sgd
DEUTSCHLANDS
FÜHRENDE FERNSCHULE

BEGLEITHEFT

BGF421110

Fachinformatiker/in – Weiterbildung zum Schwerpunkt Anwendungsentwicklung

BGF421110

Fachinformatiker/in – Weiterbildung zum Schwerpunkt Anwendungsentwicklung

Werden Personenbezeichnungen aus Gründen der besseren Lesbarkeit nur in der männlichen oder weiblichen Form verwendet, so schließt dies ausdrücklich alle anderen Geschlechtsidentitäten ein.

Falls wir in unseren Studienheften auf Seiten im Internet verweisen, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf die zukünftige Gestaltung und den Inhalt der Seiten haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

Fachinformatiker/in – Weiterbildung zum Schwerpunkt Anwendungsentwicklung

Inhaltsverzeichnis

Vorwort

Sehr geehrte Lehrgangsteilnehmerin,
sehr geehrter Lehrgangsteilnehmer,

Sie haben Ihren Lehrgang begonnen und damit einen wichtigen Schritt getan, um Ihr Ziel, Fachinformatiker mit Schwerpunkt Anwendungsentwicklung zu werden, zu erreichen.

Wir werden Sie auf Ihrem Weg in den verschiedenen Lernsituationen mit Rat und Tat unterstützen.

Dieses Heft soll Sie in den nächsten 16 Monaten begleiten. Ob Sie nun den Lehrgang aus beruflichen Gründen oder auch aus privatem Interesse belegen, Sie werden von den erworbenen Kenntnissen und Fähigkeiten auf jeden Fall profitieren.

Damit Sie Ihr Fernstudium möglichst zielstrebig und erfolgreich gestalten können, haben wir Ihnen in diesem Heft alles in Kürze zusammengestellt, was Sie zu Beginn wissen sollten.

Viel Spaß und Erfolg wünscht Ihnen

das Team Ihrer Studiengemeinschaft.

1 Lehrgangsdaten im Überblick

Studienziel	Es ist Ihre Aufgabe als Fachinformatiker, firmenspezifische Anforderungen in individuelle Softwarelösungen umzusetzen. Dabei sind Sie tätig im Bereich Planung, Entwicklung, Anpassung und Pflege von Software sowie Datenbanken. Primäres Lehrgangsziel ist es, Sie zu befähigen, diesen Anforderungen gerecht zu werden. Daher erhalten Sie eine betriebswirtschaftliche und eine IT-technische Fortbildung. Sie lernen von Grund auf, anspruchsvolle Individual-Software zu entwickeln. Auch werden Sie vertraut im Umgang mit Datenbanken, Vernetzung und Datennetzen.
Voraussetzungen	Sie benötigen Erfahrung im Umgang mit dem PC, Windows-Betriebssystemen und den gängigen Microsoft Office-Programmen. Darüber hinaus sollten kaufmännische Vorkenntnisse aus einer Berufsausbildung und mehrjährige, einschlägige Berufserfahrung vorhanden sein.
Technische Voraussetzungen	Sie benötigen folgende technische Ausstattung: • einen Standard-Multimedia-PC mit Windows 10 (64 Bit) und 8 GB RAM sowie Internetzugang, • das Programm Microsoft Access 2019.
Lernthemen	• Arbeiten mit Visual Studio Community 2019 • Objektorientierte Programmierung mit C# • Grafikprogrammierung • Datenbankprogrammierung • TreeView und XML • Betriebswirtschaftliche Grundlagen • Produktions- und Kostentheorie • Betriebliche Funktionsbereiche • Organisation – Grundlagen • Moderne Organisationsformen • Einführung in die Organisationsentwicklung • Datenbanktheorie • Projektmanagement • Präsentation und Rhetorik • Qualitätsmanagement • Vernetzung und Multimedia

Beginn und Dauer	Sie können jederzeit mit dem Lehrgang beginnen. Sie allein entscheiden auch, ob Sie den Lehrgang in der vorgesehenen Studiendauer bzw. schneller oder langsamer bearbeiten möchten. Die Lehrgangsdauer beträgt 16 Monate bei einer wöchentlichen Arbeitszeit von ca. 10–12 Stunden. Sie können sich aber auch mehr Zeit lassen, denn Ihre Betreuungszeit beträgt 24 Monate. Während der gesamten Zeit haben Sie Anspruch auf die Betreuungsleistungen der sgd ohne Mehrkosten.
Studienmaterial/ Online-Angebot	• Studienmappe mit Arbeitsmaterial und wichtigen Infos • 26 Studienhefte, Übungsdateien, ein zehnteiliges Online-Lernprogramm • Zugang zum sgd-OnlineCampus (Lösungseinsendung, Studienmaterial zum Download, Infos, Diskussionsforen, Chats, internes Mailsystem)
Zeugnis	Sie erhalten nach erfolgreicher Lehrgangsteilnahme und als Bestätigung Ihrer Leistungen das sgd-Abschlusszeugnis. Bewertungsgrundlage ist der sog. IHK-Notenschlüssel, es wird auf ganze Noten gerundet. IHK-Notenschlüssel: % von bis Note 92 100 1 81 < 92 2 67 < 81 3 50 < 67 4 30 < 50 5 0 < 30 6
Staatliche Zulassung	Der Lehrgang wurde von der Staatlichen Zentralstelle für Fernunterricht (ZFU) unter der Nummer 7158506 geprüft und zugelassen. Das besagt, dass der Lehrgang vollständig und fachlich einwandfrei ist.

Änderungen vorbehalten

2 Ziel und Aufbau des Lehrgangs

Mit dem Lehrgang „Fachinformatiker mit Schwerpunkt Anwendungsentwicklung“ haben Sie eine interessante und auch anspruchsvolle Wahl getroffen. Der Strukturwandel hin zum Einsatz moderner Informationstechnologien in Handel, Industrie und Dienstleistung ist noch längst nicht abgeschlossen. Speziell die Nutzung des Internets und die damit veränderten Möglichkeiten in Geschäftsbeziehungen führen nach wie vor zu einer Nachfrage nach IT-Fachkräften.

Während einerseits hochspezialisierte Mitarbeiter gesucht werden, besteht andererseits auch ein Bedarf an Mitarbeitern, die zwar in IT-Bereichen arbeiten und über aktuelle IT-technische Entwicklungen Bescheid wissen, aber auch genügend Hintergrundkenntnisse zu betrieblichen Abläufen und kaufmännischen Zusammenhängen haben, also die Anwenderseite kennen.

So sind es Fachinformatiker und Fachinformatikerinnen, die fachspezifische Anforderungen in komplexe Hard- und Softwaresysteme umsetzen. Sie analysieren, planen und realisieren informations- und telekommunikationstechnische Systeme und führen auch neue oder modifizierte Systeme der Informations- und Telekommunikationstechnik ein. Kunden und Benutzern stehen sie für die fachliche Beratung, Betreuung und Schulung zur Verfügung.

Wenn Sie aus dem kaufmännischen, verwaltenden oder organisatorischen Bereich kommen und damit bereits über Erfahrungen von der Anwenderseite verfügen und wenn Ihnen die Arbeit am PC schon jetzt viel Spaß macht, dann erschließen Sie sich mit diesem Kurs neue Aufgabengebiete.

Ebenso passend kann der Lehrgang für Sie sein, wenn Sie bereits in der Datenverarbeitung tätig sind und dort praktische Berufserfahrung besitzen, jedoch bisher keine fundierte Weiterbildung nachweisen können. Oder aber auch, wenn Sie zu den Neu- und Wiedereinsteigern zählen, die an IT-Technologie interessiert sind und in einem modernen Beruf tätig werden wollen.

Mit diesem Lehrgang können Sie einerseits einfach Ihre fachlichen Kompetenzen verbessern, andererseits auch befähigt werden, sich auf dem Arbeitsmarkt oder in Ihrem Unternehmen mit aktuell notwendigem Know-how zu präsentieren und neue Aufgaben übernehmen zu können.

3 Lernstoffübersicht

Arbeiten mit Visual Studio Community 2019

- Variablen, Regeln und Regelverstöße
- Steuerstrukturen, Schleifen
- Konsolenanwendungen, Benutzereingaben
- Ausdrücke: Operatoren und Operationen
- Variablen deklarieren
- Datentypen
- Arrays, Algorithmen
- Sub-Prozeduren und Funktionen
- Enumeration, Strukturen und Klassen

Objektorientierte Programmierung

- Grundbegriffe: Objekte, Instanzen, Felder, Eigenschaften
- Ereignisse
- Klassen und Klassenhierarchien
- Abstrakte Klassen und Methoden
- Grundlagen der UML
- Fehlerbehandlung
- Formatierung
- String-Verarbeitung
- Operationen mit String-Objekten

Projekt

- Erstellung einer Textverarbeitung
- Symbolleiste, Kontextmenü, Info
- Dateien, Dateisysteme und Streams
- Binäre Dateien lesen und schreiben

Grafikprogrammierung

- Geometrische Figuren
- Gestaltung

Datenbankprogrammierung

- SQL-Server
- Datensätze: einfügen, anzeigen, abfragen
- Datenbankentwurf

- Beziehungen
- Datenbankanwendung mit Assistenten erstellen
- Datenquelle bereitstellen
- DataSet, TableAdapter und BindingSource
- Datenbindung
- ADO.NET
- Datenbankanwendung selbst codieren

TreeView und XML

- TreeView-Steuerelement kennenlernen
- Laufwerke und Verzeichnisse im TreeView
- Aufbau von XML-Dateien
- Lesen und Schreiben mit XML-Dateien
- XML im TreeView

Betriebswirtschaftliche Grundlagen

- Grundlagen der Wirtschaft
- Gegenstand, Gliederung, Methoden und Entwicklung der Betriebswirtschaftslehre
- Wirtschaftsordnungen, Wirtschaftssysteme
- Betrieb und Unternehmung
- Rechtsformen der Unternehmung
- Unternehmenszusammenschlüsse
- Betrieblicher Standort

Produktions- und Kostentheorie

- Produktionstheorie
- Kostentheorie, Kostenrechnung
- Betriebswirtschaftliche Kennzahlen
- Investition
- Finanzierung

Betriebliche Funktionsbereiche

- Unternehmensführung
- Produktionswirtschaft (Fertigung)
- Materialwirtschaft
- Absatzwirtschaft (Marketing)
- Personalwirtschaft

Organisation – Grundlagen

- Grundlagen der Organisationsgestaltung und Organisationskonzeptionen

Moderne Organisationsformen

- Von der Struktur- zur Prozessorientierung, Grundkonzepte
- Neue Organisationsansätze
- Formen der Gruppenarbeit
- Information und Kommunikation in dezentralen und flexiblen Strukturen

Einführung in die Organisationsentwicklung

- Methoden der Organisationsentwicklung
- Die Entwicklung zur lernenden Organisation

Datenbanktheorie

- Datenbanken
- Relationale Datenbank
- Das ER-Modell
- SQL – Datenbanken im Internet
- Datenbank-Managementsysteme
- Access – Übersicht, Tabellen erstellen, Abfragen, Formulare, Berichte
- VBA-Editor und Entwicklungsumgebung
- Die strukturierte Programmierung
- Kontrollstrukturen, Zugriffsregelung, Parameterübergabe und Datenfelder
- Error Handling (Fehlersuche und Fehlerbehebung)
- Einführung in die Access-Programmierung
- Einführung in das Objektmodell
- Datenzugriff mit VBA
- Einführung in SQL
- Einführung in ADO – Konzeption, Datenmodell

Projektmanagement

- Begriffe und Grundlagen
- Projektorganisation
- Projektplanung
- Projektkontrolle und -steuerung

Präsentation und Rhetorik

- Zur Wirkung von optischer Rhetorik und visueller Kommunikation
- Präsentationstechniken
- Mind Mapping und Top Mapping

Qualitätsmanagement

- Begriffsbestimmung Qualität
- Bestandteile eines Qualitätsmanagementsystems
- Internationale Systeme und Normen
- Einführung eines QM-Systems
- Instrumente des QM

Vernetzung und Multimedia

- Multimedia: ein integratives Konzept
- Das Internet als Multiplikator für die schnelle Entwicklung multimedialer Technologien
- Multimedia im WWW
- Multimedia, Netzwerk- und Informationstechnologie: E-Business
- Netzwerkarchitekturen
- Fernverkehrsnetze und Datenübertragung
- Arbeiten in und mit Netzen
- Informations- und Managementsysteme: Optimierung von Geschäftsprozessen

Aktualisierungen vorbehalten!

4 Installation der Lehrgangssoftware

In diesem Lehrgang werden Sie moderne Software der Firma Microsoft nutzen und auf der sogenannten **.NET-Plattform** (sprich: „dot net“) mit einer **Entwicklungsumgebung** für die Programmiersprache C# arbeiten.

Software unterliegt laufender Aktualisierung bzw. Weiterentwicklung und muss daher zur Kenntlichmachung des jeweiligen Entwicklungsstandes versioniert werden. Die jeweilige **Software-Version** wird durch eine Kennung identifiziert, wobei unterschiedliche Kennzeichnungssysteme zur Anwendung kommen. So ist es auch bei Microsofts Programmierumgebung: Die zugrundeliegende Software-Plattform („.NET“) verwendet das traditionelle System numerischer Versionsbezeichnungen, wobei die Hauptversion vor und die Unterversion hinter einem Punkt angegeben werden (z.B. „.NET 4.7“). Die auf dieser Plattform aufsitzende Entwicklungsumgebung verwendet hingegen in ihren Produktnamen Jahreszahlen (z.B. „Visual Studio 2019“). Tab. 4.1 stellt die zusammengehörigen Versionen seit Platzierung von .NET im Markt einander gegenüber:

Tab. 4.1: Zuordnung von .NET- und Visual Studio-Versionen

.NET-Versionen	Visual Studio	
	Produktversionen	interne Versionen
1.0 und 1.1	.NET [2002, 2003]	7.0, 7.1
2.0	2005	8.0
3.0 und 3.5	2008	9.0
4.0	2010	10.0
4.5	2012	11.0
4.6	2015	14.0
4.6	2017	15.0
4.7	2019	16.0

Das Produkt „Visual Studio“ ist eine Entwicklungsumgebung, weil es die Nutzung der .NET-Plattform in der Umgebung von grafischen Benutzeroberflächen bei Unterstützung durch zahlreiche Software-Werkzeuge für den Software-Entwickler komfortabel erleichtert. Weil sie Softwarebibliotheken, Tools und hilfreiche Oberflächen integriert, bezeichnet man ein solches Produkt auch mit der englischen Abkürzung **IDE** (Integrated Development Environment / Integrierte Entwicklungsumgebung). Microsofts IDE „integriert“ zudem noch mehrere Sprachen, in denen man innerhalb dieser IDE alternativ oder kombinierend programmieren kann. Seit Version 2010 sind dies: Visual Basic, C++, C# sowie die funktionale Sprache F#.

Es stehen mehrere Produktversionen in den Versionen Professional, Enterprise und Test-Prof. bereit. Die noch bis Visual Studio 2012 bekannten Versionen Premium, Ultimate, Team System und Academic wurden für Visual Studio 2019 nicht mehr erstellt.

Die Versionen Standard und Express gibt es nicht mehr, da Microsoft die Lizenz für Visual Studio änderte. Mit der Community-Version lassen sich kommerzielle Projekte erstellen. Sie ist aber dabei beschränkt auf Unternehmen von maximal 1 Mio. US$ und 5 Nutzer. Private Anwender, Schüler, Studierende und Bildungseinrichtungen können diese Edition uneingeschränkt nutzen. Weiterhin ist sie für die Verwendung von Open-Source-Projekten kostenlos verfügbar.

Die IDE arbeitet nicht ohne den Unterbau der .NET-Plattform, die als **.NET-Framework** zur Verfügung gestellt wird. Außerdem ist für ein erfolgreiches Programmieren eine gute **Dokumentation** erforderlich, die darüber aufklärt, wie die .NET-Plattform und die C#-IDE zu nutzen sind. Ergänzend beschreibt die Dokumentation die im Framework enthaltene umfangreiche anwendungsorientierte Software, die Sie beim Programmieren nutzen werden.

Schließlich haben die Microsoft-Übersetzungscomputer viel zu tun gehabt, um die Benutzeroberflächen der Software und die zugehörige Dokumentation auch ins Deutsche zu übersetzen (manchmal haben auch menschliche Übersetzer an kritischen Stellen dabei geholfen). Deshalb besteht die zu installierende Software aus mehreren teilweise üppigen Modulen, die sich technisch auf noch mehr Pakete aufteilen:

- dem .NET-Framework 4.7
- einer deutschen Sprachergänzung zum .NET-Framework („language pack“)
- der Entwicklungsumgebung „Visual Studio 2019 Community“
- dem Microsoft Help Viewer 2.3 mit deutschem Sprachpaket zur Präsentation der umfangreichen Hilfe
- ferner einige Server-Komponenten.

4.1 Installationsvoraussetzungen

Die vorgenannten Softwarepakete sind zwar günstig zu beziehen, haben aber ihren Preis in den Anforderungen an die Computerausstattung. Je nach Zusammenstellung der installierbaren Produkte werden, wie es bereits in der Übersicht (Kapitel 1) angesprochen wurde, 5 GB freier **Festplattenspeicher** (oder auch mehr) auf der System-Festplatte benötigt. Wesentlich mehr Festplattenspeicher benötigen Sie jedenfalls, wenn Sie größere Teile der .NET-Dokumentation auf ihrem lokalen Rechner bereitstellen wollen, wofür manches spricht (dazu mehr in Kapitel 6).

Außerdem arbeitete schon bislang die für eine neue Generation von **Betriebssystemen** konzipierte .NET-Plattform nicht mehr mit jedem Microsoft-Betriebssystem der Vergangenheit zusammen. Die Unterstützung von Windows 2000 wurde längst eingestellt. Von .NET 4 wurden noch Windows XP mit SP3, Windows Vista mit SP2 und Windows 7 unterstützt. Nun schränkt Visual Studio 2019 mit .NET 4.7 die zulässigen Betriebssysteme weiter drastisch ein und verlangt mindestens Windows 7. Der **Prozessor** sollte mit 2,4 GHz getaktet sein, besser sind Mehrkernprozessoren. An Arbeitsspeicher sind 4 GB (oder auch mehr) wünschenswert. Auch der **Bildschirm** sollte nicht zu klein ausfallen und jedenfalls 1280 x 1024 Pixel unterstützen, damit die komplexe, aus mehreren Teilfenstern zusammengesetzte Entwicklungsumgebung produktiv genutzt werden kann.

4.2 Installationsbeschreibung

Zur Installation der Entwicklungsumgebung „Visual Studio 2019 Community“ mitsamt .NET-Framework laden Sie sich zuerst die Software von Microsoft herunter:

https://visualstudio.microsoft.com/de/downloads/

Nachdem der Visual Studio Installer installiert wurde, zeigt Ihnen dieser die zur Verfügung stehenden Varianten wie in Abb. 4.1 an:

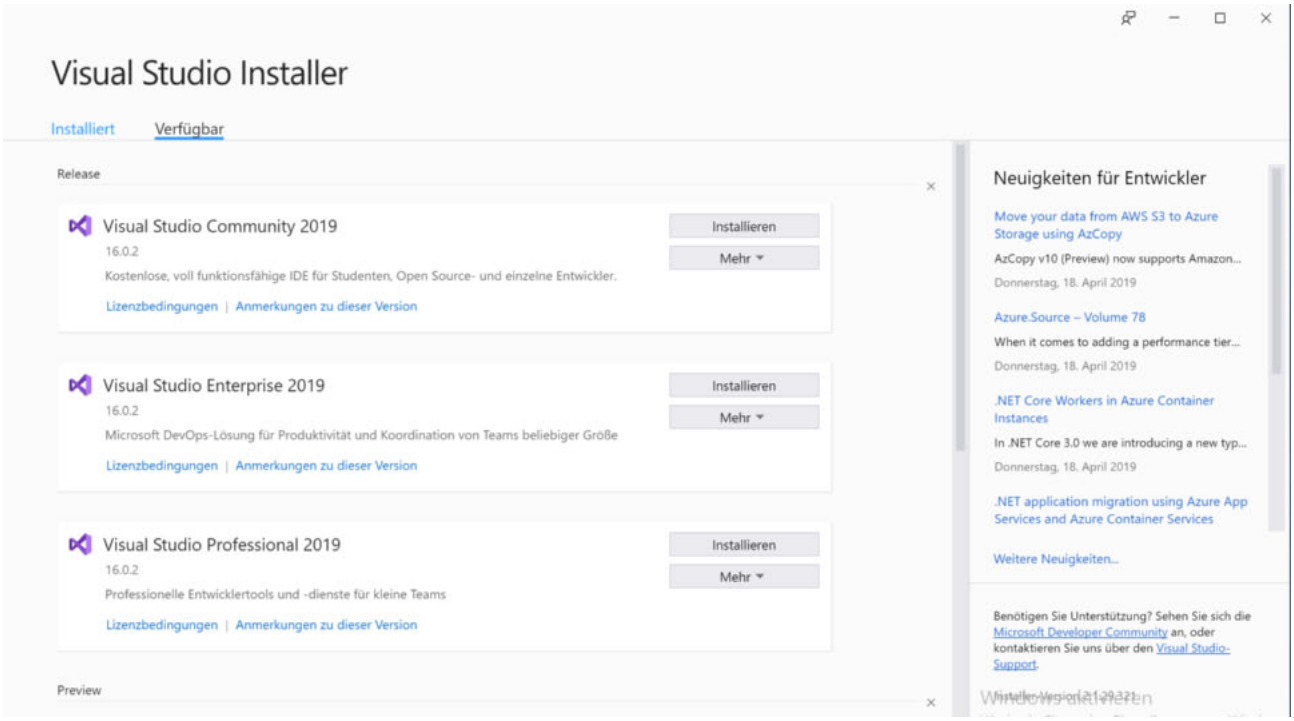


Abb. 4.1: Visual Studio Installer Start

Anschließend erscheint ein Fenster wie Abb. 4.2, in welchem Sie die Möglichkeit haben, zu entscheiden, welche Komponenten installiert werden sollen. Aktivieren Sie zunächst „*.NET-Desktopentwicklung*“.

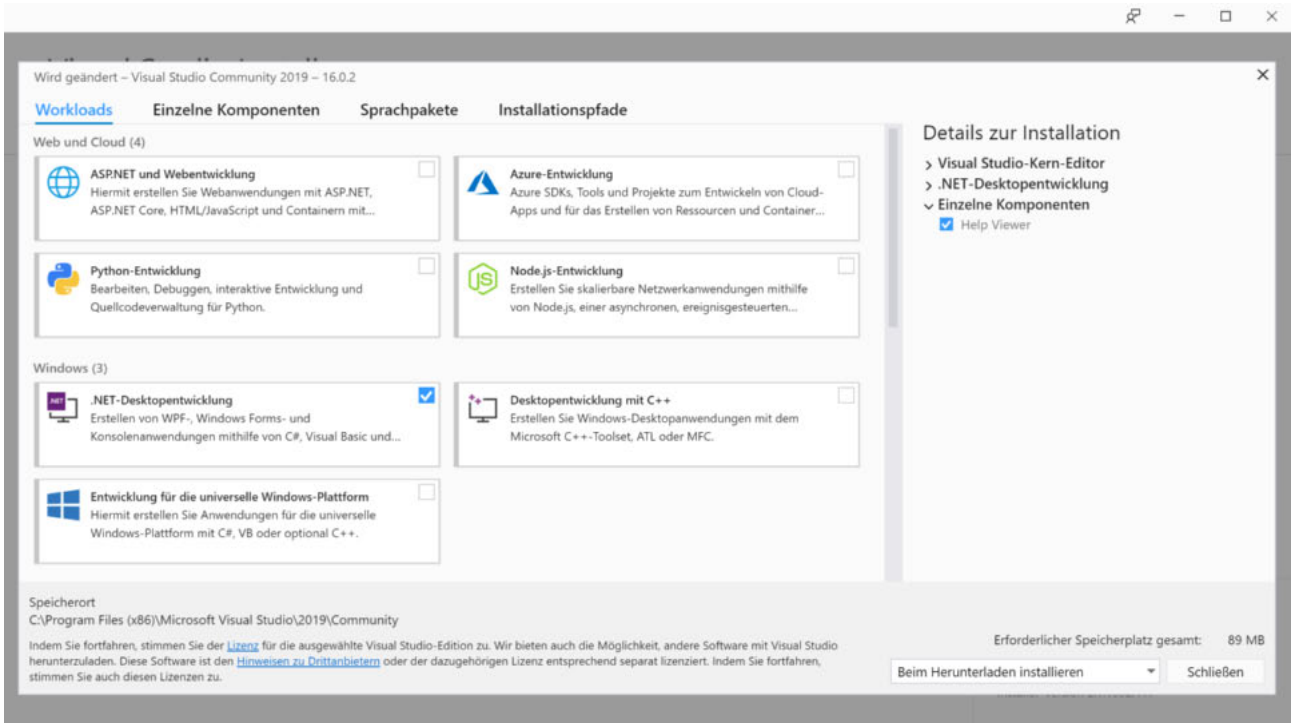


Abb. 4.2: Auswahl der Komponenten für die Installation

Unter „*Einzelne Komponenten*“ können Sie einzeln wählen, welche Komponenten installiert werden sollen. Dies kann auch zusätzlich zu Ihrem gewählten *Workload* geschehen. Weitere Sprachpakete können Sie unter „*Sprachpakete*“ vor der Installation auswählen. Sollten Sie jedoch eine Komponente oder auch ein Sprachpaket vergessen haben, können Sie diese nachträglich installieren. Unter „*Installationspfade*“ besteht die Möglichkeit, den Ort der Installation zu ändern bzw. anzupassen. Anschließend können Sie auf „*Installieren*“ klicken.

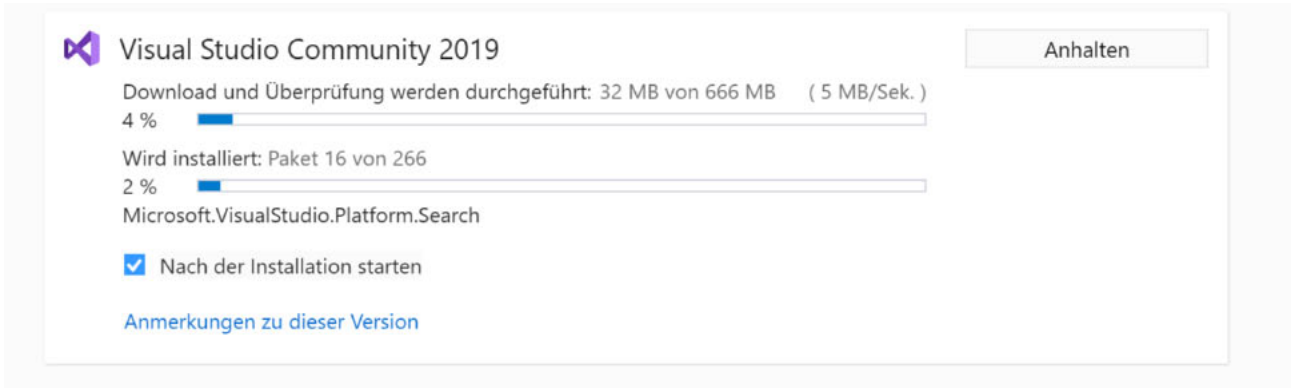


Abb. 4.3: Installationsprozess von Visual Studio Community 2019

Anders als in früheren Visual Studio-Versionen wird im Zuge der Installation auch die MSDN-Dokumentation zu den Community-Produkten der .NET-Plattform nicht mehr angeboten. Diese **Dokumentation** ist keinesfalls weniger wichtig geworden. Microsoft hat jedoch sein Hilfe- und Dokumentationssystem bereits mit Visual Studio 2010 so um-

gestellt, dass die Inhalte nur noch Online zu betrachten bzw. Online auf einen lokalen Rechner zu installieren sind. Wir werden darauf in Kapitel 6 dieses Begleithefts noch detailliert zurückkommen.

Damit das, was sich hinter den Kulissen zugetragen hat, nicht völlig verdeckt bleibt, lohnt ein Blick in die Windows-Einstellungen mit Auswahl der Kategorie „Apps & Features“. Sortieren Sie die Auflistung der auf Ihrem Rechner installierten Software nach *Datum* („Installationsdatum“) und schauen Sie, was sich soeben ereignet hat.

4.3 Erststart der Windows Desktop Express-IDE

Wenn Sie das Feld „Nach der Installation starten“ aktiviert haben (siehe Abb. 4.3), startet Visual Studio nach der Installation automatisch. Andernfalls führen Sie den Erststart der frisch installierten IDE über den im Startmenü hinzugekommenen Eintrag durch.

Die Startmenü-Verknüpfung verweist auf die IDE-Start-Datei *„<Systemlaufwerk>\Program Files(x86)\Microsoft Visual Studio \2019\Community\Common7\IDE\devenv.exe“*. Dieses Element wäre auch auszuwählen, wenn Sie sich eine IDE-Verknüpfung auf den Desktop legen wollen.

Nun folgt alsbald ein Dialog, der zur Registrierung auffordert (Abb. 4.4):

Abb. 4.4: Dialog zur Onlineregistrierung

Die notwendige Registrierung hat Microsoft ab Visual Studio 2010 auch für die Express-Produkte eingeführt; ohne Registrierungsschlüssel kommen Sie nicht über eine 30 Tage-Probefrist hinaus. Der Registrierungsschlüssel wird für den anfragenden Rechner generiert. Die Produktregistrierung kann auch noch später über das IDE-Menü „Hilfe > Produkt registrieren“ nachgeholt werden.

Nutzen Sie (bei aktivem Internetzugang) den Link „Registrieren“ um auf die Microsoft-Registrierungsseite zu gelangen. Microsoft fordert nun allerlei persönliche Informationen, ehe ein **Product Key** generiert wird, der in das Eingabefeld des Visual Studio-Dialogs einzutragen ist. Ist dabei nichts schiefgegangen, meldet der Dialog „*Ihr Product Key wurde erfolgreich angewendet*“.

