

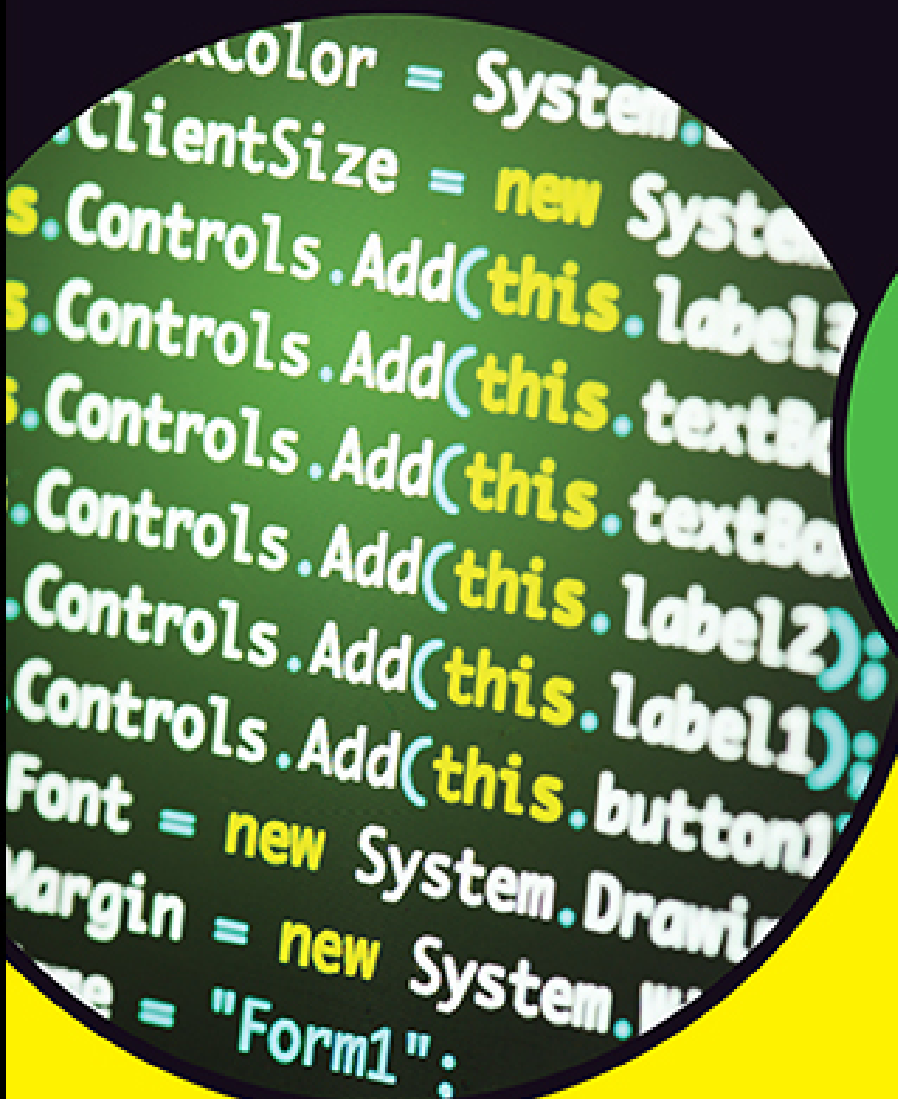
LEARNING MADE EASY



C# 10.0

ALL-IN-ONE

for
dummies[®]
A Wiley Brand



John Paul Mueller
Has used C# to create a vacation
countdown timer



C# 10.0

ALL-IN-ONE

by John Paul Mueller

for
dummies
A Wiley Brand

C# 10.0 All-in-One For Dummies®

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2022 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

<p>LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: WHILE THE PUBLISHER AND AUTHORS HAVE USED THEIR BEST EFFORTS IN PREPARING THIS WORK, THEY MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED</p>
--

WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES, WRITTEN SALES MATERIALS OR PROMOTIONAL STATEMENTS FOR THIS WORK. THE FACT THAT AN ORGANIZATION, WEBSITE, OR PRODUCT IS REFERRED TO IN THIS WORK AS A CITATION AND/OR POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE PUBLISHER AND AUTHORS ENDORSE THE INFORMATION OR SERVICES THE ORGANIZATION, WEBSITE, OR PRODUCT MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING PROFESSIONAL SERVICES. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR YOUR SITUATION. YOU SHOULD CONSULT WITH A SPECIALIST WHERE APPROPRIATE. FURTHER, READERS SHOULD BE AWARE THAT WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. NEITHER THE PUBLISHER NOR AUTHORS SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included

with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2021951074

ISBN: 978-1-11-983907-1 (pbk)

ISBN 978-1-11-984012-1 (ebk); ISBN 978-1-11-983908-8 (ebk)

C# 10.0 All-in-One For Dummies®

To view this book's Cheat Sheet, simply go to www.dummies.com and search for “C# 10.0 All-in-One For Dummies Cheat Sheet” in the Search box.

