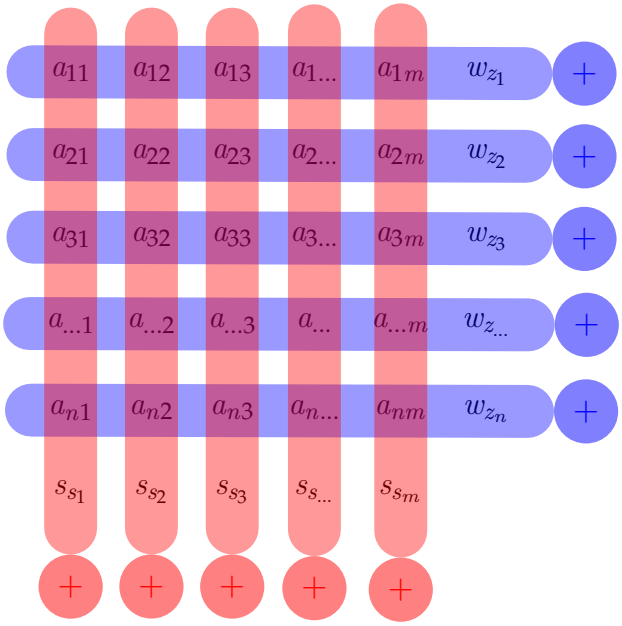


Abbildung 1.5: Zeilenweise und spaltenweise Addition (modulo 2)

Erläuterung zum Freihandversuch In dem in Abbildung 1.5 dargestellten Tabelau[5] – n Zeilen (waagerecht) und m Spalten (senkrecht) werden die Inhalte ($n \cdot m$ Bit) um die beiden Ränder (also $n + m$ Bit) ergänzt, damit nach der Datenübertragung die Daten geprüft und ggf. rekonstruiert werden können. Dazu werden (vor der Datenübertragung) $w_{z_{1\ldots n}}$ und $s_{s_{1\ldots m}}$ so ergänzt, dass die Anzahl an $\times$-Zeichen immer gerade ist (0 ist eine gerade Zahl).

Nach der Datenübermittlung können wir eine Fehlerkorrektur vornehmen. **Ein** falscher Eintrag führt dazu, dass sowohl in der zugehörigen Zeile als auch in der Spalte des falschen Eintrages der rechts $w_{z_{1\ldots n}}$ bzw. unten $s_{s_{1\ldots m}}$ am Rand stehende Eintrag nicht stimmt und wir damit genau ermitteln können, wo der falsche Eintrag steht. Diesen Eintrag können wir dann rekonstruieren. Auf diese Weise haben wir eine einfache Korrekturmöglichkeit gefunden, die z. B. bei einer CD dazu führt, dass ein Kratzer dem Hörgenuss nicht abträglich ist.

[5] Abbildung nach einer Idee von Alain Matthes:
vgl. http://www.texample.net/tikz/examples/mnemonic-rule-for-matrix-determinant/

1.2.4 Einordnung in die Fachgebiete und in die Geschichte der Informatik

Fachgebiete

Die bisher angesprochenen Elemente der Informatik können den Fachgebieten zugeordnet werden (vgl. Tabelle 1.1). Dabei ist diese Zuordnung nicht immer eindeutig, weil es durchaus möglich ist, dass ein Element für mehrere Fachgebiete bedeutsam ist.

Am Beispiel der Wegauswahl kann man dies gut deutlich machen: Dargestellt ist ein Problem aus dem Alltag, wir alle kennen Navigationsgeräte oder Programme, die in Form von Informatiksystemen dieses Problem für uns lösen → Zuordnung 1: Angewandte Informatik.

Andererseits tritt dieses Problem innerhalb der Informatik auf, da auch in den Netzen der Informatik (z. B. im Internet) entschieden werden muss, wohin Datenpakete weitergegeben werden (das sogenannte Routingproblem) – auch innerhalb von Betriebssystemen tauchen solche Problemlagen auf → Zuordnung 2: Praktische Informatik und Technische Informatik.

Offenbar handelt es sich um eine häufiger auftretende Problemlage, für die es sich lohnt, herauszufinden, wie man es effizient – mit möglichst wenig Zeitaufwand – lösen kann: hier kommt die → Zuordnung 3: Theoretische Informatik ins Spiel.

Dennoch haben wir uns für die Tabelle bei den Problemstellungen jeweils für eine primäre Zuordnung entschieden.

Kurzbezeichnung	**Bereich**	**Fachgebiete**
mi2mu	künstliche Sprache	Theoretische Informatik
CD-Kratzer	Fehlerkorrektur	Technische Informatik
Wegauswahl	Navigation/Routing	Praktische Informatik

Tabelle 1.1: Zuordnung der Beispiele zu den Fachgebieten der Informatik

Geschichte

Mit unseren Beispielen läßt sich eine weitere Frage beantworten: Seit wann wurden/werden solche Fragen überhaupt systematisch im Zusammenhang mit der Informatik bearbeitet?

Die Entwicklung der Fachwissenschaft Informatik kann von drei Seiten betrachtet werden – einmal die Entwicklung der technischen Basis (hardwareorientiert) → Zuordnung: Technische Informatik. Diese Sicht ist weit verbreitet, man kann sich **Artefakte** der Informatik ansehen z. B. im HeinzNixdorfMuseumsforum in Paderborn.

Betrachten wir die Ideen, die mit der Informatik verbunden sind, sieht »die Welt« ganz anders aus: es stellt sich heraus, dass wir sehr viele Dinge tun, die mit den Mitteln der Informatik gut beschrieben werden können. Dies ist der Hauptgrund, der dazu führt, dass Informatik inzwischen überall präsent ist – nicht nur in Form von Informatiksystemen – sondern deutlich darüber hinaus: sobald Strukturen eine Rolle spielen, kann die Informatik helfen, dies zu beschreiben und mit diesen Beschreibungen zu arbeiten. Die damit verbundenen Elemente können den Fachgebieten der Praktischen und der Theoretischen Informatik zugeordnet werden: es geht um Strukturen → **Datenstrukturen** und um die Beschreibung von Abläufen → **Algorithmen**.

Eine dritte Sicht auf die Informatik ist die Sicht von außen – hier kommen die Fachgebiete ins Spiel, die nicht zur Kerninformatik gehören. Hier wird »das Geld verdient« → Angewandte Informatik, hier fragt man nach den gesellschaftlichen Veränderungen durch die Informatik → Informatik und Gesellschaft und beschäftigt sich damit, wie Menschen (insbesondere Lernende) Informatikthemen verstehen und sie erkunden können → Didaktik der Informatik.

Zur ersten Sicht kann man sagen: da **Artefakte der Informatik** überall und permanent präsent sind, fällt es schwer, Bereiche zu finden, in denen keine Informatik(-produkte) auftauchen. Allerdings ist es schon bemerkenswert, dass es bereits deutlich vor den ersten Informatiksystemen Pläne für solche Systeme gab: So entwickelte Charles BABBAGE ab ca. 1822 die Ideen der »Analytical Engine« (Menabrea 1842). Durch die klare Ausgestaltung der Mechanismen in dem o. g. Manuskript fällt Ada LOVELACE die Rolle der ersten Programmiererin zu – sie machte sich bereits Gedanken über die Grenzen der Maschine.

Die zweite Sicht ist stärker an der Ideengeschichte der Informatik orientiert: Wann beginnen die Menschen, sich mit Strukturen in der Weise zu beschäftigen, dass sie diese beschreiben, um damit handeln zu können?

Wenn Sie sich an den Anfang Ihrer Kindergarten- oder Schulzeit zurückerinnern, geschieht dies recht früh: Sie lernen, aus kleinen Elementen größere Strukturen zu bauen; das kann man bereits als **informatisches Tun** ansehen: Eine Abfolge von Schritten, die durchgeführt wird, um ein Problem zu lösen. Wenn Sie mit anderen etwas gemeinsam tun, um eine Aufgabe zu bewältigen, kommen weitere informatikspezifische Fragen hinzu: Wie wird festgelegt, welche [Zwischen-]Ergebnisse nötig sind, um weiter miteinander erfolgreich zu arbeiten? Ein – aus Informatiksicht – typisches Problem: Festlegung und Gestaltung einer Schnittstelle, an die sich alle halten müssen, die die Schnittstelle nutzen (wollen, müssen) oder sie »bauen«.

In der Geschichte gibt es ganz früh Hinweise darauf: jedes Ritual stellt durch klare Vorschriften sicher, dass der beschriebene Vorgang genau in dieser Weise durchgeführt wird – auch die Ingredienzien sind häufig sehr genau spezifiziert.

Manche Überlegungen in dieser Hinsicht gehen sehr weit:

- Gottfried Wilhelm LEIBNIZ (um 1680)

  > Es wird dann beim Auftreten von Streitfragen für zwei Philosophen nicht mehr Aufwand an wissenschaftlichem Gespräch erforderlich sein als für zwei Rechnerfachleute. Es wird genügen, Schreibzeug zur Hand zu nehmen, sich vor das Rechengerät zu setzen und zueinander [. . .] zu sagen: Laßt uns rechnen.

  (Dreschler-Fischer 2000, S. 169)

Zur Vertiefung und zu dem sehr bekannten Informatiker Alan M. TURING kann das Tondokument (Bertram 2014) herangezogen werden.

1.2.5 Lernerfolgskontrollen

Aufgaben 1.4

a) a) **(8 Punkte) Informatik – zum Begriff**

i. Geben Sie **Ihre** Definition für Informatik an.

ii. Ordnen Sie die folgenden beiden Aussagen einer der Ebenen **Pragmatik, Syntax** oder **Semantik** zu und begründen Sie Ihre Zuordnung:

- »Eine Studentin sucht Literatur zu einem bestimmten Thema.«
- »Bildarchive werden häufig von Journalistinnen in Anspruch genommen, um einen Artikel zu illustrieren; dabei ist meist das Thema vorgegeben, aber nicht der Bildinhalt.«

iii. Benennen Sie die sechs Fachgebiete, in die Informatik üblicherweise aufgeteilt wird.

iv. Ordnen Sie die folgenden Begriffe den von Ihnen in a(a)iii genannten Fachgebieten zu:

MP3-Player, Software, Programmiersprache, Datenschutz, Linux, Fahrroutenoptimierung

v. Grenzen Sie die Begriffe **Information, Daten** und **Wissen** voneinander ab.

b) **(8 Punkte) Informatik – zum Begriff**

i. Erläutern Sie die Begriffe **Semantik, Pragmatik, Syntax** und grenzen Sie die Begriffe voneinander ab.

ii. Nennen Sie die sechs Fachgebiete der Informatik und ordnen Sie die folgenden Begriffe den Fachgebieten zu:

Programmiersprache Python, Datenbank, Persönlichkeitsschutz, Informatische Bildung, Hardware, Betriebssystem

iii. Ordnen Sie die folgende Aussage einer der Ebenen **Daten**, **Wissen**, **Information** zu und begründen Sie Ihre Zuordnung:

»Ein Dokument wird als Folge von Zeichen/Symbolen aufgefasst. Auf dieser Ebene kann beispielsweise mit Methoden agiert werden, die Zeichenketten in Texten oder die nach Merkmalen wie Farbe, Textur und Kontur suchen.«

iv. Geben Sie **Ihre** Definition für Informatik an.

1.3 Betriebssysteme – von Äpfeln und Pinguinen

Dieser Abschnitt wurde der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zum Thema »Betriebssysteme« entnommen (dort: Kapitel 3) und an einigen Stellen geändert.

Betriebssysteme sind wie Behörden: Eigentlich machen sie selbst nichts wirklich Produktives, aber ohne sie läuft nichts. Wir alle haben schon – zu Recht! – über dieses oder jenes Amt geschimpft, sind aber im Großen und Ganzen mit einer recht effizienten Verwaltung gesegnet (wie es sich in einer komplett verwaltungsfreien Welt lebt, kann man in dem Buch *Snow Crash* (Stephenson 1995) schön nachlesen).

Es gibt für »normale« Computer derzeit drei wichtige Betriebssysteme: Window, von der Firma Microsoft entwickelt; MacOS, von der Firma Apple entwickelt; und Linux, welches ursprünglich von Linus TORVALDS programmiert wurde und heute von einer so genannten *Community* weiterentwickelt wird. Im Gegensatz zu Window und MacOS ist Linux komplett frei verfügbar – wer Spaß daran hat, kann Linux völlig legal beliebig abändern und erweitern. Erfahrungsgemäß haben Informatiker daran Spaß, andere Menschen nicht.

In diesem Abschnitt soll es nicht um die Details einzelner Betriebssysteme gehen, diese ändern sich sowieso alle paar Jahre. Wichtiger ist es, die grundlegenden Ideen zu verstehen, welche auch in allen Betriebssystemen letztendlich sehr ähnlich umgesetzt sind:

1. Betriebssysteme *verwalten* »alles, worum sich Programme streiten könnten«. Beispielsweise streiten sich Programme sehr gerne um den Zugriff auf Drucker, Soundkarten, Grafikkarten oder externe Speichermedien[6], weshalb hier das Betriebssystem »schlichten« muss.

[6] Externe Speicher sind z. B. Festplatten oder SSDs.

2. Betriebssysteme sind in zwei Schichten getrennt, einen hardwareunabhängigen Teil und hardwareabhängige *Treiber*.
3. Daten werden von allen gängigen Betriebssystemen in Form von *Verzeichnisbäumen* organisiert.

Das Betriebssystem

Das Betriebssystem ist ein großes Programm, das unter anderem folgende Dinge leistet:

- Druckauftragsverwaltung
- Dateiverwaltung
- Nutzerverwaltung
- Prozessverwaltung

Aufgabe 1.5

Um welche Ressourcen könnte sich ein Betriebssystem noch kümmern? L

Welche Betriebssysteme gibt es?

Unterscheidungsmerkmale von Betriebssystemen

Man kann Betriebssysteme unterscheiden nach:

- Funktionsumfang
  - Vom Betriebssystem eines Fahrradcomputers …
  - … über Window und Linux …
  - … bis zum Betriebssystem einer »Mainframe«.
- Abstammung
  - Unix-Familie: Linux, MacOS X, Solaris, AIX, Android, iOS, …
  - MS-DOS-Familie: MS-DOS, DR-DOS, Window
  - Eigenentwicklungen: PalmOS, BeOS, OS/2, …

Zur Historie von Window

1981 IBMs erste »Personal Computer« werden mit dem »Microsoft Disk Operating System« (*MS-DOS*) ausgeliefert, da ein eigenes nicht zur Verfügung steht.

1986 Microsofts *Window* wird vorgestellt.

1987 IBMs eigenes Betriebssystem *OS/2* wird vorgestellt.

1991 Microsoft treibt Window voran, das gegen OS/2 durchgesetzt wird.

Zur Historie von Linux

ab 1960 Verschiedene Firmen stellen Varianten des Betriebssystems *Unix* her.

1983 Das *GNU-Projekt* (»Gnu is Not Unix«) macht es sich zur Aufgabe, ein *freies* Unix zu schaffen.

1991 *Linus* TORVALDS startet ein Projekt, das GNU-Unix auf PCs mit Intel-Prozessoren portiert.

heute *Linux* ist das größte und verbreitetste freie Betriebssystem.

1.3.1 Ressourcen

Druckauftragsverwaltung

Ressource I: Die Verwaltung von Druckaufträgen ist scheinbar einfach.

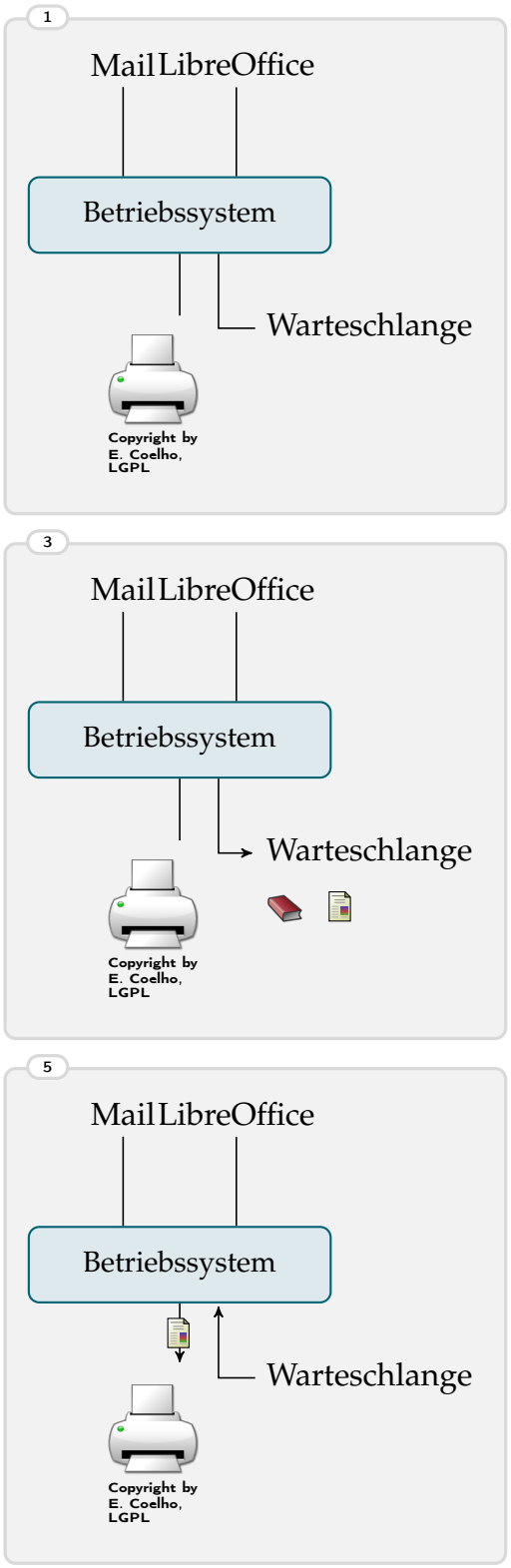
Problemstellung

- Anwendungsprogramme sollen Texte und Grafiken drucken können.
- Jedoch können mehrere Anwendungsprogramme gleichzeitig versuchen, etwas zu drucken.

Lösung

- Anstatt Daten direkt an die Drucker zu schicken, erstellen Anwendungsprogramme *Druckaufträge*, die sie an das Betriebssystem schicken.
- Das Betriebssystem reiht die Druckaufträge in eine *Warteschlange* ein und schickt sie dann in der Reihenfolge ihres Eintreffens an den Drucker.

Beispiel, wie die Druckauftragsverwaltung arbeitet



Dateiverwaltung

Ressource II: Daten

Problemstellung

- Die *Daten* der Benutzerinnen und Benutzer müssen verwaltet werden.
- Daten schließt *Texte*, *Grafiken*, *Filme*, *Programme* und viele Dinge mehr ein, die *sehr unterschiedliche Größe* haben.
- Weiterhin soll der Zugriff *schnell* sein.

Lösung

- Alle Daten werden als *Folgen von Bytes* gespeichert.
- Jede Einheit von Daten wie ein Dokument oder ein Programm oder ein Film bildet eine *Datei* (File) und bekommt einen *Namen*.
- Zur größeren Übersichtlichkeit werden diese Dateien in einem *Verzeichnisbaum* angeordnet.

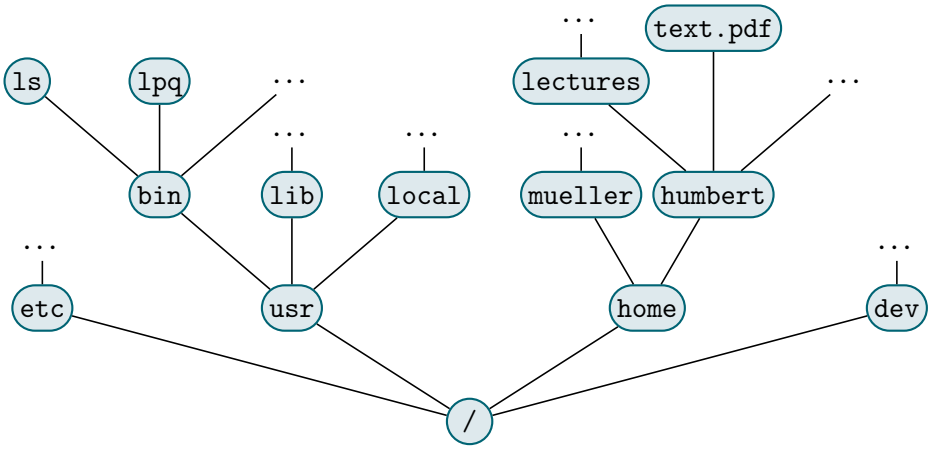
Was man über Dateien wissen sollte

- Dateien sind Folgen von Bytes.
- Ihre *Länge* oder *Größe* wird in Byte gemessen, wobei die kürzesten Dateien Null Bytes haben, die längsten mehrere Gigabyte.
- Ihr *Name* ist eine vom Inhalt unabhängige Zeichenfolge, die am besten keine Sonderzeichen enthält. Der Name besteht oft aus zwei durch einen Punkt getrennte Teile, wobei der zweite Teil (*Suffix* oder *Dateierweiterung* genannt) angibt, um welche Art Datei es sich handelt.

Was man über Verzeichnisse wissen sollte

- Verzeichnisse enthalten Dateien und wieder Verzeichnisse (Unterverzeichnisse).
- Genau wie Dateien haben Verzeichnisse einen Namen.
- Das oberste Verzeichnis nennt man das *Wurzelverzeichnis* (root directory).

Ein typischer Unix-Verzeichnisbaum



Das Konzept des absoluten Pfads

- Die Folge von (Unter-)Verzeichnisnamen, die zu einer Datei führt, bezeichnet man als *absoluten Pfad*.
- Innerhalb des Pfades werden die verschiedenen Verzeichnisnamen durch Schrägstriche getrennt (Unix) oder durch umgedrehte Schrägstriche (Window).

  Beispiel: `/home/humbert/text.pdf`

  Beispiel: `\Window\system32\Window.exe`
- Kommt innerhalb eines Pfades »`..`« vor, so ist damit das Verzeichnis eine Ebene höher gemeint.

Aufgabe 1.6

Welche Datei bezeichnet der Pfad

Beispiel: `/usr/../home/humbert/../../dev/null`? H L

Das Konzept des relativen Pfads

- Ein *relativer Pfad* setzt einen Pfad relativ zu einem *aktuellen Verzeichnis* fort.
- Auf das aktuelle Verzeichnis kann man sich mittels »`.`« beziehen.
- Beispiel: Wir befinden uns im Verzeichnis `/home` – dann ist `./humbert/text.pdf` die Datei `/home/humbert/text.pdf` (kompletter Pfad).
- Beispiel: Wir befinden uns im Verzeichnis `/home/mueller` – dann ist die Datei `../humbert/text.pdf` dieselbe Datei wie oben.

Aufgabe 1.7

Sie befinden sich im Verzeichnis: `/usr/local/bin`

Welche Datei bezeichnet dann `../../local/../bin/lpq`? H L

Nutzerverwaltung

Ressource III: Nutzer und Rechte

Problemstellung

- Die Daten von vielen Personen (Benutzerinnen und Benutzern) sollen auf einem Informatiksystem gespeichert werden.
- Dabei soll es Benutzerinnen und Benutzern gestattet sein, manche Daten anderer Benutzerinnen und Benutzern zu lesen, andere aber wieder nicht.

Lösung

- Jede Benutzerin und jeder Benutzer bekommt ein Verzeichnis, in dem ihre/seine Daten liegen.
- Für jede Datei gibt es *Rechte*, die festlegen, wer was darf.
- Für jede Benutzerin und jeden Benutzer speichert das Betriebssystem den Hashwert eines *Passworts* und die Nutzerinnen und Nutzer müssen sich *einloggen* und damit *authentifizieren*.

Dateibesitz in Unix

Das Unix-Betriebssystem regelt zunächst den *Besitz* von Dateien:

1. Jede Datei gehört genau einer bestimmten *Nutzerin* bzw. einem bestimmten *Nutzer*.
   - Besitzerin oder Besitzer ist, wer die Datei angelegt hat.
   - Nur die Besitzerin rsp. der Benutzer kann die Rechte ändern.
2. Jede Datei gehört genau einer bestimmten *Gruppe*.
   - Eine Gruppe ist eine Menge von Benutzerinnen und Benutzern, wobei Benutzerinnen und Benutzer aber Mitglieder mehrerer Gruppen sein können.

   Genau wie Dateien werden Gruppen auch vom Betriebssystem verwaltet.

Zugriffsrechte in Unix

Das Unix-Betriebssystem kennt drei Arten von Rechten:

1. *Leserecht (r-Recht)*

   Das Recht, den Inhalt einer Datei / eines Verzeichnis zu lesen.

2. *Schreibrecht (w-Recht)*

   Das Recht, den Inhalt einer Datei / eines Verzeichnis zu ändern.

3. *Ausführungsrecht (x-Recht)*

   Das Recht, ein Programm auszuführen, dessen Code in der Datei liegt. Fehlt dieses Recht für ein Verzeichnis, so kann man es nicht in einem Dateipfad benutzen (»man kann nicht hineinwechseln«).

Die Rechtetabelle einer Datei

Für jede Datei wird gespeichert, welche der drei Rechte bestimmte Personengruppen haben:

1. Es wird gespeichert, welche Rechte die Besitzerin bzw. der Besitzer hat.
2. Es wird gespeichert, welche Rechte die Angehörigen der Gruppe haben.
3. Es wird gespeichert, welche Rechte alle anderen Benutzerinnen und/oder Benutzer haben.

Dies führt zu einer Rechtetabelle wie im folgenden Beispiel:

	read	write	execute
user	yes	yes	yes
group	yes	no	yes
other	no	no	no

Dies schreibt man auch kurz so: `rwxr-x---`.

Zur Übung

Aufgabe 1.8

Die Nutzerin `eva` und der Nutzer `adam` sind beide in der Gruppe `eden`, aber nur `adam` ist in der Gruppe `men`. In einem Verzeichnis finden sich folgende Dateien:

Dateiname	Besitzer*in	Gruppe	Rechte
`apple.txt`	`eva`	`eden`	`rwxr-----`
`snake.txt`	`adam`	`eden`	`rw-rw-r--`
`eat`	`adam`	`men`	`rwxrwxr-x`

Welche Dateien können jeweils Adam und Eva lesen und welche schreiben? H L

Prozessverwaltung

Ressource IV: Prozesse

Problemstellung

- Benutzer wollen mehrere Programme gleichzeitig ausführen.
- Falls Prozesse dabei *böse Dinge* versuchen, muss man sie gewaltsam stoppen können.

