

# Der Inhalt (im Überblick)

	Einführung	xxix
1	Erste Schritte mit C#: <i>Apps im Handumdrehn</i>	1
2	Es ist alles bloß Code: <i>Unter der Motorhaube</i>	53
3	Objekte: Orientieren Sie sich!: <i>Aufgabenspezifischer Code</i>	101
4	Typen und Referenzen: <i>Es ist 10:00. Wissen Sie, wo Ihre Daten sind?</i>	141
	C# Workshop 1: <i>Ein Tag beim Rennen</i>	187
5	Kapselung: <i>Alles zu zeigen, ist nicht immer richtig</i>	197
6	Vererbung: <i>Der Stammbaum Ihres Objekts</i>	237
7	Schnittstellen und abstrakte Klassen: <i>Klassen, die Versprechen halten</i>	293
8	Enums und Auflistungen: <i>Daten in Massen speichern</i>	351
9	Dateien lesen und schreiben: <i>Speichere das Array, rette die Welt</i>	409
	C# Workshop 2: <i>Die Suche</i>	465
10	Windows Store-Apps mit XAML: <i>Ihre Apps auf die nächste Stufe bringen</i>	487
11	XAML, Datei-I/O und Datenkontraktserialisierung: <i>Entschuldigen Sie die Unterbrechung</i>	535
12	Exception-Handling: <i>Fehler-Prävention</i>	569
13	Captain Amazing: <i>Der Tod des Objekts</i>	611
14	Datenabfrage und App-Bau mit LINQ: <i>Bekommen Sie Ihre Daten in den Griff</i>	649
15	Events und Delegates: <i>Was Ihr Code macht, wenn Sie nicht gucken</i>	701
16	App-Entwurf mit dem MVVM-Muster: <i>Tolle Apps, außen wie innen</i>	745
	C# Workshop 3: <i>Invaders</i>	807
17	Bonusprojekt: <i>Erstellen Sie eine Windows Phone-App</i>	831
A	Was übrig bleibt: <i>Die Top 11 der Themen, die es nicht ins Buch geschafft haben</i>	845
	Index	877

## Der Inhalt (jetzt ausführlich)

### Einführung

**Ihr Gehirn und C#.** Sie versuchen, etwas zu lernen, und Ihr Hirn tut sein Bestes, damit das Gelernte nicht *hängen bleibt*. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z.B. für das Wissen darüber, welche Tiere einem gefährlich werden könnten oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, wie schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, etwas über C# zu wissen?

Für wen ist dieses Buch?	xxx
Wir wissen, was Ihr Gehirn denkt	xxxi
Metakognition	xxxiii
Machen Sie sich Ihr Hirn untertan	xxxv
Was Sie für dieses Buch brauchen	xxxvi
Lies mich	xxxvii
Die Fachgutachter	xxxviii
Danksagungen	xxxix

# 1 Erste Schritte mit C# Apps im Handumdrehn

## Sie wollen tolle Apps in Windeseile aufbauen?

Mit C# steht Ihnen eine **ausgezeichnete Programmiersprache** und ein **wertvolles Werkzeug** zur Verfügung. Dank der **Visual Studio IDE** werden Sie nie wieder Stunden damit verbringen müssen, obskuren Code zu schreiben, um einen simplen Button funktionsfähig zu machen. Noch besser ist, dass Sie **richtig coole Programme aufbauen können**, ohne dass Sie sich einprägen müssen, welche Teile Ihres Codes den *Namen* eines Buttons repräsentieren und welche für sein *Label* stehen. Klingt das interessant? Blättern Sie um, und legen Sie mit dem Programmieren los.

Warum Sie C# lernen sollten	2
C# und die Visual Studio IDE erleichtern Ihnen eine Menge Dinge	3
Was Sie in Visual Studio tun ...	4
Was Visual Studio für Sie tut ...	4
Angriff aus dem All!	8
Das werden Sie erstellen	10
Beginnen wir mit einer leeren Anwendung	12
Das Grid für Ihre Seite anpassen	18
Dem Grid Steuerelemente hinzufügen	20
Das Aussehen über Eigenschaften ändern	22
Steuerelemente machen das Spiel funktionsfähig	24
Sie haben die Bühne für das Spiel vorbereitet	29
Eine Methode, die etwas macht	31
Den Code für Ihre Methode einfügen	32
Letzte Schritte und erste Ausführung	34
Das haben Sie bislang getan	36
Den Spielablauf mit Timern steuern	38
Den Start-Button funktionsfähig machen	40
Führen Sie das Programm erneut aus	41
Fügen Sie Code ein, damit die Controls mit dem Spieler interagieren können	42
Jetzt ist das Spiel spielbar	45
Die Feinde zu Aliens machen	46
Splash-Screen und Kachel	47
Die App veröffentlichen	48
Nutzen Sie den Remote Debugger, um Ihre App auf einem anderen Rechner zu installieren	49
Das Remote-Debugging starten	50

