

Inhaltsverzeichnis

	Über den Autor	15
	Vorwort	17
1	Warum zahlt es sich aus, Objective-C zu lernen?	19
1.1	Warum ausgerechnet Objective-C?	19
1.2	Welche Programme Sie hier programmieren werden	21
1.3	Was Sie mitbringen müssen	21
1.4	Wie Sie mit diesem Buch Erfolg haben	23
I	Arbeiten mit Xcode: Die ersten Programme	25
2	Von null auf hundert: Ihr erstes Programm!	27
2.1	Computer unter sich – Warum Sie überhaupt Programmiersprachen brauchen	27
2.2	Xcode von Apple	28
2.3	Die richtigen Anlagen für Ihr erstes Projekt	29
2.4	Hallo Welt	35
2.5	Der iPhone-Simulator	39
2.6	Was Xcode sonst noch alles kann	40
2.7	Ausgaben: Sie können auch anders	44
2.8	Aufgaben	45
3	Erste Anwendungen: Mit Ihrem Mac können Sie rechnen	47
3.1	Ein- und Ausgabe von Daten	48
3.2	Programme unabhängig von Xcode starten	52
3.3	Ausgabe von Daten	54
	3.3.1 Zeichenketten im Programm	54
	3.3.2 Ausgabe von Strings	54
	3.3.3 Umwandlung von Zahlen nach Strings	55
	3.3.4 Kombination mehrerer Ausgaben	57
3.4	Der umgekehrte Weg: Eingabe von Zahlen	59
3.5	Der Taschenrechner	60

3.6	Lesbaren Programmtext schreiben	62
	3.6.1 Gute Kommentare	62
	3.6.2 Formatieren von Programmtext und Namensgebung	65
3.7	Aufgaben	67
II	Richtig programmieren: Grundlagen von Objective-C	69
4	Alles über Variablen, Datentypen und Bedingungen mit if	71
4.1	Variablen: Es kommt auf die Werte an	72
4.2	Fließkommazahlen: Punkt, Punkt, Komma, Strich	74
4.3	Boolean – Nichts als die Wahrheit	76
	4.3.1 BOOL – Der (Daten-)Typ der Wahrheit	76
	4.3.2 Die if-Anweisung: Wahl zwischen zwei Alternativen	77
	4.3.3 Zahlen vergleichen – alles ist relativ	80
	4.3.4 Operationen mit BOOL – Rechnen mit Wahrheiten	81
4.4	Es gibt Konstanten im Variablen-Leben	82
4.5	Gültigkeit	83
4.6	Fortgeschrittenes	85
	4.6.1 Unterschiede zwischen Zahlentypen	85
	4.6.2 Unicode	87
4.7	Aufgaben	89
5	Funktionen, for-Schleifen und Verzweigungen mit else	91
5.1	Funktionen: Definieren und richtig benutzen	92
5.2	Praxisbeispiel: Fizz Buzz	93
	5.2.1 Die for-Schleife	94
	5.2.2 Mehrere Alternativen mit der else-Anweisung	95
	5.2.3 Division und der ganze Rest: Modulo-Funktion	97
	5.2.4 Das fertige FizzBuzz-Programm	97
5.3	Praxisbeispiel: Grammatisch korrekte Sätze mit zunehmenden Jahren	102
5.4	Praxisbeispiel: Zinsen – das zahl ich Ihnen zurück!	104
	5.4.1 Zinsen mit einer for-Schleife	104
	5.4.2 Was lohnt sich und was nicht?	106
5.5	Fortgeschrittenes	109
	5.5.1 Typen, die sich wandeln	109
	5.5.2 Speicher und Bytes	110
	5.5.3 Speichern von Strings	110
5.6	Aufgaben	112

