

C# 13 and .NET 9

*Guide to C# 13, .NET 9,
data management, and deployment strategies*

Jefferson S. Motta



www.bpbonline.com

First Edition 2025

Copyright © BPB Publications, India

ISBN: 978-93-65898-521

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

To View Complete
BPB Publications Catalogue
Scan the QR Code:



Dedicated to

To the sisters: M. A. A.

Prof. Cássio Saldanha, My Mentor

Natali Fonseca

About the Author

Jefferson S. Motta is currently working as a consultant with over 25 years of experience in desktop, web, and cloud development. His professional journey began in 1994 as a programming intern at Janssen Consultoria, where he helped develop an MRP II System for the Olvebra Group.

In 1996, he founded Memphis - Intelligent Systems, focusing on IT solutions for Olvebra and new clients. His most notable achievement was the development of Advocati.NET, a management system for law firms that has earned a respectable place in Brazilian legal technology. The system has undergone continuous improvement for over 20 years and is built using C# 13, ASP.NET Web Forms, and WinForms .NET 9.0, incorporating elements of artificial intelligence and big data.

Jefferson has been a pioneer in technological innovation, developing applications with artificial intelligence as early as 2012, followed by implementing bigdata solutions and automated control systems for physical equipment. In 2005, he created a management system for Medical Clinics.

He has made significant contributions to the tech community as an editor at C# Corner, a respected platform for technical articles. As a four-time MVP, his articles in English have garnered over 850,000 views, elevating him to approximately the 90th position in the global ranking.

He is also a graduate with a software engineering degree and has a diverse educational background, including digital marketing analysis (SENAC), digital security (Sisnema), and various technical courses.

About the Reviewer

Rositsa is an experienced engineering manager with 20 years of experience in the software industry, with a significant portion of that time leading teams that developed Telerik UI components for .NET MAUI, and Telerik UI for WPF and WinForms.

Her focus is on building high-performing teams that enjoy crafting software that they are proud of. She has also been a Microsoft MVP for .NET for the last 5 years (2020-2025).

Acknowledgement

Hey there! We want to thank everyone from the bottom of our hearts who helped make this book happen!

First, a big hug to our family and friends who stuck by us throughout this journey. Your love kept us going when things got tough, you know?

We owe a big debt of gratitude to the team at BPB Publications, who gave us expert publishing support. You guys were amazing in helping us navigate this crazy world of book publishing!

We can't forget the reviewers, technical experts, and editors who gave us that honest feedback to improve our manuscript. Your suggestions made all the difference!

Last but not least, a special thanks to you, the reader of our book. Your support means everything to us!

Thanks to everyone who pitched in to make this book a reality. You rock!

Preface

If you are new to programming or an experienced professional looking to enhance your skills, these pages will provide you with the information and tips you need to succeed in the .NET ecosystem, which is constantly evolving.

From Chapter 1's Introducing to C# 13 and .NET 9 to Chapter 16's Packaging and Deployment, each chapter is carefully organized to build on what you already know while presenting new ideas at a pace that you can easily handle.

In this book, you will discover the remarkable flexibility of the C# language and the .NET system. It will guide you through object-oriented programming, working with data using Entity Framework Core, utilizing LINQ to modify data in a concise and elegant manner, and leveraging ASP.NET Core and Blazor for modern web development.

I hope that this book not only helps you learn how to code but also inspires you to do so, providing you with the tools to solve complex problems in innovative ways and adding your own unique perspective and will to the world of software development.

Chapter 1: Introducing to C# 13 and .NET 9 - This chapter guides you through the key stages of configuring your development environment and acquiring fundamental knowledge of C# programming. This first chapter establishes a solid foundation for your coding journey.

Here, you will find the step-by-step installation procedure for the **.NET software development kit (SDK)**—an essential tool for any developer—as well as guidance on how to select an appropriate **integrated development environment (IDE)**.

Chapter 2: C# Fundamentals - Originally part of the .NET framework, C# has proven to be a top option in software development for Microsoft because of its scalability, dependability, and simple syntax. The first part looks at the basic structural guidelines and conventions needed to write C# programs. From here, the trip delves into an in-depth study of data types, revealing the nuances of integers, floats, strings, and Booleans—the fundamental building blocks of data storage and processing.

Chapter 3: Harnessing the Code - In this chapter, we will use iterative techniques to explore iteration and learn how to effectively navigate data. We will clarify type conversion and type casting as we proceed, thereby equipping you with the tools to properly manage data. Furthermore, we will cover the principles of robust error management and exception handling, enabling you to produce consistent and reliable code.

Chapter 4: Functions In-depth - In this chapter, we will find ways to improve your code's modularity and adaptability. Techniques for exception handling are discussed to ensure robustness and to safeguard your code from unexpected errors. The need for unit testing in C# is also addressed since it helps to preserve code quality and dependability.

Chapter 5: Building Apps with OOPs- In this chapter,thorough investigations of key ideas such as encapsulation and abstraction will help to create strong and safe programs. Moving into advanced subjects, we explore inheritance and polymorphism, which enable dynamic behavior and code reuse. Along with static and instance members, you will investigate constructors, acquiring a clear knowledge of how objects are initialized and controlled.

Chapter 6: Mastering Interfaces and Inheriting Classes- This chapter explores polymorphism, which enables objects to take on multiple forms; examines inheritance as a foundational concept in object-oriented programming, facilitating code reuse; and discusses the value of interfaces as contracts for class behavior. Mastering these basic concepts will help you build a solid foundation to enhance your programming skills.

Chapter 7: .NET Toolbox - Forming one of the basic data types within the .NET framework, strings are crucial in many applications. They provide many features for text manipulation. Focusing on the string class and its closely related types—such as `DateTime`, `TimeSpan`, encoding, and regex—this chapter guides you on an in-depth exploration of the `System` namespace. We will demonstrate their practical applications in various situations and reveal the architectural ideas and design choices that underpin these classes

Chapter 8: Data in Motion- This chapter covers the fundamentals of file operations in .NET, including reading and writing text files, handling file and directory exceptions, and guaranteeing safe file operations. We also examine the key role of .NET streams in supporting data reading and writing across various sources, including files, networks, and memory. You will also learn about .NET serialization, a strong tool for transforming data structures or objects into digital forms for storage or transport.

Chapter 9: Data Handling with EF Core - This chapter explores **Entity Framework Core (EF Core)**, a sophisticated **Object-Relational Mapping (ORM)** framework designed for .NET applications. EF Core simplifies processes and increases output by utilizing .NET objects instead of SQL queries, thereby enabling efficient data management.

Chapter 10: LINQ Unleashed- This chapter delves into the intricacies of **Language Integrated Query (LINQ)**, a powerful C# feature that enables programmers to query and manipulate data seamlessly across multiple data sources. You will learn to use LINQ with various data types—including collections, databases, XML, and JSON—by means of

a coherent and expressive syntax. You will also investigate leveraging LINQ operators, lambda expressions, and deferred execution to create efficient designs and adaptable queries.