Table of Contents

[Cover](#)

[Title Page](#)

[Copyright](#)

[Introduction](#)

[About This Book](#)

[Foolish Assumptions](#)

[Icons Used in This Book](#)

[Beyond the Book](#)

[Where to Go from Here](#)

[**Book 1: The Basics of C# Programming**](#)

[**Chapter 1: Creating Your First C# Console Application**](#)

[Getting a Handle on Computer Languages, C#, and .NET](#)

[Creating Your First Console Application](#)

[Making Your Console App Do Something](#)

[Reviewing Your Console Application](#)

[Replacing All that Ceremonial Code: Top-Level Statements](#)

[Introducing the Toolbox Trick](#)

[Interacting with C# Online](#)

[Working with Jupyter Notebook: The Short Version](#)

Chapter 2: Living with Variability — Declaring Value-Type Variables

[Declaring a Variable](#)

[What's an int?](#)

[Representing Fractions](#)

[Handling Floating-Point Variables](#)

[Using the Decimal Type: Is It an Integer or a Float?](#)

[Examining the bool Type: Is It Logical?](#)

[Checking Out Character Types](#)

[What's a Value Type?](#)

[Comparing string and char](#)

[Calculating Leap Years: DateTime](#)

[Declaring Numeric Constants](#)

[Changing Types: The Cast](#)

[Letting the C# Compiler Infer Data Types](#)

Chapter 3: Pulling Strings

[The Union Is Indivisible, and So Are Strings](#)

[Performing Common Operations on a String](#)

[Comparing Strings](#)

[What If I Want to Switch Case?](#)

[Looping through a String](#)

[Searching Strings](#)

[Getting Input from Users in Console Applications](#)

[Controlling Output Manually](#)

[Formatting Your Strings Precisely](#)

[StringBuilder: Manipulating Strings More Efficiently](#)

Chapter 4: Smooth Operators

[Performing Arithmetic](#)

[Performing Logical Comparisons — Is That Logical?](#)
[Matching Expression Types at TrackDownAMate.com](#)

Chapter 5: Getting into the Program Flow

[Branching Out with if and switch](#)

[Here We Go Loop-the-Loop](#)

[Looping a Specified Number of Times with for](#)

Chapter 6: Lining Up Your Ducks with Collections

[The C# Array](#)

[Processing Arrays by Using foreach](#)

[Sorting Arrays of Data](#)

[Using var for Arrays](#)

[Loosening Up with C# Collections](#)

[Understanding Collection Syntax](#)

[Using Lists](#)

[Using Dictionaries](#)

[Array and Collection Initializers](#)

[Using Sets](#)

Chapter 7: Stepping through Collections

[Iterating through a Directory of Files](#)

[Iterating foreach Collections: Iterators](#)

[Accessing Collections the Array Way: Indexers](#)

[Looping Around the Iterator Block](#)

Chapter 8: Buying Generic

[Writing a New Prescription: Generics](#)

[Classy Generics: Writing Your Own](#)

[Understanding Variance in Generics](#)

Chapter 9: Some Exceptional Exceptions

[Using an Exceptional Error-Reporting Mechanism](#)

[Can I Get an Exceptional Example?](#)

[Working with Custom Exceptions](#)

[Planning Your Exception-Handling Strategy](#)

[Grabbing Your Last Chance to Catch an Exception](#)
[Throwing Expressions](#)

Chapter 10: Creating Lists of Items with Enumerations

[Seeing Enumerations in the Real World](#)
[Working with Enumerations](#)
[Creating Enumerated Flags](#)
[Defining Enumerated Switches](#)
[Working with Enumeration Methods](#)

Book 2: Object-Oriented C# Programming

Chapter 1: Showing Some Class

[A Quick Overview of Object-Oriented Programming](#)
[Defining a Class and an Object](#)
[Accessing the Members of an Object](#)
[Working with Object-Based Code](#)
[Discriminating between Objects](#)
[Can You Give Me References?](#)
[Classes That Contain Classes Are the Happiest Classes in the World](#)
[Generating Static in Class Members](#)
[Defining const and readonly Data Members](#)

