Grundlagen der Programmkonstruktion

Kontrollfragen

Kapitel 1 - Maschinen und Programme

Was ist eine Deklaration bzw. Definition?
Eine Deklaration legt den Typ, Name und bzw. in manchen Fällen die Sichtbarkeit einer Variablen fest.(z.B.: private int zahl) Wird einer Variablen zusätzlich ein Wert zugewiesen spricht man von einer Definition.

In der Deklaration wird der Name und der Datentyp vereinbart, beim Aufruf wird außerdem noch Speicherplatz reserviert
Initialisierung = Erstmaliges Belegen eines Wertes auf einen Speicherplatz.

Definition = Deklaration + Initialisierung.

Wozu braucht man Variablen und was haben Variablen mit dem Speicher eines Computers zu tun?
Variablen sind benannte Speicheradressen, in denen Werte eines bestimmten Typs gespeichert werden können.
Der Name Variable ergibt sich daraus, dass die Variable zur Programmlaufzeit zu unterschiedlichen Zeitpunkten unterschiedliche Werte eines Typs enthalten kann.
Der Wert aller Variablen, der Objektzustand, ähnelt dem Zustand des Speicherinhaltes eines Computers.

Was haben Nachrichten mit Methoden und Funktionen zu tun, und wie hängen diese Begriffe mit den Befehlen eines Prozessors zusammen?
Methoden kommunizieren mit Methodenaufrufen miteinander. Diese Methodenaufrufe beinhalten den Methodennamen (Empfänger) und evtl. Parameter (Nachrichteninhalt). Eine aufgerufene Methode kann der (Absender-) Methode eine weitere Nachricht schicken, sofern sie einen Rückgabetyp besitzt. Diese Rückgabe wird mit dem Schlüsselwort “return” im Quellcode explizit angegeben.

Wodurch unterscheidet sich ein formaler Parameter von einem aktuellen Parameter bzw. Argument? Bestehen diese Unterschiede auf syntaktischer oder semantischer Ebene?
Der formale Parameter besteht nur temporär, während er von der Methode benutzt wird und die Methode aktiv ist. Sobald die Methode zum Aufrufer zurückkehrt, enthält der formale Parameter keinen Wert mehr. Daher ist der formale Parameter nur im Methodenrumpf gültig. Außerdem

Kommentar [1]: modmew8:
Ob unter Deklaration und Definition mit einer Wertzuordnung tatsächlich unterschieden wird, bin ich skeptisch, da die Frage eher beide Begriffe zusammenpackt...

christoph.stampfel:
Irgendwie ist das ganze umstritten, damals in der HTL hat man mir schon gesagt, dass teilweise zwischen den Begriffen nicht unterschieden wird. Irgendwo im Skriptum habe das was ich hier geschrieben habe gefunden.

Anonymous:
Beide Begriffe beschreiben grundsätzlich verschiedenes. Bei der Deklaration wird lediglich ein Speicherbereich für eine Variable reserviert, wobei der Inhalt dieser Speicherstelle praktisch zufällig ist. Unter dem Definieren einer Variable versteht man schlicht und einfach eine Wertzuweisung.

tanja.travnicek:
stimmt, laut Skriptum Seite 14

Anonymous:
Ich würde mich bei den beiden Begriffen nicht auf Variablen beschränken. Deklaration stimmt soweit (wie schon gesagt laut Seite 14 im Skriptum). Definition hingegen bedeutet dass man ALLE notwendigen Eigenschaften festlegt und sich diese auch nicht ändern. Bei einer Variable wäre das unsinnig, da (wie der Name schon sagt) der Wert von Variablen sich ändern kann. Definition wäre bei Methoden richtig: Der Methodenkopf enthält alle Eigenschaften einer Methode, das heißt man kann die Methode damit definieren.

können formale Parameter nur von den Anweisungen in der selbigen Methode „gesehen“ werden. Unterschied liegt auf syntaktischer Ebene.

Beispiel:

- **formaler Parameter** — Bezeichner, der in einer Methode verwendet wird, um einen Wert aufzunehmen, der an die Methode vom Aufrufer übergeben wird.
    - Zum Beispiel ist betrag ein formaler Parameter von verarbeiteEinzahlung (int betrag)
- **aktueller Parameter** — der tatsächliche Wert, der an die Methode durch den Aufrufer übergeben wird.
    - Zum Beispiel werden die 200, die beim Aufruf an die Methode verarbeiteEinzahlung(int betrag) übergeben werden, als aktueller Parameter bezeichnet.

Wozu braucht und woran erkennt man Kommentare?
Kommentare werden benötigt um ein Programm leserlicher zu machen.
Da im Laufe der Zeit, Programmierer die sich an einem Programm beteiligen, wechseln können, ist es hilfreich seine Absichten in Kommentaren zu hinterlegen oder auch um sich selbst nach längerer Zeit wieder in einem Programm zurechtzufinden.

In manchen Fällen können auch Zusicherungen mittels Kommentaren gemacht werden.
Man erkennt Kommentare an der (meistens) natursprachlichen Syntax und an der für die jeweilige Programmiersprache relevanten Anfangszeichen (z.B. //, /*, --, #).

Kommentare werden (in Java) folgendermaßen geschrieben:

```
// Kommentar

/*
 * Mehrzeiliger Kommentar
 */
```

Was sind Klassen und Objekte und welche Beziehung gibt es zwischen diesen Begriffen?
Eine Klasse ist eine Beschreibung der Struktur eines Objektes.
Durch eine Klasse werden somit gleichartige Objekte beschrieben.

Objekte sind Instanzen einer Klasse, d.h. konkrete Ausformungen von Klassen, die sich in einem konkreten Zustand befinden (Variablen sind gesetzt u.ä.).

Was macht ein Konstruktor und wie unterscheidet er sich von einer Methode?
Ein Konstruktor wird automatisch beim Erzeugen eines neuen Objektes mit dem new-Operator aufgerufen. Er wird vorwiegend für Initialisierungsarbeiten eines Objektes verwendet. Er unterscheidet sich von einer normalen Methode der Klasse dadurch, dass er keinen Rückgabewert besitzt und den gleichen Namen wie die Klasse besitzen muss.

Was versteht man unter dem Initialisieren eines Objekts oder einer Variablen?
Bei einer Variable versteht man unter “initialisieren” das erstmalige Zuweisen eines Wertes zu einer Variable. Beim Objekt bezeichnet “initialisieren” die Herstellung eines Initialzustandes, was (in Sprachen wie Java) durch den Aufruf eines Konstruktors geschieht.

Welche Beziehungen bestehen zwischen den Begriffen Datenkapselung, data hiding und Datenabstraktion?
Die Eigenschaften eines Objekts, Variablen und Methoden zu einer Einheit zusammenzuführen nennt man Datenkapselung. Zusammen mit data hiding, dem Verstecken von Details durch private, spricht man von Datenabstraktion.

Was ist eine Schleifenbedingung, ein Schleifenzähler, ein Schleifenrumpf und eine Iteration?
- Die Schleifenbedingung legt fest unter welchem Umstand die Schleife wiederholt ausgeführt werden soll (wenn Schleifenbedingung boolean == true).
- Als Iteration bezeichnet man **einen** Durchlauf der Schleife.
- Der Schleifenzähler ist eine (häufig lokal deklarierte) Variable (=nur innerhalb der Schleife verfügbar; (kommt darauf an, was für ein Schleifentyp verwendet wird)), die vor der ersten Iteration initialisiert wird.
- Der Schleifenrumpf ist der Anweisungsblock der schrittweise durchlaufen und wiederholt wird.

Wie hängen bedingte Anweisungen mit Programmzweigen und Fallunterscheidungen zusammen?
Nur durch bedingte Anweisungen wie if(..) … else … ist es möglich Programmzweige und Fallunterscheidungen zu vollführen. (bzw. switch)

Was ist ein Bit, ein Byte und ein Wort, und warum hat ein Kilobyte 1024 Byte?

Bit	binäre Ziffer(0 oder 1)
Byte	8 Bit ergeben 1 Byte
Wort	Man verwendet eine fixe Anzahl von Bits als grundlegende Einheit für die Darstellung von Daten - Wird auch als Maschinenwort bezeichnet. (typischerweise zwischen 16 und 64b)

1 Kilobyte = 1024 Byte … weil 2^{10} Byte = 1024 Byte
Nicht im Skriptum: 2^{10} wurde gewählt, weil diese Zweierpotenz der Zahl 1000 am nächsten liegt, und weil die Adressierung von Adressen am einfachsten ist, wenn diese in Schritten von Zweierpotenzen liegen.

Was ist ein Algorithmus, und wodurch unterscheidet sich ein Algorithmus von einem Programm bzw. einer Methode?
- Ein Algorithmus ist eine abstrakte Handlungvorschrift aus endliche vielen Schritten, die ein Problem löst.
- Unterschied: Methoden/Programme müssen nicht terminieren, und keine Lösung liefern. Wobei Methoden/Programme üblicherweise die konkrete Umsetzung eines Algorithmus sind.