6	Ihr erstes Spiel mit noch mehr Schleifen programmieren	113
6.1	Züge im Streichholzspiel	114
6.1.1	Zufallszahlen erzeugen	114
6.2	Züge des menschlichen Spielers	117
6.2.1	Schleifen mit der while-Anweisung	118
6.3	Das fertige Spiel im Hauptprogramm	120
6.3.1	Ich will raus! – Schleifen mit break verlassen	121
6.4	Refaktorisieren – Verschönern Ihrer Programme	124
6.5	Usability – Der Spaßfaktor	127
6.6	Fortgeschrittenes	130
6.6.1	Kontrolle – Ihre do-while-Schleife, bitte	130
6.7	Aufgaben	131
III	Die besondere Stärke:	
	Objektorientierte Programmierung mit Objective-C	133
7	Objekte – Was sind sie und wie können sie Ihnen helfen, Arbeit zu sparen?	135
7.1	Objekte in Objective-C	135
7.1.1	Definition von Objekten	136
7.1.2	Implementierung von Objekten	137
7.1.3	Verwendung von Objekten	139
7.1.4	Methodenaufrufe im Detail	141
7.2	Praxisbeispiel: Das Unternehmenspiel	142
7.2.1	Anlegen des Projektes und Ihres ersten Objektes	142
7.2.2	Die Definition des WSUnternehmer-Objektes	144
7.2.3	Die Implementierung des WSUnternehmer-Objektes	146
7.2.4	Das Hauptprogramm	148
7.2.5	Alternative Fälle: switch	151
7.3	Speichermanagement	152
7.4	Fortgeschrittenes	157
7.4.1	Namensräume	157
7.4.2	Eigenschaften von Klassen	158
7.4.3	Hintergrund: Müllschlucker und Erbsenzähler	159
7.4.4	Überwachung total	160
7.5	Aufgaben	162
8	Wie Sie sich mit Vererbung bei Objekten richtig Zeit und Mühe sparen können	163
8.1	Wie geht Vererbung?	163
8.1.1	Vererbung: Funktionsreichtum ohne Mehrarbeit	164
8.1.2	Den Arbeiter in das Unternehmenspiel aufnehmen	166

8.1.3	Für die Einstellung vorbereiten: Den Angestellten definieren .	169
8.1.4	Die Arbeit mit dem neuen Mitarbeiter	171
8.2	Polymorphismus ist keine Krankheit, sondern gut für Sie	173
8.2.1	Das Spiel mit Praktikanten	175
8.3	Protokolle: Gemeinsamkeiten über Klassengrenzen hinweg	177
8.4	Praxisbeispiel: Die Firma – das Computerspiel!	181
8.4.1	Die neuen Spielregeln	182
8.4.2	Die Firma: Alle für ein Objekt, ein Objekt für alle	183
8.4.3	Das neue Hauptprogramm	187
8.5	Fortgeschrittenes	189
8.5.1	SOLID – Solide Programme bauen	189
8.5.2	Klassenmethoden	191
8.6	Aufgaben	192
9	Was Cocoa-Objekte alles können: Arbeiten mit Strings und was Objekte gemeinsam haben	193
9.1	NSString: Dieses Wort ist mächtiger als das Schwert	193
9.1.1	So holen Sie Hilfe	194
9.1.2	Zahlen, Formate und Ersetzungen	198
9.1.3	Vergleiche von Strings	202
9.1.4	Suchen und Ersetzen	204
9.2	Praxisbeispiel: Schon wieder ein neues Passwort	205
9.2.1	Ein einzelnes Zeichen setzen	206
9.2.2	Lottospielen mit Ziffern	207
9.2.3	Jetzt aber richtig: Großbuchstaben, Kleinbuchstaben und alles zusammen	209
9.3	NSObject: Was alle Objekte gemeinsam haben	212
9.3.1	Erzeugen und Vernichten eines Objektes	213
9.3.2	Kopieren von Objekten	216
9.4	Fortgeschrittenes	220
9.4.1	Eine Variable für irgendein Objekt	220
9.4.2	Das RAII-Entwurfsmuster	220
9.4.3	Eigene Dokumentation	221
9.5	Aufgaben	222
10	Verladen von Objekten – Alles über Datencontainer	225
10.1	Arrays – sortiert nach Zahlen	226
10.1.1	Arrays benutzen – schnell und bequem	226
10.1.2	Arrays, die zu Mutanten werden	228
10.1.3	Zahlen und andere Daten	229
10.1.4	FizzBuzz mit Objekten	230
10.2	Dictionaries – Informationen sortiert abspeichern	238