Anschließend erscheint noch ein Fenster über die Farbwahl. Wählen Sie dabei ein Farbmuster Ihrer Wahl. Nun müssen Sie nur noch – begleitet von einem Fortschrittsbalken – ein wenig warten, bis die IDE sich initialisiert hat und ihre Oberfläche präsentiert (Abb. 4.5):

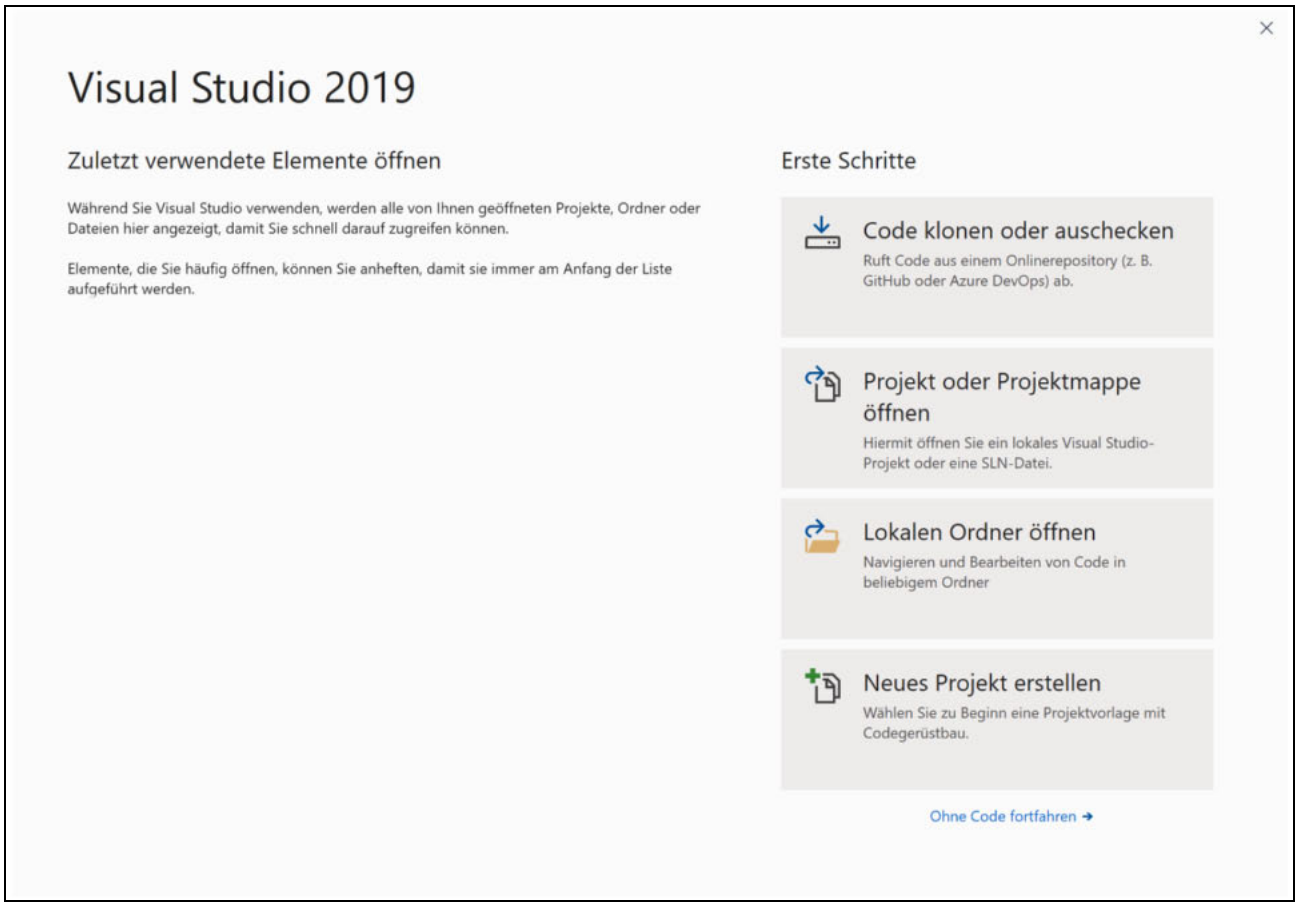


Abb. 4.5: Die Startansicht der Community-IDE

Das Fenster ist intern in zwei Teile gegliedert. Im Bereich links erscheinen später die zuletzt verwendeten Elemente. Auf der rechten Seite haben Sie die Wahl ein Projekt bzw. eine Projektmappe zu öffnen. Weiter können Sie auch bestimmte Ordner öffnen, um dort auf die gewünschten Dateien zuzugreifen oder ein neues Projekt erstellen. Ganz unten besteht die Möglichkeit, auch ohne Code fortzufahren.

Die Studienhefte nehmen dieses Angebot zur **Projektorganisation** auf und ordnen in aller Regel jedem Studienheft eine Projektmappe zu. Die Projektmappen-Untergliederung in Projekte ordnet dann meist einer Lektion ein einzelnes Projekt zu.

Klicken Sie schon mal zum Ausprobieren auf das Startangebot „**Neues Projekt erstellen**“. Es folgt ein Dialog, in dem Sie „Vorlagen“ (Templates) für Visual Studio-Projekte auswählen können. Gegenüber den früheren Visual Studio Express-Versionen, die immer auf eine der .NET-Sprachen begrenzt waren (also entweder auf C# oder Visual Basic oder C++) hat Microsoft seit der Version 2012 diese Sprachenaufteilung aufgegeben und bietet nun alle drei .NET-Hauptsprachen zur Programmierung von Desktop-Anwendungen an. Sie können also zunächst oben unter „Sprachen“ entscheiden, mit welcher Sprache Sie programmieren wollen. Dann können Sie die Auswahl auf eine Plattform begrenzen, für die das neue Projekt erstellt werden soll. Ganz rechts kann der Projekttyp ausgewählt werden – etwa die „Konsole“, die lediglich mit der Windows-Konsole CMD arbeitet (Abb. 4.6). Wir kommen auf diesen Projekteinstieg noch einmal im Abschnitt 6.1 dieses Begleithefts zurück, Sie können jedoch zum Testen einmal eine „Konsolen-App (.NET Framework)“ erstellen, um sich die Grundeinstellungen anzuschauen, die nachfolgend kurz betrachtet werden.

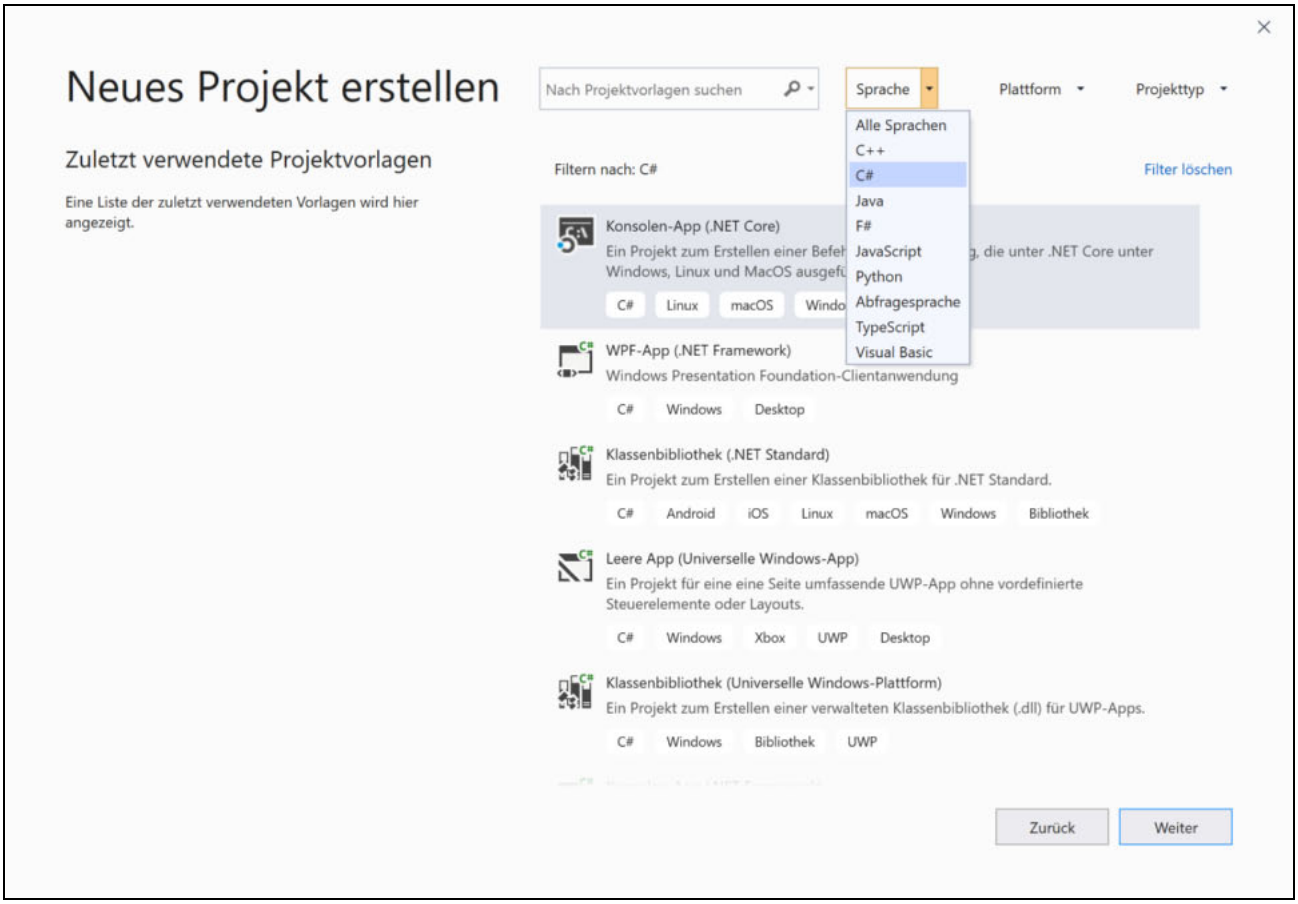


Abb. 4.6: Dialog zur Programmiersprachenwahl und zur Vorlagenauswahl für ein neues Projekt

4.4 Grundeinstellungen

Beim ersten Kontakt mit einer neuen Software sollten Sie einen Blick in die **Konfigurations**möglichkeiten werfen. Das entsprechende Angebot findet sich (wie bei den meisten Windows-Programmen) im Menü „Extras > Optionen…“. Der Dialog organisiert im linken Bereich zahlreiche Einstellkategorien über eine Baumstruktur. Je nach Vorwahl werden dann rechts die Optionen angeboten, wobei das Angebot für einen Haupteintrag (z.B. „Umgebung“) immer identisch ist mit dessen ersten Untereintrag (zu „Umgebung“ wäre das „Allgemein“). Zur „Umgebung“ können Sie z.B. die **Statusleiste** aktivieren/deaktivieren (sonst meist im Ansicht-Menü). In diesem Bereich lässt sich auch über den Klappladen oben das Erscheinungsbild der IDE von „hell“ auf „dunkel“ umstellen. Je nachdem, welche Farbwahl Sie gewählt haben, beispielsweise „hell“, die Alternative „dunkel“ macht daraus nach kurzer heftiger Festplattenaktivität eine schwarze IDE-**Grundfarbe**, wie sie inzwischen für Grafikprogramme Mode geworden ist (z.B. Photoshop Elements oder Microsofts Blend-IDE für die Gestaltung grafischer Benutzeroberflächen). Da auch alle Editoren in diese schwarze Grundfarbe getaucht werden, ist das für Programmcodebearbeitung eher keine gute Wahl, weil die Lesbarkeit des in verschiedene Farben unterschiedenen Codes leidet.

Wichtiger ist hingegen eine für die Orientierung im Programmcode bedeutsame Einstellung: Die Aktivierung von **Zeilennummern**. Für die Lokalisierung von Fehlerquellen oder einfach nur für die Bezeichnung einer bestimmten Stelle im Programmcode ist die Nummerierung der Codezeilen unerlässlich. Die Zeilennummern sind eine informelle Ergänzung der IDE und gehören selbst nicht zum Programmcode, obwohl sie im Editorfenster den Codezeilen vorangestellt werden; sie sind per Voreinstellung nicht eingeblendet. Das können Sie ändern, wenn Sie im linken Teil des Optionen-Dialogs den Eintrag „Text-Editor“ expandieren und den Untereintrag „Alle Sprachen“ auswählen. Zum Haupteintrag (wieder identisch mit der Unterkategorie „Allgemein“) nehmen Sie die Einstellung vor und aktivieren unter den Optionen auf der rechten Seite die Checkbox „Zeilennummern“, wie es Abb. 4.7 durch Umrahmung hervorhebt:

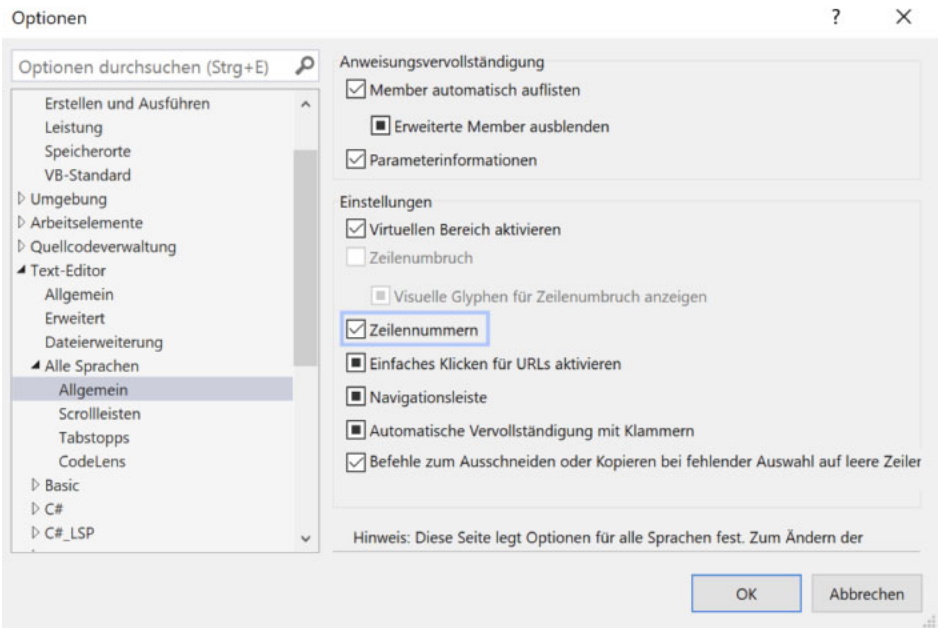


Abb. 4.7: Aktivierung der Zeilennummern in den Editoren über den Dialog „Extras > Optionen“, Kategorie „Text-Editor > Alle Sprachen“

Eine weitere wichtige Voreinstellung könnte die Einstellung der **Schriftgröße** sein, die mit 10 pt per Voreinstellung gegebenenfalls etwas klein ausfällt. Wollen Sie das größer haben, so wählen Sie die Kategorie „Umgebung > Schriftarten und Farben“. Abb. 4.8 zeigt die zugehörige Eingabemaske, in der neben der Schriftgröße für eine Schriftart und deren Verwendungsbereich in der IDE noch allerlei weitere Einstellungen möglich sind – insbesondere die Zuordnung von Farben zur Strukturierung der Syntax (Schlüsselworte blau, Kommentare grün usw.) – allgemein **Syntax-Highlighting** genannt. Bei Auswahl von „Nur Text“ wirkt die hierzu vorgenommene Schriftgrößenwahl auch auf alle anderen Textkategorien.

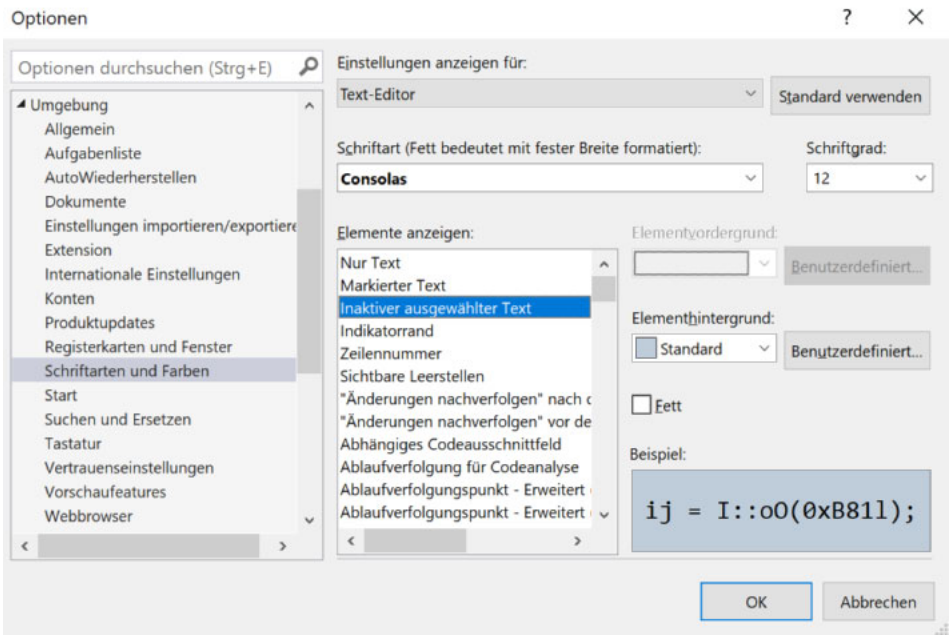


Abb. 4.8: Optionen zu Schriften und ihren Farben

5 Das Microsoft-Hilfesystem

Programmieren ist eine komplexe Angelegenheit – zumal dann, wenn nicht nur die verwendete Sprache zu beherrschen ist, sondern wenn auch noch eine umfangreiche Softwarebibliothek zur Verfügung steht, die beim Programmieren genutzt werden kann. So ist es insbesondere auch auf der .NET-Plattform, die unzählige Typen wie Klassen, Interfaces usw. bereithält, die Sie bei Ihrer Programmierung nutzen können und müssen. Angesichts dieser Vielfalt ist eins klar: ohne eine gute Dokumentation lassen sich die gegebenen Möglichkeiten nicht nutzen. Ein gutes Dokumentations- bzw. Hilfe-System ist somit eine der wichtigsten Grundlagen erfolgreicher Programmierung.

Microsoft hat bereits mit der Version Visual Studio 2010 ein neues **Hilfesystem** eingeführt, das seine Informationen sowohl Online von den Microsoft-Servern als auch lokal auf Ihrem Entwicklungsrechner liefern kann. Dieses Hilfesystem wurde mit der Visual Studio-Version 2012 und 2015 erneut tiefgreifend überarbeitet. Nun scheiden sich die Wege derart, dass die Online-Hilfe von den Microsoft-Servern im Browser dargestellt wird, die lokal installierte Hilfe hingegen in einem Programm namens „Microsoft Help Viewer“, der Visual Studio 2015 (und folgende) in der Version 2.3 beigesellt ist.

Der Help Viewer muss zunächst als Komponente jedoch zu Visual Studio 2019 hinzugefügt werden. Dazu öffnen Sie den Visual Studio Installer. Unter „Installiert“ finden Sie die aktuell installierten Visual Studio Versionen. Neben jeder Version gibt es eine Auswahl, was Sie mit einer Version tun können. Um neue Komponenten zu installieren, klicken Sie einfach auf „Ändern“ wie in Abb. 5.1 zu sehen ist.

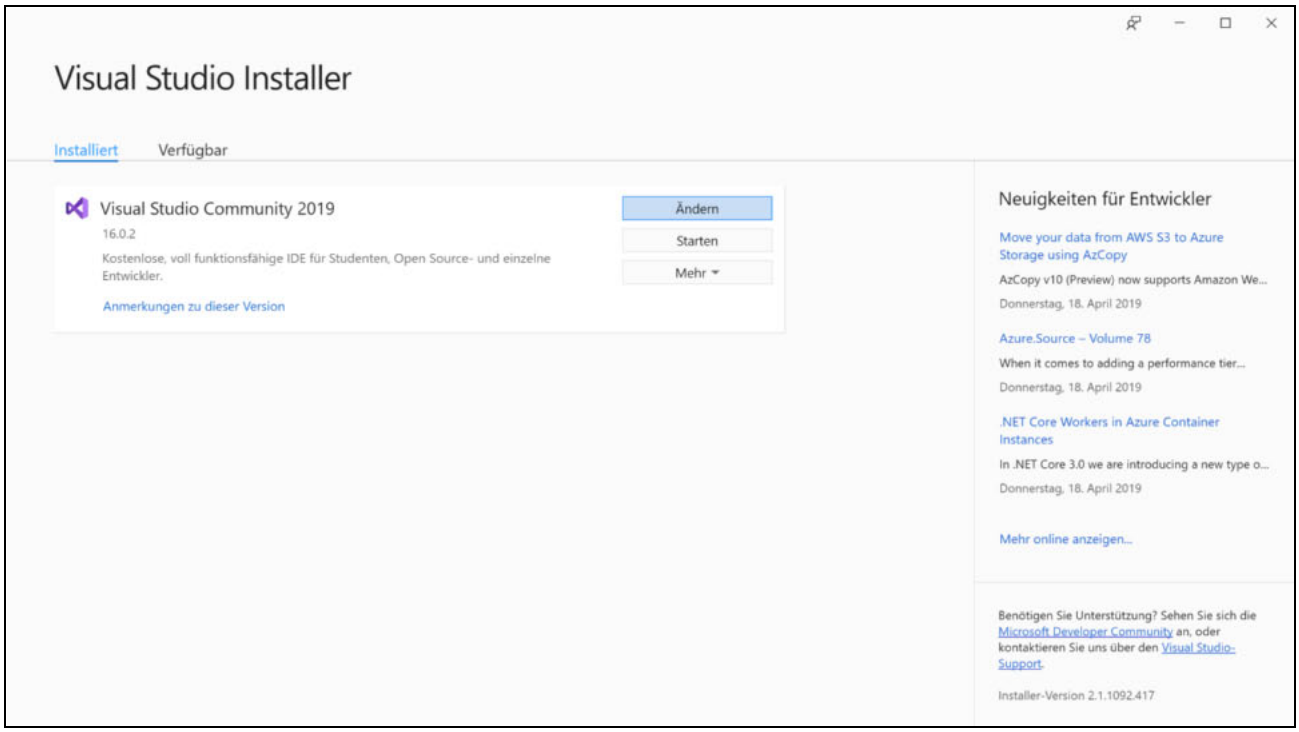


Abb. 5.1: Übersicht der installierten Visual Studio Versionen (in unserem Fall ist natürlich nur eine Version installiert)

Es erscheint nun das Fenster, welches schon bei der ersten Installation (Abb. 4.2) zu sehen war. Unter „Einzelne Komponenten“ und „Codetools“ aktivieren Sie nun die Checkbox „Help Viewer“ und starten die Installation über Betätigen des Buttons „Ändern“ rechts unten (Abb. 5.2).

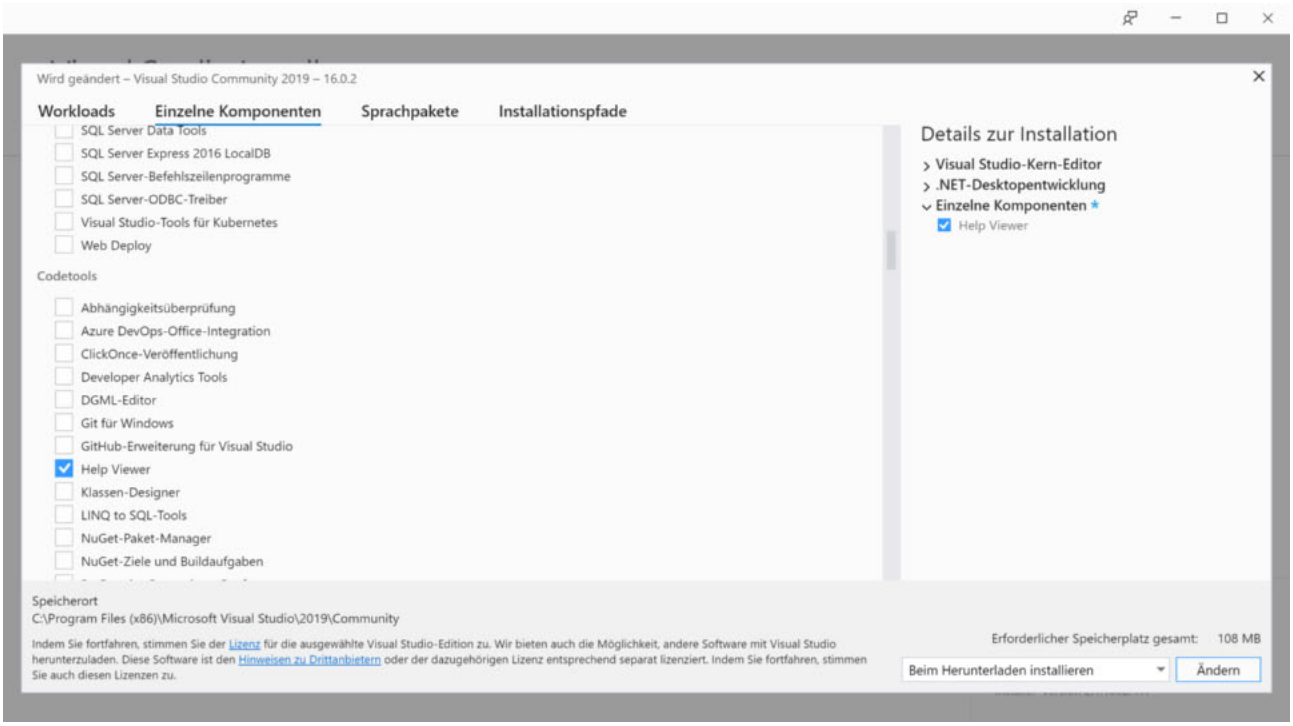


Abb. 5.2: Help Viewer hinzufügen

Um diesen jetzt nutzen zu können, müssen Sie die entsprechende Vorwahl über das IDE-Menü: *Hilfe > Hilfeeinstellungen festlegen > In Browser starten | In Help Viewer starten* vollziehen. Abb. 5.3 zeigt diese Auswahlfolge mit dem oberen Teil des IDE-Fensters:

Abb. 5.3: Auswahl der Hilfedarstellung (Browser oder Help-Viewer) über das Menü „Hilfe“

5.1 Die Online-Hilfe

Die Anzeige selbst erfolgt dann im gleichen Menü über den ersten Eintrag „Hilfe anzeigen“. Wählen Sie zunächst „in Browser starten“. Dann öffnet sich Ihr Standardbrowser und zeigt die deutschsprachige Visual Studio-**Bibliothek** auf docs.microsoft.com/de-de/visualstudio/.

Hier finden Sie umfangreiche Hilfe-Dokumentationen. Unter „Erste Schritte“ finden Sie Informationen und Hilfe rund um die IDE Visual Studio.

Über das Menü am oberen Rand „Weitere Dokumentationen“ und „…“ findet man schnell weitere Dokumentationen, wie z.B. für das .NET-Framework.

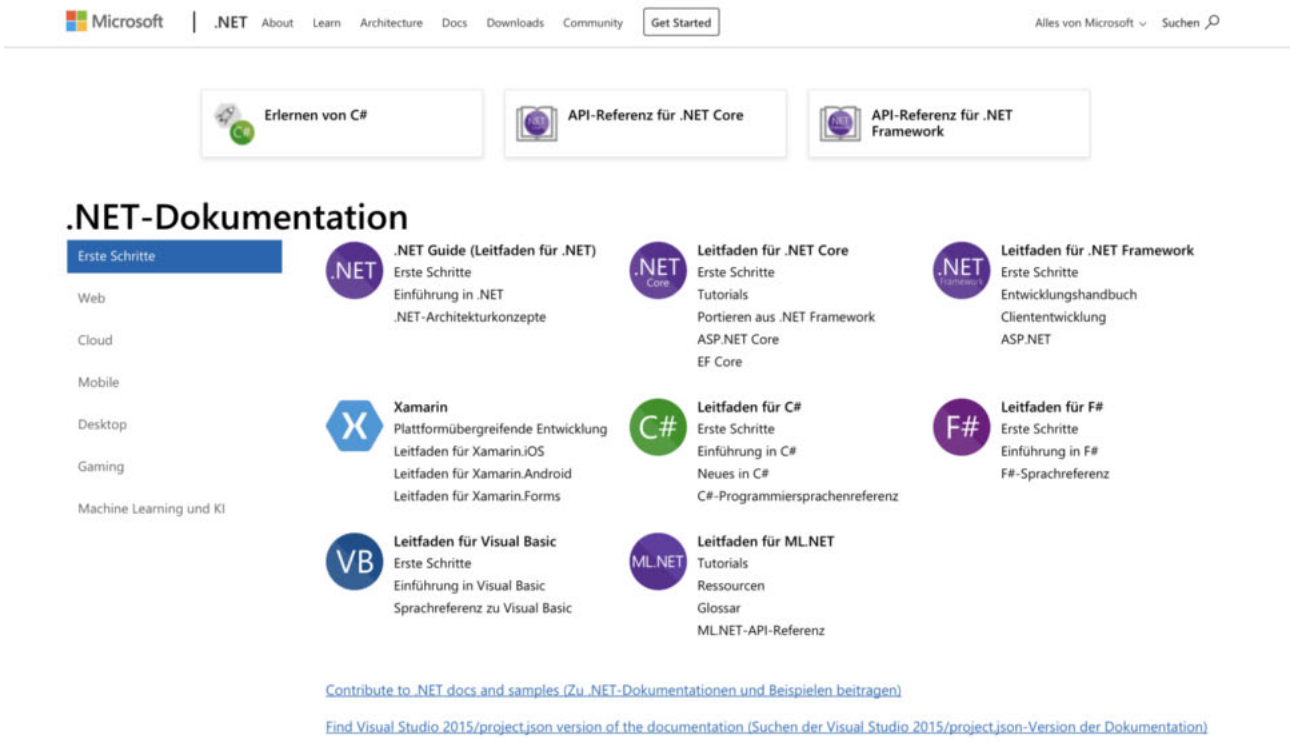


Abb. 5.4: .NET-Dokumentation online

In diesem Lehrgang arbeiten Sie mit .NET 4.7.