Aus welchen Teilen setzt sich die Von Neumann-Architektur zusammen? Wodurch unterscheidet sie sich von der Harvard-Architektur?

- ALU(Arithmetic Logic Unit), Control Unit, Memory, I/O Unit (Input - output),
- Im Gegensatz zur Harvard-Architektur liegen Befehle und Daten nicht in getrennten Speichern

Welche Schritte werden von einer Von Neumann-Architektur ständig wiederholt ausgeführt, welche von einem Interpreter?

- Von Neumann-Architektur
  - PC zeigt auf Adresse → Befehl der Adresse aus Speicher in Control Unit gelesen
  - PC erhöht und zeigt nun auf nächsten Befehl
  - Befehl wird von Control Unit interpretiert und ausgeführt
- Interpreter
  - Genau dieselben Schritte wie in der Von Neumann-Architektur
  - Der Unterschied liegt darin, dass der Interpreter nicht auf einer realen sondern auf einer abstrakten Maschine operiert.

Was ist eine Assembler-Sprache?

Befehle werden durch Maschinenworte dargestellt, also durch Bitmuster, mit denen Maschinen einfach umgehen können, Menschen aber nicht. Für die Hardware-nahe Programmierung verwendet man daher Assembler- Sprachen und nicht direkt Maschinen-Sprachen. Eine ***Assembler-Sprache*** stellt Befehle, Adressen und Daten durch für Menschen besser lesbare Symbole dar, und ein Assembler übersetzt die Symbole in die eigentlichen Befehle.

Warum verwenden wir abstrakte Maschinen wie die JVM?

Abstrakte Maschinen wie die JVM sind für viele reale Maschinen implementiert.
Ein Programm, dass für die abstrakte Maschine geschrieben wurde, kann auf allen realen Maschinen ausführen für die die abstrakte Maschine implementiert wurde.
Ein solches Programm ist damit *portabel*.

Was sind Berechnungsmodelle und wie hängen sie mit Programmiersprachen zusammen?

- Berechnungsmodelle sind Abstrakte Maschinen mit hohem Abstraktionsgrad
- Berechnungsmodelle unterliegen keinen technischen Grenzen
- Hinter jeder Programmiersprache steckt ein Berechnungsmodell
- stellen die formalen Grundlagen für Programmiersprachen zur Verfügung (was ist theoretisch möglich?)

Welche Eigenschaften von Objekten helfen dabei, Objekte als abstrakte Maschinen zu betrachten? Was ist eine Komponente?

- Eigenschaften
  - Objektzustand
  - Objektverhalten
  - Objektidentität
- Komponenten
  - Zusammengehörige Teile einer Software

Kommentar [2]: hektortrichter:
was ist ein interpreter und wo steht das im skriptum? direkt bei der neumann architektur ist das ja nicht angeführt

Anonymous:
Skriptum: S.47 1.4.4 Übersetzung und Ausführung

andreas.taranetz:
Marked as resolved

andreas.taranetz:
Re-opened

- Bei kleineren Teilen werden sie auch Module genannt.

Wie stehen die Begriffe Architektur, Implementierung und Schnittstelle sowohl in der Hardware als auch in der Software zueinander?

- Architektur
  - Beschreibung der wichtigsten Teile der Hard- bzw. Software und in welcher Beziehung sie zueinander stehen.

- Implementierung
  - Ist die konkrete Umsetzung von Problemen in Programmcode.

- Schnittstelle
  - Bestimmt das Zusammenspiel der Komponenten in Form von Verträgen, an die diese sich in der Kommunikation halten müssen.

Können Sie Beziehungen zwischen den Begriffen Objektzustand, - verhalten und -identität einerseits und Daten, Algorithmen und Umgebung andererseits erkennen?
Jedes Objekt hat eine *Identität*, die es in einer *Umgebung* eindeutig identifiziert. Das *Verhalten,* also die Zustandsänderungen, wird durch Methoden (Implementierung eines *Algorithmus*) definiert. Der *Objektzustand* (=*Daten*) durch die Werte der gespeicherten Variablen.

Was bedeuten Syntax, Semantik und Pragmatik, und wozu verwendet man eine Grammatik?

- *Syntax*: regelt den Aufbau einer Sprache (wird in der Grammatik festgelegt)
- *Semantik*: legt die Bedeutung der Sprachen fest
- *Pragmatik*: untersucht das Verhältnis der Sprache zum Sprecher und zum Angesprochenen
- *Grammatik*: regelt die Wohlgeformtheit von Konstrukten einer Sprache. Schreibt beispielsweise vor, wie Deklarationen, Definitionen und Anweisungen aufgebaut sind (z.B.: dass einfache Anweisungen mit Strichpunkt enden und Rümpfe von Methoden in geschwungene Klammern eingeschlossen sind)

Wozu dienen deklarierte Typen?

- Besseres Verständnis (erhöhte Lesbarkeit von Programmen)
- Wertebereich der Variable bekannt
- Ermöglichen es einem Compiler, statische Typüberprüfungen durchzuführen um somit Fehlern bereits im Vorfeld vorzubeugen.
- Optimale Ressourcennutzung

Was unterscheidet einen Compiler von einem Interpreter?

- Compiler
  - Compiler übersetzen den Quellcode in den Zielcode
  - Typischerweise ist der Zielcode entweder Maschinencode oder ein Zwischencode der von einem Interpreter zur Laufzeit übersetzt (interpretiert) wird
  -

Kommentar [3]: hektortrichter:
gemeint sind hier wohl eher ausdrückte, befehle, methoden, was auch immer. aber "der sprachen"? ist das richtig?

konzi.h:
Würd ich allgemein schon so sagen... Wenn du nur Programmiersprachen beschreiben willst dann sollte Bedeutung der Ausdrücke genügen... Ist sicherlich beides richtig

hektortrichter:
weil so wies da steht würde ich lesen: "die bedeutung von java" und java hat ja wohl kaum wirklich eine bedeutung in dem sinn

andreas.taranetz:
aber ein Ausdruck in Java hat einen Sinn

hektortrichter:
ja schon.. genau das war doch meine frage. demnach wäre was hier steht aber ein wenig verwirrend..

andreas.taranetz:
ich würde sagen die Semantik legt die bedeutung eines Ausdrucks der Sprache fest

andreas.taranetz:
Marked as resolved

andreas.taranetz:
Re-opened

konzi.h:
Im Allgemeinen (also für alle Sprachen oder ähnliche Gebilde) gilt, dass die Syntax bestimmt wie Zeichen angeordnet sein dürfen. und die Semantik beschreibt was die Zeichen allgemein ausdrücken wollen. (=Die Bedeutung)

Aber wie bereits gesagt is das eigentlich ziemlich egal. Wenn du bei der Frage

Syntax=Aufbau; und Semantik=Bedeutung schreibst wärs auch richtig.

- Interpreter
    - Er interpretiert den Quell- oder Zwischencode und führt diesen auf der realen Maschine aus.
    - Im Fall von Java wird der Zwischencode (Bytecode) durch die Virtuelle Maschine interpretiert.

Was versteht man unter Quell-, Ziel- und Zwischencode?
Quellcode: Der vom Programmierer geschriebene Code, z.B. in Java
Zwischencode: Durch die JVM ausführbarer Code - noch nicht in Maschinensprache!
Zielcode: Der von der JVM erzeugte Maschinencode.

Der Zwischencode kann auch übersprungen werden, indem man mittels eigener Compiler (zB gjc) direkt von Quell- in Zielcode übersetzt, wodurch kein Interpreter mehr benötigt wird.

Wie kommen Java-Programme üblicherweise zur Ausführung (Übersetzung und/oder Interpretation)?
    - Übersetzung des Quellcodes in den Bytecode mittels CompileIm Fall von Java übersetzt der Compiler den Quellcode in einen Zwischencode, auch Java-Bytecode genannt.
1. r: **javac** Datei.java (erstellt Datei.class)
2. Interpretation des Bytecodes durch die JVM mittels **java**

Wofür steht die Abkürzung JIT? In welchem Zusammenhang ist sie von Bedeutung?
(seite 64 im Skript)
JIT … Justin Time

Im Gegensatz zu herkömmlichen Compilern übersetzen Just-in-Time Compiler Programmstücke nur bei Bedarf. Programmteile die selten ausgeführt werden, werden unter Umständen nur interpretiert.

Wie wirken sich Laufzeitfehler im Vergleich zu Fehlermeldungen und Warnungen aus?
Der Unterschied zwischen Laufzeitfehler und Fehlermeldungen oder Warnungen ist der, dass Fehlermeldungen oder Warnungen schon während der Kompilierung auftreten und auf einen syntaktischen (oder semantischen, falls vom Compiler feststellbar) Fehler hinweisen. Laufzeitfehler treten erst während der Ausführung (Laufzeit) auf und weisen meistens auf semantische Fehler hin, welche vom Compiler nicht entdeckt werden können.
Ein Code kann trotz Warnungen ausgeführt werden. Diese sollen den Programmierer lediglich auf Fehler aufmerksam machen, die eventuell während dem Ausführen auftreten könnten.