Lösung

- Das Betriebssystem verwaltet eine *Prozessliste*.
- Jeder Prozess hat eine *Nummer* und *gehört* einem Benutzer.
- (Nur) der Benutzer kann seine Prozesse *töten*.

Aufgabe 1.9

Was könnten *böse Dinge* alles sein? L

1.3.2 Aufbau von Betriebssystemen

Schichten

Die Druckauftragsverwaltung

Beim Drucken gibt es *zwei unterschiedliche Probleme*:

1. Die Druckaufträge müssen verwaltet werden. Dies beinhaltet Tätigkeiten wie:
   - Nummerieren der Druckaufträge,
   - Verschicken von Rückmeldungen,

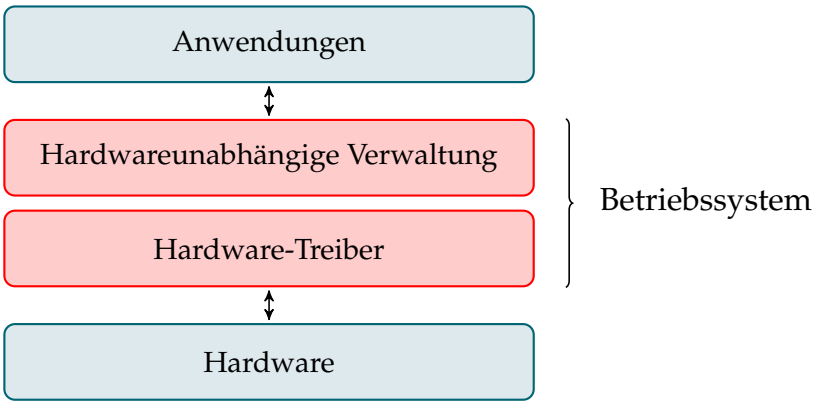
- Auflistung der angeschlossenen Drucker.

2. Das Übermitteln der Daten an den Drucker. Dies beinhaltet Tätigkeiten wie:
   - Umwandlung der Daten in ein Format, das der Drucker verarbeiten kann (jeder Drucker spricht eine eigene »Sprache«),
     oder Senden der Daten an das richtige Kabel und den richtigen Anschluss.

Wichtige Beobachtung

Die erste Aufgabe ist *hardwareunabhängig*. Die zweite ist *druckerabhängig* und muss *für jeden Drucker anders programmiert werden*.

Ein Betriebssystem besteht aus zwei Schichten



Untere Schicht: Treiber

Was ist ein Hardware-Treiber?

- Ein *Treiber* ist ein austauschbarer Teil des Betriebssystems.
- Er ist ein Programm, das für *eine bestimmte Hardware* eine *bestimmte Aufgabe* löst.
- Baut man eine neue Hardware in einen Computer ein, so muss man in der Regel einen passenden Treiber installieren.

Beispiel

Ein *Druckertreiber* kümmert sich darum,

- zu druckende Daten in ein Format zu konvertieren, das ein bestimmter Drucker *versteht*, oder
- den richtigen Anschluss (beispielsweise USB) anzusprechen.

Er kümmert sich *nicht* darum, Druckaufträge zu verwalten.

Obere Schicht: Shells, Systemaufrufe

Wie sagt man dem Betriebssystem, was man von ihm will?

Benutzer können auf zwei Arten mit dem Betriebssystem kommunizieren:

1. Mittels einer *Shell*.

   Dazu mehr im nächsten Abschnitt.

2. Mittels eines *graphischen Werkzeuges*.

   Diese findet man in graphischen [Benutzungs-]Oberflächen in der Regel in einer Art »Systemmenü«.

Programme kommunizieren mit dem Betriebssystem auf zwei Arten:

1. Ebenfalls mittels einer *Shell*.
2. Mittels so genannter *Systemaufrufe*.

   Hier gibt es für jeden *Verwaltungsakt* des Betriebssystems ein kleines Programm, das man aufrufen kann.

Aufgabe 1.10

Android

Sie können Ihr Smartphone nutzen: Installieren Sie dazu z. B. Terminal IDE `http://is.gd/tpVorQ` und öffnen Sie diese Applikation – einige der unten stehenden Punkte können auf einem Smartphone nicht umgesetzt werden. Grundlegende Elemente von Unix können auch mit dieser Applikation erkundet werden: es wird ein Prompt angezeigt, Sie können Kommandos eingeben – prüfen Sie die Installation durch Eingabe der folgenden Kommandos:

- `pwd`
- `ls -lat`
- `ls -latR`
- `ls -l /`

Den letzte Aufgabenteil der folgenden Liste können Sie ausführen.
Zum zweiten Aufgabenteil: da unter Android keine explizite Benutzerverwaltung zugänglich ist, wechseln Sie in das Verzeichnis `~`, indem Sie zunächst das Kommando `cd ~` ausführen.

Setzen Sie sich an ein Informatiksystem im Selbstlernzentrum und starten Sie es so, dass Linux (Ubuntu) als Betriebssystem benutzt wird oder nutzen Sie ein Linux-System zu Hause.

Probieren Sie folgende Dinge aus:

1. Versuchen Sie, eine CD-ROM in Ihr Laufwerk einzulegen und diese dann im Verzeichnisbaum wiederzufinden. Klappt das auch mit einem USB-Stick?
2. Klicken Sie im Startmenü auf »Befehl ausführen« (Sie können auch die Tasten ALT und F2 drücken) und geben Sie `acroread` ein. Klicken Sie auf »Ausführen«. Was passiert? Können Sie das Programm `acroread` im Verzeichnisbaum finden? Versuchen Sie dasselbe mit dem Programm `kate`!
3. Schauen Sie sich das Verzeichnis `/usr/local` an. Können Sie dort ein ausführbares Programm finden?
4. Führen Sie den Befehl `konsole` aus, um eine Shell zu öffnen. Geben Sie dort (nacheinander) die Befehle `cat /proc/cpuinfo` und `cat /proc/meminfo` ein, um sich den Inhalt der Dateien `/proc/cpuinfo` und `/proc/meminfo` anzeigen zu lassen. Was steht in diesen Dateien?

. . . . . .

1. Erstellen Sie in Ihrem Home-Verzeichnis eine Textdatei `test.txt` mit beliebigem Inhalt. Eine leere Datei können Sie mit dem Kommando `touch test.txt` erstellen. Geben Sie dieser Datei folgende Zugriffsrechte:

   ```
   ------rw-
   ```

2. Können Sie diese Datei nun noch öffnen? Hat Ihre Partnerin Zugriff darauf? Wie verhält es sich mit folgenden Rechten:

   ```
   ---rw-rw-
   ```

3. Schauen Sie sich die Zugriffsrechte der anderen Dateien und Verzeichnisse in Ihrem Home-Verzeichnis an. Ist alles so konfiguriert, wie Sie es gerne hätten? Gibt es im Home-Verzeichnis Ihres Nachbarn Dateien, die Sie verändern oder löschen *könnten*?

. . . . . .

1. Finden Sie das Programm, das die auf dem System momentan laufenden Prozesse anzeigt! Versuchen Sie, den Prozess `init` zu beenden. Klappt das? Was passiert, wenn Sie den Prozess `kicker` beenden?
2. Beenden Sie nun möglichst viele Prozesse. Können Sie das System so beschädigen, dass es neu gestartet werden muss?

Finden Sie das Programm zur Anzeige der Druckerwarteschlange. Können Sie auch Druckaufträge anzeigen lassen, die bereits abgearbeitet sind? Entspricht die Anzeige der Ihrer Nachbarin?

. . . . . .

Geben Sie jeweils die zutreffende Antwort an!

1. Wobei handelt es sich *nicht* um ein Betriebssystem?

   • Linux ☐ • LaTeX ☐ • Window ☐

2. Die Verwaltung einer der folgenden drei Baumarten ist eine der Hauptaufgaben eines Betriebssystems. Welche ist es?

   • Binärbaum ☐ • Dateibaum ☐ • Suchbaum ☐

1.4 Shells – Dialog mit der Waschmaschine

Dieser Abschnitt wurde der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zum Thema Betriebssysteme entnommen (dort: Kapitel 4) und an einigen Stellen geändert.

Zur Erinnerung (vgl. Abschnitt: 1.3): Betriebssysteme sind die Behörden eines Informatiksystems. Bekanntermaßen ist es recht schwierig, mit einer Behörde zu »reden«, zum einen wegen der abstrusen Öffnungszeiten, zum anderen wegen des »Behördendeutsch«, das dort gerne gesprochen wird.

Bei Betriebssystemen liegen die Dinge (leider) ähnlich, auch hier bedient man sich einer eigenen »Sprache« zur Kommunikation mit dem Betriebssystem. Die Programme, die eine solche Kommunikation herstellen, heißen *Shells*. Auch die »Amtssprache« solcher Shells ist etwas kryptisch, wie Sie sehen werden.

Shells arbeiten *dialogbasiert*. Es wird immer abwechselnd eine *(An)frage* durch Sie, den Benutzer, gestellt, woraufhin das Betriebssystem die *Antwort* anzeigt. Die Shell zeigt ein Protokoll dieses fortlaufenden Dialogs an. Solche Dialoge können lustige Komödien sein (geben Sie mal `banner WBGE` ein), sie können aber auch Zeugnis einer Tragödien von Shakespeare'scher Tragweite sein (geben Sie mal `cd /; rm -rf *` ein – oder vielleicht doch lieber nicht). Wenigstens wirkt die Tragödie im letzten Fall sehr kartharsisch.

1.4.1 Einführung zu Shells

Wozu dienen Shells?

Shells stellen eine Dialog zwischen einem *Benutzer oder einem Programm* und einem *Programm* her. Der Ablauf:

1. Die Shell zeigt einen *Prompt* an

   Der/Das Prompt ist eine Eingabeaufforderung, eine Frage der Form »Meisterin, wie kann ich Euch dienen?«.

2. Die Benutzerin gibt einen *Befehl* ein.
3. Die Shell führt diesen aus.
4. Danach zeigt sie wieder einen Prompt und die Sache beginnt von neuem.

Die Befehle, die *eine Shell versteht*, hängen von der Shell ab.

Viele Programme haben Shells

- Shells zum Unix-Betriebssystemkern:
  - `sh`, `bash`, `csh`, `ksh`.
- Shell zum Window-Betriebssystemkern:
  - `command.com`.
- Shell zu Python:
  - `python`.
- Shell zu anderen Informatiksystemen:
  - `ssh` (secure shell).
- Shells zu Datenbanken:
  - `mysql`.
- Shells zu Quake, einem Computerspiel.

Der/Das Prompt einer Shell

Bash-Prompt

Shells melden sich mit einem *Prompt*, auch *Eingabeaufforderung* genannt.

Beispiel

Die Unix-Shell Bash meldet sich mit

```
1 humbert@www-madin:~/uebung06>
```

Dies heißt so viel wie »Hallo Benutzer `humbert`! Hier ist die Bash-Shell auf dem Informatiksystem `www-madin`. Du bist gerade im Verzeichnis `~/uebung06` und ich wüsste jetzt gerne, wie es weitergehen soll.«

Der/Das Prompt einer Shell

Python- und MySQL-Prompts

Beispiel

Der Interpreter `python` meldet sich mit

```
1 >>>
```

Dies heißt so viel wie »Hallo Benutzer, hier ist `Python`. Ich sage nicht, was gerade los ist. Teil mir mit, was ich tun soll«.

Beispiel

Das Datenbankmanagementsystem `mysql` meldet sich mit

```
1 mysql>
```

Dies heißt so viel wie »Hallo Benutzer, hier ist das `mysql` Programm. Ich sage nicht, was gerade los ist.«

1.4.2 Unix-Shell-Befehle

Start und Benutzung einer Unix-Shell

- Zum Starten einer Unix-Shell ruft man ein *Terminal-* oder *Konsolenprogramm* auf. Dieses präsentiert ein Fenster, in dem dann eine Standard-Shell startet.
- Als Antwort auf den Prompt gibt man den Namen eines Befehls ein. Jeder Befehl wird durch ein kleines Programm realisiert.
- Dieser wird dann ausgeführt und seine Ausgabe angezeigt.
- Ist der Befehl (das Programm) fertig, so erscheint wieder das Prompt.

Beispiel

```
1 humbert@www-madin:~> ls
2 public_html  tmp  uebung06
3
4 humbert@www-madin:~> pwd
5 /home/humbert
6
7 humbert@www-madin:~> exit
```

Optionen bei Befehlen

- Dem Befehlsnamen folgen bei Unix-Shells durch Leerzeichen getrennte *Optionen* und *Parameterwerte*.
- Meistens werden Optionen mit einem oder zwei einleitenden Minus-Zeichen angedeutet,
- Parameter(werte) werden hingegen einfach so angegeben.
- Welche Optionen und Parameterwerte ein Befehl akzeptiert, muss man wissen oder nachschauen.

Beispiel

```
1 humbert@www-madin:~/uebung06> ls -l test*
2 -rw-r--r-- 1 humbert ddi 408 29. Aug 11:43 test.py
3 -rw-r--r-- 1 humbert ddi 575 29. Aug 11:43 test.pyc
4 humbert@www-madin:~/uebung06>
```

Dateiverwaltung

`cd`, `pwd`, `ls`

Der aktuelle Pfad

- Während man mit einer Unix-Shell arbeitet, gibt es immer ein *aktuelles Verzeichnis*, das in der Regel im Prompt angezeigt wird.
- Alle *relativen Pfade* beziehen sich auf dieses Verzeichnis.

Der `pwd` (print working directory) Befehl

- Der Befehl gibt den aktuellen Pfad aus.

Der `cd` (change directory) Befehl

- Man gibt einen andern Verzeichnisnamen an, dieser wird dann zum neuen aktuellen Verzeichnis.
- Insbesondere wechselt `cd ..` eine Verzeichnisebene nach oben (»oben« bedeutet hier: in Richtung der Wurzel).

Anzeigen des Inhalts eines Verzeichnisses

Der `ls` (list) Befehl

- Er zeigt eine Liste aller Dateinamen im aktuellen Verzeichnis an.
- Der Befehl kennt viele Optionen. Die wichtigste ist `-l`, die dafür sorgt, dass die Anzeige sehr detailliert ist.
- Als *Parameterwert* kann man bestimmte Dateien oder Verzeichnisse angeben. Dann werden Angaben zu diesen Dateien oder der Inhalt dieser Verzeichnisse angezeigt.
- Parameterwerte können auch *Sternchen* enthalten. *Ein Sternchen steht immer für einen beliebigen Text.* So bezeichnet `*.pdf` alle Dateien, die auf »`.pdf`« enden.

`cat, more, less`

Anzeigen des Inhalts einer Datei

Ganze Dateien ausgeben

Der `cat` (concatenate) Befehl

- Der Befehl nimmt die Namen mehrerer Dateien als Parameterwerte und gibt deren Inhalt aneinandergereiht aus.
- Nützlich ist dieser Befehl hauptsächlich in Verbindung mit Um- und Weiterleitung der Ausgabe.

Der `more` Befehl

- Der Befehl zeigt eine Datei seitenweise an.
- Die verbesserte Variante dieses Befehls haben verspielte Informatiker `less` getauft.

`head, tail`

Anzeigen des Inhalts einer Datei

Teile von Dateien ausgeben

Die `head` und `tail` Befehle

- Die Befehle zeigen die ersten oder die letzten Zeilen einer Datei an.
- Mit der Option `-`n, wobei n eine Zahl ist, kann man angeben, wie viele Zeilen man sehen möchte.

Beispiel: `head -5 beispiel.txt`

`rm, cp, mv`

Bearbeiten von Dateien

Der `rm` (remove) Befehl

- Der Befehl löscht die Dateien, die als Parameterwerte übergeben werden, *unwiederbringlich.*
- Er löscht jedoch *keine* Verzeichnisse und
  er löscht *nicht* rekursiv Dateien in Unterverzeichnissen.
- Mit der Option `-R` kann man allerdings erzwingen, dass Dateien doch rekursiv gelöscht werden.

Welchen Effekt haben folgende Befehle?

```
1 cd /
2 rm -R *
```

Kopieren und Umbenennen

Der `cp` (copy) Befehl

- Der Befehl kopiert eine Datei. Parameterwerte sind der alte und der neue Name.
- Es werden keine Verzeichnisse kopiert.

Mit der Option `-R` kann man erreichen, dass rekursiv ein ganzes Verzeichnis kopiert wird.

Der `mv` (move) Befehl

- Dieser Befehl verschiebt Dateien (erster bis vorletzter Parameterwert) an einen neuen Ort (letzter Parameterwert).
- Man kann den Befehl »missbrauchen«, um eine Datei umzubenennen. Dazu ist der erste Parameterwert der alte Dateiname und der zweite Parameterwert der neue Name (der neue »Ort«).

`mkdir, rmdir`

Erstellen und Löschen von Verzeichnissen

Der `mkdir` (make directory) Befehl

- Der Befehl nimmt als Parameterwert den Namen eines neu anzulegenden Verzeichnisses.

Der `rmdir` (remove directory) Befehl

- Der Befehl löscht ein Verzeichnis, das leer sein muss.
- Typischerweise löscht man dazu mittels `rm *` vorher den Inhalt des Verzeichnisses.

Rechteverwaltung

`chmod`

Der Befehl zur Änderung von Rechten

Der `chmod` (change access mode) Befehl

- Dieser Befehl bekommt eine *Rechteänderung* und einen *Dateinamen* als Parameterwert.
- Die Rechteänderung besteht aus
  1. der Personengruppe, um die es geht (also `u`, `g` oder `o`),
  2. einem Minus- oder Pluszeichen (für »Recht wird entzogen« oder »Recht wird gegeben«)
  3. einem Recht (also `r`, `w` oder `x`).

Beispiel: `chmod o-r geheim.txt`

```
1 humbert@abercorn:~/temp$ ls -l
2 insgesamt 0
3 -rwxr----- 1 humbert humbert 0 2014-08-29 11:50 beispiel.txt
```

Die Datei `beispiel.txt` soll »welt-lesbar« werden und das Ausführbarkeitsrecht soll gelöscht werden. Wie lauten die nötigen Befehle?

Nützliche Befehle

echo, wget

Einfach mal etwas sagen

Der echo Befehl

Dieser Befehl gibt einfach alle seine Parameterwerte aus. Dieser Befehl ist hauptsächlich nützlich in der Shell-Programmierung.

Beispiel

```
1 humbert@abercorn:~/temp$ echo Hallo Welt
2 Hallo Welt
3 humbert@abercorn:~/temp$
```

Daten aus dem Internet besorgen

Der wget (network downloader) Befehl

Dieser Befehl bekommt als Parameterwert eine Internetadresse (also z. B. die Adresse einer Webseite auf einem anderen Informatiksystem). Der Inhalt der Datei wird geholt und es wird eine lokale Kopie mit diesem Dateinamen angelegt.

Beispiel

```
1  humbert@abercorn:~/temp$ wget http://ddi.uni-wuppertal.de/index.html
2  --2014-08-29 12:40:27--  http://ddi.uni-wuppertal.de/index.html
3  Auflösen des Hostnamen >>ddi.uni-wuppertal.de (ddi.uni-wuppertal.de)<< ...
       132.195.116.183
4  Verbindungsaufbau zu ddi.uni-wuppertal.de
       (ddi.uni-wuppertal.de)|132.195.116.183|:80... verbunden.
5  HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
6  Länge: 11417 (11K) [text/html]
7  In >>index.html<< speichern.
8
9  100%[==================================================>] 11.417    --.-K/s   in 0,08s
10
11 2014-08-29 12:40:27 (132 KB/s) -- >>index.html<< gespeichert [11417/11417]
12
13 humbert@abercorn:~/temp$ head -3 index.html
14 <!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN"
15    "http://www.w3.org/TR/html4/loose.dtd">
16 <html>
17 ...
18 humbert@abercorn:~/temp$
```

`wc, diff, grep`

Wörter zählen

Der `wc` (word count) Befehl

- Der Befehl bekommt einen Dateinamen als Parameterwert und zählt die Zeichen, die Wörter und die Zeilen in der Datei.
- Mit den Optionen `-c`, `-w` und `-l` kann man entscheiden, was ausgegeben werden soll:

Option	englischer Begriff	deutsche Übersetzung
`-c`	characters	Zeichen
`-w`	words	Wörter
`-l`	lines	Zeilen

Beispiel

```
1 humbert@abercorn:~/temp$ cat beispiel.txt
2 Hallo Welt.
3 humbert@abercorn:~/temp$ wc -c beispiel.txt
4       12 beispiel.txt
5 humbert@abercorn:~/temp$ wc -w beispiel.txt
6        2 beispiel.txt
7 humbert@abercorn:~/temp$ wc -l beispiel.txt
8        1 beispiel.txt
```

Also wissen wir nun – ohne die Datei geöffnet zu haben – dass sie aus 12 Zeichen besteht, die zwei Worte darstellen und eine Zeile umfasst.

Vergleichen und Suchen

Der `diff` (difference) Befehl

- Parameterwerte des Befehls sind zwei Dateien.
- Die Ausgabe sind alle Differenzen zwischen den Dateien in einem menschenlesbaren Format.

Der `grep` Befehl

- Die Abkürzung steht für »global search for a regular expression and print out matched lines«.
- Parameterwerte sind ein regulärer Ausdruck und Dateinamen,

- Ausgabe sind alle Zeilen in den Dateien, in denen der reguläre Ausdruck vorkommt.
- Beispiel: `grep humbert beispiel.txt` – finde alle Zeilen in `beispiel.txt`, in denen `humbert` vorkommt.

1.4.3 Unix-Shell-Programmierung

Um- und Weiterleitung

Umleitung der Ein- und Ausgabe

- Die *Ausgabe* eines Befehls wird normalerweise einfach in der Shell ausgegeben.
- Gibt man hinter einem Befehl `> dateiname` an, so wird die Ausgabe stattdessen in die angegebene Datei geleitet.
- Analog kann man mittels `< dateiname` die *Eingabe* aus einer Datei lesen lassen, statt vom Benutzer.

Beispiel

```
1 humbert@www-madin:~/public_html> ls
2 index.html  index.html~
3 humbert@www-madin:~/public_html> ls > listing
4 humbert@www-madin:~/public_html> cat listing
5 index.html
6 index.html~
7 listing
```

Umleitung der Ein- und Ausgabe

- Man kann die *Ausgabe* eines Programmes an die *Eingabe* eines anderen Programmes leiten.
- Dazu wird eine *Pipe* benutzt, ein senkrechter Strich.

Beispiel

```
1 humbert@www-madin:~/public_html> ls
2 index.html  index.html~
3 humbert@www-madin:~/public_html> ls | wc -w
4 2
```

Beispiel

```
humbert@abercorn:~/IF_GK_EF$ grep Humbert ~/texmf/bibtex/bib/Komplett.bib|head -10
  Ludger Humbert
@inproceedings{BorchelReinertzHumbert2005,
   author    = {Christiane Borchel and Ludger Humbert and Martin
       and Ludger Humbert and Dirk Pommerenke and Detlef
       and Ludger Humbert and Dirk Pommerenke and Detlef
       and Ludger Humbert},
   author    = {Christian F. G{\"o}rlich␣and␣Ludger␣Humbert},
@inproceedings{GoerlichHumbert2002,
␣␣␣author␣␣␣␣=␣{Christian␣F.␣G{\"o}rlich and Ludger Humbert},
@inproceedings{GoerlichHumbert2003,
humbert@abercorn:~/IF_GK_EF$
```

Shell-Skripte

Was sind Shell-Skripte?

- Ein *Shell-Skript* ist eine Folge von Befehlen, die man immer wieder benutzen möchte.
- Man schreibt einfach die Folge der Befehle in eine Textdatei, jeden Befehl in eine neue Zeile.
- Heißt die Datei beispielsweise `myscript.bash`, so kann man das Skript mittels `bash myscript.bash` aufrufen.

Beispiel

Inhalt von `myscript.bash`:

```
echo Das aktuelle Verzeichnis ist
pwd
echo Es enthaelt die folgende Anzahl an Dateien:
ls | wc -w
```

Aufruf des Skriptes:

```
humbert@abercorn:~/Documents$ bash myscript.bash
Das aktuelle Verzeichnis ist
/home/humbert/Documents
Es enthaelt die folgende Anzahl an Dateien:
      16
humbert@abercorn:~/Documents$
```

Parameter und -werte für Shell-Skripte

- Shell-Skripte können *Parameter* mit *Parameterwerten* erhalten.

- Innerhalb des Skriptes wird jedes Vorkommen von `$1` durch den ersten Parameterwert ersetzt.
- Analog wird jedes Vorkommen von `$2` durch den zweiten Parameterwert ersetzt und so weiter.

Beispiel

Inhalt von `backup.bash`:

```
1 echo Erstellung eines Backups von $1
2 cp $1 $1.bak
```

Aufruf des Skriptes:

```
1 humbert@abercorn:~/temp$ ls
2 backup.bash     wichtig.txt
3 humbert@abercorn:~/temp$ bash backup.bash wichtig.txt
4 Erstellung eines Backups von wichtig.txt
5 humbert@abercorn:~/temp$ ls
6 backup.bash     wichtig.txt     wichtig.txt.bak
7 humbert@abercorn:~/temp$ diff wichtig.txt wichtig.txt.bak
8 humbert@abercorn:~/temp$
```

Shells

- Eine *Shell* ist ein Dialogprogramm, das die Kommunikation zwischen einem Nutzer und einer Anwendung ermöglicht.
- Eine *Unix-Shell* dient dazu, mit einem Unix-Betriebssystem zu kommunizieren.
- In dem Dialog gibt die Shell einen *Prompt* vor, der Benutzer gibt einen *Befehl* ein, die Shell zeigt *die Ausgaben des Befehls* an und die Sache beginnt von vorne.

Wichtige Shell-Befehle

Die folgenden Shell-Befehle sollte man kennen: `cd`, `pwd`, `ls`, `cat`, `less`, `head`, `tail`, `rm`, `cp`, `mv`, `chmod`, `echo`, `wget`, `grep`, `wc` und `diff`.

Umleitung

- Schreibt man `> dateiname` hinter einen Shell-Befehl, so werden die Ausgaben des Befehls in die Datei geschrieben.
- Schreibt man `< dateiname` hinter einen Shell-Befehl, so werden alle Eingaben des Befehls aus der Datei gelesen.

