EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

C# für IT-Berufe

```
using System;

namespace CSharp_IT_Berufe
{
  class Program
  {
     static void Main(string[] args)
     {
       Console.WriteLine("Informationsteil:");
       Console.WriteLine("- Einführung C#");
       Console.WriteLine("- Windows-Forms");
       Console.WriteLine("- WPF");
       Console.WriteLine("- Windows Store-Apps");
       Console.WriteLine();
       Console.WriteLine("Aufgabenpool");
       Console.WriteLine();
       Console.WriteLine("Lernsituationen");
       Console.WriteLine();
     }
  }
}
```

3. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 · 42781 Haan-Gruiten

Europa-Nr.: 85542

Verfasser:
Dirk Hardy, 46049 Oberhausen

Verlagslektorat:
Alexander Barth

3. Auflage 2015

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-8563-4

http://www.europa-lehrmittel.de
Satz: Reemers Publishing Services GmbH, Krefeld
Umschlagestaltung: braunwerbeagentur, 42477 Radevormwald
Druck: Medienhaus Plump GmbH, 53619 Rheinbreitbach

Vorbemerkung

Die Firma Microsoft suchte in den späten 90er-Jahren eine Antwort auf die enorm erfolgreiche Programmiersprache Java, die zugleich mit einer neuen Technologie verbunden war. Durch die virtuellen Maschinen, in denen der übersetzte Java-Quellcode ausgeführt wurde, war die Grundlage einer Plattformunabhängigkeit und weiterer Vorteile gegeben. Microsoft entwickelte daraufhin die Softwareplattform **.NET**, die die Möglichkeiten der Java-Technologie und zusätzliche Vorzüge haben sollte. Die Sprache **C#** wurde dann speziell für .NET entworfen. C# ist eine moderne und vollständig objektorientierte Sprache. Sie ist syntaktisch an die Sprache C++, konzeptionell aber eher an die Sprache Java angelehnt. Das Erlernen der Sprache C# beinhaltet auch die intensive Auseinandersetzung mit der .NET-Technologie. Diese Auseinandersetzung ist für die Ausbildung im IT-Bereich ein wichtiger Aspekt.

Aufbau des Buches

Das vorliegende Buch möchte die Sprache C# möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **drei Teile** getrennt. Der **erste Teil** des Buches dient als **Informationsteil** und bietet eine **systematische Einführung in die Sprache C# und in die Grundlagen von .NET.**

Ein ausführlicher Einstieg in die **Windows-Programmierung** rundet den Informationsteil ab. Dabei werden sowohl die klassische GUI-Programmierung mit *Windows-Forms* als auch die zukunftsweisende Programmierung mit der *Windows Presentation Foundation (WPF)* betrachtet, die mit einem Einstieg in die Programmierung von **Windows Store-Apps** endet. Die *WPF* ist ein Grafik-Framework, das die Geschäftslogik mithilfe der Auszeichnungssprache *XAML* von der Präsentationslogik trennt.

Der **zweite Teil** des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen. Zur **3. Auflage** wurden auch hier Aufgaben zur App-Entwicklung hinzugefügt.

Der **dritte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Lernfeld Entwickeln und Bereitstellen von Anwendungssystemen aus dem Rahmenlehrplan für die IT-Berufe (speziell Fachinformatiker-Anwendungsentwicklung). Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen (speziell Fachinformatiker), aber auch von den informationstechnischen Assistenten genutzt werden.

Als Entwicklungswerkzeug wird in diesem Buch Visual Studio 2015 von Microsoft genutzt. Diese Entwicklungsumgebung ist kostenfrei als Download im Internet verfügbar.

Für Anregungen und Kritik zu diesem Buch sind wir Ihnen dankbar (gerne auch per E-Mail).

Dirk Hardy

E-Mail: Hardy@DirkHardy.de

Im Sommer 2015

Verlag Europa-Lehrmittel

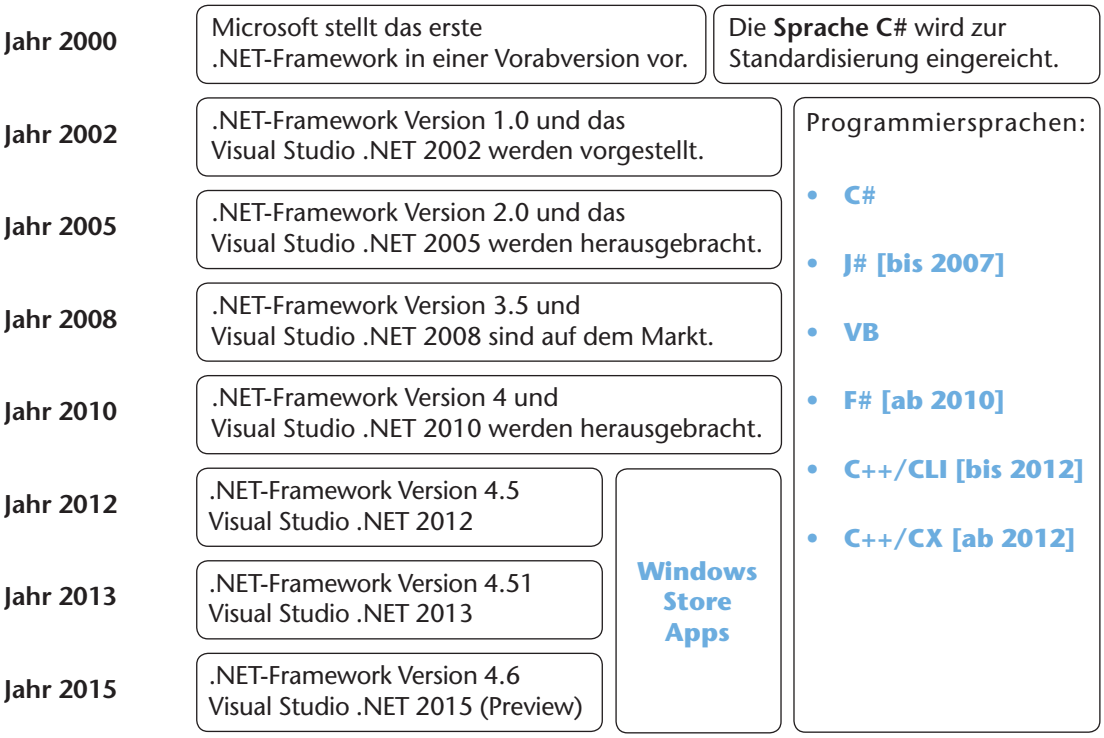
E-Mail: Info@Europa-Lehrmittel.de

1 Einführung in .NET und C#

1.1 Das .NET-Framework

1.1.1 Entstehung des Frameworks

In den 90er-Jahren wurde mit Java eine Technik geschaffen, die nicht nur sehr erfolgreich war, sondern auch die Zukunft von Microsoft im Bereich der Programmierung ernsthaft gefährden konnte. Das lag einerseits an der modernen objektorientierten Programmiersprache Java, aber auch an der Plattformunabhängigkeit von Java-Programmen, die mithilfe der Java-Laufzeitumgebung auf den verschiedensten Rechnern und Betriebssystemen ausgeführt werden können. Aus diesen Gründen brauchte Microsoft eine Antwort auf diese neue Technik – und zwar das .NET-Framework. Das Framework kann als eine Weiterentwicklung der Java-Technologie gesehen werden, allerdings zugeschnitten auf die Windows-Betriebssysteme. Inzwischen gibt es auch eine Linux-Variante des .NET-Frameworks, das so genannte MONO-Projekt. Die folgende Grafik zeigt den zeitlichen Verlauf der .NET-Framework-Entwicklung:



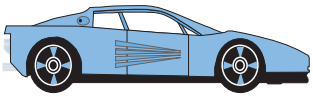
1.1.2 Eigenschaften des .NET-Frameworks

Der große Vorteil des .NET-Frameworks liegt darin, dass es auf den verschiedenen Windows-Betriebssystemen installiert werden kann. Damit sind Programme auf den verschiedenen Windows-Betriebssystemen lauffähig – vorausgesetzt, das entsprechende .NET-Framework ist vorhanden. Die verschiedenen Versionen des Frameworks können alle parallel installiert werden, so dass einem .NET-Programm immer die richtige Version zur Verfügung stehen kann. Die wichtigsten Eigenschaften des Frameworks sind:

- **Sprachunabhängigkeit**: Ein .NET-Programm kann in verschiedenen Sprachen geschrieben werden. Der Zugriff aus einer Sprache wie C# auf Klassen aus anderen Sprachen wie C++/CLI ist möglich.
- **Objektorientierung**: So wie in der Java-Technologie ist die Programmierung unter .NET vollständig objektorientiert.

