Complete C# Masterclass

Your progress

Share

What are Inner Classes in C#?

Inner Classes in C#

Introduction

In this section of the course, we have explored Object-Oriented Programming (OOP) concepts such as classes, objects, and inheritance. However, one aspect that was not covered in depth in the video lectures is the concept of Inner Classes. To ensure a complete understanding of C# and its capabilities, this article will introduce Inner Classes, explain their significance, and walk through their usage with clear examples.

This article is structured to first explain what Inner Classes are, why they are useful, and how they compare to other class structures. We will then go step by step through syntax, examples, best practices, and common pitfalls to ensure a complete grasp of the topic.

1. What is an Inner Class?

An Inner Class (also known as a nested class) is a class that is declared inside another class. This means that an inner class exists within the scope of an outer class. It can be useful when a class is only relevant within the context of another class.

Office departments?

Imagine you are organizing a corporate office. The office itself represents an outer class, while different departments within the office represent inner classes. Each department (inner class) functions within the office (outer class) but is not meant to be used separately. The departments depend on the office, just like an inner class depends on its outer class.

2. Declaring and Using Inner Classes

Basic Syntax

An inner class is declared inside another class. Here’s a simple example of how to define and use an inner class:

```
using System;

public class OuterClass
{
    private string outerField = "I am from OuterClass";

    public class InnerClass
    {
        public void DisplayMessage()
        {
            Console.WriteLine("Hello from InnerClass");
        }
    }
}

class Program
{
    static void Main()
    {
        // Creating an instance of the inner class
        OuterClass.InnerClass innerObject = new OuterClass.InnerClass();
        innerObject.DisplayMessage();
    }
}
```

Output:

```
Hello from InnerClass
```

Key Takeaways from this Example:

- **InnerClass is defined inside OuterClass.**
- **The inner class does not have direct access to OuterClass's members.**
- **It can be instantiated using OuterClass.InnerClass.**

3. Accessing the Outer Class Members

An inner class can access members of the outer class if they are marked as public or protected, or if it has a reference to the outer class.

```
using System;

public class OuterClass
{
    private string outerField = "I belong to OuterClass";

    public class InnerClass
    {
        private OuterClass outer;

        public InnerClass(OuterClass outer)
        {
            this.outer = outer;
        }

        public void DisplayOuterField()
        {
            Console.WriteLine(outer.outerField);
        }
    }
}

class Program
{
    static void Main()
    {
        OuterClass outerObject = new OuterClass();
        OuterClass.InnerClass innerObject = new OuterClass.InnerClass(outerObject);
        innerObject.DisplayOuterField();
    }
}
```

Output:

```
I belong to OuterClass
```

4. Why Use Inner Classes?

Inner classes can be beneficial in specific scenarios:

- **Encapsulation: Inner classes help group related logic together, improving readability and maintainability.**
- **Restricting Scope: If a class is only meant to be used inside another class, it makes sense to keep it enclosed.**
- **Better Organization: When a class is tightly coupled to another class, defining it as an inner class can improve code structure.**

When to Use It?

Use inner classes when: ✓ The class is only relevant to its enclosing class. ✓ You want to improve encapsulation and avoid cluttering the global namespace. ✓ The inner class requires access to private members of the outer class.

5. Comparing Inner Classes with Other Class Types

Inner Classes vs. Regular Classes

Feature Inner Class Regular Class Scope Limited to its outer class Available throughout the project Encapsulation Higher Lower Readability Better for related classes Can be scattered Access to Outer Class Yes, if referenced No

Inner Classes vs. Static Nested Classes

Inner classes should not be confused with static nested classes. A static nested class does not require an instance of the outer class.

```
public class OuterClass
{
    public static class StaticNestedClass
    {
        public static void ShowMessage()
        {
            Console.WriteLine("Hello from Static Nested Class");
        }
    }
}

class Program
{
    static void Main()
    {
        OuterClass.StaticNestedClass.ShowMessage();
    }
}
```

Key Difference: A static nested class is independent of an instance of the outer class, whereas an inner class depends on it.