- Schreibt man `befehl1 | befehl2`, so dienen die Ausgaben von `befehl1` als Eingaben von `befehl2`.

Shell-Skripte

- Ein Shell-Skript ist eine Datei, in der jede Zeile einen Shell-Befehl enthält.
- Man ruft ein Shell-Skript `foo.bash` auf mittels `bash foo.bash parameterwert1 parameterwert2 ...`.
- Dann werden die Zeilen des Skript nacheinander genau so ausgeführt, als hätte man sie »per Hand« nacheinander eingegeben.
- Wenn im Shell-Skript irgendwo `$1` auftaucht, so wird dieses durch den ersten Parameterwert ersetzt, analog mit `$2` und so weiter.

1.5 Der Algorithmusbegriff

Von der vagen Idee zur Instruktionsfolge[7]

1.5.1 Kompetenzen, die mit der Bearbeitung erreicht werden sollen

- Den Begriff des Algorithmus verstehen
- Einfache Lösungsideen als Algorithmen formulieren können
- Probleme spezifizieren können
- Programmiersprachen als Kommunikationsmittel für Algorithmen begreifen
- Das Konzept des Übersetzers verstehen

[7] Einige Elemente dieses Abschnitts wurden der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zum Thema »Der Algorithmusbegriff« entnommen (dort: Kapitel 8).

1.5.2 Der Begriff, seine Herkunft und der Bezug zum »Leben«

Das Wort »Algorithmus« ist schon ein wenig unheimlich, finden Sie nicht? Für Menschen mit einer Mathematikphobie klingt es irgendwie nach Logarithmus – und mit diesem standen sie schon immer auf Kriegsfuß. Für Technikverliebte klingt es irgendwie griechisch und damit altmodisch und damit uninteressant. Hypochonder hoffen inständig, nie einen Algorithmus zu bekommen, da die Apothekenrundschau noch nie über ein Heilmittel berichtet hat. Selbst einige Informatikstudenten weigern sich, Algorithmen als eigenständigen Wesen die nötige Aufmerksamkeit zu schenken – für sie gibt es Programmcode und alles andere ist von Übel.

Dabei benutzen wir alle Algorithmen Tag ein, Tag aus. Der Grund ist, dass jedem einigermaßen strukturierten Arbeitsablauf ein Algorithmus zu Grunde liegt. Wenn Sie zwei Zahlen schriftlich multiplizieren (was man zugegebenermaßen nicht jeden Tag macht), so gehen Sie ja nicht völlig planlos zu Werke, sondern Sie haben in der Schule mehr oder minder mühselig gelernt, was alles wie in welcher Reihenfolge aufgeschrieben werden muss. Das, was Sie gelernt haben, ist ein Algorithmus, nämlich der »Algorithmus für die schriftliche Multiplikation nach der Schulmethode«. Ein anderes Beispiel ist das Suchen eines Namens in einem Telefonbuch, wo man ja nicht Zeile für Zeile nach dem gesuchten Namen durchgeht, sondern zwischen den Seiten »hin- und herspringt« (wieder zugegebenermaßen kein alltägliches Geschäft heutzutage; was aber hauptsächlich daran liegt, dass wir das Ausführen des Im-Telefonbuch-suchen-Algorithmus heutzutage unserem Telefon überlassen).

Was könnte man tun, um dem Wort »Algorithmus« ein besseres Image zu verpassen? Am einfachsten wäre es wohl, einfach ein anderes, freundlicheres Wort zu benutzen. Dies hat sich in der Geschichte immer wieder bewährt: bei der Bundeswehr ist aus einem schrecklichen »Krieg« der an eine gemütliche, holzkohleofengewärmte Amtsstube erinnernde »Verteidigungsfall« geworden; die mittelalterliche, voraufklärerische »Folter« ist den zackigen »verschärften Verhörmethoden« gewichen; und »Raider« heißt jetzt »Twix« (warum auch immer). Wie wäre es mit »Berechnungsvorschrift«? Zu dirigistisch. Vielleicht »Rechenanleitung«? Schon besser, klingt aber etwas nach einer Mathematiknachhilfestunde. Letzter Vorschlag: »Kochrezepte für Computer«. Da stellt man sich unwillkürlich tomatensoßebespritzte Roboter in einer Küche vor und ist damit schonmal in einer positiven mentalen Grundhaltung. Am Ende bleiben wir dann vielleicht doch einfach bei »Algorithmus«.

Über das Wort »Algorithmus«.

John L. Esposito, Public domain

- Das Wort stammt von Namen des Gelehrten *Muhammad ibn Musa, Abu Dscha'far al-Chwarizmi* (∗ ca. 783, † ca. 850)
- Entscheidend war sein Buch *Al-kitab al-muchtasar fi hisab al-dschabr wa-l-muqabala* (»Rechnen durch Ergänzung und Ausgleich«).

Das erste Programm stammt von einer Frau.

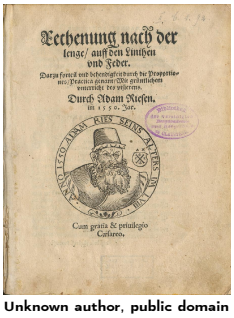
Unkown author, public domain

Aus *de.wikipedia.org/wiki/Algorithmus*

> Der erste für einen Computer gedachte Algorithmus wurde 1842 von Ada LOVELACE, in ihren Notizen zu Charles BABBAGEs Analytical Engine, festgehalten. Sie gilt deshalb als die erste Programmiererin. Weil Charles BABBAGE seine Analytical Engine nicht vollenden konnte, wurde Ada Lovelaces Algorithmus nie darauf implementiert.

Radiobeitrag zu »Ada LOVELACE«, den Sie sich anhören sollten: (Rhia 2012)

Duplieren nach Adam Riese (1574).



Unknown author, public domain

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit
einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Die Definition des Begriffs

Was sind Algorithmen?

- Ein *Algorithmus* ist eine abstrakte Folge von Handlungsanweisungen zur Lösung eines Problems.
- Eher schlechtes Beispiel: *Backrezept*.
- Eher gutes Beispiel: *Anleitung zum schriftlichen Multiplizieren*.

Was sind die Unterschiede zwischen Algorithmen und Programmen?

Eigenschaften von Algorithmen:

- Ein Algorithmus beschreibt *abstrakt*, wie ein Problem zu lösen ist.
- Ein Algorithmus ist *nicht* an einen bestimmten Computer oder eine *bestimmte Programmiersprache* gebunden.
- Derselbe Algorithmus kann oft *in vielen Situationen* benutzt werden.

Eigenschaften von Programmen:

1. Programme sind immer ganz *konkrete* Instruktionsfolgen.
2. Programme werden in einer ganz *bestimmten Programmiersprache* geschrieben.

3. Ein Programm löst *genau ein Problem* und nichts anderes.

Algorithmusbeschreibungen können unterschiedlich »genau« sein.

Die einzelnen Handlungsanweisungen eines Algorithmus können ganz unterschiedlich umfangreich sein:

- Eher elementar:

  »1. Tausche die ersten beiden Zahlen in der Liste.«

  »2. Wenn die erste Zahl größer ist als die zweite, mache bei Schritt 19 weiter.«

- Eher komplex:

  »1. Sortiere die Liste.«

  »2. Entferne das Maximum.«

Zur Übung

Aufgabe 1.11

Ein möglicher Algorithmus zum Sortieren von drei Spielkarten funktioniert wie folgt:

1. Falls die linke Karte größer ist als die mittlere, tausche sie.
2. Falls die mittlere Karte größer ist als die rechte, tausche sie.
3. Falls die linke Karte größer ist als die mittlere, tausche sie.

Schreiben Sie einen Algorithmus zum Sortieren von vier Karten auf. Zusatzfrage: Ist Ihr Algorithmus optimal?

Der Aufbau von Algorithmen – Steuerungsanweisungen

Es gibt immer wiederkehrende Strukturen in Algorithmen.

Die allermeisten Algorithmen benutzen Varianten der folgenden so genannten *Steuerungsanweisungen*:

1. Hintereinanderausführung von Anweisungen.

   Beispiel: Erst tue dies, dann jenes, dann dieses.

2. Bedingte Anweisungen.

   Beispiel: Falls dies gilt, tue jenes, sonst dieses.

3. Schleifen (Zyklen).

   Beispiel: Solange jenes gilt, tue folgendes:

   Beispiel: Tue folgendes 100 mal:

4. Sprünge (evil!).

   Beispiel: Mache nun dort weiter.

Steuerungsanweisungen bei Adam Riese

1. Hintereinanderausführung.
2. Bedingte Anweisungen.
3. Schleifen (Zyklen).
4. Sprünge.

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

1.5.3 Spezifikationen

Spezifikationen sind Problembeschreibungen.

- Probleme lassen sich nur dann sinnvoll lösen, wenn das Problem klar gestellt ist.
- Eine »klare Problemstellung« nennt man eine *Spezifikation*.
- Sie sollte die folgende Fragen umfassend beantworten:
  1. Was sind die relevanten Rahmenbedingungen?
  2. Welche Hilfsmittel und Basisoperationen sind zugelassen?
  3. Wann ist eine Lösung korrekt oder zumindest akzeptabel?

Gute Spezifikationen zu erstellen ist schwierig.

- Spezifikationen dienen oft dazu, zu klären, was eine zu erstellende Software später leisten soll. Leider wissen Kunden aber nicht, was sie eigentlich wollen.
- Selbst Spezifikationen zu einfachsten Problemen lassen oft Fragen offen.

  Beispiel: Berechne die Summe der ersten n Zahlen. Als elementare Operationen stehen Additionen und Vergleiche zur Verfügung.
- Größe Spezifikationen sind in aller Regel in sich widersprüchlich.

  Beispiel: Auf Seite 50 der Spezifikation steht, dass beim Drücken des ersten Knopfes der Text rot wird. Auf Seite 150 derselben Spezifikation steht dann, dass beim Drücken dieses Knopfes der Text grün werden soll.

1.6 Programmiersprachen

1.6.1 Wozu dienen Programmiersprachen?

Von der Idee zur Instruktionsfolge.

Spezifikation
Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Idee
Verdopple die Zahl stellenweise von rechts nach links und beachte den Übertrag.

Algorithmus (Notation von Riese)

Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus

Algorithmus (modernere Notation)

Für jede Ziffer, beginnend mit der letzten, tue folgendes:

1. Verdopple die Ziffer.
2. Addiere einen eventuell vorhandenen Übertrag hinzu.
3. Schreibe die letzte Ziffer der Summe unter die gerade behandelte Ziffer.
4. Merke die erste Ziffer des Übertrags.

Falls noch ein Übertrag vorhanden ist, schreibe diesen links von allem.

Algorithmus (Pseudocode)

```
input Ziffern z_1, ..., z_n
c ← 0
for i ← n, ..., 1 do
  d ← 2z_i + c
  z'_i ← d mod 10
  c ← ⌊d/10⌋
z'_0 ← c
for i ← 0, ..., n do
  output z'_i
```

Algorithmus (Python3)

```
def verdopple(ziffern, n):
  ziffern_verdoppelt= [0]*(n+1)
  uebertrag= 0
  for i in range(n, 0, -1):
    d= 2*ziffern[i-1] + uebertrag
```

```
6     ziffern_verdoppelt[i]= d % 10
7     uebertrag= d // 10
8
9   ziffern_verdoppelt[0]= uebertrag
10  for i in range(0, n+1):
11    print (ziffern_verdoppelt[i], end="")
```

Vorhaben 2

Grundlagen der objektorientierten Analyse und Modellierung anhand von Beispielkontexten

2.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (IF1, M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (IF1, M),
- nutzen Railroad-Diagramme, um syntaktisch korrekte Strukturen zu entwickeln und zu prüfen (IF1, I),
- erstellen syntaktisch korrekte Bezeichner für Objekte, Attribute und Methoden (IF3, I),
- stellen den Zustand eines Objekts dar – Objektkarte (IF1, D),
- modellieren Objekte mit ihren Attributen, Attributwerten, Methoden und Beziehungen (IF1, M),
- stellen die Kommunikation zwischen Objekten grafisch dar (IF1, M),
- stellen die Ergebnisse der Modellierungsüberlegungen der objektorientierten Analyse grafisch dar – Objektkarten und Objektdiagramme (IF1, D),
- modellieren die Kommunikation zwischen Objekten (IF1, M),
- stellen die Ergebnisse der Modellierungsüberlegungen zum Ablauf der Kommunikation der Objekte grafisch dar – Sequenzdiagramme (IF1, D),
- setzen Sequenzdiagramme in die Punktnotation um (IF1, I),
- analysieren und erläutern eine objektorientierte Modellierung (IF1, A).

2.2 Informatische Modellierung

2.2.1 Modell und Modellierung

Bei dem Begriff Modellierung fällt uns zunächst sicher nicht Informatik ein, sondern eher z. B. Kunst, man sieht förmlich eine Töpferin, die aus einem Klumpen Ton auf einer Drehscheibe in eine schöne Vase formt. Beim Begriff Modell denken wir vielleicht an eine »schöne Frau« oder einen »schönen Mann«. Es kann auch sein, dass wir zunächst an ein Finanzmodell, an ein Modell für den Wasserkreislauf oder ein Atommodell denken.

Eins ist allen Modellen gemein: sie stellen etwas – unter Vernachlässigung gewisser Aspekte – dar, was für den jeweiligen Zusammenhang in den Vordergrund gerückt wird, um es zu verdeutlichen. Damit entstehen Modelle einerseits durch Weglassen »uninteressanter Details« und andererseits durch die Fokussierung auf die Punkte, die hervorgehoben werden sollen. Jedes Modell hat einen Zweck und durch das Modell selbst wird etwas so beschrieben/dargestellt, dass der Zweck klar hervortritt.

Bei der Töpferin ist es die Idee, aus der eine Vase wird, d. h. die Vase ist das Modell der Idee. Die Modelle des zweiten Beispiels stellen Exemplare der Gattung Mensch dar, die unter dem Gesichtspunkt »schön erscheinen« als Modell für die Vorstellung von Schönheit zu einer bestimmten Zeit gelten. Ein Finanzmodell ist die abstrakte Darstellung einer finanziellen Entscheidungssituation. Bei den beiden Modellen für den Wasserkreislauf und das Atom wird der beschreibende Charakter zur Erklärung deutlich. Man sieht dem Wassertropfen, der vom Himmel fällt, den Wasserkreislauf nicht an, also erstellt man ein Modell, durch das deutlich wird, was mit dem Wassertropfen vorher war und was anschließend in idealisierter/abstrakter Weise mit ihm geschehen wird. Beim Atommodell ist die Situation so, dass der Blick auf die atomare Ebene nicht direkt dazu führt, dass für das Zusammenwirken wichtige Punkte erkannt werden können, also bedient man sich des Modells, das für die Erklärung und Erläuterung bestimmter Zusammenhänge nützlich sein soll.

2.2.2 Modellierung und Informatik

Die *Informatische Modellierung* nimmt hier keine gänzlich neue Rolle ein: ein Weltausschnitt wird – unter Vernachlässigung gewisser Aspekte – so beschrieben, dass man sich auf den Zweck der Modellierung konzentriert, aber nur die wichtigen und nötigen Aspekte des Ausschnitts berücksichtigt. Grundlegender Zweck der informatischen Modellierung besteht darin, durch die Verarbeitung von Daten mit einem automatischen Verfahren, einen Teil des Weltausschnitts zu automatisieren. Damit führt die informatische Modellierung zu einem Programm, das auf einem Informatiksystem zum Ablauf gebracht wird. Damit gehört das Programmieren und Testen zur informatischen Modellierung.

Die Abbildung 2.1 verdeutlicht den Kreisprozess der informatischen Modellierung. Ausgehend von einer realweltlichen Situation wird ein informatisches Modell entwickelt, das durch die Verarbeitung Konsequenzen zeigt, die interpretiert werden und Ergebnisse für die Realwelt haben. Durch die Überprüfung der Ergebnisse ergibt sich eine geänderte Situation, die wiederum in ein Modell überführt werden kann …

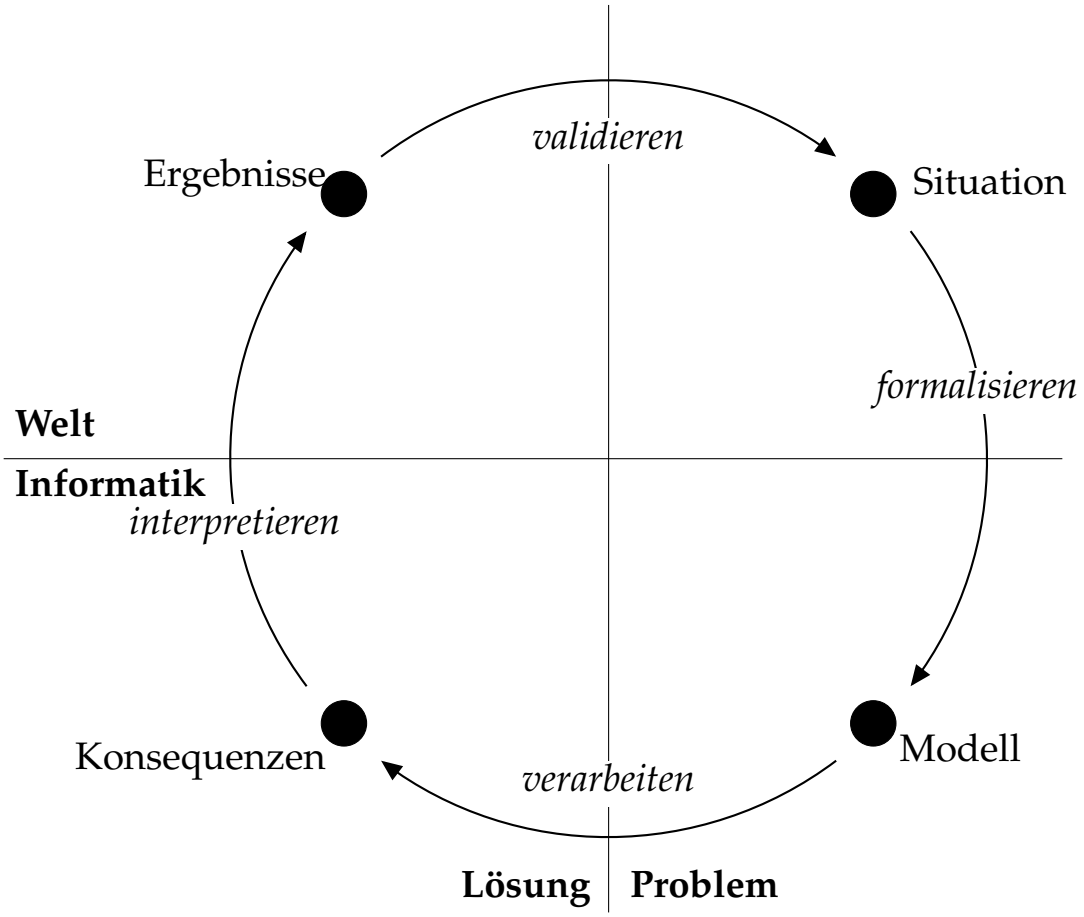


Abbildung 2.1: Informatische Modellierung – der Modellierungskreis

2.2.3 Arten der Informatischen Modellierung

Über die Zeit wurden verschiedene Techniken entwickelt, um die informatische Modellierung durchzuführen. Die in der Geschichte der Informatik frühen Techniken sind die **funktionale** und die **wissensbasierte** Modellierung, ergänzt um die ***imperative*** Modellierung, die – nahe an der technischen Informatik – dem Prozessor vorgibt, was er in welcher Reihenfolge zu tun hat, woher die Daten kommen und was mit den Daten nach der Verarbeitung zu geschehen hat.

Seit Beginn der 90er Jahre des letzten Jahrhunderts wird die **objektorientierte Modellierung** durchgesetzt. Sie stellt als begrifflichen Schlüssel zur Betrachtung des Weltausschnitts das Werkzeug *Objekt* bereit. Ein Objekt vereinigt die beiden oben angegebenen Punkte: es hat Eigenschaften (Daten als Attributwerte) und kann »gewisse Dinge« tun (ausführen). Zu Beispiel kann ein Objekt, wenn es »gefragt« wird, eine Antwort liefern, wenn es »beauftragt« wird, eine bestimmte Aufgabe erledigen. Die Fragesituation wird als »Anfrage« bezeichnet, die Aufforderung, etwas zu erledigen, nennt man »Auftrag«.

2.3 OOM – Vorgehensweise

2.3.1 Das Verfahren von Abbott

Zu Beginn der Modellierung steht eine Situationsbeschreibung. Diese dokumentiert einen Ablauf. Ihre Aufgabe besteht darin, mit Hilfe einer Heuristik – genauer dem »**Verfahren von Abbott**« (in der Anlage das **Rezept C.3, S. 114–115**) – aus einer Beschreibung die zur Lösung nötigen Objekte herauszupräparieren. Das Verfahren liefert eine Grundlage zur Identifikation der Objekte.

Aufgabe 2.1

Lesen Sie sich das Rezept C.3 zunächst komplett und in Ruhe durch – machen Sie sich Notizen, damit Sie, **ohne nachzusehen**, die Punkte mit eigenen Worten wiedergeben können.

Situationsbeschreibung

Alice und Bob befinden sich mit Claire in ihrem grünen Zimmer und sprechen miteinander über Liebe. Pauline kommt aus dem hellen Wohnzimmer dazu und fragt: »Worüber habt ihr gerade gelästert?«

Bearbeiten Sie die folgenden Punkte:

1. Identifizieren Sie alle Objekte in der Situationsbeschreibung.
2. Geben Sie **syntaktisch korrekte** Bezeichner für die identifizierten Objekte an. H
3. Welche Aktionen können Sie identifizieren? Beschreiben Sie und geben Sie **syntaktisch korrekte** Bezeichner für die identifizierten Aktionen an.

4. Nachfolgend präparieren Sie die für die identifizierten Objekte die im dritten Punkt des Verfahrens von ABBOTT angegebenen Elemente heraus, geben Sie **syntaktisch korrekte** Bezeichner an und notieren Sie die Werte.
5. Fassen Sie die Ergebnisse der vorgängigen Punkte für jedes Objekt auf einer **Objektkarte** zusammen. H

2.3.2 Objektdiagramm – Objektspiel

Die identifizierten Kandidaten für Objekte sollen eine Aufgabe erledigen. Dies geschieht dadurch, dass die Objekte miteinander kommunizieren (interagieren). Damit ein Objekt a einem anderen Objekt b einen Auftrag erteilen kann oder ein anderes Objekt b etwas fragen kann (Anfrage), brauchen wir eine Verbindung von dem Objekt a zu dem Objekt b. Die Verbindungen werden als Objektbeziehungen bezeichnet (oder einfach: Beziehungen).

Wir nehmen dazu die Objektkarten und legen sie vor uns aus und versuchen, die Geschichte, das Szenario, die Problemlösung »durchzuspielen«. Dabei erkennen wir die Objektbeziehungen, die sich im Verlauf des Durchspielens auch ändern können.

Diese Beziehungen werden in einem sogenannten Objektdiagramm dargestellt. Die Objektkarten erhalten dazu weitere Attribute – die Beziehungsattribute. Diese enthalten Verbindungen zu dem Bezugsobjekt, das zur Durchführung/Abarbeitung der Anfrage/des Auftrags benötigt wird.

In Anhang C.5 (Seite 116) wird an einem Beispiel diese Form der Darstellung gezeigt.

Vorhaben 3

Datenschutz aus informatischer Perspektive

3.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

- führen Handlungsanweisungen im Kontext von Rollenbeschreibungen aus (IF2, D),
- werten aggregierte Datensammlungen interessensgeleitet und zielgerichtet aus (IF1, K),
- bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (IF5, A),
- nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und **verantwortungsbewusst** (IF4, D),
- nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (IF5, K).

3.2 Was ist eigentlich Datenschutz?

Der Begriff **Datenschutz** ist eigentlich eine falsche Bezeichnung, denn nicht die Daten, sondern die Person(en) müssen geschützt werden.

Im **Grundgesetz der Bundesrepublik Deutschland** wird in Artikel 1 Absatz 1 festgestellt: »Die Würde des Menschen ist unantastbar. Sie zu achten und zu schützen ist Verpflichtung aller staatlichen Gewalt«.

Artikel 2 Absatz 1 lautet: »Jeder hat das Recht auf die freie Entfaltung seiner Persönlichkeit, soweit er nicht die Rechte anderer verletzt und nicht gegen die verfassungsmäßige Ordnung oder das Sittengesetz verstößt«. Dieser Absatz wird als **Allgemeines Persönlichkeitsrecht** bezeichnet.

. . . . . .

Im Zusammenhang mit dem sogenannten **Volkszählungsurteil** von 1983 hat das **Bundesverfassungsgericht** dem **Recht auf informationelle Selbstbestimmung** als Ausprägung des allgemeinen Persönlichkeitsrechts den Rang eines Grundrechts zugesprochen.

. . . . . .

Einige Landesverfassungen (wie die **Verfassung des Landes Nordrhein-Westfalen** im Jahr 1978) wurden so erweitert, dass Datenschutz Grundrechtcharakter hat. Im Artikel 4 Absatz 2 der **Landesverfassung Nordrhein-Westfalen** heißt es: »Jeder hat Anspruch auf Schutz seiner personenbezogenen Daten. Eingriffe sind nur in überwiegendem Interesse der Allgemeinheit auf Grund eines Gesetzes zulässig.«

. . . . . .

Die **Charta der Grundrechte der Europäischen Union** enthält einen eigenen Artikel zum Datenschutz:

Artikel 8 – Schutz personenbezogener Daten

1. Jede Person hat das Recht auf Schutz der sie betreffenden personenbezogenen Daten.
2. Diese Daten dürfen nur nach Treu und Glauben für festgelegte Zwecke und mit Einwilligung der betroffenen Person oder auf einer sonstigen gesetzlich geregelten legitimen Grundlage verarbeitet werden. Jede Person hat das Recht, Auskunft über die sie betreffenden erhobenen Daten zu erhalten und die Berichtigung der Daten zu erwirken.
3. Die Einhaltung dieser Vorschriften wird von einer unabhängigen Stelle überwacht.

3.3 Planspiel zum Datenschutz

Das Planspiel wird an fünf Stationen durchgeführt, an denen die Teilnehmer ihre Aufgaben erledigen. Jede Station führt (mindestens) eine Liste, um die Aktionen aufzuzeichnen, die dort durchgeführt werden.