Beispiel:

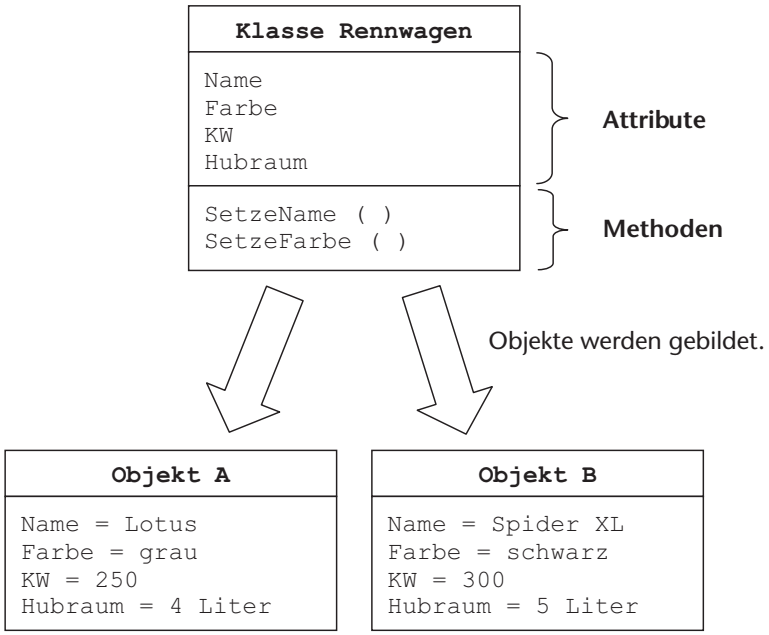
Diese Rennwagen sind konkrete Objekte. Sie haben Attribute wie Farbe, Leistung in KW und Hubraum.



Name:	Lotus
Farbe:	blau
KW:	250
Hubraum:	4 Liter

Name:	Spider XL
Farbe:	schwarz
KW:	300
Hubraum:	5 Liter

Beide Rennwagen haben dieselben Attribute. Sie unterscheiden sich nur in den Attributwerten. Der Spider XL hat beispielsweise eine höhere Leistung als der Lotus. Man könnte sagen, dass beide Rennwagen mithilfe desselben Bauplanes hergestellt worden sind. Der zugrunde liegende Bauplan könnte als **Klasse** Rennwagen bezeichnet werden. Die folgende Darstellung der Klassen und Objekte entspricht schon ungefähr der Form, die die formale Sprache UML benutzt, um Klassen und Objekte darzustellen.



Hinweise

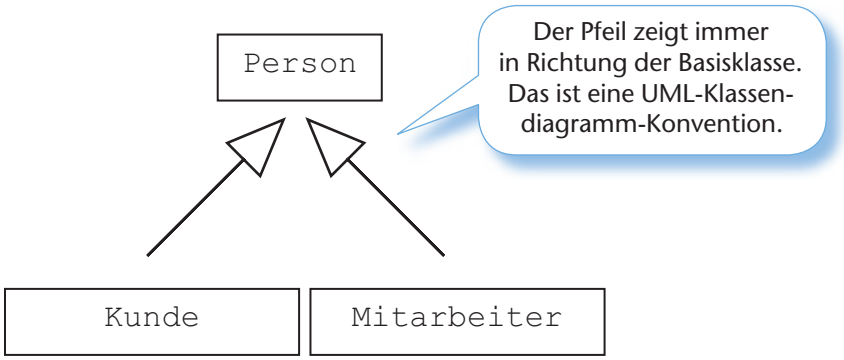
- Die Objektorientierung und die neuen Begriffe erscheinen gerade am Anfang recht abstrakt und es scheint nur wenig vorstellbar, wie eine neue Software objektorientiert programmiert werden soll. Dagegen hilft nur eins: Schritt für Schritt die Aspekte der objektorientierten Programmierung (OOP) kennen lernen und an konkreten Beispielen umsetzen. Gute objektorientierte Programmentwicklung hat auch viel mit Erfahrung zu tun.
- Neben der veränderten Sichtweise der Programmierung hat die OOP auch ganz praktische Vorteile gegenüber der strukturierten oder prozeduralen Programmierung. Diese Vorteile sind beispielsweise die Kapselung von Daten in den Objekten oder die Vererbung. Kapselung von Daten bedeutet, dass der Zugriff auf die Attribute eines Objektes kontrolliert abläuft. Dieser kontrollierte Zugriff geschieht über die Methoden eines Objektes. Dadurch wird beispielsweise verhindert, dass ein wichtiges Attribut eines Objektes aus Versehen mit einem falschen Wert beschrieben wird. Die Vererbung erspart dem Programmierer ungemein viel Arbeit, weil er einmal geschriebene Klassen an andere Klassen vererben kann.
- Das komplette Konzept der OOP wird allerdings erst dann richtig deutlich, wenn die Kapitel Klassenkonzept, Überladung von Operatoren, Vererbung und Polymorphismus bearbeitet wurden.

7 Vererbung in C#

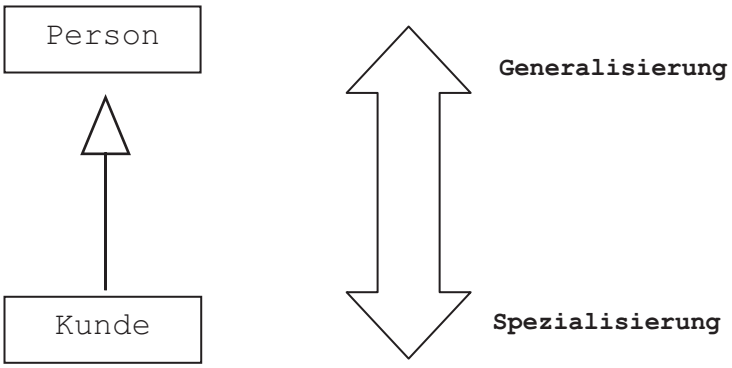
Das Konzept der Vererbung ist ein zentrales Thema in der OOP. Durch Vererbung können einerseits Situationen aus der „realen" Welt besser in die Programmiersprache umgesetzt werden, andererseits kann bereits existierender Programmcode (in Form von Klassen) wiederverwendet werden. Dadurch ergeben sich mehr Effizienz und Sicherheit in der Softwareentwicklung durch bereits vorhandenen und geprüften Programmcode. Bei der Vererbung spricht man von einer so genannten **Ist-Beziehung**.

Beispiel:

Die Basisklasse `Person` vererbt an die Klassen `Kunde` und Mitarbeiter. Der Kunde bzw. der Mitarbeiter sind eine Person (Ist-Beziehung). Der Kundenklasse bzw. Mitarbeiterklasse stehen nun alle Elemente der Basisklasse zur Verfügung (mit gewissen Einschränkungen, siehe später). Wenn beispielsweise die Personenklasse ein Attribut `name` hat, so erben sowohl die Kunden – als auch die Mitarbeiterklasse dieses Attribut.



Die Klasse `Kunde` bzw. `Mitarbeiter` ist eine spezielle Klasse `Person`. Die Klasse `Person` ist eine Verallgemeinerung der Klasse `Kunde` bzw. `Mitarbeiter`. Aus diesem Grund spricht man auch von Generalisierung und Spezialisierung.



Hinweis:

Die Klasse, die vererbt (`Person`), wird in der Regel Basisklasse oder Oberklasse genannt. Die Klasse, die erbt (`Kunde`), wird abgeleitete Klasse oder Unterklasse genannt.

7.1 Die Vererbung in C#

7.1.1 Die einfache Vererbung

Solange eine Klasse immer nur von einer Klasse erbt, spricht man von **einfacher Vererbung**. Die einfache Vererbung bedeutet aber nicht, dass nicht mehrere Klassen hintereinander erben können. Die folgenden Beispiele sind einfache Vererbungen.

Als Tabelle ist das zweidimensionale Array so vorstellbar:

	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	5	3	7	2
Zeile 1	8	6	9	1
Zeile 2	2	7	3	4

Der Zugriff auf ein Element des Arrays erfolgt nun mit einem **Doppelindex**. Beispielsweise ist das Element aus Zeile 0 und Spalte 1 so ansprechbar: `tabelle[ 0 , 1 ] == 3`

Dreidimensionale Arrays können sich als eine Sammlung von Tabellenblättern vorgestellt werden, die hintereinander angeordnet sind.

Beispiel:

Es wird ein dreidimensionales Array angelegt.

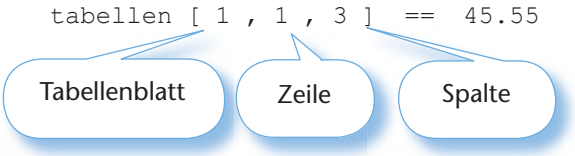
```
float  [ , , ] tabellen = new float [ 3 , 3 , 4 ];
```

Das dreidimensionale Array ist dann so vorstellbar:

Blatt 2	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	25.3

Blatt 1	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	45

Blatt 0	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	77.5
Zeile 1	124	99.99	453	67.89
Zeile 2	12	90.2	2727.5	22



Nach der dritten Dimension hört die menschliche Vorstellungskraft auf. Die mehrdimensionalen Arrays mit mehr als drei Dimensionen sind dann auch nicht mehr so konkret vorstellbar wie beispielsweise die Tabellen, aber dennoch sind sinnvolle Einsätze dieser Arrays denkbar.