Chapter 11: ASP.NET Core the Future of Web Development- This chapter introduces a modern and flexible framework meant for creating web apps and services, ASP.NET Core. Among its cross-platform support, great performance, modular architecture, and strong security, the fundamental features and benefits of ASP.NET Core will be revealed. Through hands-on activities, you will acquire knowledge of building a simple web application, investigating its architecture, adding features, running tests, refining features, and efficiently deploying it.

Chapter 12: Building Powerful Websites with Razor- This chapter covers Razor Pages, a key feature of ASP.NET Core that utilizes a page-oriented approach to transform web development. Emphasizing obvious separation of concerns and effective data management, Razor Pages simplifies the process by combining C# and HTML into a single file. Using Razor syntax, Tag Helpers, layouts, and partials, you will learn how to create and control Razor Pages using the .NET Core framework to build dynamic and responsive online applications. Form management, user input processing, data validation, and fundamental **Create, Read, Update, Delete (CRUD)** activities using Razor Pages are all covered in the chapter.

Chapter 13: Website Development Using MVC Pattern- This chapter explores the **Model-View-Controller (MVC)** architecture within the ASP.NET Core framework, a powerful tool for creating modern web applications. You will learn to create views and controllers, set up MVC routing, use Entity Framework Core to handle data, apply strong form validation, and use ViewModels to simplify your development process.

Chapter 14: Mastering Web Services- This chapter provides a comprehensive overview of the various web service domains within the .NET environment, including an analysis of their potential as well as their challenges. Emphasizing strong security policies against current cyber threats, it unpacks the complexity of building and using web services from basic ideas like RESTful APIs in ASP.NET Core to advanced technologies like gRPC and GraphQL.

Chapter 15: Blazor for UI Development- This chapter explores Blazor, a state-of-the-art UI framework using C# for both client-side and server-side programming. Key subjects are thoroughly investigated in it, including Blazor component development, project organization, form handling and validation, JavaScript interoperability, CRUD operations

with Blazor and Entity Framework Core, differences between Server and WebAssembly models, routing, and application testing.

Chapter 16: Packaging and Deployment- This chapter explores the complexities of packaging and distributing .NET applications, including a thorough guide to components and libraries, NuGet's important role, and the process of publishing code for deployment. It highlights the importance of .NET Standard and .NET Core libraries, assembly versioning, the **Global Assembly Cache (GAC)**, and strongly named assemblies.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the *Coloured Images* of the book:

<https://rebrand.ly/i244egu>

The code bundle for the book is also hosted on GitHub at

<https://github.com/bpbpublications/C-Sharp-13-and-.NET-9>.

In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at
<https://github.com/bpbpublications>. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



Table of Contents

| | |
|---|----------|
| 1. Introduction to C# 13 and .NET 9..... | 1 |
| Introduction..... | 1 |
| Structure..... | 1 |
| Objectives | 2 |
| Introduction to C# and .NET | 2 |
| <i>What is new in C# 13</i> | <i>4</i> |
| Choosing the right tools for C# programming | 5 |
| <i>Integrated development environment</i> | <i>5</i> |
| <i>Compiler</i> | <i>6</i> |
| <i>Libraries and frameworks</i> | <i>6</i> |
| <i>Source control.....</i> | <i>6</i> |
| <i>Unit testing</i> | <i>6</i> |
| Understanding application types in C# | 6 |
| <i>Differences between types of applications</i> | <i>7</i> |
| <i>Console applications</i> | <i>7</i> |
| <i>Windows Forms applications</i> | <i>7</i> |
| <i>Windows Presentation Foundation applications</i> | <i>7</i> |
| <i>ASP.NET applications.....</i> | <i>8</i> |
| <i>Blazor applications</i> | <i>8</i> |
| <i>Mobile applications with .NET MAUI</i> | <i>8</i> |
| <i>Game development with Unity</i> | <i>8</i> |
| <i>Universal Windows Platform applications</i> | <i>8</i> |
| <i>NET Multi-platform App UI</i> | <i>9</i> |
| <i>Subtypes of .NET MAUI</i> | <i>9</i> |
| <i>Model-View-ViewModel and Model-View-Update architectures</i> | <i>9</i> |
| Deploying cross-platform applications..... | 10 |
| Choosing the right framework..... | 10 |
| Requirements..... | 10 |
| Pros and cons of choosing WFP and WinForms..... | 11 |
| Designing for cross-platform compatibility | 12 |
| Utilizing .NET's cross-platform tools | 13 |

| | |
|--|-----------|
| <i>Deploying with platform-specific configurations</i> | 13 |
| <i>Ensuring thorough testing</i> | 13 |
| <i>Implementing a CI/CD pipeline</i> | 14 |
| <i>Windows applications deployment</i> | 14 |
| <i>WebForms applications deployment</i> | 15 |
| <i>Console applications deployment</i> | 15 |
| <i>Mobile applications deployment</i> | 15 |
| <i>Android deployment generating APK and AAB Files</i> | 16 |
| <i>.NET MAUI deployment</i> | 16 |
| <i>iOS deployment crafting the IPA File</i> | 17 |
| <i>Mac Catalyst deployment the APP and PKG files</i> | 17 |
| <i>Role of integrated development environments</i> | 17 |
| <i>Consolidating essential tools</i> | 18 |
| <i>Code editing and management</i> | 18 |
| <i>Debugging and testing</i> | 19 |
| <i>Integration and extensions</i> | 19 |
| <i>Collaboration and deployment</i> | 19 |
| <i>.NET Framework vs. .NET Core</i> | 20 |
| <i>Overview of C# and .NET ecosystem</i> | 21 |
| <i>Installing and configuring Visual Studio</i> | 21 |
| <i>Exploring Visual Studio Code for C# development</i> | 23 |
| <i>Conclusion</i> | 24 |
| <i>Key terms</i> | 25 |
| <i>Questions</i> | 25 |
| <i>Answers</i> | 26 |
| 2. C# Fundamentals | 27 |
| <i>Introduction</i> | 27 |
| <i>Structure</i> | 28 |
| <i>Objectives</i> | 28 |
| <i>Basics of C# syntax</i> | 29 |
| <i>Features of C# language</i> | 29 |
| <i>Working with variables in C#</i> | 31 |
| <i>Storing and manipulating data in C#</i> | 32 |
| <i>Understanding C# operators</i> | 33 |