Wann ist etwas statisch und wann dynamisch?
Der statische Typ einer Variablen v ist der Typ, mit dem die Variable im Quelltext der Klasse deklariert wurde.
Datenabstraktion Der dynamische Typ einer Variablen v ist der Typ des Objektes, das momentan konkret in der Variablen v gehalten wird.

...das Wort statisch bezieht sich darauf, dass etwas vor der Ausführung passiert, während dynamisch sich auf etwas zur Laufzeit bezieht. (Skriptum Seite 48)

Wann spricht man von statischer und wann von dynamischer Semantik?

- Statische Semantik
  - Sie beschreibt welche Fälle zu einem übersetzten Programm bzw. welche Fälle zu Fehlermeldung führen.
- Dynamische Semantik
  - Sie beschreibt, wie sich ein Programm zur Laufzeit verhält.
  - Quell- und Zielcode müssen nur hinsichtlich der dynamischen Semantik äquivalent sein.
  -

Was ist eine Programmoptimierung und wer macht diese?
Bei einer Programmoptimierung wird der Programmcode analysiert und in einen semantisch äquivalenten aber effizienteren Programmcode umgewandelt. Wichtig ist, dass die Semantik erhalten bleibt, da sich ansonsten das Programmverhalten ändern würde.
Solche Programmoptimierungen nimmt in der Regel ein Compiler beim Übersetzen eines Programmes vor.
Optimierungen können aber auch durch den Programmierer u.a. durch die Verwendung geeigneter(er) Datenstrukturen oder besserer Algorithmen vorgenommen werden.

Was unterscheidet formale von natürlichen Sprachen?
Formale Sprachen ähneln eher den in der Mathematik verwendeten Modellen und Ideen. Die Präzision steht im Vordergrund. Natürliche Sprachen besitzten weniger Formalismen, da der Gesprächspartner Intelligenz besitzt. Rechner können nur formale Sprachen fehlerfrei verstehen, da natürliche Sprachen mehrdeutige Aussagen besitzen.

Was versteht man unter Abstraktion, und welche Arten davon haben wir schon kennengelernt?
Bei der Abstraktion eines Problems werden nur wesentliche Informationen übernommen, damit reduziert sich die zu bearbeitende Datenmenge und das Problem wird vereinfacht.

- Nebensächlichkeiten treten in den Hintergrund
- Konzentration auf das Wesentlichehttps://docs.google.com/document/d/13hi-vx1_DZbJHhn-L9azBAgmrSw3-tbC21u3nqRvPvY/edit?pli=1
- komplexere Zusammenhänge können ohne Abstraktion gar nicht mehr verstanden werden, bzw. ist die Fehleranfälligkeit wesentlich größer

Beispiele:
- Datenabstraktion
- Funktionsabstraktion

Wodurch unterscheiden sich reine Funktionen von Methoden, Prozeduren und Routinen?
Funktionen
Bei Funktionen hängt der Rückgabewert nur von den Eingabeparametern ab, der allerdings nicht zwingend nötig ist. D.h. Funktionen haben keine Seiteneffekte (bis auf ihren evtl. return-Wert).
Funktionen verändern keine Werte. Sie können diese nur abrufen.

Methoden

Methoden verüwenden einen Eingabewert und geben einen Ausgabewert zurück, Seiteneffekte sind möglich, aber selten.

Prozeduren
Prozeduren funktionieren wie Zuweisungen oder andere Ausdrücke mit Seiteneffekt in Java. z.B. Parameter deren Wert auch wieder zurückgegeben wird - Call by Reference oder Durchgangsparameter

Im Gegensatz zur Funktion liefert eine Prozedur keinen Rückgabewert auf direktem Weg.[1]

Routinen = Prozeduren

Welche Programmierparadigmen haben wir unterschieden? Wodurch zeichnen sie sich jeweils aus?

- **Imperative Sprachen**
  An oberster Stelle steht die Möglichkeit, neue Werte an Variablen zuzuweisen. (Befehle und Variablen)
  - Prozedurale Sprachen
    Abstraktion durch Prozeduren mit Seiteneffekten.
  - Objektorientierte Sprachen
    Objekte und Datenabstraktion wichtiger als Abstraktion über Prozeduren.
- **Deklarative Sprachen**
  Berechnungsmodelle an mathematische Modelle angelehnt. Es existieren keine Zuweisungen, welche Werte von Variablen überschreiben.
  - Funktionale Sprachen
    Abstraktion durch Funktionen.
  - Logikorientierte Sprachen
    Programmausführung gleichbedeutend mit Beweis einer logischen Aussage. Ähnlich wie prozedurale Programme jedoch werden Werte nur freien Variablen zugewiesen.

Wozu dient der Lambda-Kalkül? Welche Eigenschaften hat er, und warum ist er in der Informatik so bedeutend?
Der Lambda-Kalkül bildet eine formale Basis für Funktionen und daher für praktisch alle Programmiersprachen.

Eigenschaften:

[1] http://de.wikipedia.org/wiki/Prozedur_(Programmierung)#cite_note-0

Kommentar [4]: flo.groh:
ist das nicht genau umgekehrt?
Anonymous:
nein grundsätzlich stimmt das schon
segfaulthunter:
Das stimmt so nicht, nur bei REINEN Funktionen gibt es keine Seiteneffekte und sie hängen nur von den Argumenten ab.

Kommentar [5]: Anonymous:
welcher idiot verschiebt das immer zum Halteproblem -.-

Kommentar [6]: christoph.stampfel:
Konkrete Erklärung nicht im Skriptum gefunden.
zachariass:
Jein... Das Skriptum definiert eine Funktion als Unterprogramm ODER Routine.
benjamin.pazdernik:
Sind hier die allgemeine Unterschiede gemeint oder Java-spezifische?
rpuenguentzky:
Naja, da die Frage ist, wie sich Funktionen vom Rest unterscheiden: Dadurch, dass sie außer evtl. im return-Wert keine Seiteneffekte haben und somit den Programmzustand nicht beeinflussen. Methoden, Prozeduren und Routinen sind ja wirklich fast dasselbe...

- *Turing-Vollständigkeit*[2]
  Alles, was berechenbar ist, ist auch mittels Lambda-Kalkül berechenbar.
- *Endlos-Reduktionen*
  Manchmal sind β-Reduktionen endlos wiederholt anwendbar. Solche endlosen Reduktionen sind in Turing-vollständigen Systemen prinzipiell nicht vermeidbar.
- *Reihenfolge von Reduktionen*
  Die Reihenfolge von Reduktionen spielt in der Regel keine Rolle, außer ein Teilausdruck wird endlos reduziert.
- *Funktionen erster Ordnung*
  Funktionen werden wie Daten behandelt und sind dadurch Elemente erster Ordnung
- *Keine Kontrollstrukturen*
  Kontrollstrukturen können durch "Funktionen erster Ordnung" ersetzt werden.
- *Currying*
  Eine Funktion wird im Lambda-Kalkül immer nur auf ein einziges Argument angewandt. Durch "Funktionen erster Ordnung" ist dies keine Einschränkung.

  z.B. λu.(λv.(v u)) … Könnte als Funktion mit 2 Parametern betrachtet werden.

Was ist ein λ-Ausdruck? Welche Arten von λ-Ausdrücken kann man unterscheiden?
(siehe Skriptum Seite 54)
Exp = Menge von syntaktisch richtigen λ-Ausdrücken
Basis davon ist V - eine unendliche Menge von Variablen

- Variable: v ∈ Exp wenn v ∈ V
- Funktionsanwendung: f e ∈ Exp wenn f ∈ Exp und e ∈ Exp
- Funktionsabstraktion: λv.f ∈ Exp wenn v ∈ V und f ∈ Exp

Was versteht man unter freien bzw. gebundenen Variablen und was unter Ersetzung?

- **Gebundene Variable**
  Stehen für Argumente einer Funktion
- **Freie Variable**
  Alle Variablen die nicht gebunden sind.
- **Ersetzung**
  Unter Ersetzung versteht man das Ersetzen eine jedes Vorkommens einer freien Variable durch einen λ - Ausdruck.

Was bewirkt eine α-Konversion bzw. beta- und η-Reduktion?

α-Konversion	λv.f ↔ λu.[u/v]f	*Umbenennung von gebundenen Variablen*
β-Reduktion	(λv.f)e → [e/v]f	*Funktionsanwendung*
η-Reduktion	λv.(f v) → f	*Erweiterungsregel*

Kommentar [7]: christoph.stampfel: Wichtig: Nur gebundene Variablen können mit der Alpha-Konversion umbenannt werden.