Jede Teilnehmerin und jeder Teilnehmer erhält eine Rollenkarte, auf der angegeben ist, was in dieser Rolle zu tun ist. Die Teilnehmenden haben damit zwei Rollen – einerseits betreuen sie eine der Stationen und andererseits führen sie aus, was auf der individuellen Rollenkarte angegeben ist – daher werden die Stationen möglichst mit zwei Personen besetzt. Über die Rollenbeschreibungen darf nicht gesprochen werden, damit die im zweiten Teil des Planspiels nötige Bearbeitung der Vorfälle nicht durch Vorwissen zu Ergebnissen führt, die nur durch Kenntnis einzelner Rollendetails gewonnen wurden. Vielmehr ist es so, dass die Vorfälle einzig mit Hilfe der Daten geklärt werden, die an den einzelnen Stationen »rumliegen«.

Abbildung 3.1: Bezeichnungen für die Stationen

Die Vorfälle sind so gestaltet, dass völlig verschiedene Situationen mit Hilfe der überall vorhandenen Aufzeichnungen (= Daten) bearbeitet werden. Hier tritt das Problem auf, dass die Daten im Sinne der Vorfälle interpretiert werden müssen, damit die Vorfälle geklärt werden können. Diese Interpretation stellt die eigentliche Herausforderung dar, da es naheliegende Interpreationen gibt, die allerdings durchaus falsch sein können.

Insgesamt kommt dieses Planspiel ohne den Einsatz von Informatiksystemen aus. Der Vorteil der papiergebundenen Dokumentation der Ereignisse liegt darin, dass es auch möglich ist, die Einträge »von« Hand vorzunehmen, auszuwerten und damit transparent zu interpretieren.

So zeigt sich, dass bei einem Vorfall regelmäßig ein Verdächtiger gefunden wird, der nicht der Täter ist – dies ist der Tatsache geschuldet, dass nicht jede Aktion zwangsläufig eine Datenspur hinterläßt und umgekehrt nicht jede Datenspur, die im Kontext plausibel erscheint, zu einem Kriminalfall gehört. Hier geht es um die anlassfreie Sammlung/Speicherung von Daten, die dann im Rahmen der »Rasterfahndung« reinterpretiert werden.

Vorhaben 4

Strukturierung und Organisation von Daten

4.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

- analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (IF2, D)
- entwerfen einen weiteren Algorithmus zum Sortieren (IF2, M)
- beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeitaufwand und Speicherplatzbedarf (IF2, A)
- testen Programme schrittweise anhand von Beispielen (IF2, I)
- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (IF1, M)

4.2 Lineares und binäres Suchen in einer Liste

Um in Datenbeständen zu suchen, wählen wir häufig die Strategie, bei dem ersten Element zu beginnen, nachzusehen, ob es das passende ist und dann fortlaufend immer das nächste, folgende Element zu betrachten und dort nachzusehen.

Diese Suche nennt man **lineare Suche**. Wenn man »Glück« hat, ist es direkt das erste Element und man ist schon fertig. Hat man »Pech«, so muss man alle Elemente vergleichen und das gesuchte Element ist nicht in der Liste vorhanden. Der »Glücksfall« wird in der Informatik als **best case** bezeichnet, während der ungünstigste Fall als **worst case** bezeichnet wird.

4.2.1 Lineares Suchen: Aufwand – am Beispiel – allgemein

Beispiel

Schauen wir uns eine Liste mit 35 Elementen an, so müssen wir im besten Fall – **best case** genau einmal vergleichen und im schlimmsten Fall – **worst case** alle Elemente einmal ansehen, um festzustellen, dass nach 35 Fragen herauskommt, dass es nicht drin ist (oder es das letzte Element ist). Aus der Beobachtung kann abgeleitet werden, dass im mittleren Fall – **average case** die Hälfte der Elemente geprüft werden muss – in dem Fall mit 35 Elementen also ungefähr 18 Vergleiche nötig sind, um ein Ergebnis zu erhalten.

Lineare Suche – Liste mit n Elementen – Anzahl der Vergleiche

best case 1 Vergleich

worst case n Vergleiche

averarge case $\frac{n}{2}$ Vergleiche

Schnell kommt einem die Idee, dass es sinnvoll sein könnte, eine Liste, in der man **häufig** nachsehen muss, zu sortieren. Der Suchvorgang bei einer sortierten Liste kann so gestaltet werden, dass man (mit Mittel) viel schneller weiß, ob das gesuchte Element nun in der Liste steht oder nicht.

4.2.2 Binäres Suchen: Aufwand – am Beispiel – allgemein

Wir kennen sortierte Sammlungen, in denen wir schnell suchen und finden können: ob Wörterbuch, Lexikon oder Listen von Personen, häufig haben wir es mit wiederholten Suchvorgängen in solchen Sammlungen zu tun. Dann können wir – im Unterschied zu dem beim Linearen Suchen verwendeten Verfahren – schneller suchen.

Nach einigen Beispielen findet man heraus, dass es sinnvoll ist, zunächst in der Mitte der Liste mit den sortierten Einträgen nachzusehen, ob das gesuchte Element vor oder nach dieser Stelle steht. Steht es vor/nach dieser Stelle, spielt die andere Hälfte keine Rolle mehr. Bei der »richtigen Seite« wendet man das Verfahren erneut an.

Beispiel

Um das Verfahren konkret durchzuführen, gehen wir von einer **sortierten** Liste mit 1.000 Einträge aus und versuchen, herauszufinden, wieviele Fragen wir **maximal** stellen müssen, wir betrachten also den **worst case**.

1. Frage: 500
2. Frage: 250
3. Frage: 125
4. Frage: 63
5. Frage: 32
6. Frage: 16
7. Frage: 8
8. Frage: 4
9. Frage: 2
10. Frage: 1 → fertig

Jede Frage ist eine Aktion, bei der zwei Werte miteinander verglichen werden müssen. Daher zählen wir diese Fragen. Die Anzahl der Fragen ist der **Aufwand**, den wir leisten müssen, um ein Element in der sortierten Liste zu finden. Verallgemeinern wir das Beispiel von 1.000 auf n Elemente, so suchen wir die natürliche Zahl k, für die die Ungleichung $2^k \geq n$ erfüllt ist.

Im Beispiel – mit $n = 1.000$ – ergibt sich $k = 10$, weil $2^{10} = 1.024$ und damit $2^{10} \geq 1.000$ erfüllt ist.

Die Zahl k gibt die Anzahl der Fragen an, die wir höchstens stellen müssen, um herauszufinden, ob sich unser Element in der sortierten Liste befindet.

Binäre Suche – Liste mit n Elementen – Anzahl der Vergleiche

best case 1 Vergleich

worst case $\log_2 n$ Vergleiche

averarge case weniger als der **worst case**

4.3 Die Landau-Notation – das Symbol $\mathcal{O}$

Weil in der Informatik oft der Aufwand für die Durchführung eines Verfahren nicht detailliert angegeben werden soll oder kann, haben die Wissenschaftler entschieden, für den ungefähren Aufwand ein eigenes Symobl zu verwenden: das $\mathcal{O}$. Dieses große O stellt die Abkürzung für **Ordnung** dar – gemeint ist die Größenordnung, die für die Durchführung gilt. Das Symbol $\mathcal{O}$ nennt man im deutschen Sprachraum **Landau-Symbol**. Da das Symbol groß geschrieben wird, bezeichnet man beim Sprechen die Schreibweise mit **Groß-O**.

Wenn wir den (Zeit-)Aufwand für die Durchführung des linearen Suchens mit Hilfe der $\mathcal{O}$-Notation angeben, würden wir für den **worst case** und für den **average case** schreiben: $\mathcal{O}(n)$.

Bei der **binären Suche** hingegen können wir für diese beiden Fälle $\mathcal{O}(\log n)$ angeben.[1]

[1] Beachte, dass wir **nicht** $\mathcal{O}(\log_2 n)$ schreiben, weil die Basis 2 für die Größenordnung keine Rolle spielt.

4.4 Sortieren einer Liste von Elementen

Motivation – warum sollte man manchmal aufräumen?

Sortieren – nichts leichter als das! Das können wir doch und haben wir alle schon gemacht. Leider ist der Schreibtisch trotzdem nicht aufgeräumt und beim Suchen müssen wir doch alles der Reihe nach durchsuchen. Das nennt man – wie wir jetzt wissen – **lineares Suchen**. Dass der Aufwand extrem hoch sein kann, merken wir vor allem, wenn wir etwas **nicht finden** – der **worst case**; dann nämlich müssen wir alle Elemente prüfen. Sind also n Elemente auf dem Schreibtisch, so ist der Aufwand proportional n – man schreibt dann normalerweise auch $\sim n$, wie in den Naturwissenschaften üblich. Verwenden wir die $\mathcal{O}$-Notation, können wir stattdessen $\mathcal{O}(n)$ schreiben.

Warum räumen wir unseren Schreibtisch denn nicht auf? Dann würden wir doch alles schneller finden – etwa in der Größenordnung $\mathcal{O}(\log n)$.

Wir räumen nicht auf, weil wir keine Zeit dafür haben. Zeit bedeutet Aufwand – in der Informatik spricht man von Zeitkomplexität. Vielleicht haben wir auch keinen (oder zuwenig) Platz, um Sachen, die wir erst vom Schreibtisch runter nehmen müssen, ablegen zu können – dabei geht es um die sogenannten Speicherplatzkomplexität.

Wir müssen also überlegen, ob die Zeitersparnis, die wir beim Suchen haben (wenn wir oft suchen müssen, kommt da ja durchaus einiges zusammen) durch den Aufwand für das Sortieren ausgeglichen wird. Als Grundregel kann man sagen: Muss man häufiger als $\log n$ mal auf seinem Schreibtisch suchen, sollte man sortieren.

Wie sortieren Menschen?

Als Menschen können wir eine gewisse Anzahl (7 ± 2) von Elementen, bei denen wir die Ordnungseigenschaft »sehen«, fast direkt richtig anordnen (vgl. Miller 1956). Damit sind wir in der Lage, die Spielkarten, die wir in die Hand nehmen, recht zügig so anzuordnen, wie es für das jeweilige Spiel sinnvoll ist – z. B. nach dem Kartenwert für das Spiel.

Stellen Sie sich mal ein Kind vor, bei dem nicht alle Karten direkt auf die Hand passen, weil die Hände noch so klein sind[2]. Beobachten Sie das Kind, wenn es die Karten steckt (z. B. hintereinander nach der Größe geordnet). Bei dem Kind funktioniert die Idee mit dem *Überblick* über viele Elemente nicht – daher muss es eine andere Strategie überlegen, wenn die Karten sortiert sein sollen.

[2] »Sind so kleine Hände« (Titel eines Stücks auf der LP: Kinder Wegner 1978)

Vorhaben 5

Klassen und ihre Implementierung durch Umsetzung einer objektorientierten Modellierung

5.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

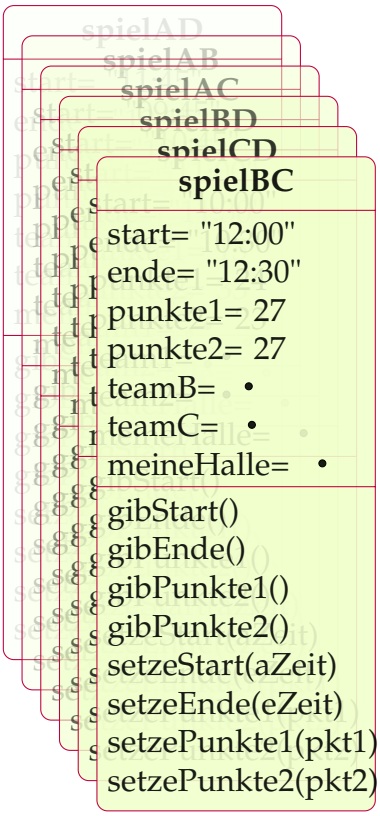
- ermitteln gemeinsame Attribute und Methoden verschiedener Objekte (IF1, M),
- klassifizieren Objekte nach Attributen und Methoden (IF1, M),
- identifizieren die Art von Attributwerten verschiedener Objekte und geben die Art fachgerecht an als Zeichenkette, Zahl, Wahrheitswert, Objekt, Sammlung von Objekten (IF1, M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (IF1, M),
- modellieren Klassen unter Verwendung von Vererbung (IF1, M),
- stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (IF1, D),
- dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (IF1, D),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (IF1, I).

5.2 Von Objekten zur Klasse

Betrachen Sie die im Zusammenhang mit der objektorientierten Modellierung des Basketballturniers identifizierten Objekte, so stellen Sie fest, dass es möglich ist, die Objektkarten nach Ähnlichkeiten zu ordnen.

Führen Sie dies für die Objekte durch, die die einzelnen Spiele repräsentieren, ergibt sich ein Stapel von Objekten:

Beispiel



BasketballSpiel

start: Zeichenkette
ende: Zeichenkette
punkte1: Zahl
punkte2: Zahl
teamA: Mannschaft
teamB: Mannschaft
meineHalle: Sporthalle

gibStart(): Zeichenkette
gibEnde(): Zeichenkette
gibPunkte1(): Zahl
gibPunkte2(): Zahl
setzeStart(aZeit: Zeichenkette)
setzeEnde(eZeit: Zeichenkette)
setzePunkte1(pkt1: Zahl)
setzePunkte2(pkt2: Zahl)

Dieser Vorgang stellt die erste Vorstufe zur Klassifikation von Objekten dar. Ziel ist es, für diese Objekte eine Abstraktion in Form einer sogenannten Klasse zu finden. Es fällt auf, dass alle oben dargestellten Objekte dieselben Attributbezeichner und dieselben Methodenbezeichner haben. Unterschiede bestehen lediglich bei den Objektebezeichnern und bei den Attributwerten.

Eine Klasse stellt eine Abstraktion von ähnlichen Objekten dar. Die Klasse muss daher alle Attributbezeichner und alle Methodenbezeichner enthalten, die bei den zugrunde liegenden Objekten vorhanden sind. Allerdings enthält eine Klasse nicht die Attributwerte, die die Objekte enthalten, sondern hinter den Attributbezeichnern wird – abgetrennt durch einen Doppelpunkt – angegeben, von welcher Art (von welchem Typ) die Attributwerte sind/sein sollen/sein dürfen.

Um eine Klasse grafisch darzustellen, wird ein Rechteck (ohne abgerundete Ecken) verwendet. Der Bezeichner für eine Klasse beginnt (bei uns) immer mit einem Großbuchstaben (im Unterschied zu Objektbezeichnern).

Vorhaben 6

Kontrollstrukturen zur Steuerung des Ablaufs von Algorithmen

6.1 Welche Kompetenzen sollen Sie in diesem Vorhaben erwerben?

Die Schülerinnen und Schüler

- analysieren und erläutern einfache Algorithmen und Programme (IF2, A)
- modifizieren einfache Algorithmen und Programme (IF2, I)
- entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (IF2, M)
- implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (IF2, I)
- ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (IF1, M),
- testen Programme schrittweise anhand von Beispielen (IF2, I)
- implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (IF3, I)
- interpretieren Fehlermeldungen und korrigieren den Quellcode (IF3, I)

Im Anhang B.6 ab S. 108 haben wir die Arbeitsmaterialien dokumentiert, die im Kurs eingesetzt wurden.

Anhang A

Hinweise

0.1 c Je nach System ist dieses Detail an einer anderen Stelle zu finden – in der Regel gibt es eine Funktionalität, die Ihnen Auskünfte »Über das System« zur Verfügung stellt. Zurück

1.1 1.2.1 Eine Übersicht zu den Fachgebieten findet sich in der Abbildung 1.2. Zurück

1.2 1

- Die Regeln dürfen in beliebiger Reihenfolge angewendet werden
- Notieren Sie die Folge der Umformungen in der Form $mi \overset{2}{\vdash} mii \overset{\ldots}{\vdash} \ldots$ die über dem Pfeil stehende Zahl bezeichnet dabei die Nummer der Regel, die Sie anwenden.

Zurück

1.6 12 Berücksichtigen Sie, was "../" bewirkt. Zurück

1.7 13 Berücksichtigen Sie zunächst, wo Sie sich im Verzeichnisbaum befinden. Dann schauen Sie, wohin Sie durch "../../" geraten. Der Rest sollte dann (wie bei der vorgängigen Aufgabe) klar werden. Zurück

1.8 18 Schauen Sie genau nach, ob die Rechte der Gruppe für die jeweilige Person gelten, wenn nicht, gelten die Rechte für »alle anderen Benutzer«. Zurück

2.1 2 Unter B.2, S. 82 finden Sie Hinweise zur Erstellung und Prüfung von sytaktisch korrekten Bezeichern (identifier). Um zu prüfen, ob Sie in der Lage sind, syntaktisch korrekte Bezeichner zu identifizieren, bearbeiten Sie die Aufgaben auf S. 84. Zurück

2.1 5 Ein Beispiel für eine Objektkarte findet sich auf Seite 115. Zurück

Anhang B

Arbeits-, Informationsblätter und Lernzielkontrollen

B.1 Vorhaben EF-1

B.1.1 Definition – Was ist Informatik?

Definition von Informatik

Beachten Sie
Nach dem Ende der Gruppenarbeit muss jede/jeder aus Ihrer Gruppe das **Gruppenergebnis** vorstellen können. Für die komplette Arbeit in der Gruppe ist eine Arbeitszeit von 15 Minuten vorgesehen. Es ist wichtig, dass Sie sich bei der Bearbeitung der Aufgaben in der Gruppe einigen.

Arbeitsaufträge/Aufgaben

1. Einigen Sie sich in der Gruppe auf 5–10 Begriffe, die die Wissenschaft **Informatik** charakterisieren

a)	f)
b)	g)
c)	h)
d)	i)
e)	j)

2. Erarbeiten Sie ein Definition des Begriffs Informatik:

   Definition: Informatik ist die Wissenschaft, die …

3. Prüfen Sie, ob Sie Ihre Begriffe nach dieser Erarbeitung noch einmal ändern müssen. Vor allem sollten Sie die Begriffe aus der obigen Liste entfernen, die nicht gut zu Ihrer Definition passen. Geben Sie hier Ihre modifizierte Liste aus 1 an:

   a)
   b)
   c)
   d)
   e)
   f)
   g)
   h)
   i)
   j)

. . . . . .

Fachgebiete der Informatik

Kurzdarstellung – Fachgebiete der Informatik

Theoretische Informatik formale Methoden, mathematische Modelle für Algorithmen, z. B. Berechenbarkeit, Kryptographie, Komplexitätstheorie

Praktische Informatik Methoden, um Informatiksysteme zu entwickeln, z. B. Programmiersprachen, Betriebssysteme, Übersetzer

Technische Informatik Funktioneller Aufbau des Computers, z. B. Rechnerarchitektur, Rechnernetze

Angewandte Informatik Einsatz in anderen Gebieten, z. B. Medizin, Biologie, Büroarbeit bzw. Organisation

Didaktik der Informatik, Informatik und Gesellschaft

Über die Fachgebiete hinaus gibt es die sogenannten Bindestrich-Informatiken: Bioinformatik, Bauinformatik, Medieninformatik, Wirtschaftsinformatik, Sportinformatik etc.

. . . . . .

Etwas ausführlichere Darstellung – Fachgebiete der Informatik

Was das Wesen der Informatik ausmacht, wird deutlich, wenn man sich damit befasst, mit welchen Inhalten sich die Fachwissenschaft in ihren Teildisziplinen beschäftigt. An dieser Stelle werden fünf Fachgebiete genannt, wie sie im »Duden Informatik« aufgeführt werden.

Theoretische Informatik Sowohl für die Formulierung und Untersuchung von Algorithmen als auch für die Rechnerkonstruktion spielen formale Methoden und mathematische Modelle eine wesentliche Rolle.

Praktische Informatik Die praktische Informatik entwickelt Methoden, um Programmsysteme erstellen zu können, sowie konkrete Entwicklungsumgebungen und Softwarewerkzeuge zur Unterstützung von Programmierern und Anwendern.

Technische Informatik In der technischen Informatik befasst man sich mit dem funktionellen Aufbau von Computern und den zugehörigen Geräten sowie mit dem logischen Entwurf von Rechnern, Geräten und Schaltungen.

Angewandte Informatik Unter angewandter Informatik fasst man Anwendungen von Methoden der Kerninformatik in anderen Wissenschaften und dabei entstehende spezielle Erkenntnisse und Techniken zusammen. Die angewandte Informatik untersucht Abläufe in den unterschiedlichsten Bereichen auf ihre Automatisierbarkeit.

Gesellschaft und Informatik Ein relativ neuer Aspekt der Informatik, der sich zugleich mit dem großen Bereich der »Technikfolgen-Abschätzung« auseinander setzt, wird durch den Begriff »Informatik und Gesellschaft« ausgedrückt. Dieser Bereich behandelt die Auswirkungen der Informatik auf gesellschaftliche Entwicklungen.

Fachgebiete der Informatik: Darstellung aus einem Schulbuch

In der Informatik gibt es Teilbereiche, die bestimmte Problemstellungen in den Fokus rücken. Diese werden hier zuerst einmal vorgestellt und können dann in Aufgaben weiter beispielhaft erkundet werden.

Technische Informatik Die technische Informatik hat die größte Nähe zur Elektrotechnik und Nachrichtentechnik. Sie beschäftigt sich z. B. mit den hardwareseitigen Problemstellungen der Informatik. Das sind z. B. neue Prozessoren oder Übertragungstechniken. Noch sind die technischen Grenzen nicht erreicht, die das Moor'sche Gesetz widerlegen, das grob eine Verdoppelung der Rechenleistung von Mikroprozessoren alle zwei Jahre vorhersagt.

Theoretische Informatik Trotz dieser Steigerung in der Mikroelektronik, die sich auf Geschwindigkeit und Speicherplatz auswirkt, können Schachcomputer aber immer noch nicht eine Schachpartie bis zu ihrem Ende vorausberechnen und so zu einem perfekten Spiel kommen. Theoretisch ist dies jedoch möglich. Dies mit

mathematischen Methoden zu beweisen ist ein Thema der Theoretischen Informatik. Dabei geht es grundsätzlich um die Einordnung von Problemen bezüglich ihrer Komplexität. Die Planung einer möglichst kurzen Rundreise ist für wenige Zwischenstopps schnell lösbar. Allerdings wächst der Zeit und Speicheraufwand exponentiell mit der Anzahl an zu besuchenden Stationen. Dies führt schnell zu einer Überlastung der Systeme und Rechenzeiten von mehreren Jahren. Daneben gibt es auch noch Probleme, die überhaupt nicht berechenbar sind. Mit Hilfe dieser sehr abstrakten Probleme zeigt die Theoretische Informatik die Grenzen der Berechenbarkeit und somit die Grenzen der Informatik auf.

Praktische Informatik Die Praktische Informatik liefert auf der einen Seite Algorithmen und Strukturen für die Grundprobleme der Informatik, z. B. das Suchen in großen Datenmengen oder das Finden von kürzesten Wegen, die in verschiedenen Anwendungsbereichen wiederverwendet werden können. Auf der anderen Seite entwickelt die Praktische Informatik die Methoden und Werkzeuge, mit deren Hilfe man die heutigen komplexen Softwareprodukte entwirft. Dabei ist das Programmieren nur noch ein kleiner Teil der Aufgabe, da schon der vorgelagerte Schritt der Modellierung eines Problems sehr komplex werden kann. Hier wird festgelegt welche Dinge, Beziehungen und Interaktionen der realen Situation in den Rechner übertragen werden sollen und somit programmiert werden müssen.

Anwendungen und Auswirkungen Neben diesen drei Gebieten gibt es noch die Angewandte Informatik, die zu konkreten Produkten, wie z. B. Datenbanken, kommt und das Gebiet »Informatik und Gesellschaft«, das sich mit den Auswirkungen von Informatiksystemen auf den Menschen und die Gesellschaft beschäftigt.

. . . . . .

Aufgabe

Erarbeiten Sie mit Hilfe der Hinweise auf diesem Arbeitsblatt die wichtigsten Aspekte für das Fachgebiet Ihrer Gruppe. Überlegen Sie außerdem welche der vorher gemeinsam gesammelten Begriffe zu ihrem Fachgebiet dazugehören und notieren Sie diese. Finden Sie zusätzlich weitere Beispiele die zu ihrem Fachgebiet gehören. Notieren und kommunizieren sie alle Aufgabeninhalte innerhalb der Gruppe so, dass jeder von Ihnen diese danach vorstellen kann.

. . . . . .

Informatik im Alltag

Arbeitsmethode
Die Bearbeitung des Arbeitsblattes erfolgt nach der **Think-Pair-Share Methode**. Zuerst setzen Sie sich alleine mit der Aufgabe auseinander (Think). Im nachfolgenden Schritt tauschen Sie sich mit ihrem Sitzpartner aus (Pair). Zuletzt besprechen Sie in der gesammten Gruppe die Aufgabe (Share). Nach dem Ende der Gruppenarbeit muss jede/jeder aus Ihrer Gruppe das **Gruppenergebnis** vorstellen können. Es ist wichtig, dass Sie sich bei der Bearbeitung der Aufgaben in der Gruppe einigen.

Arbeitsaufträge/Aufgaben

1. Beschreiben Sie mindestens fünf Situationen im Alltag, in denen Informatik auftaucht bzw. von Bedeutung ist. Diese Situationen sollen in Satzform beschrieben werden und müssen nicht ihrem eigenen Alltag entsprechen. Sie können zum Beispiel auch Meldungen aus den Nachrichten wiedergeben (**Einzelarbeit – Think**).

   a) f)

   b) g)

   c) h)

   d) i)

   e) j)

2. Vergleichen Sie ihre Ergebnisse und ergänzen Sie gegebenfalls ihre obige Liste um weitere Situationen so, dass zehn Situationen beschrieben sind. Diskutieren Sie gemeinsam für jede Situation die zugehörigen Fachgebiete. Notieren sie diese für ihre Beispiele bei der jeweiligen Nummerierung (**Partnerarbeit – Pair**).

   a) f)

   b) g)

   c) h)

   d) i)

   e) j)

3. Diskutieren Sie innerhalb der Gruppe die aus Ihrer Sicht vier interessantesten Alltagssituationen und überprüfen Sie gemeinsam die Korrektheit der Fachge-

biete. Diese vier Situationen sollen im Anschluss vorgestellt werden (**Gruppenarbeit – Share**).

a)

b)

c)

d)

B.1.2 Freihandversuche

$mi \overset{*}{\vdash} mu$ – Sprachbeschreibung

Symbole m i u

Startwort mi

Regeln

1. Ist Xi ein Wort, dann auch Xiu
2. Ist mX ein Wort, dann auch mXX
3. In jedem Wort kann iii durch u ersetzt werden
4. uu kann aus jedem Wort weggelassen werden

X in Regel 1 und 2 steht für eine beliebige Folge von Symbolen
XX bezeichnet die Verdoppelung der Symbolfolge X

Aufgabe Geben Sie die Folge von Umformungen an, die vom Startwort mi durch Anwendung der Regeln zu mu führt.