| | |
|--|-----------|
| Fundamental C# data types..... | 37 |
| Understanding C# namespaces | 39 |
| Introduction to C# statements, expressions, and operators..... | 42 |
| Using value and reference types in C# | 43 |
| C# type conversion techniques | 45 |
| C# keywords and identifiers | 47 |
| <i>C# keywords</i> | 48 |
| <i>Guidelines for identifiers</i> | 48 |
| Conclusion..... | 49 |
| 3. Harnessing the Code | 51 |
| Introduction..... | 51 |
| Structure..... | 51 |
| Objectives | 52 |
| Control flow in C# with selection statements | 52 |
| <i>If statement</i> | 52 |
| <i>if-else statement</i> | 53 |
| <i>If-else if-else statement</i> | 54 |
| Iteration statements in C# | 56 |
| <i>Foreach statement</i> | 56 |
| <i>Goto statement</i> | 57 |
| <i>Yield statement</i> | 58 |
| Understanding data type conversion in C# | 59 |
| Looping constructs in C# | 65 |
| <i>For loop</i> | 65 |
| <i>While loop</i> | 65 |
| <i>Do-while loop</i> | 66 |
| Understanding and using nullable types in C# | 67 |
| <i>Lifted operators</i> | 68 |
| <i>Identifying a nullable value type</i> | 69 |
| Unchecked and checked operators in C# | 70 |
| <i>Unchecked operator</i> | 71 |
| <i>Checked operator</i> | 71 |
| Overview of exception handling in C# | 71 |
| <i>Best practices</i> | 74 |

| | |
|--|-----------|
| Implementing robust exception handling | 75 |
| Conclusion..... | 78 |
| 4. Functions In-depth..... | 79 |
| Introduction..... | 79 |
| Structure..... | 79 |
| Objectives | 80 |
| Basics of function writing in C# | 80 |
| Function declaration | 80 |
| Function parameters | 81 |
| Function calls and return types | 81 |
| Functions as parameters..... | 82 |
| Passing objects as parameter | 83 |
| Using lambda expressions in C# functions | 85 |
| Identifying lambda expressions | 86 |
| Lambdas expression | 86 |
| Lambdas statement | 86 |
| Understanding the different types of lambda expressions..... | 87 |
| The Action | 87 |
| LINQ lambda expressions | 87 |
| Lambda expressions scope | 88 |
| Anonymous lambda functions..... | 88 |
| Sample using local lambda Func..... | 88 |
| Debugging techniques during development | 90 |
| Stepping through code..... | 90 |
| Examining variables..... | 91 |
| Debugging windows | 91 |
| Handling exceptions..... | 92 |
| Debugging multi-threaded applications..... | 92 |
| Remote debugging | 93 |
| Attach to Process..... | 93 |
| Tips for debugging..... | 94 |
| Implementing unit testing in C# | 95 |
| Making unit tests | 95 |
| Unit test execution and management | 96 |

| | |
|---|-----|
| <i>Best practices for unit testing</i> | 96 |
| Understanding C# methods and parameters..... | 97 |
| <i>Creating a method</i> | 97 |
| <i>Arguments and method parameters</i> | 98 |
| <i>Parameters of value</i> | 98 |
| <i>Parameters of reference</i> | 98 |
| <i>Arrays of parameters</i> | 99 |
| <i>Methods of overloading</i> | 100 |
| <i>Using classes to simplify functions with many parameters</i> | 100 |
| Introduction to .NET debugging tools | 102 |
| <i>Debugger for Visual Studio</i> | 102 |
| <i>Advanced debugging options</i> | 103 |
| <i>SOS debugging extension</i> | 103 |
| Writing testable code, the best practices | 104 |
| <i>The principle of single responsibility</i> | 104 |
| <i>Dependency injection</i> | 104 |
| <i>Avoid using static methods</i> | 105 |
| <i>.NET unit testing</i> | 106 |
| <i>Stubbing and mocking</i> | 106 |
| <i>Prefer composition to inheritance</i> | 107 |
| Using testing frameworks, the NUnit and xUnit | 107 |
| <i>NUnit</i> | 107 |
| <i>xUnit</i> | 109 |
| <i>NUnit sample</i> | 111 |
| <i>xUnit sample</i> | 114 |
| <i>xUnit extensibility sample</i> | 117 |
| Working with return types and void methods..... | 119 |
| <i>Methods with return types</i> | 120 |
| <i>Void methods</i> | 120 |
| <i>Using return types and void methods together</i> | 120 |
| <i>Returning tuples</i> | 121 |
| <i>Return type named tuples</i> | 122 |
| <i>Asynchronous return type</i> | 123 |
| <i>Using awaiter and GetResult with asynchronous methods</i> | 124 |

| | |
|---|------------|
| Understanding and applying recursion in C# | 125 |
| <i>Learning the recursion</i> | 125 |
| <i>Recursion in C#</i> | 126 |
| <i>Understanding the recursive call stack</i> | 126 |
| <i>Tail recursion</i> | 126 |
| <i>Iteration vs. recursion</i> | 127 |
| Conclusion | 128 |
| 5. Building Apps with OOPs | 129 |
| Introduction | 129 |
| Structure | 130 |
| Objectives | 130 |
| Creating classes and objects in C# | 130 |
| <i>Recognizing classes and objects</i> | 130 |
| <i>Creating a C# class</i> | 131 |
| <i>Application in practice</i> | 132 |
| Record object | 133 |
| <i>Defining and using records</i> | 134 |
| SqlDataReader reading | 134 |
| <i>Understanding record x class</i> | 136 |
| Using properties for data access in C# | 136 |
| Properties in C# | 137 |
| <i>Property syntax in C#</i> | 137 |
| Using properties to access data | 137 |
| Storing data with fields in C# | 139 |
| Fields | 139 |
| <i>Fields and encapsulation</i> | 140 |
| <i>Using fields to store data from a database</i> | 140 |
| Getters and setters | 141 |
| Modified getter and setter | 142 |
| <i>Getter and setter with lambda expression</i> | 142 |
| <i>Using PostgreSQL and fields</i> | 143 |
| <i>Data storage in memory</i> | 143 |
| <i>Data entry into the database</i> | 145 |
| Writing and calling methods in C# | 147 |

| | |
|---|-----|
| <i>Defining method in C#</i> | 147 |
| <i>Method return types</i> | 148 |
| <i>Passing multiple parameters</i> | 148 |
| <i>Optional parameters</i> | 148 |
| <i>Overloading methods</i> | 149 |
| <i>Default parameters values</i> | 149 |
| <i>Polymorphism and OOP parameters</i> | 150 |
| <i>Creating extension methods</i> | 151 |
| <i>Practical example</i> | 151 |
| <i>Collections example</i> | 152 |
| <i>Overloading extension method</i> | 153 |
| <i>Extension changing type return</i> | 154 |
| Understanding encapsulation and abstraction | 155 |
| <i>Encapsulation</i> | 155 |
| <i>Abstraction</i> | 156 |
| Implementing inheritance in C# | 157 |
| <i>Understanding inheritance</i> | 158 |
| <i>Example 1 of creating a simple inheritance hierarchy</i> | 158 |
| <i>Example 2 of abstract classes and overriding</i> | 159 |
| Understanding polymorphism in C# | 160 |
| <i>Recognizing polymorphism</i> | 160 |
| <i>Static/compile time polymorphism</i> | 160 |
| <i>Overloading polymorphism</i> | 160 |
| <i>Polymorphism with overriding</i> | 161 |
| Creating and using constructors in C# | 163 |
| <i>Constructor default</i> | 163 |
| <i>Parameterized builder</i> | 163 |
| <i>Using inheritance constructors</i> | 164 |
| <i>Constructor with inheritance</i> | 165 |
| Working with static and instance members in C# | 166 |
| <i>Understanding instance members</i> | 166 |
| <i>Understanding static members</i> | 167 |
| <i>Static vs. instance members</i> | 169 |
| Conclusion..... | 169 |