# Es ist alles bloß Code

## Unter der Motorhaube

# 2

### Sie sind Programmierer, nicht bloß IDE-Benutzer.

Mit der IDE können Sie eine Menge Dinge erledigen, aber dennoch kann sie Sie nur ein bestimmtes Stück Ihres Wegs begleiten. Sicher gibt es eine Menge **sich wiederholender Aufgaben**, die Sie erledigen müssen, wenn Sie eine Anwendung aufbauen. Diese Aufgaben kann Ihnen die IDE wunderbar abnehmen. Aber die Arbeit mit der IDE ist *erst der Anfang*. Sie können Ihre Programme dazu bringen, noch viel mehr zu machen – und das geht nur, indem Sie **C#-Code schreiben**. Haben Sie das Programmieren einmal im Griff, gibt es *nichts mehr*, was Ihre Programme nicht tun könnten.

Wenn Sie das tun ...	54
... macht die IDE das	55
Wo Ihre Programme herkommen	56
Die IDE hilft Ihnen beim Programmieren	58
Anatomie eines Programms	60
Zwei Klassen können im gleichen Namensraum sein	65
Ihre Programme nutzen Variablen, um mit Daten zu arbeiten	66
C# nutzt vertraute mathematische Symbole	68
Mit dem Debugger Variablen beobachten	69
Mit Schleifen Aktionen wiederholen	71
if/else-Anweisungen fällen Entscheidungen	72
Eine App von null auf aufbauen	73
Die Buttons etwas tun lassen	75
Bedingungen aufstellen und prüfen	76
Windows Desktop-Apps lassen sich leicht erstellen	87
Die App für Windows Desktop neu erstellen	88
Ihre Desktop-App weiß, wo sie anfangen muss	92
Sie können den Einstiegspunkt Ihres Programms ändern	94
Wenn Sie Dinge in der IDE ändern, ändern Sie auch Ihren Code	96

Objekte: Orientieren Sie sich!

# 3

## Aufgabenspezifischer Code

### Jedes Programm, das Sie schreiben, löst ein Problem.

Bevor Sie ein Programm erstellen, überlegen Sie sich am besten, welches *Problem* Ihr Programm lösen soll. Deswegen sind **Objekte** so ungemein hilfreich. Mit diesen können Sie Ihren Code auf Basis des Problems strukturieren, das er lösen soll. Und dann können Sie Ihre Zeit damit verbringen, *über das Problem nachzudenken*, an dem Sie arbeiten müssen, anstatt gleich beim Schreiben des Codes im Sumpf der Implementierungsanforderungen zu versinken. Wenn Sie Objekte richtig einsetzen, erhalten Sie Code, der *intuitiv* zu schreiben sowie leicht zu lesen und zu ändern ist.

Wie Mark über sein Problem nachdenkt	102
Wie Marks Navi über das Problem nachdenkt	103
Die Methoden von Marks Navigator-Klasse	104
Nutzen Sie das Gelernte, um eine einfache Anwendung zu erstellen	105
Was wir damit erreicht haben	106
Mark hat eine Idee	107
Mark kann sein Problem mit Objekten lösen	108
Sie nutzen eine Klasse, um ein Objekt zu erstellen	109
Wenn Sie ein neues Objekt einer Klasse erstellen, bezeichnet man dieses als Instanz der Klasse	110
Eine bessere Lösung ... dank Objekten!	111
Eine Instanz nutzt Felder, um Dinge festzuhalten	116
Erstellen wir mal ein paar Instanzen!	117
Objekte auf dem Speicher	118
Was Ihr Programm im Kopf hat	119
Sie können Klassen- und Methodennamen verwenden, um Ihren Code intuitiv zu machen	120
Geben Sie Ihren Klassen eine natürliche Struktur	122
Klassendiagramme helfen Ihnen, Ihre Klassen auf sinnvolle Weise zu organisieren	124
Eine Klasse für die Arbeit mit ein paar Typen erstellen	128
Ein Projekt für die Jungs erstellen	129
Erstellen Sie ein Formular zur Interaktion mit den Jungs	130
Die Objektinitialisierung kann noch einfacher sein	133

## Typen und Referenzen

# 4

### Es ist 10:00. Wissen Sie, wo Ihre Daten sind?

**Datentyp, Datenbank, Lieutenant Commander Data ... all das ist wichtiges Zeug.** Ohne Daten sind Ihre Programme nutzlos. Sie brauchen **Informationen** von Ihren Anwendern und nutzen diese, um andere Informationen nachzuschlagen oder neue Informationen zu produzieren, die Sie dann wieder an Ihre User zurückliefern. Eigentlich schließt fast alles, was Sie beim Programmieren tun, auf die eine oder andere Weise **Arbeit mit Daten** ein. In diesem Kapitel werden Sie alles über die **Datentypen** von C# lernen, erfahren, wie Sie in Ihrem Programm mit Daten arbeiten, und Sie werden sogar auf einige schmutzige Geheimnisse bei **Objekten** stoßen (*psst ... auch Objekte sind Daten*).