Die bereits kurz angesprochene Dokumentation aller .NET-Typen – das sind alle Klassen, Interfaces, Enumerationen, Strukturen und Delegates einer .NET-Version – stellen das wesentlichste Hilfsmittel Ihrer Programmierung dar. Sie erschließen sich in der Online-Hilfe (Browser) diese Dokumentation über diesen Pfad in der Navigationsbaumstruktur:

```
Visual Studio Library
  .NET Framework <Version>
    .NET Framework-Klassenbibliothek
```

Die „Klassenbibliothek“ gliedert sich in sogenannte „**Namensräume**“, in denen mehr oder weniger viele Typen (Klassen, Interfaces usw.) unter fachlichen Aspekten zusammengefasst sind. Die wichtigsten Quellen für Ihre Programmierung liegen im Haupt-

Namensraum „**System**“, der sich in eine riesige Zahl von Unter-Namensräumen differenziert. Man benennt derartige Unternamensräume, indem ein Namensraumbezeichner an den Hauptnamensraumbezeichner getrennt durch einen Punkt angehängt wird, z.B. „System.Text“. Ein weiterer wichtiger Haupt-Namensraum „Microsoft“ gliedert sich ebenfalls in Unter-Namensräume (allerdings nicht so viele) und sammelt Typen, die für die Arbeit der .NET-Plattform selbst verwendet werden. Die Haupt-Namensräume sind in alphabetischer Anordnung in der Dokumentation aufgelistet. Unter dem Haupt-Namensraum „System“ finden Sie zum Beispiel die Klasse „Array“, deren Dokumentationsbeginn Abb. 5.5 zeigt.

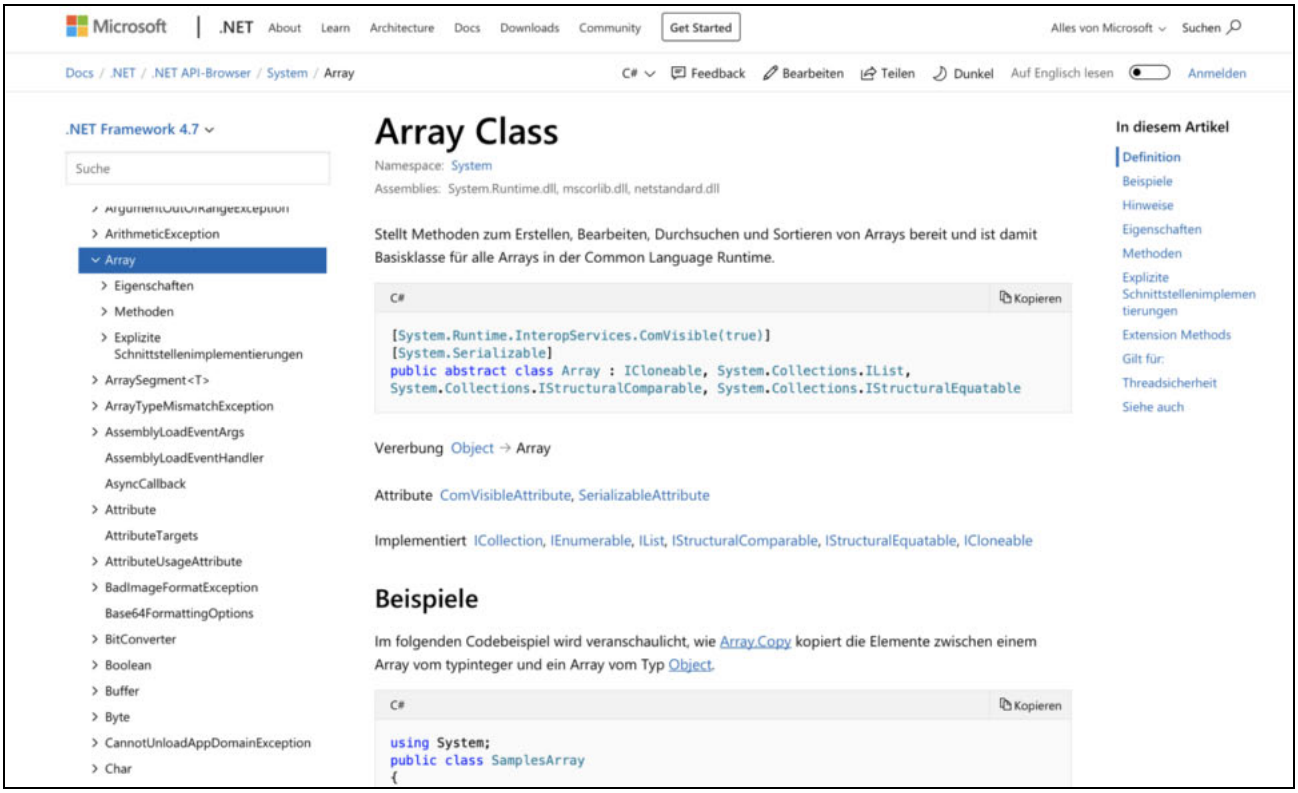


Abb. 5.5: Beginn der Dokumentation zur Klasse „System.Array“ in der .NET 4.7-Bibliothek

Ein Typ (also eine Klasse usw.) wie „Array“ liegt immer in einem Namensraum. Dadurch ist es möglich, den gleichen Typennamen in verschiedenen Namensräumen zu verwenden (wenn sich das als nötig erweist). Deshalb ist der vollständige Name eines Typs immer die Kombination von Namensraumbezeichner und dem eigentlichen **Klassenname**, getrennt durch einen Punkt, also <Namensraum>.<Typenname>, wobei die Ausdrücke mit spitzen Klammern durch konkrete Bezeichner zu ersetzen sind.

Ein Typ (also eine Klasse, ein Interface usw.) enthält Elemente wie Methoden oder Variablen, für die sich der Sammelbegriff „**Members**“ (Mitglieder) eingebürgert hat. Welche Member-Arten ein Typ besitzt, erkennen Sie bereits in der Navigationsstruktur links. Für die Klasse „System.Array“ sind das nur zwei: Methoden und Eigenschaften. Schauen Sie aber z.B. die wichtige Klasse „System.String“ an, so finden Sie dort schon fünf Memberabteilungen: Konstruktoren, Felder, Methoden, Operatoren und Eigenschaften.

Über den .NET API-Browser finden Sie eine Übersicht aller Namensräume. Über das Suchfeld oben können Sie schnell nach Elementen suchen, ohne selbst die ganze Liste durchzugehen.

Seit Visual Studio Express 2012 konnten wieder wichtige Dokumentationen zur „Klassenbibliothek“ lokal bereitgestellt werden und damit die Hilfeeinträge während der Programmierarbeit deutlich schneller zur Verfügung stehen. Seit Visual Studio 2017 und auch für 2019 steht zwar die lokale Hilfe zur Verfügung, aber dort wird aktuell nur die .NET Framework Dokumentation für die Versionen 4.6 und 4.5 bereitgestellt. Diese lokale Hilfe ist jedoch bei der Programmierarbeit sehr nützlich und steht dadurch schnell zur Verfügung.

5.2 Die lokal bereitgestellte Hilfe

Befassen Sie sich deshalb nun mit der lokalen Hilfe-Bereitstellung. Dazu ändern Sie zunächst im Hilfemenü die Vorauswahl über „*Hilfeeinstellungen festlegen > In Help Viewer starten*“. Beim ersten Mal kommt dann ein kleiner Dialog mit der Anfrage:

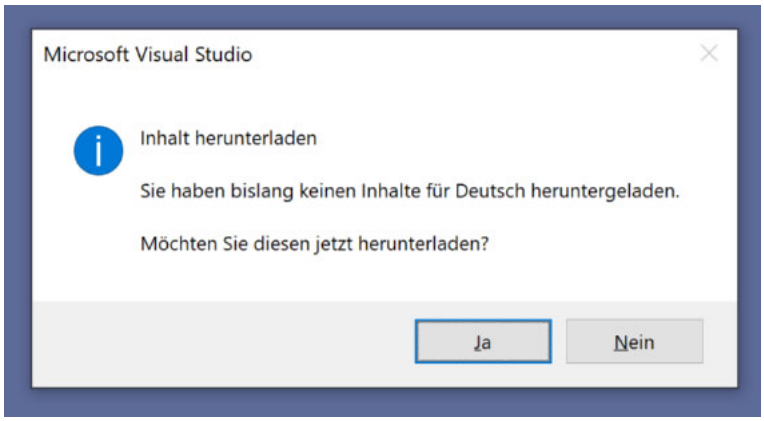


Abb. 5.6: Herunterladen der Dokumentationen

Bei Bestätigung startet der „Microsoft Help Viewer 2.3“ und zeigt eine lange Liste möglicher Dokumentationsmodule – viele in deutscher und englischer Sprache gedoppelt. Mit einem Mausklick auf einen der Links „Hinzufügen“ können Sie einen Eintrag auswählen – er wird dann in die Spalte „Ausstehende Änderungen“ übernommen, der angeklickte Link wandelt sich in die Bezeichnung „Abbrechen“. Mit einem weiteren Klick darauf können Sie die getroffene Modulauswahl wieder rückgängig machen (nach erfolgreicher Installation eines Hilfemoduls heißt der Link dann „Entfernen“). Abb. 5.7 zeigt den Beginn der Auswahlliste nebst einiger Module, die in die Installationsliste rechts aufgenommen wurden, darunter auch die angebotene deutsche Version (4.6) zum .NET-Framework. Auch wenn wir die Spezifika von .NET 4.6 in diesem Lehrgang nicht ausschöpfen, enthält diese Dokumentation doch (kumulativ) alle Typen der früheren Versionen 3.5 und 4 und ist somit eine der wichtigsten Informationsquellen.

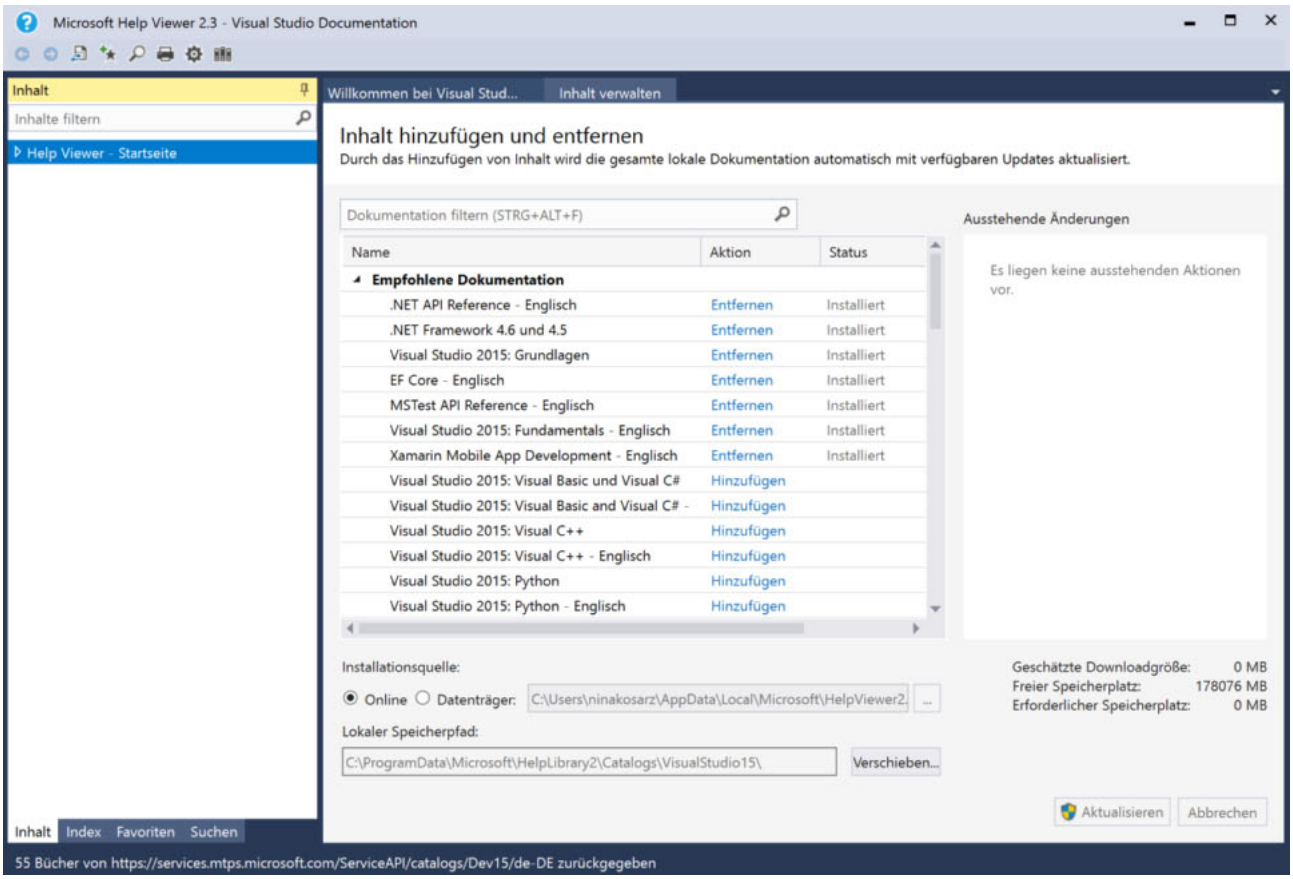


Abb. 5.7: Auswahl von Hilfemodulen im Help Viewer zum Download und zur lokalen Verfügbarmachung. Die lediglich vier ausgewählten Module (unter „Ausstehende Änderungen“) benötigen bereits 8,8 GB Festplattenspeicher! Der Zugriff auf die Hilfe über vier verfügbare Registerkarten (Inhalt | Index | Favoriten | Suchen) ist hier auf Einträge am linken Fensterrand minimiert (siehe dazu auch Abb. 5.8).

Unten rechts in der Ecke werden Sie über das Downloadvolumen sowie den Speicherbedarf für die entpackten Hilfemodule auf Ihrer Festplatte informiert. Den lokalen Speicherpfad für die Hilfedateien *<Systemlaufwerk>:\ProgramData\Microsoft\HelpLibrary2\Catalogs\VisualStudio15* sollten Sie beibehalten wie er vorgegeben ist (auch wenn die Bezeichnung etwas verwirrend ist) und solange es nicht zu eng auf Ihrer Systemplatte (~-partition) wird.

Ein Klick auf „Aktualisieren“ gefolgt von der Genehmigung des Eingriffs auf den Rechner startet den Download, dessen Fortschritt rechts unten in der Statusleiste des Help Viewer angezeigt wird. Nach dem Download folgt eine Überprüfungsphase mit sehr intensiver Festplattenaktivität, sodann werden die Hilfepakete entpackt und indiziert. Der Vorgang schließt (wenn alles gut ging) mit der Statusleistenmeldung „Update war erfolgreich“.

Für den Zugriff auf die nun lokal verfügbare Hilfe nutzen Sie den **Navigationsbereich** links, der über Reiter am Fuß den Wechsel zwischen Inhalt, Index, Favoriten und Suchen anbietet. Mit einem Klick auf den Pin im Titelbalken des Navigationsbereichs kann dieser Bereich auf den linken Rand des Help Viewer-Fensters minimiert werden (so die Darstellung in Abb. 5.7). Zurzeit gibt es die Visual Studio Grundlagen nur für die Version

2015 auf deutsch, 2017 und 2019 stehen nur als englische Fassung zur Verfügung. Mit ausgeklappter Navigation und Startseite der Hilfe zeigt sich der Help Viewer mit den in Abb. 5.7 zum Download angezeigten Modulen wie in Abb. 5.8:

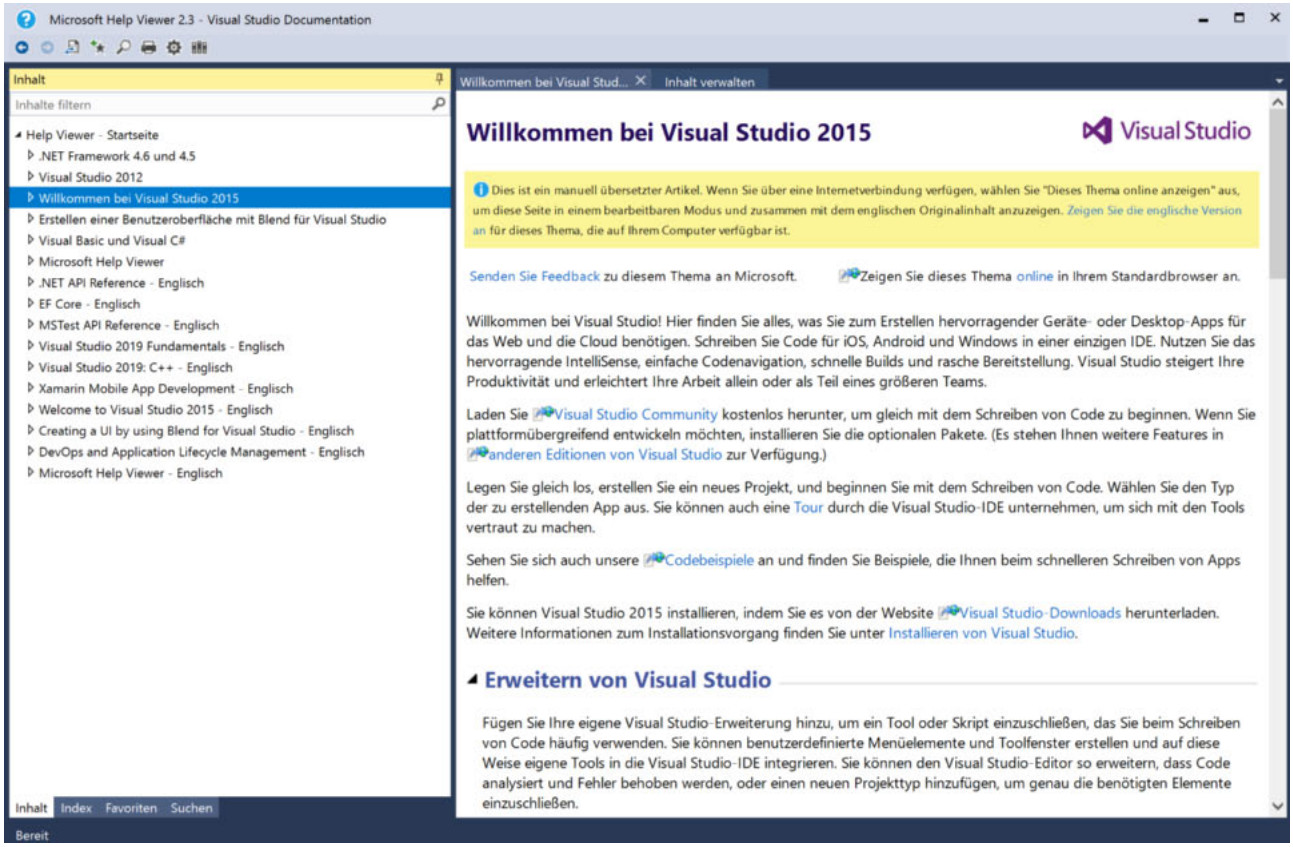


Abb. 5.8: Einstieg in die lokale Hilfe zu Visual Studio 2015, präsentiert vom Help Viewer 2.3; Angebot gem. Abb. 5.7 installierten Hilfemodule.

Die *lokale* Präsentation der Hilfe unterscheidet sich in manchem von der Online-Hilfe im Browser. So ist hier der Navigationsbereich links im Fenster wieder konventionell aufgebaut: Seine Breite wird vom Trennbalken im Fenster geregelt (nicht sichtbar abgesetzt am rechten Rand des vertikalen Scrollbalkens gelegen), die Baumstruktur wird durchgängig hierarchisch präsentiert und rückt immer mehr ein, je tiefer Sie expandieren. Im Hauptfenster macht ein farbig unterlegter Kasten zu Beginn deutlich, dass es sich um eine maschinelle Übersetzung handelt (das merkt man dann auch hin und wieder leidvoll). Insofern ist das Angebot nützlich, die (originalsprachliche) **Online-Dokumentation** durch direkten Link im Browser hinzunehmen zu können.

Werfen Sie zum Vergleich mit Abb. 5.5 auch einen Blick in die „Klassenbibliothek“. Dazu expandieren Sie den Haupteintrag „.NET API Reference - Englisch“ und scrollen bis zu dem Namensraum „System“ (vgl. Abb. 5.9). Dort scrollen Sie in der alphabetischen Auflistung aller zu „System“ gehörenden Typen bis hin zur Klasse „Array“. Nach deren Auswahl wird dieser Auswahlgegenstand im Hauptteil des Fensters angezeigt und sein Titel in den Reiter des Dokumentations-Teilfensters übernommen (Abb. 5.9).

Am Beginn einer Typendokumentation steht immer die Syntax seiner Deklaration. Der Typ „System.Array“ ist z. B. eine öffentliche abstrakte Klasse, worauf die drei (blau gestellten) Schlüsselworte in der Deklaration hinweisen: „public abstract class“. Diese De-

klarationssyntax unterscheidet sich in den diversen **Programmiersprachen**, die von der .NET-Plattform unterstützt werden. Für alle unterstützten Sprachen werden die .NET-Typen in der Dokumentation berücksichtigt. Sie können zwischen den sprachspezifischen Syntaxformen einfach über den **Sprachfilter** wechseln, der von einer Registerkartenleiste angeboten wird. In der Dokumentationswiedergabe von Abb. 5.9 sind die Angebote für C#, C++, F# und VB aktiviert (nicht hingegen JavaScript und JScript) und „C#“ ausgewählt.

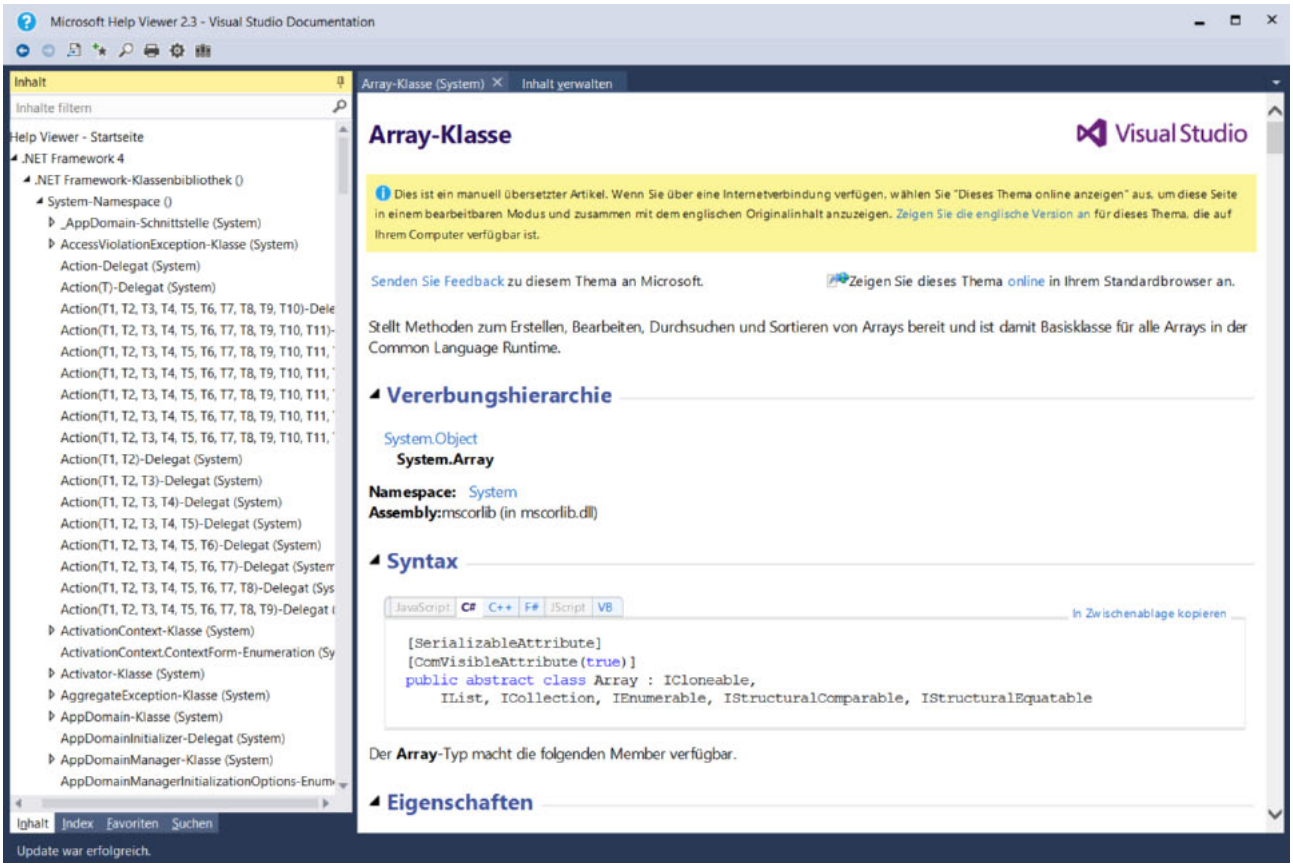


Abb. 5.9: Dokumentation der Klasse „System.Array“ in der lokalen Hilfe (Help Viewer 2.3). Die Navigation wurde nach Auswahl der Klasse „Array“ zum Beginn der Baumstruktur zurückgescrollt. (.NET Framework 4 > .NET Framework-Klassenbibliotheken () > System-Namespace () > Array)

Die Hilfe ist ein komplexes Gebilde vielfach untereinander verlinkter Ressourcen. Da kostet es oft nur einen Klick und Sie befinden sich im rechten Hauptteil des Help Viewer-Fensters in der Dokumentation eines Typs, der links in der Baumstruktur gar nicht angezeigt wird. Es gibt aber einen Weg, das angezeigte Dokumentationsthema mit der Baumstruktur zu synchronisieren. Das ist vor allem praktisch, wenn man sich in der Baumstruktur ansehen will, welche **Member**gruppen der gerade im Hauptfenster gezeigte Typ anbietet und darauf über die Baumstruktur gezielt zugreifen will.

Betrachten Sie die **Symbolleiste** des Help Viewer-Fensters in Abb. 5.10 (Pendants zu deren Symbolfunktionen in einem Menü gibt es nicht):

Abb. 5.10: In der asketischen Oberfläche von Visual Studio 2019 erscheint die Symbolleiste des Help Viewer 2.3 (beginnend mit dem Pfeil nach links) nicht mehr abgesetzt von der Titelleiste (beginnend mit dem Symbol der Anwendung, dem Fragezeichen) – alle Symbole sind im unkonturierten grauen Block des Fensterkopfes angeordnet.

- Die Pfeile nach links und rechts unterstützen die **Navigation** durch die bereits besuchten Hilfeseiten. Komfortabel sind auch die zugehörigen Tastaturkommandos: nach links (zurück) mit [←] oder [Alt] + [←], nach rechts (vor) nur mit: [Alt] + [→].
- Das dritte Symbol mit den zwei winzigen, nach rechts und links weisenden Pfeilen auf der linken unteren Ecke eines stilisierten Blatts Papier (mit Eselsohr) verbirgt die **Synchronisation** zwischen Dokumentations- und Navigationsbereich im Help Viewer-Fenster (wobei die Navigation an den gerade präsentierten Dokumentationsinhalt angepasst wird). Sie ist im Popup-Text mit „Thema in Inhalten anzeigen“ beschrieben und auch über die Tastenkombination [Strg] + [S] auszulösen („S“ wie „Synchronisation“).
- Mit dem nachfolgenden Symbol „add Stern“ können Sie das aktuelle Hilfethema in eine **Favoritenliste** übernehmen, die über die Registerkarte „Favoriten“ im Fuß des Navigationsbereichs einen schnellen Zugriff auf die darin versammelten Elemente gewährt. Jeder dort gelistete Eintrag hat ein Kontextmenü, über das er u.a. auch wieder aus dieser Liste entfernt werden kann (Tastenkombination [Strg] + [D]).
- Das Lupensymbol blendet rechts oben ein Eingabefeld zur **Suche** auf der aktuell dokumentierten Themenseite ein (auch mit [Strg] + [F] – wie „find“).
- Das Druckersymbol bietet den **Druck** des aktuellen Hilfethemas an, ergänzt um eine Vorschauoption, die im sich öffnenden Fenster korrekt „Druckvorschau“, im Menü zum Drucksymbol hingegen „Seitenansicht…“ heißt. An dieser Druckfunktion ist angenehm, dass Sie lediglich das Hilfethema druckt – nicht gestört durch die Navigation links. Im Vorschaufenster können die Druckseiten weiter eingerichtet werden. Achtung: die Navigation durch die Druckseiten wird mit blassgrauen Pfeilen in der Fußleiste des Vorschaufensters unterstützt ([Strg] + [P]).
- Hinter dem Zahnrad-Symbol mit der Popup-Beschriftung „Viewer Optionen“ und dem Tastenkürzel [Strg] + [O] wird ein Optionen-Dialog aktiviert.
- Das letzte Symbol mit den drei Buchrücken und dem Popup-Text „Inhalt verwalten“ wechselt zur gleichnamigen Registerkarte … was man auch direkt mit einem Mausklick erledigen kann ([Strg] + [⇧] + [M]).

Sie können jederzeit (jedenfalls solange eine Internetverbindung aktiv ist) die lokal bereitgestellte Dokumentation ausbauen und weitere Module herunterladen. Die eingangs (Abb. 5.7) genutzte Oberfläche mit Download-Liste ist nach wie vor präsent und wird auf die Registerkarte „**Inhalt verwalten**“ geladen, sobald Sie darauf wechseln. Falls das Help Viewer-Fenster noch nicht geöffnet ist, führt auch die Menüfolge „*Hilfe > Hilfeinhalt hinzufügen oder entfernen*“ direkt auf diese Registerkarte.

5.3 Hilfekapitel entfernen (lokale Hilfe)

Die per Download installierten Hilfemodule werden nicht unter den auf Ihrem Rechner installierten Programmen gelistet (*Systemsteuerung > Programme und Funktionen*), lassen sich somit von dort aus auch nicht entfernen. Zur Beseitigung der opulenten Hilfe-Datenmassen von Ihrem Rechner ist die Oberfläche auf der Help Viewer 2.3-Registerkarte „Inhalt verwalten“ vorgesehen (direkt zugänglich über „*Hilfe > Hilfeinhalt hinzufügen oder entfernen*“), wo für die zu entfernenden Hilfekapitel in der Gesamtauflistung die Links „Entfernen“ zu klicken sind. Vollzogen wird dann die Operation über den Button „Aktualisieren“ rechts unten.

Das sagt sich aber an dieser Stelle so einfach. Leider werden Sie wohl kaum noch an diesen Hinweis denken, wenn Sie irgendwann Visual Studio von Ihrem Rechner entfernen und somit die alten Dokumentationsdateien nicht mehr gebraucht werden, aber im Bereich „<Systemlaufwerk>\ProgramData“ viel Platz wegnehmen.