| | |
|---|------------|
| 6. Mastering Interfaces and Inheriting Classes..... | 171 |
| Introduction..... | 171 |
| Structure..... | 172 |
| Objectives | 172 |
| Understanding the concept of interfaces in C# | 172 |
| <i>History of OOP and its relationship to biology</i> | <i>173</i> |
| <i>Introduction to OOP with C# interfaces</i> | <i>173</i> |
| <i>Impact on teamwork and testing.....</i> | <i>173</i> |
| <i>Interfaces and .NET</i> | <i>174</i> |
| Implementing interfaces in C# | 175 |
| Class inheritance in C# | 177 |
| <i>Vehicle is the base class.....</i> | <i>177</i> |
| <i>Derived class: SportsCar</i> | <i>178</i> |
| <i>The derived class LuxuryCar</i> | <i>178</i> |
| <i>The derived class Truck.....</i> | <i>178</i> |
| <i>The derived class UtilityVehicle</i> | <i>179</i> |
| <i>Code reusability and organization</i> | <i>179</i> |
| <i>Easy maintenance.....</i> | <i>179</i> |
| <i>Agile expansion</i> | <i>180</i> |
| <i>Team collaboration</i> | <i>180</i> |
| Polymorphism in C# via interfaces | 180 |
| <i>Understanding polymorphism</i> | <i>180</i> |
| <i>Polymorphism via interfaces</i> | <i>181</i> |
| <i>A sample e-commerce application.....</i> | <i>181</i> |
| <i>Advantages</i> | <i>183</i> |
| Reusability and extensibility through interfaces and inheritance..... | 183 |
| <i>Leveraging polymorphism with interfaces</i> | <i>185</i> |
| <i>Interface inheritance</i> | <i>185</i> |
| Memory management of the reference types and value types..... | 186 |
| <i>Value types</i> | <i>186</i> |
| <i>Reference types</i> | <i>187</i> |
| <i>Nullable interfaces in C#</i> | <i>187</i> |
| Understanding virtual, override, and new keywords | 190 |
| <i>Key virtual.....</i> | <i>190</i> |

| | |
|---|------------|
| <i>Key override</i> | 190 |
| <i>Key new</i> | 190 |
| Abstract classes vs. interfaces | 194 |
| <i>The impact of knowing when to use abstract classes versus interfaces in C#</i> | 196 |
| Role of interfaces in enforcing contracts | 197 |
| Understanding System.Object and the inheritance hierarchy in C# | 199 |
| <i>Inheritance</i> | 200 |
| <i>Polymorphism</i> | 200 |
| Conclusion..... | 201 |
| 7. .NET Toolbox..... | 203 |
| Introduction..... | 203 |
| Structure..... | 203 |
| Objectives | 204 |
| Understanding System.String and string manipulation techniques | 204 |
| <i>The importance of the System.String</i> | 205 |
| Memory | 206 |
| <i>Maximum character count</i> | 206 |
| <i>Memory restrictions</i> | 206 |
| The Unicode..... | 207 |
| <i>Goals and features of Unicode</i> | 207 |
| <i>Importance of Unicode</i> | 207 |
| What is ASCII..... | 208 |
| ISO encoding..... | 209 |
| <i>String manipulation techniques</i> | 210 |
| Working with dates and times in .NET | 215 |
| Regular expressions in .NET..... | 218 |
| Understanding and using .NET collection types..... | 221 |
| <i>.NET collection types</i> | 222 |
| Working with arrays and tuples..... | 235 |
| Understanding nullable types in C# | 237 |
| Manipulating strings with StringBuilder..... | 239 |
| Parsing dates, times, and numbers in .NET | 240 |
| Understanding .NET format strings..... | 242 |
| Using System.Guid and other common .NET types..... | 246 |

| | |
|--|------------|
| <i>About unsigned and signed integers</i> | 247 |
| Conclusion..... | 248 |
| 8. Data in Motion | 249 |
| Introduction..... | 249 |
| Structure..... | 249 |
| Objectives | 250 |
| Basics of file operations in .NET..... | 250 |
| <i>System.IO namespace</i> | 250 |
| <i>Reading data from a file</i> | 251 |
| <i>Creating a file</i> | 252 |
| Understanding streams in .NET..... | 254 |
| <i>Stream operations</i> | 254 |
| Reading and writing text files in C#..... | 257 |
| <i>Reading text files</i> | 257 |
| <i>Using StreamReader</i> | 257 |
| <i>Using File class</i> | 258 |
| <i>Using StreamWriter</i> | 258 |
| <i>Error handling</i> | 259 |
| Introduction to serialization in .NET..... | 260 |
| <i>Main types of serialization in .NET</i> | 260 |
| <i>Binary serialization</i> | 260 |
| <i>XML serialization</i> | 260 |
| <i>Custom serialization</i> | 261 |
| XML and JSON serialization in .NET | 261 |
| <i>XML and JSON</i> | 261 |
| <i>XML serialization sample</i> | 263 |
| <i>JSON serialization sample</i> | 264 |
| Binary serialization in .NET..... | 267 |
| <i>Important considerations</i> | 268 |
| Understanding file and directory classes in System.IO..... | 269 |
| Working with paths, buffers, and memory streams..... | 270 |
| Handling file and directory exceptions..... | 272 |
| <i>Typical exceptions</i> | 272 |
| <i>Exception handling fundamentals</i> | 272 |