Beispiel eines fünf-dimensionalen Arrays:

Für ein psychologisches Experiment werden drei unterschiedliche Gruppen mit jeweils 15 Probanden festgelegt. Jeder Proband erhält 10 Fragebögen mit jeweils 12 Fragen. Jede Frage hat 3 Antworten, die angekreuzt werden können. Das Array, das diesen Versuch widerspiegelt, könnte so aussehen:

```
bool [ , , , , ] experiment = new bool [3,15,10,12,3];
```

Es soll nun die Antwort des 5. Probanden aus Gruppe 2 für den 7. Fragebogen und die 4. Frage gespeichert werden. Die drei Antworten waren: `Ja, Nein, Ja.`

Der Einfachheit halber wird ein „Ja“ mit dem booleschen Wert `true`, ein „Nein“ mit dem booleschen Wert `false` gespeichert.

```
experiment [1,4,6,3,0] = true;
experiment [1,4,6,3,1] = false;
experiment [1,4,6,3,2] = true;
```

Mehrdimensionale Arrays durchlaufen

Mithilfe der Methode `GetLength(Index_Dimension)` kann die Tiefe einer einzelnen Dimension eines Arrays abgefragt werden. Damit können mehrdimensionale Arrays leicht durchlaufen werden, wie das folgende Beispiel zeigt:

11 Fortgeschrittene Themen in C#

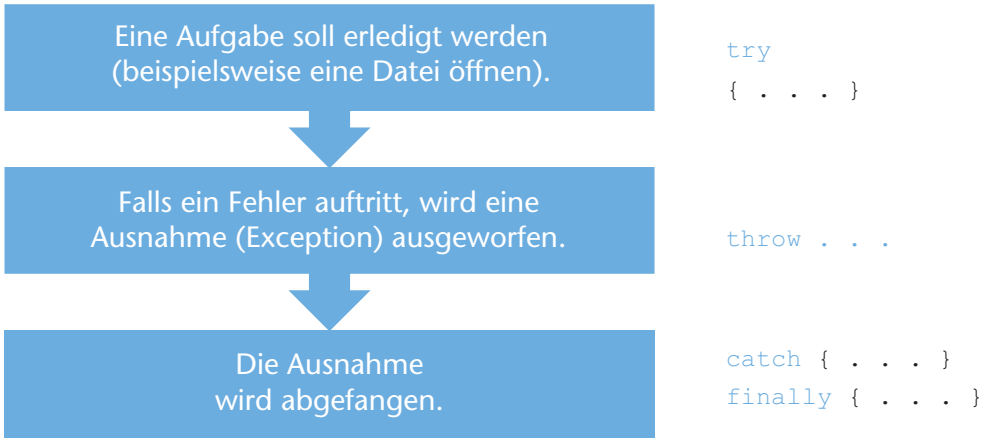
11.1 Ausnahmen – Exceptions

Das Abfangen von Fehlern ist eine wichtige Aufgabe in der Programmierung. Oftmals können Fehler durch Rückgabewerte von Methoden oder beispielsweise durch spezielle überladene Operatoren identifiziert werden. Der Nachteil dieser Vorgehensweise ist, dass es dem Programmierer selbst überlassen bleibt, ob er die Rückgabewerte bzw. Fehler auswertet und darauf reagiert oder nicht.

Mögliche Fehlerquellen sind:

- Über den reservierten Bereich eines Arrays schreiten
- Division durch null
- Eingabe von nicht erwarteten Zeichen über die Tastatur
- Fehler bei Dateioperationen
- Fehler bei Datenbankzugriffen

Die Ausnahmebehandlung in C# hilft dabei, diese Probleme zu bewältigen. Dabei wird die Fehlerbehandlung vom eigentlichen Programmcode separiert. Die folgende Abbildung zeigt den schematischen Ablauf einer Ausnahmebehandlung:



11.1.1 Versuchen und Auffangen (try and catch)

Die Ausnahmebehandlung startet mit dem so genannten `try`-Block. Innerhalb dieses Blockes steht der Programmcode, der möglicherweise einen Fehler verursachen kann. Deshalb das Schlüsselwort `try`- für einen Versuch. In dem folgenden Beispiel soll eine Zahl über die Tastatur eingelesen werden. Wenn der Benutzer allerdings Buchstaben statt Zahlen eingibt, dann wird eine Ausnahme „geworfen".

Beispiel:

```
using System;
namespace IT_BERUFE_CSHARP
{
    class Program
    {
        static void Main(string[] args)
        {
            int zahl;
```

```
    private string name;
    public CPerson()
    {
        name = "LEER";
    }

    public CPerson(string nameParam)
    {
        name = nameParam;
    }

    public override string ToString()
    {
        return name;
    }
}

class GenBeispiel<T>
{
    private T attribut;

    public GenBeispiel(T param)
    {
        attribut = param;
        Console.WriteLine("Wert des Attributes: " +
                          attribut.ToString());
        Console.WriteLine("Typ des Attributes: " +
                          attribut.GetType().ToString());
    }
}
class Program
{
    static void Main(string[] args)
    {

        GenBeispiel<int> integerObj;

        GenBeispiel<CPerson> personObj;

        integerObj = new GenBeispiel<int>(10);
```

Die Methode `ToString` implementieren

Den Typparameter direkt hinter dem Klassennamen angeben

Ein privates Attribut vom Typ `T`

Wert (mit `ToString`) und Typ ausgeben

Einen Verweis mit einem Integer-Typ anlegen

Einen Verweis mit einem Personentyp anlegen

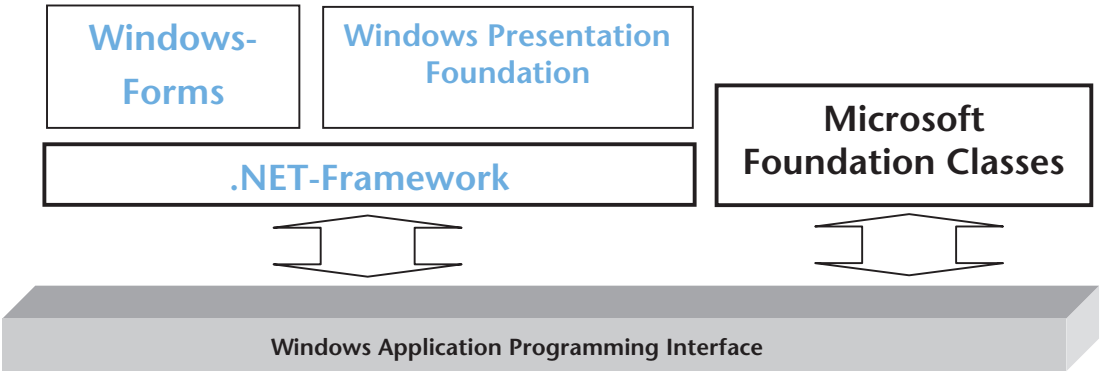
Eine Instanz der Klasse mit dem Integer-Typ

12 Windows-Forms-Programmierung – Grundlagen

12.1 Windows-Programmierung

12.1.1 Historische Entwicklung der Windows-Programmierung

Die Windows-Programmierung gibt es seit der Version Windows 1.0, die im Jahr 1985 vorgestellt wurde. Die Windows-Programme basieren seitdem auf Fenstern. Die Programmierung von Fenstern wird auch **GUI-Programmierung** genannt (für Graphical-User-Interface-Programmierung). Damit ist eine Interaktion zwischen Mensch und Maschine gemeint, die auf einer grafisch gestalteten Oberfläche basiert (im Gegensatz zur Konsolenanwendung). Besonders wichtig ist dabei auch der Einsatz eines Zeigegerätes wie der Maus. Die Windows-Programmierung geschieht über eine Schnittstelle, die so genannte **API** (Application Programming Interface). Diese Schnittstelle bietet alle Funktionalitäten, um ein Windows-Programm zu entwickeln. Die Funktionen der API sind in den Programmiersprachen C und Assembler geschrieben und damit sehr systemnah und schnell. Die direkte Windows-Programmierung nur mit der API ist durchaus möglich, aber relativ komplex. Eine Verbesserung bei der Windows-Programmierung bot eine objektorientierte Klassenbibliothek, die **MFC** (Microsoft Foundation Classes). Diese Bibliothek wurde 1992 mit den ersten Microsoft C/C++-Compilern ausgeliefert. Im Laufe der Jahre wurde die Bibliothek erweitert und ist auch heute noch ein Standard bei der Windows-Programmierung. Seit der Markteinführung des .NET-Framework gibt es eine weitere Alternative für die Windows-Programmierung – und zwar die Programmierung mit der Klassenbibliothek **Windows-Forms**. Das .NET-Framework kapselt die API und bietet eine objektorientierte Variante der Windows-Programmierung an. Die grundlegende Neuentwicklung der .NET-Technologie bringt auch für die Windows-Programmierung einige Vorteile im Vergleich zur konventionellen Windows-Programmierung. Neben der Windows-Programmierung mit Forms gibt es seit dem Betriebssystem Vista und dem .NET-Framework 3.0 eine weitere Bibliothek, die **Windows Presentation Foundation (WPF)**. Im Gegensatz zu Windows-Forms-Anwendungen, die beispielsweise auch unter Windows 98 lauffähig sind (vorausgesetzt, das .NET-Framework ist installiert) sind WPF-Anwendungen nur ab Windows XP mit Service Pack 2 lauffähig.