Vielleicht hilft es ja, wenn Sie dann zur Orientierung einen Blick in die Ablage der Programmdaten werfen können, deren Ordner per Voreinstellung im Windows Explorer ausgeblendet sind:

Versteckte Windows 10-Ordner und -Dateien anzeigen

Unter Windows 10 sind per Voreinstellung wichtige Ordner der Betriebssystem-Installation versteckt und somit nicht im Windows Explorer sichtbar. Das betrifft z. B. auch den Pfad, in dem die MSDN-Dokumentation zu Visual Studio 2019 lokal gespeichert wird (vgl. in Abb. 5.7 die Angabe zum lokalen Speicherpfad in der Benutzeroberfläche links unten). Sie können auch diese per Voreinstellung *ausgeblendeten* Ordner sichtbar machen:

Öffnen Sie den **Windows Explorer**. Dort gehen Sie ins Menü „Ansicht“ und wählen die Option „Optionen“. Im Dialogfenster wechseln Sie auf die Registerkarte „Ansicht“. In der Checkbox-Liste zu „Erweiterte Einstellungen“ gibt es weiter unten den Haupteintrag „Versteckte Dateien und Ordner“, unterhalb dessen über zwei Radiobuttons ausgeblendete Dateien, Ordner oder Laufwerke angezeigt oder wieder versteckt werden können.

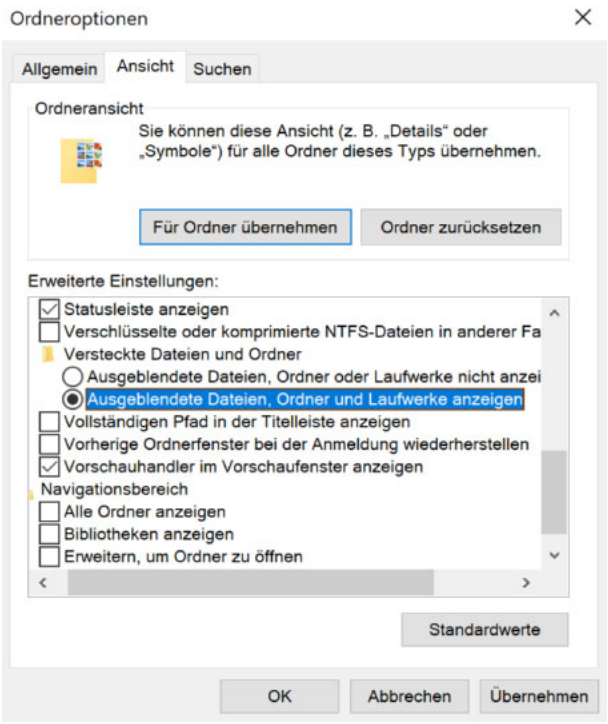


Abb. 5.11: Ein-/Ausblenden versteckter Dateien in einem Dialog des Windows 10-Explorers (Menü „Organisieren > Ordner- und Suchoptionen“)

Klicken Sie auf die Anzeige-Option und bestätigen Sie den Dialog mit „OK“, so werden die versteckten Ordner im Windows Explorer (mit einem blasseren Symbol für den Hauptordner) eingeblendet. Beachten Sie, dass es sich bei den nun sichtbaren Objekten meist um Systemdateien handelt, die von Programmen oder vom Betriebssystem genutzt werden. Sie sollten auf diese Dateien deshalb nicht ändernd zugreifen.

6 Visual C# Referenz

Abb. 4.5 hatte den ersten Blick auf die IDE „Visual Studio Community 2019“ geöffnet. Damit Sie die Erläuterungen auch praktisch nachvollziehen können, wird im ersten Abschnitt knapp beschrieben, wie Sie zu einem einfachen Visual Studio-Projekt gelangen. Dieser Vorgang wird in den fachlichen Studienheften ständige Praxis sein.

Im zweiten Abschnitt dieser Referenz stehen die Techniken im Zentrum, mit denen Sie sich die Entwicklungsumgebung nach Ihren Vorstellungen einrichten können.

Der dritte Abschnitt gibt eine Übersicht über die wesentlichen Teilfenster und ihre Aufgaben. Der vierte Abschnitt befasst sich mit der Verwaltung von Visual Studio-Projekten – der organisatorischen Hülle für Ihre Programmierideen.

6.1 Neues Visual Studio-Projekt

Ein neues Visual Studio-Projekt können Sie rechts (vgl. Abb. 4.5) über den Button „Neues Projekt erstellen…“ beginnen. Die gleiche Funktion wird im Menü über „*Datei > Neues Projekt…*“, über das dritte Symbol in der Symbolleiste (nach den Navigationspfeilen) oder über die Tastenkombination Strg + ⇧ + N unterstützt. Es öffnet sich ein Dialog, in dem die Art des Projekts auszuwählen ist (schon einmal in Abb. 4.6 wiedergegeben, mit anderer Auswahl in Abb. 6.1). Unter Abänderung der Voreinstellungen wurde bereits die Programmiersprache C# (links) sowie der Projekttyp „Konsolenanwendung“ (Mitte) ausgewählt.

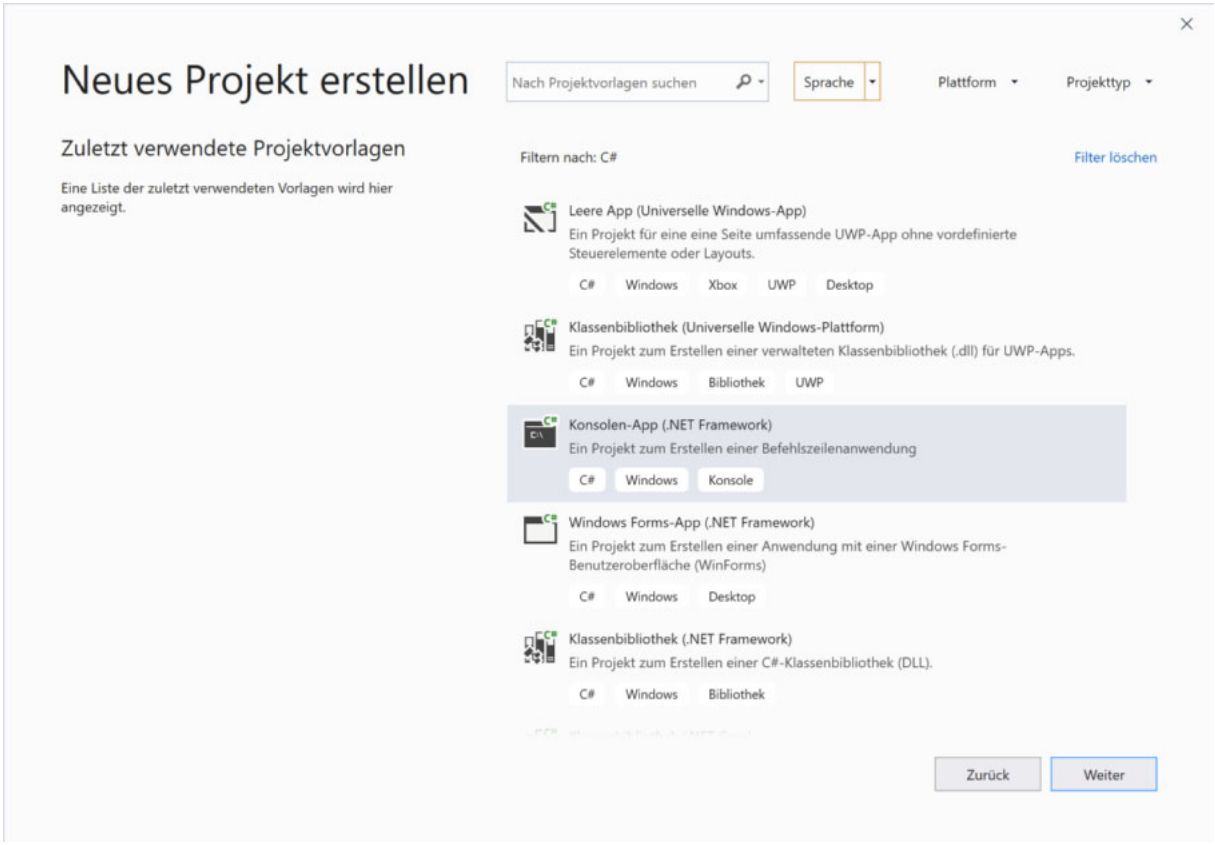


Abb. 6.1: Konfiguration eines neuen Visual Studio-Projekts

Wenn Sie Ihre Wahl getroffen haben, klicken Sie auf *Weiter*. Daraufhin erscheint ein neues Fenster (Abb. 6.2), in dem drei weitere wichtige Dinge festzulegen sind, mit denen dann alle Eckdaten des gewünschten Projekts definiert sind:

- Das Projekt muss einen aussagekräftigen **Projektnamen** erhalten – der voreingestellte Dummy „ConsoleApplication1“ (beim nächsten Projekt würde dann „ConsoleApplication2“ vorgeschlagen usw.) erfüllt dies Kriterium zweifellos nicht.
- Dann folgt ein Eingabefeld für den Speicherort des neuen Projekts, der einfach über den benachbarten Button „Durchsuchen…“ festgelegt werden kann.
- In der untersten Zeile ist außerdem ein **Projekt*mappen*name** zu vergeben. Die Voreinstellung schlägt den gleichen Namen vor wie für das Projekt. Das ist ebensowenig sinnvoll.

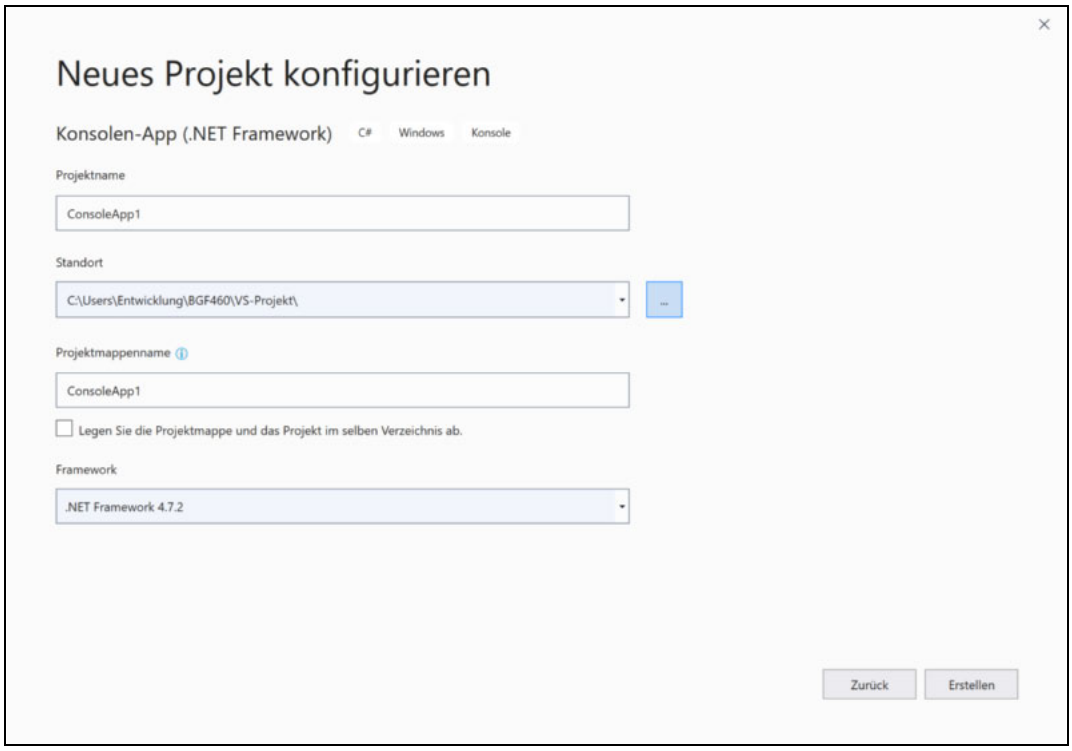


Abb. 6.2: Neues Projekt konfigurieren

Statt Ihre Projekte, wie vom Dialog vorgeschlagen, in den Untiefen der Microsoft Benutzerverwaltung zu versenken, sollten Sie (jedenfalls solange Sie auf einem Einzelplatzrechner arbeiten) eher eine übersichtliche Ordnerstruktur für diesen Lehrgang anlegen, in der jedes Studienheft seinen Ordner erhält. Darin kann dann in einem Unter-ordner mit dem Namen „VS-Projekte“ eine Projektmappe angelegt werden, die – wie schon erwähnt – in diesem Lehrgang für jede Lektion mit Programmierinhalt ein Visual Studio Projekt umschließen wird. Die Projektmappe würden Sie dann wie das Studienheft, und das Projekt wie die Lektion benennen, ggf. kommt hin und wieder auch ein inhaltlicher Projektname in Frage. So könnte sich z.B. die folgende Struktur ergeben, die sich durch weitere Unterordner zu den Studienheftordnern (z.B. für Bearbeitungsnotizen, Einsendeaufgaben usw.) ergänzen ließe:

```
<Laufwerk>
  Entwicklung
    CSH-Lehrgang
      BGF460
        VS-Projekte
          BGF460 [Projektmappe]
            Fenster-Handling [Projekt]
      CSH01
        VS-Projekte
          CSH01
            Lektion2
            Lektion3
            ...
      CSH02
      ...
```

Aus diesem Entwurf eines lehrgangsbezogenen Dateisystems ergäbe sich dann aktuell als Projektmappenname „BGF460“. Als Name des ersten Projekts in dieser Projektmappe wird „Fenster-Handling“ vorgeschlagen (Konsolenanwendung). So entsteht der fertig ausgefüllte Dialog in Abb. 6.3. Der Teilpfad „*C:\Users\Entwicklung\BGF460*“ wurde mithilfe des Durchsuchen-Buttons festgelegt, der gewünschte Unterordner „*\VS-Projekt*“ ergänzend eingetragen. Visual Studio legt jeden Ordner in diesem Pfad automatisch an, der noch nicht existiert.

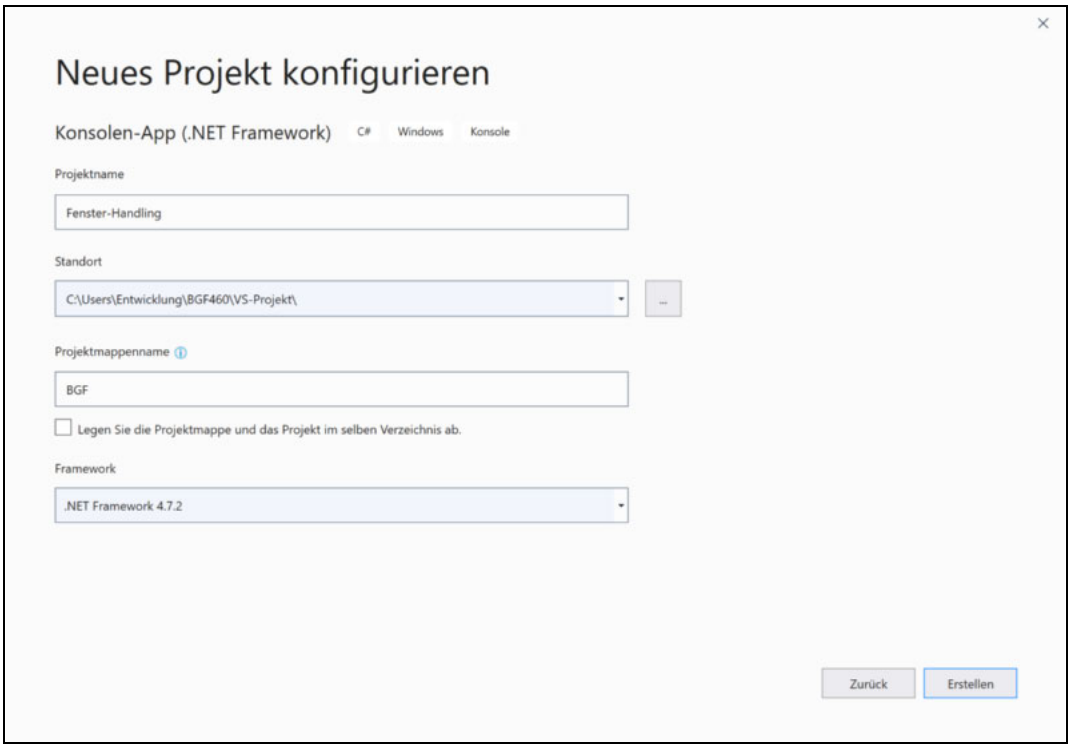


Abb. 6.3: Fertig ausgefüllter Dialog zur Konfiguration eines neuen Projekts

Beachten Sie:

Visual Studio bietet nur einmal die Möglichkeit, den Projektmappennamen festzulegen, nämlich im Dialog von Abb. 6.2 bzw. Abb. 6.3. Mit den Einträgen in diesem Dialog werden auch die Namen der Projektordner im Dateisystem festgelegt. Eine

Funktion, die gesamte Projektmappe mit ihren eingeschlossenen Projekten unter anderen Namen an anderer Stelle zu speichern, gibt es nicht. Sie können später lediglich das gesamte Dateisystem zu einer Projektmappe im Windows Explorer an eine andere Stelle verlagern und dann erneut in der IDE öffnen. Überlegen Sie sich also vor dem Einstieg in ein neues Projekt immer gut, wie Sie das erste Projekt und seine umhüllende Projektmappe nennen.

Wenn Sie ein Projekt im Dateisystem (Windows Explorer) verlagert haben, wird es von der IDE dennoch auf der Startseite unter „Zuletzt“ (... bearbeitete Projekte) angeboten. Ein Klick darauf führt dann allerdings zu einem Dialog, der das Entfernen des nicht mehr von der IDE referenzierten Projekts anbietet (Abb. 6.4):

Abb. 6.4: Dialog, der sich nach einem Klick auf ein im Dateisystem nicht mehr auffindbares Projekt öffnet.

Danach können Sie das verlagerte Projekt ganz normal über „Projekt öffnen...“ laden, wobei Sie die **Solution-Datei** (*.sln) des Projekts auswählen, die im Projektmappenordner liegt.

Die Projekterstellung beginnt nach einem Klick auf „OK“ und wird von einem kleinen Dialog mit Fortschrittsanzeige begleitet. Der Vorgang generiert nicht nur ein neues Projekt im Arbeitsspeicher, sondern legt dies auch gleich im Dateisystem ab. Dies führt zu einer Projektstruktur im Dateisystem, die Abb. 6.5 als Screenshot des Windows Explorers zeigt:

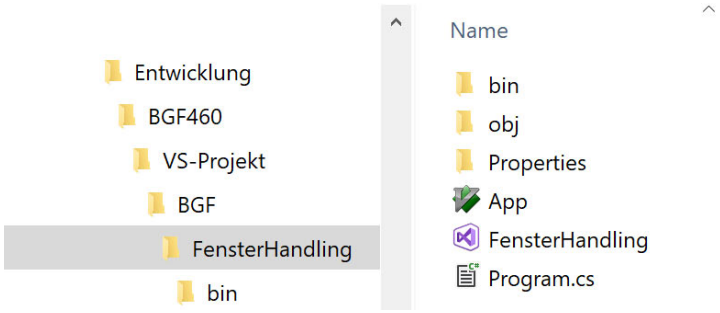


Abb. 6.5: Dateistruktur zum C#-Programmiererlehrgang, beginnend mit dem generierten Visual Studio-Projekt zum Lehrgangsbegleitheft.

Im Projektmappenordner „BGF“ ist eine *.sln-Datei mit den Eckdaten zu dieser „Solution“ (= Projektmappe) entstanden. Dabei handelt es sich um eine Text-Datei mit hierarchischer Gliederung, die Eckwerte der Projektsammlung beschreibt. Ferner liegt dort eine zugehörige Binärdatei vom Typ *.suo.

Im Projektordner „Fenster-Handling“ findet sich u.a. eine Datei „Program.cs“, die den gesamten Programm-Quellcode dieser Anwendung umfasst (Abb. 6.5, rechter Explorer-Teil).

In der IDE hat sich auch allerlei getan (Abb. 6.6):

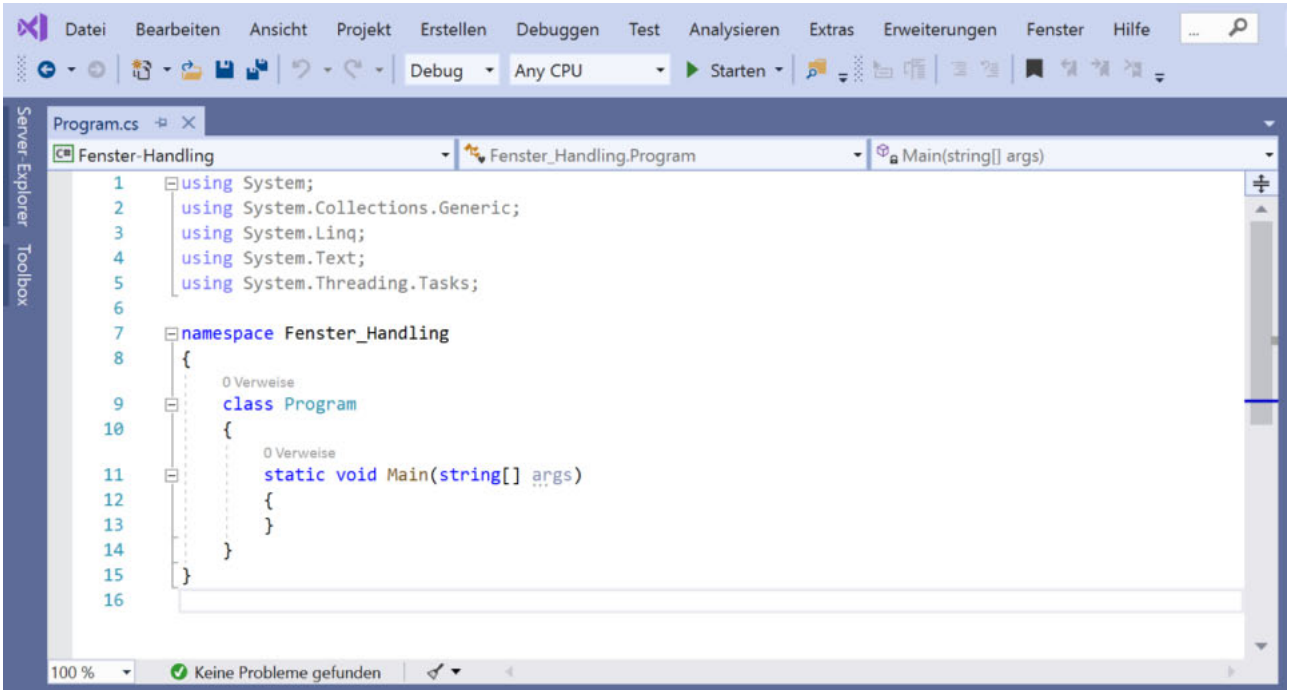


Abb. 6.6: Die IDE mit dem ersten Projekt im Projektmappen-Explorer und dessen Datei „Program.cs“ im Editorbereich unmittelbar nach Projektgenerierung aus dem Dialog in Abb. 6.3 (IDE für den Screenshot zusammengeschoben).

Der Projektmappen-Explorer ist mit Einträgen zum ersten Projekt gefüllt. Darunter hat sich ein weiteres Teilfenster „Eigenschaften“ eingefunden. Im Hauptbereich der IDE wurde die Startseite durch einen Editor ersetzt, der die C# Quellcode-Datei „Program.cs“ (siehe auch Windows Explorer in Abb. 6.5) anzeigt und zur Bearbeitung bereithält. Zurück zur Startseite geht es, wie bereits gesagt, über das Ansicht-Menü.

Der Name des Projekts ist auch zum Bezeichner des Namensraums geworden, der die erste (und bislang einzige) Klasse des Projekts umschließt – mit einem kleinen Unterschied: Da Namensraumbezeichner im Wesentlichen Buchstaben und den Unterstrich zulassen, wurde aus dem Projektnamen „Fenster-Handling“ der Namensraumbezeichner „Fenster_Handling“. Dieser Namensraum ist in seiner Funktion grundsätzlich mit den „großen“ Namensräumen „System“ nebst Unternamensräumen vergleichbar, die Sie bereits im Zusammenhang mit der Dokumentation gesehen haben, ist aber offensichtlich nicht so bedeutend. Auch unser neuer winziger Namensraum sammelt Typen, hier aber nur die Klasse „Program“. Ein Namensraum wird mithilfe des Schlüsselworts „namespace“ festgelegt, das die IDE aufgrund der Voreinstellungen in den Optionen (siehe Abschnitt 5.4) wie alle Schlüsselwörter blau einfärbt.

6.2 Fensteranordnung

Wie schon im Vorspann zu diesem Kapitel erwähnt, kann sich die Visual Studio-IDE aus allerlei Teilfenstern zusammensetzen, also aus mehr als nur den beiden Teilfenstern in Abb. 6.5. und Abb. 6.6 hatte schon ein drittes gezeigt („Eigenschaften“). Diese zwei (bzw. drei) repräsentieren aber bereits die zwei grundsätzlich verfügbaren Typen von Teilfenstern:

- **Dokumentfenster** dienen zur Darstellung und Bearbeitung von Dateien oder anderen Objekten. Dazu zählt das Startfenster, insbesondere fallen aber in diese Rubrik diverse Editorfenster zur Bearbeitung von Programmcode- oder von Konfigurationsdateien.
- **Werkzeugfenster** bieten Hilfsfunktionen bei der Arbeit. Sie werden vor allem im Ansicht-Menü angeboten.

Das Menü-Angebot für Fenster beider Art hängt vom Typ der aktuellen Anwendung ab. Zu den Werkzeugfenstern zählen bereits der Projektmappen-Explorer und das Eigenschaften-Fenster. Weitere (darunter auch Dokumentenfenster wie die Einbindung eines Webbrowsers) zeigt Abb. 6.7 im geöffneten Ansicht-Menü, bzw. in einem daraus über den Eintrag „Weitere Fenster“ zu öffnenden Untermenü:

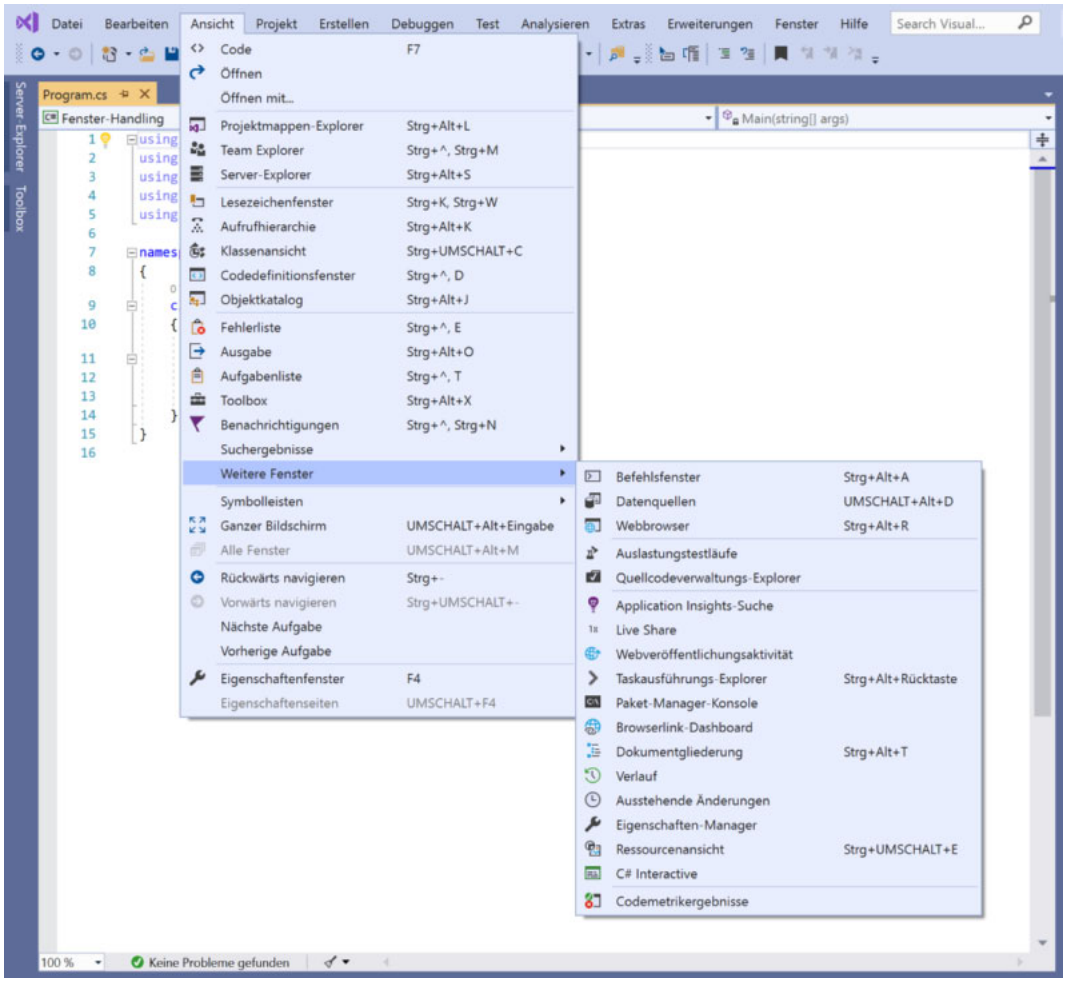


Abb. 6.7: Zugang zur Visual Studio-Fenstervielfalt über das Menü „Ansicht“ sowie dessen Untermenü-Eintrag „Weitere Fenster“

Alle Werkzeug-Teilfenster können verschoben werden, sodass sie entweder – auch über den Fensterraum der IDE hinaus – frei **schweben** oder an einer Stelle innerhalb des IDE-Fensters „**andocken**“. Außerdem können Teilfenster noch mit anderen Teilfenstern zusammen über **Registerkarten** kombiniert werden. Die zugrundeliegende Technik ist allerding nicht ganz unkompliziert gestaltet und erfordert etwas Einübung.

Wichtig ist die Vorbereitung, dass die zu verlagernden Teilfenster (Werkzeugfenster) dauerhaft sichtbar sein sollten – sonst funktioniert das Ganze nochmal anders. Die Fixierung eines Teilfensters erreichen Sie über den **Pin** in seiner Titelleiste, der durch Mausklick „umgelegt“ werden kann:

- Steht der Pin senkrecht, so ist das Fenster fixiert. Dem entspricht die Funktion im Menü „Fenster“ bzw. im Kontextmenü „Automatisch im Hintergrund“.
- Liegt der Pin (nach einem Mausklick darauf) hingegen flach, so verschwindet das Fenster, sobald Sie an eine andere Stelle in der IDE klicken. Diese Umschaltung müssen Sie im Menü „Fenster“ bzw. im Kontextmenü über den Eintrag „Andocken“ vornehmen. Logischer wäre wohl gewesen, die Funktion „Automatisch im Hintergrund“ als aktivierbar/deaktivierbar zu gestalten. Einfacher und intuitiver ist jedenfalls ein Mausklick auf den Pin.