Der Typ einer Variablen bestimmt, welche Art Daten sie speichern kann	142
Eine Variable ist wie ein Daten-to-go-Becher	144
10 Kilo Daten in einem 5-Kilo-Sack	145
Auch eine Zahl der richtigen Größe können Sie nicht einfach irgendeiner Variablen zuweisen	146
Wandeln Sie einen zu großen Wert um, passt C# ihn automatisch an	147
Einige Umwandlungen macht C# automatisch	148
Beim Aufruf einer Methode müssen die Variablen den Typen der Parameter entsprechen	149
Den Kilometerrechner debuggen	153
Einen Operator mit = kombinieren	154
Auch für Objekte braucht man Variablen	155
Mit Referenzvariablen auf Objekte verweisen	156
Referenzen sind wie Etiketten für Ihr Objekt	157
Gibt es keine Referenzen mehr, holt die Müllabfuhr Ihr Objekt	158
Mehrere Referenzen und ihre Nebenwirkungen	160
Zwei Referenzen bedeuten ZWEI Möglichkeiten, die Daten eines Objekts zu ändern	165
Ein Sonderfall: Arrays	166
Arrays können auch einen Haufen Referenzvariablen speichern	167
Willkommen beim Sandwich-Discount vom Strammen Max	168
Objekte nutzen Referenzen, um zu kommunizieren	170
Wohin noch kein Objekt gekommen ist	171
Erstellen Sie ein Tippspiel	176

# 5 Kapselung

## Alles zu zeigen, ist nicht immer richtig

### Denken Sie auch, dass man manches nicht gleich allen offenbaren muss?

Manchmal geht es Ihren Objekten ebenfalls so. Genau so, wie Sie nicht jeden Ihr Tagebuch oder Ihre Kontoauszüge lesen lassen, lassen gute Objekte **andere** Objekte nicht in ihre Felder blicken. In diesem Kapitel werden Sie die Macht der **Kapselung** kennenlernen. Sie **werden die Daten Ihrer Objekte privat machen** und Methoden hinzufügen, **die schützen, wie auf diese Daten zugegriffen wird.**

Kathrin ist Event-Managerin	198
Was der Kostenrechner macht	199
Sie werden ein Programm für Kathrin erstellen	200
Kathrins Testlauf	206
Jede Option sollte einzeln berechnet werden	208
Objekte werden leicht unabsichtlich missbraucht	210
Kapselung bedeutet, dass man einige Daten in einer Klasse privat hält	211
Nutzen Sie Kapselung, um den Zugriff auf die Methoden und Felder Ihrer Klassen zu steuern	212
Aber ist das Feld WahrerName WIRKLICH geschützt?	213
Auf private Felder und Methoden kann nur aus der Klasse selbst heraus zugegriffen werden	214
Kapselung hält Ihre Daten rein	222
Eigenschaften vereinfachen die Kapselung	223
Eine Anwendung zum Testen der Klasse Bauer erstellen	224
Automatische Eigenschaften	225
Was ist, wenn wir den FutterMultiplikator ändern möchten?	226
Initialisieren Sie private Felder mit einem Konstruktor	227

## Vererbung

## 6

**Der Stammbaum Ihres Objekts**

**Es gibt Momente, da *WILL* man wie die eigenen Eltern sein.**

Ist Ihnen schon einmal ein Objekt über den Weg gelaufen, das **fast** genau das macht, was **Ihr** Objekt machen soll? Und haben Sie sich gewünscht, Sie **bräuchten einfach nur ein paar Dinge zu ändern** und das Objekt wäre perfekt? Genau das ist der Grund dafür, dass **Vererbung** eins der mächtigsten und wichtigsten Konzepte in C# ist. Bevor Sie mit diesem Kapitel durch sind, werden Sie gelernt haben, wie Sie von einer Klasse **ableiten**, um ihr Verhalten zu bekommen, dabei aber die **Flexibilität** bewahren, Änderungen an diesem Verhalten vorzunehmen. Sie **vermeiden doppelten Code**, **modellieren die wahre Welt** präziser nach und erhalten Code, der **leichter zu warten** ist.