10.3	Daten speichern und laden	239
10.3.1	Wo geht's hier zum Verzeichnis?	240
10.3.2	Daten speichern und wieder laden	241
10.3.3	Ich kann mir nicht alles merken!	242
10.4	Fortgeschrittenes	244
10.4.1	Vergleiche	245
10.4.2	Hashwerte	245
10.4.3	Wenn es mal schnell gehen muss	246
10.4.4	Mengen	247
10.4.5	Wo sind alle Arrays hin?	247
10.5	Aufgaben	248

IV Grafische Oberflächen: Richtig benutzerfreundliche Apps 251

11 Jetzt schreiben Sie grafische Oberflächen auf dem iPhone 253

11.1	Button, übernehmen Sie	253
11.1.1	Einen Button einfügen	254
11.1.2	Der Button bekommt Arbeit	255
11.2	Vom Programm zum UI: Es geht auch anders herum	260
11.2.1	Grafische Elemente als Eigenschaften	260
11.2.2	Beispiel: Den Button deaktivieren	263
11.3	Schiebung mit dem Regler: Arbeiten mit Slidern	263
11.3.1	Einen Slider einfügen	263
11.3.2	Den Slider näher kennenlernen: Bereiche und Voreinstellungen	266
11.4	Drehen ohne Schwindel	266
11.4.1	Der Simulator kriegt den Drehwurm	268
11.4.2	Regeln beim Rotieren	268
11.4.3	Auf den Kopf stellen bringt nichts	275
11.5	Fortgeschrittenes	277
11.5.1	Apps für die ganze Welt	277
11.6	Aufgaben	283

12 Wie Sie Tabellen und mehrere Bildschirme auf dem iPhone verwenden 285

12.1	Tabellen auf dem iPhone	285
12.2	Praxisbeispiel: FizzBuzz mit Tabellen	287
12.2.1	Die Vorlage für Tabellen	287
12.2.2	Das Einrichten der Datenquelle	291
12.2.3	Ganze Geschichten erzählen: Alle Bildschirme im Blick behalten	296
12.2.4	Navigation in Storyboards	297

12.2.5	Grafische Übergänge und Animationen	299
12.2.6	Das Ergebnis: Wenig Arbeit für Sie	302
12.3	MVC – Das Geheimrezept für richtig elegante Programme	303
12.4	Aufgaben	306
13	Nun sind Sie dran: Zwei grafische Apps auf dem iPhone	307
13.1	Praxisbeispiel: Wie Sie Ihren eigenen Webbrowser schreiben	308
13.1.1	Präsentation	310
13.1.2	Die Steuerung	311
13.1.3	Der fertige Netzbrowser	316
13.2	Passwörter	318
13.2.1	Modell der Passwörter	319
13.2.2	Präsentation der Passwörter	323
13.2.3	Steuerung der Passwörter	326
13.2.4	Wie Sie Ihre App richtig schön machen: Icons und Splashscreen	328
13.3	Fortgeschrittenes	336
13.3.1	Arbeiten im Hintergrund	336
13.3.2	Was Textfelder alles können	337
13.4	Aufgaben	338
14	Eine richtig umfangreiche App: Das Streichholzspiel auf dem iPhone	339
14.1	Das Projekt und die Vorlage	339
14.2	Das Datenmodell	341
14.2.1	Die bisherige Strategie	344
14.3	Die grafische Oberfläche	346
14.3.1	Der Spielbildschirm	346
14.3.2	Der Konfigurationsbildschirm	347
14.4	Die Steuerung	348
14.4.1	Die erste lauffähige Version	349
14.5	Mehr und schönere Funktionen	352
14.5.1	Gehe zurück auf Los, ziehe Hölzer ein	352
14.5.2	Strategien mit Polymorphismus	356
14.5.3	Eine alternative Strategie	357
14.5.4	Der Konfigurationsbildschirm	361
14.6	Fortgeschrittenes	370
14.6.1	Reaktion auf den roten Knopf	370
14.6.2	Entwurfsmuster	372
14.7	Aufgaben	372