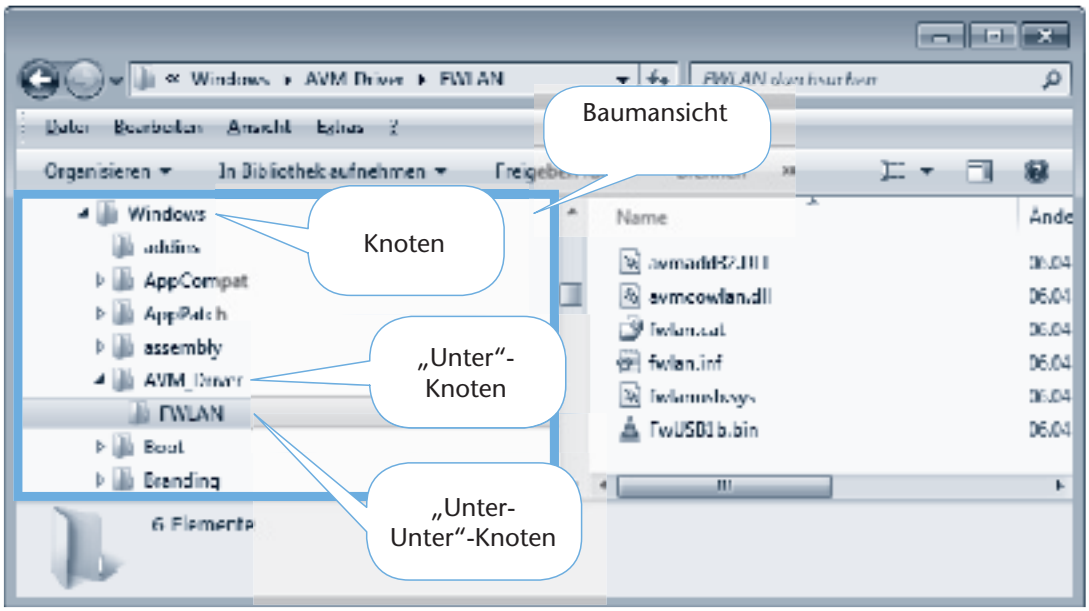
12.1.2 Ereignisgesteuerte Programmierung

Die bisherigen Konsolenprogramme haben mit dem Benutzer über Tastatureingaben kommuniziert. Dabei wartet ein Konsolenprogramm so lange, bis der Benutzer die Eingabe getätigt hat. Erst dann werden die nächsten Anweisungen ausgeführt. Bei der Windows-Programmierung wird ein anderes Konzept verwendet, um mit dem Benutzer zu interagieren – und zwar mithilfe der ereignisgesteuerten Programmierung. Der Benutzer kann dabei verschiedene Aktionen ausführen (beispielsweise auf einen Button klicken) und mit einem solchen Ereignis ist dann eine Methode verbunden, die ausgeführt wird (eine so genannte Ereignisbehandlungsmethode). Im Prinzip wartet ein Windows-Forms-Programm in einer Art Schleife darauf, dass ein Ereignis eintritt, welches behandelt werden

14 Komplexe Steuerelemente und Menüs

14.1 Die Baumansicht (TreeView)

Die Baumansicht (`TreeView`) ist ein Steuerelement, welches dem Windows-Benutzer sehr vertraut ist – und zwar durch den Windows Explorer, der in der linken Hälfte die Verzeichnisse in einer solchen Ansicht darstellt. Ebenso verwendet der Verzeichnis-suchen-Dialog (`FolderBrowserDialog`) eine solche Baumansicht. Die Baumansicht kann mit beliebigen Inhalten gefüllt werden, nicht nur mit Verzeichnissen und Dateinamen. Die Elemente einer Baumansicht werden Knoten (`Nodes`) genannt. Die folgende Abbildung zeigt eine typische Baumansicht im Windows Explorer:



Eine Baumansicht wird wie jedes andere Steuerelement aus der Toolbox auf dem Formular platziert. Der Designer legt dann automatisch ein Objekt vom Typ `TreeView` an:

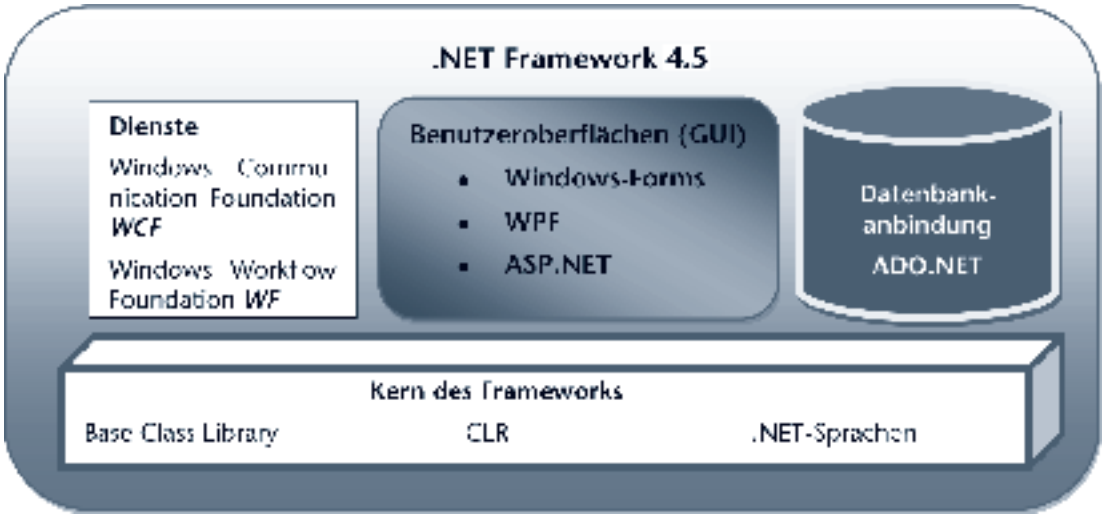
```
private System.Windows.Forms.TreeView treeView1;
private void InitializeComponent()
{
    this.treeView1 = new System.Windows.Forms.TreeView();
    :
}
```

15 Windows Presentation Foundation – Grundlagen

15.1 Windows-Programmierung mit WPF

15.1.1 Wichtige Aspekte der WPF

Mit der ***WPF*** (Windows Presentation Foundation) leitete Microsoft 2006 eine neue Ära der GUI-Programmierung unter Windows ein. Im Gegensatz zu Forms wurde ein grundlegend anderes Konzept entwickelt, das in einigen Aspekten der Java-Swing-Technologie ähnelt. Beispielsweise werden die grafischen Elemente nicht mehr mit den Funktionen der ***API*** (Application Programming Interface) gezeichnet, sondern mit WPF-eigenen Mitteln. Dadurch ergeben sich weitaus mehr Möglichkeiten zur Darstellung. Auch das Konzept der Ereignisbehandlung wurde deutlich verändert. In Zukunft setzt Microsoft deshalb auf die GUI-Entwicklung mit WPF, trotzdem wird Forms als zweite Möglichkeit noch lange Zeit Bestand haben, denn viele der modernen Anwendungen sind mit Forms umgesetzt worden, da auch die visuelle Unterstützung der WPF-Entwicklung erst mit der Entwicklungsumgebung ab ***Visual 2010*** eine echte Alternative zum Forms-Designer bietet. Parallel zur WPF hat Microsoft auch weitere neue Komponenten entwickelt, um beispielsweise Kommunikation und Workflow zu verbessern. Die folgende Übersicht zeigt die Bestandteile des .NET-Frameworks in der Version 4.0 mit den neuen Komponenten:



15.1.2 Eigenschaften der WPF

Die WPF bietet vielfältige Möglichkeiten eine Applikation zu gestalten. Neben den klassischen Windows-Desktop-Anwendungen werden sowohl 2-D-Grafiken als auch 3-D-Grafiken unterstützt. Dabei arbeitet WPF direkt mit der Grafikkarte zusammen, da es auf der DirectX-Technik basiert. Dadurch sind Farbübergänge, Schatten- und Spiegeleffekte und auch Animationen leicht realisierbar. Der Kern der WPF ist dabei ein **auflösungsunabhängiges, vektorbasiertes Renderingmodul**[1], mit dem die visuellen Elemente gezeichnet werden. Durch die vektorbasierte Darstellung sind die Elemente immer scharf gezeichnet, egal in welcher Größe sie angezeigt werden sollen. Das unterscheidet WPF beispielsweise auch von Windows-Forms. Die Größe der Grafiken werden dabei in geräteunabhängigen Pixeln berechnet. Ein solches Pixel entspricht 1/96 Zoll (ca. 0,26 mm). Damit ist gewährleistet, dass WPF-Anwendungen bei unterschiedlichsten Bildschirmauflösungen immer die korrekte Größe haben.

1 Unter Rendering versteht man das Zeichnen (Erstellen) eines grafischen Elements aus einem Modell oder einer bestimmten Vorlage.