Kathrin plant auch Geburtstagsfeiern	238
Party-Planer 2.0	240
Eine Sache noch ... können Sie eine Gebühr von 100 € für Partys mit mehr als 12 Personen hinzufügen?	247
Nutzen Ihre Klassen Vererbung, müssen Sie Ihren Code nur einmal schreiben	248
Erstellen Sie Ihr Klassenmodell, indem Sie allgemein beginnen und immer spezifischer werden	249
Wie würden Sie einen Zoo-Simulator entwerfen?	250
Nutzen Sie Vererbung, um doppelten Code in Unterklassen zu vermeiden	251
Die Klassenhierarchie erstellen	254
Jede Unterklasse erweitert Ihre Basisklasse	255
Erweiterungen geben Sie mit einem Doppelpunkt an	256
Eine Unterklasse kann Methoden überschreiben, um geerbte Methoden zu ändern oder zu ersetzen	260
Überall dort, wo Sie die Basisklasse verwenden können, können Sie stattdessen ein Unterklasse verwenden	261
Unterklassen können Oberklassenmethoden ausblenden	268
Nutzen Sie override und virtual, um Verhalten zu ändern	270
Basisklassenzugriff über das Schlüsselwort base	272
Hat die Basisklasse einen Konstruktor, braucht ihn die Unterklasse auch	273
Jetzt können Sie die Arbeit für Kathrin abschließen!	274
Bauen Sie eine Bienenstockverwaltung	279
Wie Sie die Bienenstockverwaltung aufbauen werden	280
Erweitern Sie die Bienenverwaltung mit Vererbung	287

## 7

## Schnittstellen und abstrakte Klassen

**Klassen, die Versprechen halten****Taten sagen mehr als Worte.**

Gelegentlich müssen Sie Ihre Objekte auf Basis dessen gruppieren, **was sie tun können, und** nicht auf Basis der Klassen, von denen sie erben. Dann kommen **Schnittstellen** ins Spiel – diese ermöglichen Ihnen, mit jeder Klasse zu arbeiten, die bestimmte Aufgaben erledigen kann. Aber mit **großer Macht geht eine große Verantwortung einher**: Jede Klasse, die eine Schnittstelle implementiert, muss versprechen, **all ihre Verpflichtungen zu erfüllen ...** oder der Compiler tritt Ihnen vors Schienbein, Verstanden()?

Kehren wir zu den Bienen zurück	294
Mit Vererbung können wir Klassen für die unterschiedlichen Arten von Bienen erstellen	295
Eine Schnittstelle sagt einer Klasse, dass sie bestimmte Methoden und Eigenschaften implementieren muss	296
Schnittstellen definieren Sie mit dem Schlüsselwort interface	297
Klassen, die Schnittstellen implementieren, müssen ALLE Methoden der Schnittstelle einschließen	299
Sammeln Sie Erfahrung im Umgang mit Schnittstellen	300
Eine Schnittstelle können Sie nicht instantiieren, aber Sie können sie referenzieren	302
Schnittstellenreferenzen funktionieren wie Objektreferenzen	303
Mit »is« können Sie ermitteln, ob eine Klasse eine bestimmte Schnittstelle implementiert	304
Eine Schnittstelle kann von einer anderen erben	305
Eine RoboBiene kann die Arbeit von Arbeiterinnen erledigen und braucht keinen Honig	306
Umwandeln funktioniert mit Objekten und Schnittstellen	309
Mit Downcasting können Sie Ihr Gerät wieder zu einer Kaffeemaschine machen	310
Upcasting und Downcasting funktionieren auch mit Schnittstellen	311
Es gibt nicht nur public und private	315
Zugriffsmodifizierer ändern die Sichtbarkeit	316
Manche Klassen sollten nicht instantiiert werden	319
Eine abstrakte Klasse ist vergleichbar mit einer Kreuzung aus normaler Klasse und Schnittstelle	320
Manche Klassen sollten nicht instantiiert werden	322
Eine abstrakte Methode hat keinen Rumpf	323
Der Deadly Diamond of Death!	328



## 8

## Enums und Auflistungen

**Daten in Massen speichern****Eigentlich tritt alles immer in Massen auf.**

Im wahren Leben wird man mit Daten nie in kleinen Fragmenten konfrontiert. Nein, sie begegnen Ihnen immer in **Massen, Bergen und Haufen**. Und Sie brauchen ziemlich leistungsfähige Werkzeuge, um sie zu organisieren. Genau das sind **Auflistungen**. Mit diesen können Sie alle Daten **speichern, sortieren und verwalten**, die Ihr Programm durchforsten muss. Sie können sich darauf konzentrieren, wie Ihr Programm mit den Daten arbeitet, und es den Auflistungen überlassen, sich für Sie um die Daten selbst zu kümmern.