Chapter 2: We Have Our Methods

[Defining and Using a Method](#)
[Method Examples for Your Files](#)
[Having Arguments with Methods](#)
[Using the Call-by-Reference Feature](#)
[Defining a Method with No Return Value](#)
[Returning Multiple Values Using Tuples](#)

Chapter 3: Let Me Say This about this

[Passing an Object to a Method](#)
[Comparing Static and Instance Methods](#)
[Accessing the Current Object](#)

[Using Local Functions](#)

Chapter 4: Holding a Class Responsible

[Restricting Access to Class Members](#)

[Why You Should Worry about Access Control](#)

[Defining Class Properties](#)

[Using Target Typing for Your Convenience](#)

[Dealing with Covariant Return Types](#)

[Getting Your Objects Off to a Good Start — Constructors](#)

[Using Expression-Bodied Members](#)

Chapter 5: Inheritance: Is That All I Get?

[Why You Need Inheritance](#)

[Inheriting from a BankAccount Class \(a More Complex Example\)](#)

[IS_A versus HAS_A — I'm So Confused_A](#)

[Other Features That Support Inheritance](#)

Chapter 6: Poly-what-ism?

[Overloading an Inherited Method](#)

[Polymorphism](#)

[C# During Its Abstract Period](#)

[Sealing a Class](#)

Chapter 7: Interfacing with the Interface

[Introducing CAN_BE_USED_AS](#)

[Knowing What an Interface Is](#)

[Using an Interface](#)

[Using the C# Predefined Interface Types](#)

[Looking at a Program That CAN_BE_USED_AS an Example](#)

[Unifying Class Hierarchies](#)

[Hiding Behind an Interface](#)

[Inheriting an Interface](#)

[Using Interfaces to Manage Change in Object-Oriented Programs](#)

Chapter 8: Delegating Those Important Events

[E.T., Phone Home — The Callback Problem](#)

[Defining a Delegate](#)

[Pass Me the Code, Please — Examples](#)

[A More Real-World Example](#)

[Shh! Keep It Quiet — Anonymous Methods](#)

[Stuff Happens — C# Events](#)

Chapter 9: Can I Use Your Namespace in the Library?

[Dividing a Single Program into Multiple Source Files](#)

[Working with Global using Statements](#)

[Dividing a Single Program into Multiple Assemblies](#)

[Putting Your Classes into Class Libraries](#)

[Going Beyond Public and Private: More Access Keywords](#)

[Putting Classes into Namespaces](#)

[Working with Partial Methods](#)

Chapter 10: Improving Productivity with Named and Optional Parameters

[Exploring Optional Parameters](#)

[Looking at Named Parameters](#)

[Using Alternative Methods to Return Values](#)

[Dealing with null Parameters](#)

Chapter 11: Interacting with Structures

[Comparing Structures to Classes](#)

[Creating Structures](#)

[Working with Read-only Structures](#)

[Working with Reference Structures](#)

[Using Structures as Records](#)

[Using the New Record Type](#)

Book 3: Designing for C#

Chapter 1: Writing Secure Code

[Designing Secure Software](#)

[Building Secure Windows Applications](#)

[Using System.Security](#)

Chapter 2: Accessing Data

[Getting to Know System.Data](#)

[How the Data Classes Fit into the Framework](#)

[Getting to Your Data](#)

[Using the System.Data Namespace](#)

Chapter 3: Fishing the File Stream

[Going Where the Fish Are: The File Stream](#)

[StreamWriting for Old Walter](#)

[Pulling Them Out of the Stream: Using StreamReader](#)

[More Readers and Writers](#)

[Exploring More Streams than Lewis and Clark](#)

Chapter 4: Accessing the Internet

[Getting to Know System.Net](#)

[How Net Classes Fit into the Framework](#)

[Using the System.Net Namespace](#)

Chapter 5: Creating Images

[Getting to Know System.Drawing](#)

[How the Drawing Classes Fit into the Framework](#)

[Using the System.Drawing Namespace](#)

Chapter 6: Programming Dynamically!

