

Inhalt

Vorwort	13
Einführung	15
E Danksagung	23
I Was Sie zunächst wissen sollten (Einführung in die objektorientierte Programmierung)	27
1.1 Wesentliche Bestandteile der Objektorientierung	27
1.1.1 Objekte	28
1.1.2 Klassen	29
1.1.3 Klassenattribute und -methoden	30
1.1.4 Vererbung	32
1.1.5 Polymorphismus	37
1.2 Andere objektorientierte Konzepte	43
1.2.1 Schnittstelle versus Implementierung	43
1.2.2 Aggregation	44
1.2.3 Generische Programmierung	46
1.2.4 Persistenz	48
1.3 Terminologie: ein paar (schon zu viele) Worte	49
1.4 Falls Sie mehr wissen wollen	50
1.4.1 Bücher	51
1.4.2 Websites	51
1.4.3 Newsgroups	52
1.5 Zusammenfassung	52
2 Was Sie außerdem noch wissen sollten (Auffrischung in Perl)	55
2.1 Perl-Grundlagen	55
2.1.1 Skalare	55
2.1.2 Arrays	56
2.1.3 Hashes	61
2.1.4 Subroutinen	64
2.1.5 Verweise und anonyme Hashes	80
2.1.6 Packages	83

2.2	Weniger grundlegende (aber sehr nützliche) Perl-Elemente	92
2.2.1	Module	92
2.2.2	Automatisches Laden	98
2.2.3	Closures	99
2.2.4	Typeglobs	102
2.3	Das CPAN	110
2.3.1	Wie man auf das CPAN zugreift	111
2.3.2	Wie man im CPAN sucht	112
2.3.3	Wie man Module vom CPAN installiert	112
2.4	Falls Sie mehr wissen wollen	115
2.4.1	Grundlegende Bücher	115
2.4.2	Weitere nützliche Bücher	115
2.4.3	Die PERL Dokumentation	116
2.4.4	Das Perl Journal	117
2.4.5	Websites	118
2.4.6	Newsgroups	118
2.5	Zusammenfassung	119
3	Es geht los	121
3.1	Drei einfache Regeln	121
3.1.1	Regel 1: Um eine Klasse zu erzeugen, lege ein Package an	121
3.1.2	Regel 2: Um eine Methode anzulegen, schreibe eine Subroutine	122
3.1.3	Regel 3: Um ein Objekt zu erzeugen, verwende bless ()	125
3.2	Eine einfache Perl-Klasse	130
3.2.1	Der Code	130
3.2.2	Die Klasse CD::Music benutzen	139
3.3	Erleichterungen	140
3.3.1	Klassenmodule	140
3.3.2	use strict und der -w Indikator	141
3.3.3	Den Zugriff auf Datenelemente automatisieren	143
3.3.4	Eine Klasse dokumentieren	148
3.4	Das Erzeugen und Zerstören von Objekten	150
3.4.1	Konstruktoren	150
3.4.2	Destruktoren	161
3.5	Die Klasse CD::Music, komplett	170
3.6	Zusammenfassung	173
4	Objekte aus Arrays und Skalaren	175
4.1	Was ist falsch mit einem Hash?	175
4.2	Ein Array-Objekt	176
4.2.1	Ein Array-spezifisches Beispiel – Iteratoren	181
4.2.2	Falls Sie mehr wissen wollen	184
4.3	Einen Pseudo-Hash-Objekt	184
4.3.1	Pseudo was???	184
4.3.2	Vorteile von Pseudo-Hashes	186
4.3.3	Das Schlechteste beider Welten?	188
4.3.4	Compilerunterstützung für Performanz zur Laufzeit	189
4.3.5	Typed lexicals	190