Wann ist ein λ-Audruck in Normalform?

[2] http://de.wikipedia.org/wiki/Turing-Vollständigkeit

Ein λ-Ausdruck ist in Normalform wenn weder β- noch η-Reduktion anwendbar ist.

Was besagt die Unentscheidbarkeit des Halteproblems?
Es ist nicht entscheidbar ob jemals eine Normalform erreicht wird oder nicht. Aus diesem Grund kann kein Turing-vollständiges System so eingeschränkt werden, dass nur entscheidbare Probleme ausdrückbar sind.

Was sind Zusicherungen? Wie können wir sie ausdrücken?
Zusicherungen beschreiben einen erwarteten Zustand eines Objektes oder eines Ausschnittes an der richtigen Stelle im Programm.

Zusicherungen können einerseits durch Kommentare oder in Java zusätzlich durch ***assert***-Anweisungen ausgedrückt werden.

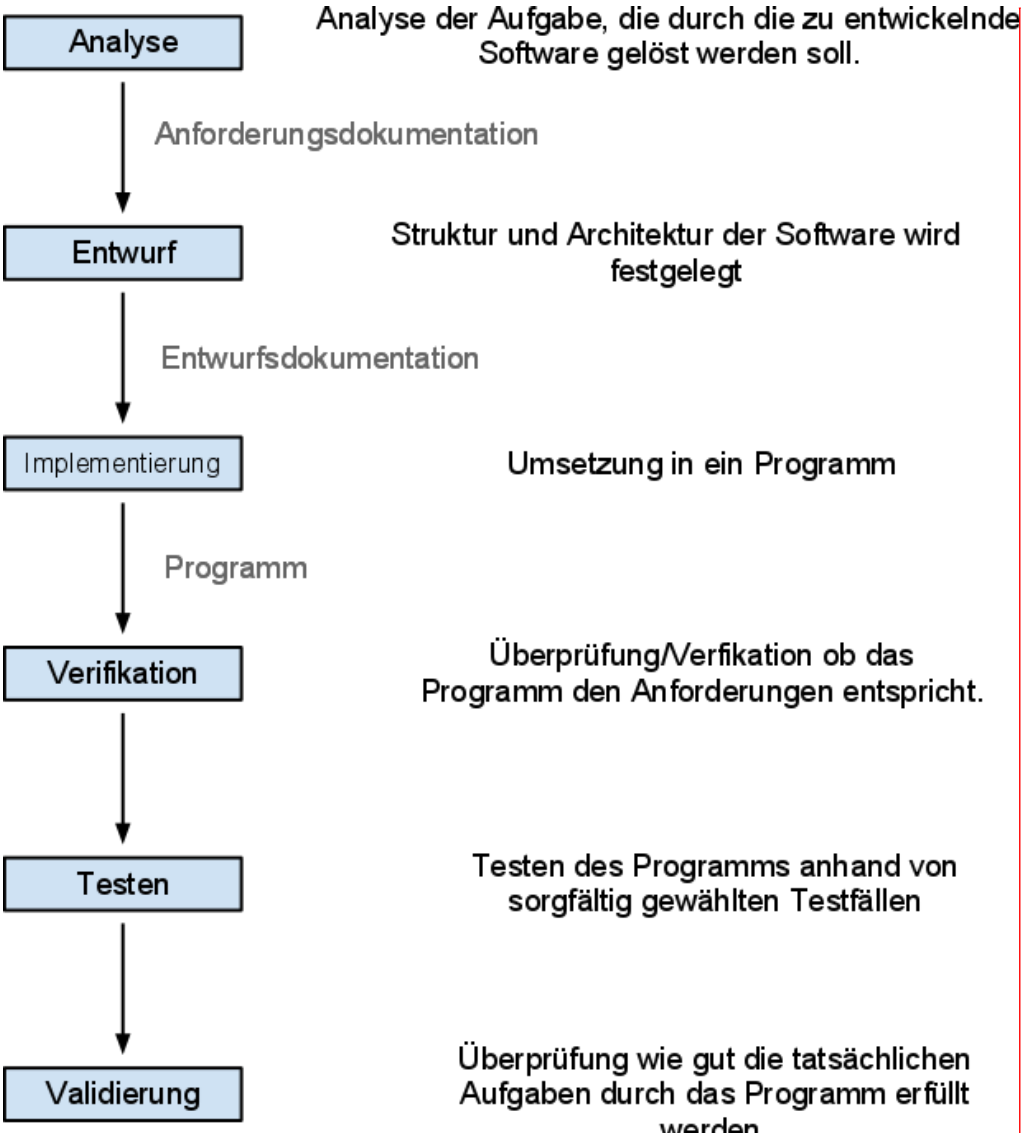
Wodurch unterscheiden sich Vor- von Nachbedingungen?
- Vorbedingungen
  - Vorbedingungen müssen vor der Ausführung einer Methode oder eines Konstruktor erfüllt sein. Sie werden von dem Aufrufenden erfüllt.
- Nachbedingungen
  - Bedingungen in Methoden und Konstruktoren, die nach der Ausführung erfüllt sein müssen. Sie müssen durch den Aufgerufenen erfüllt werden.

Was versteht man unter der Entwicklung, Wartung und Anwendung von Software?
- Entwicklung
  Unter Entwicklung versteht man die erstmalige Herstellung einer Software.

- Wartung
  Unter Wartung versteht man Fehlerbehebungen oder Anpassungen bestehender Software.

- Anwendung
  Benutzung der Software durch ihre Benutzer.

Welche Entwicklungsschritte werden unterschieden?



Vorgänge werden meist überlappend durchgeführt

Kommentar [8]: hektortrichter:
is euch bissi fad? :)

Philipp354:
:D vorher wars schon übersichtlich genug! aber brav brav ;)

paul106:
sehr gut! gibts eigentlich noch mehr Ausarbeitungen wie diese für die anderen LVA's?

Kommentar [9]: Anonymous:
im vowi normalerweise

Was versteht man unter den Begriffen zyklischer Softwareentwicklungsprozess und schrittweise Verfeinerung?

- Zyklischer Softwareentwicklungsprozess
  - Die einzelnen Schritte des Entwicklungsprozesses werden zyklisch wiederholt.

- Schrittweise Verfeinerung
  - Das Programm liegt am Beginn nur in groben Zügen vor und wird stetig verfeinert.

Welche Schritte führt man beim Programmieren zyklisch wiederholt aus?

- ~~Planen (Analyse, Design + Entwurf)~~
  ~~Zurechtlegen eines Plans für den aktuellen Durchlauf~~
- ~~Implementierung~~
  ~~Schreiben und Ändern des Programmcodes~~
- ~~Verifikation~~
  ~~Überprüfung, ob die implementierte Software den Anforderungen entspricht.~~
- ~~Testen~~
  ~~Der Programmcode wird unter Zuhilfenahme von Testfällen getestet.~~
- ~~Validierung~~
  ~~Überprüfung der Software, z.B. hinsichtlich Qualität, Praxisrelevanz~~

- Planen
  Zurechtlegen eines Plans was für den aktuellen Durchlauf erreicht werden soll
- Editieren
  Schreiben bzw. Ändern von Programmcode mittels Editor
- Übersetzen
  Wenn erforderlich wird der Compiler und weiter Werkzeuge zur Erzeugung von ausführbaren Code verwendet.
- Testen
  Durchlaufen von diversen Testfällen um festzustellen ob bzw. inwieweit die Ziele erreicht wurden.
- Debuggen
  Suchen nach noch vorhandenen Fehlern bzw unerwünschten Programmverhalten.
  (Dabei gewonnenes Wissen fließt wieder direkt in die Planung)

siehe Skriptum S. 64

Welche Softwareentwicklungswerkzeuge kennen Sie?

- Texteditor
- Compiler und Interpreter
- Debugger
- Integrierte Entwicklungsumgebung (z.B. Eclipse, Netbeans, Microsoft Visual Studio, ...)

Was ist eine Klassen-Bibliothek?

Eine Klassen-Bibliothek ist eine Sammlung vorgefertigter Programmteile. In Java bedeutet dies meist eine Sammlung von Klassen, die in einem Paket (.jar-File) zusammengefasst sind.

Wovon hängt die Qualität von Software ab?

- Brauchbarkeit
  - Zweckerfüllung
    Aufgaben werden zufriedenstellend für alle Anwendungsfälle gelöst. Unnötige Eigenschaften liefern keinen Beitrag zur Zweckerfüllung (z.B. größerer Ressourcenbedarf, schlechtere Bedienbarkeit)

  - Bedienbarkeit
    Wie einfach sind die Aufgaben lösbar, Einlernaufwand,
    Häufig zu lösende Aufgaben sollen wenige Arbeitschritte benötigen.

