

Bernhard Lahres, Gregor Rayman

Praxisbuch Objektorientierung

Von den Grundlagen zur Umsetzung

Inhalt

1	Einleitung	11
1.1	Was ist Objektorientierung?	11
1.2	Hallo liebe Zielgruppe	12
1.3	Was bietet dieses Buch (und was nicht)?	13
1.3.1	Bausteine unseres Buchs	14
1.3.2	Crosscutting Concerns: übergreifende Anliegen	16
1.3.3	Und was ist mit Übungsaufgaben?	18
1.3.4	Die Rolle von Programmiersprachen	19
1.4	Warum überhaupt Objektorientierung?	21
1.4.1	Gute Software: Was ist das eigentlich?	21
1.4.2	Die Rolle von Prinzipien	23
1.4.3	Viele mögliche Lösungen für ein Problem	23
2	Die Basis der Objektorientierung	25
2.1	Die strukturierte Programmierung als Vorläufer der Objektorientierung	26
2.2	Die Kapselung von Daten	29
2.3	Polymorphie	30
2.4	Die Vererbung	32
2.4.1	Vererbung der Spezifikation	32
2.4.2	Erben von Umsetzungen (Implementierungen)	33
3	Die Prinzipien des objektorientierten Entwurfs	37
3.1	Prinzip 1: Prinzip einer einzigen Verantwortung	38
3.2	Prinzip 2: Trennung der Anliegen	43
3.3	Prinzip 3: Wiederholungen vermeiden	45
3.4	Prinzip 4: Offen für Erweiterung, geschlossen für Änderung	48
3.5	Prinzip 5: Trennung der Schnittstelle von der Implementierung ...	51
3.6	Prinzip 6: Umkehr der Abhängigkeiten	54
3.6.1	Umkehrung des Kontrollflusses	58
3.7	Prinzip 7: Mach es testbar	60

4	Die Struktur objektorientierter Software	63
4.1	Die Basis von allem: das Objekt	63
4.1.1	Eigenschaften von Objekten: Objekte als Datenkapseln	64
4.1.2	Operationen und Methoden von Objekten	72
4.1.3	Kontrakte: Ein Objekt trägt Verantwortung	77
4.1.4	Die Identität von Objekten	79
4.1.5	Objekte haben Beziehungen	81
4.2	Klassen: Objekte haben Gemeinsamkeiten	82
4.2.1	Klassen sind Modellierungsmittel	82
4.2.2	Kontrakte: die Spezifikation einer Klasse	86
4.2.3	Klassen sind Datentypen	90
4.2.4	Klassen sind Module	100
4.2.5	Sichtbarkeit von Daten und Methoden	103
4.2.6	Klassenbezogene Methoden und Attribute	110
4.2.7	Singleton-Methoden: Methoden für einzelne Objekte ...	114
4.3	Beziehungen zwischen Objekten	115
4.3.1	Rollen und Richtung einer Assoziation	117
4.3.2	Navigierbarkeit	118
4.3.3	Kardinalität	119
4.3.4	Qualifikatoren	122
4.3.5	Beziehungsklassen, Attribute einer Beziehung	123
4.3.6	Implementierung von Beziehungen	125
4.3.7	Komposition und Aggregation	126
4.3.8	Attribute	129
4.3.9	Beziehungen zwischen Objekten in der Übersicht	130
4.4	Klassen von Werten und Klassen von Objekten	130
4.4.1	Werte in den objektorientierten Programmiersprachen	131
4.4.2	Entwurfsmuster Fliegengewicht	134
4.4.3	Aufzählungen (Enumerations)	137
4.4.4	Identität von Objekten	144
5	Vererbung und Polymorphie	153
5.1	Die Vererbung der Spezifikation	153
5.1.1	Hierarchien von Klassen und Unterklassen	153
5.1.2	Unterklassen erben die Spezifikation von Oberklassen ...	155
5.1.3	Das Prinzip der Ersetzbarkeit	159
5.1.4	Abstrakte Klassen, konkrete Klassen und Schnittstellen-Klassen	165

5.1.5	Vererbung der Spezifikation und das Typsystem	174
5.1.6	Sichtbarkeit im Rahmen der Vererbung	181
5.2	Polymorphie und ihre Anwendungen	191
5.2.1	Dynamische Polymorphie am Beispiel	193
5.2.2	Methoden als Implementierung von Operationen	198
5.2.3	Anonyme Klassen	207
5.2.4	Single und Multiple Dispatch	208
5.2.5	Die Tabelle für virtuelle Methoden	227
5.3	Die Vererbung der Implementierung	238
5.3.1	Überschreiben von Methoden	240
5.3.2	Das Problem der instabilen Basisklassen	248
5.3.3	Problem der Gleichheitsprüfung bei geerbter Implementierung	253
5.4	Mehrfachvererbung	260
5.4.1	Mehrfachvererbung: Möglichkeiten und Probleme	260
5.4.2	Delegation statt Mehrfachvererbung	267
5.4.3	Mixin-Module statt Mehrfachvererbung	270
5.4.4	Die Problemstellungen der Mehrfachvererbung	272
5.5	Statische und dynamische Klassifizierung	288
5.5.1	Dynamische Änderung der Klassenzugehörigkeit	289
5.5.2	Entwurfsmuster Strategie statt dynamischer Klassifizierung	293