[Shifting C# Toward Dynamic Typing](#)

[Employing Dynamic Programming Techniques](#)

[Putting Dynamic to Use](#)

[Running with the Dynamic Language Runtime](#)

[Using Static Anonymous Functions](#)

Book 4: A Tour of Visual Studio

Chapter 1: Getting Started with Visual Studio

[Versioning the Versions](#)

[Installing Visual Studio](#)

[Breaking Down the Projects](#)

Chapter 2: Using the Interface

[Designing in the Designer](#)

[Paneling the Studio](#)

[Coding in the Code Editor](#)

[Using the Tools of the Trade](#)

[Using the Debugger as an Aid to Learning](#)

Chapter 3: Customizing Visual Studio

[Setting Options](#)

[Creating Your Own Templates](#)

Book 5: Windows Development with WPF

Chapter 1: Introducing WPF

[Understanding What WPF Can Do](#)

[Introducing XAML](#)

[Diving In! Creating Your First WPF Application](#)

[Whatever XAML Can Do, C# Can Do Better!](#)

Chapter 2: Understanding the Basics of WPF

[Using WPF to Lay Out Your Application](#)

[Arranging Elements with Layout Panels](#)

[Exploring Common XAML Controls](#)

Chapter 3: Data Binding in WPF

[Getting to Know Dependency Properties](#)

[Exploring the Binding Modes](#)

[Investigating the Binding Object](#)

[Editing, Validating, Converting, and Visualizing Your Data](#)

[Finding Out More about WPF Data Binding](#)

Chapter 4: Practical WPF

[Commanding Attention](#)

[Using Built-In Commands](#)

[Using Custom Commands](#)

[Using Routed Commands](#)

Chapter 5: Programming for Windows 10 and Above

[What is the Universal Windows Platform \(UWP\)?](#)

[Devices Supported by the UWP](#)

[Creating Your Own UWP App](#)

[Working with .NET Core Applications](#)

Book 6: Web Development with ASP.NET

Chapter 1: Creating a Basic ASP.NET Core App

[Understanding the ASP.NET Core Templates](#)

[Developing a Basic Web App](#)

Chapter 2: Employing the Razor Markup Language

[Avoiding Nicks from Razor](#)

[Creating Variables](#)

[Keeping Things Logical](#)

[Implementing Loops](#)

Chapter 3: Generating and Consuming Data

[Understanding Why These Projects Are Important](#)

[Serialized Data Isn't for Breakfast](#)

[Developing a Data Generator and API](#)

[Creating a Consumer Website](#)

Index

About the Author

Advertisement Page

Connect with Dummies

End User License Agreement

List of Tables

Book 1 Chapter 2

[TABLE 2-1 Size and Range of C# Integer Types](#)

[TABLE 2-2 Size and Range of Floating-Point Variable Types](#)

[TABLE 2-3 Special Characters](#)

[TABLE 2-4 Common Constants Declared along with Their Types](#)

Book 1 Chapter 3

[TABLE 3-1 Format Specifiers Using String.Format\(\)](#)

Book 1 Chapter 4

[TABLE 4-1 Simple Operators](#)

[TABLE 4-2 Logical Comparison Operators](#)

[TABLE 4-3 The Compound Logical Operators](#)

Book 1 Chapter 6

[TABLE 6-1 The Most Common Collection “Shapes”](#)

Book 1 Chapter 8

[TABLE 8-1 Generic Constraint Options](#)

Book 3 Chapter 1

[TABLE 1-1 Common Namespaces in System.Security](#)

Book 3 Chapter 2

[TABLE 2-1 The System.Data Namespaces](#)

Book 3 Chapter 4

[TABLE 4-1 A Listing of Important System.Net-Associated Namespaces](#)

List of Illustrations

Book 1 Chapter 1

[FIGURE 1-1: Creating a new project starts you down the road to a better Windows...](#)

[FIGURE 1-2: The Visual Studio App Wizard is eager to create a new program for y...](#)

[FIGURE 1-3: The Visual Studio App Wizard is eager to create a new program for y...](#)

[FIGURE 1-4: Visual Studio displays the project you just created.](#)

[FIGURE 1-5: Changing the default project location.](#)

Book 1 Chapter 6

[FIGURE 6-1: The term *swapping two objects* means swapping references to two objects.](#)

Book 1 Chapter 7

[FIGURE 7-1: Adding a path for the files to list.](#)

Book 1 Chapter 9

[FIGURE 9-1: Where, oh where can a handler be found?](#)

[FIGURE 9-2: Providing XML comments for your methods.](#)

Book 2 Chapter 1

[FIGURE 1-1: Two references to the same object.](#)

Book 2 Chapter 2

[FIGURE 2-1: Obtaining a copy of Program.cs from the CalculateInterestTableMoreF...](#)

[FIGURE 2-2: The IDE will tell you what is wrong with the passing of arguments.](#)

[FIGURE 2-3: Visual Studio tells you about the default method parameter values.](#)