  - Effizienz
    Schonender Umgang mit den vorhanden Ressourcen (Prozessorzeit, Speicherbedarf)

- Zuverlässigkeit
  Eine Software soll korrekte Ergebnisse liefern, Programmabstürze vermieden werden und möglichst schnelle Reaktionen auf Ereignisse vorweisen können. Man soll sich auf die Software verlassen können.

  - Bewährtheit
    Software die sich über einen langen Zeitraum bewährt hat ist zuverlässiger als neue, wenig getestete Software. Erhöhung der Qualität durch Testen unter realistischen Bedingungen.

  - Formale Korrektheit
    Durchführung von formalen Korrektheitsbeweisen.
    Durch diese Methode können jedoch nicht alle Fehler eliminiert werden.

  - Fehlerresistenz
    Nicht jeder Fehler ist vermeidbar, jedoch sollten die Auswirkungen möglichst gemildert werden.

- Wartbarkeit
  Die Wartung von Software ist oft viel teurer als die Entwicklung. Fehlerkorrekturen oder Änderungen können in gut wartbarer Software rascher und zuverlässiger durchgeführt werden.

  - Einfachheit
    Einfachere Programme sind leichter und zuverlässiger änderbar. Programme sollen so einfach wie möglich gehalten werden. Berücksichtigung aller Eventualitäten bereits in der ersten Version.

- Lesbarkeit
  Lesbarkeit des Programmcodes - Programmierstil

- Lokalität
  Effekte einer Programmänderung sollen auf kleine Programmteile beschränkt bleiben.

- Faktorisierung
  Zerlegung des Programms in kleinere Einheiten. Zusammenfassung von gleichen Programmteilen. z.B. Abstraktion über Objekte oder Funktionen.

Wie kann man erwartete Eigenschaften von Software festlegen?

- Informelle Beschreibung
  - Beschreibungen in Form eines informellen Textes, erfordert keine Spezialkenntnisse, muss aber präzise sein (ist die Form, in der Anforderungen in der Arbeitswelt oft gestellt werden, da die Ausarbeitung oft auch durch Nicht-Informatiker erfolgt)

- Anwendungsfälle
  - Eine Reihe konkreter Anwendungsfälle (use cases)
  - Für jeden Anwendungsfall beschreibt man genau, welche Eingaben der Anwender machen kann und welche Ergebnisse er zu erwarten hat

- Testfälle
  - Programm wird direkt über Testfälle spezifiziert (formal)

- Formale Spezifikation
  - Sehr formal und braucht spezielles Expertenwissen, erlaubt aber dadurch eine formale Verifikation

Was unterscheidet funktionale von nichtfunktionalen Eigenschaften?

Funktionale Eigenschaften:
Funktionale Eigenschaften können in der Regel in Form einer Spezifikation ausgedrückt werden.
z.B. Welche Ergebnisse werden von Berechnungen für bestimmte Daten erwartet?

Nichtfunktionale Eigenschaften:
Nichtfunktionale Eigenschaften sind schwer oder gar nicht formal zu fassen.
z.B. Zuverlässigkeit, Wartbarkeit oder Bedienbarkeit ohne konkrete Vorgaben wie diese Eigenschaften erreicht werden sollen.

Kapitel 2 - Grundlegende Sprachkonzepte

Was ist ein Algorithmus?
Eine aus endlich vielen Schritten bestehende eindeutige Handlungvorschrift, die zur Lösung eines Problems führt.

Wie unterscheidet sich ein Kochrezept von einem Algorithmus?
Bei einem Kochrezept werden viele Schritte nicht explizit erwähnt. Bei einem Algorithmus muss man das aber unbedingt machen. Der Grund liegt darin, dass eine Maschine für die der Algorithmus geschrieben wird nicht eigenständig denken kann.

Was sind Ähnlichkeiten und Unterschiede von Objekten, Werten und Daten?

- Objekte
  - Werden durch entsprechende Daten im Rechner abgebildet; setzen sich aus Werten zusammen. Werden aber bei einer eher abstrakteren Sichtweise verwendet.
- Werte
  - z.B. konkrete Werte wie Zahlen oder Variableninhalte
- Daten
  - Größere Ansammlung von Werten

Welche Möglichkeiten zur Steuerung des Programmflusses gibt es?

- Fallunterscheidungen → Selektion (Wenn... dann... sonst...)
- Sequenz (Hintereinanderreihung von Handlungen)
- Iteration

Was ist eine Sequenz, Fallunterscheidung oder Iteration?
Möglichkeiten den Programmfluss zu steuern:
Sequenz = Hintereinanderreihung von Handlungen
Fallunterscheidung = Selektion zwischen zwei oder mehreren möglichen Fällen
Iteration = Anweisung zur Wiederholung

Was ist ein Ausdruck?
Ein Ausdruck in einer Programmiersprache ist ein Konstrukt, das einen Wert liefert.

Was ist eine Variable?
Ein veränderbarer Datenbehälter / Speicherbereich.

Was ist eine Zuweisung?
Eine Zuordnung eines Wertes zu einer Variable.

Was ist eine Speicheradresse?
Die eindeutige Adresse (meist hexadezimal angegeben) eines Speicherorts i
nnerhalb eines Speichers (z.B. RAM), in dem ein Wert (z.B. eine Variable) gespeichert ist. Der Wert innerhalb der Speicheradresse ist dabei variabel (d.h. er kann nachträglich geändert werden)

Was ist der Unterschied zwischen Lebensdauer, Gültigkeitsbereich und Sichtbarkeitsbereich?

- Lebensdauer
  - Zeitraum zwischen Reservieren von Speicher (Memory Allocation) und dessen Freigabe (Memory Deallocation)
  - z.B. lokale Variable:
    Lebensdauer ist auf die Methode, in der die Variable deklariert ist, begrenzt
- Gültigkeitsbereich
  - Der Gültigkeitsbereich bestimmt, von wo aus im Programm auf eine Variable zugegriffen werden kann.
- Sichtbarkeitsbereich
  - Eine Variable kann in ihrer Sichtbarkeit eingeschränkt werden (z.B. private). Die Variable kann in solchen Fällen zwar gültig aber durch die Einschränkung nicht sichtbar sein.

Was ist eine Initialisierung?
Bei der Initialisierung wird einer Variable oder einem Objekt ein Initialwert (=erster Wert) zugewiesen. Bei einem Objekt spricht man eher von einem Initialzustand.

Erfolgt die Initialisierung automatisch?
Das hängt sehr stark von der jeweiligen Programmiersprache ab. Im Falle von Java werden alle Variablen, **außer** lokale Variablen, mit einem Default-Wert versehen, sofern sie nicht manuell initialisiert werden. Im Falle von Arrays werden die Elemente des Arrays immer mit einem Default-Wert initialisiert.

Defaultwerte:

Datentyp	Default-Wert
Ganzzahlige Typen (z.B. byte, short, int, long)	0
float	0.0f
double	0.0
Referenztypen	null

Was sind Datentypen?
Ein Datentyp legt die Wertemenge bzw. die Operationen fest, die auf diesen Mengen durchgeführt werden können.

Welche elementaren Datentypen gibt es?

- byte
- char
- short
- int

Kommentar [10]: rpuenguentzky:
hmm.. Arrays ja, aber Variablen mMn. nicht. Wenn ich i deklariere und anschließend ausgebe, bekomme ich einen Fehler, i sei noch nicht initialisiert worden. Andere Meinungen?

hoellp:
Seh ich genau so. Variablen werden nicht automatisch initialisiert. Das passiert wie immer erst mit der Erstzuweisung. Wenn noch jemand zustimmt, schlage ich eine Änderung vor.

captain_flag:
arrays sind ja geschachtelte variablen. also sehe ich es so, dass "arrayvariablen" automatisch initialisiert werden.

Anonymous:
Grundsätzlich steht korrekt da "außer lokale Variablen". Membervariablen werden ja schon mit Defaultwerten initialisiert. Ist aber eher unglücklich formuliert mM nach.

captain_flag:
das stimmt. z.B. objektvariablen müssen auch initialisiert werden.

christoph.stampfel:
Es stimmt schon so, es werden alle Variablen von Java automatisch initialisiert außer eben lokale Variablen.

hoellp:
> Wenn ich i deklariere und anschließend ausgebe, bekomme ich einen Fehler

Wie erklärst du dann das? Das passiert mir immer wieder unter verschiedenen Umständen.

<-

Am 9. November 2011 20:22 schrieb christoph.stampfel (Google Docs) <

christoph.stampfel:
Wenn i bei dir eine lokale Variable ist, dann musst du sie vor der ersten Verwendung initialisieren. Ist i keine lokale Variable, dann wird sie automatisch Initialisiert (z.B. Member-Variablen in einer Klasse)

hoellp:
Gut zu wissen, das Skript unterscheidet dabei nicht. Dort steht auf Seite 86, dass jede Variable initialisiert werden muss, es kann aber zur gleichen Zeit vorgenommen werden.