4.3.6	Noch eine Version von CD::Music	192
4.3.7	Wenn Sie mehr wissen wollen	194
4.4	Einen Skalar als Objekt erzeugen	195
4.4.1	Und warum auch nicht?	195
4.4.2	Ein objektorientiertes Passwort	196
4.4.3	Eine Bit-String-Klasse	198
4.4.4	Wenn Sie mehr wissen wollen	202
4.4.5	Zusammenfassung	203
5	Objekte aus anderen Dingen	205
5.1	Ein regulärer Ausdruck als Objekt	205
5.1.1	Der qr-Operator	205
5.1.2	Warum eine objektorientierte Klasse für reguläre Ausdrücke?	207
5.1.3	Entwurf eines alternativen Mechanismus für reguläre Ausdrücke	209
5.1.4	Ein genauerer Blick auf die beiden Klassen	212
5.1.5	Zur Trennung von Suche und Zustand	214
5.1.6	Falls Sie mehr wissen wollen	215
5.2	Eine Subroutine instanziieren	215
5.2.1	Wie kann also eine Subroutine ein Objekt sein?	215
5.2.2	Subroutine als Objekt – warum?	217
5.2.3	Ein Lexer-Objekt	218
5.2.4	Beispiel: Ein einfacher Pretty Printer	222
5.2.5	Wenn Sie mehr wissen wollen	223
5.3	Einen Typeglob als Objekt benutzen	224
5.3.1	Seitenweise Ausgabe mit STDOUT	224
5.3.2	Seitenweise Ausgabe als Mehrfachprozess	225
5.3.3	Die Klasse Pager mit Threads	231
5.3.4	Falls Sie mehr wissen wollen	233
5.4	Zusammenfassung	234
6	Vererbung	235
6.1	Wie Perl die Vererbung implementiert	235
6.1.1	Das @ISA-Array	235
6.1.2	Was Vererbung in Perl bedeutet	236
6.1.3	Wo ein Methodenaufruf hingeht	238
6.1.4	Konstruktoren und Vererbung	239
6.1.5	Verbotene Diamanten	243
6.1.6	Destruktoren und Vererbung	244
6.2	Tricks und Fallen	247
6.2.1	Die Methode isa()	247
6.2.2	Die Methode can()	248
6.2.3	Das Package UNIVERSAL	251
6.2.4	Das Pseudo-Package SUPER	253
6.2.5	Abstrakte Methoden implementieren	255
6.2.6	Attribute abgeleiteter Klassen benennen	257
6.2.7	Der Test der leeren Unterklasse	260
6.2.8	Vererbung und Pseudo-Hashes	262

6.3	Beispiel: Die Klasse CD::Music vererben	265
6.3.1	Angewandte Faulheit	266
6.3.2	Zugriff auf Klassendaten im Lichte der Vererbung	268
6.3.3	Eine alternative Lösungen	271
6.4	Wenn Sie mehr wissen wollen	275
6.5	Zusammenfassung	275
7	Polymorphismus	277
7.1	Polymorphismus in Perl	277
7.1.1	Schnittstellenpolymorphismus	277
7.1.2	Vererbungspolymorphismus	278
7.2	Beispiel: polymorphe Methoden für die Klasse Lexer	279
7.3	Der einfache Pretty Printer nach objektorientierter Art	282
7.4	Alternativer Schnittstellenpolymorphismus	284
7.5	Falls Sie mehr wissen möchten	287
7.6	Zusammenfassung	287
8	Die Klassenerzeugung automatisieren	289
8.1	Das Modul Class::Struct	289
8.1.1	Klassen erzeugen	289
8.1.2	Attributtypen	292
8.1.3	Hierarchische Klassenstrukturen	294
8.1.4	Objekte initialisieren	295
8.1.5	Vererbung und generierte Klassen	296
8.1.6	Ein vollständiges Beispiel – die automatisierte Klasse CD::Music	297
8.2	Das Modul Class::MethodMaker	298
8.2.1	Konstruktoren	299
8.2.2	Skalare Attribute	301
8.2.3	Gruppierte skalare Attribute	302
8.2.4	Indikator-Attribute	303
8.2.5	Schlüssel-Attribute	304
8.2.6	Nichtskalare Attribute	305
8.2.7	Klassenattribute	306
8.2.8	Geschachtelte Objekte als Attribute	306
8.2.9	Subroutinen als Attribute	309
8.2.10	Abstrakte Methoden	311
8.2.11	Vererbung und generierte Klassen	312
8.2.12	Ein vollständiges Beispiel: Die Klasse CD::Music erneut automatisiert	312
8.3	Wenn Sie mehr wissen möchten	314
8.4	Zusammenfassung	314
9	Bindungen	315
9.1	Mit Frack und Binder	315
9.1.1	Beschränkungen beim Binden	316
9.2	Einen Skalar binden	317
9.2.1	Einen Skalar ent-binden	318
9.2.2	Ein einfaches Beispiel	318

