

mitp

Sebastian Meyer • Torben Wichers

Objective-C 2.0

**Programmierung für Mac OS X
und iPhone**

Objekte, Klassen, Nachrichten,
Kategorien, Properties, Protokolle,
Ausnahmebehandlung

Foundation Framework, Memory
Management, Threading, Bundles

Design Patterns für Objective-C

Sebastian Meyer, Torben Wichers

Objective-C 2.0



Bibliografische Information Der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Bei der Herstellung des Werkes haben wir uns zukunftsbewusst für umweltverträgliche und wiederverwertbare Materialien entschieden. Der Inhalt ist auf elementar chlorfreiem Papier gedruckt.

ISBN 978-3-8266-5966-9

1. Auflage 2009

E-Mail: kundenbetreuung@hjr-verlag.de

Telefon: +49 89/2183-7928

Telefax: +49 89/2183-7620

© 2009 mitp, eine Marke der Verlagsgruppe Hüthig Jehle Rehm GmbH
Heidelberg, München, Landsberg, Frechen, Hamburg

-fachportal.de

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Lektorat: Sabine Schulz
Sprachkorrektorat: Petra Heubach-Erdmann
Satz: III-satz, Husby, www.drei-satz.de
Druck: Köppl & Schönfelder, Stadtbergen

Inhaltsverzeichnis

Danksagung	13
Über die Autoren	13
Einleitung	15
An wen richtet sich das Buch?	16
Aufbau des Buches	17
Beispiele und verwendete Werkzeuge	21
Typografische Konventionen	22
Historische Einordnung	23
Objective-C als Erweiterung zu ANSI-C	27
Teil I Die Programmiersprache Objective-C 2.0	31
I Objekte und Klassen	33
I.1 Was ist objektorientierte Programmierung?	33
I.2 Objekte und Dynamic Typing	35
I.3 Nachrichten und Dynamic Binding	37
I.4 Klassen	42
I.4.1 Static Typing als Alternative zum Dynamic Typing	44
I.4.2 Aufbau einer Klasse	45
I.4.3 Vererbung	49
I.4.4 Instanzvariablen	52
I.4.5 Methoden und Klassenmethoden	58
I.5 Erzeugung von Objekten	64
I.5.1 Initialisierungsmethoden	66
I.5.2 Convenience-Konstruktoren	69
I.6 Übungsaufgaben	72
2 Wie werden Nachrichten verarbeitet?	75
2.1 Übersetzung von Methodenaufrufen	75
2.2 Struktur einer Methodenimplementierung	78
2.3 Auflösung der dynamischen Bindung	80

2.4	Forward Invocation	83
2.5	Übungsaufgaben	87
3	Kategorien	89
3.1	Erweiterung durch Kategorien	89
3.2	Verstecken von Methoden	92
3.3	Kategorien und die Klasse »NSObject«	94
3.4	Extensions	96
3.5	Übungsaufgaben	98
4	Properties	101
4.1	Zugriffsmethoden	102
4.2	Deklarative Zugriffsmethoden	103
4.2.1	Synthetische Properties	104
4.2.2	Attribute für Properties	106
4.2.3	Eigene Implementierungen von Properties	110
4.2.4	Neudeklaration von Properties	112
4.3	Der Punkt-Operator	113
4.4	Übungsaufgaben	116
5	Protokolle	119
5.1	Protokolle und Hierarchien	120
5.2	Verwendung von Protokollen	122
5.3	Grenzen von Protokollen	129
5.4	Optionale Protokollmethoden	131
5.5	Informelle Protokolle	138
5.6	Übungsaufgaben	139
6	Ausnahmebehandlung	141
6.1	Auslösen von Ausnahmen	143
6.2	Fangen von Ausnahmen	148
6.3	Übungsaufgaben	154

Teil II Objective-C-Programmierung für Mac OS X und iPhone OS 157

7	Die Cocoa-Umgebung	159
7.1	Das Cocoa-Framework	161
7.1.1	Das Foundation Framework	162
7.1.2	Application Kit	165

7.2	Ein erstes Beispielprogramm mit dem Cocoa-Framework	167
7.2.1	Ein neues Projekt in Xcode	168
7.2.2	Eine Anwendung mit grafischer Oberfläche	171
7.3	Cocoa Touch auf dem iPhone.	178
7.3.1	Die erste iPhone-Anwendung	180
7.3.2	Ausführung auf dem iPhone.	184
7.4	Übungsaufgaben	189
8	Memory Management.	193
8.1	Erzeugen von neuen Objekten	199
8.2	Kopieren von Objekten	201
8.3	Reference Counting.	204
8.3.1	Besitz und Freigabe von Objekten	206
8.3.2	Autorelease Pools	215
8.3.3	Die dealloc-Methode.	221
8.3.4	Properties, Getter und Setter.	223
8.3.5	Zyklische Abhängigkeiten	225
8.3.6	Implementierung von Reference Counting.	230
8.4	Garbage Collection	237
8.4.1	Barrieren und Generationen	242
8.4.2	Die finalize-Methode	243
8.4.3	Schwache Referenzen	248
8.5	Übungsaufgaben	250
9	Grundlegende Klassen	253
9.1	Byteorientierte Daten	254
9.2	Die Klasse NSCoder	259
9.3	Zeichenketten	263
9.4	Zeit- und Datumswerte.	275
9.4.1	Zeitpunkte und -intervalle.	275
9.4.2	Kalender	277
9.4.3	Zeitzonen	281
9.4.4	Ausgabe und Erkennung von Zeitwerten	283
9.5	Kapselung primitiver Datentypen	287
9.6	Ausnahmen mit NSError	292
9.7	Fehler mit NSError	293
9.8	Übungsaufgaben	298