Gelegentlich sind Strings nicht gut genug, wenn es darum geht, Kategorien von Daten zu speichern	352
Mit Enumerationen können Sie erlaubte Werte aufzählen	353
Mit Enums können Zahlen über Namen repräsentiert werden	354
Die Arbeit mit Arrays ist nicht immer einfach	358
Listen erleichtern das Speichern von Sammlungen ... jeder Art	359
Listen sind flexibler als Arrays	360
Listen schrumpfen und wachsen dynamisch	363
Generische Auflistungen können beliebige Typen speichern	364
Auflistungsinitialisierer funktionieren wie Objektinitialisierer	368
Listen sind leicht, aber SORTIEREN kann verzwickelt sein	370
Enten sortieren mit IComparable<Ente>	371
Mit IComparer sagen Sie Ihren Listen, wie sie sortieren sollen	372
Erstellen Sie eine Instanz Ihrer Comparer-Klasse	373
Komplexe IComparer	374
Die ToString()-Methode von Objekten überschreiben	377
foreach aktualisieren, damit sich Enten und Karten selbst darstellen	378
foreach nutzt ein IEnumerable<T>	379
Über IEnumerable ganze Listen upcasten	380
Auch Sie können Methoden überladen	381
Schlüssel und Werte speichern Sie in Wörterbüchern	387
Die Dictionary-Funktionalität im Überblick	388
Ein Programm mit einem Wörterbuch erstellen	389
Noch MEHR Auflistungstypen ...	401
Eine Queue ist FIFO – First-in-First-out	402
Ein Stack ist LIFO – Last-in-First-out	403

## Dateien lesen und schreiben

## 9

**Speichere das Array, rette die Welt**

**Gelegentlich zahlt es sich aus, wenn man Dinge festhält.** Bisher hatten alle Ihre Programme nur ein Kurzzeitgedächtnis. Sie starten, laufen eine Weile und enden dann. Manchmal reicht das nicht aus, insbesondere wenn Sie mit wichtigen Daten arbeiten. Sie müssen **Ihre Arbeit speichern können**. In diesem Kapitel werden wir uns ansehen, wie man **Daten in einer Datei speichert** und diese **Informationen dann wieder aus einer Datei einliest**. Sie werden etwas über die **.NET-Stream-Klassen** lernen und sich mit den Mysterien **Hexadezimal** und **Binär** auseinandersetzen.

C# nutzt Streams, um Daten zu lesen und zu schreiben	410
Verschiedene Streams lesen und schreiben verschiedene Dinge	411
Ein FileStream schreibt Bytes in eine Datei	412
Wie man Daten in eine Textdatei schreibt	413
Lesen und Schreiben mit zwei Objekten	417
Daten können durch mehrere Streams laufen	418
Öffnen Sie mit eingebauten Objekten Standarddialoge	421
Dialoge sind einfach ein weiteres .NET-Steuerelement	422
Mit den eingebauten Klassen File und Directory können Sie mit Dateien und Verzeichnissen arbeiten	424
Dateien mit vorgefertigten Dialogen öffnen und speichern	427
IDisposable sorgt dafür, dass Ihre Objekte entsorgt werden	429
Vermeiden Sie Dateisystemfehler mit using-Anweisungen	430
Wählen Sie Ihre Optionen mit einer switch-Anweisung	437
Ein überladener Kartenstapel()-Konstruktor, der einen Kartenstapel aus einer Datei einliest	439
Wird ein Objekt serialisiert, werden die Objekte, die es referenziert, ebenfalls serialisiert ...	443
Mit Serialisierung können Sie ein ganzes Objekt auf einmal lesen oder schreiben	444
.NET speichert Text in Unicode	449
Mit byte-Arrays können Daten verschoben werden	450
Binäre Daten schreiben Sie mit einem BinaryWriter	451
Sie können serialisierte Dateien auch manuell lesen und schreiben	453
Die Arbeit mit Binärdateien kann kompliziert sein	455
Mit Datei-Streams ein Hex-Dump-Programm erstellen	456
Mit Stream.Read() Bytes aus einem Stream lesen	458

## 10

## Windows Store-Apps mit XAML

**Ihre Apps auf die nächste Stufe bringen**

**Sie sind für ein neues Kapitel der App-Entwicklung bereit.**

Mit WinForms und dem Aufbau von Windows-Desktop-Programmen kann man wichtige C#-Konzepte leicht erlernen, aber *Ihre Programme könnten noch so viel mehr leisten.*

In diesem Kapitel werden Sie **XAML** nutzen, um Windows Store-Apps zu gestalten. Sie werden lernen, wie man **Seiten so aufbaut, dass sie auf alle Geräte passen**, wie man mit **Datenbindungen** Daten in Seiten **einbindet** und wie man sich mit Visual Studio einen Weg zu den innersten Geheimnissen von XAML-Seiten bahnt, indem man die Objekte untersucht, die von Ihrem XAML-Code erstellt werden.