- long
- float
- double
- boolean

Was sind Referenztypen?
- Arrays, Objekte, String

Referenztypen heißen so, weil jede Instanz davon – das ist ein Objekt – in einem separaten Speicherbereich abgelegt wird und eine Variable nur eine Referenz auf den Speicherbereich des Objekts enthält.

Was ist ein Operator, was ein Operand?
Operator: z.B. + - * (mathematische); && || (logische); < > == (relationale)
Operand: dasjenige Konstrukt, auf das der Operator angewendet wird (z.B. -**3, true**)
Ausdrücke können durch Anwendung eines Operators zu einem komplexeren Ausdruck verknüpft werden. Dabei übernehmen die zu verknüpfenden Ausdrücke die Rolle von Operanden für diesen Operator. Jeder Ausdruck kann so als Operand wieder zu einem Bestandteil eines komplexeren Ausdrucks werden.
In Java unterscheidet man einstellige (*unäre*) von zweistelligen (*binären*) Operatoren. Zusätzlich gibt es einen dreistelligen (*ternären*) Operator, den Bedingungoperator(?:).

Was heißt infix, präfix und postfix?
Jede endliche Teilfolge von aufeinander folgenden Symbolen eines Wortes *w* wird **Infix** oder **Teilwort** des Wortes *w* genannt (z.B. a **+** b).
Ein **Präfix** eines Wortes ist ein Infix am Anfang dieses Wortes (z.B. **++**x).
Ein **Postfix** eines Wortes ist ein Infix am Ende dieses Wortes (z.B. x**++**).

Was ist ein L-Wert, was ein R-Wert?
~~L-Wert: Das was links vom =(Wertzuweisung) steht.~~
~~R-Wert: Das was rechts davon steht.~~
~~Bsp: x = 1~~
~~x => L-Wert~~
~~1 => R-Wert~~
L-Wert: Beschreibt die Speicheradresse (Links vom Zuweisungsoperator)
R-Wert: Steht ein Variablenname rechts vom Zuweisungsoperator, so entspricht der Name dem Wert bzw. R-Wert der Variablen. R-Werte kommen nicht nur rechts von Zuweisungsoperatoren vor, sondern jeder Ausdruck, der kein L-Wert ist, ist ein R-Wert.

Was bedeutet final?
Final deklariert einen Eintrag (Variable, Klasse, Methode) als unveränderbar (konstant).

Was ist eine Ausdrucksanweisung?
Ausdrucksanweisungen sind Konstrukte die eine Mischform von Ausdrücken und Anweisungen darstellen.
D.h. eine Ausdrucksanweisung liefert beispielsweise einen Wert und hat zusätzlich einen Nebeneffekt (schlechter Stil!).
Bsp.: int y = x++; //x++ Ausdruck, int y Anweisung

In welcher Reihenfolge werden Ausdrücke ausgewertet?
In Java werden Ausdrücke entweder von links nach rechts (linksassoziativ) bzw. von rechts nach links (rechtsassoziativ) ausgewertet. Außerdem binden manche Operatoren stärker als andere. Diese werden vor den anderen Operatoren berücksichtigt. (Beispielsweise Punkt vor Strich)

Gleichwertige zweistellige arithmetische Operatoren werden linksassoziativ ausgewertet.

Was ist eine Operatorpriorität?
Welcher Operator zuerst behandelt wird → * vor + etc.

Was ist die Assoziativität?
Assoziativität bestimmt in welcher Reihenfolge Ausdrücke ausgewertet werden. Die Assoziativität ist abhängig vom verwendeten Operator.

Welche einstelligen und zweistelligen Operatoren gibt es in Java?
Einstellig: ++,--, positiv +, negativ -, not !, usw.
Zweistellig: +, -, *, /, %, <<, >>, &&, ||, &, |, usw.
Vollständige Liste: http://download.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Gibt es in Java dreistellige Operatoren?
Ja, es gibt ternäre Operatoren. ~~Ein~~ Das Beispiel: der ?: Operator.

Wie wird bei einer Division gerundet?

Round towards zero.
Bei der Division von ganzzahligen Werten, werden die Nachkommastellen abgeschnitten (truncate).

Was bedeutet Infinity und NaN?
Infinity = Unendlicher Wert
NaN = Not a Number = Keine Zahl

Was ist der Restwertoperator?
% “Modulo”, gibt den Restwert einer Division zurück (5%2 = 1)

Was ist ein Verkettungsoperator?
+ → verkettet Strings.
“Hallo“ + “ “ + “Welt” = “Hallo Welt”

Wie verhalten sich Zuweisungsoperatoren?
Rechtsassoziativ

Welche relationale Operatoren gibt es?
<,>,<=, >=, ==, !=

Wie unterscheiden sich logische Operatoren von Bitoperatoren?
Logische Operatoren verknüpfen mehrere Boolean-Werte zu einem Boolean-Wert.
Bitoperatoren führen verschiedene Operationen bitweise für einen Wert aus (z.B. Bitshifting).

Kommentar [11]: christoph.stampfel:
Steht das überhaupt im Skriptum?

Hängt das nicht von der realen Maschine ab auf der das Programm läuft.

Meines Wissens nach werden nur mit strictfp deklarierte Klassen/Methoden streng nach IEEE behandelt

zachariass:
Zitat aus dem Skriptum: Bei der Division von ganzen Zahlen (das heißt,

beide Operanden haben ganzzahlige Typen) wird zum Wert 0 hin ab- oder

aufgerundet, sprich die Nachkommastellen werden verworfen, und das Ergebnis

ist wieder eine ganze Zahl.

christoph.stampfel:
Ok, es kann sein, dass auch das gemeint ist. Frage ist etwas unglücklich formuliert.

geo.haidelber:
sollte man die Rundung nicht als truncate bezeichnen?! gerundet wird ja defakto gar nicht..

zachariass:
Ja...passt.

Kommentar [12]: benniklaus:
5/0 wäre infinity und nicht NaN

benniklaus:
NaN wäre z.B. 0/0

geo.haidelber:
beide beispiele sind falsch.. Division durch 0 liefert bekanntlich java.lang.ArithmeticException: / by zero also einen fehler

Anonymous:
es ist ja nur die bedeutung gemeint und nicht ob es in java möglich ist

Was macht der Bedingungsoperator?
Der Bedingungsoperator ist nichts anderes als eine andere Schreibweise für ein if-else Konstrukt.

```
boolean expression ? if-block : else-block

entspricht:
if(expression){
        //if-block
}else{
        //else-block
}
```

Was ist eine Typumwandlung?
Eine Umwandlung eines Wertes von einem Datentyp in einen anderen, wobei eventuell Informationen durch casting verloren gehen.

Was passiert bei einer Typumwandlung?
Ein Datentyp wird in einen anderen umgewandelt. Hierbei ist jedoch Vorsicht geboten, da Informationen beim "casten" (Umwandeln) verloren gehen können. zB beim Cast von double zu int gehen die Kommastellen verloren (truncate). Somit verfälscht sich unser Ergebnis. Casting wird bei einer einschränkenden Typumwandlung nicht vom Compiler durchgeführt, sondern muss explizit im Programmcode gefordert werden, indem vor dem ursprünglichen Wert der neue Datentyp in Klammer definiert wird:
double d = 2.5;
int i = (int)d;
System.out.println("i: " + i); → i: 2

Wie unterscheidet sich die einschränkende von der erweiternden Typumwandlung?
Einschränkende : Größerer Datentyp wird zu kleinerem umgewandelt
Erweiternder : Kleinerer Datentyp wird zu Größerem umgewandelt.

Unter Größer und Kleiner versteht man in dieser Hinsicht die Bitbreite der Typen bzw. dessen Genauigkeit. z.B. *float* kann größere Zahlen als *long* darstellen und wird somit bei der Umwandlung von *long* → *float* als *erweiternde* Typumwandlung bezeichnet.

Kommentar [13]: Anonymous:
Ich würde eher behaupten, dass eine einschränkende Typumwandlung von Typen mit größerem Wertebereich zu Typen mit kleinerem Wertebereich erfolgt und umgekehrt. Ich würde daher eher nicht die Genauigkeit oder Bitbreite als Argument anführen.

Anonymous:
von double auf float würds stimmen aber bei long auf int geb ich dir recht

Wann erfolgt eine implizite Typumwandlung, wann benötige ich eine explizite Typumwandlung?
Implizite Typumwandlungen erfolgen bei Erweiterenden Typumwandlungen.
Explizite Typumwandlungen werden bei Einschränkenden Typumwandlungen benötigt.

Was sind konstante Ausdrücke?
Ein Ausdruck ist konstant, wenn bereits vom Compiler der Rückgabewert des Ausdrucks berechnet wird. Das ist nur möglich, wenn der Ausdruck keine Variablen enthält.

z.B. wird der Ausdruck 5 + 2 vom Compiler in 7 umgesetzt, somit wird das Programm optimiert.