16 XAML und der WPF-Designer

16.1 XAML

16.1.1 Extensible Application Markup Language XAML

Mit dem .NET-Framework 3.0 wurde die Auszeichnungssprache (*Markup*-Sprache) ***XAML*** eingeführt. Sie basiert auf XML und hat einen Aufbau mit eigenen Tags (engl. für Etiketten). Mit diesen Tags werden Elemente der GUI-Anwendung beschrieben. Das können Rechtecke, Buttons, Layout-Container oder andere Steuerelemente sein. Durch den XML-Aufbau werden die einzelnen Elemente verschachtelt, was genau dem Inhaltsprinzip der WPF entspricht. Der folgende Quellcode ist gültiger XAML-Code und beschreibt einen einfachen Button:

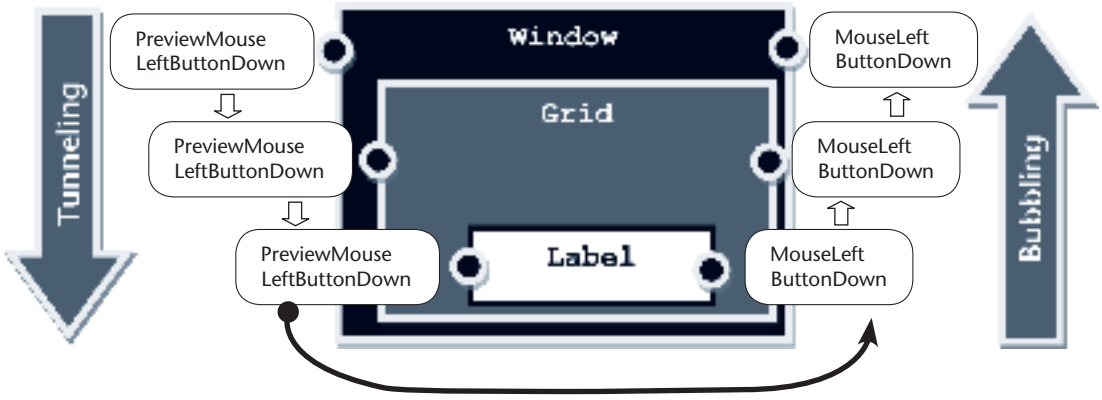
```
<Button xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
Name="knopf" FontSize="20pt" Margin="10,10,100,50">Ein Knopf</Button>
```

Das ist natürlich schwer zu interpretieren und wird deshalb noch einmal genauer beleuchtet:

Es ist erkennbar, dass ein XAML-Element immer ein öffnendes und ein schließendes Tag haben muss. Dazwischen werden Eigenschaften oder auch der Inhalt festgelegt. Speichert man den obigen XAML-Code in einer Datei mit der Endung „*.xaml“, dann kann diese Datei mit einem Doppelklick[1] gestartet werden und der Button wird im Internet-Explorer angezeigt:

1 Voraussetzung ist natürlich, dass das .NET-Framework 3.0 oder höher installiert ist.

Es zeigt sich, dass die sogenannten `Preview`-Ereignisse zuerst aufgerufen werden – und zwar in einer Reihenfolge vom äußeren Element (`Window`) zum inneren Element (`Label`). Danach werden die anderen Ereignisse aufgerufen, aber in umgekehrter Reihenfolge. Bei der ersten Richtung spricht man vom sogenannten ***Tunneling***, bei der anderen Richtung vom ***Bubbling***. Die folgende Grafik soll den Zusammenhang noch einmal verdeutlichen:



Hinweis:

Das Weiterleiten der Ereignisse kann vor allem dann sinnvoll sein, wenn beispielsweise ein Container über viele weitere Elemente verfügt und ebenfalls benachrichtigt wird, wenn irgendein Ereignis für die Elemente vorhanden ist. In manchen Fällen reicht es sogar aus, dass nur eine Ereignismethode auf Container-Ebene definiert ist. Diese Ereignismethode könnte dann differenziert auf Ereignisse der Elemente reagieren, indem sie die Quelle des Ereignisses abfragt:

```
private void Allgemeine_Ereignismethode(
                              object sender, MouseButtonEventArgs e)
{
        MessageBox.Show("Quelle: " + e.Source.ToString());
}
```

Für das obige Beispiel wäre die Quelle immer das `Label`.

16.4.3 Datenbindungen

Mithilfe des Konzepts der Datenbindungen können Steuerelemente sehr einfach an andere Objekte (auch Steuerelemente oder Datenbankobjekte) gebunden werden. Dabei ist ein Objekt immer das Ziel und ein Objekt immer die Quelle des Datenaustausches. Ein einfaches Beispiel wäre die Anbindung der `Content`-Eigenschaft eines `Buttons` an die `Text`-Eigenschaft einer `Textbox`. Sobald der Benutzer einen neuen Text in die `TextBox` eingibt, ändert sich auch die Beschriftung des `Buttons`:

Die Beschriftung des `Buttons` ändert sich immer parallel zum `TextBox`-Inhalt.

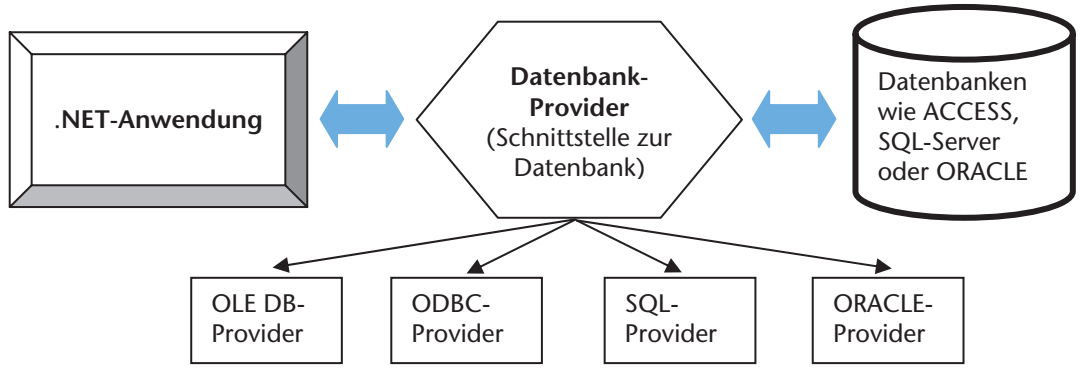
17 Datenbankanbindung mit Forms und WPF

17.1 Datenbankzugriff

Das Speichern von Daten kann eine Anwendung natürlich mithilfe von Dateioperationen selbst durchführen. Für wenige Daten ist das wahrscheinlich auch die erste Wahl bei der Entwicklung einer Anwendung, weil sie damit relativ unabhängig ist. Wenn allerdings viele Daten (oder Datensätze) zu speichern sind und die Daten zusätzlich einen komplizierten Aufbau haben, dann ist die Speicherung in einer Datenbank in Betracht zu ziehen. Der große Vorteil bei einer Datenbankanbindung ist die Unabhängigkeit der Anwendung von der technischen Umsetzung der Datenspeicherung. Das erledigt die Datenbank im Hintergrund. Auch das Ändern oder Löschen von Daten ist bequem durch einige Datenbankbefehle (SQL-Befehle) zu realisieren. Die Abfragesprache **SQL** (Structured Query Language) spielt hierbei eine wichtige Rolle. Bei den folgenden Ausführungen werden deshalb auch grundlegende Kenntnisse in SQL vorausgesetzt.

17.1.1 Datenbankanbindung unter dem .NET-Framework

Das .NET-Framework bietet eine Vielzahl von Klassen, um die Anbindung an eine Datenbank zu realisieren. Diese Klassen sind unter dem Oberbegriff **ADO.NET** gesammelt. Dabei steht ADO für *ActiveX Data Objects* und ist eine Erweiterung der bereits vorhandenen Technik von Microsoft. Mit ADO.NET kann ein Zugriff auf Datenquellen wie **SQL-Server** oder auch auf **OLE DB**- und **ODBC**-Datenquellen erfolgen. Die folgende Abbildung zeigt das Grundprinzip von ADO.NET:



Die einzelnen Provider (Datenanbieter) stehen dabei für bestimmte Datenbankanbindungen:

- **OLE DB-Provider**: OLE DB steht für **Object Linking and Embedding Database** und ist eine Technik, die bereits bei den Microsoft-Office-Anwendungen zum Einsatz kam. Beispielsweise ist es möglich, eine Excel-Tabelle in ein Word-Dokument so einzubinden, dass Änderungen an der Original-Tabelle auch immer in der Word-Tabelle sichtbar sind (und umgekehrt). Der OLE DB-Provider kann immer dann angewendet werden, wenn für eine Datenbank ein solcher Provider zur Verfügung steht (beispielsweise ACCESS).
- **ODBC-Provider**: ODBC steht für **Open Database Connectivity** und war eine der ersten Schnittstellen, die eine Vereinheitlichung des Datenbankzugriffs umsetzte. Jede Datenbank brauchte nur eine ODBC-Schnittstelle mitzuliefern und war damit für eine Windows-Anwendung einsetzbar.
- **SQL-Provider**: Dieser Provider stellt die Funktionalitäten für einen Zugriff auf den Microsoft SQL-Server zur Verfügung.
- **ORACLE-Provider**: Dieser Provider stellt die Funktionalitäten für einen Zugriff auf die ORACLE-Datenbank zur Verfügung.