9.2.3	Einen Skalar mit einem Nicht-Skalar implementieren	321
9.3	Einen Hash binden	324
9.3.1	Beispiel: Groß-/Kleinschreibungsneutrale Hashes	325
9.4	Ein Array binden	330
9.4.1	Beispiel: Ein Basen/Codon Array	332
9.5	Eine Filehandle binden	339
9.5.1	Ein Beispiel: gefilterte Filehandles	341
9.6	Von einem bindefähigen Package erben	346
9.6.1	Beispiel: Sortierte Hashes	347
9.6.2	Ein weiteres Beispiel: Mikro-Verfolgung von Skalaren	349
9.7	Gebundene Variablen als Objekte	350
9.7.1	Eine DNA-Klasse	351
9.7.2	Objektorientierte gebundenen Filehandles	354
9.7.3	In das gleiche Package binden und verweisen	356
9.8	Wenn Sie mehr wissen wollen	362
9.9	Zusammenfassung	362
10	Das Überladen von Operatoren	363
10.1	Das Problem	363
10.2	Perls Mechanismus für das Überladen von Operatoren	365
10.2.1	»Automagische« Operatoren	368
10.2.2	Abfangoperationen	369
10.2.3	Konvertierungsoperationen spezifizieren	370
10.3	Beispiel: eine Klasse für römische Zahlen	373
10.3.1	Klassenkonstanten erzeugen	377
10.4	Unerwünschte Semantik von Verweisen umgehen	381
10.5	Gebrauch und Missbrauch von Operatoren	383
10.5.1	Wann überladen?	385
10.6	Wenn Sie mehr wissen wollen	386
10.7	Zusammenfassung	387
11	Kapselung	389
11.1	Die Gefahren des Vertrauens	389
11.2	Kapselung über Closures	390
11.2.1	Eine Variation für Paranoiker	394
11.3	Kapselung mit Skalaren	396
11.3.1	Name, Rank und Dienstnummer	397
11.3.2	Kontrollierter Zugriff	400
11.3.3	Objekte iterieren	401
11.3.4	Eine Frage der Identität	401
11.3.5	Eine Variation für wirkliche Paranoiker	402
11.4	Kapselung mit Hilfe von Bindungen	406
11.4.1	Ein Hash mit begrenztem Zugriff	407
11.4.2	Einen SecureHash erzeugen	408
11.4.3	SecureHash-Einträge deklarieren	408
11.4.4	Auf SecureHash-Einträge zugreifen	410

11.4.5	Einen SecureHash iterieren	412
11.4.6	Mehrdeutige Schlüssel in einem SecureHash	413
11.4.7	Fehlersuche in einem SecureHash	418
11.4.8	»Schnelle« SecureHashes	419
11.4.9	»Strikte« SecureHashes	422
11.4.10	Die formalen Zugriffsregeln	424
11.5	Wenn Sie mehr wissen wollen	426
11.6	Zusammenfassung	426
12	Generisches	427
12.1	Warum Perl keinen spezifischen generischen Mechanismus braucht	427
12.2	Trotzdem spezielle Mechanismen benutzen	429
12.2.1	Closures als generische Methoden	429
12.2.2	Generische Klassen mit eval generieren	435
12.3	Implizite generische Techniken über Polymorphismus	437
12.3.1	Die generische Klasse Tree	438
12.3.2	Eine spezifische Knotenklasse	441
12.3.3	Die Konstruktion verwandter Knotenklassen	445
12.3.4	Aufräumarbeiten: eine abstrakte Basisklasse	446
12.3.5	Eine nicht verwandte Knotenklasse	448
12.3.6	Gegenüberstellung	452
12.3.7	Eine philosophische Bemerkung	453
12.4	Wenn Sie mehr wissen möchten	454
12.5	Zusammenfassung	454
13	Mehrfachzuordnung	455
13.1	Was ist Mehrfachzuordnung?	455
13.2	Mehrfachzuordnung mittels Einfachzuordnung und Fallunterscheidung	457
13.3	Mehrfachzuordnung über eine Tabelle	461
13.3.1	Bestimmung der Initialisierungsreihenfolge für die Tabelle	465
13.3.2	Probleme mit der Reihenfolge	466
13.4	Vergleich der beiden Ansätze	468
13.5	Dynamische Zuordnungstabellen	469
13.5.1	Nichts ist umsonst ...	473
13.6	Einige verbleibende Schwierigkeiten	474
13.7	Das Modul Class::Multimethods	475
13.7.1	Die geeignetste Multimethode identifizieren	477
13.7.2	Die geeignetste Multimethode finden	478
13.7.3	Implikationen der Suche in die Breite für die Multimethodenzuordnung	481
13.7.4	Wenn die Zuordnung scheitert	486
13.7.5	Multimethoden außerhalb ihrer Klassen definieren	488
13.7.6	Multimethoden als reguläre Subroutine	490
13.7.7	Parametertypen, die keine Klassen sind	491
13.7.8	Ein Parameter der letzten Zuflucht	493
13.7.9	Rekursive Mehrfachzuordnung	494
13.7.10	Fehlersuche in einer Multimethode	495