Was ist ein Literal?

Mit *Literal* bezeichnet man in Programmiersprachen eine Zeichenfolge, die zur direkten Darstellung der Werte von Basistypen (z. B. Ganzzahlen, Gleitkommazahlen, Zeichenketten) definiert bzw. zulässig sind.

Welche Fehlermöglichkeiten gibt es bei der Definition von Literalen?

```
short n = 5 * 3;          //wird übersetzt
short n = 5000 * 30;      //Compilerfehler (da der Wert zu groß ist für short)

long n = 2147483647 * 2L;     // n == 4294967294L
long n = 2147483647 * 2;      // n == -2L !Fehler! - Die Berechnung wird in int durchgeführt und
                              //dort kommt es zu einem Überlauf
```

Was ist ein Block?

Ein Block umfasst eine oder mehrere Anweisungen. Die Begrenzungssymbole für einen Block sind die geschwungenen Klammern { und }. (eine if-Anweisung ist z.B. ein Block)

Was ist ein innerer Block?

Ein innerer Block ist ein Block in einem Block. Ein innerer Block wäre zB eine weitere if-Abfrage in einer if-Abfrage, quasi eine Art Verschachtelung.

Darf in einem inneren Block eine Variable mit dem selben Namen wie in einem äußeren Block verwendet werden?

Auf die Variable darf zugegriffen werden, jedoch darf keine gleichnamige Variable deklariert werden.

Welche Anweisungen können zur Selektion verwendet werden?

<u>if - else</u>

```
if ( boolescher Ausdruck )
        Anweisung1
else
        Anweisung2
```

<u>switch</u>

```
switch ( Ausdruck ) {
        case konstanterAusdruck1:
                Anweisungen1
                break;
        case konstanterAusdruck2:
                Anweisungen2
                break;
        ..
        case konstanterAusdruck:
                Anweisungen
                break;
        default:
                defaultAnweisungen
}
```

Bedingungsoperator
variable = (boolescher Ausdruck) ? (Anweisung1) : (Anweisung2);

Erklären Sie die switch-Anweisung!
Eine Switch Anweisung definiert eine Fallunterscheidung, die je nach Wert (meistens jener einer Variable) unterschiedliche Anweisungen durchführt. Dabei können entweder genaue Werte oder auch "default" als Option gewählt werden. Wichtig ist in diesem Zusammenhang auch "break", da es sonst zu einem "fall-through" kommt, also auch jene Codeteile ausgeführt werden, die unter dem zutreffenden Fall stehen, obwohl die Bedingung garnicht zutrifft. Durch den Befehl break; wird die Fallunterscheidung abgebrochen.

Was ist eine Funktion?
Eine Ansammlung von Anweisungen, die durch verschiedene Eingangsparameter verschiedene Ergebnisse erzeugt.

laut Skript:
In Java werden Funktionen als spezielle Methoden (ohne Seiteneffekte) definiert.

Was sind Unterprogramme, was sind Routinen?
im Skript steht das:
In praktisch allen Programmiersprachen gibt es Sprachmittel, die Anweisungssequenzen, die mehrfach an verschiedenen Stellen im Programm, oder auch in verschiedenen Programmen gebraucht werden, wiederverwendbar machen. Durch diese soll nicht nur Codewiederholung vermieden werden, sondern auch die Möglichkeit zur Strukturierung bzw. Modularisierung des Softwareentwurfs geschaffen werden. Man nennt solche Programmteile allgemein Unterprogramme oder Routinen. Ein Unterprogramm soll eine in sich abgeschlossene und gut beschreibbare Teilaufgabe erledigen (in Java: z.B: Methode).

Was ist eine Methode?
Eine Methode ist ein Anweisungsblock, der unter einem Namen aufgerufen werden kann.Methoden sind immer Bestandteil einer Klasse.

Wie unterscheiden sich Funktionen, Methoden, Unterprogramme, Routinen und Prozeduren?
Wer hat das gelöscht???? → Ich zwar nicht, aber es ist weiter oben schon beschrieben.

Was ist der Unterschied zwischen formalen und aktuellen Parametern?
Formale Parameter sind spezielle lokale Variablen einer Methode (stehen im Methodenkopf)
Aktuelle Parameter werden implizit den formalen Parametern zugewiesen (sind jene, die im Methodenaufruf stehen)

Welche Aufgabe hat die return-Anweisung?
Die return Anweisung definiert den Rückgabewert einer Methode. Der Return Wert kann dabei entweder leer sein, oder aber dem im Methodenkopf definiertem Datentyp (zB public int Zufallszahl() → Rückgabewert int , etc.) entsprechen. Wird die Methode mit Rückgabewert "void", z.B. die Main Methode, definiert, darf KEIN return mit einem Rückgabewert stattfinden, ein einfaches "return" ist möglich.

Kommentar [14]: Huggy1992:
Afaik benötigt eine Funktion nicht zwingend Parameter, sondern kann auch auf globale Variablen zurückgreifen. Funktionen müssen nur einen entsprechenden Ausgabewert besitzen.

christoph.stampfel:
Sind das dann nicht Methoden oder Prozeduren? Wenn hier mit Funktion, math. Funktionen gemeint sind, sollten sie einen Rückgabewert besitzen da sie sonst keinen Sinn ergeben.

Huggy1992:
Ob Prozedur oder nicht hängt meines Wissens von der Ausgabe ab (retun oder nicht), aber nicht von der Eingabe. Auf die Schnelle hab ich auch im Scriptum nur die "reinen Funktionen" gefunden, die sich ausschließlich auf lokale (ich nehme an, damit sind die Argumente gemeint) Werte berufen. Laut Scriptum : "Die Methoden eines Objekts erlauben anderen Objekten, mit diesem Objekt in Kontakt zu treten.Man tritt mit einem Objekt in Kontakt, indem man ihm eine Nachricht schickt, und das Objekt beantwortet die Nachricht". Für mich klingt das so, als würde eine METHODE laut Definition Input (argumente) und return benötigen, während die Funktion (im ggsz. zur Prozedur, die kein output hat) nur eine Ausgabe benötigt.

Was bedeutet Überladen?
Durch Überladung ist es möglich, mehrere Methoden unter dem selben Methodennamen zu definieren. Anhand der Anzahl und der Typen der übergebenen Argumente (--> der geforderten Parameter) wird zur Laufzeit entschieden, welche Methode gemeint ist.

Auch ist in Java der Operator + ein “überladener Operator”, da er sowohl als Additions-Operator, als auch als Verkettungs-Operator fungiert.

Was geschieht bei einem Methodenaufruf?
Siehe Skript Seite 124/2.4.5 (Beispiel mit dem ggT Programm)

Was bedeutet Rekursion?
Die Rekursion beschreibt den Aufruf einer Funktion durch sich selbst. Dabei wird die aufrufende Methode erst fortgesetzt, wenn die “innere” aufgerufene Methode selbst terminiert ist. (Um Rekursion zu verstehen, muss man verstehen was Rekursion ist...)

Wie unterscheiden sich Rekursion und Iteration?
Iteration ist Wiederholung durch Aneinanderreihung (Kontrollstruktur: Schleife)
Rekursion ist Wiederholung durch Ineinanderschachtelung (Kontrollstruktur: Verzweigung)

Was sind Zusicherungen?
Zusicherungen beschreiben relevante Ausschnitte aus dem erwarteten Zustand eines Objekts des Systems an der richtigen Stelle im Programm auf systematische Weise. (Skriptum S. 58f)

Welche Arten von Zusicherungen gibt es?

Kommentare bilden die einfachste Form der Zusicherungen, zusätzlich gibt es “assert” Anweisungen, die vom Intepreter zur Laufzeit überprüft werden und nötigenfalls Laufzeitfehler generieren. Weiters unterscheidet man die Zusicherungen:

- Vorbedingung - eine Bedingung die bereits vor dem Aufruf einer Methode, Funktion erfüllt sein muss.
- Nachbedingung - eine Bedingung die während der Ausführung der Methode erfüllt werden muss. (Skriptum S 59f).

Warum verwendet man Zusicherungen?
Zusicherungen dienen dem schnellen und einfachen Verständnis bestimmter Programmabschnitte, die besonders im Bereich der objektorientieren Programmierung leicht unübersichtlich werden können.
Sie sind auch besonders dann relevant, wenn man zB auf Methoden zugreift, deren Implementierung man nicht kennt und daher darauf angewiesen ist, sich auf bestimmtes Verhalten derer (v.a. in Extremfällen, z.B. Werte unter/über 0, Werte nahe dem Überlauf etc) verlassen zu können.

Mit welchen Anweisungen kann eine Iteration implementiert werden?
Schleifen (for, while, do-while, for-each)

Was ist ein Array?