Brian nutzt Windows 8	488
Windows Forms nutzen einen Objektgraphen	494
Den Objektgraphen mit der IDE erforschen	497
Windows Store-Apps erstellen UI-Objekte mit XAML	498
Verwandeln Sie Go Fish! in eine Windows Store-App	500
Das Seitenlayout setzt auf den Steuerelementen auf	502
Zeilen und Spalten können sich an die Seitengröße anpassen	504
App-Seiten mit dem Grid-System gestalten	506
Datenbindung verbindet Ihre XAML-Seiten mit Ihren Klassen	512
XAML-Steuerelemente können vieles enthalten	514
Datenbindung für den Strammen Max	516
Mit statischen Ressourcen Objekte im XAML deklarieren	522
Objekte mit einer Datenvorlage anzeigen	524
INotifyPropertyChanged benachrichtigt gebundene Objekte	526
Lassen Sie MenüMaker Änderungen des Erstellungsdatums melden	527

## 11

## XAML, Datei-I/O und Datenkontraktserialisierung

**Entschuldigen Sie die Unterbrechung****Niemand mag, warten müssen, insbesondere Nutzer nicht.**

Computer sind sehr geschickt, wenn es darum geht, mehrere Dinge auf einmal zu tun – es gibt also keinen Grund dafür, dass Ihre Apps das nicht auch können sollten. In diesem Kapitel werden Sie lernen, wie Sie Ihre Apps reaktionsfähig halten, indem Sie **asynchrone Methoden erstellen**. Sie werden auch erfahren, wie Sie die eingebauten **Dateiwähler- und Benachrichtigungsdialoge** sowie **asynchrone Dateieingabe und -ausgabe** nutzen, damit Ihre Apps nicht einfrieren. Kombinieren Sie das mit der **Datenkontraktserialisierung**, und Sie haben das Fundament einer absolut modernen App.

Brian hat Dateiprobleme	536
Windows Store-Apps nutzen await	538
Dateien schreiben und lesen Sie mit der Klasse FileIO	540
Ein etwas komplexerer Texteditor	542
Ein Datenkontrakt ist eine abstrakte Definition der Daten Ihres Objekts	547
Dateien mit asynchronen Methoden suchen und öffnen	548
KnownFolders hilft Ihnen beim Zugriff auf wichtige Ordner	550
Der gesamte Objektgraph wird in XML serialisiert	551
Schreiben Sie Typ-Objekte in den lokalen Ordner Ihrer App	552
Den Typ-Serialisierer testen	556
Mit einem Task eine asynchrone Methode aus einer anderen heraus aufrufen	557
Erstellen Sie Brian eine neue Ausredeverwaltung	558
Seite, Ausrede und Manager trennen	559
Die Hauptseite für die Ausredeverwaltung	560
Der Hauptseite eine App-Leiste hinzufügen	561
Die Klasse Ausredeverwaltung	562
Der Unterstützungscod für die Seite	564

## Exception-Handling

## Fehler-Prävention

## 12

**Programmierer sind keine Feuerlöscher.** Sie haben sich auf die Hinterbeine gesetzt, einen Haufen trockener Handbücher und ein paar ansprechende Von Kopf bis Fuß-Bücher durchgearbeitet und haben den Gipfel Ihres Berufsstands erreicht: **Meisterprogrammierer**. Dennoch erhalten Sie immer mitten in der Nacht panische Anrufe, weil **Ihr Programm abstürzt** oder **sich nicht so verhält, wie es sich verhalten soll**. Nichts kann einem die Programmierstimmung so verhegeln wie die Forderung, einen seltsamen Fehler zu beheben ... aber mit **Exception-Handling** kann sich Ihr Code **um die aufkommenden Probleme kümmern**. Und Sie können auf diese Probleme sogar reagieren und dafür sorgen, **dass das Programm weiterläuft**.

Brians Ausreden müssen mobil werden	570
Löst Ihr Programm eine Exception aus, erzeugt .NET ein Exception-Objekt	574
Brians Code machte etwas Unerwartetes	576
Alle Exception-Objekte erben von Exception	578
Der Debugger hilft Ihnen, Exceptions in Ihrem Code aufzuspüren und zu verhindern	579
Mist, der Code hat immer noch Probleme ...	583
Exceptions mit try und catch behandeln	585
Was passiert, wenn eine Methode gefährlich ist?	586
Folgen Sie try/catch mit dem Debugger	588
Nutzen Sie finally, wenn Sie Code haben, der IMMER ausgeführt werden soll	590
Nutzen Sie das Exception-Objekt, um Informationen zum Problem zu erhalten	595
Mehrere Typen von Exceptions mit mehreren Catch-Blöcken abfangen	596
Eine Klasse löst eine Exception aus, eine andere fängt sie ab	597
Ein einfaches Mittel, um viele Probleme zu vermeiden: using gibt Ihnen try und finally kostenlos	601
Exception-Vorsorge: Implementieren Sie IDisposable für eigene Aufräumarbeiten	602
Der SCHLIMMSTE catch-Block: Kommentare	604
Einige einfache Gedanken zum Exception-Handling	606