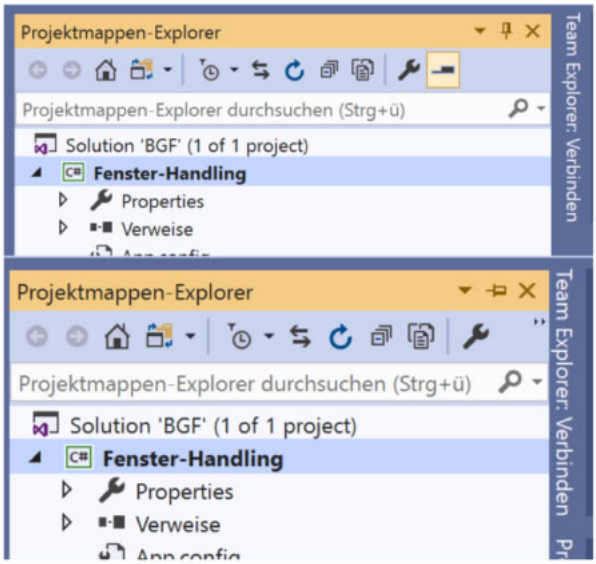


Abb. 6.8: Der Pin in einer Teilfenster-Titelleiste: oben vertikal (= Fenster dauerhaft sichtbar), unten horizontal (= Fenster nur nach Klick sichtbar), letztere Einstellung kombiniert mit Minimierungsposition am Fensterrand, repräsentiert durch die Titelleistenbeschriftung und einen Balken.

Ein ausgeblendetes Teilfenster können Sie wieder anzeigen, indem Sie auf seine Beschriftung klicken, die für nicht fixierte Fenster vertikal am linken oder rechten Rand der IDE neben einem Balken angeordnet wird (Abb. 6.8). Auch dieser Balken variiert. Normalerweise ist er grau, schwebt hingegen die Maus darüber, wird er blau.

Ein Werkzeugfenster greifen Sie mit der Maus an seiner Titelleiste und beginnen mit gedrückter linker Maustaste zu ziehen. Dann schwebt das Fenster frei auf dem Bildschirm. Halten Sie die Maustaste gedrückt! Sonst wird das Fenster unabhängig von der IDE auf dem Bildschirm abgelegt. Sie können es allerdings erneut an der Titelleiste anfassen und wieder bewegen.

Solange das abgelöste Teilfenster (genauer: der dies Teilfenster an der Titelleiste führende Mauszeiger) über dem IDE-Fenster schwebt, wird das IDE-Fenster mit allerlei Symbolen überblendet. In der Mitte erscheint ein Kreuz mit 9 Einzelsymbolen. Weitere vier derartige Symbole leuchten an den Rändern des IDE-Fensters auf. Microsoft nennt diese Symbole „**Diamant-Führungssymbole**“. Jedes dieser Führungssymbole repräsentiert einen bestimmten Fensterbereich innerhalb des IDE-Fensters, d. h. ein dort anzulegendes Teilfenster (daher auch die Symbolik mit dem oberen dicken Balken, der die Titelleiste repräsentieren soll).

Abb. 6.9 zeigt den Beginn der Werkzeugfensterverschiebung am Beispiel des Projektmappen-Explorers in einer ersten Variante. Denn die Lage des abgelösten Teilfensters entscheidet darüber, wo das zentrale Kreuz der Führungssymbole erscheint – erneut genauer: die Lage des das Teilfenster führenden Mauszeigers entscheidet. Da dieser über dem Editorfenster (mit der C#-Datei „Program.cs“) schwebt, wird auch über diesem Fenster das zentrale Kreuz mit den Führungssymbolen eingeblendet.

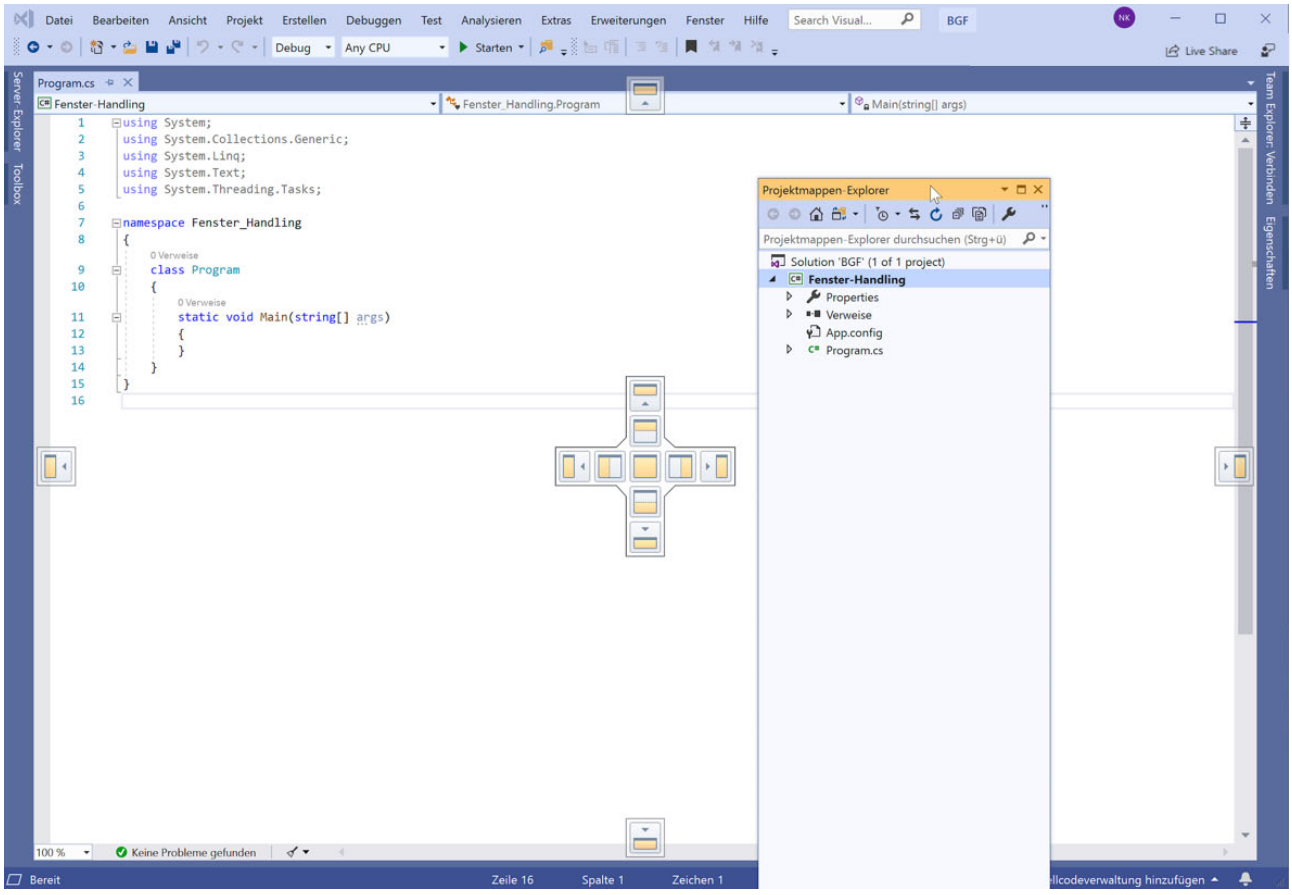


Abb. 6.9: Der Mauszeiger, der das abgelöste Teilfenster (Projektmappen-Explorer) führt, schwebt über dem Editor-Teilfenster. Deshalb wird dies mit dem zentralen Führungssymbole-Kreuz ergänzt.

Abb. 6.10 stellt dem eine zweite Variante gegenüber, in der der Mauszeiger über dem Eigenschaften-Fenster schwebt. Dann erscheint das zentrale Führungssymbole-Kreuz in diesem Fensterbereich (allerdings in der Anzahl seiner Symbole reduziert).

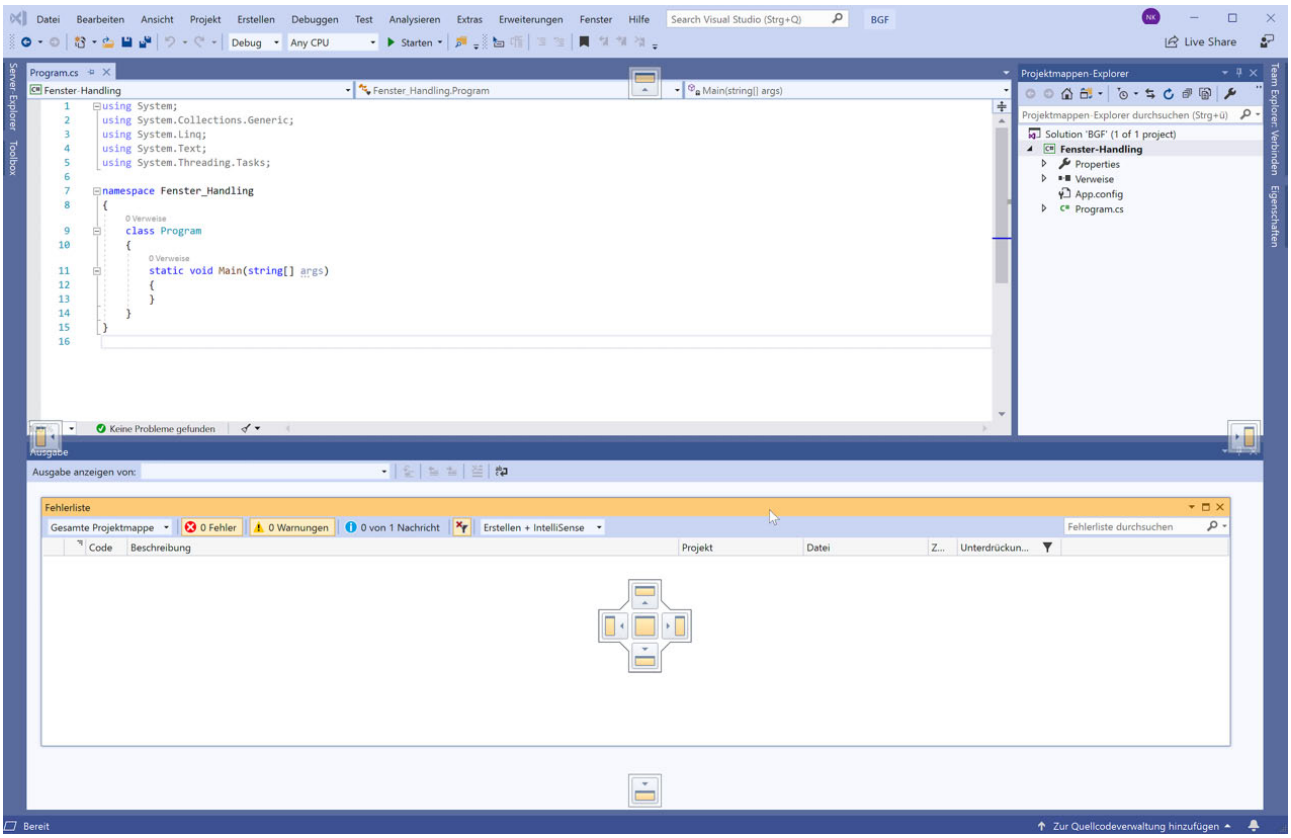


Abb. 6.10: Da der Mauszeiger, der das abgelöste Teilfenster führt, nun über dem Eigenschaften-Fenster schwebt, erscheint das zentrale Führungssymbole-Kreuz (in reduzierter Form) nun über diesem Teilfenster.

Die eingeblendeten Führungssymbole nutzen Sie, indem Sie den Mauszeiger auf eines dieser Symbole ziehen, solange der Mauszeiger ein schwebendes Teilfenster führt. In diesem Moment wird der IDE-Teilbereich, der mit diesem Führungssymbol assoziiert ist, transparent blau hinterlegt. Lassen Sie nun die Maustaste los, so wird das geführte Teilfenster in diesem **Zielbereich** platziert. Dabei lassen sich vier Zielbereiche unterscheiden:

1. Die vier äußeren Führungssymbole nahe der IDE-Ränder repräsentieren genau diese **IDE-Ränder**. Wählen Sie eins dieser Symbole aus, so wird das bewegte Teilfenster an dieser Seite über die gesamte IDE-Breite bzw. -Höhe angedockt. Es überspannt also ggf. mehrere Teilfenster, sofern diese bereits in der IDE angeordnet sind.

Alle anderen Optionen im zentralen Führungssymbol-Kreuz beziehen sich auf das Teilfenster, über dem der führende Mauszeiger gerade schwebt. Sie haben bereits in den Abb. 6.8 und Abb. 6.9 gesehen, dass dies Kreuz entweder neun (über dem Dokumentfenster) oder nur fünf Führungssymbole (über dem Werkzeugfenster) aufweisen kann. Der Unterschied liegt in den gestrichelt unterteilten Teilfenstersymbolen des 9-er Kreuzes. Dieses Kreuz ist noch einmal in Abb. 6.11 zu sehen. Die Zielbereiche 2 bis 4 sind dort ergänzend nummeriert.

Es handelt sich um folgende IDE-Bereiche:

2. Der jeweilige Rand (oben, rechts, unten oder links) des gerade unterliegenden Teilfensters. Das dort angedockte Teilfenster erstreckt sich lediglich auf die Höhe oder Breite des referenzierten Teilfensters und erhält eine durchgehende Titelleiste.
3. Andocken im Registerkartenformat: Hier wird ebenfalls der jeweilige Rand des gerade unterliegenden Teilfensters angesteuert. Jetzt (mit dem gestrichelt unterteilten Symbol) wird aber das dort anzudockende Teilfenster so gestaltet, dass es als erste Registerkarte auftritt und sodann durch weitere Teilfenster ergänzt werden kann, die im gleichen Bereich als Registerkarten geschaltet werden. So kann mit einem ersten Teilfenster vorbereitet werden, dass in seinem Bereich platzsparend mehrere Teilfenster alternativ anzuordnen sind.
4. Bei Wahl des zentralen Feldes wird das schwebende Teilfenster als weitere Registerkarte zum unterliegenden Teilfenster ergänzt und ist dann nur alternativ zu diesem über einen Mausklick auf den Registerkartenreiter anzuzeigen.

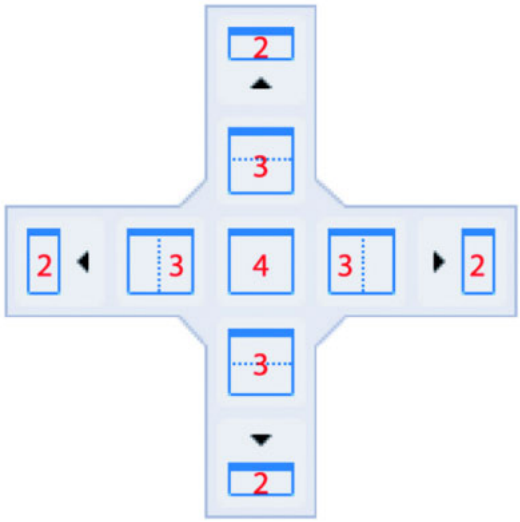


Abb. 6.11: Teilfenster-bezogenes Kreuz der Führungssymbole mit den Zielbereichen 2 bis 4 (Bereich 1 steht in dieser Systematisierung für die Ränder des IDE-Fensters als Ganzes)

Für das wichtigste, das **Dokumentfenster**, haben Sie neben der Option, es frei schweben zu lassen, nur die Option des **Registerkarten-Führungssymbols**: Sie können das Dokumentfenster nur als Hauptelement in den IDE-Rahmen einfügen und sich diesen Raum – gegliedert über Registerkarten – gegebenenfalls mit weiteren dort schon vorhandenen Fenstern teilen. Abb. 6.12 zeigt diese Situation.

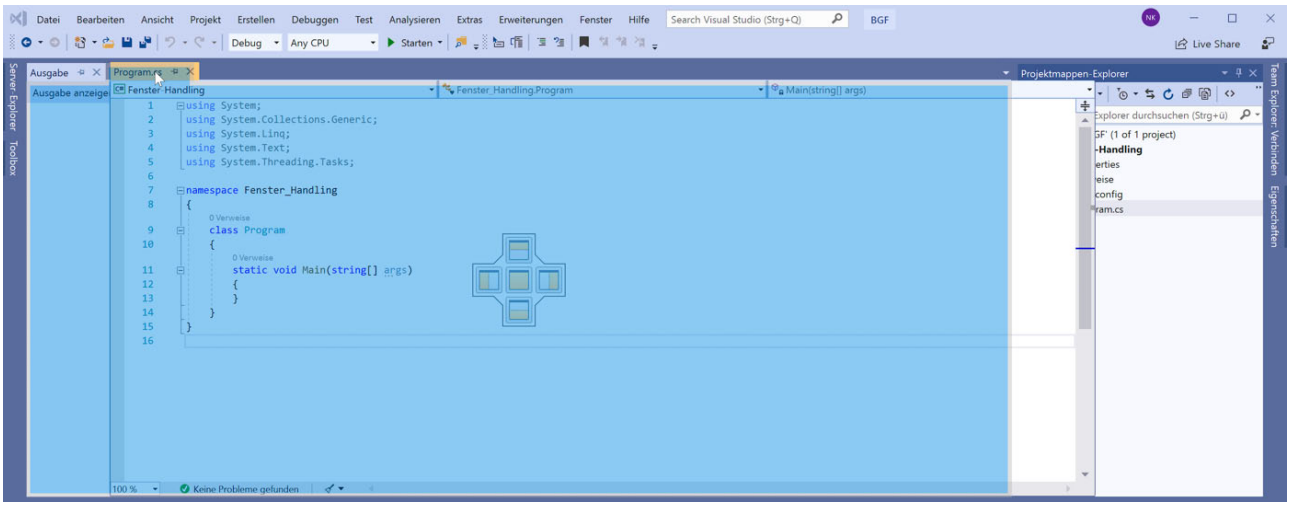


Abb. 6.12: Das zentrale Dokumentfenster (Editor mit „Program.cs“) ist aus seiner Verankerung gezogen – als Zielbereich wird nur der verlassene Raum mit Registerkartenanordnung angeboten.

Die Situation der Abb. 6.12 entsteht, wenn Sie das Dokumentfenster „Program.cs“ an seinem Reiter mit der Maus angreifen und aus seiner Verankerung ziehen. In der Mitte erscheint das Führungssymbole-Kreuz lediglich mit dem zentralen Registerkarten-Zielbereich.

Ergänzbare Teilfenster aus dem Ansicht-Menü bringen eine Voreinstellung mit, wie sie in den IDE-Rahmen eingefügt werden. Nehmen wir als Beispiel die Fensteranwahl „Ansicht > Ausgabe“. Das Ausgabefenster zeigt Ausgaben der IDE-Arbeit und wird automatisch unter dem Dokumentfenster angeordnet – das entspricht dem Zielbereich „2 unten“ (gemäß Abb. 6.11). Abb. 6.13 präsentiert eine mögliche Nutzung dieses Werkzeugfensters:

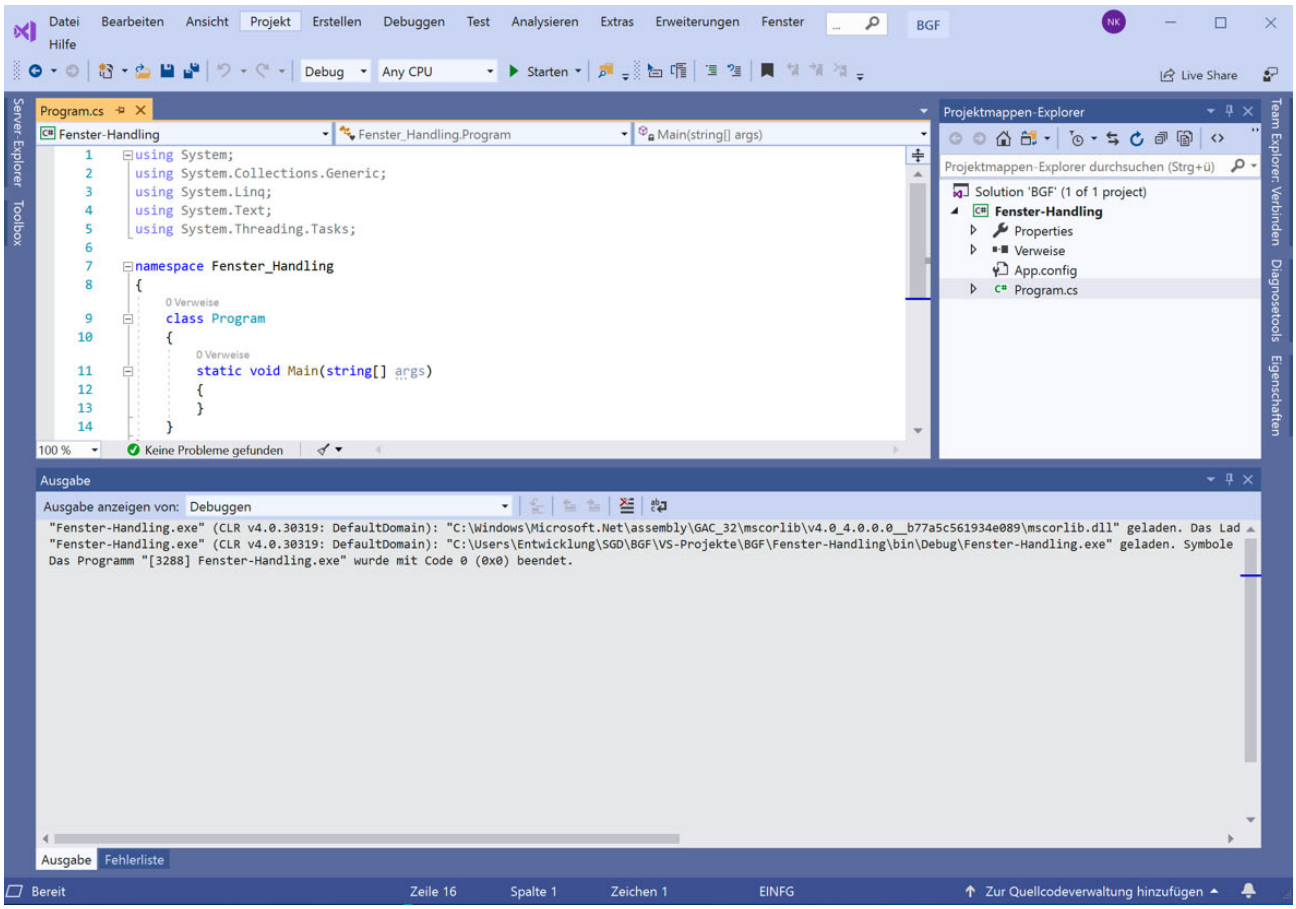


Abb. 6.13: Das Ausgabefenster unter dem Dokumentfenster mit einer Protokollierung nach Ausführung des aktuellen Programms.

Die Funktionen zur Anordnung von Teilfenstern lassen sich auch über die Menüs – das jeweilige Kontextmenü bzw. das Fenster-Menü – bedienen. Abb. 6.14 zeigt das Kontextmenü zum eben eingefügten Ausgabe-Werkzeugfenster. Die Funktion „Andocken“ ist ausgegraut, also deaktiviert, weil sich das Werkzeugfenster schon im Andockmodus befindet. Was bewirken die anderen Menüfunktionen?

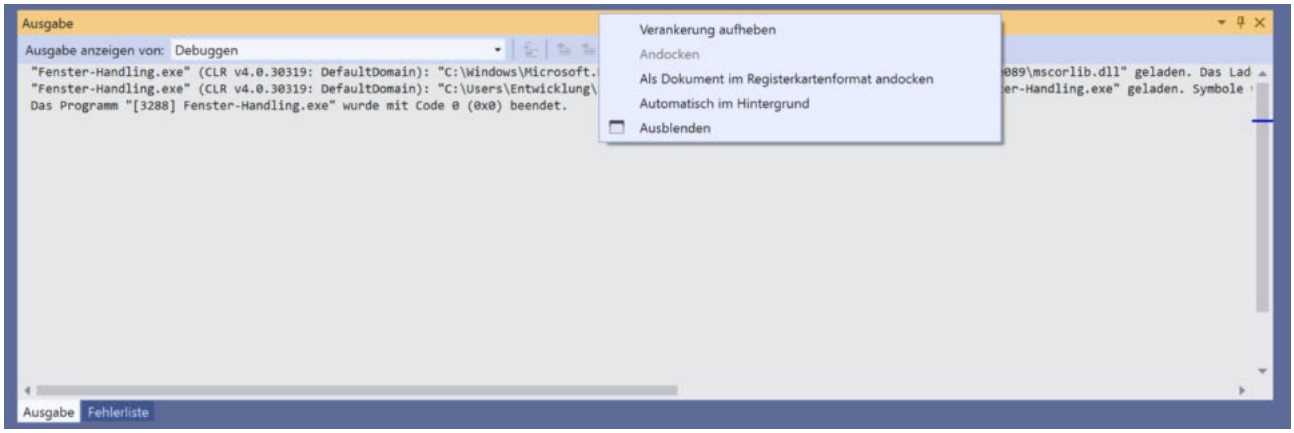


Abb. 6.14: IDE-Ausschnitt mit Kontextmenü zum Werkzeugfenster „Ausgabe“

- „Verankerung aufheben“ löst das Teilfenster aus dem IDE-Verbund, sodass es frei auf dem Bildschirm schwebt. Das entspricht dem Herausziehen eines Teilfenster an der Titelleiste (bzw. seiner Registerkarte) mit der Maus. Nun ist im Kontextmenü die Funktion „Andocken“ aktiviert, über die das Teilfenster an seine alte Stelle zurückkehrt.
- Der Menüeintrag „Als Dokument im Registerkartenformat andocken“ führt das Teilfenster nicht in den Zielbereich 3 gemäß Abb. 6.11, sondern gliedert es als Registerkarte zum anliegenden Teilfenster ein – auf „Ausgabe“ angewandt, wird dies Teilfenster also zur Registerkarte neben „Program.cs“.
- Der Menüeintrag „Automatisch im Hintergrund“ entspricht dem ‚Umlegen‘ des Pins. Das Teilfenster wird also auf einen Eintrag im Rand der IDE minimiert. „Ausgabe“ wird so zur Schaltfläche am unteren Rand. Diese Funktion steht nur im Andockmodus, also nicht in der Registerkartenvariante zur Verfügung. Die Hintergrund-Funktion ist nützlich, wenn es im IDE-Fenster eng wird. Die Werkzeugfenster stehen dann prinzipiell zur Verfügung, werden aber nur bei Bedarf per Mausklick auf den Randeintrag eingeblendet.
- Der Menüeintrag „Ausblenden“ entfernt das Teilfenster aus dem IDE-Rahmen. Danach muss es ggf. erneut aus dem Ansicht-Menü geholt werden.

Die beschriebenen Funktionen der Kontextmenüs werden auch im **Fenster-Menü** der IDE zum gerade aktiven (ausgewählten / angeklickten) Teilfenster angeboten. Ein aktives Teilfenster zeigt sich immer mit blauer Titelleiste (bzw. blauer Registerkarte). Ebenso bietet der Klappladen / das Dropdown-Menü hinter dem nach unten gerichteten Dreieck in der Titelleiste eines Werkzeugfensters die aufgelisteten Funktionen an.

6.3 Werkzeugfenster-Übersicht

In früheren Visual Studio-Versionen wurde in der Symbolleiste noch ein Schnellzugriff auf die wichtigsten Werkzeugfenster bereitgestellt. Die grafische Ausnüchterung der Benutzeroberfläche, die ab Visual Studio 2012 einherging und mit der auch die farbenfroh gestalteten Symbole verschwanden, hatte auch zur Begleiterscheinung, dass nun gar keine Werkzeugfenster für den Direktzugriff über Symbole mehr angeboten werden; Sie sind also vollständig auf das zuständige Menü „Ansicht“ verwiesen (vgl. Abb. 6.7).

Abb. 6.15: Drei wichtige Werkzeugfenstersymbole in Visual Studio 2019 und die dazugehörigen Tastaturkurzbefehle

In diesem Abschnitt sollen Sie sich die wichtigsten Werkzeugfenster noch etwas genauer betrachten. Neben dem Zugang über das Ansicht-Menü hat Microsoft auch Tastaturkommandos zur Aktivierung von Teilfenstern vorgesehen, über die die Menüeinträge informieren.

6.3.1 Projektmappen-Explorer

Der Projektmappen-Explorer ist das wichtigste Werkzeugfenster, in dem alle Projekte einer Projektmappe erstellt sowie deren Eigenschaften, Referenzen und Dateien verwaltet werden. Es sollte immer im IDE-Rahmen vorhanden sein, deshalb erübrigt sich auch das Merken des komplizierten und nicht-intuitiven Tastaturkommandos. Falls es auf der IDE zu eng werden sollte, legen Sie ggf. den Pin um, sodass der Explorer am rechten IDE-Rand in Bereitschaft verbleibt.