15	Wie Sie die eigene Sicht der Dinge vermitteln: Ihre eigenen grafischen Ausgaben entwickeln	375
15.1	Ich zeige Sie an – unter iOS	376
15.2	Der grafische Streichholzhaufen	378
15.3	Streichhölzer – sie können auch anders	383
15.4	Einen Haufen machen	388
15.5	Grafische Animationen	392
15.5.1	Was können Sie grafisch animieren?	392
15.5.2	Wie Sie Animation durchführen können	393
15.5.3	Die Animation im Streichholzspiel	396
15.6	Lang lebe die Streichholz-App	398
15.7	Fortgeschrittenes	401
15.7.1	Makros	401
15.7.2	Strukturiertes Vorgehen	403
15.8	Aufgaben	404
16	Wie Sie für das iPad entwickeln und darauf portieren	405
16.1	Was macht das iPad besonders?	406
16.2	Passwörter auf dem iPad	407
16.2.1	Modell der Passwörter	407
16.2.2	Präsentation der Passwörter auf dem iPad	410
16.2.3	Steuerung der Passwörter auf dem iPad	411
16.3	iPhone und iPad – eine App für alle Situationen	412
16.3.1	Passwörter auf dem iPad – ein Rückblick	420
16.4	Streichhölzer auf dem iPad	421
16.4.1	Das Datenmodell	421
16.4.2	Die Präsentation	421
16.4.3	Die Steuerung	425
16.5	Fortgeschrittenes	431
16.5.1	Die Empfehlungen von Apple	431
16.6	Aufgaben	433
17	Wie Sie eine grafische Anwendung auf Mac OS X portieren	435
17.1	Besonderheiten von Mac OS X	435
17.2	Praxisbeispiel: Das Passwort-Programm mit grafischer Oberfläche unter Mac OS X	436
17.2.1	Das Datenmodell	438
17.2.2	Die Präsentation	439
17.2.3	Die Steuerung	443
17.3	Passwörter mit mehreren Fenstern	448

17.4	Fortgeschrittenes	451
17.4.1	Empfehlungen von Apple für Mac OS X	451
17.5	Aufgaben	451
V	Die nächsten Schritte	453
18	Was Sie häufig brauchen: Entwanzen von Programmen	455
18.1	Typische Fallstricke – Arten von Fehlern	455
18.1.1	Syntaktische und semantische Fehler	456
18.1.2	Usability-Probleme	456
18.1.3	Falsche Ergebnisse von Programmen	457
18.1.4	Umgang mit Fehlern	457
18.2	Verwenden des Debuggers	459
18.2.1	Praxisbeispiel: Ein fehlerhaftes FizzBuzz-Programm	460
18.2.2	Anforderungen	467
18.3	Testen, Testen und nochmals Testen	468
18.3.1	Was automatisierte Tests tun	469
18.3.2	Praxisbeispiel: Das HolzHaufen-Modell des Streichholzspiels	469
18.3.3	Wie Sie Tests noch einsetzen können	473
18.4	Fortgeschrittenes	473
18.4.1	Ausnahmen und Fehlerbehandlung	473
18.4.2	Versionsmanagement	477
18.5	Aufgaben	478
19	Wie Sie mit Problemlösungen Geld verdienen: Der weitere Weg zum Software-Ingenieur	479
19.1	Vom Programmierer zum Software-Ingenieur	479
19.2	Vorschläge formulieren und Lösungen unterbreiten	480
19.3	Wie lange dauert das?	482
19.4	Der wöchentliche Statusbericht	483
19.5	Kontakte knüpfen und gute Fragen stellen	484
VI	Anhänge	487
A	Geschichte und Bedeutung von Objective-C	489
B	Lösung der Aufgaben	493
	Stichwortverzeichnis	505