# 13

Ihre letzte Chance, etwas zu TUN ... der Finalisierer Ihres Objekts	618
Wann GENAU läuft ein Finalisierer?	619
Dispose() arbeitet mit using, Finalisierer mit der Garbage Collection	620
Finalisierer dürfen nicht von der Stabilität abhängen	622
Lassen Sie ein Objekt sich selbst in Dispose() serialisieren	623
Ein Struct <i>sieht aus</i> wie ein Objekt ...	627
... ist aber <i>nicht</i> auf dem Heap	627
Werte werden kopiert, Referenzen werden zugewiesen	628
Structs sind Werttypen, Objekte sind Referenztypen	629
Stack vs. Heap: mehr zum Speicher	631
Mit Parametern Methoden mehrere Werte liefern lassen	634
Mit dem Modifizierer ref Referenzen übergeben	635
Mit optionalen Parametern Standardwerte setzen	636
Nullbare Typen vertreten nicht existierende Werte	637
Nullbare Typen machen Programme robuster	638
Captain Amazing ... nicht ganz	641
Erweiterungsmethoden fügen BESTEHENDEN Klassen neue Verhalten hinzu	642
Einen elementaren Typ erweitern: string	644

## 14

## Datenabfrage und App-Bau mit LINQ

**Bekommen Sie Ihre Daten in den Griff****Die Welt ist datengesteuert ... Sie sollten damit zu leben wissen.**

Vorbei sind die Zeiten, in denen Sie Tage, sogar Wochen programmieren konnten, ohne sich mit **Massen von Daten** befassen zu müssen. Heute **dreht sich alles um Daten**. Und mit **LINQ** lässt sich all das bewältigen. Mit LINQ können Sie **Ihre Daten** nicht bloß auf leichte, intuitive Weise **abfragen**, sondern auch **gruppieren** und **Daten von unterschiedlichen Quellen zusammenführen**. Und wenn Sie Ihre Daten zu Happen gebündelt haben, mit denen man arbeiten kann, bieten Ihnen Windows Store-Apps **Steuerelemente**, über die Benutzer diese Daten erforschen, durchlaufen oder auch detailliert ansehen können.

Tim ist der größte Captain Amazing-Fan ...	650
... aber seine Sammlung ist ein einziges Chaos	651
LINQ kann Daten aus vielen Quellen ziehen	652
.NET-Auflistungen sind bereits für LINQ eingerichtet	653
LINQ vereinfacht Abfragen	654
LINQ ist einfach, Ihre Abfragen sind das nicht immer	655
Tim könnte unsere Hilfe gebrauchen	658
Bauen Sie Tims App auf	660
Mit dem Schlüsselwort new anonyme Typen erstellen	663
LINQ ist vielseitig	666
Neue Abfragen für Tims App	668
LINQ kann Ergebnisse zu Gruppen zusammenfassen	673
Tims Werte gruppieren	674
Zwei Auflistungen in einer Abfrage mit join kombinieren	677
Tim hat eine Menge Kohle gespart	678
Daten mit semantischem Zoom navigieren	684
Geben Sie Tims App semantischen Zoom	686
Sie haben Tim große Freude bereitet	691
Die Split App-Vorlage der IDE vereinfacht den Aufbau von Apps, die sich um die Navigation durch Daten drehen	692

## 15

## Events und Delegates

**Was Ihr Code macht, wenn Sie nicht gucken****Ihre Objekte beginnen, für sich selbst zu denken.**

Sie können nicht immer kontrollieren, was Ihre Objekte machen. Manche Dinge passieren einfach. Und wenn sie passieren, sollten Ihre Objekte schlau genug sein, **auf alles eine Antwort zu wissen**, was eintreten könnte. Darum geht es bei Events (oder Ereignissen). Ein Objekt *veröffentlicht* ein Event, andere Objekte *abonnieren* es, und alle arbeiten zusammen, damit die Dinge im Fluss bleiben. Das geht so lange gut, bis Sie steuern können wollen, wer ein Event abonnieren darf. Dann werden sich **Callbacks** als praktisch erweisen.

Möchten auch Sie, dass Ihre Objekte für sich selbst denken?	702
Aber woher weiß ein Objekt, dass es reagieren soll?	702
Wenn Events eintreten ... lauschen Objekte	703
Ein Objekt setzt ein Event ab, andere lauschen darauf ...	704
Dann behandeln andere Objekte das Event	705
Die Punkte verbinden	706
Die IDE erzeugt Event-Handler automatisch	710
Mit generischen Event-Handlern eigene Event-Typen definieren	716
Windows Forms nutzen viele verschiedene Events	717
Windows Store-Apps nutzen Events für die Abwicklung des Prozesslebenszyklus	720
Geben Sie Tims Comics eine Lebenszyklusverwaltung	721
XAML-Steuerelemente nutzen Routing-Events	724
Eine App zur Erforschung von Routing-Events	725
Event-Sender mit Event-Empfängern verbinden	730
Ein Delegate VERTRITT die eigentliche Methode	731
Delegates in Aktion	732
Jedes Objekt kann ein öffentliches Event abonnieren ...	735
Nutzen Sie Callbacks statt Events, um genau ein Objekt mit einem Delegate zu verbinden	736
Callbacks nutzen Delegates, aber KEINE Events	738
Callbacks und MatDialog-Befehle	740
Mit Delegates den Einstellungen-Charm von Windows nutzen	742