Ein Array ist ein Objekt, das aus mehreren Komponenten zusammengesetzt ist. Die Komponenten heißen Elemente und sind alle vom selben Datentyp (Elementtyp => Typ der Arrayvariable).
(Def. lt. Skript S 134)

Was sind Indextypen, was sind Elementtypen?
Elementtypen sind die Datentypen der Elemente im Array/in der Liste/der HashMap...
Indextypen sind die Typen der Indizes der Elementtypen

Werden Arrays automatisch initialisiert?
Arrays werden beim Erstellen mittels new-Operator immer automatisch initialisiert.

Was ist eine Initialisierungsliste?
Initialisierungslisten helfen bei der Erstellung von Arrays, sie ermöglichen das Weglassen der Arraygröße:
int[] data = { 3, 13, 23, 33 };

Was ist ein mehrdimensionales Array?
Ein Array, dessen Elemente wieder den Datentyp eines Arrays haben.

Wie kann eine for-Schleife spezifiziert werden?
Eine for-Schleife wird als Iteration spezifiziert
Die for Schleife ist eine spezielle Iteration, falls eine Laufvariable benötigt wird. Sie stellt eine syntaktische Vereinfachung der while Schleife für diesen Fall dar.

Was ist der Programmzustand?
Der Programmzustand ist ein ”Schnappschuss” aller definierten Variablen. Er umfasst also die Zustände aller vom Programm verwendeten Variablen zu einem bestimmten Zeitpunkt der Ausführung.

Was ist ein Seiteneffekt?
Unter Seiteneffekt versteht man die Änderung des Programmzustands. Die Auswertung von Ausdrücken kann üblicherweise keine Seiteneffekte bewirken, da nur ein Wert berechnet wird. Eine Zuweisung dagegen hat einen Seiteneffekt, da sie den Programmstatus ändert.

Bsp.: x+1 ist ein Ausdruck. Die Auswertung dieses Ausdrucks liefert einen Wert zurück, bewirkt jedoch noch keine Änderung des Programmstatus. Dieser Wert kann jedoch für eine Zuweisung verwendet werden, was einen Seiteneffekt bewirkt zB.: x = x + 1.

Was ist der Unterschied zwischen Funktionen und Prozeduren in der Programmiersprache Pascal?
Funktionen übernehmen aktuelle Parameter und kopieren sie in formale Parameter(Wertparameter). Dann liefern sie einen Rückgabewert. Prozeduren machen selbiges, allerdings übernehmen sie Variablenparameter(diese besitzen eine Referenz auf eine Variable außerhalb der Prozedur und verändern diesen Wert außerhalb auch),der dann als Seiteneffekt der Prozedur verändert wird.

Was ist ein Wertparameter, was ist ein Referenzparameter?
In Java: Primitive Datentypen sind Wertparameter, Objekte/Arrays sind Referenzparameter.

Kommentar [15]: benniklaus:
"Indextyp" ist im ganzen Skript nur in den Fragen zu finden... und im internet findet man auch nicht wirklich was...

manuel.geier:
Da es hier um Arrays geht, nehme ich an mit Indextyp ist der Typ der Variable, um auf ein Element im Array zuzugreifen. Meist ein ganzzahliger Wert oder auch Zeichenketten.
data[4] oder data['text']

Elementtyp ist der Typ der Elemente des Arrays.
int[]

benniklaus:
data['text'] geht auch?! wusste ich nicht... wie wird das dann definiert mit welchem "label" die elemente bezeichnet werden?! Also mit ganzzahligen Werten ist es ja klar, da wird einfach durchnummeriert, aber wie kommt man dazu, dass ein element über data['text'] angesprochen werden kann?

Anonymous:
bei nem Array gehts glaub ich nicht, aber bei ner Hash table wird damit auf die elemente zugegriffen

michi.mittermayr:
meines wissens nach ist es in java nicht möglich etwas anderes als eine zahl als index zu verwenden, bzw einen index für etwas anderes als ein array zu verwenden (Index im sinne von [0], [1], ...)

anders würde es in .net (c#) aussehen, dort kann alles als index verwendet werden

oder ist hier auch ein methodenaufruf mit einer variable als index gemeint ala charAt(INDEX) ?

manuel.geier:
In Java ist es bei Arrays nicht möglich, in PHP hingegen schon. In Java ist als Indextyp daher nur int, short und long sinnvoll möglich.

Am 9. November 2011 11:21 schrieb michi.mittermayr (Google Docs) <

masturbator3001:
ich hab mal das hingeschrieben, was ich gelernt hab

Anonymous:
In Java gibt es auch Hasch Tables und bei Haschtabels darf der Index einen Beliebigen typ haben

christoph.stampfel:
Bei den Hashtables kann man auch nicht mittels [index] darauf zugreifen.

Kommentar [16]: hoellp:
Durch genauere Erklärung aus Skript ersetzt.

Was bedeutet call-by-value, was call-by-reference?
call-by-value: Kopieren der aktuellen Parameter in Wertparameter.
call-by-reference: Variablenparameter der eine Referenz auf eine Variable außerhalb der Prozedur/Methode ist.
In Java gibt es nur call-by-value (beachte: bei Referenztypen wird die Adresse kopiert! Call by value erlaubt damit zwar kein Ersetzen der Adresse, es kann aber auf das eigentliche Objekt zugegriffen werden (ist keine Kopie!)).

Wie unterscheidet sich der funktionale vom prozeduralen Programmierstil?
Funktionaler Stil ist die Verschachtelung von Ausdrücken. In diesem Stil hat die Auswertung von Ausdrücken und Funktionen keine Zustandsänderung zur Folge. Es gibt nur Ausdrücke ohne Seiteneffekte.
Prozeduraler Stil ist das hauptsächliche Nutzen von Operationen mit Seiteneffekten.

Was ist ein transformatives, was ein reaktives System?
Ein transformatives System wird mit einer bestimmten Eingabe gestartet und nach begrenzter Laufzeit mit einer Ausgabe terminiert. Das System arbeitet ansonsten entkoppelt von
der Außenwelt und es gibt keinen nach außen sichtbaren Zustand, d.h. die Ausgabe hängt ausschließlich von der Eingabe ab. Sie besitzen keine zeitliche Dimension.
Ein reaktives System läuft potentiell unendlich lange und besitzt einen durch Eingaben veränderbaren inneren Zustand, der das Verhalten des Systems bestimmt. Das System ist in eine Umgebung eingebunden und reagiert auf Eingaben aus unterschiedlichen Quellen (z.B. Sensordaten) mit entsprechenden Ausgaben. Hier spielt der Zustand, welcher sich im Laufe der Zeit ändert, eine wesentliche Rolle.

Was ist das besondere an der Methode main?
Die Methode main ist automatisch immer der Einstiegspunkt eines Java-Programmes. Wird eine Java-Klasse ausgeführt, ist die main Methode die erste, die vom Laufzeitsystem ausgeführt wird. Sie muss immer wie folgend aussehen:
public static void main(String [] args)
Die Methode ist außerdem keine Funktion, da sie keinen Rückgabewert hat.

Kommentar [17]: Anonymous:
In Java existiert doch nur call-by-value oder??

Huggy1992:
Sehe ich auch so - Java kennt lediglich CBV. Wers nicht glaubt, siehe Java-ist-auch-eine-Insel, Kapitel 3.6 (http://www.eshca.net/java/books/javainsel8/javainsel_03_006.htm#mj4375f35ed5e61c4d8120182955b44e1f)

Anonymous:
JA... bei Funktionsaufrufen wird immer nur eine Kopie übergeben... in C z.B. kann man einen Pointer übergeben

Huggy1992:
Wobei ich ganz ehrlich gerade etwas irritiert bin - wenn jemand PP Beispiel 3 schon erledigt hat, die Methode fill() wollte ja als Parameter ein Grafikarray, wenn man innerhalb der Funktion das lokale Array bearbeitet hat, wurde auch das original array bearbeitet. Steh' ich um die Uhrzeit auf dem Schlauch, oder ist das keine Art der Referenz? Oder übergibt Java die Werte (also nicht Referenz), und "kopiert" Änderungen auf das Original?

Anonymous:
ein char array ist ja eine Referenz auf den speicherbereich.. damit übergiebst du eine Kopie der Referenz. aber call bei referenz währe jedoch, das du eine Referenz auf die Referenz übergiebst... wenn du zum beispiel ein Objekt übergibts währe call by value einfach eine Kopie des Zeigers, call by referenz hingegen würde eine Refernz auf die Referenz übergeben...

Huggy1992:
Also eine Zeigerkopie - passt, danke. Beim Schreiben hab ich mir da nichtmal einen Kopf drüber gemacht ;)

Kommentar [18]: david.weidhofer:
"Sie muss immer wie folgend aussehen" - stimmt das ? ich denk es gibt sicher fälle wo man main anders benutzt...

ffwoerister:
Ich würd eher sagen: "Ihr Methodenkopf muss immer wie folgend aussehen:".

nikola.ilo:
http://www.java-blog-buch.de/040305-sonderfall-main/