6. Best Practices and Common Mistakes

Best Practices

✅ Use inner classes only when they are strongly related to the outer class.

✅ Keep inner classes private unless external access is necessary.

✅ Use static inner classes if they don't require an outer instance.

✅ Ensure clean separation of responsibilities.

Common Mistakes

❌ Overusing inner classes: If a class can exist independently, it should not be an inner class.
❌ Accessing outer class members incorrectly: Use a reference to the outer class when accessing private members. ❌ Ignoring encapsulation: Avoid exposing inner class functionality unless required.

7. Conclusion

Inner classes in C# provide a powerful way to encapsulate logic that is closely tied to an outer class. By using inner classes appropriately, we can improve code organization, maintainability, and encapsulation.

While not used frequently in everyday programming, inner classes are a useful tool for structuring code efficiently when needed. If you encounter a scenario where a class should only be used within another class, consider using an inner class.

If you have any questions, feel free to ask in the Q&A section.

Happy coding! 🎉

Course content

Course content

Overview

Q&AQuestions and answers

Notes

Announcements

Reviews

Learning tools

Section 1: UPDATED: Introduction, Overview of Visual Studio, DataTypes And Variables

51 / 56 | 3hr 6min51 of 56 lectures completed3hr 6min

Section 2: UPDATED: Making Decisions

20 / 28 | 1hr 33min20 of 28 lectures completed1hr 33min

Section 3: UPDATED: Loops

22 / 24 | 1hr 37min22 of 24 lectures completed1hr 37min

Section 4: UPDATED: Functions and Methods

17 / 20 | 1hr 34min17 of 20 lectures completed1hr 34min

Section 5: UPDATED: Object Oriented Programming (OOP)

18 / 43 | 3hr 10min18 of 43 lectures completed3hr 10min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

109. Objects Intro

2min

- **Lecture completed. Progress cannot be changed for this item.**

Play

110. Introduction To Classes And Objects

3min

- **Lecture completed. Progress cannot be changed for this item.**

Play

111. Creating our First own Class

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

112. Member Variable and Custom Constructor

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

113. Properties - Autogenerated - to protect our member variable

6min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

114. Defining how a property is set

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

115. Modifying the Get of our Property Part 1

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

116. Modifying the Get of our Property part 2

5min

- **Lecture completed. Progress cannot be changed for this item.**

Play

117. Having Multiple Constructors

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

118. Default Constructor and Use Cases

6min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Start

Quiz 12: Understanding Constructors

- **Lecture completed. Progress cannot be changed for this item.**

Play

119. Methods in Classes

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

120. Methods in Classes in more detail

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Start

121. Expression Bodied Members in C#

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

122. What are Inner Classes in C#?

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

123. Partial Classes and Methods

3min

- **Lecture completed. Progress cannot be changed for this item.**

Play

124. Optional Parameters

4min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

125. Named Parameters

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

126. Operator Overloading in C#

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

127. Passing Arguments by Value and by Reference

4min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

128. Computed Properties and No Constructor

3min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

129. Static Methods

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 10: Using Static Methods

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

130. Static Fields

3min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

131. Static Keyword Considerations

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

132. The is Operator and the as Operator in C#

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

133. Public and Private Keywords

5min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

134. ID Key and readonly

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

135. Read Only Properties

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 11: Working with Read-Only Properties

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

136. Write Only Properties

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

137. Const and ReadOnly

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Quiz 13: Working with Read-Only Properties

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

138. Quiz Project Introduction

4min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

139. QuizApp - Question Class

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

140. Keyword This

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

141. Displaying Questions

6min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

142. Displaying Answers, Console.Write and Console.ForegroundColor

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

143. Getting the UserInput and checking if it is right

6min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

144. Displaying Multiple Questions and if we are right or wrong

8min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

145. Displaying the Results

8min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

146. CHEATSHEET - Object Oriented Programming in C#

0min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 12: ADVANCED EXERCISE: Creating a Class with Properties and Methods