Hinweise

- Die Regeln dürfen in beliebiger Reihenfolge angewendet werden
- Notieren Sie die Folge der Umformungen in der Form $mi \overset{2}{\vdash} mii \overset{\dots}{\vdash} \dots$
  die über dem Pfeil stehende Zahl bezeichnet dabei die Nummer der Regel, die Sie anwenden.

. . . . . .

Funktionsweise Navigationsgerät

Es ist die kürzeste Verbindung zwischen zwei Orten zu ermitteln.

Um die Problemstellung zu konkretisieren, zeichnen wir das zur konkreten Situation gehörende Netz[1] (vgl. Abbildung). Die Punkte (= Orte) werden hier mit Kleinbuchstaben bezeichnet – die Abstände durch einheitenfreie Zahlen oberhalb der jeweiligen Verbindung.

[1] Eine ähnliche Abbildung und die LaTeX -Quelle wurden von Kjell Magne Fauske auf der Webseite http://www.texample.net/tikz/examples/prims-algorithm/ veröffentlicht.

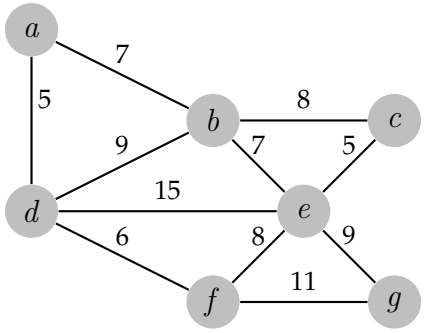


Abbildung: Schema eines Wegenetzes

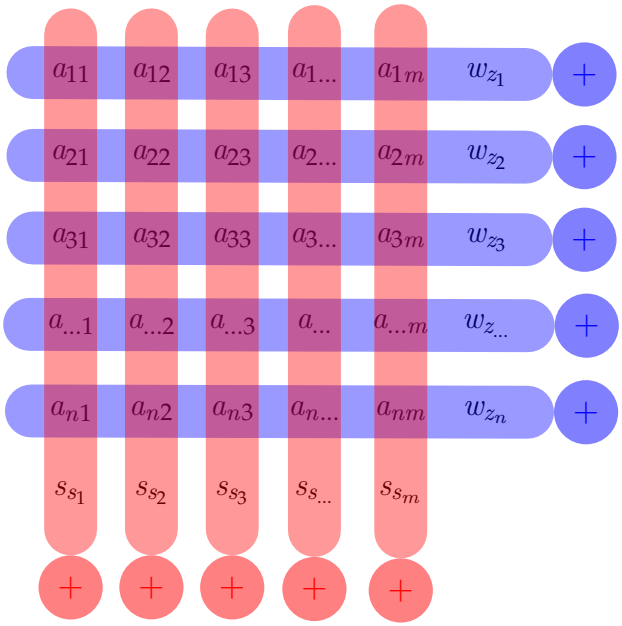
Arbeitsaufträge

1. Bestimmen Sie den kürzesten Weg von a nach g
2. Beschreiben Sie stichwortartig, wie Sie vorgegangen sind
3. Verallgemeinern Sie das Verfahren – beschreiben Sie, wie man vorgehen kann, um den kürzesten Weg von einem Punkt zu einem beliebigen anderen in einem Netzwerk zu finden

. . . . . .

Durchführung des Informatiktricks – Paritätsbit

1. Ich habe Objekte, die zwei verschiedene Zustände z_1, z_2 haben können, wie zum Beispiel Münzen (»Kopf«/»Zahl«).
2. Ein Zuschauer legt diese Objekte mit beliebigem Zustand in einer $m \times n$ Matrix aus.
3. Ich lege eine weitere Spalte und eine weitere Zeile mit den Objekten genau so an, dass eine gerade Anzahl eines bestimmten Zustandes z_1 (z. B. gerade Anzahl »Kopf«) in jeder Zeile und jeder Spalte ist.
4. Der Zuschauer wechselt nun den Zustand eines Objektes, ohne dass ich hinsehe.
5. Um den »Fehler« zu finden, suche ich zeilenweise nach einer ungeraden Anzahl von z_1 (»Kopf«). Ich merke mir die Zeile und suche dann spaltenweise nach einer ungeraden Anzahl von z_1. Der Schnittpunkt der gefundenen Spalte und der gemerkten Zeile ist genau das vom Zuschauer umgedrehte Objekt



Verbindung zur Informatik

Mit dem Anfügen eines Paritätsbits für jede Zeile und Spalte einer binären Matrix wird redundante (überzählige) Information angefügt. Dadurch wird ein zweidimensionaler Fehlerkorrekturcode erschaffen, sodass man einen Fehler nicht nur finden, sondern auch wieder den ursprünglichen Zustand der Bits herstellen kann. Bei 2 oder 3 Fehlern, kann noch Fehler erkennen aber nicht mehr reparieren. Sind 4 Fehler in der Matrix, so kann er nicht mehr erkannt werden.

Paritätsbits werden bei Datenübertragung und Datenspeicherung benutzt, um Fehler zu vermeiden. Ein praktisches Beispiel ist die zerkratzte CD, bei der Kompression und Redundanz gleichzeitig sinnvoll sind.

B.1.3 Geschichte der Informatik

Die Geschichte der Informatik

Arbeitsmethode
Im Zuge des Studientages sollen die Aufgabestellungen als erweiterte Hausaufgabe bearbeitet werden. Schreiben Sie zu der Aufgabe ihrer Gruppe **mindestens 15 Sätze** auf. Die Strukturierung bzw. der Aufbau Ihrer Ausführungen ist dabei Ihnen überlassen. Sie entscheiden selbstständig, welche Hinweise Ihnen am wichtigsten im Bezug auf die Fragestellung erscheinen. Geben Sie allerdings bei wichtigen Ereignissen stets das entsprechende Jahr an. **Die 4. Aufgabe sollte zusätzlich jeder von Ihnen bearbeiten**. Benutzen Sie dabei ihre Notizen aus der vorherigen Stunde zur Ergänzung.

Arbeitsaufträge/Aufgaben

1. Recherchieren Sie Hinweise zu **Gottfried Wilhelm LEIBNIZ** und beschreiben Sie die wichtigsten Aspekte seiner Biographie. Beschreiben Sie dabei vor allem, welche Ideen und Aussagen von LEIBNIZ im Zusammenhang mit der Informatik stehen. Beschreiben Sie außerdem, was sein Beitrag zur heutigen Informatik ist.

2. Recherchieren Sie Hinweise zu **Ada LOVELACE** und beschreiben Sie die wichtigsten Aspekte ihrer Biographie. Beschreiben Sie dabei vor allem, welche Ideen und Aussagen von Ada LOVELACE im Zusammenhang mit der heutigen Informatik stehen. Berücksichtigen Sie auch die Rolle die Charles BABBAGE dabei spielte und erläutern Sie auch seine Beiträge zur Informatik.

3. Recherchieren Sie Hinweise zu **Konrad ZUSE** und beschreiben Sie die wichtigsten Aspekte seiner Biographie. Beschreiben Sie dabei vor allem, welche Ideen und Aussagen von ZUSE im Zusammenhang mit der Informatik stehen. Außerdem sollten sie eine einfache Beschreibung angeben, wie die ersten ZUSE-Rechner funktionierten.

4. Recherchieren Sie Hinweise zu **Alan TURING** und beschreiben Sie die wichtigsten Aspekte seiner Biographie. Beschreiben Sie, in welchem Zusammenhang TURINGs Ideen zur Informatik stehen bzw. was sein Beitrag zur heutigen Informatik ist. Außerdem sollten Sie ausführen, was es mit der ENIGMA auf sich hatte und welche Rolle TURING dabei spielte.

B.1.4 Lernzielkontrolle – Elemente aus EF-1

1. Aufgabe (7 Punkte) Informatik - zum Begriff

a) Geben Sie **Ihre** Definition für Informatik an.

b) Erläutern Sie die Begriffe **Syntax, Semantik und Pragmatik** und grenzen Sie die Begriffe voneinander ab.

c) Stellen Sie einen Zusammenhang zwischen den beiden Teilaufgaben 1a und 1b her.

2. Aufgabe (7 Punkte) Fachgebiete der Informatik

a) Benennen Sie die Fachgebiete, in die die Wissenschaft Informatik üblicherweise aufgeteilt wird.

b) Ordnen Sie die folgenden Elemente den von Ihnen in 2a genannten Fachgebieten zu und begründen Sie kurz (Stichwort reicht) die von Ihnen gewählte Zuordnung:

- Das Ausspionieren von Informatiksystemen.
- Warten auf die Antwort einer Suchmaschine.
- Ein Dokument wird ausgedruckt.
- Jeder Mensch soll programmieren können.
- Die Geschwindigkeit eines Prozessors hat zugenommen.
- Soziale Netzwerke

3. Aufgabe (6 Punkte) Geschichte der Informatik

a) Nennen Sie zwei Personen, die für die Geschichte der Informatik eine wichtige Rolle gespielt haben und beschreiben Sie in wenigen Sätzen die Beiträge dieser Personen zur Informatik.

b) Ordnen Sie die Beiträge der beiden Personen aus 3a den Fachgebieten der Informatik zu.

4. Aufgabe (6 Punkte) BONUSAUFGABE

a) Nennen Sie mindestens vier Dinge, die ein Betriebssystem leistet.

b) Addieren Sie die drei Elemente Ihres Geburtsdatums: tt+mm+jj und wandeln Sie das Ergebnis in eine Dualzahl um. Prüfen Sie Ihr Ergebnis durch Rückrechnung in das Dezimalsystem.

c) Geben Sie die binäre Darstellung für die hexadezimale Zahl $a4b_{16}$ an. Wandeln Sie die Zahl in das Dezimalsystem um und schreiben Sie die oktale Darstellung nieder.

B.2 Vorhaben EF-2

Syntax – Begriffe

Regeln, mit denen »richtige« (= syntaktisch korrekte) Ergebnisse erstellt werden, werden als Grammatik oder Syntax bezeichnet. Die Syntax von (Programmier-)Sprachen wird mit drei verschiedenen Strukturelementen beschrieben.

Bezeichnung	Bedeutung	Beispiel
terminale Symbole	finden sich im Ergebnis als hingeschriebene Symbole wieder	L
nicht terminale Symbole	Zwischenelemente für die Erstellung (oder Analyse)	identifier
Metasymbole	Bildungsstruktur (Sequenz, Alternative, Wiederholung)	|

BNF – EBNF – ISO-EBNF

Um 1960 wurde eine Darstellung für die Syntax von Programmiersprachen entwickelt, die leicht gelesen und verstanden werden kann. Diese Darstellung wurde nach den Entwicklern als Backus-Naur Form (BNF) bezeichnet. Ein Beispiel aus der Syntax von ALGOL 60 zeigt die Regel, um für diese Sprache ein korrektes nicht terminales Symbol »identifier« zu erzeugen. BNF-Darstellung:

```
<identifier>::= <letter> | <identifier> <letter> | <identifier> <digit>
```

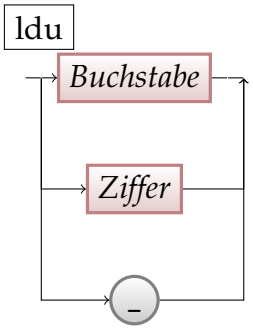
Die BNF wurde zur Extended BNF (EBNF) erweitert (indem weitere Metasymbole hinzugenommen wurden) und 1996 genormt: ISO-EBNF – Standard Extended BNF. EBNF-Darstellung:

```
identifier = letter { letter | digit }.
```

Die Regeln besagen (in Worten): Ein »identifier« wird gebildet durch genau ein nicht terminales Element »letter« gefolgt von beliebig vielen Elementen, die »letter« oder »digit« sein dürfen.

Syntax-Diagramm als alternative Darstellungsform zu [E]BNF

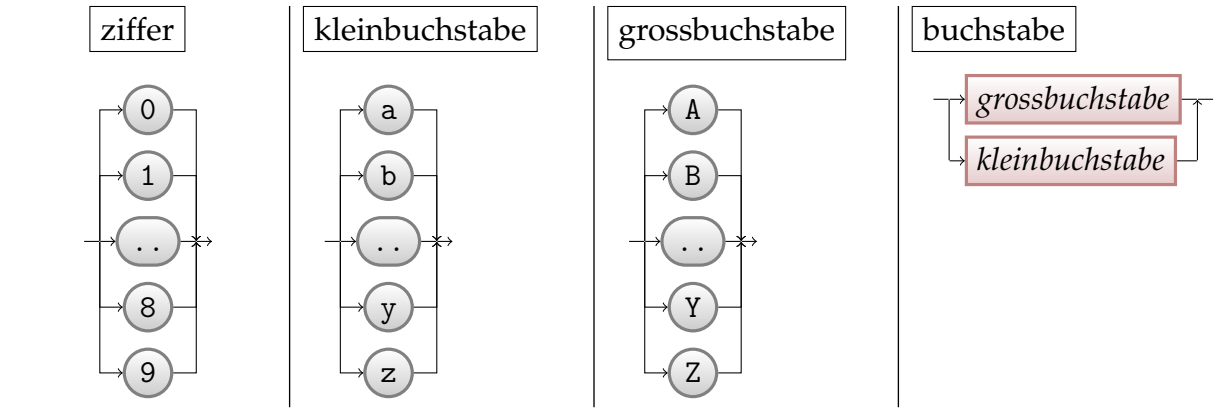
Mit Hilfe sogenannter *Syntax*-Diagramme[a] werden die Metasymbole so dargestellt, dass sie nicht als Zeichen in den Regeln auftreten, sondern in eine grafische Darstellung eingebettet sind. Die Durchlaufrichtung erfolgt von links nach rechts bzw. in Pfeilrichtung. Knotenpunkte sind Verzweigungen im Sinne von Alternativen oder Zusammenführungen von Wegenden.
Im nebenstehenden Beispiel ist das Nonterminal »ldu« aus der Syntax der Programmiersprache Python als Syntax-Diagramm abgebildet. Einträge in den ovalen Textboxen bezeichnen terminale Symbole, während in rechteckigen Textboxen nichtterminale Symbole stehen.



[a] durch ihre Form auch teilweise Railroad (deutsch: Schienenweg) genannt

Bezeichner – Syntax – »Railroad-Diagramm«

NIKLAUS WIRTH entwickelte für seine Studierenden an der ETH-Zürich eine Form zur Präsentation von Regeln, die heute unter dem Namen »Railroad-Diagramm« bekannt ist.



Zeichnen Sie das Railroad-Diagramm für Bezeichner

bezeichner

Überprüfung: Sind folgende Zeichenketten syntaktisch korrekte Bezeichner?

	Zeichenkette	Bezeichner (Ja | Nein)	Begründung (bei Nein) durch Angabe der Regel
a	`Wartezimmer`		
b	`4711`		
c	`__init__`		
d	`Liste2`		
e	`meinStift`		
f	`bildhöhe`		
g	`3komma4`		
h	`vier␣u␣drei`		
i	`_34_`		
j	`art.anfuegen`		
k	`vierUnd3`		

Geben Sie für die Überprüfungen mit dem Ergebnis **Nein** an, warum die Zeichenkette kein zulässiges Element der Art Bezeichner ist. Dabei ziehen Sie die Syntax heran und geben an, gegen welche Regel verstossen wird – durch konkrete Angabe des nicht terminalen Symbols, bei dem der Regelverstoss auftritt. Markieren Sie die erste fehlerhafte Stelle in der geprüften Zeichenkette (durch Unterstreichen mit einem Bleistift).

Problembeschreibung: Online-Buchversand

Ausgangssituation

Der Online-Buchversand »Lesemaus« bietet zwei Bücher besonders günstig im Angebot an. Das sind die Bücher:

- »Marry Plotter« von Joanne Howling, ISBN-Nummer 978-355157777-1, Preis: 8,95 €
- »Exakte Voraussagen« von Rainer Zufall, ISBN-Nummer 978-382737152-2, Preis: 35,75 €

Von »Marry Plotter« gibt es noch 7 Exemplare, von »Exakte Voraussagen« noch 23. Martin und Steffi haben ein Kundenkonto bei »Lesemaus«. Martin hat die Kundennummer 123-456-789 und Steffi die Kundennummer 555-321-890.

Aufgabe

1. Ermitteln Sie die vorkommenden Objekte und legen Sie die Objektbezeichner fest.
2. Bestimmen Sie die Attribute für die Objekte und legen Sie die Attributbezeichner fest.
3. Notieren Sie die Attributwerte auf den Objektkarten.

Problembeschreibung: Supermarkt

Ausgangssituation

Der Supermarkt »Kundenfreund« verkauft Lebensmittel und sonstige Waren. Beim Einkauf werden alle mit einem Barcode ausgezeichneten Waren von dem Scanner der Kasse erfasst. Bei Ware wie Obst und Gemüse muss die Kassiererin eine Produktnummer eingeben und die Ware wird gewogen. Am Schluss wird ein Kassenbon mit dem Gesamtwert gedruckt. Die Mitarbeiterin Ute Kasimir arbeitet gerade an Kasse 7 und wiegt dort eine Tomate ab. Der Kilopreis beträgt 1,99 €.

Aufgabe

1. Ermitteln Sie die vorkommenden Objekte und legen Sie die Objektbezeichner fest.
2. Bestimmen Sie die Attribute für die Objekte und legen Sie die Attributbezeichner fest.
3. Notieren Sie die Attributwerte auf den Objektkarten.

Problembeschreibung Fahrkartenauskunft

Ausgangssituation

Das örtliche Nahverkehrsunternehmen »NahUnt« will an den Bushaltestellen Fahrscheinautomaten installieren. An dem Automaten kann der Kunde eine Entfernungszone per Knopfdruck wählen. Es gibt drei Entfernungszonen mit unterschiedlichen Preisen: 1.Zone: 1,10 €, 2.Zone: 1,90 €, 3.Zone: 4,20 €. In einem Display steht als erstes der Text »Bitte wählen Sie eine Entfernungszone aus«. Nach der Betätigung einer Entfernungszonentaste soll die ausgewählte Zone und der Preis angezeigt werden.

Aufgabe

1. Ermitteln Sie die vorkommenden Objekte und legen Sie die Objektbezeichner fest.
2. Bestimmen Sie die Attribute für die Objekte und legen Sie die Attributbezeichner fest.
3. Notieren Sie die Attributwerte auf den Objektkarten.

B.2.1 Lernzielkontrolle – erste Elemente aus EF-2

1. Aufgabe **(7 Punkte)** Korrigieren Sie die Fehler auf der folgenden Objektkarte gemäß der Regeln für gültige Bezeichner und der Vereinbarungen über die Schreibweise von Objektkarten, indem Sie eine korrigierte Objektkarte neben die gegebene zeichnen.

Der USB-Stick = Stick
Aussehen : Herzchen Kaufdatum = 8. März kapazität : 1 Terabyte
mounten Inhaltsverzeichnis lesen

2. Aufgabe **(8 Punkte)** Suchen Sie aus dem folgenden Text die Objekte, Attribute und Methoden nach dem Verfahren von Abbott heraus. Erstellen Sie die zugehörigen Objektkarten mit syntaktisch korrekten Bezeichnern und gemäß den Vereinbarungen über die Schreibweise von Objektkarten auf einem gesonderten Blatt. Identifizieren Sie nur die Objekte, Attribute und Methoden, die aus dem Text hervorgehen!

> *Ela muss sich aufraffen, weil heute der letzte Termin für die Prüfung ist. Sie ist gut vorbereitet, hat sich mit den Prüfungsgegenständen beschäftigt und glaubt, ganz gut Bescheid zu wissen. Dann sieht sie die Aufgaben und merkt bei der Bearbeitung, dass sie nicht schnell genung arbeitet, weil ihr die Zeit davon läuft. Wird sie es schaffen, die Prüfung zu bestehen?*

3. Aufgabe **(4 Punkte)**

Erläutern Sie die Begriffe Modellierungskreis, wissensbasierte Modellierung, funktionale Modellierung und objektorientierte Modellierung.

B.2.2 Übergreifende Problemstellung – OOM

Problembeschreibung: Basketballturnier

Am Dienstag, den 16.08.2037[2] wird an der Willy-Brandt-Gesamtschule Bergkamen von 9^{30} Uhr bis 12^{30} Uhr ein Basketballturnier der EF stattfinden.

Die vier Deutschgrundkurse der Stufe werden jeweils als Team gegeneinander antreten. Das Team des Kurses EFd1 nennt sich »Die Kängurus« und spielt in blauen Trikots, das des Kurses EFd2 (die »Crazy Frogs«) spielt in Grün. »Die Überflieger« sind die Teilnehmer des Kurses EFd3, welche in gelben Leibchen antreten. Als besondere Ehrung eines weiterhin populären und mittlerweile ergrauten Basketballspielers nennen sich die Schülerinnen und Schüler des Kurses EFd4 »Die Nowitzkis«. Sie tragen graue Trikots.

Die Spiele finden in Halle A im Altbau und Halle B im Neubau der Schule statt. In insgesamt sechs Spielen spielt jede Mannschaft gegen jede andere. Das Eröffnungsspiel findet zwischen den Kängurus und den Nowitzkis von 9^{45} Uhr bis 10^{15} Uhr in Halle A statt und endet mit dem Ergebnis 32:20.

Aufgabe

1. Ermitteln Sie mithilfe des Verfahrens nach Abbott die in der Ausgangssituation vorkommenden Objekte und ihre zugehörigen Attribute und Methoden.
2. Zeichnen Sie für die in Aufgabe 1 identifizierten Objekte die jeweiligen Objektkarten.

Beziehungen zwischen Objekten in Objektdiagrammen

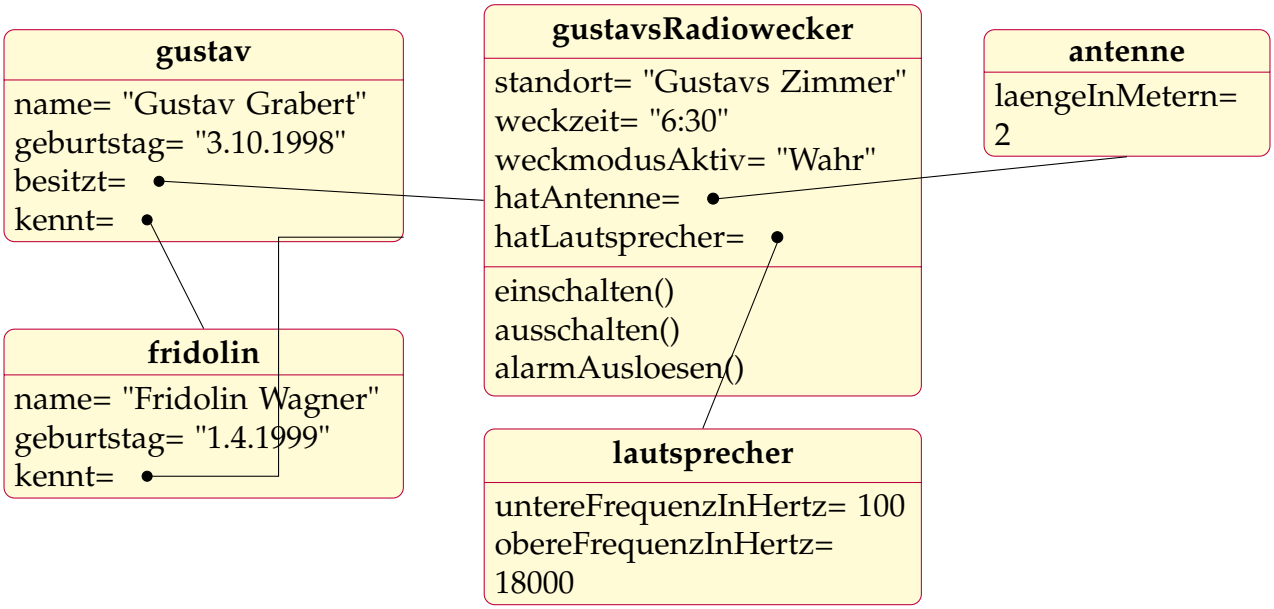
Mit Hilfe des Verfahrens nach ABBOTT werden mögliche Objekte identifiziert. Für die als relevant befundenen Objekte werden Objektkarten erstellt. Bisher sind die Objektkarten noch isoliert, da die Beziehungen, die zwischen den Objekten bestehen, nicht berücksichtigt wurden.

Die Beziehungen zwischen den Objekten werden im **Objektdiagramm** dargestellt. In diesem Diagrammtyp gibt es zwei Arten von Beziehungen: Die Kennt-Beziehung

[2] Es wurde eine handelsübliche Kristallkugel benutzt. Eine bessere Vorhersagewahrscheinlichkeit als die der Wettervorhersage für das kommende Jahr darf deshalb leider nicht erwartet werden.

und die Hat-Beziehung. Für jede **Objektbeziehung** wird bei der Objektkarte des Objektes, von dem die Beziehung ausgeht, ein neues Attribut benötigt, das einen sinnvollen Bezeichner erhält. Die Verbindung wird mit einem schwarzen Punkt hinter dem Beziehungsattribut gekennzeichnet. Von dem Punkt aus wird eine Linie zu dem Zielobjekt gezeichnet.

Beispiel für eine einfaches Objektdiagramm mit fünf Objekten.



Die **Hat-Beziehung** kann in der Regel als die Beziehung zwischen »dem Ganzen« und seinen Einzelteilen angesehen werden. Im obigen Beispiel verfügt der Radiowecker über eine Antenne und einen Lautsprecher. Beide Einzelteile wären ohne ihre Kombination innerhalb des Radioweckers (in diesem Modellierungskontext) nicht sinnvoll einsetzbar. Ferner kann man sich vorstellen, dass der Konstrukteur des Radioweckers sich beim Zusammenbau auch um die Anfertigung/Beschaffung der Antenne und des Lautsprechers kümmern muss.