6 Persistenz 299

6.1	Serialisierung von Objekten	299
6.2	Speicherung in Datenbanken	300
6.2.1	Relationale Datenbanken	300
6.2.2	Struktur der relationalen Datenbanken	301
6.2.3	Begriffsdefinitionen	302
6.3	Abbildung auf relationale Datenbanken	307
6.3.1	Abbildung von Objekten in relationalen Datenbanken	307
6.3.2	Abbildung von Beziehungen in relationalen Datenbanken	311
6.3.3	Abbildung von Vererbungsbeziehungen auf eine relationale Datenbank	315
6.4	Normalisierung und Denormalisierung	320
6.4.1	Die erste Normalform: Es werden einzelne Fakten gespeichert	321

6.4.2	Die zweite Normalform: Alles hängt vom ganzen Schlüssel ab	323
6.4.3	Die dritte Normalform: Keine Abhängigkeiten unter den Nichtschlüssel-Spalten	325
6.4.4	Die vierte Normalform: Trennen unabhängiger Relationen	329
6.4.5	Die fünfte Normalform: Einfacher geht's nicht	331

7 Abläufe in einem objektorientierten System 337

7.1	Erzeugung von Objekten mit Konstruktoren und Prototypen	338
7.1.1	Konstruktoren: Klassen als Vorlagen für ihre Exemplare	338
7.1.2	Prototypen als Vorlagen für Objekte	342
7.1.3	Entwurfsmuster Prototyp	348
7.2	Fabriken als Abstraktionsebene für die Objekterzeugung	349
7.2.1	Statische Fabriken	352
7.2.2	Abstrakte Fabriken	355
7.2.3	Konfigurierbare Fabriken	360
7.2.4	Registaturen für Objekte	364
7.2.5	Fabrikmethoden	368
7.2.6	Erzeugung von Objekten als Singletons	377
7.2.7	Dependency Injection	386
7.3	Objekte löschen	397
7.3.1	Speicherbereiche für Objekte	397
7.3.2	Was ist eine Garbage Collection?	399
7.3.3	Umsetzung einer Garbage Collection	400
7.4	Objekte in Aktion und in Interaktion	412
7.4.1	UML: Diagramme zur Beschreibung von Abläufen	412
7.4.2	Nachrichten an Objekte	421
7.4.3	Iteratoren und Generatoren	421
7.4.4	Funktionsobjekte und ihr Einsatz als Eventhandler	433
7.4.5	Kopien von Objekten	442
7.4.6	Sortierung von Objekten	452
7.5	Kontrakte: Objekte als Vertragspartner	455
7.5.1	Überprüfung von Kontrakten	455
7.5.2	Übernahme von Verantwortung: Unterklassen in der Pflicht	457
7.5.3	Prüfungen von Kontrakten bei Entwicklung und Betrieb	470

7.6	Exceptions: Wenn der Kontrakt nicht eingehalten werden kann	471
7.6.1	Exceptions in der Übersicht	472
7.6.2	Exceptions und der Kontrollfluss eines Programms	478
7.6.3	Exceptions im Einsatz bei Kontraktverletzungen	484
7.6.4	Exceptions als Teil eines Kontraktes	488
7.6.5	Der Umgang mit Checked Exceptions	493
7.6.6	Exceptions in der Zusammenfassung	501

8 Module und Architektur 503

8.1	Module als konfigurierbare und änderbare Komponenten	503
8.1.1	Relevanz der Objektorientierung für Softwarearchitektur	503
8.1.2	Erweiterung von Modulen	505
8.2	Die Präsentationsschicht: Model, View, Controller (MVC)	511
8.2.1	Das Beobachter-Muster als Basis von MVC	512
8.2.2	MVC in Smalltalk: Wie es ursprünglich mal war	513
8.2.3	MVC: Klärung der Begriffe	514
8.2.4	MVC in Webapplikationen: genannt »Model 2«	518
8.2.5	MVC mit Fokus auf Testbarkeit: Model-View-Presenter	523

9 Aspekte und Objektorientierung 527

9.1	Trennung der Anliegen	527
9.1.1	Kapselung von Daten	531
9.1.2	Lösungsansätze zur Trennung von Anliegen	532
9.2	Aspektorientiertes Programmieren	539
9.2.1	Integration von aspektorientierten Verfahren in Frameworks	539
9.2.2	Bestandteile der Aspekte	540
9.2.3	Dynamisches Crosscutting	541
9.2.4	Statisches Crosscutting	548
9.3	Anwendungen der Aspektorientierung	551
9.3.1	Zusätzliche Überprüfungen während der Übersetzung ...	551
9.3.2	Logging	552
9.3.3	Transaktionen und Profiling	553
9.3.4	Design by Contract	556
9.3.5	Introductions	559
9.3.6	Aspektorientierter Observer	560

9.4	Annotations	563
9.4.1	Zusatzinformation zur Struktur eines Programms	563
9.4.2	Annotations im Einsatz in Java und C#	565
9.4.3	Beispiele für den Einsatz von Annotations	566
Anhang		575
A	Anhang	575
A.1	Verwendete Programmiersprachen	575
A.1.1	C++	575
A.1.2	Java	578
A.1.3	C#	581
A.1.4	JavaScript	582
A.1.5	CLOS	584
A.1.6	Python	587
A.1.7	Ruby	590
B	Literaturverzeichnis	593
B.1	Allgemeine Bücher zur Softwareentwicklung	593
B.2	Bücher über die UML und die verwendeten Programmier Sprachen	595
Index		597