Book 2 Chapter 6

[FIGURE 6-1: A UML description of the HighSchool and University classes.](#)

[FIGURE 6-2: Inheriting HighSchool simplifies the University class but introduces...](#)

[FIGURE 6-3: Base both HighSchool and University on a common School Class.](#)

[FIGURE 6-4: Class factoring usually results in added layers of inheritance hier...](#)

[FIGURE 6-5: Breaking down classes is partially a function of the problem being ...](#)

Book 2 Chapter 7

[FIGURE 7-1: A tale of two class hierarchies and one interface.](#)

Book 2 Chapter 8

[FIGURE 8-1: Sending your delegate to the bungee-jump on your behalf.](#)

[FIGURE 8-2: Choose the Windows Forms App \(.NET Framework\) template.](#)

[FIGURE 8-3: The Common Controls Group contains the controls you use most often.](#)

[FIGURE 8-4: Create the form you use to demonstrate the use of a progress bar.](#)

Book 2 Chapter 9

[FIGURE 9-1: Use the Add New Item dialog box to add a new item to your project.](#)

[FIGURE 9-2: The Class Library \(.NET Framework\) is for use with Windows alone.](#)

[FIGURE 9-3: Configure the class as needed.](#)

[FIGURE 9-4: Configure the options for the application used to test the DoMath C...](#)

[FIGURE 9-5: Organizing two projects in an all-in-one-folder.](#)

[FIGURE 9-6: Organizing two projects side by side.](#)

[FIGURE 9-7: The projects used to create the TestClass solution are listed indiv...](#)

[FIGURE 9-8: Add a reference for your class library.](#)

Book 3 Chapter 1

[FIGURE 1-1: The Windows Security application sample.](#)

[FIGURE 1-2: The WindowsSecurity tab of the My Project configuration file.](#)

Book 3 Chapter 2

[FIGURE 2-1: Choose a source type for the application data.](#)

[FIGURE 2-2: Choose a database model to use to model the data.](#)

[FIGURE 2-3: Choosing your data connection.](#)

[FIGURE 2-4: The Choose Data Source dialog box.](#)

[FIGURE 2-5: Specify the location of the database file used for this example.](#)

[FIGURE 2-6: Selecting data objects.](#)

[FIGURE 2-7: Table Options drop-down list.](#)

[FIGURE 2-8: Creating a Parts Detail data form.](#)

[FIGURE 2-9: Generated code. Huh?](#)

Book 3 Chapter 4

[FIGURE 4-1: Many controls come with SmartTags that let you configure them easil...](#)

[FIGURE 4-2: Configure the `StatusStrip` to provide the user with useful informati...](#)

[FIGURE 4-3: Configuring the form to accept email information.](#)

Book 3 Chapter 5

[FIGURE 5-1: A traditional cribbage board.](#)

[FIGURE 5-2: The digital cribbage board.](#)

[FIGURE 5-3: The basic board.](#)

[FIGURE 5-4: Add a Leave event handler for each of the `TextBox` controls.](#)

Book 3 Chapter 6

[FIGURE 6-1: The Dynamic Language Runtime.](#)

Book 4 Chapter 1

[FIGURE 1-1: The Visual Studio Community edition provides lots of project types.](#)

[FIGURE 1-2: Solutions and projects appear in Solution Explorer.](#)

Book 4 Chapter 2

[FIGURE 2-1: Creating a UWP application is a process that Visual Studio helps wi...](#)

[FIGURE 2-2: UWP applications rely on the use of pages to display information.](#)

[FIGURE 2-3: The WPF Designer.](#)

[FIGURE 2-4: The Windows Forms Designer.](#)

[FIGURE 2-5: When you use Data View, who needs SQL Management Studio?](#)

[FIGURE 2-6: The explorer of solutions.](#)

[FIGURE 2-7: Modifying object properties.](#)

[FIGURE 2-8: The Toolbox, with tools.](#)

[FIGURE 2-9: Server Explorer.](#)

[FIGURE 2-10: A view with Class.](#)

[FIGURE 2-11: The Breakpoints window.](#)

[FIGURE 2-12: Executing to a line of code.](#)

[FIGURE 2-13: Use the Locals window to see local variable values.](#)

[FIGURE 2-14: Use the Watch window to create custom variable views.](#)

Book 4 Chapter 3

[FIGURE 3-1: The default Options screen.](#)

[FIGURE 3-2: Determine whether you want to create a project or item template.](#)

[FIGURE 3-3: Provide enough information for others to use your template.](#)

