Interface in C# 12 - Complete Tutorial

# 🔷 What is an Interface in C#?

An interface is a contract that defines a set of members (methods, properties, events, or indexers) without implementing them (though this has changed slightly in C# 8+). Any class or struct that implements the interface must provide implementations for all its members.  
  
Think of an interface as a blueprint or protocol that tells a class what to do, but not how to do it.

# 🔷 Syntax

public interface IShape  
{  
 double Area();  
 double Perimeter();  
}

# 🔷 Implementing an Interface

public class Circle : IShape  
{  
 public double Radius { get; set; }  
  
 public Circle(double radius)  
 {  
 Radius = radius;  
 }  
  
 public double Area()  
 {  
 return Math.PI \* Radius \* Radius;  
 }  
  
 public double Perimeter()  
 {  
 return 2 \* Math.PI \* Radius;  
 }  
}

# 🔷 Interface vs Class – Key Differences

|  |  |  |
| --- | --- | --- |
| Feature | Interface | Class |
| Inheritance | Can only inherit from other interfaces | Can inherit from classes and implement interfaces |
| Implementation | No implementation (except default methods from C# 8+) | Contains full implementation |
| Constructors | ❌ Not allowed | ✅ Supported |
| Fields | ❌ Not allowed | ✅ Supported |
| Multiple Inheritance | ✅ Supports multiple interfaces | ❌ Only single class inheritance |
| Access Modifiers | Only public members allowed | Can have private, protected, internal, etc. |
| Instantiation | ❌ Cannot be instantiated | ✅ Can be instantiated |

# 🔷 New Features in C# 12 for Interfaces

C# 12 builds on earlier versions. Here's what's modern:

✅ Default Interface Methods (Since C# 8)  
You can now have methods with implementation in interfaces!  
  
public interface ILogger  
{  
 void Log(string message);  
  
 void LogWarning(string warning)  
 {  
 Console.WriteLine($"Warning: {warning}");  
 }  
}

✅ Static Abstract Members (C# 11) – Extended in C# 12  
Interfaces can now declare static abstract methods. This enables generic math and other advanced patterns.  
  
public interface IAddable<T>  
{  
 static abstract T Add(T a, T b);  
}  
  
public class Calculator : IAddable<int>  
{  
 public static int Add(int a, int b) => a + b;  
}

# 🔷 Rules of Interfaces in C#

• Only public members (implicitly public).

• Cannot contain: Fields, Constructors, Destructors.

• Can contain: Methods, Properties, Events, Indexers, Static methods/properties (C# 8+), Default implementations (C# 8+), Static abstract methods (C# 11+).

• Multiple inheritance allowed for interfaces.

• A class must implement all members of the interface unless it’s abstract.

• You cannot use access modifiers like private, protected, internal in interface members.

# 🔷 Limitations of Interfaces

|  |  |
| --- | --- |
| Limitation | Explanation |
| No fields | You can't store data inside an interface |
| No constructors | Interfaces can’t have constructors |
| No non-public members | Everything is public |
| No operator overloads | Unlike classes |
| No instance fields or constants | Only static ones allowed |
| Can’t be instantiated directly | Must be implemented by a class or struct |

# 🔷 Real-world Analogy

Imagine an interface is like a remote control. It has buttons (methods), but not the logic behind how the TV responds when you press them. That’s the job of the TV (class implementing the interface).

# 🔷 Example: Using Interface Polymorphism

public interface IMessageSender  
{  
 void SendMessage(string message);  
}  
  
public class EmailSender : IMessageSender  
{  
 public void SendMessage(string message)  
 {  
 Console.WriteLine($"Email: {message}");  
 }  
}  
  
public class SmsSender : IMessageSender  
{  
 public void SendMessage(string message)  
 {  
 Console.WriteLine($"SMS: {message}");  
 }  
}  
  
public class Notification  
{  
 private IMessageSender \_sender;  
  
 public Notification(IMessageSender sender)  
 {  
 \_sender = sender;  
 }  
  
 public void Notify(string message)  
 {  
 \_sender.SendMessage(message);  
 }  
}

var emailNotifier = new Notification(new EmailSender());  
emailNotifier.Notify("Meeting at 3PM");  
  
var smsNotifier = new Notification(new SmsSender());  
smsNotifier.Notify("Meeting at 3PM");

# ✅ Summary

|  |  |
| --- | --- |
| Topic | Notes |
| Interface | Blueprint for class/struct |
| Can contain | Methods, Properties, Indexers, Events |
| Cannot contain | Fields, constructors |
| C# 12 features | Default implementations, static abstract members |
| Use cases | Dependency injection, abstraction, plugin systems |
| Difference from class | No implementation, multiple inheritance, no instantiation |