| | |
|---|------------|
| Secure file operations and file I/O permissions | 273 |
| Conclusion..... | 275 |
| 9. Data Handling with EF Core..... | 277 |
| Introduction..... | 277 |
| Structure..... | 277 |
| Objectives | 278 |
| Introduction to Entity Framework Core | 278 |
| Core features of Entity Framework Core..... | 278 |
| Why use Entity Framework Core..... | 279 |
| Working with databases | 280 |
| Steps to create a model..... | 282 |
| Querying data using LINQ..... | 284 |
| Learn to query data using LINQ in EF Core..... | 285 |
| EF Core and LINQ x ADO.NET..... | 286 |
| EF Core and LINQ comparisons | 289 |
| Understanding EF Core change tracking..... | 290 |
| Change tracking vs. manual tracking in ADO.NET..... | 290 |
| Change tracking in EF Core..... | 291 |
| Configuring a database using EF Core..... | 292 |
| PostgreSQL database..... | 293 |
| MySQL database | 294 |
| The Microsoft SQL Server | 294 |
| Oracle database..... | 295 |
| Final steps..... | 295 |
| Performing CRUD operations using EF Core | 296 |
| Navigating relationships with EF Core..... | 301 |
| Explanation | 303 |
| Learning how relationships are modeled | 303 |
| Navigation properties..... | 303 |
| Migrations in Entity Framework Core | 306 |
| Learning migrations makes developer's life easier | 306 |
| Before migration technology..... | 306 |
| Sample for MySQL and SQLServer | 306 |
| Challenges that migrations solve in EF Core..... | 307 |

| | |
|---|------------|
| Implementing inheritance with Entity Framework Core | 308 |
| <i>Learn inheritance for concise, efficient, and clean code</i> | 310 |
| <i>Adaptability for unpredictable future needs</i> | 310 |
| Performance considerations when using Entity Framework Core | 311 |
| <i>Financial consequences</i> | 311 |
| <i>User experience and product quality</i> | 311 |
| <i>Steve Jobs and saving time</i> | 312 |
| <i>Actions to make EF Core efficient</i> | 312 |
| <i>Lazy loading vs. eager loading</i> | 313 |
| <i>Tracking vs. no tracking</i> | 315 |
| <i>Batch operations</i> | 316 |
| <i>Filtering and paging</i> | 317 |
| <i>Indexing</i> | 317 |
| <i>Avoiding SELECT * queries</i> | 319 |
| <i>Compiled queries</i> | 319 |
| <i>Considerations of EF Core performance</i> | 320 |
| Conclusion | 321 |
| Exercises | 321 |
| <i>Answers</i> | 324 |
| 10. LINQ Unleashed | 325 |
| Introduction | 325 |
| Structure | 325 |
| Objectives | 326 |
| Understanding Language Integrated Query | 326 |
| <i>The benefits of using LINQ</i> | 326 |
| <i>LINQ and databases</i> | 327 |
| <i>Lists and collections in LINQ</i> | 327 |
| <i>Difficulties before LINQ</i> | 328 |
| Writing simple LINQ queries | 328 |
| Grouping and aggregation with LINQ | 330 |
| <i>Aggregation</i> | 334 |
| Using LINQ with Entity Framework Core | 334 |
| Understanding deferred execution in LINQ | 338 |
| <i>Advantages</i> | 339 |

| | |
|---|------------|
| <i>Making developers work faster and easier</i> | 339 |
| <i>Using deferred execution with EF Core</i> | 340 |
| LINQ to XML and LINQ to JSON | 341 |
| <i>XML, the markup language</i> | 341 |
| <i>LINQ's user-friendliness</i> | 343 |
| <i>Working with XML and JSON using LINQ</i> | 343 |
| <i>LINQ to XML example</i> | 344 |
| <i>LINQ for XML reading</i> | 347 |
| <i>LINQ to JSON example</i> | 348 |
| <i>Reading from JSON using System.Text.Json</i> | 350 |
| Using LINQ to manipulate collections and arrays..... | 350 |
| <i>Considerations about LINQ operations</i> | 352 |
| Advanced LINQ operators | 353 |
| Understanding lambda expressions with LINQ..... | 355 |
| <i>Anatomy of a lambda expression</i> | 355 |
| <i>Sample filtering a list of integers using Where</i> | 355 |
| Implementing custom comparers and equality in LINQ | 356 |
| <i>Custom comparer for complex type</i> | 358 |
| What is new in .NET 9 | 360 |
| Conclusion..... | 362 |
| Exercise | 363 |
| <i>Answers</i> | 363 |
| 11. ASP.NET Core the Future of Web Development | 365 |
| Introduction..... | 365 |
| Structure..... | 365 |
| Objectives | 366 |
| Introduction to ASP.NET Core | 366 |
| <i>Creating a basic web application with ASP.NET Core</i> | 368 |
| <i>Setting up the environment</i> | 368 |
| <i>Understanding the structure</i> | 373 |
| <i>Developing the application</i> | 373 |
| <i>Testing the application</i> | 374 |
| <i>Enhancing the application</i> | 374 |
| <i>Deploying the application</i> | 375 |

| | |
|---|-----|
| Understanding Razor Pages in ASP.NET Core | 375 |
| <i>Understanding the Razor syntax</i> | 375 |
| <i>Razor Pages structure</i> | 376 |
| <i>Handling requests with page handlers</i> | 376 |
| <i>Form submission and model binding</i> | 377 |
| <i>Razor Pages routing and URL generation</i> | 379 |
| <i>Creating the Razor Page sample</i> | 380 |
| <i>Use cases</i> | 382 |
| Introduction to the Model-View-Controller pattern | 384 |
| <i>Model-View-Controller pattern in web development pattern</i> | 385 |
| <i>Model</i> | 385 |
| <i>View</i> | 386 |
| <i>Controller</i> | 388 |
| <i>Sample Controller for the View and Model</i> | 388 |
| <i>Creating and using View components</i> | 391 |
| Understanding middleware in ASP.NET Core | 392 |
| <i>Understanding what a pipeline is</i> | 393 |
| <i>Pipeline workflow example</i> | 393 |
| <i>Why we need middleware</i> | 394 |
| <i>Middleware order</i> | 396 |
| History of middleware in ASP.NET Core | 398 |
| <i>Goals of middleware</i> | 398 |
| <i>Steps of pipeline workflow</i> | 399 |
| <i>Avoiding middleware with route short-circuiting</i> | 399 |
| Working with dependency injection in ASP.NET Core | 402 |
| <i>Basics of DI</i> | 402 |
| <i>Dependency injection fundamentals</i> | 402 |
| <i>Overview of routing in ASP.NET Core</i> | 408 |
| <i>Types of routing</i> | 408 |
| <i>Utilizing route parameters</i> | 409 |
| <i>Advanced routing options</i> | 409 |
| <i>Understanding ASP.NET Core security</i> | 410 |
| <i>Basics of error handling in ASP.NET Core</i> | 413 |
| <i>Understanding error handling in ASP.NET Core</i> | 413 |