[FIGURE 3-4: Your template will normally appear first in the list after you remo...](#)

[FIGURE 3-5: Determine which item to export from the project.](#)

[FIGURE 3-6: Select the references needed to use your item successfully.](#)

Book 5 Chapter 1

[FIGURE 1-1: A typical combo box.](#)

[FIGURE 1-2: Visualizing data — a WPF combo box.](#)

[FIGURE 1-3: Configuring the WPF project.](#)

[FIGURE 1-4: WPF Application solution structure.](#)

Book 5 Chapter 2

[FIGURE 2-1: Vertical Stack panel.](#)

[FIGURE 2-2: Horizontal layout showing clipped content.](#)

[FIGURE 2-3: Two WrappanelS housed in a StackPanel.](#)

[FIGURE 2-4: A Dock panel fills in controls in the area specified in XAML order.](#)

[FIGURE 2-5: Canvas sample.](#)

[FIGURE 2-6: Basic Grid with proportional \(*\) row heights.](#)

[FIGURE 2-7: Grid with row and column spans.](#)

[FIGURE 2-8: An example using Margin and Padding.](#)

[FIGURE 2-9: Multiple Grids with shared sizing.](#)

[FIGURE 2-10: Simple data entry form.](#)

[FIGURE 2-11: Display-only controls.](#)

[FIGURE 2-12: All the basic input controls.](#)

[FIGURE 2-13: The ComboBox \(left\), ListBox \(center\), and TreeView \(right\) contro...](#)

Book 5 Chapter 3

[FIGURE 3-1: Data binding to properties of a DataContext.](#)

[FIGURE 3-2: Editing data using a TwoWay binding mode.](#)

[FIGURE 3-3: TwoWay data binding with INotifyPropertyChanged.](#)

[FIGURE 3-4: Displaying error messages using Styles.](#)

[FIGURE 3-5: Rendering a collection of data using a value converter and data tem...](#)

Book 5 Chapter 4

[FIGURE 4-1: Using the Copy and Paste features of the application.](#)

[FIGURE 4-2: Selecting the text enables the Copy button.](#)

[FIGURE 4-3: Typing a name \(or other text\) enables both the button and the menu.](#)

[FIGURE 4-4: Clicking either control produces simple output.](#)

Book 5 Chapter 5

[FIGURE 5-1: You can't create a UWP app without setting Developer Mode on.](#)

[FIGURE 5-2: Microsoft warns you about the possible problems in enabling Develop...](#)

[FIGURE 5-3: Make sure that local group policies don't get in the way of sidelo...](#)

[FIGURE 5-4: Using the search field on the taskbar makes it easy to find the Dev...](#)

[FIGURE 5-5: The Settings app provides an overview of developer-related settings...](#)

[FIGURE 5-6: The actual settings for Developer Mode appear in a number of places...](#)

[FIGURE 5-7: Use the Remote Desktop settings with care.](#)

[FIGURE 5-8: A listing of UWP templates.](#)

[FIGURE 5-9: A UWP project requires some additional configuration.](#)

[FIGURE 5-10: After completing the project wizard, you see a list of development...](#)

[FIGURE 5-11: The app interface is very flexible and should work well for most d...](#)

[FIGURE 5-12: The example app shown in a 13.5" Surface Book \(3000 x 2000\) form.](#)

[FIGURE 5-13: The example app shown in a 6" Phone \(1920 x 1080\) form.](#)

[FIGURE 5-14: The Run button includes options for alternative deployments.](#)

[FIGURE 5-15: Locating the .NET Core templates is made easier using a search.](#)

Book 6 Chapter 1

[FIGURE 1-1: Even an empty template provides you with a considerable number of f...](#)

[FIGURE 1-2: The version of .NET and app features both contribute to the file st...](#)

[FIGURE 1-3: The Web App template provides you with basic content as a starting ...](#)

[FIGURE 1-4: Use the generated SSL certificate or install your own.](#)

[FIGURE 1-5: You must install any certificate you use in Windows as well as Visu...](#)

[FIGURE 1-6: Separating functionality in MVC means yet more files.](#)

[FIGURE 1-7: The ASP.NET Core Web API focuses on the API, not the client.](#)

[FIGURE 1-8: Configure your new project with identifying information.](#)

[FIGURE 1-9: The example uses fewer template features to reduce project complexi...](#)

[FIGURE 1-10: Add a new item to the wwwroot folder.](#)

Book 6 Chapter 3

[FIGURE 3-1: Locate the ASP.NET Core Web API template.](#)

[FIGURE 3-2: Provide a name and location for the API.](#)