Section 6: UPDATED: Collections in C#

0 / 27 | 2hr 1min0 of 27 lectures completed2hr 1min

Section 7: UPDATED: Error Handling

0 / 14 | 45min0 of 14 lectures completed45min

Section 8: UPDATED: Inheritance

0 / 22 | 1hr 21min0 of 22 lectures completed1hr 21min

Section 9: UPDATED: Interfaces and Polymorphism

0 / 24 | 1hr 22min0 of 24 lectures completed1hr 22min

Section 10: UPDATED: Structs in C#

0 / 9 | 58min0 of 9 lectures completed58min

Section 11: UPDATED: Events and delegates

0 / 14 | 1hr 21min0 of 14 lectures completed1hr 21min

Section 12: UPDATED: Regular Expressions

0 / 11 | 43min0 of 11 lectures completed43min

Section 13: WPF - Windows Presentation Foundation

0 / 42 | 2hr 31min0 of 42 lectures completed2hr 31min

Section 14: WPF Project - Currency Converter - Part 1

0 / 8 | 1hr 14min0 of 8 lectures completed1hr 14min

Section 15: Using Databases With C#

0 / 12 | 2hr 2min0 of 12 lectures completed2hr 2min

Section 16: WPF Project - Currency Converter - Part 2

0 / 9 | 1hr 31min0 of 9 lectures completed1hr 31min

Section 17: Linq

0 / 13 | 2hr 18min0 of 13 lectures completed2hr 18min

Section 18: WPF Project - Currency Converter with GUI Database and API - Part 3

0 / 3 | 31min0 of 3 lectures completed31min

Section 19: The exercises for your coding interviews

0 / 4 | 5min0 of 4 lectures completed5min

Section 20: C# Clean Code

0 / 24 | 1hr 37min0 of 24 lectures completed1hr 37min

Section 21: C# Generics

0 / 18 | 1hr 38min0 of 18 lectures completed1hr 38min

Section 22: Threads

0 / 8 | 1hr 10min0 of 8 lectures completed1hr 10min

Section 23: Unit Testing - Test Driven Development TDD

0 / 36 | 3hr 24min0 of 36 lectures completed3hr 24min

Section 24: UNITY - Basics

0 / 16 | 1hr 35min0 of 16 lectures completed1hr 35min

Section 25: UNITY - Building the Game Pong with Unity

0 / 20 | 2hr 34min0 of 20 lectures completed2hr 34min

Section 26: UNITY - Building a Zig Zag Clone With Unity

0 / 18 | 2hr 11min0 of 18 lectures completed2hr 11min

Section 27: UNITY - Building a Fruit Ninja Clone With Unity

0 / 14 | 2hr 8min0 of 14 lectures completed2hr 8min

Section 28: Thank you for completing the course!

0 / 1 | 4min0 of 1 lecture completed4min

Section 29: Bonus

0 / 1 | 1min0 of 1 lecture completed1min

Explore top skills and certifications

Certifications by Issuer

- Amazon Web Services (AWS) Certifications
- Six Sigma Certifications
- Microsoft Certifications
- Cisco Certifications
- Tableau Certifications
- See all Certifications

Web Development

- Web Development
- JavaScript
- React JS
- Angular
- Java

IT Certifications

- Amazon AWS
- AWS Certified Cloud Practitioner
- AZ-900: Microsoft Azure Fundamentals
- AWS Certified Solutions Architect - Associate
- Kubernetes

Leadership

- Leadership
- Management Skills
- Project Management
- Personal Productivity
- Emotional Intelligence

Certifications by Skill

- Cybersecurity Certification
- Project Management Certification
- Cloud Certification
- Data Analytics Certification
- HR Management Certification
- See all Certifications

Data Science

- Data Science
- Python
- Machine Learning
- ChatGPT
- Deep Learning

Communication

- Communication Skills
- Presentation Skills
- Public Speaking
- Writing
- PowerPoint

Business Analytics & Intelligence

- Microsoft Excel
- SQL
- Microsoft Power BI
- Data Analysis
- Business Analysis

About