Im Folgenden wird der Zugriff auf eine ACCESS-Datenbank mit dem OLE DB-Provider vorgestellt. Das Prinzip ist aber übertragbar auf andere relationale Datenbanken wie beispielsweise den Microsoft-SQL-Server. Für eine einfache Datenbankanbindung sind die Klassen `OleDbConnection`, `OleDbCommand` und `OleDbDataReader` nötig. Damit kann eine Verbindung zur Datenbank auf-

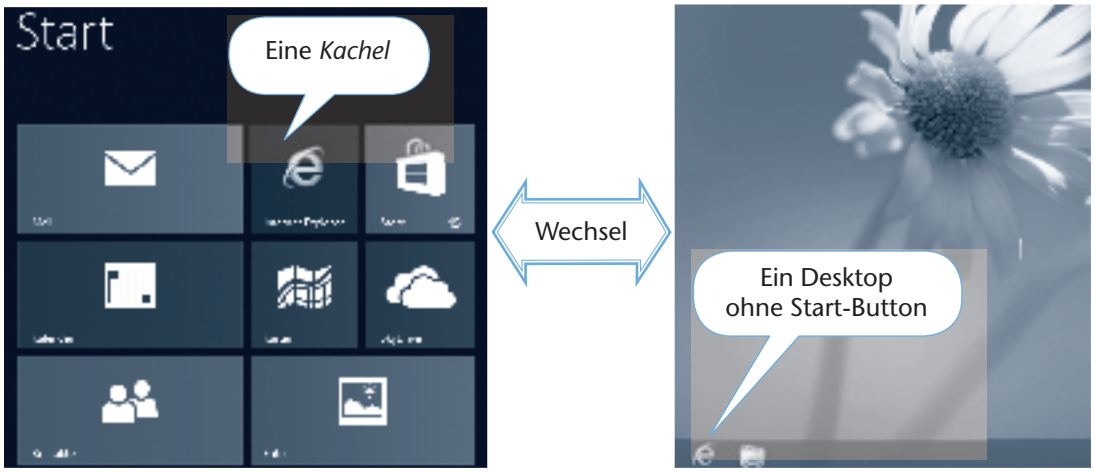
18 Windows Store Apps

18.1 Applikationen unter Windows 8

18.1.1 Neue Windows-Konzeption

Mit Windows 8 brachte Microsoft 2012 ein neues Betriebsystem heraus, das sich konzeptionell deutlich von den vorherigen Systemen unterscheidet. Auf der einen Seite ist es auf die Anforderungen der mobilen Geräte (Smartphones und Tablet-PCs) vorbereitet, indem es eine völlig veränderte Oberfläche anbietet, auf denen die Anwendungen in Form von *Kacheln* dargestellt werden. Diese Kacheln dienen auf den mobilen Geräten zur manuellen Eingabesteuerung (Touchscreen). Die Anwendungen, die mit solchen Kacheln gestartet werden, heißen *Windows Store-Apps* und zeichnen sich unter anderem dadurch aus, dass sie den ganzen Bildschirm ausfüllen und nicht mehr von einem Fensterrahmen begrenzt werden. Zusätzlich sind diese Apps ausschließlich über den Windows Store zu beziehen und werden in einer völlig anderen Form installiert als die konventionellen Windows-Applikationen. Die Entwicklung solcher Apps[1] wird das Thema der folgenden Kapitel sein.

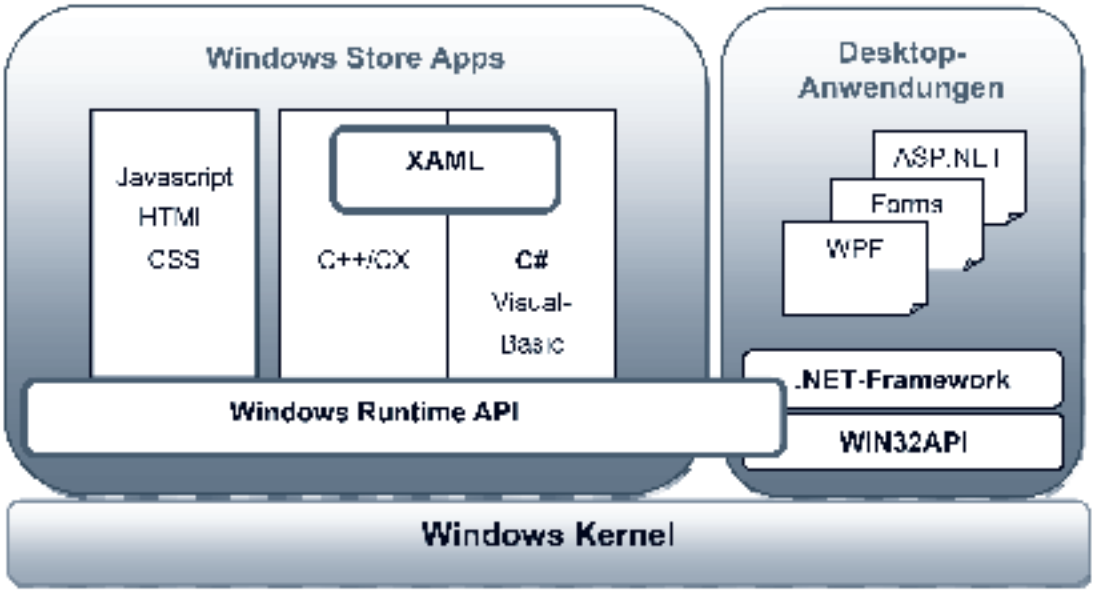
Auf der anderen Seite bietet Windows 8 aber auch noch eine klassische Desktop-Oberfläche, die ähnlich wie die Oberfläche von Windows 7 (oder auch Windows XP) zu nutzen ist. Es fehlt nur der gewohnte Start-Button, der aber einfach mit kostenfreien Tools nachzuinstallieren ist.



18.1.2 Windows Runtime

Die Entwicklung von Windows Store Apps basiert auf einer völlig neuen Schnittstelle, der **Windows Runtime API** (API = **A**pplication **P**rogramming **I**nterface). Diese objektorientierte Schnittstelle basiert auf dem COM[2]-Modell und wurde in C++ geschrieben. Im Gegensatz zu der .NET-Technologie steht die Windows Runtime für eine native Programmentwicklung. Das bedeutet, dass die Programmentwicklung sofort auf eine Schnittstelle zugreift, die Bestandteil des Betriebssystems ist. Damit entfällt eine Zwischenschicht wie unter .NET. Ein großer Vorteil ist die schnelle Geschwindigkeit solcher Anwendungen. Der native Zugriff auf die Windows Runtime geschieht deshalb auch mit C++ (genauer gesagt mit der Erweiterung der Sprache C++/CX). Für Sprachen wie C# oder Visual Basic ist der Zugriff über bestimmte Klassen (*Wrapper*[3]-Klassen) ebenfalls möglich – allerdings müssen dann Übersetzungen von der .NET-Technologie (dem verwalteten Code) in die Windows Runtime erfolgen. Ebenso gibt es die Möglichkeit, dass Windows Store Apps mit Javascript und HTML entwickelt werden. Microsoft schafft damit eine Grundlage für eine relativ sprachunabhängige App-Entwicklung. Dabei gilt es zu berücksichtigen, dass jede Sprache spezielle Vorteile hat (C++ die Schnelligkeit, C# den einfacheren Aufbau und Javascript/HTML5 die große Entwicklergemeinde im Internet). Die folgende Grafik soll das Zusammenspiel der verschiedenen APIs und Sprachen noch einmal darstellen.

1 Dazu müssen das Betriebssystem Windows 8 (oder 8.1) sowie das Visual Studio für Windows 8 installiert sein.
2 COM (**C**omponent **O**bject **M**odel) ist eine Technik von Microsoft, mit der Softwarekomponenten erstellt werden können. Dabei ist die Wahl der Programmiersprache unerheblich.
3 Wrapper-Klassen „umhüllen" andere Klassen und bieten damit die Möglichkeit bestimmte Funktionalitäten auf ein anderes System zu übertragen.



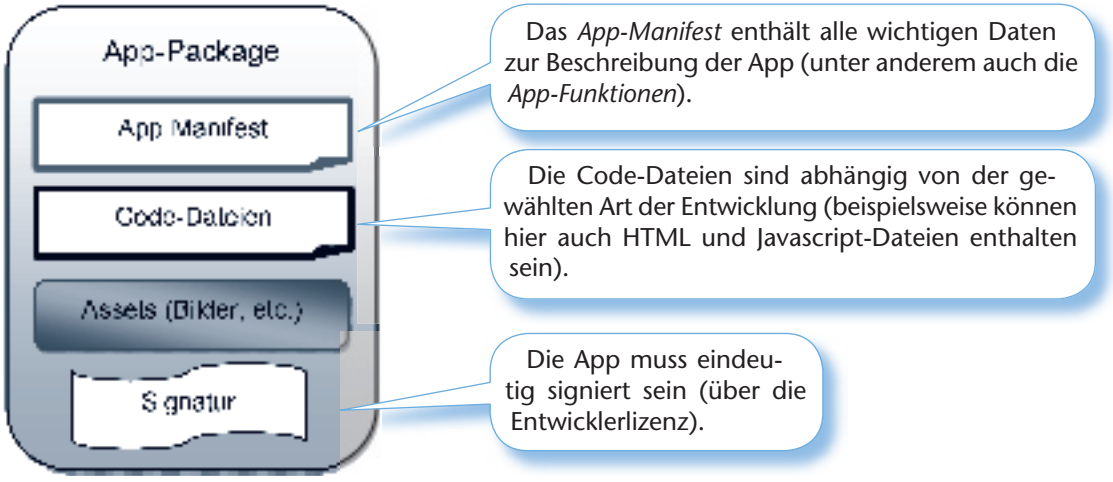
In der Übersicht ist erkennbar, dass die Auszeichnungssprache ***XAML*** (e**X**tensible **A**pplication **M**arkup **L**anguage), die mit dem Grafik-Framework WPF (**W**indows **P**resentation **F**oundation) eingeführt wurde, auch in der App-Entwicklung eine entscheidende Rolle spielt. Sowohl unter C# als auch unter Visual Basic (oder auch C++/CX) werden die Oberflächen mit dieser Sprache beschrieben.