Er ist ähnlich dem Windows-Explorer als Baumstruktur organisiert und hat zu jedem Projekt vor allem zwei Unterordner: „Properties“ und „Verweise“. Der Projektmappen-Explorer war bereits ab Abb. 6.6 mehrfach zu sehen. Abb. 6.16 modifiziert die bisherigen Darstellungen durch Expansion seiner beiden Projekt-Unterordner:

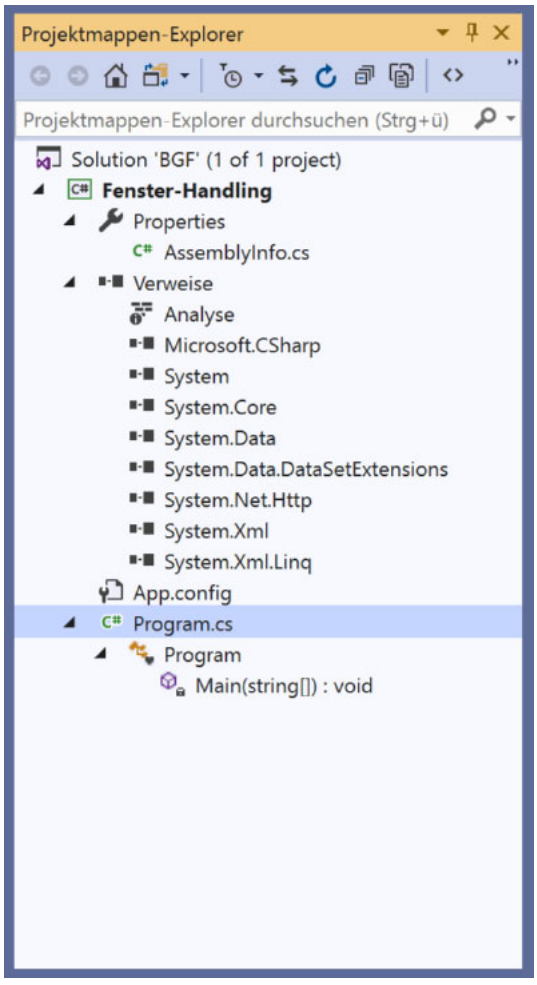


Abb. 6.16: Projektmappen-Explorer zum Projekt „Fenster-Handling“ – Alle Einträge expandiert

Im Ordner „**Properties**“ („Eigenschaften“) verwaltet Visual C# Projekteinstellungen in CSharp-Dateien, die immer den Dateityp „*.cs“ besitzen. Sie können die Datei „AssemblyInfo.cs“ per Doppelklick (oder Kontextmenü > Öffnen) in einem Dokument-Fenster der IDE mit C#-Editor öffnen. Dieses zweite Dokumentfenster wird dann als weitere Registerkarte neben das erste platziert. Ein „X“ am rechten Ende des gerade aktiven Registerkartenreiters bietet an, das Fenster mit einem Mausklick wieder zu schließen.

Der Ordner „**Verweise**“ verwaltet Referenzen auf Programmbibliotheken oder andere Projekte. In der Standardeinstellung einer einfachen Konsolenanwendung werden hier bereits sieben Referenzen aufgelistet. Die Schreibweise dieser Verweise erweckt den Eindruck, als handele es sich um Namensräume. Sie werden in der Hilfe („.NET Framework-Klassenbibliothek“) jedoch nicht all diese „Namensräume“ wiederfinden. Denn hinter diesen Verweisen mit Namen wie denen von Namensräumen stehen Referenzen auf Dateien, in denen die .NET-Typen real gespeichert sind. Es handelt sich dabei um Programmbibliotheken vom Dateityp **dll** (Dynamic Link Library).

Das können Sie sofort sehen, sofern noch Ihr Eigenschaften-Werkzeugfenster offen ist und Sie auf eine der Referenzen im Projektmappen-Explorer klicken. Dann wird im Eigenschaftenfenster angezeigt, dass es sich z.B. bei der Referenz „System.Core“ um die Referenz auf die Datei „System.Core.dll“ handelt. Zur Eigenschaft „Pfad“ ist sodann im Eigenschaftenfenster angegeben, dass diese Datei unter *<Systemlaufwerk>\Program Files(x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.7.x\System.Core.dl* abgelegt ist.

Solche Verweise auf die Bibliotheksdateien sind technische Voraussetzung dafür, dass im Programmcode diese Namensräume bekannt gemacht und sodann genutzt werden können. In einer Konsolenanwendung werden fünf der eingebundenen Namensräume bereits automatisch durch sogenannte **using-Direktiven** im Programmcode bekannt gemacht (vgl. Dokumentfenster „Program.cs“ im Projekt „Fenster_Handling“, Abb. 6.13):

```
using System; using System.Collections.Generic; using System.Linq;
using System.Text; using System.Threading.Tasks;
```

Aufgrund der technischen Dateiverweise im Projektmappen-Explorer, kombiniert mit using-Direktiven im Programmcode „weiß“ die IDE, in welchen Dateien sie die .NET-Klassen, Strukturen usw. suchen soll, die Sie als Typen in Ihrer Programmierung benutzen.

6.3.2 Eigenschaftenfenster

Dies Werkzeugfenster hatte die IDE bereits mit dem ersten Projekt automatisch eingeblendet (vgl. Abb. 6.6). Wir hatten es soeben genutzt, um nähere Informationen über die „Eigenschaften“ der Verweise im Projektmappen-Explorer zu erlangen. Das war aber nur eine Verwendungsmöglichkeit dieses Fensters.

Besonders wichtig wird dies Werkzeugfenster, wenn Sie mit Visual Studio **grafische Benutzeroberflächen** erstellen. Wenn Sie es praktisch nachvollziehen wollen, so arbeiten Sie bitte folgende Punkte ab:

- Im Projektmappen-Explorer aktivieren Sie mit der rechten Maustaste das Kontextmenü zur Projektmappe *„BGF“ > Hinzufügen > Neues Projekt…*
- Im schon prinzipiell aus Abb. 6.1 und Abb. 6.3 bekannten Dialog wählen Sie „Windows-Forms-Anwendung“ und geben unten einen qualifizierten Namen an, z.B. „FormDemo“. Das Projekt wird im Projektmappen-Explorer als zweites Projekt eingetragen.

- Das erste Projekt „Fenster-Handling“ ist noch fett gestellt, das zweite „FormDemo“ hingegen mager. So signalisiert die IDE, welches Projekt gerade zur Ausführung bereitsteht. Das müssen Sie nun umstellen, indem Sie über das Kontextmenü zu „FormDemo“ den Eintrag mit dem Zahnradsymbol „Als **Startprojekt** festlegen“ auswählen. Nun ist dies Projekt fett gestellt.

Im Editor-Bereich der IDE hat sich ein mit „Form1“ beschriftetes kleines Fenster eingefunden, das bereits über die übliche Grundfunktionalität in der Titelleiste (Symbolbild der Anwendung links, Minimier-, Maximier- und Schließ-Schaltflächen rechts) verfügt. Dies wird später die Arbeitsfläche sein, auf der Sie eine Benutzeroberfläche interaktiv gestalten können (Bearbeitung solcher Benutzeroberflächen ab Studienheft CSH04B).

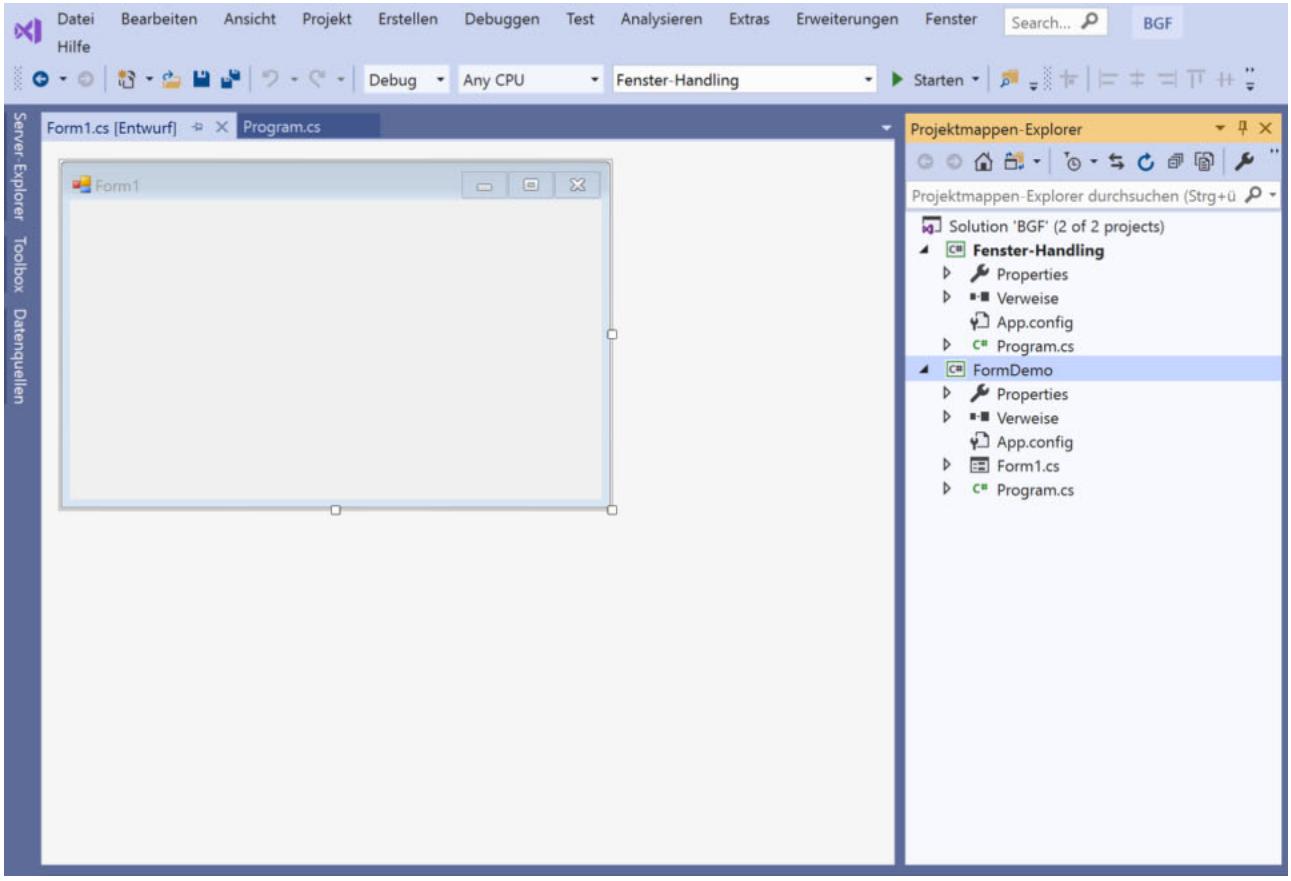


Abb. 6.17: Ein zweites Projekt im Projektmappen-Explorer (Typ „Windows Forms-Anwendung“) und die zugehörige Oberfläche im Editorbereich.

Die Inhalte des Eigenschaftenfensters sehen Sie, sobald Sie mit einem Mausklick die grafische Oberfläche mit dem noch provisorischen Namen „Form1“ aktivieren. Beachten Sie den Scrollbalken im Eigenschaften-Fenster und bewegen Sie ihn ein wenig. Erst jetzt werden Sie merken, wie lang die Liste der Eigenschaften einer „Form“ ist. All diese Eigenschaften können Sie im Eigenschaften-Fenster ändern bzw. konfigurieren.

Abb. 6.18 zeigt das Eigenschaftenfenster – die Länge des Scrollbalkens verweist aber darauf, dass noch mehr als 2/3 aller Einträge nicht zu sehen sind. Zudem können einzelne Kategorien in der Liste weiter expandiert werden (hier noch die ‚klassischen‘ Expandierschalter, die zwischen [+] und [-] wechseln und zum Erscheinungsbild bis Windows XP gehören).

Bei der Bearbeitung eines grafischen Objekts steht im Eigenschaften-Fenster unter der Titelleiste immer der Name und der Typ dieses Objekts – in Abb. 6.18 der von der IDE zunächst vergebene Name „Form1“ und die Information, dass dies ein Objekt der Klasse „Form“ ist, die Sie im Unternamensraum „System.Windows.Forms“ finden.

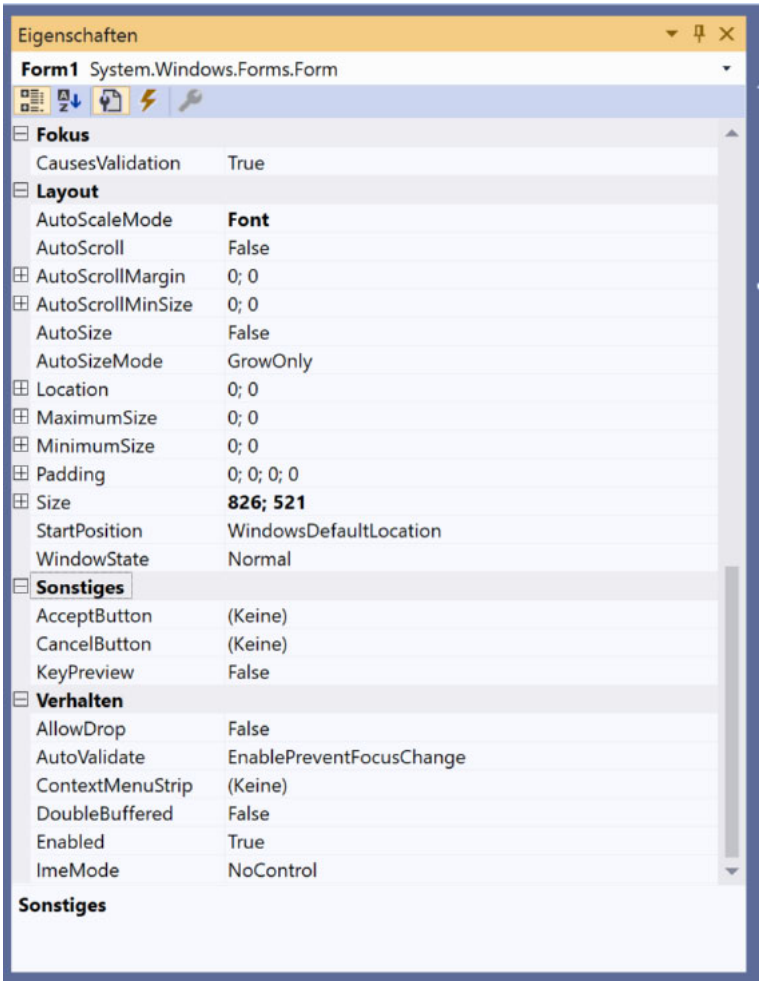


Abb. 6.18: Ausschnitt aus der Liste der Form-Eigenschaften im Eigenschaften-Werkzeugfenster

Alle Eigenschaften von grafischen Objekten werden im Eigenschaftenfenster tabellarisch dargestellt. In der linken Spalte steht der **Name** der Eigenschaft, in der rechten ihr **Wert**. In Abb. 6.18 ist der Eigenschaftsname „Text“ ausgewählt, in der rechten Spalte steht daneben der Wert „Form1“. Es handelt sich offensichtlich um die Beschriftung der Titelleiste. Markieren Sie mit der Maus diesen Werteintrag und überschreiben Sie ihn mit einem anderen Text. Nach Bestätigung mit der Enter-Taste wird diese neue Beschriftung in die Titelleiste der Form übernommen.

Manchmal versteckt sich zu einer Eigenschaft in der Wertspalte des Eigenschaften-Fensters ein kompletter Editor oder ein Dialog. Auch in diesen Fällen wird der Wert in dieser Spalte editiert und mit Enter bestätigt. Auf diese Weise können Sie mit dem Eigenschaften-Fenster interaktiv ganze Benutzeroberflächen gestalten. Denn diese Arbeit setzt sich fort, sobald in die Form weitere Steuerelemente (Buttons, Labels usw.) eingebaut werden und diese auf gleiche Weise mit dem Eigenschaften-Fenster zu gestalten sind.

6.3.3 Toolbox

Die bereits zum Eigenschaften-Fenster erwähnten Steuerelemente einer grafischen Benutzeroberfläche werden von einer Toolbox bereitgestellt. Sofern sich dieses Werkzeugfenster nicht bereits am linken IDE-Rand in minimierter Form versteckt, aktivieren Sie es über das Ansicht-Menü. Das Fenster wird standardmäßig im angedockten Zustand am linken Rand der IDE platziert. Abb. 6.19 zeigt den ersten Auftritt dieses Werkzeugfensters (dies sozusagen im doppelten Sinne) mit einer Kapitelübersicht all seiner mannigfaltigen Inhalte.

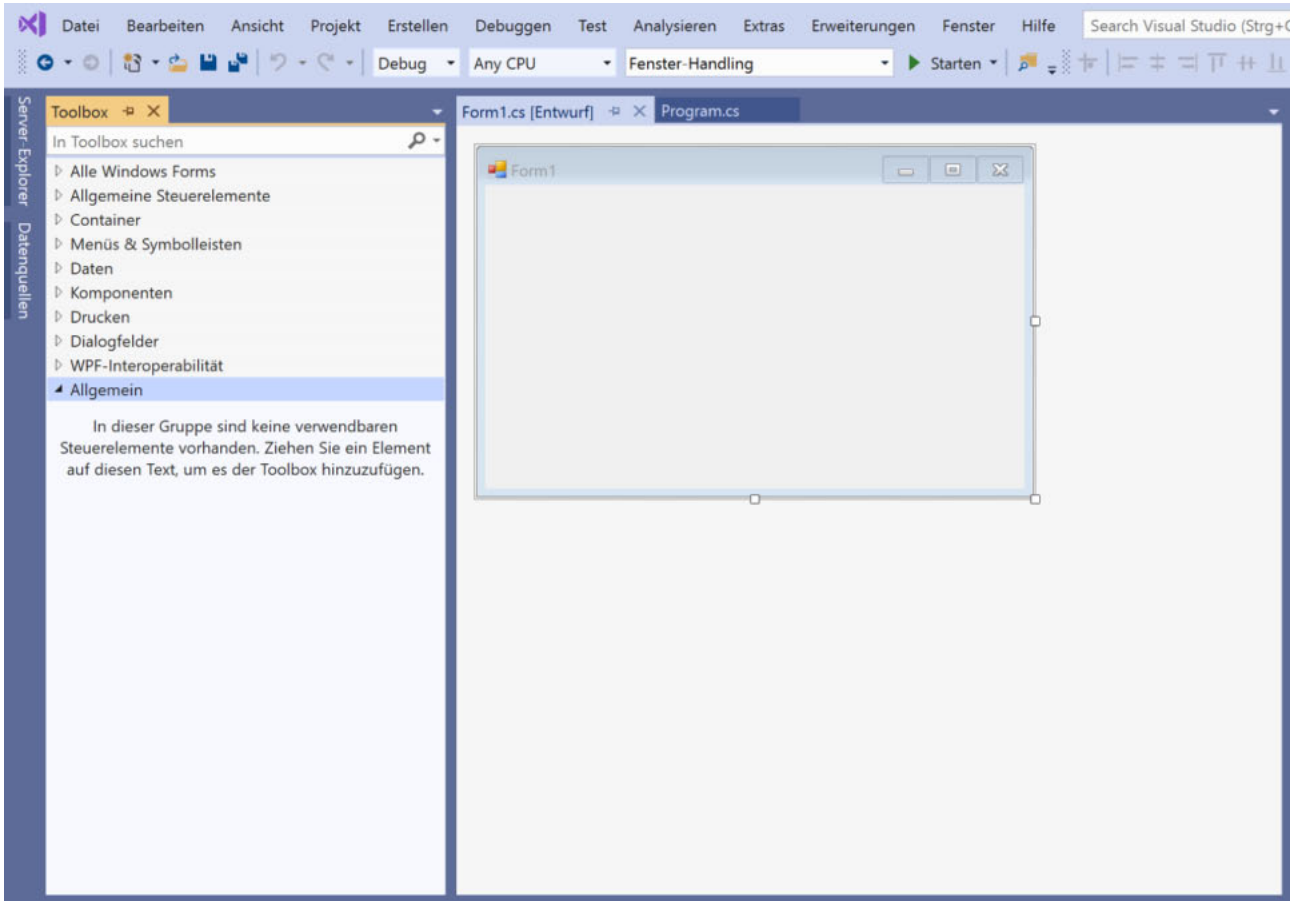


Abb. 6.19: Die Toolbox am linken Rand der IDE.

Die Einträge in der Kapitelliste der Toolbox lassen sich jeweils expandieren (hier mit den neuen, ab Windows Vista eingeführten dreieckigen Expandierschaltern). Abb. 6.20 zeigt als Beispiel den Beginn der langen Steuerelementeliste, die sich hinter der Überschrift „Allgemeine Steuerelemente“ verbirgt.

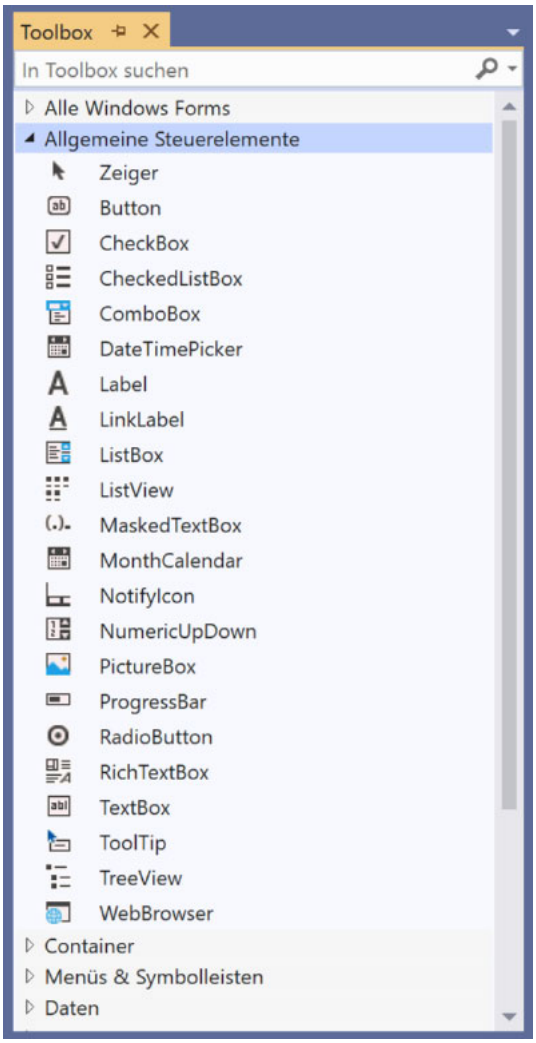


Abb. 6.20: Allgemeine Steuerelemente zur Gestaltung von Benutzeroberflächen – Button, Label, Checkbox und all das, was der Namensraum „System.Windows.Forms“ noch zu bieten hat.

Durch Auswahl in der Toolbox und Klick in die Form können diese Objekte dort platziert und dann im Eigenschaftenfenster weiterbearbeitet werden.

Da Sie das Eigenschaften-Fenster später vorrangig im Zusammenhang mit der Gestaltung grafischer Oberflächen benutzen werden, ist es ggf. sinnvoll, die beiden Werkzeugfenster „Toolbox“ und „Eigenschaften“ am linken IDE-Rand über Registerkarten zu kombinieren. So nehmen sie am wenigsten Platz weg (mit Toolbox links und Eigenschaftenfenster rechts wird es hingegen schon ganz schön eng für die Form in der Mitte). Zudem kann sich dort das Eigenschaftenfenster auf die volle Höhe erstrecken, die es angesichts der vielen Einträge zu Objekten einer grafischen Oberfläche auch braucht. Schließlich spricht für diese Anordnung, dass Sie zwar Werkzeug- und Eigenschaften-Fenster häufig in Kombination, aber immer nur im Wechsel benötigen: Entweder Sie platzieren ein Steuerelement oder sie konfigurieren es. Da passt die Registerkartenanordnung gut. Die Registerkartenreiter werden am Fuß angeordnet, wie es Abb. 6.21 als Ausschnitt aus der IDE zeigt:

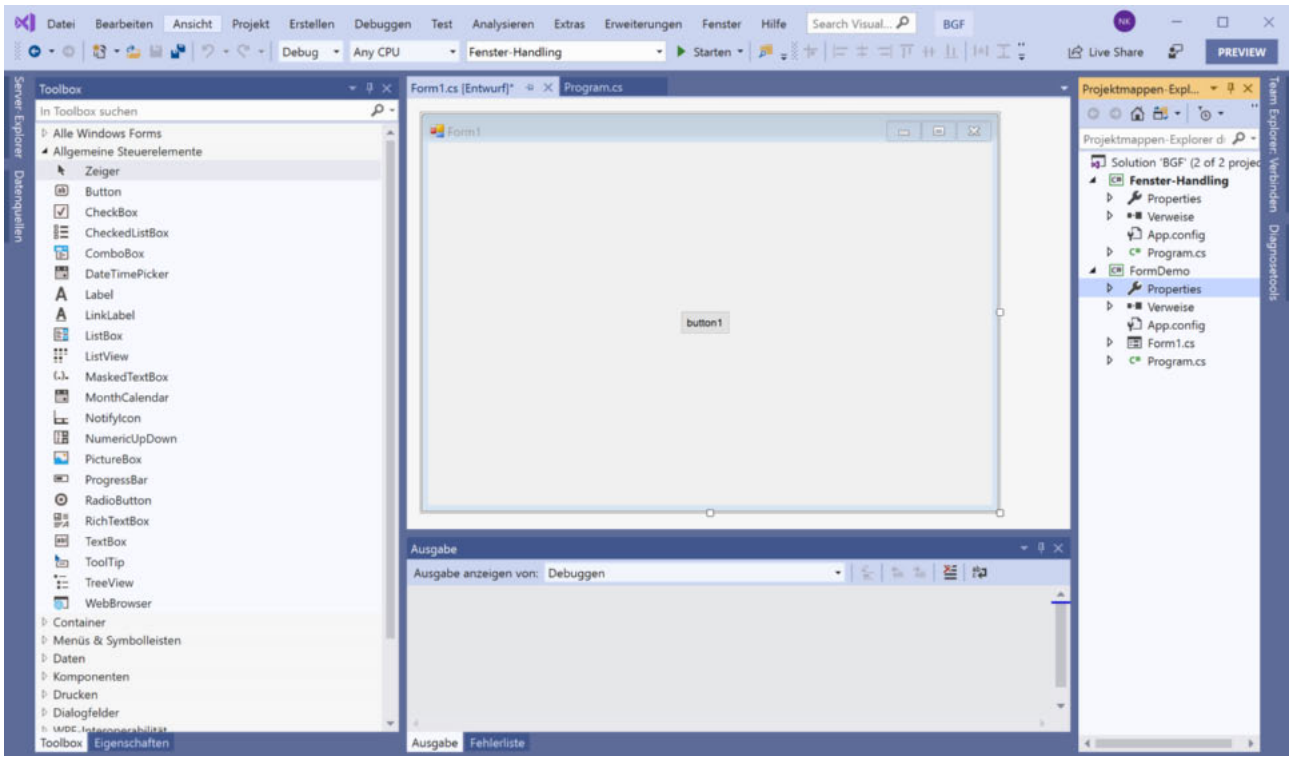


Abb. 6.21: Ausschnitt aus der IDE mit Eigenschaften-Fenster als Registerkarte zur Toolbox (Reiter am Fuß des Fensters) mit Anzeige der Eigenschaften eines in die Form eingefügten Buttons.

Solange Sie nur im Programmcode einer Konsolenanwendung arbeiten – z.B. im Rahmen unserer Einführungsprojekte: Auswahl der Registerkarte „Program.cs“ – bleibt das Eigenschaftenfenster ebenso wie die Toolbox grau und leer. Dann ist es angebracht, den Pin ‚umzulegen‘, d.h. das Registerkartenpärchen mit diesem einen Klick in den linken Rand zu minimieren. Jetzt haben Sie genügend Platz für den Programmcode.

6.3.4 Fehlerliste

Ein für die praktische Arbeit sehr wichtiges Werkzeugfenster ist die „Fehlerliste“. Sie wird (wie alle Werkzeugfenster) aus dem Ansicht-Menü heraus aktiviert und sollte in der IDE immer präsent sein (ggf. minimiert im Wartezustand). Abb. 6.22 zeigt die „Fehlerliste“ in Aktion, weil im Editor ein ziemlich sinnfreier und jedenfalls nicht im Einklang mit der C#-Syntax stehender Eintrag „abcdrf;“ vorgenommen wurde. Dieser wird von der IDE automatisch rot unterkringelt, womit sie auf einen **Fehler** hinweist. Da Sie in ernsthafteren Fehlerfällen wissen wollen, was es mit einer Fehlerunterkringelung auf sich hat, informiert Sie das Werkzeugfenster „Fehlerliste“ über mögliche Ursachen (bei „abcdrf;“ tut sie sich allerdings ziemlich schwer).

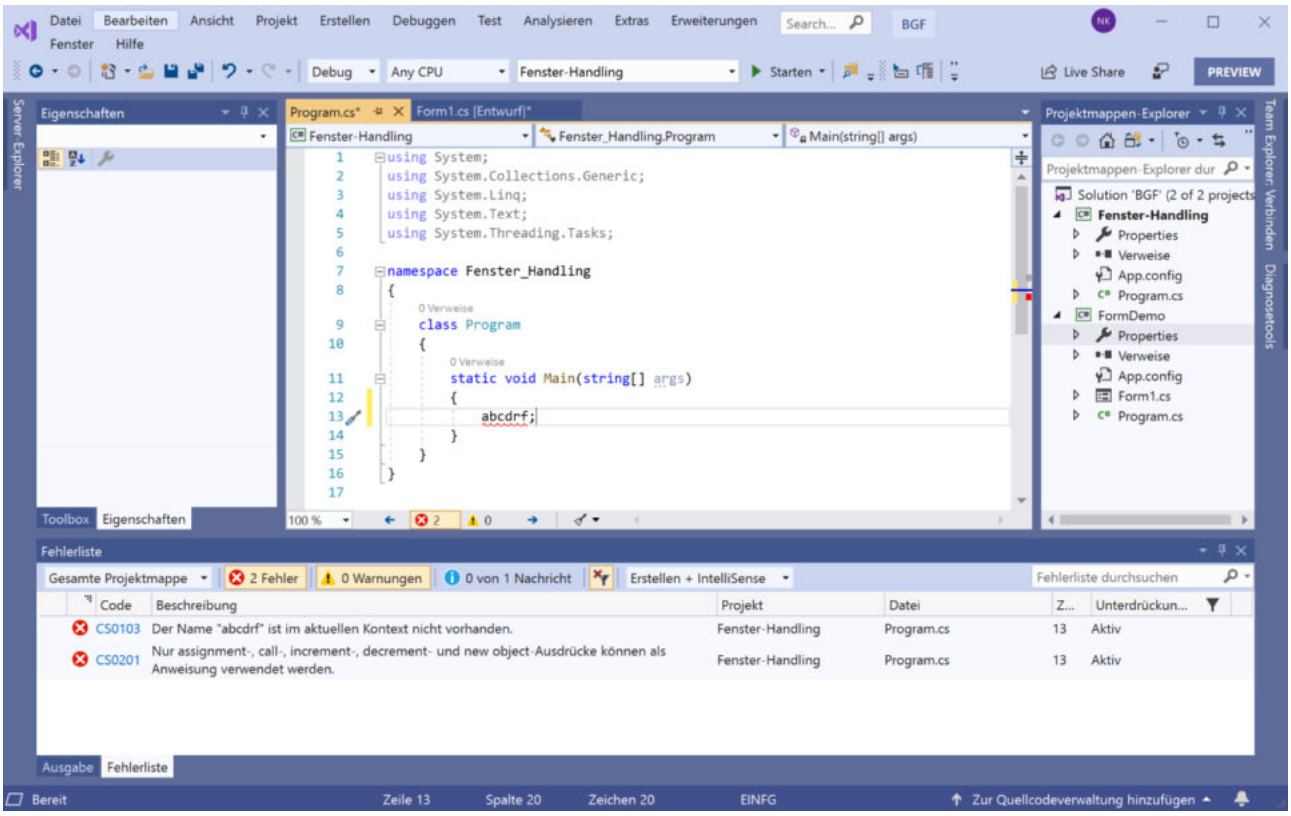


Abb. 6.22: Beispiel einer eingerichteten IDE: links die beiden vor allem für grafische Benutzeroberflächen wichtigen Fenster „Toolbox“ und „Eigenschaften“ in minimierter Form, rechts der Projektmappen-Explorer, ebenfalls minimiert. Unten ist das Werkzeugfenster „Fehlerliste“ neu hinzugekommen und zeigt zwei mögliche Ursachen für den Fehler in Zeile 13 des Programmcodes zu „Program.cs“.