## 16

## App-Entwurf mit dem MVVM-Muster

## Tolle Apps, außen wie innen

## Beeindruckendes Aussehen allein reicht nicht.

Woran denken Sie, wenn Sie den Begriff Design hören? An ein Beispiel hervorragender Gebäudearchitektur? Eine aufregend gestaltete Seite? Ein Produkt, das gleichermaßen ästhetisch ansprechend wie gut gebaut ist? Genau diese Prinzipien gelten auch für Apps. In diesem Kapitel werden Sie das **Model-View-ViewModel-Muster** kennenlernen und erfahren, wie Sie mit seiner Hilfe gut gebaute, locker gebundene Apps aufbauen. Dabei werden Sie etwas über **Animationen und Control-Templates** für die visuelle Gestaltung Ihrer Apps erfahren, lernen, wie Sie sich Datenbindungen mit **Konvertierern** erleichtern und wie Sie all das zusammenbringen, um ein **solides C#-Fundament** für alle Apps zu gestalten, die Sie erstellen wollen.

Die Von Kopf bis Fuß-Basketball-Liga braucht eine App	746
Aber können Sie sich über die Gestaltung einigen?	747
Entwerfen Sie für die Datenbindung oder für die Arbeit mit Daten?	748
MVVM berücksichtigt Bindung und Daten	749
Mit dem MVVM-Muster die Basketball-App angehen	750
Eigene Steuerelemente erstellen	753
Der Schiri braucht eine Stoppuhr	761
MVVM heißt: Den Zustand der App im Blick	762
Das Model für die Stoppuhr-App	763
Events informieren den Rest der App über Zustandsänderungen	764
Den View für eine einfache Stoppuhr erstellen	765
Das Stoppuhr-ViewModel	766
Konvertierer wandeln Werte für Bindungen automatisch um	770
Konvertierer können mit unterschiedlichen Typen arbeiten	772
Visuelle Zustände lassen Steuerelemente auf Änderungen reagieren	778
Mit DoubleAnimation double-Werte animieren	779
Objektwerte mit Objektanimationen animieren	780
UI-Steuerelemente können auch mit C#-Code instantiiert werden	786
C# kann auch »echte« Animationen aufbauen	788
Das Bild mit einem Benutzersteuerelement animieren	789
Lassen Sie die Bienen über die Seite fliegen	790
Mit ItemsPanelTemplate Steuerelemente an ein Canvas binden	793

## Bonusprojekt

## 17

**Erstellen Sie eine Windows Phone-App****Sie können bereits Windows Phone-Apps schreiben.**

Klassen, Objekte, XAML, Kapselung, Vererbung, Polymorphie, LINQ, MVVM ... Sie haben alle Werkzeuge, die Sie zur Erstellung beeindruckender Windows Store- und Desktop-Apps benötigen. Aber wussten Sie auch, dass Sie die **gleichen Werkzeuge nutzen können, um Apps für *Windows Phones* zu erstellen?** Sie haben richtig gehört! In diesem Bonus-Projekt werden wir Sie durch den Aufbau eines Spieles für die Windows Phone-Plattform begleiten. Und sollten Sie kein Windows Phone-Gerät haben, können Sie den **Windows Phone-Emulator** nutzen, um es zu spielen. Legen wir los!

Bienenalarm!	832
Bevor Sie loslegen	833

## Anhang: Was übrig bleibt



### Die Top 11 der Themen, die es nicht ins Buch geschafft haben

#### Der Spaß fängt gerade erst an!

Wir haben Ihnen viele wunderbare Werkzeuge gezeigt, mit denen Sie mit C# richtig **mächtige Software** aufbauen können. Aber es war unmöglich, **alle Werkzeuge, Technologien und Techniken** in dieses Buch einzuschließen – so viele Seiten hat es einfach nicht. Wir mussten einige *sehr harte Entscheidungen* in Bezug darauf treffen, was wir aufnehmen und was wir weglassen. Hier sind ein paar Themen, die es nicht geschafft haben. Aber auch wenn wir zu ihnen nicht gekommen sind, denken wir trotzdem, dass sie **wichtig und nützlich** sind, und möchten Ihnen einen kleinen Ausblick auf sie bieten.

1. Der Windows Store hat noch viel mehr zu bieten	846
2. Die Grundlagen	848
3. Namensräume und Assemblies	854
4. Mit BackgroundWorker das UI reaktiver machen	858
5. Die Klasse Type und GetType()	861
6. Gleichheit, IEquatable und Equals()	862
7. Mit yield return enumerierbare Objekte erzeugen	865
8. Umgestalten (Refactoring)	868
9. Anonyme Typen und Methoden sowie Lambda-Ausdrücke	870
10. LINQ to XML	872
11. Windows Presentation Foundation	874
Wussten Sie, dass Sie mit C# und dem .NET Framework ...	875