| | |
|---|------------|
| Error handling mechanisms in ASP.NET Core | 413 |
| Sample implementations | 414 |
| Summary..... | 415 |
| Understanding server-side vs. client-side in ASP.NET Core | 416 |
| Server-side in ASP.NET Core | 416 |
| Client-side in ASP.NET Core..... | 417 |
| Improvements in .NET 9 | 419 |
| Support for polymorphic types in SignalR hubs..... | 420 |
| Conclusion..... | 422 |
| 12. Building Powerful Websites with Razor | 423 |
| Introduction..... | 423 |
| Structure..... | 423 |
| Objectives | 424 |
| Creating Razor Pages in ASP.NET Core..... | 424 |
| Understanding the lifecycle of a Razor Page | 429 |
| Lifecycle of a Razor Page | 429 |
| Example 1 contact form..... | 430 |
| Example 2 data display page | 433 |
| Working with forms in Razor Pages..... | 436 |
| Form validation in ASP.NET Core Razor Pages | 438 |
| Understanding Tag Helpers in Razor Pages | 444 |
| Razor syntax and its usage in Razor Pages | 447 |
| Features of Razor syntax..... | 448 |
| Razor Pages key features for programming model | 451 |
| The advantages of using Razor Pages | 451 |
| Explanation of the Razor Page's key features | 453 |
| Best practices | 454 |
| A broad perspective | 455 |
| Creating layouts and partials in Razor Pages | 458 |
| Creating a layout..... | 459 |
| Understanding Razor Pages partials | 462 |
| Using partials in Razor Pages..... | 463 |
| Best practices | 464 |
| Typical use cases..... | 464 |

| | |
|--|------------|
| Building a CRUD operation using Razor Pages | 469 |
| Conclusion..... | 471 |
| Exercise | 472 |
| <i>Answers</i> | 473 |
| 13. Website Development Using MVC Pattern | 475 |
| Introduction..... | 475 |
| Structure..... | 475 |
| Objectives | 476 |
| Introduction to the MVC pattern in ASP.NET Core | 476 |
| Creating Controllers and Views in MVC | 477 |
| <i>MVC Controllers</i> | 478 |
| <i>MVC Views</i> | 478 |
| <i>Controllers and Views interaction</i> | 478 |
| <i>MVC and n-tier architecture</i> | 478 |
| <i>ASP.NET Core MVC with n-tier implementation</i> | 478 |
| Understanding MVC routing | 480 |
| Working with data in MVC..... | 482 |
| <i>Understanding POCO</i> | 483 |
| Implementing form validation in MVC | 486 |
| <i>Understanding the validation concept</i> | 487 |
| <i>Why validate on client and server-side</i> | 487 |
| <i>Validation's role in improving UI for a better experience</i> | 488 |
| <i>Validation's influence on UX and overall satisfaction</i> | 488 |
| <i>The cost of poor business validation</i> | 489 |
| <i>The advantages of effective validation in business</i> | 489 |
| Understanding ViewModel in MVC | 500 |
| <i>Making a ViewModel</i> | 501 |
| <i>ViewModel binding to Views</i> | 501 |
| Building a CRUD operation using MVC | 504 |
| <i>MVC and CRUD in ASP.NET Core</i> | 505 |
| Using Entity Framework Core with MVC..... | 514 |
| <i>Entity Framework Core's key features</i> | 515 |
| <i>Benefits of using EF Core</i> | 516 |
| <i>Key concepts of DbContext, DbSet, Migrations</i> | 516 |

| | |
|--|------------|
| <i>Our gateway to database operations with DbContext</i> | 516 |
| <i>Managing entity collections with DbSet</i> | 516 |
| <i>Changing our database schema with Migrations</i> | 517 |
| <i>Configuring Entity Framework</i> | 518 |
| <i>Code-first vs. database-first approaches</i> | 519 |
| <i>Code-first methodology</i> | 520 |
| <i>Database-first strategy</i> | 520 |
| <i>Understanding LINQ queries with Entity Framework Core</i> | 522 |
| <i>LINQ query creation</i> | 523 |
| <i>Setting up the development environment</i> | 524 |
| Producing and consuming APIs using MVC | 525 |
| <i>Making APIs in ASP.NET Core using MVC</i> | 525 |
| <i>Create a folder named WebApi in the root project</i> | 525 |
| Testing in MVC applications..... | 531 |
| Conclusion..... | 538 |
| 14. Mastering Web Services | 539 |
| Introduction..... | 539 |
| Structure..... | 540 |
| Objectives | 540 |
| Introduction to web services in .NET | 540 |
| Creating RESTful APIs with ASP.NET Core..... | 541 |
| <i>Exploring RESTful APIs</i> | 542 |
| <i>RESTful API history</i> | 542 |
| Consuming web services in .NET | 546 |
| Working with gRPC in .NET | 548 |
| <i>Key points and benefits</i> | 549 |
| <i>Using gRPC as a service</i> | 549 |
| <i>Useful applications</i> | 550 |
| <i>Server implementation in C#</i> | 550 |
| Introduction to GraphQL and .NET | 553 |
| Understanding HTTP Client in .NET | 556 |
| <i>Basics of HttpClient</i> | 556 |
| <i>Advanced features</i> | 557 |
| <i>Best practices</i> | 557 |

| | |
|---|------------|
| Security considerations in web services..... | 559 |
| <i>Understanding the content security policy header</i> | <i>562</i> |
| Understanding API versioning | 563 |
| Implementing pagination in APIs..... | 565 |
| <i>Understanding pagination</i> | <i>566</i> |
| <i>Pagination techniques</i> | <i>566</i> |
| <i>Implementing pagination.....</i> | <i>567</i> |
| API testing with Postman and Swagger | 568 |
| <i>Understanding APIs in .NET and C#</i> | <i>569</i> |
| <i>Swagger for API documentation and testing.....</i> | <i>569</i> |
| Conclusion..... | 571 |
| Exercises..... | 572 |
| <i>Answers.....</i> | <i>573</i> |
| 15. Blazor for UI Development..... | 575 |
| Introduction..... | 575 |
| Structure..... | 575 |
| Creating Blazor components..... | 576 |
| <i>Additional points to consider</i> | <i>576</i> |
| <i>Understanding Blazor components.....</i> | <i>576</i> |
| <i>Blazor components</i> | <i>577</i> |
| Key features of Blazor components..... | 577 |
| <i>Creating a Blazor server-side project.....</i> | <i>578</i> |
| <i>Creating a Blazor client-side project</i> | <i>578</i> |
| <i>Using solution properties</i> | <i>579</i> |
| <i>Multiple startup projects.....</i> | <i>579</i> |
| <i>Blazor prerequisites</i> | <i>580</i> |
| Understanding the Blazor project structure | 580 |
| Root directory/ Program.cs | 581 |
| Main() method's dual role | 581 |
| Blazor building blocks | 582 |
| Navigating the Blazor router | 584 |
| <i>Configuring the router</i> | <i>585</i> |
| <i>Navigating the found routes.....</i> | <i>585</i> |
| <i>Handling unfound routes.....</i> | <i>585</i> |