Das Werkzeugfenster „Fehlerliste“ könnten Sie mit dem Fenster „Ausgabe“ im Registerkartenmodus am unteren Rand kombinieren. Ggf. sollten Sie die beiden sogar über die volle IDE-Breite einrichten (äußeres Führungssymbol am unteren Rand, nicht das untere im Führungssymbol-Kreuz), weil die Meldungen oft recht lang sind und deshalb eine möglichst breite Ansicht benötigen.

Unterhalb der Fehlerliste-Titelleiste sind drei Marken vorbereitet, die im Ausgangszustand nur schwach grau zu sehen sind. Sobald eine entsprechende Meldung ansteht, werden Fehler zum üblichen Icon „×“ auf rotem Kreis (so in Abb. 6.22), Warnungen mit „!“ auf gelbem Dreieck (so in Abb. 6.23) und einfache Meldungen mit „i“ im weißen Kreis signalisiert. Ergänzend gibt eine Zahl an, wie viele Fehler, Warnungen oder einfache Meldungen die IDE aus der aktuellen Projektmappe ermittelt.

Solange im Quellcode Fehler enthalten sind, ist eine Übersetzung und Ausführung dieses Programms nicht möglich. Sie sollten also einen Fehler möglichst immer gleich beseitigen und keinesfalls Fehler anhäufen lassen.

Anders sieht das bei **Warnungen** aus: Sie verweisen auf minder schwerwiegende Dinge, die die Übersetzung und Ausführung eines Programms nicht verhindern. Das betrifft meist Unsauberkeiten in der Programmierung. Warnungen können auch in noch unfertigem Code auftreten. Insofern ist es nicht immer sinnvoll oder möglich, ihre Ursache sofort zu beseitigen. Doch am Ende einer Entwicklung sollten auch die Ursachen von Warnungen aufgearbeitet sein.

Im Quellcode markiert eine grüne Unterkringelung die Quelle, in der „Fehlerliste“ werden Warnungen durch das gelbe Warndreieck gekennzeichnet. In Abb. 6.23 entsteht die Warnung, weil das mit `int abcdrf;` deklarierte Objekt nicht verwendet wird, also – bei diesem Stand der Entwicklung – unnötig wäre.

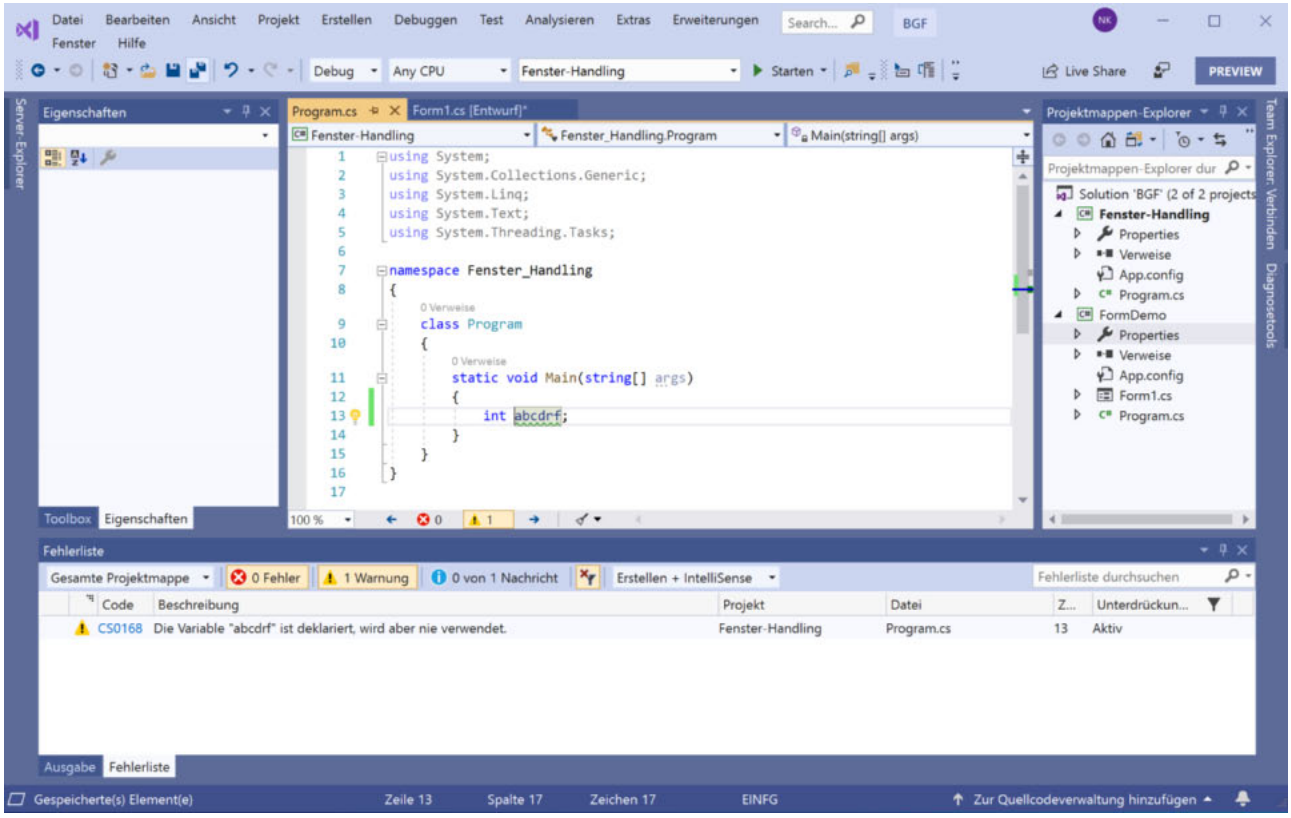


Abb. 6.23: Nach Änderung in der Quellcodezeile 13 (vgl. Abb. 6.22) verbleibt im Werkzeugfenster „Fehlerliste“ eine Warnung.

`int abcdrf;` nennt man eine „**Deklaration**“ oder auch Bekanntmachung, weil hier zum Typ „int“ (steht für ganze Zahlen) ein Bezeichner „abcd“ festgelegt wird, dem man im weiteren Verlauf irgendeine ganze Zahl zuweisen könnte. Damit ist übrigens der Fehler in Abb. 6.22 beseitigt: jetzt ist „bekanntgemacht“ (deklariert) dass „abcdrf“ der Name einer Variablen sein soll, die eine ganze Zahl aufnehmen kann. Da ihr aber noch keine ganze Zahl zugewiesen wurde, wird die Variable noch nicht verwendet, worauf die IDE mit einer Warnung aufmerksam macht. Wenn später die Codes umfangreicher werden, ist ein solcher Hinweis auf „Ruinen“ im Code oft recht nützlich.

6.3.5 Aufgabenliste

Wenn Sie dieses Werkzeugfenster aus dem Ansicht-Menü heraus aktivieren, springt es automatisch als Registerkarte zu den bereits vorhandenen Werkzeugfenstern „Fehlerliste“ und „Ausgabe“ und teilt somit deren Andockposition.

Dieses Werkzeugfenster wertet drei Typen von **Notizen** aus, die Sie im Quellcode anbringen können und in denen Sie Arbeitsaufgaben vermerken. Diese Notizen sind in **Kommentaren** unterzubringen. Eine Kommentarzeile wird durch einen Doppelslash eingeleitet. Damit legen Sie fest, dass diese Zeile nicht zum ausführbaren Programmcode gehören soll. Die IDE färbt den Text solcher Zeilen grün.

Im Rahmen solcher Kommentare sind Notizen, die vom Fenster „Aufgabenliste“ ausgewertet werden, möglich: **TODO**, **UNDONE** und **HACK**. Das Ganze funktioniert auch, wenn Sie diese Worte in Kleinbuchstaben schreiben, doch dann fallen sie weniger auf. Mit **Großbuchstaben** wird sofort deutlich, dass sie eine besondere Bedeutung haben. Was ist ihr Zweck?

Als Programmierer können Sie nicht sämtliche Aufgaben auf einmal erledigen. Also werden Sie sich immer auf das Wesentliche bzw. zunächst Wichtigste konzentrieren und andere Dinge zunächst einmal zurückstellen. Damit im Projekt nichts vergessen wird, können Sie noch zu erledigende Punkte und andere hervorhebenswerte Dinge im Programmcode kenntlich machen, sodass Sie leichter dorthin zurückfinden.

Das Werkzeugfenster „Aufgabenliste“ sammelt all diese Kenntlichmachungen übersichtlich geordnet. Mit einem Maus-Doppelklick auf einen der Einträge in der Aufgabenliste gelangen Sie unmittelbar in die zugehörige Programmcode-Zeile. Abb. 6.24 zeigt für TODO, UNDONE und HACK jeweils ein Beispiel:

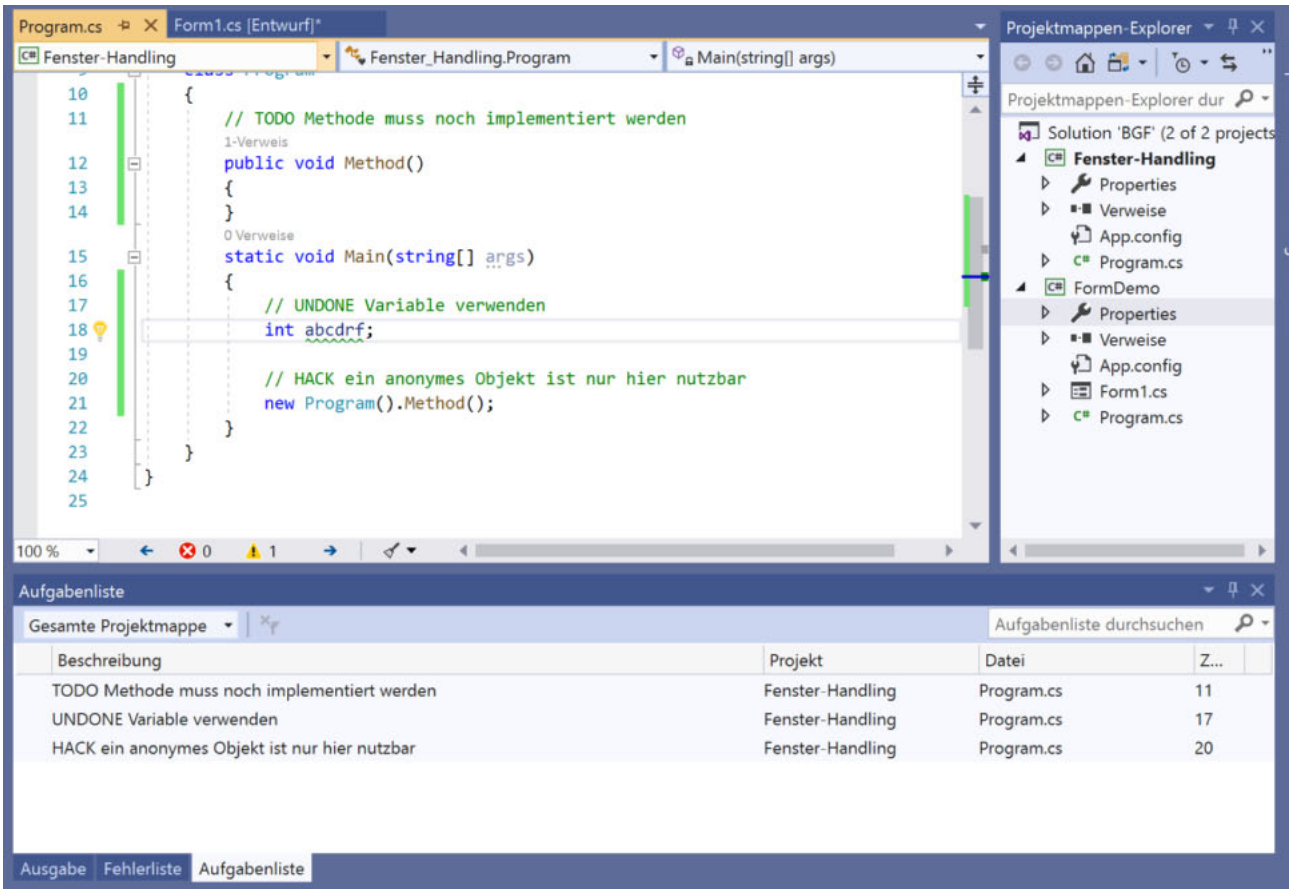


Abb. 6.24: Beispiele für TODO, UNDONE und HACK im Programmcode mit Auswertung im Werkzeugfenster „Aufgabenliste“

TODO

Hinter diesem Vermerk werden Sie eine Aufgabe beschreiben, die unbedingt noch bei der weiteren Projektvervollständigung erledigt werden muss.

Als Beispiel zeigt Abb. 6.24 die Kommentierung der Methode „Method“, in deren Programmblock noch jegliche Anweisung fehlt. Wir sprechen hier von einer „**leeren Implementierung**“. Entsprechend tut die Methode nichts, wenn sie aufgerufen wird. Das ist aber nicht Sinn von Programmierung, dass nichts passiert, also muss hier noch weitergearbeitet werden.

UNDONE

Das Merkwort ist mit „unerledigt“ am besten übersetzt. Sie können diesen Vermerk einsetzen, wenn die nötigen Codeergänzungen weniger dringlich sind, als die mit TODO bezeichneten Aufgaben.

In Abb. 6.24 ist die nicht verwendete Variable entsprechend kommentiert. Das wäre im Grunde nicht nötig, weil diese nicht verwendete Variable auch in der Fehlerliste als Warnung auftaucht.

HACK

Hinter dieser Kennung kann – je nach gemeintem Zusammenhang – Unterschiedliches notiert werden:

- Wenn getreu dem Programmierer-Jargon, dass ein Hack eine schnell dahingeworfene, etwas schlampige Programmierung darstellt, der Code noch verbessert werden soll,
- oder wenn mit „Hack“ eine besonders trickreiche, anspruchsvolle Programmierlösung gemeint ist und diese besonders hervorgehoben werden könnte.

Im Beispiel Abb. 6.24 ist mit „HACK“ ein besonders kompakter Methodenaufruf kommentiert – eine geeignete Erläuterung werden Sie später im Zuge des Lehrgangs wiederfinden.

Wie schon gesagt, sind die drei Aufgaben-Kennungen nicht case-sensitiv, können also groß wie klein geschrieben werden. Üblich ist die Großschreibung. Manche Programmierer setzt dahinter noch einen Doppelpunkt um die spezielle Funktion noch weiter hervorzuheben. Auch dies wird vom Werkzeugfenster „Fehlerliste“ verarbeitet.

Zusammenfassend:

```
// TODO: Vermerk für eine wichtige Aufgabe
// UNDONE: Vermerk für eine weniger wichtige Aufgabe
// HACK: schnell hingeworfene oder elegante Lösung
```

Es kann sein, dass Sie diese Ausführungen praktisch nachvollzogen und den Programmcode entsprechend ergänzt haben, ohne irgendeine Auswertung im Fenster „Aufgabenliste“ zu sehen.

Beachten Sie:

Die in Kommentare verpackten TODO-, UNDONE- und HACK-Vermerke werden vom Fenster „Aufgabenliste“ nur dann angezeigt, wenn Sie die nötige Voreinstellung vorgenommen haben.

6.4 Projektverwaltung

Eine Entwicklungsumgebung wie Visual Studio eignet sich weniger für Programmierzwecke der Art „mal eben etwas ausprobieren“. Weil jede Programmieraktivität von der IDE im Rahmen eines „Projekts“, und dies (optional) wiederum in einer Projektmappe verwaltet wird, sind vor der praktischen Codeerstellung wesentliche konzeptionelle Entscheidungen zu treffen.

6.4.1 Speicherort und Projektnamen

Die Vorüberlegungen beginnen mit der Frage, wo ein neues C#-Projekt im Dateisystem entstehen soll. Visual Studio schlägt per Voreinstellung einen **Speicherort** auf der Systempartition vor: *<Systemlaufwerk>:\Users\<Benutzer>\source\repos*. Diese Einstellung können Sie zum einen projektbezogen ändern, indem Sie für die Projektmappe im initialen Dialog „Neues Projekt“ einen anderen Pfad festlegen (so in Abb. 6.3). Die IDE merkt sich dann diesen Ort für nachfolgende Projekte.

Sie können den Speicherort aber auch in der Konfiguration der Entwicklungsumgebung abändern. Den zugehörigen Optionen-Dialog erreichen Sie (wie schon in Abschnitt 4.4) über „Extras > Optionen“. Unter der Kategorie „Projekte und Projektmappen“ sowie dort unter „Allgemein“ kann mit einem Klick auf eine der Schaltflächen mit drei Punkten ein anderer Speicherort für Projekte und/oder Vorlagen ausgewählt werden (Abb. 6.25).

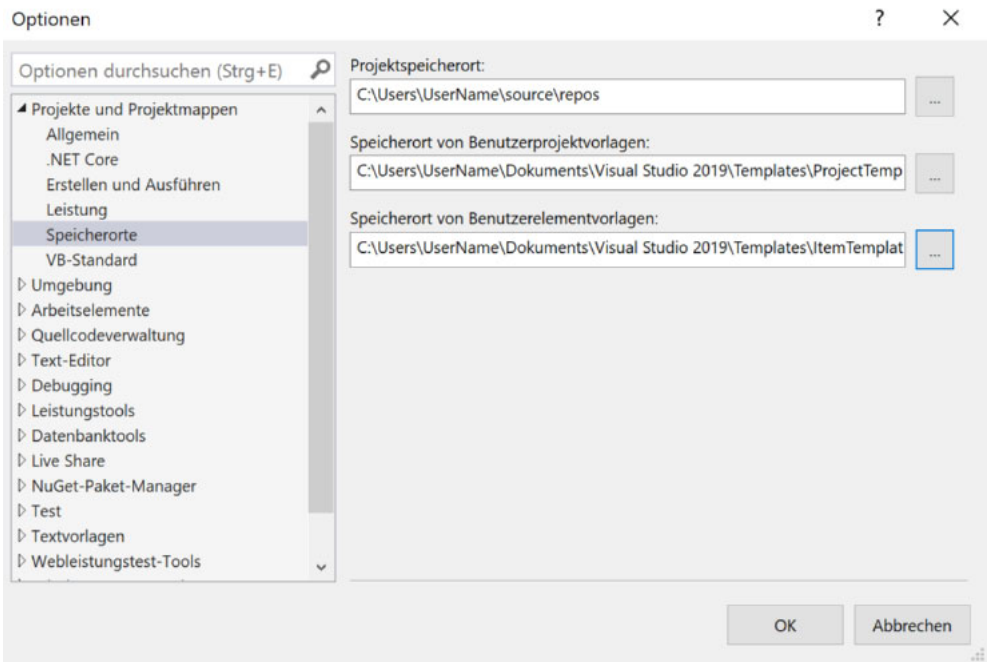


Abb. 6.25: Voreinstellungen zur Ablage von Projekten und Vorlagen nebst weiterer Speicheroptionen im IDE-Konfigurationsdialog (Extras > Optionen)

Statt einer Speicherung der Visual Studio-Projekte im Benutzerpfad, wie ihn der Dialog in Abb. 6.25 vorschlägt, bietet sich für diesen Lehrgang eher die Projektablage im Zusammenhang mit der Lehrgangsbearbeitung an. In Abschnitt 6.1 wurde bereits ein Vorschlag für ein passendes Dateisystem gemacht. Dann würde sich für „Projektspeicherort“ der Pfad *<Laufwerk>\Entwicklung\CSH-Lehrgang* anbieten. Das konkrete Studienheft kann dann ergänzend in die Maske des Dialogs „Neues Projekt“ als Unterordner ergänzt werden.

In diesem Dialog werden ferner der Projektmappenname sowie der Name des darin eingeschlossenen ersten Projekts festgelegt. Auch dazu hatte Abschnitt 6.1 bereits Kriterien genannt.

6.4.2 Änderung von Projekt- und Projektmappen-Namen

Sie sollten sich vor Neuanlage eines Projekts genau überlegen, wie Sie Ihr Projekt (und die zugehörige Projektmappe) nennen wollen. Die einmal gewählten Namen lassen sich nur in gewissen Grenzen ändern.

Der **Projektmappenname** lässt sich über das Kontextmenü zum Haupteintrag im Projektmappen-Explorer ändern (Abb. 6.26).

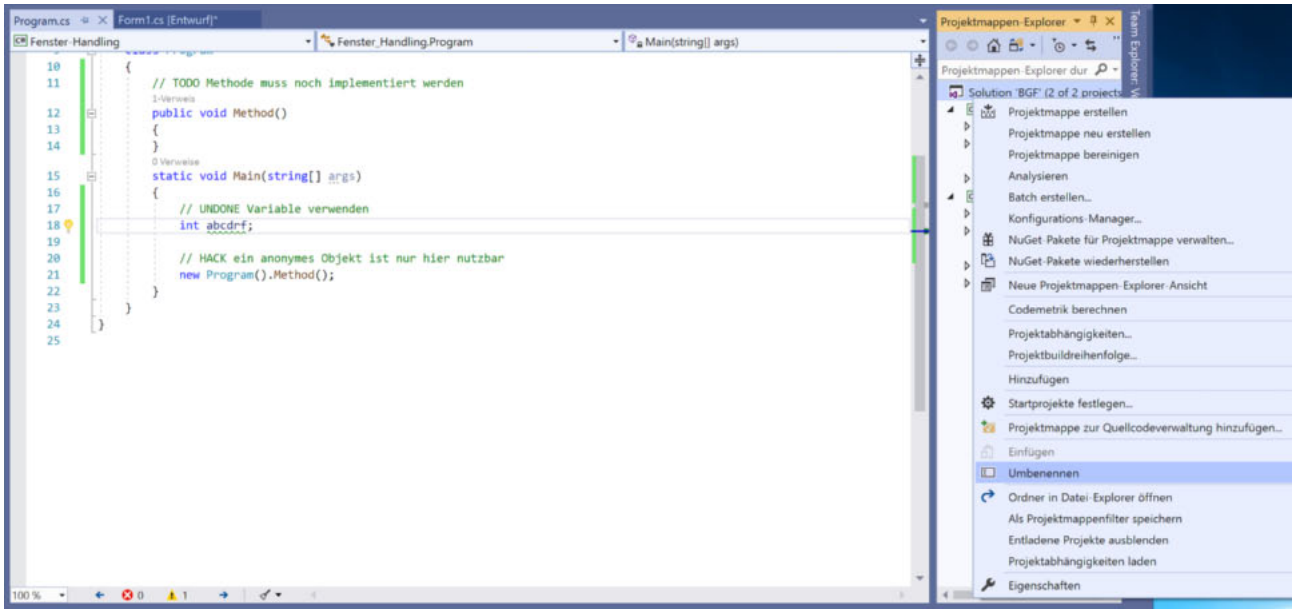


Abb. 6.26: Einstieg in die Umbenennung einer Projektmappe per Kontextmenü (rechte Maustaste) zur Projektmappe (im Projektmappen-Explorer) sowie den Eintrag „Umbenennen“

Wählen Sie dort „Umbenennen“, so wird die Beschriftung der Projektmappe in den Editiermodus geschaltet und Sie können einen neuen Namen eingeben. Diese Änderung betrifft aber ausschließlich den Namen, unter dem die Projektmappe innerhalb der IDE verwaltet wird; sie hat hingegen *keine* Auswirkungen auf den Ordnername im Dateisystem! Nach einer solchen Namensänderung sind somit der Projektmappenname und der Name des zugehörigen Ordners im Dateisystem nicht mehr identisch. Das ist ggf. verwirrend und sollte zwecks Klarheit in der Projektstruktur durch eine sorgfältige vorab-Namenswahl vermieden werden.

Ein **Projektname** kann ebenfalls per Kontextmenü im Projektmappen-Explorer und Auswahl von „Umbenennen“ geändert werden. Auch diese Namensänderung hat keine Auswirkungen auf den zugehörigen Ordnernamen im Dateisystem. Darüber hinaus wirkt sich eine Projektnamensänderung auch nicht auf den Bezeichner des Namensraums aus, der bei Projekterzeugung aus dem Projektnamen abgeleitet wurde. Letzteres lässt sich aber noch anpassen (siehe weiter unten unter „Refactoring“).

In beiden Fällen zeigt sich also, dass die Namensgebung vor dem Erstellen eines neuen Projekts gut überlegt werden sollte – sonst unterscheiden sich nach Umbenennungen die Projekt(mappen)namen innerhalb der IDE von denen im Dateisystem.

Die Speicherung einer kompletten Projektmappe unter einem neuen Namen wird von der IDE nicht angeboten. Ebensowenig bietet sie die Möglichkeit, ein komplettes Visual C#-Projekt aus dem Dateisystem zu löschen. Sie können ein Projekt (oder eine Projektmappe) über den Windows-Explorer löschen, wenn Visual Studio geschlossen ist. Wenn Sie dann die IDE erneut öffnen und das bereits gelöschte Projekt innerhalb der Entwicklungsumgebung laden wollen, meldet sie in einem Dialog (wie schon in Abb. 6.4 wiedergegeben)

<Projektmappe oder Projekt> konnte nicht geöffnet werden. Möchten Sie die Verweise aus den „Zuletzt geöffnet“-Listen entfernen?

Sie sagen „Ja“, und das Projekt ist auch in der IDE weg.

6.4.3 Refaktorisierung / Umgestaltung

Beim Erstellen eines neuen Projekts generiert Visual Studio gleich die Startklasse mit Main-Methode unter dem Namen „**Program**“ (englische Schreibweise, daher nur ein „m“; vgl. Abb. 6.6). Auch insofern gibt es Auswirkungen im Dateisystem: Die cs-Quellcodedatei trägt ebenfalls den Namen der Startklasse, heißt also „Program.cs“ (vgl. Abb. 6.5).

Dieser 08/15-Name „Programm“ muss nicht unbedingt bleiben – auch er kann in einen aussagekräftigeren Namen geändert werden, der den Zweck der Startklasse präziser und deutlicher bezeichnet. Diese Umbenennung läuft jedoch grundsätzlich anders als bei der Änderung von Projekt- bzw. Projektmappennamen. Berücksichtigen Sie bitte, dass ein Klassenname einen Typ bezeichnet, auf den es an ein oder mehreren Stellen in einem Programm Referenzen geben kann.

Beispiel 6.1:

In der .NET-Bibliothek gibt es die Klasse „System.String“ (Namensraum „System“, Klassenname „String“). Ein Objekt dieser Klasse lässt sich in irgendeiner anderen Klasse so definieren:

```
String string = "String";
```

Das Objekt mit dem Namen „string“ (kleiner Anfangsbuchstabe) soll vom Typ der Klasse „String“ (Großbuchstabe) sein. Ihm wird per Zuweisungsoperator „=“ die Zeichenkette „String“ zugewiesen (Zeichenketten vom Typ String erkennen Sie immer an den umschließenden Anführungszeichen). Zugleich ist diese Objektdefinition auch eine Referenz auf die Klasse „System.String“.

Wenn Sie nur den Namen des referenzierten Typs ändern, gehen alle Referenzen auf diesen Typ ins Leere und nichts funktioniert mehr. Sie könnten natürlich auch an allen referenzierenden Stellen den Typnamen ändern – doch zumindest in komplexeren Projekten würden Sie mit Sicherheit die eine oder andere Stelle übersehen. Im Übrigen wäre diese Verfahrensweise sehr mühsam.

Deshalb kennen moderne Entwicklungsumgebungen eine spezielle Technik der Umbenennung. Um diese Technik zu demonstrieren, ergänzen Sie in der Projektmappe „BGF“ (Abschnitt 6.1) ein weiteres Projekt. Dazu öffnen Sie erneut das Kontextmenü zur Projektmappe, wählen dort „Hinzufügen“ sowie im Untermenü „Neues Projekt“ (so schon im Abschnitt 6.3.2 praktiziert).

Es folgt wieder der Dialog zur Konfiguration eines neuen Projekts, der nun aber passgerecht „Neues Projekt hinzufügen“ heißt. Wir bleiben auch diesmal bei einer Konsolenanwendung und nennen das Projekt „Refactor“. Ergänzen Sie in der erneut „Program.cs“ genannten generierten Datei eine Methode „TuWas“ sowie die beiden Zeilen innerhalb der geschweiften Klammern zur Methode „Main“ gemäß Code 6.1 (Ergänzungen dort fett gestellt):

```
namespace Refactor
{ class Program {
  // Ergänzung einer Methode: void TuWas( )
  {
    Console.WriteLine( "Ich tu was" );
    }
    static void Main( string[] args )
    {
      // Erzeugung eines Objekts vom Typ "Program"
 Program programm = new Program();
      // Aufruf der Methode "TuWas":
 programm.TuWas();
    }
  }
}
```

Code 6.1: Beispielprogramm zum Refactoring

In diesem Code tritt der Klassenname „Program“ dreimal auf. Dieser Name soll an allen Stellen mit einer einzigen Aktion geändert werden. Weil auf diese und ggf. noch viel komplexere Weise Bezeichner nicht isoliert stehen, aber überall konsistent geändert werden müssen, bieten moderne Entwicklungsumgebungen eine passende Unterstützung. Im englischen heißt diese Technik **Refactoring** oder eingedeutscht „Refaktorisierung“. Microsoft übersetzt diesen Fachbegriff mit „**Umgestalten**“ ins Deutsche.

Diese Funktion wird im IDE-Menü „Bearbeiten“ angeboten (letzter Menüeintrag). Im Kontextmenü findet man seit Visual Studio nur noch den Befehl „Umbennenen…“, was die Arbeit erleichtert. Das Kontextmenü ist zweifellos praktischer, denn Sie müssen lediglich im Editor mit der rechten Maustaste auf den Namen einer Klasse (oder Methode usw.) klicken und erhalten das Umgestalten-Angebot zu diesem Bezeichner.