[FIGURE 3-3: Select the additional requirements for the API.](#)

[FIGURE 3-4: Determine whether you want to install the SSL certificate.](#)

[FIGURE 3-5: Make Windows feel better by answering the question a second time.](#)

[FIGURE 3-6: The browser output shows serialized weather forecast data.](#)

[FIGURE 3-7: Look for the IIS Express icon in the Notification Area to see which...](#)

[FIGURE 3-8: Stop the site so that you can see changes to your user-interface co...](#)

[FIGURE 3-9: Add the WeatherForecast class to your project.](#)

[FIGURE 3-10: See the generated weather forecast from the WeatherForecast API.](#)

Introduction

C# is an amazing language that is currently ranked the fifth most popular language in the world, according to the Tiobe Index (<https://www.tiobe.com/tiobe-index/>)! You can use this single language to do everything from desktop development to creating web applications and even web-based application programming interfaces (APIs). In addition, C# now makes it possible to target a multitude of platforms, including macOS and Linux. While other developers have to overcome deficiencies in their languages to create even a subset of the application types that C# supports with aplomb, you can be coding your application, testing, and then sitting on the beach enjoying the fruits of your efforts. Of course, any language that does this much requires a bit of explanation, and *C# 10.0 All-in-One For Dummies* is your doorway to this new adventure in development.

So, why do you need *C# 10.0 All-in-One For Dummies* specifically? This book stresses learning the basics of the C# language before you do anything else. With this in mind, the book begins with all the C# basics in [Books 1](#) through [3](#), helps you get Visual Studio 2022 installed in [Book 4](#), and then takes you through more advanced development tasks, including basic web development, in [Books 5](#) through [6](#). Using this book helps you get the most you can from C# 10.0 in the least possible time.

About This Book

Even if you have past experience with C#, the new features in C# 10.0 will have you producing feature-rich applications in an even shorter time than you may have before. *C# 10.0 All-in-One For Dummies* introduces you

to all these new features. For example, you discover how to work with both Universal Windows Platform (UWP) and Windows 10 and above applications (besides using all the old standbys). You also find all the new features provided for object-oriented development, and new IDE features designed to make your development experience easier. Make sure you don't miss out on the new Record type discussed in Book 2, [Chapter 11](#). This book is designed to make using C# 10.0 fast and easy; it removes the complexity that you may have experienced when trying to learn about these topics online.

To help you absorb the concepts, this book uses the following conventions:

- » Text that you're meant to type just as it appears in the book is in **bold**. The exception is when you're working through a step list: Because each step is bold, the text to type is not bold.
- » Words for you to type that are also in *italics* are meant as placeholders; you need to replace them with something that works for you. For example, if you see "Type **Your Name** and press Enter," you need to replace *Your Name* with your actual name.
- » I also use *italics* for terms I define. This means that you don't have to rely on other sources to provide the definitions you need.
- » Web addresses and programming code appear in monofont. If you're reading a digital version of this book on a device connected to the Internet, you can click the live link to visit a website, like this: www.dummies.com.
- » When you need to click command sequences, you see them separated by a special arrow, like this: File ⇒ New File, which tells you to click File and then click New File.

Foolish Assumptions

You might have a hard time believing that I've assumed anything about you — after all, I haven't even met you yet! Although most assumptions are indeed foolish, I made certain assumptions to provide a starting point for the book.

The most important assumption is that you know how to use Windows, have a copy of Windows properly installed, and are familiar with using Windows applications. Even though this book covers developing applications that run on multiple platforms, the development environment always assumes that you're working with Windows. If installing an application is still a mystery to you, you might find this book a bit hard to use. While reading this book, you need to install applications, discover how to use them, and create simple applications of your own.

You also need to know how to work with the Internet. Many of the materials, including the downloadable source, appear online, and you need to download them in order to get the maximum value from the book. In addition, [Book 6](#) assumes that you have a certain knowledge of the Internet when working through web-based applications and web-based services.

Icons Used in This Book

As you read this book, you encounter icons in the margins that indicate material of special interest (or not, as the case may be!). Here's what the icons mean:



TIP

Tips are nice because they help you save time or perform some task without a lot of extra work. The tips in this book are time-saving techniques or pointers to resources that you should try so that you can get the maximum benefit when performing C#-related tasks.



WARNING

I don't want to sound like an angry parent or some kind of maniac, but you should avoid doing anything that's marked with a Warning icon. Otherwise, you might find that your configuration fails to work as expected, you get incorrect results from seemingly bulletproof processes, or (in the worst-case scenario) you lose data.