Die **Kennt-Beziehung** ist eine allgemeinere Beziehung zwischen zwei Objekten. Um die Art der Beziehung zu konkretisieren, kann die Verbindungslinie zwischen den jeweiligen Objekten beschriftet werden – im obigen Beispiel etwa mit »ist befreundet mit«.

Beispielsweise besitzt Gustav seinen Wecker (und nicht anders herum)[3].

[3]Obwohl umgangssprachlich auch die Formulierung »Gustav *hat* einen Wecker« gebräuchlich ist, handelt es sich *nicht* um eine Hat-Beziehung im Sinne der objektorientierten Modellierung, da der Radiowecker kein Teil Gustavs ist und auch unabhängig von diesem existiert.

Aufgabe

Überlegen Sie sich, welche Beziehungen zwischen den Objekten, die bei der Analyse des Problems »Basketballturnier« gefunden wurden, bestehen und zeichnen Sie das zugehörige Objektdiagramm.

Spielplan des Basketballturniers

Der Organisator des Basketballturniers an der WBGE hat einen genauen Spielplan aller Begegnungen erstellt.

Halle A	Halle B
A gegen B 9:45–10:15	C gegen D 10:00–10:30
A gegen C 10:45–11:15	B gegen D 11:00–11:30
A gegen D 11:45–12:15	B gegen C 12:00–12:30

Weil sich die Mannschaftsnamen eventuell sogar noch während des Turniers ändern könnten, finden sich nur Kürzel im Spielplan wieder (Mannschaftbezeichnungen A, B, C, D).

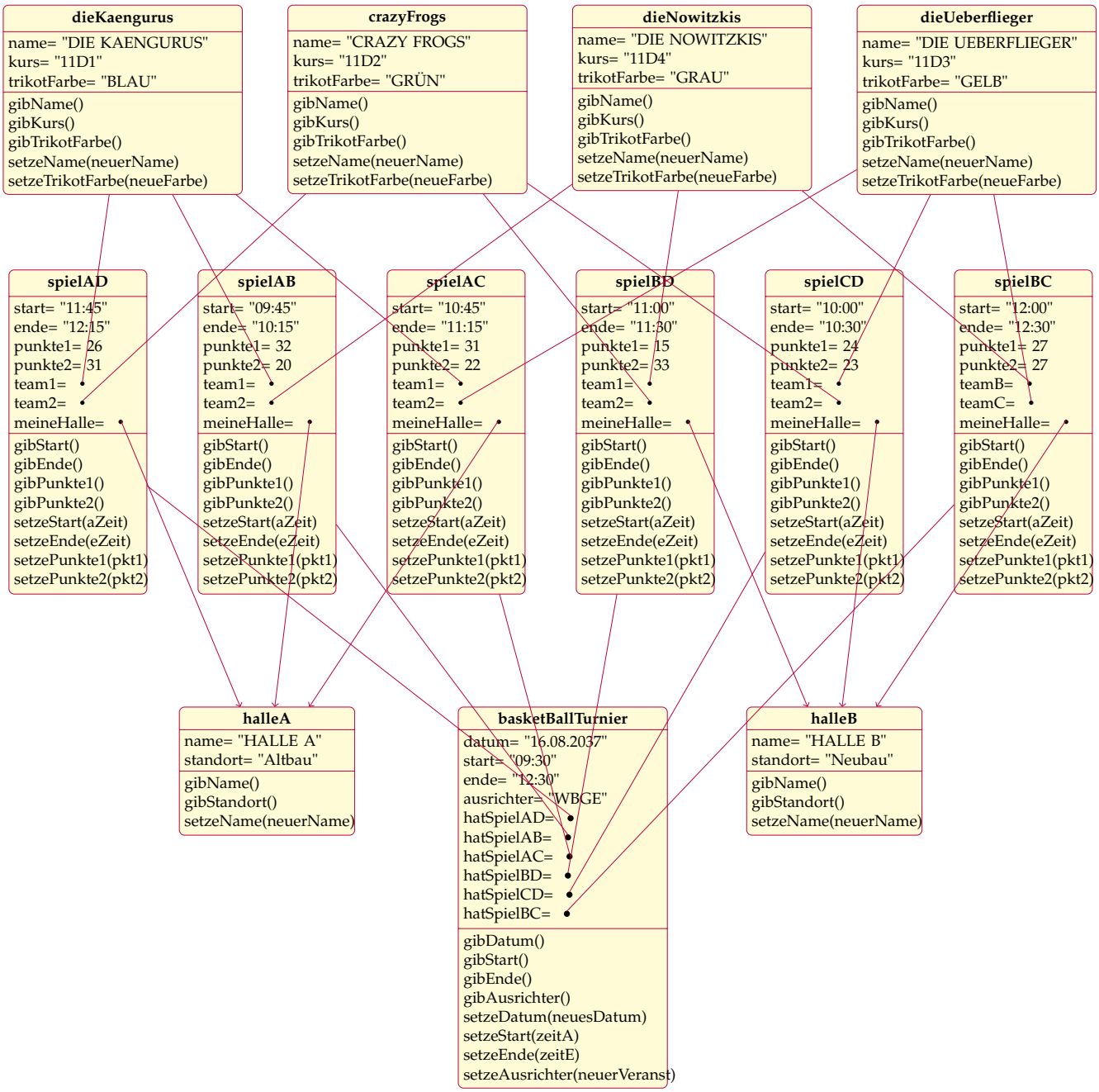
Die Kürzel übersetzen sich nach folgender Tabelle in reale Mannschaftsnamen:

A: »Die Kängurus«
B: »Die Nowitzkis«
C: »Die Überflieger«
D: »Crazy Frogs«

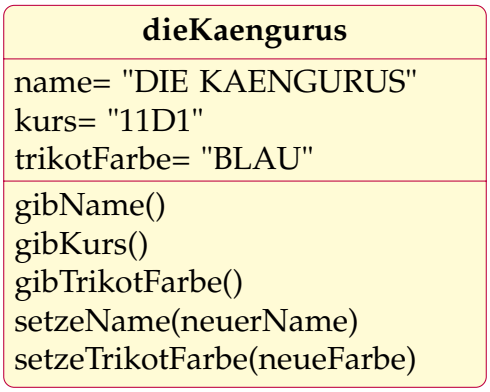
Nach Abpfiff eines jeden Spiels steht das Ergebnis fest und kann notiert werden. Der Spielleiter gibt nach jedem Spiel die aktuelle Tabelle bekannt.

Die Resultate der Begegnungen lauten:

A:B	32:20
C:D	24:23
A:C	31:22
B:D	15:33
A:D	26:31
B:C	27:27



Beschreibung der Methoden eines Objekts



Die Objektkarte zeigt das Objekt `dieKaengurus` inklusive einiger Methoden. Im Folgenden wird dargestellt, wie eine Beschreibung der Methoden aussehen kann.

Um arbeiten zu können, benötigen Methoden häufig Daten, die ihnen der Aufrufer der Methode zur Verfügung stellt. Diese Daten werden der Methode als *Parameterwerte* (oder *Argumente*) übergeben.

Methoden können nicht nur Objekte manipulieren (in der Regel das Objekt, zu dem die Methode gehört), sondern auch Daten an den Aufrufer zurückgeben – dann spricht man vom *Rückgabewert*.

Man könnte sich beispielsweise ein Objekt `rechner` vorstellen, das mathematische Berechnungen ausführt. Die Methode `pi()` des Objekts `rechner` kann bei jedem Aufruf den Wert 3,14159 zurückgeben. Einer Methode wie `quadrat()` müsste jedoch die Zahl, für die die Quadratberechnung ausgeführt werden soll, als Parameterwert mitgegeben werden: `rechner.quadrat(3)` hätte den Rückgabewert 9.

Die Beschreibung einer Methode sollte zumindest eine Übersicht der Parameter und (sofern vorhanden) des Rückgabewerts umfassen. Eine kurze verbale Beschreibung der Arbeitsweise ist äußerst sinnvoll. Zwei Beispiele:

- Methode `gibName()`

  Parameter: keine
  Rückgabewert: Wert des Attributs `name`
  Beschreibung: Die Methode liest den im Attribut `name` gespeicherten Wert und gibt diesen zurück.

- Methode `setzeName(neuerName)`

  Parameter: `neuerName` – der neue Mannschaftsname
  Rückgabewert: keiner
  Beschreibung: Die Methode weist dem Attribut `name` den in `neuerName` angegebenen Wert zu. Der alte Wert wird überschrieben.

In vielen objektorientierten Programmiersprachen (hier am Beispiel von Python) können die Methoden nach folgendem Schema genutzt werden:

`print (dieKaengurus.gibName())`	Gibt zu Beginn `"DIE KAENGURUS"` aus
`dieKaengurus.setzeName("SKIPPY-TEAM")`	überschreibt den Wert des Attributs `name` durch `"SKIPPY-TEAM"`
`print (dieKaengurus.gibName())`	Gibt nun `"SKIPPY-TEAM"` aus

An diesem Beispiel erkennt man gut, dass der Objektbezeichner `dieKaengurus` im Prinzip nichts über den Inhalt des Objekts aussagt. Der Bezeichner könnte genausogut `meineMannschaft` heissen.

Ein direkter Zugriff auf das Attribut `name` anstelle des »Umwegs« über Methoden wäre prinzipiell möglich, es ist jedoch »guter Stil«, nicht von außen direkt auf Attribute zuzugreifen, damit sich jedes Objekt eigenständig verwalten kann (Geheimnisprinzip). So lassen sich Teile eines komplexen Programms ändern, ohne dass die Auswirkungen auf andere Programmteile unüberschaubar werden.

Objektspiel zum Basketballturnier

Das Objektdiagramm und die Spielergebnisse zum Basketballturnier werden als bekannt vorausgesetzt und sind Arbeitsgrundlage für dieses Arbeitsblatt. Außerdem sollte das Informationsblatt zum Objektspiel vorliegen.

Das Objektspiel auf diesem Arbeitsblatt befasst sich ausschließlich mit den ersten beiden Spielen des Basketballturniers in Halle A (siehe Spielplan des Basketballturniers). Es spiegelt also nur einen kleinen Teil des Turnierverlaufs wieder.

Problembeschreibung für das Objektspiel:

Zum ersten Spiel des Tages in Halle A kommt Team A leider zu spät. Die Turnierleitung (repräsentiert durch das Objekt `basketBallTurnier`) ändert daher die Startzeit auf 10 Uhr und passt dementsprechend auch die Endzeit an. Auch beim zweiten Spiel verschieben sie die Zeiten um 5 Minuten nach hinten. Team A gewinnt das Spiel trotz ihrer Verspätung verdient mit 32:20. Team A möchte einen Überblick über ihre erzielten Körbe im Turnierverlauf behalten. Daher informieren Sie sich über die erzielten Körbe im 1. Spiel und notieren diese. Außerdem möchte Team A wissen, wer ihr nächster Gegner ist. Auch darüber informieren Sie sich und notieren sich den Teamnamen.

Team B ist frustriert nach der Niederlage und möchte daher unbedingt die Trikotfarbe wechseln. Sie waren so begeistert von Team A, dass sie deren Trikotfarbe einfach kopieren möchten. Außerdem ändern sie ihren Teamnamen in "Die Jordans".

Team C spielt im Spiel 2 in Halle A. Das Team hat aber leider vergessen, ob Halle A der Altbau oder der Neubau ist. Nachdem sie es herausgefunden haben, findet Spiel 2 pünktlich statt und endet mit 31:22 für Team A. Team A notiert sich danach erneut ihre erzielten Körbe.

. . . . . .

Arbeitsaufträge/Aufgaben

1. Ermitteln Sie alle beteiligten Objekte an den ersten beiden Spielen in Halle A mit Hilfe des Spielplans und des Objektdiagramms und geben Sie diese an:

   a) f)

   b) g)

   c) h)

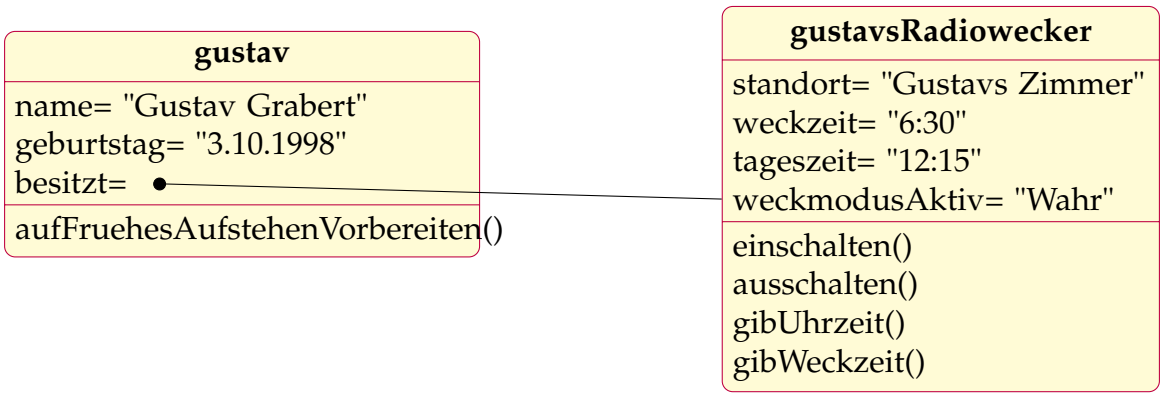
   d) i)

   e) j)

2. Erstellen Sie in Gruppenarbeit für je ein Objekt ein Plakat nach Vorgaben der Regeln für das Objektspiel. Nehmen Sie dafür das Objektdiagramm als Grundlage. Die Eigenschaften der Objekte sollten den Moment direkt vor Turnierbeginn widerspiegeln. Ergebnisse sind also z. B. noch unbekannt.
3. Führen Sie das Objektspiel mit der angegebenen Problembeschreibung durch.

Interaktion von Objekten: Sequenzdiagramme

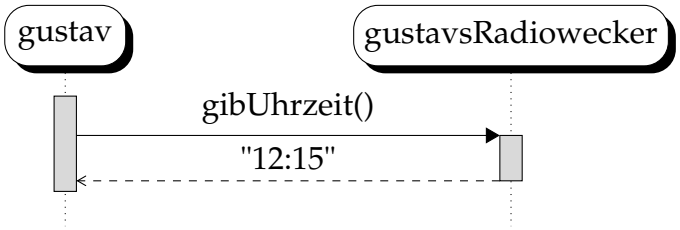
Objektdiagramme zeigen lediglich statische Beziehungen zwischen den vorkommenden Objekten und Momentaufnahmen von Attributwerten.



Will man jedoch darstellen, wie die Objekte miteinander interagieren, so muss der zeitliche Ablauf beschrieben werden. Dies geschieht in den sogenannten *Sequenzdiagramm*en.

Es ist möglich, sich die Interaktion der Objekte als Kommunikation (d. h. als einen Austausch von »Nachrichten«) vorzustellen. Jedes Objekt bietet allen anderen an, seine »Dienste« zu nutzen – in der Fachsprache: Seine Methoden aufzurufen.

Das folgende Sequenzdiagramm zeigt, wie das Objekt `gustav` beim Objekt `gustavsRadiowecker` die aktuelle Uhrzeit abfragt.



Aufgabe

1. Erstellen Sie ein Sequenzdiagramm, das beschreibt, wie Gustavs Mutter ihren Sohn dazu auffordert, seinen Wecker einzuschalten!
2. Überlegen Sie sich, wie eine der teilnehmenden Mannschaften (z. B. die »Crazy Frogs«) den Sieger des Basketballturniers in Erfahrung bringen kann.

   Klären Sie dazu zunächst, über welche zusätzlichen Methoden die Objekte noch verfügen müssen (beispielsweise eine Methode `ermittleSieger()` des Eröffnungsspiels) und halten Sie kurz schriftlich fest, was diese Methoden leisten.

   Erstellen Sie das zugehörige Sequenzdiagramm!

B.2.3 Problemstellung »Schwarzes Brett« – OOM

Die Firma Schoolsoft hat den Auftrag erhalten, die örtlichen Schulen mit einem digitalen Informationssystem auszustatten. Dieses soll mittelfristig die Informationsbretter an den Schulen ablösen, sodass die Schülerinnen und Schüler auch von zuhause jederzeit die relevanten Mitteilungen abfragen und eigene Gesuche eintragen können. Zu Planungszwecken hat Schoolsoft eine Stunde lang ein schulisches Informationsbrett beobachtet und die Ereignisse als Problembeschreibung protokolliert.

Problembeschreibung

An einer Schule gibt es ein Informationsbrett, welches drei Meter breit, zwei Meter hoch und weiß lackiert ist. Auf dem Informationsbrett können Zettel angebracht werden, auf denen Gesuche der Schülerinnen und Schüler oder Mitteilungen der Lehrer notiert sein können.

Der Schüler mit dem Namen Hans Müller erstellt sich einen karierten Zettel und beschriftet ihn mit »Ich suche meinen Füller, Hans Müller«. Er lässt das Informationsbrett den karierten Zettel aufnehmen.

Die Schülerin Susi Meyer erstellt sich einen linierten Zettel und beschriftet ihn mit »Morgen ist schulfrei!«. Anschließend veranlasst sie das Informationsbrett, den linierten Zettel aufzunehmen.

Etwas später sucht der fünfzehnjährige Schüler Peter das Informationsbrett ab. Er erhält vom Informationsbrett die Daten »Ich suche meinen Füller, Hans Müller« und »Morgen ist schulfrei!«.

Nachdem der Schüler Hans seinen Füller wiedergefunden hat, nimmt er seinen Zettel vom Informationsbrett ab und wirft ihn weg.

Aufgaben

1. Entwerfen Sie zu der gegebenen Problembeschreibung mit Hilfe des Verfahrens von Abbott (siehe Informationsblatt) ein objektorieniertes Modell, indem Sie die relevanten Objekte mit ihren Attributen und Methoden identifizieren. Notieren Sie die Objekte als Objektkarten.
2. Aus der Problembeschreibung wird ersichtlich, dass Objekte miteinander interagieren. Identifizieren Sie alle relevanten Beziehungen zwischen den Objekten.

Nr.	Startobjekt	Zielobjekt	Dienst Zielobjekt	Antwort	Bemerkung
1	außen	schueler1	erzeugeZettel		zettel1 ist erzeugt
2	außen	schueler1	schreibeZettel		Parameter?
3	schueler1	zettel1	setzeText ("Mein Füller ...")		
4	außen	schueler1	haengeZettelAuf (infoboard)		
5	schueler1	infoboard	nimmZettelAuf (zettel1)		
6	außen	schueler2	erzeugeZettel		zettel2 ist erzeugt
7	außen	schueler2	schreibeZettel ("Morgen fällt die Schule aus.")		
8	schueler2	zettel2	setzeText ("Morgen ...")		
9	außen	schueler2	haengeZettelAuf (infoboard)		
10	schueler2	infoboard	nimmZettelAuf (zettel2)		
11	außen	schueler3	schaueAn (infoboard)		
12	schueler3	infoboard	zeigeOeffentlich	"Mein Füller ... Morgen fällt ..."	Textausgabe?
13	infoboard	zettel1	gibText	"Mein Füller ..."	
14	infoboard	zettel2	gibText	"Morgen ..."	
15	außen	schueler1	nimmZettelAb		Parameter infoboard?
16	schueler1	infoboard	entferneZettel (zettel1)		
17	außen	schueler1	wirfZettelWeg		
18	schueler1	zettel1	zerstoere		

B.2.4 Projektarbeit zum EF-2

Schriftliche Ausarbeitung zur objektorientierten Modellierung

Bewertung der Projektarbeit: Für die Bewertung der schriftlichen Ausarbeitung ist neben dem inhaltlich sinnvollen und syntaktisch korrekten Modellieren besonders das saubere Arbeiten wichtig. Daher empfiehlt es sich, bei Diagrammen zuerst vorzuzeichnen und erst danach das fertige Diagramm strukturiert und sauber aufzuzeichnen.

Inhalt der schriftlichen Ausarbeitung

Für die schriftliche Ausarbeitung sollen fünf verschiedene Aspekte einer objektorientierten Modellierung bearbeitet werden. Dabei sind Sie in Ihrer Partnerarbeit sehr frei in der konkreten Gestaltung der Inhalte. Jeder Modellierungsschritt soll in der Ausarbeitung auf einer neuen DIN A4 Seite beginnen. Denken Sie an die unbedingte Verwendung von kariertem Papier.

1. **Problemstellung:**

   Nachdem Sie nun einige Szenarien objektorientiert modelliert haben, ist es an der Zeit, dass Sie selbst ein geeignetes Szenario (Aufgabenstellung – Problemstellung – Situation) entwickeln. Beschreiben Sie in einem Fließtext Ihr selbst gewähltes Szenario.

   Achten Sie darauf, dass Ihr Szenario nicht zu einfach, aber auch nicht zu komplex ist. Ungefähr 5 bis 10 Sätze, in denen insgesamt 4 bis 7 Objekte vorkommen, sollten als Richtwert reichen. Achten Sie außerdem bei der Problemstellung darauf, dass Sie von jeder Objektsorte, die sich aus der Problemstellung ergeben, mindestens zwei Objekte vorsehen.

   Sie sollten die Problemstellung so formulieren, dass Sie für eine objektorientierte Modellierung geeignet ist. Denken Sie also schon beim Schreiben des Textes darüber nach, ob Sie aus Ihrem Szenario auch ein sinnvolles Sequenzdiagramm erstellen können.

2. **Objektdiagramm:**

   Erstellen Sie für Ihr Szenario die Objektkarten. Stellen Sie die Beziehungen der Objektkarten in einem Objektdiagramm dar. Verwenden Sie nur Attribute und Methoden, die für die Modellierung ihrer Problemstellung wichtig sind. Diese müssen auch nicht zwingend im Text vorkommen.

3. **Beschreibung der Methoden:**

   Beschreiben Sie Ihre gewählten Methoden und deren Parameter und Rückgabewerte. Die Beschreibungsstruktur sollte aufgebaut sein wie auf dem Infoblatt »Beschreibung der Methoden eines Objekts«.

4. **Objektinteraktionsprotokoll:**

   Erstellen Sie das Objektinteraktionsprotokoll für Ihr Szenario.

5. **Sequenzdiagramm:**

   Erstellen zu Ihrem Objektinteraktionsprotokoll ein Sequenzdiagramm.

B.3 Vorhaben EF-3: Datenschutz – Arbeitsblätter zur Nachbearbeitung

Einordnung des Planspiels in das Fach und die Welt der Wissenschaften

Durch die Arbeit mit den Rollen, Stationen und Vorfällen im Zusammenhang mit dem Planspiel Datenschutz haben Sie einen Einblick in die Aufnahme, Erfassung, Speicherung, Nutzung, Verarbeitung, Weitergabe und Auswertung von Daten erhalten.

Wir haben uns im Informatikunterricht mit diesem Thema beschäftigt.

1. Aufgabe Begründen Sie, warum das Thema Datenschutz Teil des Informatikunterrichts ist.

2. Aufgabe Erläutern Sie, welche Fachgebiete der Informatik mit den Elementen zu tun haben, die in dem Planspiel vorkommen.

3. Aufgabe Welche weiteren Schulfächer müssten sich auch mit Fragen des Datenschutzes beschäftigen?

4. Aufgabe Welche Wissenschaft, die an der Schule nicht durch ein Fach vertreten ist, spielt für Fragen des Datenschutzes eine große Rolle?

Wen schützt der Datenschutz?

Der Begriff Datenschutz ist irreführend.

Wenn Sie sich den Ablauf des Planspiels und die folgende Auswertung der Daten vergegenwärtigen, fällt schnell auf, dass es nicht um den Schutz der Daten geht. Es geht um die Personen, deren Daten erfasst, erhoben, abgegeben wurden.

Ohne weitere Kenntnis einer konkreten Person – nur durch Sichtung der Daten, die diese Person im Planspiel »produziert« – wurden dieser Person bestimmte Eigenschaften zugeschrieben.

Denken Sie an die Auswertung des Kaufs von Werkzeug, die je nach Vorfall auf drei Arten »bewertet« wurde:

- könnte mit dem Werkzeug seinen kaputten Roller repariert haben → verdächtig
- interessiert sich in seiner Freizeit für technische Fragen → engagierter Mitarbeiter
- repariert offenbar Sachen selbst – hat also wenig Geld → interessiert sich nicht für überflüssige Angebote

Was können wir daraus schließen? Daten, die sich auf eine Person beziehen, werden zielgerichtet interpretiert und können verschiedene Konsequenzen für diese Person haben. Dabei wird oft nicht die Frage gestellt, welches Ziel die Auswertung hat – noch weniger wird die Frage gestellt, wozu die Daten ursprünglich erhoben oder gesammelt wurden.

Die Personen müssen geschützt werden.

Sie werden dadurch geschützt, dass man mit den Daten einer Person nicht beliebige Dinge tun darf.

Im Datenschutz wird hier der Begriff Datenerfassung verwendet und davon gesprochen, dass Daten nur zu dem Zweck verwendet werden dürfen, aus dem heraus sie erfasst wurden.

Zur weiteren Bearbeitung bilden wir Kleingruppen, die eine der folgenden Fragen/-Punkte klären und sich eine weitere Fragestellung vornehmen, die bisher noch nicht beleuchtet wurde.

1. Aufgabe Persönlichkeitsschutz und Datenschutz – Begriffsklärung

2. Aufgabe Was sind denn nun eigentlich schützenswerte Daten?

3. Aufgabe Wann greift Datenschutz: Erfassung(), Verarbeitung(), Weitergabe()

4. Aufgabe Darf die Polizei/Verfassungsschutz meine Telefongespräche mithören?

5. Aufgabe Gibt es für das Modellieren irgendwelche Konsequenzen?

B.4 Vorhaben EF-4: Suchen und Sortieren – Arbeitsblätter

Bearbeitung der Vorfälle

Bei der Klärung der Vorfälle haben Sie mit den Daten gearbeitet, die im Planspiel in die zur Verfügung stehenden Blätter eingetragen wurden.

Gehen wir mal davon aus, dass Sie die Vermutung haben, die Person mit der Personalnummer 234 könnte für den Vorfall, den Sie bearbeiten, so wichtig sein, dass Sie nun die Adresse dieser Person benötigen.

1. Aufgabe Beschreiben Sie Ihr Vorgehen.

2. Aufgabe Geben Sie an, wieviele Zeilen auf dem Blatt Sie sich **mindestens, höchstens** ansehen müssen, um die passende Zeile auf dem Blatt zu finden.

3. Aufgabe Wieviele Zeilen müssen Sie sich **im Mittel** ansehen.

Datensammlung ordnen

Die übergeordnete Aufgabe besteht darin, eine Menge von Daten nach einem Kriterium zu ordnen.