Abb. 6.27 zeigt diesen Vorgang am Beispiel des Klassennamens „Program“ und dem zugehörigen Kontextmenü:

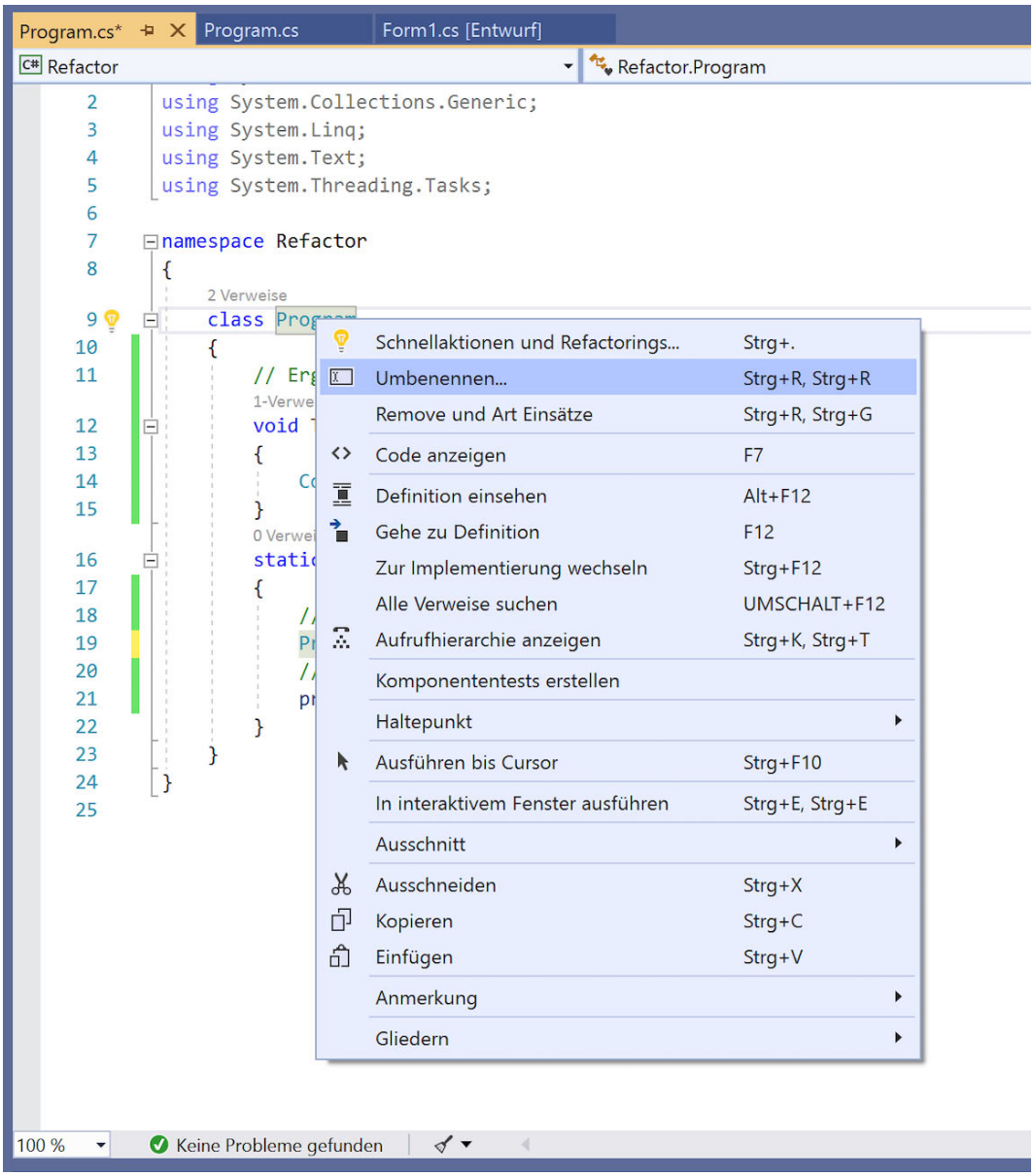


Abb. 6.27: Einstieg in die Umbenennung (Refactoring) des Klassennamens „Program“ über sein Kontextmenü

Nach Auswahl dieses Angebots kommt ein Dialog, in dessen Eingabefeld ein neuer Bezeichner eingetragen werden kann (Abb. 6.28):

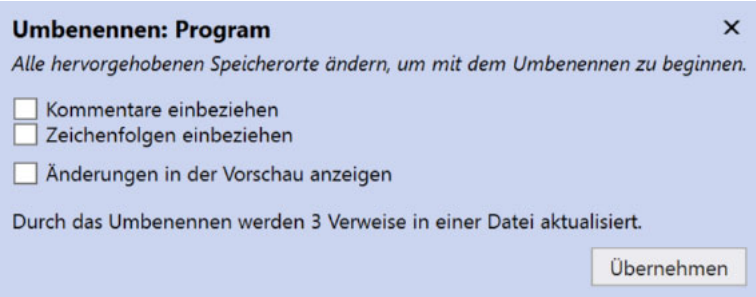


Abb. 6.28: Dialog zur Eingabe eines neuen Bezeichners – Vorschau auf die Änderungen aktiviert, Optionen zur Suche auch in Kommentaren bzw. in Strings.

Nach Eingabe eines neuen Namens (im Beispiel ist das „Writer“) und Bestätigung mit „OK“ können ggf. vorgesehene Umbenennungen über vorangestellte Checkboxen deaktiviert werden. Dies ist aber nur bei Änderungen in Kommentaren oder Strings sinnvoll, sofern diese Optionen aktiviert wurden. Denn eine Deaktivierung einer Änderung im Programmcode hätte die Inkonsistenz dieses Codes zur Folge.

Diese Refaktorisierungstechnik ist nicht nur auf Klassennamen sondern auch auf die Bezeichner von Klassenmitgliedern („Members“), also Variablen, Konstanten, Methoden etc. anwendbar. Mit dieser Technik kann auch ein Namensraum umbenannt werden.

Denken Sie immer daran, Namen von Namensräumen, Klassen, Methoden etc. nie direkt im Code, sondern immer über die Menüoption „Umgestalten – Umbenennen“ zu ändern, sodass mithilfe dieses IDE-Features alle Referenzen von der IDE angepasst werden können.

Refactoring / Umgestalten ist ein ungemein nützliches Werkzeug vor allem zur Verbesserung eines Programmcodes. Die Namen von Klassen, Methoden usw. sollten möglichst aussagekräftig, „sprechend“ sein. Eine geeignetere Bezeichnung fällt Ihnen vielleicht nicht gleich beim ersten Entwurf ein, später kommen Sie ggf. auf bessere Ideen … und dann ist es ungemein praktisch, den besseren Bezeichner mit einer einzigen Operation an allen Stellen auszutauschen, an denen er verwendet wird. Nutzen Sie also dies Feature in diesem Sinne zur Verbesserung Ihres Programmcodes.

Eine etwas unschöne Randbedingung gilt es allerdings noch zu betrachten:

Wenn Sie – wie eben beschrieben – die Klasse „Program“ im Namensraum „Refactor“ in „Writer“ umbenannt haben, heißt der zugehörige Dateiname, wie er im Projektmappen-Explorer angezeigt wird, immer noch „Program.cs“. Sie könnten nun in einer zweiten Operation auch dort umbenennen – nicht per Refactoring, sondern einfach mit der Kontextmenü-Funktion „Umbenennen“. Dann heißt der Typ und seine Datei wieder gleich. Diese Änderung wirkt sich auch auf das Dateisystem aus (da ja „Program.cs“ ein Dateiname ist).

Sie können aber auch beide Operationen zusammenfassen, wenn Sie die Umbenennung im Projektmappen-Explorer *beginnen*. Dies Verfahren greift aber nur dann, wenn der umzubenennende Typ (Klasse usw.) im Projektmappen-Explorer mit einer eigenen Datei vertreten ist und solange beide gleich heißen (was z. B. nach Generierung eines neuen Projekts für die Program-Klasse der Fall ist). Dann wird nach Umbenennung der Datei im Projektmappenexplorer der Dialog in Abb. 6.29 zwischengeschaltet:

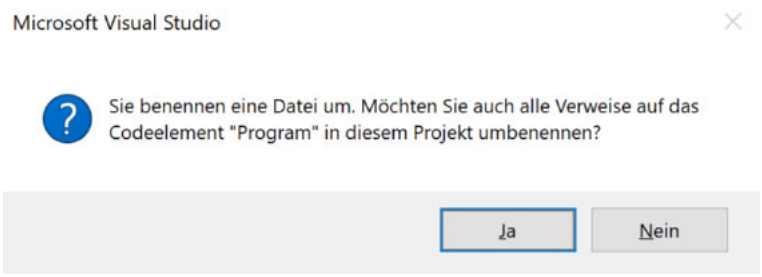


Abb. 6.29: Rückfrage nach Umbenennung einer Quellcode-Datei im Projektmappen-Explorer, zu der ein gleichnamiger Typ existiert

Bestätigen Sie mit „Ja“, so werden Dateiname und Typenname einheitlich umbenannt. Die Umbenennung des Typen (Klasse usw.) erfolgt zudem im Refactoring-Wege – d. h. es werden auch alle Referenzen umbenannt!

7 Wichtiges zur Kursorganisation

7.1 Lehrgangsdauer

Die Lehrgangsdauer beträgt 16 Monate bei einer wöchentlichen Arbeitszeit von ca. 10–12 Stunden. Sie können jederzeit mit dem Lehrgang beginnen und Sie allein entscheiden auch, ob Sie den Lehrgang in der vorgesehenen Studiendauer bzw. schneller oder langsamer bearbeiten möchten. Die Betreuungszeit beträgt 24 Monate, während der Sie Anspruch auf die Betreuungsleistungen der sgd ohne Mehrkosten haben.

7.2 Studienmaterialversand

Sie erhalten das Studienmaterial in vierteljährlichen Lieferungen. Beim Material der ersten Lieferung finden Sie einen Lieferplan, der die Bestandteile Ihres Kurses auflistet. Die horizontalen Trennlinien markieren jeweils das Studienmaterial einer Quartalssendung. Überprüfen Sie bitte anhand dieser Unterlage, ob die Sendung vollständig ist. Sollte etwas fehlen, wenden Sie sich bitte an Ihre Studienbetreuerin.

Die rechte Spalte listet die Kurzbezeichnungen der Studienmaterialien auf. Diese Kurzzeichen werden für unsere interne Logistik benutzt und lassen eine zweifelsfreie Identifikation zu. Haben Sie eine Frage zu einem bestimmten Heft, dann nennen Sie bitte Ihrer Studienbetreuerin bzw. Ihrer Fernlehrerin immer zuerst das Kürzel, das auch auf dem Umschlag des betreffenden Heftes abgedruckt ist.

Im sgd-OnlineCampus finden Sie ebenfalls eine Studienmaterialübersicht und die entsprechend zugeordneten Regelstudienmonate. Je nach Lehrgang stimmt dabei der Lieferplan nicht unbedingt mit der Studienmaterialübersicht im sgd-OnlineCampus überein. So sind z. B. Lernprogramme in der Regel im sgd-OnlineCampus nicht aufgeführt.

7.3 Das sgd-Betreuungsteam

Wir haben ein umfassendes Betreuungssystem für Sie eingerichtet. Dazu gehört:

- **Ihre Studienbetreuerin**

  Frau Alix Diehl
  Tel.: 06151 3842-732
  Mo–Do 8 bis 20 Uhr
  Fr 8 bis 18 Uhr
  E-Mail: alix.diehl@sgd.de

  In dem Begrüßungsbrief, der der Studienmaterialsendung beiliegt, stellt sich Frau Diehl bei Ihnen vor. Sie ist für die komplette Studienzeit Ihre Hauptansprechpartnerin und für alle grundsätzlichen Fragen zu Organisation und Ablauf des Kurses da. Sie beantwortet Ihnen Fragen wie beispielsweise:

  – Kann mein Material schneller geliefert werden?
  – Sind meine Einsendeaufgaben eingetroffen?
  – Kann meine Betreuungszeit verlängert werden?
  – …

- **Ihre Fernlehrerinnen und Fernlehrer (Tutoren)**

  Ihre Fernlehrer und Fernlehrerinnen (Tutoren) korrigieren, kommentieren und benoten Ihre Einsendeaufgaben. Da sie alle Spezialisten für einen ganz bestimmten Themenbereich aus Ihrem Lehrgang sind, werden Sie von mehreren Personen betreut. Sie werden sich Ihnen in einem Schreiben vorstellen. Vielleicht haben Sie Lust, auch über sich selbst zu erzählen, z.B. warum Sie gerade diesen Lehrgang belegt haben. Mit diesem Wissen im Hintergrund werden Ihre Fernlehrer und Fernlehrerinnen Ihre eingeschickten Arbeiten noch besser einordnen können. Wenn einer in Urlaub geht oder verhindert ist, organisieren wir eine Vertretung. Haben Sie Fragen zum Lernstoff, können Sie der zuständigen Fernlehrerin, dem zuständigen Fernlehrer schreiben. Sie können Ihre Frage einfach der Einsendeaufgabe beilegen. Benutzen Sie hierfür bitte die Vorlage aus dem Block „Fragen zum Lernstoff“. Oder Sie verwenden die Kontaktmöglichkeit über den sgd-OnlineCampus.

7.4 Umgang mit den Studienheften

Bevor Sie ein Studienheft genau lesen, verschaffen Sie sich zunächst einen **Überblick** darüber, welche Themen es behandelt. Hilfen dafür bieten das Inhaltsverzeichnis, die Einleitung, die Kapitelüberschriften, die Untertitel und im Text oder in der Marginalspalte Hervorgehobenes. Vielleicht ergeben sich dabei bereits Fragen an den Inhalt, Ihre Neugier wird geweckt.

Erst dann beginnen Sie bitte, den Text Satz für Satz, Abschnitt für Abschnitt aufmerksam zu lesen. Unterstreichen Sie wichtige Wörter, Teilsätze oder auch ganze Sätze. Verfahren Sie aber beim Unterstreichen sparsam, damit Sie bei späteren Wiederholungen mit einem Blick erfassen, welche wichtigen Informationen auf der betreffenden Seite stehen. Wenn Sie zu viel und zu bunt unterstreichen, verlieren Sie leicht den Überblick. Nutzen Sie auch unbedingt den breiten Rand, um sich Notizen zu machen bzw. um Fragen festzuhalten. Manches wird sich später von selbst im Zusammenhang klären. In jedem Fall aber halten diese schriftlichen Tätigkeiten Sie an mitzudenken, und helfen Ihnen dabei, sich das Gelesene einzuprägen.

Der Lernstoff ist durch Aufgaben zur Selbstüberprüfung in „Lernportionen“ unterteilt. Lösen Sie diese Aufgaben, bevor Sie weiterlesen. Damit können Sie überprüfen, ob Sie das, was Sie gerade gelesen haben, auch verstanden haben und umsetzen können. Im Anhang des Studienhefts befinden sich dazu Musterlösungen. Wenn Sie etwas nicht wussten oder Ihre Antworten sehr abweichen, überarbeiten Sie sie bitte und ergänzen Sie auch unvollständige Lösungen. Gehen Sie erst weiter, wenn Sie die Aufgaben zufriedenstellend gelöst haben. Sie zeigen Ihnen, ob Ihre Kenntnisse ausreichend sind, um den nächsten Abschnitt in Angriff zu nehmen.

7.5 Die Einsendeaufgaben

Häufig erhalten wir Fragen zur Bearbeitung von Einsendeaufgaben. Deshalb möchten wir Ihnen einige nützliche Hinweise dazu geben. Wir hoffen, sie erleichtern Ihnen die Bearbeitung Ihrer Aufgaben. Bei weiteren Fragen können Sie sich gern an unsere Studienbetreuung wenden.

Fernunterrichtschutzgesetz

Das Fernunterrichtschutzgesetz (FernUSG), das zum Schutz der Kunden und ihres Lernerfolgs erlassen wurde, regelt in § 2 die Lernerfolgskontrolle als Pflicht eines Fernunterrichtsanbieters. Die sgd hält sich nicht nur an dieses Gesetz, wir haben die geforderten Lernerfolgskontrollen zu Aufgabentypen weiterentwickelt, die Ihnen Hilfestellung für Ihre Erfolgskontrollen im Lernprozess geben.

Aufgabentypen

Ihre Studienmaterialien enthalten sogenannte Selbstkontroll- oder auch Wiederholungsaufgaben sowie Einsendeaufgaben (ESA). Die Selbstkontroll- bzw. Wiederholungsaufgaben kontrollieren Sie nach der Bearbeitung eigenständig mit den vorgegebenen Lösungen im jeweiligen Anhang eines Studienhefts. Die selbstständig von Ihnen bearbeitete ESA schicken Sie an die sgd. Wir leiten Ihre Aufgaben an die zuständige Fernlehrerin, an den zuständigen Fernlehrer zur Korrektur weiter. Der Fernlehrer benotet die Aufgabe und gibt Ihnen Rückmeldung zu Ihrem Lernerfolg. Mit diesen beiden Aufgabentypen wird gewährleistet, dass Sie Ihre Leistungen und Ihren Lernfortschritt realistisch einschätzen können.

Bearbeitung der Einsendeaufgaben

Es gibt keine zwingende zeitliche Vorgabe für die Bearbeitung Ihrer ESA. Sie können sie handschriftlich oder am Computer verfassen. Wir empfehlen Ihnen eine regelmäßige Bearbeitung, damit Sie den Lernstoff stetig wiederholen und vertiefen. Eine Rückmeldung Ihres Fernlehrers wird Ihnen eine Bestätigung geben oder Sie dazu motivieren, Ihre Fertigkeiten gezielt zu vervollkommnen. Ihnen steht die gesamte Betreuungszeit als Zeitrahmen zur Verfügung. Ihr Folgematerial wird unabhängig vom Stand der Aufgabenbearbeitung geliefert.

Den besten Erfolg erzielen Sie, wenn Sie die Einsendeaufgaben in mehreren Schritten bearbeiten:

- Lesen Sie zunächst die komplette Aufgabenstellung durch.
- Lesen Sie danach nochmals alles, was Sie im Heft farbig markiert haben.
- Überlegen Sie in Ruhe, wie der Lösungsweg aussehen soll.
- Skizzieren Sie anschließend Ihren Lösungsentwurf mit möglichst wenigen Stichworten.
- Je klarer Sie die Lösung einer Aufgabe „vor Augen haben“, desto zielstrebiger können Sie die Lösung formulieren – und desto weniger Zeit benötigen Sie für den Lösungsentwurf, die Korrekturen und die endgültige Ausarbeitung.

Zwei Versandwege

Sie können die Lösungen Ihrer ESA per Post oder in unserem sgd-OnlineCampus als Anhang per E-Mail an die Fernlehrer senden. Schicken Sie die Aufgaben per Post, machen Sie sich bitte eine Kopie Ihrer Lösungen für Ihre Unterlagen.

Oben auf der Einsendeaufgabe finden Sie ein Adressfeld. **Bitte füllen Sie es vollständig aus**. Tragen Sie in die vorbereiteten Felder des Adressfeldes Ihren Namen, den Vornamen, die Postleitzahl, den Ort sowie die Straße ein. Tragen Sie unbedingt auch Ihre Studiennummer sowie die Lehrgangsnummer ein. Wir leiten, wie bereits erwähnt, die Aufgaben an die entsprechenden Fernlehrer weiter.

Bitte verschicken Sie **nicht mehrere Einsendeaufgaben gleichzeitig**. Es ist in Ihrem eigenen Interesse, das Feedback Ihres Fernlehrers zu einer Arbeit abzuwarten, bevor Sie die nächste einsenden.

Die Noten der per Post eingegangenen Aufgaben werden ebenfalls in den sgd-Campus übertragen. Der aktuelle Notenstand ist in der Regel bis spätestens zum Anfang des Folgemonats eingetragen und für Sie einsehbar.

Die Korrekturzeit für Ihre ESA beträgt in der Regel eine Woche, hinzu kommt die Zeit für den Versand. Korrekturen über den sgd-Campus erhalten Sie schneller zurück, da per E-Mail die Zeit für den Postweg entfällt.

Selbstständiges Arbeiten

Die Lernerfolgskontrolle ist nur sinnvoll und nützlich für Sie, wenn Sie die ESA selbstständig bearbeiten, d.h. Ihre eigene Leistung zeigen. Dadurch sichern Sie sich Ihren Lernfortschritt und Ihren Lernerfolg. Sie erhalten dabei Unterstützung durch Ihre Fernlehrer.

Bei der Bearbeitung der ESA ist es grundsätzlich möglich und zum Teil auch sinnvoll, Gedanken anderer in die eigenen Ausführungen einfließen zu lassen. Die Verwendung des fremden „geistigen Eigentums“ müssen Sie allerdings durch Zitate kennzeichnen. Weiter unten finden Sie die entsprechenden **Regeln zum Zitieren**, die Sie bei dem richtigen Einsatz von Zitaten unterstützen.

Durch entsprechende Angebote im Internet kommt es leider immer häufiger vor, dass Lösungen von ESA komplett kopiert oder abgeschrieben und als eigene Arbeit abgegeben werden. Damit verlieren die ESA aber ihren Nutzen und Ihr Fernlehrer kann weder Ihren Lernfortschritt realistisch beurteilen noch Sie gezielt unterstützen. Darüber hinaus haben die vielen gewissenhaften Kursteilnehmer, die die ESA eigenständig bearbeiten, ein Recht auf faire Bewertung. Ist daher ein Fall kopierter oder erworbener Lösungen nachzuweisen, kann er als Täuschungsversuch betrachtet und mit der Note „ungenügend“ bewertet werden.

Denken Sie bitte daran: Wir unterstützen Sie gern bei der Bearbeitung von eigenständigen Lösungen für Ihre ESA. Sie werden bald feststellen, dass Sie das weiterbringt.

Wichtiger Hinweis

Damit Sie weiterhin einen anerkannten und geschätzten Abschluss bei der sgd erhalten, haben wir eine Bitte an Sie: Stellen Sie **nicht** Ihre komplett ausgearbeiteten Lösungen der Einsendeaufgaben anderen Teilnehmer(inne)n zur Verfügung. Die Auswirkungen dieser Vorgehensweise sind letztlich für niemanden von Vorteil.

Eine gegenseitige Hilfestellung bei der Bearbeitung der Studienhefte ist selbstverständlich wünschenswert und soll weiterhin stattfinden. Unser sgd-OnlineCampus unterstützt Sie dabei.

Regeln für das Zitieren, Beachtung des Copyrights ©

Wenn Sie bei der Beantwortung einer Einsendeaufgabe geistiges Eigentum anderer Autoren mit heranziehen, ist dies im Prinzip zulässig. Sie müssen aber unbedingt die **Verwendung fremder Gedanken** kenntlich machen – egal ob es sich um Quellen aus dem Internet, aus Zeitschriften oder Büchern handelt. Wir können Ihre Eigenleistung in der Einsendeaufgabe nur dann bewerten, wenn deutlich wird, welches Ihr Anteil ist und was Sie aus anderen Quellen einbringen.

Wir zeigen Ihnen im Folgenden, wie Sie fremde Quellen kennzeichnen.

- **Wie können Sie fremde Quellen in den eigenen Text einbringen?**

  Hier gibt es zwei Möglichkeiten:

  a) Wenn Sie ein Zitat **wörtlich** übernehmen, markieren Sie dies durch Anführungszeichen. Auslassungen aus dem Zitat zeigen Sie durch drei Punkte in eckigen Klammern. Nach dem Zitat geben Sie die Quelle in Klammern an:

  > „Unter der Netiquette […] versteht man das gute oder angemessene und achtende (respektvolle) Benehmen in der technischen (elektronischen) Kommunikation. Der Begriff beschrieb ursprünglich Verhaltensempfehlungen im Usenet, er wird aber mittlerweile für alle Bereiche in Datennetzen verwendet.“ (http://de.wikipedia.org/wiki/Netiquette [Stand: 24.03.2020])

  b) Wenn Sie einen Gedanken nur aufgreifen und **umschreiben**, aber **nicht wörtlich** zitieren, machen Sie dies durch ein paar einführende Worte deutlich. Auch hier geben Sie die Quelle in Klammern an:

  > Laut Schulz von Thun haben fast alle Nachrichten die Funktion, auf den Empfänger Einfluss zu nehmen (Schulz von Thun, Friedemann: Miteinander reden, Band 1, Rowohlt Verlag, Reinbek, 2003, Seite 29).

  Oder:

  > Schulz von Thun behauptet, dass fast alle Nachrichten die Funktion haben, auf den Empfänger Einfluss zu nehmen (Schulz von Thun, Friedemann: Miteinander reden, Band 1, Rowohlt Verlag, Reinbek, 2003, Seite 29).

  Wenn Sie dieselbe Quelle noch einmal verwenden, müssen Sie natürlich nicht den gesamten Titel noch einmal angeben. Schreiben Sie dann einfach: (Schulz von Thun, S. 37).

- **Wie gestalten Sie eine Literaturangabe?**

  Die Literaturangabe muss so gestaltet sein, dass Ihr Fernlehrer die Quelle ausfindig machen und das Zitat nachvollziehen kann. Dazu sind notwendig:

  a) **bei Büchern**:

  Autor, Titel, Verlag/Verlagsort, Jahr, Seite

  Beispiel:

  Birkenbihl, Vera F.: Kommunikationstraining, Moderne Verlagsgesellschaft, Landsberg am Lech, 2007, S. 35

  b) **bei Büchern, die nicht vom Urheber des Zitats herausgegeben sind (Aufsatzsammlungen etc.)**:

  Autor des Zitats, Titel des Aufsatzes, Herausgeber, Buchtitel, Verlag/Verlagsort, Jahr, Seite

  Beispiel:

  Weinkopf, Claudia: Qualifizierung in der vermittlungsorientierten Arbeitnehmerüberlassung. In: Münchhausen, Gesa (Hrsg.): Kompetenzentwicklung in der Zeitarbeit – Potenziale und Grenzen. Bundesinstitut für Berufsbildung, Bonn 2007, S. 45-54

  c) **bei Zeitschriften**:

  Autor des Zitats, Titel des Aufsatzes, Zeitschriftentitel, Erscheinungsdatum, Seite

  Beispiel:

  Gnahs, Dieter: Übergänge als Testfall. Das Konzept des lebenslangen Lernens und die Durchlässigkeit des Bildungssystems. In: DIE Zeitschrift für Erwachsenenbildung, 2007/1, S. 28-31

  d) **bei Texten aus dem Internet**:

  Autor (wenn möglich), URL, Abrufdatum

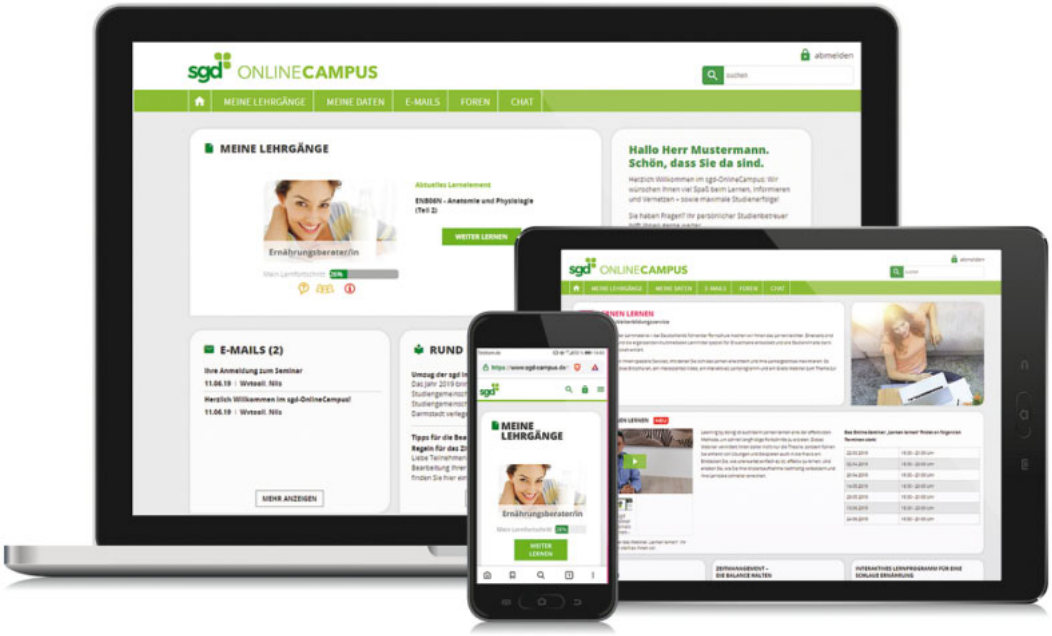
  Beispiel:

  Sorge, Nils-Viktor: Geldregen für die Klima-Gewinner. http://www.spiegel.de/wirtschaft/oekostrom-geldregen-fuer-die-klima-gewinner-a-483869.html [Stand: 24.03.2020]

7.6 Der Lehrgangsabschluss

Wenn Sie alle Einsendeaufgaben der Studienhefte zur Leistungskontrolle erfolgreich bearbeitet haben, erhalten Sie automatisch Ihr **sgd-Abschlusszeugnis**, welches die Bearbeitung der Hausarbeiten nachweist.

8 Der sgd-OnlineCampus



Sie haben während Ihrer gesamten Betreuungszeit Zugang zum sgd-OnlineCampus. Dadurch studieren Sie so komfortabel und effektiv wie nur möglich. Denn im sgd-Campus dreht sich alles um einfache Kontaktaufnahme, aktuelle Informationen und moderne Arbeitsformen.

Ihr persönliches Passwort und die zugehörige Benutzerkennung finden Sie in Ihrer Dokumentenmappe, die der Erstsendung beiliegt.

Der sgd-Campus lässt sich vielseitig nutzen, z. B.

- um Ihre Lösungen zu Einsendeaufgaben einzureichen,
- um mit Fernlehrern in Kontakt zu treten oder sich mit anderen Lehrgangsteilnehmern auszutauschen.

Außerdem finden Sie zu vielen Studienmaterialien zusätzliche Informationen oder auch Aktualisierungen. Besuchen Sie einfach einmal den sgd-Campus und wählen Sie den **Wegweiser** aus. Sie erhalten hier viele Informationen und auch kurze Videosequenzen, die Ihnen den Umgang mit dem sgd-Campus näher erläutern.