**What is an Interface in C#?**

An **interface** is like a **contract** that defines **what a class must do**, but not **how it does it**.  
It contains only the **declarations** of members (methods, properties, events), but not their implementation.

In **C#**, you **cannot** declare **instance fields** (like variables with storage) inside an interface.

For example, this is **not allowed**:

public interface IAnimal

{

int age; // ❌ Compile-time error

}

**✅ What you *can* have in an interface:**

1. **Properties**

public interface IAnimal

{

int Age { get; set; } // Property, not a field

}

1. **Methods**

public interface IAnimal

{

void MakeSound();

}

1. **Events**

public interface IAnimal

{

event EventHandler OnEat;

}

1. **Constants**

public interface IAnimal

{

const int MaxAge = 20; // Allowed, but must be assigned

}

1. **Static members (C# 8.0+ / C# 11 improvements)**  
   Since C# 8 and above, interfaces can have:

* static fields
* static methods
* default method implementations

Example (C# 11+):

public interface ICalculator

{

static int DefaultValue = 10; // ✅ Allowed (static field)

static int Add(int a, int b) => a + b; // ✅ Static method

void Calculate(); // Still requires implementation

}