Aufgabe – Modellierungssituation

Es gibt eine Menge von Daten, die geordnet werden sollen. Um die Situation auch durchführen zu können, verwenden wir konkrete Daten:

1. Erstellen Sie eine Objektkarte und schreiben Sie den Vornamen Ihrer Mutter auf die Rückseite.
2. Legen Sie den Zettel so vor sich, dass niemand den Namen lesen kann.
3. Anschließend überlegen Sie, wie man vorgehen kann, um – ohne dass jemand auf einen Schlag alle Zettel aufdecken kann – alle Vornamen in einer Reihenfolge geordnet angegeben werden können.
4. Dazu schreiben Sie auf die vordere Seite der Objektkarte zunächst einmal die benötigten Attribute ohne den Attributwert, den Sie hinten aufgeschrieben haben.
5. Ermitteln Sie zunächst die beteiligten Objekte und überlegen Sie sich das zugehörige Objektdiagramm.

6. Überlegen Sie, was jetzt getan werden muss, um die übergeordnete Aufgabe zu lösen und notieren Sie Stichworte dazu.

B.5 Vorhaben EF-5: Von Objekten zur Klasse – Arbeitsblätter

Die grafische Darstellung von Klassen – Klassenkarte

Prinzipiell ähnelt die grafische Darstellung von Klassen derjenigen von Objekten, es existieren jedoch auch einige Unterschiede.

Objekt

Klasse

Auffällig ist zunächst, dass Objekte durch abgerundete Ecken gekennzeichnet werden, Klassen hingegen nicht. Anstelle des Objektbezeichners steht bei Klassen der Klassenname/Klassenbezeichner. Er beginnt per Konvention immer mit einem Großbuchstaben.

Da Klassen Abstraktionen von Objekten sind, gibt es keine konkreten Attributwerte. Stattdessen wird angegeben, welche Art von Wert (Typ) als Attributwert zulässig ist. Beispiele für Typen sind Zeichenketten, Zahlen und Wahrheitswerte. Auch Objekte können Attributwerte bei Objekten darstellen – hier wird dann in der Klasse die zugehörige Klasse hinter dem Doppelpunkt als Typ des Attributs angegeben.

Üblicherweise wird sowohl in Klassen- als auch in Objektdiagrammen die Typen der Rückgabewerte der Methoden aufgeführt. In der obigen Abbildung ist dies nur im Klassendiagramm geschehen.

Die aus den Objektdiagrammen bekannten Beziehungen (die Hat- und die Kennt-Beziehung) existieren auch zwischen Klassen.

Aufgaben

Ihnen liegt das gemeinsam erstellte Objektdiagramm vor. Erstellen Sie das zugehörige Klassendiagramm nach der folgenden Anleitung:

1. Tragen Sie zunächst für jede erforderliche Klasse den Klassenbezeichner, die Attributbezeichner mit dem Typ für mögliche Attributwerte und die Methoden (nach dem Schema aus der Abbildung oben rechts) in die Klassenkarte ein.
2. Tragen Sie die Beziehungen zwischen den Klassen ein.

Deklaration von Klassen in der Programmiersprache Python

Der folgende Quelltext (Programmcode) legt die Klasse »Sporthalle« (Zeilen 12 bis 21) in der Programmiersprache Python fest. Anschließend wird ein Objekt aus dieser Klasse erzeugt (`halleA` in Zeile 23) und einige Methoden dieses Objekts werden aufgerufen.

```
0  # Die Namensnennung durch einen Verweis und die Lizenzangabe der
1  # ursprünglichen Urheber in den Materialien für Schülerinnen und
2  # Schüler ist erforderlich.
3  # Die Sammlung der Dokumente steht unter
4  # http://ddi.uni-wuppertal.de/material/materialsammlung/
5  # zur Verfügung.
6  #
7  # Weitere Lizenz: GPL V3.0
8  # Siehe http://www.gnu.org/licenses/gpl-3.0.html
9  #
10 # $Id: klassendef_sporthalle.py 1368 2015-12-10 22:24:28Z humbert $
11
12 class Sporthalle:
13    def __init__(self, ersterName, dieserStandort):
14       self.name= ersterName
15       self.standort= dieserStandort
16    def gibStandort(self):
17       return self.standort
18    def gibName(self):
19       return self.name
20    def setzeName(self, neuerName):
21       self.name= neuerName
22
23 halleA= Sporthalle("Halle A", "Altbau")
```

```
24 print(halleA.gibName() )
25 halleA.setzeName("Manfred-Jaeger-Halle")
26 print(halleA.gibName() )
```

Wichtig ist die Einrückung der Anweisungen durch Leerzeichen, die durch die Einrücktiefe eine Gruppierung vornehmen.

Die genaue Funktionsweise dieses kleinen Programms kann an dieser Stelle noch nicht erklärt werden. Sie können jedoch die Attribut- und Methodenbezeichner der Klasse Sporthalle wiedererkennen. Ebenfalls können Sie sehen, wie der Objektbezeichner `halleA` verwendet wird.

Aufgabe

Erstellen Sie in der Sprache Python die Deklaration der Klasse »Mannschaft« in Analogie zur obigen Definition der Klasse »Sporthalle«. Verwenden Sie dazu bitte unbedingt kariertes Papier und orientieren Sie sich beim Schreiben an den Kästchengrenzen, damit die Einrückung der Anweisungen deutlich wird.

Methoden der Klasse Spiel

Neben den bisher entwickelten Methoden, die Attributwerte verändern (`setze...`) oder zurückgeben (`gib...`), sollen noch weitere Methoden definiert werden. (Um den Sieger des Turniers ermitteln zu können, könnte es beispielsweise hilfreich sein, zunächst festzustellen, welche Mannschaft ein einzelnes Spiel gewonnen hat.)

Ein Objekt der Klasse `Spiel` soll wie folgt erstellt werden können:

```
spielAB = Spiel("09:45", "10:15", halleA, dieKaengurus, dieNowitzkis)
```

`halleA`, `dieKaengurus` und `dieNowitzkis` sind die Objekte, mit denen das Spiel `spielAB` in Beziehung steht.

Die Klasse `Spiel` soll folgende Vorgaben erfüllen:

- Der Konstruktor soll (wie oben angegeben) bereits die Werte für `beginn`, `ende`, `ort`, `mannschaftA` und `mannschaftB` als Attributwerte eintragen.
- Eine Methode `ergebnisEintragen(...)` soll das Spielergebnis eintragen, sobald das Spiel stattgefunden hat. Argumente sind `punkteA` und `punkteB`, also etwa
  `spielAB.ergebnisEintragen(32, 20)`.

- Die Methoden `gibSieger()`, `gibVerlierer()`, `gibPunkteSieger()` und `gibPunkteVerlierer()` sollen die siegreiche bzw. unterlegene Mannschaft und die erzielte Punktzahlen zurückgeben.

Arbeitsaufträge

1. Überarbeiten Sie den Konstruktor (also die Methode: `def __init__`) Ihrer Klasse `Spiel` so, dass er die oben angegebenen Vorgaben erfüllt.

   Tipp: Da die Punkte erst nachträglich über einen Methodenaufruf eingetragen werden, sollte im Konstruktor die Punktzahl jeder Mannschaft auf -1 gesetzt werden. So kann im Programm erkannt werden, ob schon ein Ergebnis eingetragen wurde oder nicht (0:0 wäre ja je nach Sportart ein mögliches Ergebnis).
2. Überarbeiten Sie die bereits erzeugten Objekte der Klasse `Spiel` in ihrem Programm so, dass diese dem neuen Konstruktor aus Aufgabe 1 angepasst sind und damit die jeweiligen Mannschaften und Hallen mit den Spielen verknüpft sind.
3. Erstellen Sie innerhalb der Klasse `Spiel` die Methode `ergebnisEintragen()`. Achten Sie auf die korrekten Einrückungen und die korrekte Verwendung der Attributbezeichner für die jeweiligen Punkte aus dem Konstruktor.
4. Aktivieren Sie für alle Spiel-Objekte die Methode `ergebnisEintragen(...)` und tragen Sie die von Ihnen gewählten Ergebnisse für alle Spiele ein.
5. Diskutieren Sie in Ihrer Gruppe, wie die Methode `gibSieger()` eines Spiels umgesetzt werden könnte, nachdem die Punkte eingetragen wurden. Notieren Sie in einer Folge von umgangssprachlichen Anweisungen, welche Schritte das Programm umsetzen müsste.

Konstruktor – die Methode `__init__`

Als **Konstruktor** wird die Methode einer Klasse bezeichnet, die automatisch beim Erzeugen (Instanziieren) eines Objekts der Klasse aktiviert wird. In Python ist der Methodenbezeichner dafür vorgegeben: `__init__`. `init` steht dabei für das Wort Initialisierung, da diese Methode bei der Objekterzeugung automatisch aktiviert wird. Wie jede andere Methode kann auch der Konstruktor `__init__` mit formalen Parametern versehen werden und mit Attributen und Attributwerten arbeiten.

Allgemeiner Konstruktor

Zur Wiederholung der Begrifflichkeiten folgt der beispielhafte Aufbau eines Konstruktors in Python:

```
def __init__(self, formalerParam1, formalerParam2, formalerParam3):
  self.attribut1= formalerParam1
  self.attribut2= formalerParam2
  self.attribut3= formalerParam3
  self.attribut4= "Ein␣beliebiger␣Text"
  self.attribut5= 361
```

Konstruktor der Klasse Spiel

Im konkreten Beispiel unseres Sportturniers sieht der Konstruktor der Klasse `Spiel` wie folgt aus:

```
def __init__(self, start1, ende1, halle1, mannschaft1, mannschaft2):
  self.beginn= start1
  self.ende= ende1
  self.halle= halle1
  self.mannschaftA= mannschaft1
  self.mannschaftB= mannschaft2
  self.punkteA= -1
  self.punkteB= -1
```

Soll das Objekt `meinSpiel` aus der Klasse `Spiel` erzeugt werden, so muss man nachsehen, welche und wieviele Parameter benötigt werden:

```
meinSpiel= Spiel("9:30", "10:00", halleAltbau, dieKaengurus, crazyFrogs)
```

Es wird deutlich, dass die Reihenfolge der aktuellen Parameterwerte mit der Reihenfolge der init-Methode übereinstimmen muss.

Import von Klassen aus einer Datei

Um Klassen zu nutzen, die in einer anderen Datei implementiert wurden, gibt es in Python den Befehl: `import`. In unserem Fall bekommen Sie die Datei `turniertabelle_ds.py`

zur Verfügung gestellt, in der verschiedene Klassen implementiert wurden, die zur Berechnung, Sortierung und Ausgabe einer Tabelle benötigt werden. Für die Nutzung in Ihrem Programm müssen Sie aus der Datei nur die Klasse `SpielTabelle` importieren. Sie bildet die Schnittstelle Ihrer Implementierung und der von uns vorbereiteten Tabellenkalkulation. Die erste Zeile Ihres Programms sollte also wie folgt aussehen:

```
from turniertabelle_ds import SpielTabelle
```

Der Name nach dem `from` ist der Dateiname (ohne `.py`) – in der Datei ist die jeweilige Klasse definiert. Hinter dem `import` steht der Klassenbezeichner für die Klasse, die aus der angegebenen Datei importiert werden soll. Durch diese Anweisung zu Beginn Ihres Programms können Sie ein Objekt der Klasse `SpielTabelle` erzeugen und ihre Methoden nutzen.

Methoden der Klasse SpielTabelle

Objekte der Klasse `SpielTabelle` haben die folgenden Methoden:

- Methode `__init__(mannschaft1, mannschaft2, mannschaft3, ...)`

  Parameter: `mannschaft1, mannschaft2, mannschaft3, ...` – die teilnehmenden Mannschaften als Objekte
  Rückgabewert: keiner
  Beschreibung: Konstruktor

- Methode `ergebnisEintragen(spiel)`

  Parameter: `spiel` – Das Objekt der Spielbegegnung, dessen Ergebnis in der Tabelle berücksichtigt werden soll
  Rückgabewert: keiner
  Beschreibung: Die Methode ändert – je nach Spielausgang – die Anzahl der Siege und die Punktdifferenz der beiden beteiligten Mannschaften und speichert sie intern in einer Tabelle.

- Methode `ausgabe()`

  Parameter: keine
  Rückgabewert: keiner
  Beschreibung: Die Methode gibt über mehrere `print`-Befehle die komplette Tabelle aus.

- Methode `gibTabellenFuehrer()`

Parameter:	keine
Rückgabewert:	Objekt der Mannschaft, die Tabellenführer ist
Beschreibung:	Die Methode überprüft, wer aktuell Tabellenführer ist und gibt diese Mannschaft als Objekt zurück.

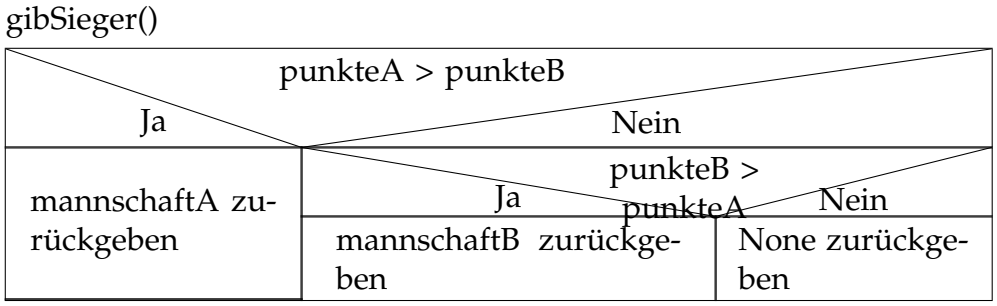
B.6 Vorhaben EF-6: Kontrollstrukturen – Arbeitsblätter

Verzweigungen

Algorithmen, wie die Methoden der Klasse `Spiel`, müssen nicht immer nur lineare Abfolgen von Anweisungen beinhalten, sondern können auch Wahlmöglichkeiten enthalten. Man spricht hier üblicherweise von **Verzweigungen**. Eine Verzweigung gibt abhängig von einer Bedingung an, welcher von zwei Algorithmenabschnitten im Folgenden durchlaufen werden soll. Durch eine Verschachtelung von Verzweigungen können auch mehr als zwei Möglichkeiten berücksichtigt werden.

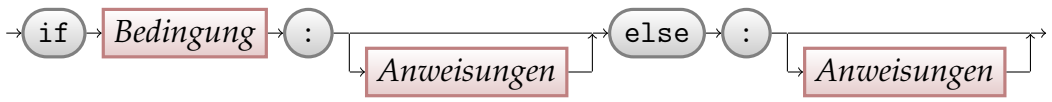
Darstellung in Struktogrammen

Eine verschachtelte Verzweigung findet sich auch bei der Methode `gibSieger()` und könnte wie folgt aussehen. Ist die Bedingung `punkteA > punkteB` erfüllt, so wird der linke Abschnitt abgearbeitet, ansonsten der rechte. Im rechten Abschnitt wird dann erneut zwischen zwei Wahlmöglichkeiten unterschieden.



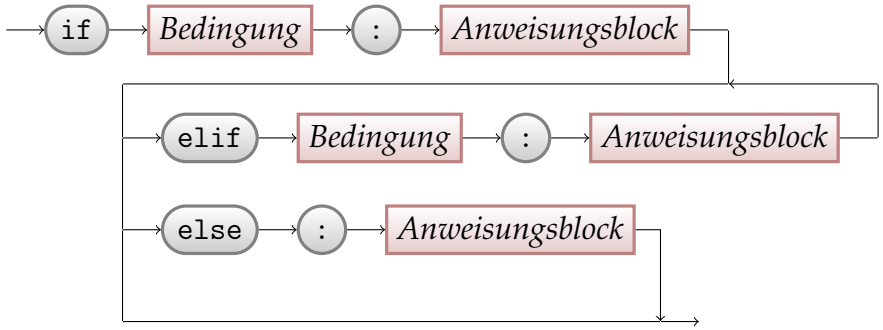
Syntax in Python

In Python dienen die Schlüsselwörter **if** (wenn), und **else** (sonst) zur Beschreibung einer einfachen Verzweigung. Die Reihenfolge kann man im folgenden Syntaxdiagramm sehen:



Sollen mehr als zwei Möglichkeiten berücksichtigt werden, muss man eine verschachtelte Verzweigung verwenden. Dafür gibt es das Schlüsselwort **elif** (sonst, wenn). Im

folgenden Syntaxdiagramm wird von drei Möglichkeiten ausgegangen. Durch weitere Verschachtelungen können allerdings auch noch mehr Möglichkeiten berücksichtigt werden.



Das oben angegebene Struktogramm der Methode `gibSieger()` könnte in Python wie folgt implementiert werden:

```
def gibSieger(self):
    if self.punkteA > self.punkteB:
        return self.mannschaftA
    elif self.punkteB > self.punkteA:
        return self.mannschaftB
    else:
        return None
```

Arbeitsaufträge

1. Integrieren Sie die oben beschriebene Methode `gibSieger()` in ihre Klasse `Spiel`.

   WICHTIG: Die Attributbezeichner Ihrer Klasse `Spiel` können sich von den oben verwendeten (punkteA, punkteB, mannschaftA, mannschaftB) unterscheiden! Passen Sie die Methode bei der Implementierung entsprechend an.

2. Testen Sie die Methode, indem Sie sich von einer ausgewählten Spielbegegnung mit einer `print`-Anweisung den Sieger ausgeben lassen.

3. Implementieren Sie analog zur Methode `gibSieger()` die Methode `gibVerlierer()`, die ebenfalls Teil der Klasse `Spiel` ist.

4. Implementieren Sie die Methoden `gibPunkteSieger()` und `gibPunkteVerlierer()` innerhalb Klasse `Spiel`. Im Gegensatz zu den beiden Methoden aus Aufgabe 1 und Aufgabe 3 sollen nicht die Mannschaften zurückgegeben werden, sondern die erzielte Punktzahl des Siegers bzw. des Verlierers.

5. Testen Sie die nun alle Methoden auf einmal, indem Sie sich von einer ausgewählten Spielbegegnung mit `print`-Anweisungen den Sieger mit seinen erzielten Punkten und den Verlierer mit seinen erzielten Punkten ausgeben lassen.

Anhang C

Rezepte

C.1 Zahl$_x$ $\rightarrow$ Dezimalzahl$_{10}$

Umrechnung von Zahldarstellungen in das Dezimalsystem

Ziel

Sichere und schnelle Umrechnung von Zahlen – benötigt werden nur die Rechenoperationen Addition und Multiplikation.

Rezept

Sollen Dualzahlen, Oktalzahlen, Hexadezimalzahlen oder . . . *in das Dezimalsystem* umgerechnet werden, so kann dies mit Hilfe des HORNER-Schemas schnell und sicher erledigt werden, da nur die Rechenoperationen Addieren und Multiplizieren benutzt werden.

Regeln zur Umrechnung in eine Dezimalzahl

Die einzelnen Ziffern der umzuwandelnden Zahl werden mit etwas Abstand aufgeschrieben. Man beginnt dann ganz links mit der höchsten Stelle. Von oben nach unten wird dann ziffernweise addiert, von unten nach schräg rechts eine Ebene höher wird mit der **Basis** des jeweiligen Zahlensystems (also 2 bei Dualzahlen, 8 bei Oktalzahlen und 16 bei Hexadezimalzahlen) multiplziert (vgl. erstes Beispiel).

Beispiele – Aufgabe

Bitte rechnen Sie zunächst alle Beispiele selbst durch und prüfen Sie, ob Ihre Ergebnisse mit den hier dargestellten übereinstimmen.

- $11101010_2 \rightarrow$

	1	1	1	0	1	0	1	0
2		2	6	14	28			
	1	3	7	14	$\cdot 2$			

Tragen Sie die fehlenden Zahlen ein und ermitteln Sie das Gesamtergebnis. Das Ergebnis ist der Dezimalwert von 11101010_2.

- $7502_8 \rightarrow$ 8

7	5	0	2
	56	488	3904
7	61	488	3906

- $3A5F_{16} \rightarrow$ 16

3	10	5	15
	48	928	14928
3	58	933	14943

- $11001_2 \rightarrow$ 2

1	1	0	0	1
	2	6	12	24
1	3	6	12	25

C.2 Dezimalzahl$_{10}$ $\rightarrow$ Zahl$_x$

Divisionsverfahren – eine Dezimalzahl in ein anderes Zahlensystem umwandeln

Ziel

Sichere Umrechnung von Dezimalzahlen – benötigt werden die Rechenoperationen Subtraktion und Division (ganzzahlig).

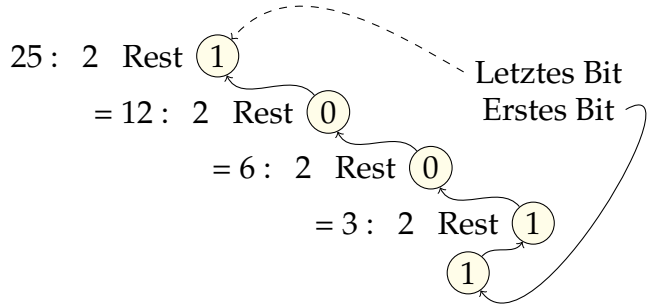
Rezept

Regeln zur Umrechnung einer Dezimalzahl

Als *Basis* wird im Folgenden die Basis des Zahlensystems bezeichnet, in das die Dezimalzahl umgewandelt werden soll: Die Dezimalzahl wird – wie in der Grundschule – fortlaufend durch die **Basis** geteilt (dividiert). Der Rest der Divisionsoperation wird rechts aufgeschrieben, das Ergebnis wird eine Zeile tiefer notiert. Mit dem Ergebnis wird ebenso verfahren. Das Gesamtergebnis ergibt sich durch das Aufschreiben der Reste – beginnend mit dem zuletzt errechneten Rest.

Beispiel – Aufgabe

Die Dezimalzahl 25_{10} soll in eine Dualzahl (Basis 2) umgewandelt werden:



Prüfen Sie das Ergebnis, indem Sie 11001_2 mit dem HORNER-Schema in eine Dezimalzahl umrechnen.

C.3 Das Verfahren von Abbott

Verfahren von* Abbott – *Texte zielgerichtet auf Objekte untersuchen

Ziel

Kandidaten für Objekte, ihre Attribute und Attributwerte sowie ihre Methoden aus einem Text heraussuchen.

Rezept

Das Verfahren nach Abbott; Objektkarten erstellen

Auf RUSSELL J. ABBOTT geht ein Verfahren zurück, das für die objektorientierte Analyse (OOA) bzw. objektorientierte Modellierung (OOM) hilfreich sein kann. Es folgt eine Zusammenfassung der drei erforderlichen Schritte[1].

Um aus einer umgangssprachlich formulierten Problembeschreibung die Objekte mit den zugehörigen Objektkarten (s. u.) zu erarbeiten, geht man wie folgt vor:

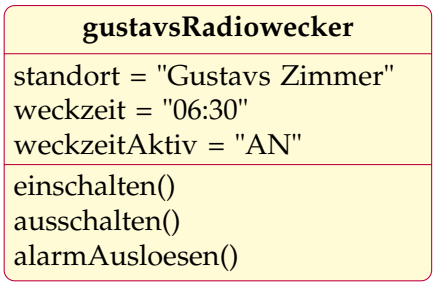
1. **Substantive (Hauptwörter) und Eigennamen herausfiltern**
   Die Hauptwörter sind mögliche *Objekte*. Meist nicht beachtet werden allerdings Mengen- und Größenangaben (»Kilogramm«), Sammelnamen (»Regierung«), Materialbezeichnungen (»Plastik«) und abstrakte Begriffe (»Liebe«, »Arbeit«). Zeitwörter (Verben), die als Hauptwörter benutzt werden (»das Betrachten eines Bildes«) werden behandelt wie die zugehörigen Zeitwörter. Gattungsnamen wie z. B. »Kraftfahrzeug«, »Säugetier« und »Einwohner« sind ebenfalls meist keine Objekte.

2. **Verben (Zeitwörter) herausfiltern**
   Sie bezeichnen häufig die Aktionen, welche von Objekten ausgeführt werden können (die sogenannten *Methoden* der Objekte). Es ist festzustellen, welchem Objekt die Methode zugeordnet werden kann.

3. **Adjektive (Eigenschaftswörter) herausfiltern**
   Sie bezeichnen häufig die »Ausprägungen« (*Attributwerte*), welche bestimmte Eigenschaften (die *Attribute*) von Objekten annehmen können. Beispielsweise wäre »ledig« ein Attributwert zum Attribut »Familienstand« oder »1216« der Attributwert des Attributs »Seitenzahl« des aktuellen Dudens. Auch hier ist wieder festzustellen, welchem Objekt der Attributwert zugeordnet und wie das zugehörige Attribut bezeichnet werden kann.

[1] Urheberin der Zusammenfassung unbekannt, hier in leicht veränderter Form wiedergegeben

Die grafische Darstellung von Objekten erfolgt durch *Objektkarten*. Eine Objektkarte wird als Rechteck mit abgerundeten Ecken gezeichnet. Die erste Zeile beeinhaltet dabei den eindeutigen Namen des Objekts. Nach einer horizontalen Trennlinie folgen zeilenweise die Attribute mit ihren jeweiligen Attributwerten. Die Methoden der Objekte sind (sofern vorhanden) von den Attributen und Attributwerten wiederum durch eine horizontale Linie getrennt.



Per Konvention beginnen **Bezeichner** für Objekte, Attribute und Methoden mit einem Kleinbuchstaben. Attributwerte werden häufig als sogenannte Zeichenketten dargestellt, die in "..." eingeschlossen sind. Es ist üblich, bei zusammengesetzten Bezeichnern neu einsetzende Worte durch Großbuchstaben hervorzuheben (siehe Beispiel).
Umlaute, Sonderzeichen und Leerzeichen sind in Bezeichnern grundsätzlich verboten. Neben den im Englischen verwendeten Groß- und Kleinbuchstaben und den Ziffern 0 bis 9 ist allenfalls noch der Unterstrich (»_«) erlaubt.

C.4 Vereinbarungen – Code-Guidelines – Bezeichner

Vereinbarungen zur Erstellung von Bezeichnern

Ziel

Verständigungsbasis zur Schreibweise, Syntax und Semantik von Bezeichnern.