10	Collections	299
10.1	Arrays	300
10.1.1	Unveränderliche Arrays	300
10.1.2	Veränderliche Arrays	303
10.1.3	Sortieren von Arrays	309
10.1.4	Filtern von Arrays	314
10.2	Sets	316
10.2.1	Unveränderliche Sets	316
10.2.2	Veränderliche Sets	318
10.2.3	Beziehungen zwischen Sets	319
10.2.4	Counted Sets	322
10.3	Dictionaries	324
10.3.1	Unveränderliche Dictionaries	325
10.3.2	Veränderliche Dictionaries	327
10.3.3	Sortieren von Dictionaries	329
10.4	Durchlaufen von Collections	331
10.5	Übungsaufgaben	335
11	Eingabe und Ausgabe	337
11.1	Verzeichnisse und Dateien	337
11.2	Operationen im Dateisystem	344
11.3	Streams	351
11.3.1	InputStreams	353
11.3.2	OutputStreams	355
11.3.3	Nicht-blockierender Zugriff auf Streams	358
11.4	Höhere Ebenen der Ein- und Ausgabe	366
11.5	Übungsaufgaben	376
12	Introspektion	379
12.1	Introspektion über NSObject	380
12.2	Introspektion mit der Runtime Library	384
12.3	Metaprogrammierung	397
12.4	Übungsaufgaben	407
13	Design Patterns für Objective-C	411
13.1	Schnelle Iteration	412
13.1.1	Vorteile der schnellen Iteration	415
13.1.2	Das Protokoll NSFastEnumeration	415

13.2	Key-Value Coding	424
13.2.1	Terminologie des Key-Value Codings.	426
13.2.2	Zuweisen und Auslesen von Werten	427
13.2.3	Implementierung von 1:m-Beziehungen	434
13.2.4	Skalare Typen und Strukturen	437
13.2.5	Key-Value-Validierung	440
13.2.6	Performance	444
13.3	Observer-Pattern	445
13.3.1	Key-Value Observing	446
13.3.2	Benutzung eines NotificationCenter	457
13.4	Delegation	470
13.4.1	Delegates zur Auslagerung von Funktionalität	470
13.4.2	Delegates als Datenquelle	473
13.5	Übungsaufgaben	475
14	Threading.	479
14.1	Operationen	481
14.1.1	Parallelisierung von Operationen	484
14.1.2	Priorisierung von Operationen	488
14.1.3	Abhängigkeiten zwischen Operationen	490
14.1.4	Die Klasse NSInvocationOperation.	491
14.2	Threads.	492
14.3	Synchronisation von Threads	501
14.3.1	Nicht-blockierende Synchronisation.	503
14.3.2	Locks	507
14.3.3	Conditions.	513
14.4	Run Loops	514
14.5	Run-Loop-Quellen	524
14.5.1	Timer-Quellen	525
14.5.2	Perform-Selektor-Quellen	530
14.5.3	Port-basierte Quellen	534
14.5.4	Benutzerdefinierte Quellen.	547
14.6	Übungsaufgaben	550
15	Bundles.	553
15.1	Anatomie eines Bundles	554
15.2	Lokalisierung	554
15.3	Verwendung von Bundles.	561
15.4	Übungsaufgaben	568

Teil III Anhänge	571
A Lösungsvorschläge	573
A.1 Objekte und Klassen	573
A.2 Wie werden Nachrichten verarbeitet?	576
A.3 Kategorien	580
A.4 Properties	581
A.5 Protokolle	582
A.6 Ausnahmebehandlung	584
A.7 Die Cocoa-Umgebung	585
A.8 Memory Management	589
A.9 Grundlegende Klassen	592
A.10 Collections	594
A.11 Eingabe und Ausgabe	598
A.12 Introspektion	602
A.13 Design Patterns für Objective-C	603
A.14 Threading	608
A.15 Bundles	615
B Eine kurze C-Einführung	617
B.1 Aufbau eines C-Programms	617
B.1.1 Kommentare	620
B.1.2 Basisdatentypen und Konstanten	621
B.1.3 Funktionen	625
B.1.4 Variablen	631
B.1.5 Kompilieren mehrerer Dateien	636
B.2 Der C-Präprozessor	638
B.2.1 Definition von Makros	638
B.2.2 Einfügen von Dateien	642
B.2.3 Bedingte Übersetzung	646
B.2.4 Vordefinierte Namen	648
B.3 Anweisungen und Kontrollstrukturen	649
B.3.1 Bedingte Anweisungen	650
B.3.2 Schleifen	653
B.3.3 Sprunganweisungen	655
B.4 Ausdrücke und Operatoren	656
B.4.1 Arithmetische Operatoren	658
B.4.2 Vergleichsoperatoren	658
B.4.3 Bit-Manipulationen	659

B.4.4	Wertzuweisungen	659
B.4.5	Logische Operatoren	661
B.4.6	Typ-Operatoren.....	662
B.4.7	Operatorprioritäten	663
B.5	Weitere Datentypen.....	664
B.5.1	Pointer.....	665
B.5.2	Arrays	668
B.5.3	Strukturen.....	671
	Stichwortverzeichnis	675