13.8	Vergleich der drei Ansätze	496
13.9	Wenn Sie mehr wissen wollen	497
13.10	Zusammenfassung	497
14	Persistente Objekte	499
14.1	Die Zutaten	499
14.1.1	Identität	500
14.1.2	Encoding/Serialisierung	501
14.1.3	Speicherung	506
14.1.4	Koordination	512
14.2	Objektorientierte Persistenz	513
14.2.1	Objekte kodieren	513
14.2.2	Objektorientierte Kodierung	514
14.3	Grobkörnige Persistenz	516
14.3.1	Klassenspezifische Persistenz	516
14.3.2	Einige Verbesserungen	520
14.3.3	Grobkörnige Persistenz für jede Art Daten	525
14.3.4	Beurteilung der Technik	529
14.4	Feinkörnige Persistenz	530
14.4.1	Plattendateien als Objekte	530
14.4.2	Dateien mit Memory Mapping als Objekte	534
14.4.3	Gebundene Datenbanken als Objekte	537
14.4.4	Feinkörnige Persistenz für jede Klasse	541
14.4.5	Einfachere Persistenz mit generischer Programmierung	546
14.4.6	Beurteilung der Technik	548
14.5	Wenn Sie mehr wissen wollen	548
14.6	Zusammenfassung	549
A	Kurzreferenz	551
B	Für Leute mit Vorkenntnissen	559
B.1	Perl und Smalltalk	559
B.1.1	Objekte	560
B.1.2	Klassen	560
B.1.3	Methoden	561
B.1.4	Kapselung	562
B.1.5	Vererbung	562
B.1.6	Polymorphismus	563
B.1.7	Kontrollstrukturen und Ausnahmebedingungen	563
B.1.8	Syntaxvergleichstabelle	564
B.1.9	Wenn Sie mehr wissen wollen	565
B.2	Perl und C++	566
B.2.1	Objekte	566
B.2.2	Klassen	567
B.2.3	Methoden	567
B.2.4	Konstruktoren und Destruktoren	569
B.2.5	Kapselung	569

B.2.6	Vererbung	569
B.2.7	Polymorphismus	570
B.2.8	Syntaxvergleichstabelle	571
B.2.9	Wenn Sie mehr wissen wollen	572
B.3	Perl und Java	573
B.3.1	Objekte	573
B.3.2	Klassen	574
B.3.3	Methoden	574
B.3.4	Konstruktoren und Finalisierungsmethoden	575
B.3.5	Kapselung	575
B.3.6	Vererbung	576
B.3.7	Polymorphismus	576
B.3.8	Ausnahmeverarbeitung	577
B.3.9	Syntaxvergleichstabelle	577
B.3.10	Wenn Sie mehr wissen wollen	579
B.4	Perl und Eiffel	579
B.4.1	Objekte	580
B.4.2	Klassen	580
B.4.3	Methoden	581
B.4.4	Kapselung	582
B.4.5	Vererbung	582
B.4.6	Polymorphismus	583
B.4.7	Generische Konzepte	583
B.4.8	Syntaxvergleichstabelle	584
B.4.9	Wenn Sie mehr wissen wollen	585
Glossar		587
Bibliographie		601
Objektorientierung		601
Perl		601
Andere objektorientierte Sprachen		602
Index		603