TECHNICAL
STUFF

Whenever you see this icon, think advanced tip or technique. You might find these tidbits of useful information just too boring for words, or they could contain the solution you need to get a C# application running. Skip these bits of information whenever you like.



REMEMBER

If you don't get anything else out of a particular chapter or section, remember the material marked by this icon. This text usually contains an essential process or a bit of information that you must know to work with C#.

Beyond the Book

This book isn't the end of your C# learning experience — it's really just the beginning. I provide online content to make this book more flexible and better able to meet your needs. Also, you can send me e-mail at John@JohnMuellerBooks.com. I'll address your book-specific questions and tell you how updates to C# or its associated add-ons affect book content through blog posts. Here are some cool online additions to this book:

- » **Cheat sheet:** You remember using crib notes in school to make a better mark on a test, don't you? You do? Well, a cheat sheet is sort of like that. It provides you with some special notes about tasks that you can do with C# that not every other person knows. To find the cheat sheet for this book, go to www.dummies.com and search for *C# 10.0 All-in-One For Dummies Cheat Sheet*. It contains really neat information such as how to figure out which template you want to use.
- » **Updates:** Sometimes changes happen. For example, I might not have seen an upcoming change when I looked into my crystal ball during the writing of this book. In the past, this possibility simply meant that the book became outdated and less useful, but you can now find updates to the book at www.dummies.com. In addition to these updates, check out the blog posts with answers to reader questions and demonstrations of useful book-related techniques at <http://blog.johnmuellerbooks.com/>.
- » **Companion files:** Hey! Who really wants to type all the code in the book manually? Most readers prefer to spend their time actually working with C#, creating amazing new applications that change the world, and

seeing the interesting things they can do, rather than typing. Fortunately for you, the examples used in the book are available for download, so all you need to do is read the book to learn C# development techniques. You can find these files at www.dummies.com and at <http://www.johnmuelเลอร์books.com/source-code/>.

Where to Go from Here

Anyone who is unfamiliar with C# should start with Book 1, [Chapter 1](#) and move from there to the end of the book. This book is designed to make it easy for you to discover the benefits of using C# from the outset. Later, after you've seen enough C# code, you can install Visual Studio and then try the programming examples found in the first three minibooks. (Note that Book 1, [Chapter 1](#) provides a brief overview of using Jupyter Notebook instead of Visual Studio 2022, but many of the new examples won't work with this setup.)

This book assumes that you want to see C# code from the outset. However, if you want to interact with that code, you really need to have a copy of Visual Studio 2022 installed. (Some examples will not work at all with older Visual Studio versions.) With this in mind, you may want to skip right to [Book 4](#) to discover how to get your own copy of Visual Studio 2022. To help ensure that everyone can participate, this book focuses on the features offered by Visual Studio 2022 Community Edition, which is a free download. That's right, you can discover the wonders of C# 10.0 without paying a dime!

The more you know about C#, the later you can start in the book. If all you're really interested in is an update of your existing skills, check out Book 1, [Chapter 1](#) to discover the changes in C#. Then, scan the first three

minibooks looking for points of interest. Install C# by using the instructions in Book 4, [Chapter 1](#), and then move on toward the advanced techniques found in later chapters. You definitely don't want to miss out on the Windows 10 and above development topics in Book 5, [Chapter 5](#). In addition, [Book 6](#) is entirely new for this edition, so even if you saw the previous edition of the book, you don't want to miss out on this new content.

Book 1

The Basics of C# Programming

Contents at a Glance

Chapter 1: Creating Your First C# Console Application

[Getting a Handle on Computer Languages, C#, and .NET](#)

[Creating Your First Console Application](#)

[Making Your Console App Do Something](#)

[Reviewing Your Console Application](#)

[Replacing All that Ceremonial Code: Top-Level Statements](#)

[Introducing the Toolbox Trick](#)

[Interacting with C# Online](#)

[Working with Jupyter Notebook: The Short Version](#)

Chapter 2: Living with Variability — Declaring Value-Type Variables

[Declaring a Variable](#)

[What's an int?](#)

[Representing Fractions](#)

[Handling Floating-Point Variables](#)

[Using the Decimal Type: Is It an Integer or a Float?](#)

[Examining the bool Type: Is It Logical?](#)

[Checking Out Character Types](#)

[What's a Value Type?](#)

[Comparing string and char](#)

[Calculating Leap Years: DateTime](#)

[Declaring Numeric Constants](#)

[Changing Types: The Cast](#)