18.1.3 Aufbau einer Windows Store App

Eine Windows Store App kann nur über den Windows Store bezogen, heruntergeladen und aktualisiert werden. Das ist natürlich eine deutliche Einschränkung für den Benutzer, denn damit bindet er sich an genau einen Anbieter (allerdings ist es bei Apple auch nicht anders). Der Vorteil dieser Vorgehensweise ist Sicherheit – die Store Apps werden ausführlich geprüft (automatisch und manuell) und durch die App-Funktionen (App-Capabilities) werden die Zugriffe auf Systemfunktionen (Dateizugriff, Internetzugriff etc.) geregelt. Beispielsweise wird eine App vom System blockiert, wenn sie versucht eine Verbindung ins Internet aufzunehmen, ohne das Recht für diese Funktionalität zu haben.

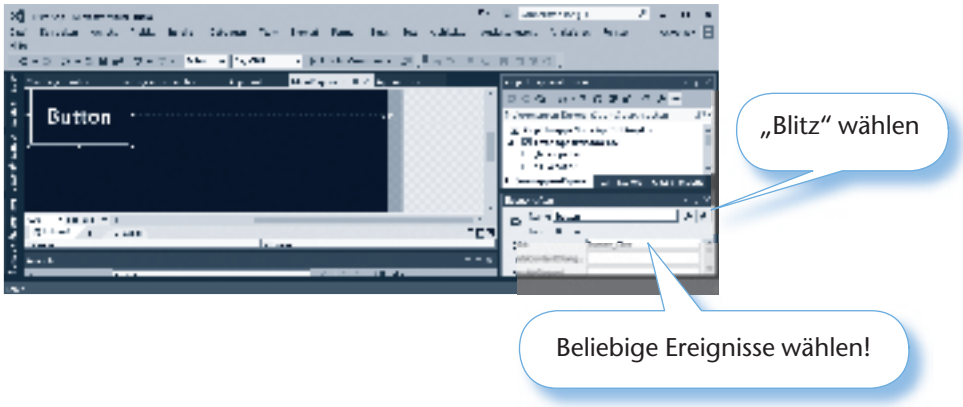
Mithilfe der Entwicklungsumgebung *Visual Studio für Windows (ab Version 2012)* werden Apps programmiert und natürlich auch lokal gestartet. Dazu muss eine kostenfreie Entwicklerlizenz von Microsoft beantragt werden. Das geschieht einfach über ein Microsoft-Konto und wird bei der Installation von Visual Studio durchgeführt. Eine Store App wird in einem sogenannten ***App-Container*** gestartet. Das ist vergleichbar mit einer Sandbox[4].

Im Einzelnen besteht eine Store App aus einem Paket von Dateien (App-Package), wie die folgende Übersicht zeigt:



4 Eine Sandbox (engl. Sandkasten) ist eine Umgebung, in der eine Software sicher ausgeführt werden kann, ohne nachhaltigen Schaden anrichten zu können. Dazu simuliert die Sandbox ein Betriebssystem, sodass die Software so arbeiten kann, als würde sie direkt mit dem System kommunizieren.

Weitere Ereignisse können einfach über die Eigenschaften des Steuerelements mit einer entsprechenden Methode versehen werden:

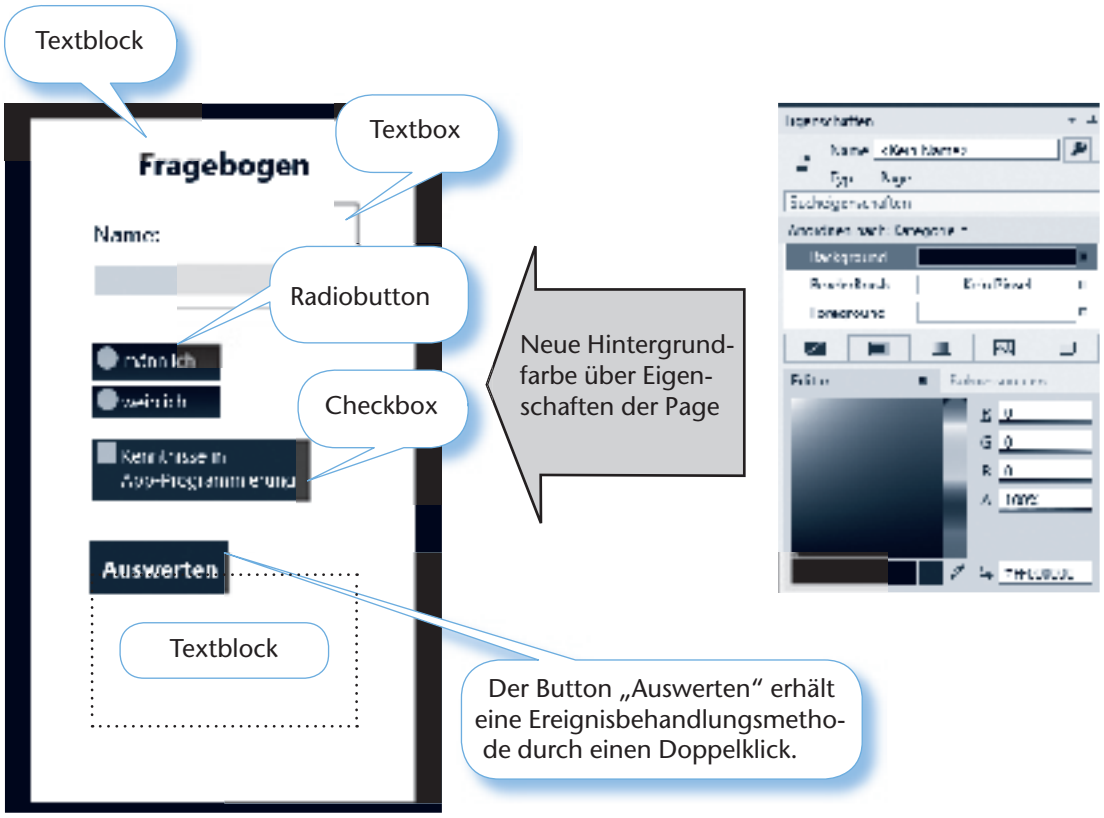


18.3 Einfache Steuerelemente

18.3.1 Textboxen, Buttons, RadioButtons und Checkboxen

Fast jede Windows-Anwendung verfügt über Textboxen, Buttons, Radiobuttons oder Checkboxen. Mit Buttons kann der Benutzer eine bestimmte Aktion wählen, die natürlich mithilfe einer Ereignisbehandlungsmethode hinterlegt wird. Mit den Radiobuttons ist hingegen eine differenzierte Auswahl zwischen mehreren Optionen möglich. Dabei ist besonders wichtig, dass immer nur eine Auswahl zugelassen wird. Deshalb werden diese Steuerelemente auch Radiobuttons genannt – so wie bei einem Radio (jedenfalls früher) kann immer nur ein Knopf (beispielsweise für den Radiosender) gedrückt sein. Soll eine Auswahl von mehreren Optionen möglich sein, so muss die Checkbox zum Einsatz kommen. Mit einem Textblock kann ein beliebiger Text auf der Oberfläche angezeigt werden. Im Unterschied zu einer Textbox kann der Text aber nicht vom Benutzer bearbeitet werden.

Das folgende Beispiel zeigt eine App mit einigen dieser Steuerelementen und einer Ereignisbehandlungsmethode:



Nach dem Starten der Anwendung wird eine Steckbriefdatei angegeben. Das Fenster zeigt dann den Steckbrief in folgender Form an:

Wenn der Benutzer die Größe des Fensters ändert, so wird auch der Inhalt entsprechend angepasst:

Hinweise:

Legen Sie im Konstruktor eine Fenstergröße fest (beispielsweise 400 * 300). Für diese Größe legen Sie dann die Bildschirmausgaben fest (mit einem Skalierungsfaktor von 1). Bei einer Größenänderung ändert sich dann auch der Skalierungsfaktor (in Relation) und die Ausgabe passt sich automatisch der neuen Fenstergröße an.