Rezept

Bezeichner werden **ohne** Sonderzeichen[2] geschrieben. Werden Bezeichner aus mehreren Worten zusammengesetzt, so werden diese ohne Leerzeichen zusammengeschrieben und jedes folgende Wort beginnt mit einem Großbuchstaben wie z. B. `gibName`. Klassenbezeichner beginnen mit einem Großbuchstaben. Bezeichner für Objekte, Attribute, Methoden und Variablen, beginnen mit einem kleinen Buchstaben. Bei Attributwerten werden einzelne Zeichen und Zeichenketten in Anführungszeichen gesetzt. Zahlen werden als Zahlenwert dargestellt – Objekte werden durch Angabe des Objektbezeichners dargestellt. Für Wahrheitswerte stehen die beiden Möglichkeiten `Wahr` bzw. `Falsch` oder `Ja` bzw. `Nein` zur Verfügung.

[2] Sonderzeichen sind alle Umlaute (ä, ö, ü, ß, Ä, Ö, Ü), aber auch sämtliche Satzzeichen sowie spezielle Symbole, wie @, %, &, ^{2}, ^{3}, ...

C.5 Objekte miteinander verbinden – Objektdiagramm

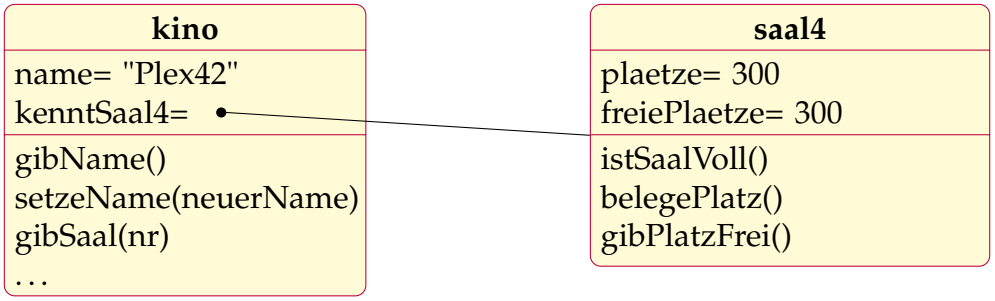
Vereinbarungen zur Erstellung von Objektdiagrammen

Ziel

Treten Objekte in Kontakt, so wird dies in einem Objektdiagramm durch Verbindungen dargestellt.

Rezept

Im Unterschied zur UML-Spezifikation wird eine Objektkarte mit runden Ecken dargestellt und der Objektbezeichner wird nicht unterstrichen. Außerdem ist es möglich, die Objektkarte um die Auflistung der Methoden zu ergänzen. Die zwei bzw. drei Bereiche der Objektkarte werden durch Linien voneinander getrennt.
Sollen in einer Objektkarte nicht alle benötigten Attribute und Methoden angegeben werden, so können diese durch drei waagerechte oder senkrechte Punkte ersetzt werden.
Werden Objektkarten für das Objektspiel erstellt, so werden die Attribute mit ihren Werten auf der Rückseite notiert, damit klar ist, dass Attribute und ihre Werte ausschließlich über Methoden zugänglich sind.
Werden Sichtbarkeiten angeben, so geschieht dies durch die Angabe von +, - oder o vor dem jeweiligen Attribut- oder Methodenbezeichner.



C.6 Das Objektspiel – Überprüfung der Modellierung

Das Objektspiel – Modellierung überprüfen

Ziel

Herausfinden, ob sich eine Szene mit den gefundenen Objekten »durchspielen« läßt.

Rezept

Objektspiel

Das Objektspiel kann für verschiedene Dinge im Bereich der objektorientierten Modellierung eingesetzt werden. So können mit ihm z. B. die Grundlagen für ein Sequenzdiagramm gelegt werden oder eine Modellierung überprüft werden.
Beim Spiel übernehmen Personen die Rolle von Objekten aus der Modellierung. Im Verlauf des Spiels wird eine Situation aus der Modellierung Schritt für Schritt durchgegangen, wobei die Objekte entsprechend agieren müssen. Zur Verdeutlichung erhalten die Objekte eine beschriftete Objektkarte. Auf der Vorderseite sind der Bezeichner des Objekts und die Bezeichner für die Methoden notiert. Auf der Rückseite ist Platz für die Attributbezeichner mit ihren Attributwerten.

Regeln

- Je nach Vereinbarung reagieren die Objekte selbstständig oder ein »Erzähler« liest schrittweise die Abfolge vor.
- Objekte können nur verbal mit Hilfe ihrer Methoden miteinander kommunizieren. Dabei ist z. B. folgende Form zu wählen: »Ich …rufe bei …die Methode …auf.«
- Änderungen an Attributwerten werden so notiert, dass der vorherige Wert durchgestrichen und der neue Wert dahinter geschrieben wird.
- Es kann nur auf Grundlage der Attributwerte gehandelt und geantwortet werden.
- Beziehungen zwischen Objekten werden mit Hilfe von Bändern verdeutlicht, die an den Objektkarten angeklebt werden. Bei gerichteten Beziehungen wird an das Band noch zusätzlich ein Pfeil gehängt.
- Objekte können nur mit Objekten kommunizieren, zu denen sie eine direkte Beziehung haben oder sie müssen über andere Objekte kommunizieren.

Alle nicht am Objektspiel beteiligten Personen haben die Aufgabe, den Ablauf genau zu protokollieren. Dazu füllen sie eine Tabelle mit folgenden Spalten aus:

Aktion von	an wen	Aktion	Parameter	Rückgabe	Bemerkung

Anhang D

LaTeX – Inhalt – Struktur – Form[1]

- Unterschied zwischen Form, Inhalt und Struktur kennen
- Benutzung von LaTeX verstehen
- Entscheiden können, wofür LaTeX geeignet ist

Wer TeX zum ersten Mal sieht, fragt sich oft besorgt, ob das wirklich ernst gemeint sei. Es fängt schon bei dem Namen an, dessen Aussprache unklar ist (er wird »tech« gesprochen). Viel schlimmer ist aber, dass man reichlich kryptische Texte schreiben muss, um auch nur einen einfachen Brief zu produzieren. Bei LaTeX (was eine »Erweiterung« von TeX darstellt) liegen die Dinge kaum besser: Der Namen erscheint eher humoristisch und die kryptischen Texte sind immernoch kryptische Texte. Nach der ersten Begegnung kehren viele Nutzer reumütig zu Word und Co. zurück.

Wer TeX häufiger nutzt, gehört aber nicht zu den bekennenden Masochisten, sondern hat seine Stärken zu schätzen gelernt. Das Programm stürzt nie (!) ab, es produziert immer exakt die gleichen Ausgaben, unabhängig von Versionen oder Druckertreibern, die Qualität ist über jeden Zweifel erhaben, es ist frei verfügbar und auch Dokumente von vielen hundert Seiten sind kein Problem (ein Beispiel ist das Skript, dass Sie gerade lesen). Der Formelsatz von TeX ist um Klassen besser als der praktisch aller anderen Programme: Eine von TeX gesetzte Formel ist ein optisches Juwel, Word verwurstet dieselbe Formel bestenfalls zu einem optischen Backstein.

Wohingegen der Kern von TeX seit mittlerweile Jahrzehnten fest und unveränderlich ist (was maßgeblich zu seiner Stabilität beigetragen hat), gibt es eine rege Szene von Benutzern, die immer neue Erweiterungen schreiben. Eine solche ist das Beamer-Paket, mit dem man TeX benutzen kann, um Präsentationen zu bauen – ein Einsatzzweck, für den TeX zwar nie vorgesehen war, bei dem es aber trotzdem recht erfreuliche Resultate vorweisen kann.

[1] Dieser Abschnitt wurde der Veranstaltung »Einführung in die Informatik – Teil 1« aus dem Wintersemester 2012/2013 von Prof. Dr. Till Tantau zu dem Thema **LaTeX** entnommen (dort: Kapitel 6) und an einigen Stellen geändert.

D.1 Inhalt – Struktur – Form

D.1.1 Drei Sichten auf einen Text

Drei grundsätzliche Dimensionen eines Dokuments.

Beispiel: Ein Beispieldokument

Für einen Kaiserschmarrn benötigt man:

1. 150g Mehl
2. 1/8l Milch
3. 3 Eier (getrennt)
4. Puderzucker und eine Prise Salz

Dieser (und auch jeder andere) Text hat

1. einen *Inhalt* – er gibt an, was der Text *bedeutet*,
2. eine *Struktur* – sie gibt an, wie der Text *aufgebaut* ist,
3. eine *Form* – sie gibt an, wie der Text *aussieht*.

Variation von Inhalt, Struktur und Form.

Beispiel: Variation des Inhalts

Für einen großen Kaiserschmarrn benötigt man:

1. 250g Mehl
2. 1/4l Milch
3. 6 Eier (getrennt)
4. Puderzucker und zwei Prisen Salz

Beispiel: Variation der Struktur

Für einen Kaiserschmarrn benötigt man 150g Mehl, 1/8l Milch, 3 Eier (getrennt), Puderzucker und eine Prise Salz.

Beispiel: Variation der Form

Für einen Kaiserschmarrn benötigt man:

1. *150g Mehl*
2. *1/8l Milch*
3. *3 Eier (getrennt)*
4. *Puderzucker und eine Prise Salz*

Ein gutes Dokumentenformat trennt Inhalt, Struktur und Form.

Ein »gutes« Dateiformat erlaubt es, *Inhalt, Struktur und Form* eines Textes *unabhängig voneinander zu notieren*.

Vorteile

- Ist die *Struktur* separat notiert, kann ein Anzeigeprogramm leicht navigieren oder ein Inhaltsverzeichnis erzeugen.
- Ist die *Form* separat notiert, lässt sie sich leicht allgemein ändern: Wird *nur einmal* notiert, wie Bibliographieeinträge aussehen, so sind sie *automatisch einheitlich*.

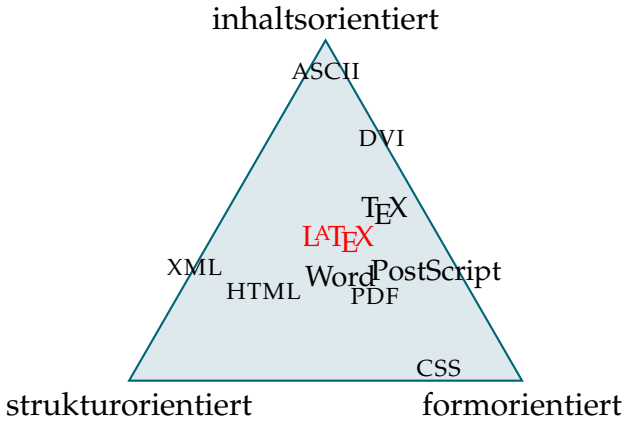
Nachteile

- Das zusätzliche Notieren von Struktur und Form macht Arbeit.

Fallen Ihnen weitere Vor- oder Nachteile ein?

D.1.2 Beschreibungssprachen

Wohin verschiedene Seitenbeschreibungssprachen tendieren.



D.1.3 Das Beispiel LaTeX

Was ist LaTeX ?

- *LaTeX* , geschrieben von Leslie Lamport, ist eine Erweiterung des Programms *TeX*, geschrieben von Donald Knuth.
- Ein *LaTeX -Dokument* ist ein *reines Textdokument*, das den *Inhalt* und die *Struktur* eines Textes enthält.
- Das LaTeX -Programm kümmert sich um *eine gute Form*.

Prinzipielles Vorgehen

- Man erstellt ein *Manuskript*. Dies ist eine reine Textdatei mit der Endung `.tex`.
- Dann benutzt man das Programm `lualatex` (eventuell sind mehrere Durchgänge nötig), um daraus eine `.pdf`-Datei zu erzeugen.
- Diese kann man dann drucken oder weiterreichen.

Vorführung des Übersetzungsprozesses an einem Beispiel

D.2 Gliederung von Dokumenten in LaTeX

D.2.1 Dokumentklassen

Der Beginn eines LaTeX -Manuskripts.

- *Am Anfang steht eine Zeile, die LaTeX sagt, welche Dokumentklasse benutzt wird. Beispiele sind:*
  - `article` *oder* `scrartcl`,
  - `report` *oder* `scrreprt`,
  - `book` *oder* `scrbook`,
  - `beamer`.
- *Die Zeile lautet dann beispielsweise wie folgt:*

```
1 \documentclass[german,11pt]{article}
```

Die Optionen in eckigen Klammern sagen LaTeX , dass der Text auf Deutsch geschrieben ist und dass eine 11pt-Schrift die Standardgröße ist.

- *Generelle Syntax für Befehle in LaTeX :*

```
\befehlsname[optionen]{argument1}{argument2}...
```

D.2.2 Die Präambel

Die Präambel und der Body.

- *Der Kopfzeile folgt die Präambel. In ihr legt man globale Einstellungen fest und lädt Erweiterungen.*
- *Nach der Präambel folgt der Körper des Manuskript. Erst hier darf der eigentliche Text stehen.*

```
\documentclass[german,11pt]{article}

% So schreibt man Kommentare in TeX

% Die Praeambel:
\usepackage{babel}               % Sprachunterstuetzung
\usepackage[utf8]{luainputenc} % Manuskript ist in Unicode
\usepackage{graphicx}            % Zum Einbinden von Graphiken

\begin{document}
% Body, der eigentliche Text
\end{document}
```

Die wichtigsten Erweiterungen, die man so braucht.

`inputenc` *Teilt TeX mit, wie der Text kodiert ist (Unicode oder* ASCII*). Sie sollten Ihre Texte immer in Unicode kodiren und müssen daher hier die Option* `utf8` *(Unicode) angeben. Die Angabe muss zur Einstellung des Editors passen.*

`luainputenc` *Wird bei LuaTeX benötigt und funktioniert dort genauso wie* `inputenc`*.*

`babel` *Lädt umfangreiche Sprachunterstützung für alle möglichen Sprachen.*

`graphicx` *Lädt Befehle, mit denen sich Graphiken gut einbinden lassen. Nachfolger von* `graphics`*.*

`xcolor` *Damit es bunt werden kann. Nachfolger von* `color`*.*

D.2.3 Abschnitte und Paragraphen

Wie man seinen Text strukturiert.

- *In den Body kommt nun der Text.*
- *In ihm finden sich besondere Befehle, die die logische Struktur des Textes angeben.*
- *Es ist die Aufgabe vom TEX (und nicht die des Autors), diese logische Struktur optisch ansprechend umzusetzen.*

```
\begin{document}

\title{Meine Facharbeit}
\author{Ich \and mein Ego}
\maketitle
\tableofcontents

\section{Einleitung}
...
\subsection{Methode}
...
\subsection{Ergebnisse}
...
\section{Zusammenfassung}
...
\end{document}
```

Die wichtigsten Kommandos zur Strukturierung

`title` *Titel der Arbeit.*

`author` *Autor der Arbeit. Mehre Autoren trennt man mit dem speziellen Befehl* `\and`.

`date` *Spezielle Datumsangabe, wenn nicht das aktuelle gewünscht wird.*

`maketitle` *Erzwingt, dass der Titel dort gesetzt wird.*

`section` *Beginn eines Abschnitts*

`subsection` *Beginn eines Unterabschnitts. Man sollte Unterunterabschnitte nicht verwenden.*

`chapter` *Beginn eines Kapitels (nur bei den Dokumentklassen* `book` *und* `scrbook`*).*

Wie gibt man Textabsätze ein?

- *Zwischen die Strukturierungsbefehle schreibt man nun den eigentlichen Text.*
- *Er besteht aus Absätzen, die durch Leerzeilen voneinander getrennt werden.*
- *Innerhalb eines Absatzes haben Leerzeichen und Zeilenumbrüche alle denselben Effekt: Sie trennen Wörter.*

```
\section{Einleitung}

Dieser Text ist Teil des ersten Absatzes. Der Umbruch
hier hat keinen Effekt, das Wort "‘hier"’ steht im fertigen
Dokument wahrscheinlich auf derselben Zeile wie "‘Umbruch"’.

Hier ist erst der zweite Absatz.              Die vielen
Leerzeichen            in         diesen Zeilen haben
denselben Effekt, als wenn man immer nur eines eintippt.
```

Besonderheiten und Probleme.

- *Als Anfänger versucht man häufig, Zeilenumbrüche und künstliche Abstände zu erzwingen. Dies ist schwierig und man sollte es bleiben lassen.*
- *In alten Systemen musste man Umlaute merkwürdig eingeben. Dies ist heute nicht mehr nötig und sollte vermieden werden.*
- *Anführungszeichen im Deutschen sehen „so“ aus. In TEX schreibt man dies aber so: "‘so"’. Sie können – da Sie mit Unicode arbeiten – auch direkt in TEX Möwchen verwenden – mit dem Ergebnis »so«.*
- *Ein Bindestrich ist kurz wie in dem Wort Binde-Strich.*
- *Ein Gedankenstrich ist – lang. Man schreibt ihn in TEX mit zwei einfachen Strichen, also `ist -- lang`.*

D.2.4 Umgebungen

Umgebungen dienen ebenfalls der Strukturierung.

- *Häufig möchte man in den Text nun Aufzählungen und nummerierte Listen einfügen.*
- *Dazu fasst man die Liste in eine Umgebung.*
- *Innerhalb der Umgebung benutzt man den `\item`-Befehl, um neue Punkte zu beginnen.*

Im Manuskript

```
Hier kommt eine nummerierte
Liste:
\begin{enumerate}
\item erster Punkt
\item zweiter Punkt
\end{enumerate}

Hier noch eine
unnummerierte Liste:
\begin{itemize}
\item ein Punkt
\item ein anderer Punkt
\end{itemize}
```

Im Ergebnis

Hier kommt eine nummerierte Liste:

1. *erster Punkt*
2. *zweiter Punkt*

Hier noch eine unnummerierte Liste:

- *ein Punkt*
- *ein anderer Punkt*

Umgebungen müssen nicht Listen enthalten.

- *Es gibt viele Umgebungen, die keine Listen darstellen.*
- *Beispielsweise schreibt man die Zusammenfassung (Abstract) in einer* `{abstract}`*-Umgebung.*

```
...
\maketitle
\begin{abstract}
  In dieser Arbeit zeige ich auf, wie man die Welt rettet und
  dabei reich wird.
\end{abstract}
```

D.3 Typographisches und Graphiken in LaTeX

D.3.1 Schriftarten

Änderung der Schriftart in Text.

- *Es gibt verschiedene Befehle, die die Schriftart ändern.*
- *Der* `\emph`*-Befehl hebt sein Argument hervor, indem er es schräg stellt (es sei denn, der Text ist schon schräg, dann macht er den Text gerade).*
- *Der* `\textbf`*-Befehl macht sein Argument fett.*
- *Der* `\textsf`*-Befehl benutzt eine Sans-Serif-Schrift für sein Argument.*
- *Mit den Befehlen* `\small`*,* `\normalsize` *und* `\large` *kann man ab einem Punkt die Schriftgröße ändern.*

Im Manuskript

```
Es ist \emph{ausgesprochen
wichtig}, dass dies
verstanden wird. Es ist aber
\textbf{typographisch
schlecht}, Wörter im normalen
Text fett zu setzen.
```

Im Ergebnis

Es ist ausgesprochen wichtig, *dass dies verstanden wird. Es ist aber* ***typographisch schlecht****, Wörter im normalen Text fett zu setzen.*

Man kann auch die Schrift generell ändern.

- *Es gibt Erweiterungspakete wie* `times`*, mit denen man die Schriftart des gesamten Textes ändern kann.*
- *Es ist allerdings schwierig, im Text mehrere unterschiedliche Schriften zu mischen . . . und das ist auch gut so.*

D.3.2 Tabellen

Tabellen haben eine eigene Syntax.

- *Für Tabellen benutzt man die* `{tabular}`*-Umgebung,*
- *die als Argument eine Formatierungsvorschrift bekommt.*
- *Jede Zeile wird durch* `\\` *beendet.*
- *Jede Zelle wird durch* `&` *beendet.*

Im Manuskript

```
\begin{tabular}{rcc}
  \emph{Spezies} & \emph{Anzahl} & \emph{cm} \\\hline
  Hund           & 100           & 7 \\
  Katze          & 50            & 37 \\
  Maus           & 6             & 47
\end{tabular}
```

Im Ergebnis

Spezies	Anzahl	cm
Hund	*100*	*7*
Katze	*50*	*37*
Maus	*6*	*47*

D.3.3 Mathematik

Die Hauptstärke von TEX: Mathematik.

- *TEX setzt Formeln viel besser, als das die meisten Menschen können.*
- *Mathematischer Text wird in* `$`*-Zeichen eingeschlossen.*
- *Es gibt eine eigene Syntax, wie man Text aufschreibt:*
  - *Tiefgestelltes wird durch* `_` *eingeleitet.*
  - *Hochgestelltes wird durch* `^` *eingeleitet.*
  - *Sonderzeichen werden durch spezielle Befehle erzeugt.*

Beispiele von mathematischem Text.

Im Manuskript

1. `$a^2 + b^2 = c^2$`
2. `$\alpha + \beta = \gamma^2$`
3. `$\sum_{i=1}^n i = n(n+1)/2$`
4. `$\sum_{i=1}^\infty \frac{1}{i^2} = \frac{\pi^2}{6}$`
5. `$\lim_{n\to\infty} \frac{1}{n^2} = 0$`
6. `$$\int_{-\infty}^\infty e^{-x^2} dx < \infty$$`

Im Ergebnis

1. $a^2 + b^2 = c^2$
2. $\alpha + \beta = \gamma^2$
3. $\sum_{i=1}^{n} i = n(n+1)/2$
4. $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$
5. $\lim_{n\to\infty} \frac{1}{n^2} = 0$
6. $$\int_{-\infty}^{\infty} e^{-x^2} dx < \infty$$

D.3.4 Graphiken

Externe Graphiken lassen sich leicht einbinden.

- *Der Befehl* `\includegraphics` *erlaubt es,* `.pdf`*-Graphiken und* `.jpg`*-Graphiken einzubinden.*
- *Als Optionen gibt man die gewünschte Größe an.*
- *Als Parameter gibt man den Dateinamen an.*

```
Folgende Graphik zeigt den Verlauf der Kurve:

\includegraphics[width=6cm]{graphik1.pdf}

Im Folgenden...
```

Beispiel einer Graphik in einer `figure`-Umgebung

- *Es bietet sich an, Graphiken in {`figure`}-Umgebungen einzuschließen. Dann kann man ihnen eine Überschrift geben und es sieht hübscher aus.*

```
...
und somit wird alles gut.

\begin{figure}
  \includegraphics[width=6cm]{graphik1.pdf}
  \caption{Verlauf der Kurve.}
  \label{graphik1}
\end{figure}

Wie wir in Abbildung \ref{graphik1} sehen, ist es nicht so,
dass...
```

D.4 Präsentationen erstellen in LaTeX

Auch Präsentationen lassen sich mit LaTeX erstellen.

- *Präsentationen lassen sich mit Hilfe von Erweiterungen von LaTeX erstellen,*
- *bei Benutzung von pdfLaTeX empfiehlt sich `beamer`.*
- *Es gibt zwei wesentliche Änderungen gegenüber normalen Dokumenten:*
  1. *Als Dokumentklasse muss man `beamer` angeben.*
  2. *Jede »Folie« kommt in eine {`frame`}-Umgebung. Diese nimmt als Parameter eine Folienüberschrift.*
- *Das Aussehen der Präsentation kann man durch Änderung des »themes« leicht ändern.*

D.5 Zusammenfassung zu LaTeX

1. *Dokumente haben einen Inhalt, eine Struktur und eine Form.*
2. *LaTeX ist ein vollkommen stabiles System, das Manuskripte in druckfertige Dokumente und Präsentationen verwandelt.*
3. *Die Syntax ist gewöhnungsbedürftig, aber dann gut benutzbar.*
4. *Besondere Stärken sind mathematischer Formelsatz und gutes Layout.*
5. *Es ist schwierig, neue Layouts zu erzeugen.*

Literatur

Bertram, Nika (2014). *60. Todestag Alan Turing: Turing Bytes – Botschaften aus einer unsichtbaren Welt*. Hrsg. von Natalie Szallies. Wiederholung der Sendung am 31.05.2016 – Regie: Matthias Kapohl. URL: https://is.gd/J2PvAf (besucht am 03. 09. 2016).

Dreschler-Fischer, Leonie (2000). »Der Gott der Informatik. Gott und das Internet? Gott und künstliche Intelligenz?« In: *Der »gott« der Fakultäten*. Hrsg. von Heiner Adamski, Axel Denecke und Wilfried Hartmann. Berlin: LIT Verlag, S. 159–177. ISBN: 3-82584935-X.

Hofstadter, Douglas R. (1985). *Gödel, Escher, Bach: ein endloses geflochtenes Band*. 5. Aufl. Original: Gödel, Escher, Bach: an Eternal Golden Braid, 1979 im Verlag Basic Books, New York. Stuttgart: Klett-Cotta. ISBN: 3-608-93037-X.

Menabrea, Luigi F. (1842). »Sketch of The Analytical Engine. Invented by Charles Babbage. With notes upon the Memoir by the Translator Ada Augusta, Countess of Lovelace«. In: *Bibliothèque Universelle de Genève* No. 82. URL: http://www.fourmilab.ch/babbage/sketch.html (besucht am 07. 02. 2015).

Miller, George Armitage (1956). »The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information«. In: *The Psychological Review* 63, S. 81–97. URL: http://web.archive.org/web/20080312114750/http://www.musanim.com/miller1956/ (besucht am 30. 04. 2015).

MSW-NW (2013). *Kernlehrplan Informatik für die gymnasiale Oberstufe*. MSW-NW – Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen. URL: http://is.gd/HbEFHU (besucht am 19. 02. 2016).

Pieper, Johannes und Dorothee Müller, Hrsg. (2014). *Material für den Informatikunterricht*. Arnsberg, Dortmund, Hamm, Solingen, Wuppertal. URL: http://uni-w.de/1t (besucht am 29. 04. 2016).

Rhia, Herrad (2012). *Die Zahlenzauberin – Das Leben der Augusta Ada Lovelace*. Radiobeitrag. URL: https://is.gd/tqbI3E (besucht am 03. 09. 2016).

Stephenson, Neal (1995). *Shnow Crash*. TB 45302. Aus dem Amerikanischen von Joachim Köhler. Die Originalausgabe erschien 1992. München: Goldmann. ISBN: 3-442-45302-X.

Wegner, Bettina (1978). *Sind so kleine Hände*. Songtext. URL: http://is.gd/0cFNfA (besucht am 27. 07. 2015).