| | |
|--|-----|
| <i>Shared folder</i> | 585 |
| <i>Structuring styles .css companion</i> | 586 |
| <i>wwwroot/css folder</i> | 586 |
| <i>wwwroot/index.html file</i> | 587 |
| <i>Gateway to Blazor WebAssembly wwwroot</i> | 588 |
| <i>wwwroot/js folder</i> | 588 |
| <i>_Imports.razor file</i> | 588 |
| Performance and new UIs in .NET 9 | 589 |
| <i>Server-side rendering that is static</i> | 590 |
| <i>CSR that is interactive</i> | 590 |
| <i>Hybrid Method using Prerendering</i> | 591 |
| <i>Configuring render modes in Program.cs</i> | 592 |
| <i>Mode of Interactive Server</i> | 592 |
| <i>Interactive WebAssembly mode</i> | 593 |
| Working with forms and validation in Blazor | 594 |
| <i>Employee edit form validation</i> | 594 |
| <i>Blazor custom form validation</i> | 597 |
| <i>Custom validation attribute example</i> | 597 |
| <i>Create a custom validation attribute</i> | 598 |
| <i>Using custom validation attribute in Blazor</i> | 598 |
| Understanding JavaScript interop in Blazor | 599 |
| <i>Unveiling JavaScript interop</i> | 600 |
| <i>Expanding horizons</i> | 600 |
| <i>Using JavaScript in Blazor</i> | 601 |
| Building a CRUD operation using Blazor | 603 |
| <i>Set up the Blazor WebAssembly project</i> | 604 |
| <i>Define the Model</i> | 604 |
| <i>Create a data service</i> | 605 |
| <i>Implement CRUD operations in the UI</i> | 606 |
| Using Entity Framework Core with Blazor | 610 |
| <i>Key features</i> | 610 |
| <i>Setting up Entity Framework Core in Blazor</i> | 611 |
| <i>Configure database connection</i> | 612 |
| <i>Performing CRUD operations</i> | 612 |

| | |
|--|------------|
| Understanding Blazor Server vs. Blazor WebAssembly..... | 614 |
| <i>Blazor hosting models</i> | 614 |
| <i>Blazor Server vs. Blazor WebAssembly</i> | 614 |
| <i>Blazor WebAssembly hosting model</i> | 615 |
| <i>Blazor Server hosting model</i> | 616 |
| <i>Grasping the distinctions of Blazor Server vs. Blazor WebAssembly</i> | 617 |
| <i>Core mechanism</i> | 617 |
| <i>Loading dynamics</i> | 618 |
| <i>Connection and interaction</i> | 618 |
| <i>Decoding Blazor Server vs. Blazor WebAssembly Dynamics</i> | 618 |
| <i>Core mechanism exploration</i> | 618 |
| <i>Loading dynamics deconstructed</i> | 619 |
| <i>Connection and interaction chronicles</i> | 620 |
| Working with routine in Blazor..... | 621 |
| <i>Basics of routing in Blazor</i> | 621 |
| <i>Static routing</i> | 622 |
| <i>Interactive routing</i> | 622 |
| Testing in Blazor applications..... | 623 |
| <i>Setting up a test environment</i> | 625 |
| <i>Passing parameters to components</i> | 626 |
| <i>Transferring components to inputs and services</i> | 628 |
| Conclusion..... | 630 |
| 16. Packaging and Deployment..... | 631 |
| Introduction..... | 631 |
| Structure..... | 631 |
| Objectives | 632 |
| Understanding .NET components and libraries..... | 632 |
| <i>Understanding .NET components</i> | 632 |
| <i>Characteristics of .NET components</i> | 632 |
| <i>Types of .NET components</i> | 633 |
| <i>Creating a simple .NET component</i> | 633 |
| <i>Using .NET libraries</i> | 633 |
| <i>Benefits of .NET components and libraries</i> | 634 |
| How to package .NET types | 634 |

| | |
|--|----------------|
| Understanding NuGet and its role in .NET | 635 |
| Publishing your .NET code for deployment | 636 |
| Working with .NET Standard and .NET Core libraries | 638 |
| Understanding assembly versioning in .NET | 640 |
| <i>Basics of assembly versioning in .NET</i> | 640 |
| <i>Assembly versioning for .NET projects</i> | 641 |
| <i>Best practices in assembly versioning</i> | 641 |
| Understanding the Global Assembly Cache | 642 |
| <i>Key concepts</i> | 642 |
| <i>How it works</i> | 642 |
| <i>Advantages</i> | 643 |
| <i>Considerations</i> | 643 |
| Strong-named assemblies in .NET | 643 |
| <i>Critical aspects of strong-named assemblies</i> | 644 |
| Using the .NET Core CLI for package management | 644 |
| Native AOT improvements in .NET 9 | 645 |
| Best practice in packaging and distributing .NET libraries | 648 |
| Conclusion | 649 |
| Index | 651-663 |

CHAPTER 1

Introduction to C# 13 and .NET 9

Introduction

Welcome to our journey into the world of C# (pronounced C-Sharp) and .NET (pronounced dot-Net) programming. This opening chapter aims to set the stage for our coding study, helping you navigate the initial, crucial steps of setting up our development environment and knowledge of C# programming. We start exploring the process of selecting an appropriate **integrated development environment (IDE)** and guide you through installing the .NET **software development kit (SDK)**, an essential tool in our C# toolkit.

We will also understand the fundamental understanding of the .NET platform, a versatile framework central to C# programming. Moreover, we will delve into the diverse types of applications that can be crafted using C#, showcasing the extensive adaptability of this programming language.

Structure

This chapter covers the following topics:

- Introduction to C# and .NET
- Choosing the right tools for C# programming
- Understanding application types in C#
- Deploying cross-platform applications

- Role of integrated development environments
- .NET Framework vs. .NET Core
- Overview of C# and .NET ecosystem
- Installing and configuring Visual Studio
- Exploring Visual Studio Code for C# development

Objectives

In this chapter, we aim to provide you with a comprehensive understanding of C# and .NET programming. Throughout this section, you will gain familiarity with the C# language and the .NET platform, learning about their functionalities, advantages, and their place in the broader programming landscape. We will guide you through setting up your development environment, including the installation and configuration of essential tools like Visual Studio, empowering you to select the right tools for various C# programming tasks, thereby enhancing your productivity and efficiency. Additionally, you will explore the diverse range of applications that can be developed using C#, from desktop and web applications to mobile and cloud-based solutions. Another crucial aspect will be to understand the intricacies involved in deploying cross-platform applications, along with platform-specific configurations and deployment techniques. By comprehending the role and utility of IDEs in streamlining the software development process, you will be better equipped to make informed decisions.

Furthermore, you will gain insights into the key differences between the .NET Framework and .NET Core, aiding you in choosing the appropriate platform for your specific development needs. We will also provide an overview of the C# and .NET ecosystem, acquainting you with related frameworks, tools, and libraries such as **.NET Multi-platform App UI (.NET MAUI)**, Unity, and ASP.NET. Finally, you will receive hands-on guidance for installing and configuring Visual Studio, preparing you to embark on your development journey. By the end of this chapter, you will have established a solid foundation in C# and .NET programming, empowering you to create and deploy your own applications confidently.