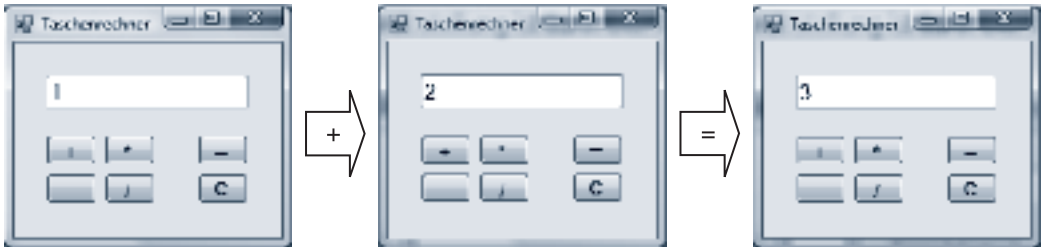
Für die Methode `DrawString()` kann eine neue Schriftart eingesetzt werden. Dazu muss nur vorher ein `Font`-Objekt instanziiert werden:

```
Font schrift = new Font ("Times New Roman", 16 ,FontStyle.Bold);
```

13 Aufgaben zum Designer und einfachen Steuerelementen

Aufgabe 13.1

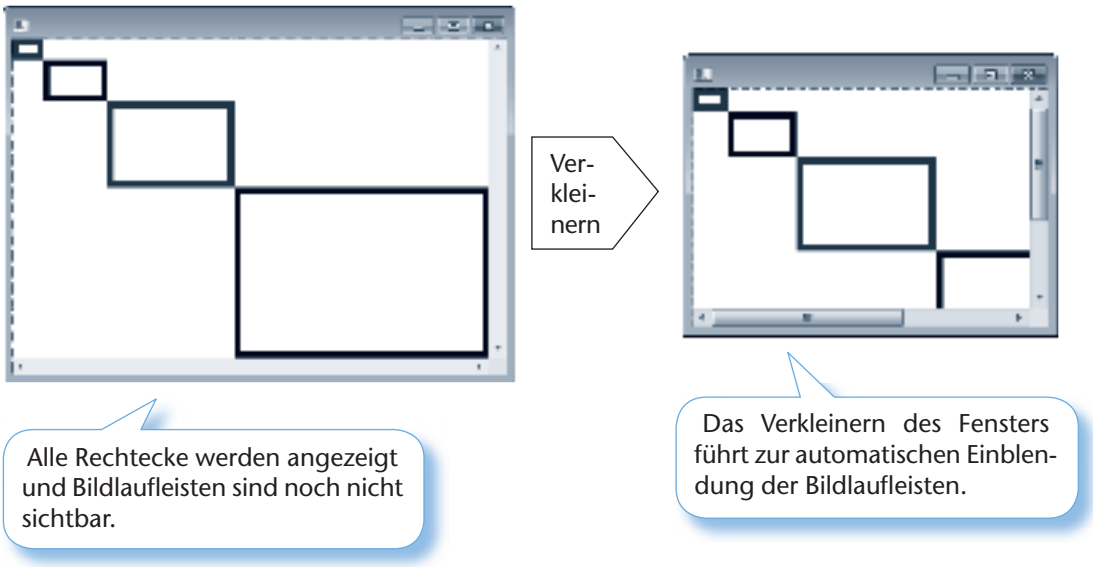
Schreiben Sie ein Windows-Forms-Programm, das einen einfachen Taschenrechner realisiert. Der Taschenrechner soll die Grundrechenarten anbieten und einen zusätzlichen Button („C"-Button), um die Anzeige zu löschen. Der Taschenrechner könnte so aussehen:



Aufgabe 15.5

Nutzen Sie die Möglichkeiten des `Canvas`-Containers und lassen Sie vier Rechtecke zeichnen, abwechselnd in der Farbe blau oder der Farbe schwarz. Die Rechtecke haben immer die doppelte Größe ihres Vorgängers. Ihr Anfangspunkt richtet sich immer nach dem rechten unteren Eckpunkt des Vorgängers. Zusätzlich sollen Bildlaufleisten dafür sorgen, dass auch bei verkleinertem Fenster alle Rechtecke angezeigt werden können.

Die fertige Anwendung könnte dann so aussehen:



16 Aufgaben zu XAML und dem WPF-Designer

Aufgabe 16.1

Für die Kundenberater der *Bank 42* soll eine Anwendung entwickelt werden, mit der ein Ratenkredit einfach ausgerechnet werden kann. Mithilfe von Schiebereglern (`slider`) sollen die Eckdaten des Kredites bestimmt werden. In der unten angeordneten `ListBox` werden diese Daten parallel angezeigt. Die monatliche Rate wird ebenfalls sofort angezeigt.

Es gelten folgende Intervalle:

Kreditsumme: 1.000 – 50.000 (Euro)

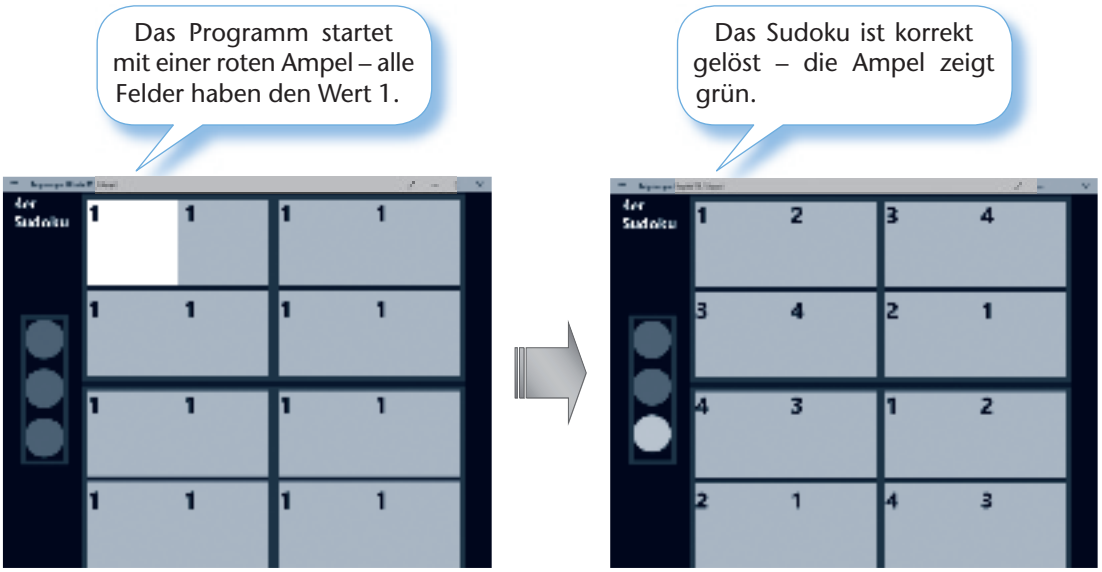
Zinssatz: 1 – 10 (Prozent)

Laufzeit: 1 – 120 (Monate)

Automatische Berechnung!

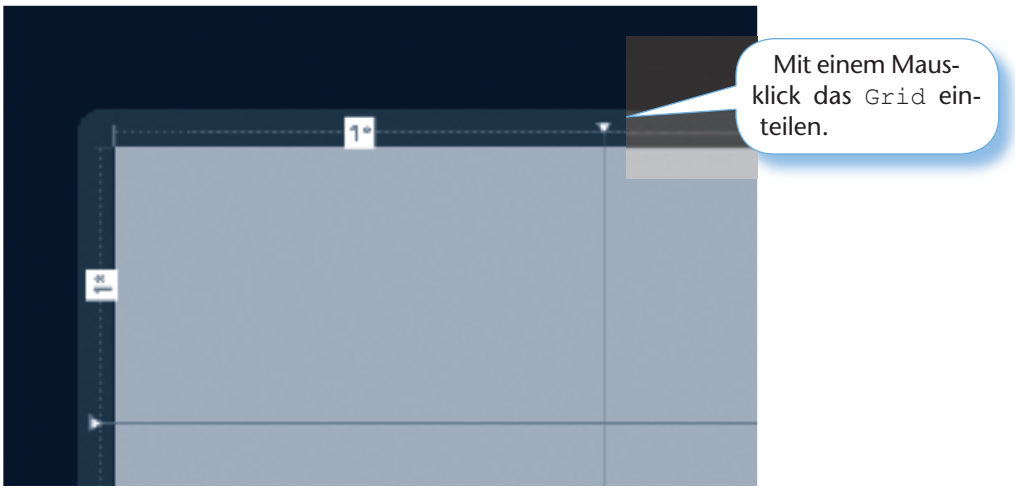
Hinweise:

- Über die Eigenschaften `Minimum` und `Maximum` des Schiebereglers können die Intervalle im XAML-Code festgelegt werden. Mit der Eigenschaft `TickFrequency` kann die Schrittweite des Reglers eingestellt werden. Das „Einrasten" des Reglers wird durch `IsSnapToTickEnabled="True"` erzeugt.
- Dem Ereignis `ValueChanged` kann eine Methode zugewiesen werden, die auf die Veränderung des Reglers reagiert und die entsprechende Berechnung durchführt.



Hinweise:

Nutzen Sie den Layout-Container `Grid` und definieren Sie die entsprechenden Zeilen und Spalten. Das kann entweder über die Entwicklungsumgebung geschehen:



Oder einfach über XAML:

```
<Grid >
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
```

Das Sternchen sorgt für eine automatische Gleichverteilung des Platzes.