Introduction to C# and .NET

C# is a modern, in constant evolution, open-sourced, object-oriented programming language developed by *Microsoft*, envisioned as a part of their **.NET initiative**. The language was designed to offer a mix of the best features from languages like Java™ and C++ while incorporating unique elements that deliver power and development flexibility.

Much like Java™, C# is a statically typed, object-oriented language, emphasizing safety and simplicity in its syntax. Both languages share similarities like garbage collection, exceptions, and the extensive use of libraries for different functionalities. They also provide capabilities for threading and synchronization, which are vital for modern, multi-threaded software.

However, there are differences, too. One significant distinction is that while Java™ was built with a *write once, run anywhere* philosophy and is thus platform-independent, C# was primarily designed for Windows development. But with the introduction of .NET Core and its evolution into .NET 5 and beyond, the latter has become increasingly cross-platform, supporting Android, Linux, iOS, and macOS.

The .NET, or Microsoft .NET initiative, is a comprehensive software development framework introduced by Microsoft in the late 1990s and officially launched in 2002. It represented a strategic initiative to provide developers with a unified, cohesive environment for developing applications that could run on Windows platforms and beyond.

At the core of the .NET initiative was the idea of language interoperability, which would allow different high-level languages to communicate seamlessly with one another. Using the **Common Language Runtime (CLR)** and the **.NET Framework Class Library (FCL)**, developers could write applications in various .NET compatible languages, such as C#, Visual Basic .NET, and F#.

Tip: Interoperability refers to the ability of different systems, devices, or applications to connect and communicate in a coordinated way without any effort from the end user.

Another significant aspect of the .NET initiative was the concept of managed code. Managed code is executed by the .NET CLR, which offers services like garbage collection, exception handling, and security, simplifying application development by automating routine or complex tasks.

The .NET Framework also introduced the idea of **software as a service (SaaS)**, highlighting the web services aspect of development. It is a model where the software is provided as a service over the Internet, which can be used by other software applications, irrespective of platform or language.

The .NET Framework, from 1.0 to 4.8, is closely tied to the Windows operating system, forming the backbone of many applications from Microsoft and other enterprises. Microsoft continues to maintain .NET Framework 4.8 despite the arrival of its new brother, .NET Core.

Hence, the .NET initiative represented a significant shift in the approach to software development by Microsoft, emphasizing interoperability, service-oriented architecture, and managed code, making application development more streamlined and efficient.

C# programming and the .NET platform form a potent combination. They have continued to evolve from their beginnings to the present day with a new **long-term support (LTS)** version every two years, offering programmers a robust, versatile, and efficient environment for creating powerful, modern applications. The LTS means it gets updated and receives security fixes for three years after the official launch. A non-LTS version is released every two years, with one year of support. So, we have one new version of .NET every year. Despite sharing roots and similarities with Java™, the pair has charted its

unique path in the programming world. With their recent cross-platform capabilities, they are showing no signs of slowing down.

What is new in C# 13

In this chapter, we will see the innovative elements that C# 13 introduces.

Let us briefly describe how it has grown since its first release. Generics were introduced with C# 2.0, allowing us to design classes, interfaces, and functions with placeholders for the data type they store or use. It enabled more type-safe collections and increased performance by eliminating boxing and unboxing requirements. Iterators and partial types were also implemented, improving how we wrote code and organized our projects. C# 3.0 introduced many new improvements, the most notable of which was **Language-Integrated Query (LINQ)**, which provided an SQL-like syntax for querying data sets directly from C#.

C# 4.0 improved interoperability with other languages and systems, making it more dynamic and versatile. Key features include the dynamic type, which allows operations to avoid compile-time type verification and defer binding until runtime. It made coding with COM objects, dynamic programming languages, and reflection easier and less error-prone.

C# 5.0 added the **async** and **await** keywords, significantly simplifying asynchronous programming. These keywords enabled developers to construct asynchronous code that is easy to read and maintain while approximating synchronous code in structure. C# 6.0 and 7.0 included syntactic sugar and new capabilities such as expression-bodied members, pattern matching, and tuples, reducing code complexity and boosting readability and conciseness. C# 8.0 included nullable reference types and more powerful pattern-matching capabilities, which improved the language's type safety and control flow. C# 9.0 and 10.0 were designed to make the language more concise and expressive by introducing records for immutable data structures, init-only properties, and pattern-matching improvements. C# 11, which builds on these advancements and becomes the language of the year 2023.

C# 13 introduces several new features and changes to improve the programming experience, streamline syntax, and give additional functionality.

The primary constructors' feature, which has been expanded to be usable in any class and struct, is a significant addition. Unlike prior versions, which restricted primary constructors to record types, developers can now use this feature more liberally, allowing for more straightforward and concise class and struct definitions.

Collection expressions are another novel feature in C# 13. This feature reveals a more concise syntax that facilitates the generation of common collection values, encouraging code brevity and enhancing readability. This improved syntax is intended to give developers a more efficient way of working with collections.

C# 13 improves lambda expressions with default lambda parameters. This enables developers to assign default values to lambda expression parameters. This enhancement offers flexibility and simplicity to the definition and use of lambda expressions in diverse circumstances.

Furthermore, an alias of any type is a new feature introduced with Visual Studio 17.6 Preview 3. This feature increases the language's adaptability by enabling more robust and flexible type aliasing.

Inline arrays were also implemented to the C# 13 runtime as part of the previous .NET 9 preview. This enhancement enhances the language's array handling features, resulting in a more integrated and fluid array utilization experience.

Finally, interceptors are introduced as an experimental feature, providing new code generation and modification opportunities. Interceptors allow generators to reroute code, allowing for context-specific optimization and improving generated code functionality and efficiency. This broadens the scope of code modification and optimization options in C# programming.

The primary constructors extend to all classes and structs, letting us add parameters directly to class declarations for property initialization and method/property accessor usage.

Lambda expressions now support default parameter values, streamlining methods, and local function argument handling. The using alias directive no longer limits to named types. Creating semantic aliases for tuples, arrays, pointers, and other types is now possible.

Choosing the right tools for C# programming

Choosing the right tools can drastically impact our efficiency, code quality, and work. Each tool serves a specific purpose and choosing which tool to use can mean the difference between a smooth workflow and a choppy one. Let us have a detailed look at the essential pillars of C# programming tools.

Integrated development environment

The IDE is the platform where we write our code. Microsoft's Visual Studio is an excellent IDE for C# programming. It is a fully featured tool that offers an integrated debugger, a code editor with IntelliSense (an auto-completion tool to increase speed and accuracy), support for various project configurations, and much more. However, it is possible to use a lighter solution using Visual Studio Code. It is a more streamlined editor that still supports a wide range of languages and has robust community support for extensions.