

Interfaces (Guia de Programação em C#)

20/02/2020 • 4 minutos para ler • 

Neste artigo

[Resumo de interfaces](#)


[Seções relacionadas](#)

[Confira também](#)

Uma interface contém definições para um grupo de funcionalidades relacionadas que uma [classe](#) não abstrata ou uma [estrutura](#) deve implementar. Uma interface `static` pode definir métodos, que devem ter uma implementação. Começando com C# 8.0, uma interface pode definir uma implementação padrão para membros. Uma interface não pode declarar dados de instância, como campos, propriedades implementadas automaticamente ou eventos semelhantes a propriedades.

Usando interfaces, você pode, por exemplo, incluir o comportamento de várias fontes em uma classe. Essa funcionalidade é importante em C# porque a linguagem não dá suporte a várias heranças de classes. Além disso, use uma interface se você deseja simular a herança para structs, pois eles não podem herdar de outro struct ou classe.

Você define uma interface usando a palavra-chave da [interface](#) como o exemplo a seguir mostra.

C#	 Copiar
<pre>interface IEquatable<T> { bool Equals(T obj); }</pre>	

O nome de uma interface deve ser um [nome identificador](#) C# válido . Por convenção, os nomes de interface começam com uma letra maiúscula `I`.

Qualquer classe ou struct que implemente a interface [IEquatable<T>](#) deve conter uma definição para um método [Equals](#) que corresponda à assinatura que a interface especifica. Como resultado, você pode contar com uma classe que implementa `IEquatable<T>` para conter um método `Equals` com o qual uma instância da classe pode determinar se é igual a outra instância da mesma classe.

A definição `IEquatable<T>` de não fornece `Equals` uma implementação para . Uma classe ou estrutura pode implementar várias interfaces, mas uma classe só pode herdar de uma única classe.

Para obter mais informações sobre classes abstratas, consulte [Classes e membros de classes abstratos e lacrados](#).

As interfaces podem conter métodos de ocorrência, propriedades, eventos, indexadores ou qualquer combinação desses quatro tipos de membros. As interfaces podem conter construtores, campos, constantes ou operadores estáticos. Para obter links para exemplos, consulte as [seções relacionadas](#). Uma interface não pode conter campos de instância, construtores de instâncias ou finalizadores. Os membros da interface são públicos por padrão.

Para implementar um membro de interface, o membro correspondente da classe de implementação deve ser público, não estático e ter o mesmo nome e assinatura do membro de interface.

Quando uma classe ou estrutura implementa uma interface, a classe ou a estrutura deve fornecer uma implementação para todos os membros que a interface declara, mas não fornece uma implementação padrão para. No entanto, se uma classe base implementa uma interface, qualquer classe que é derivada da classe base herda essa implementação.

O exemplo a seguir mostra uma implementação da interface [IEquatable<T>](#). A classe de implementação, `Car`, deverá fornecer uma implementação do método [Equals](#).

C#

 Copiar

```
public class Car : IEquatable<Car>
{
    public string Make {get; set;}
    public string Model { get; set; }
    public string Year { get; set; }

    // Implementation of IEquatable<T> interface
    public bool Equals(Car car)
    {
        return (this.Make, this.Model, this.Year) ==
            (car.Make, car.Model, car.Year);
    }
}
```

As propriedades e os indexadores de uma classe podem definir acessadores extras para uma propriedade ou o indexador que é definido em uma interface. Por exemplo, uma interface pode declarar uma propriedade que tem um acessador [get](#). A classe que

implementa a interface pode declarar a mesma propriedade tanto com um acessador `get` quanto com um [set](#). No entanto, se a propriedade ou o indexador usa a implementação explícita, os acessadores devem corresponder. Para obter mais informações sobre a implementação explícita, consulte [Implementação de interface explícita](#) e [Propriedades da interface](#).

Interfaces podem herdar de uma ou mais interfaces. A interface derivada herda os membros de suas interfaces base. Uma classe que implemente uma interface derivada deve implementar todos os membros na interface derivada, incluindo todos os membros das interfaces base da interface derivada. Essa classe pode ser implicitamente convertida para a interface derivada ou qualquer uma de suas interfaces base. Uma classe pode incluir uma interface várias vezes por meio das classes base que ela herda ou por meio de interfaces que outras interfaces herdam. No entanto, a classe poderá fornecer uma implementação de uma interface apenas uma vez e somente se a classe declarar a interface como parte da definição de classe (`class ClassName : InterfaceName`). Se a interface é herdada porque é herdada de uma classe base que implementa a interface, a classe base fornece a implementação dos membros da interface. No entanto, a classe derivada pode reimplementar qualquer membro de interface virtual em vez de usar a implementação herdada. Quando as interfaces declaram uma implementação padrão de um método, qualquer classe que implemente essa interface herda essa implementação. As implementações definidas nas interfaces são virtuais e a classe de implementação pode substituir essa implementação.

Uma classe base também pode implementar membros de interface usando membros virtuais. Nesse caso, uma classe derivada pode alterar o comportamento da interface substituindo os membros virtuais. Para obter mais informações sobre membros virtuais, consulte [Polimorfismo](#).

Resumo de interfaces

Uma interface tem as propriedades a seguir:

- Uma interface é tipicamente como uma classe base abstrata com apenas membros abstratos. Qualquer classe ou struct que implementa a interface deve implementar todos os seus membros. Opcionalmente, uma interface pode definir implementações padrão para alguns ou todos os seus membros.
- Uma interface não pode ser instanciada diretamente. Seus membros são implementados por qualquer classe ou struct que implemente a interface.
- Uma classe ou struct pode implementar várias interfaces. Uma classe pode herdar uma classe base e também implementar uma ou mais interfaces.

Seções relacionadas

- [Propriedades de interface](#)
- [Indexadores em interfaces](#)
- [Como implementar eventos de interface](#)
- [Classes e structs](#)
- [Herança](#)
- [Métodos](#)
- [Polimorfismo](#)
- [Classes e membros de classes abstract e sealed](#)
- [Propriedades](#)
- [Eventos](#)
- [Indexadores](#)

Confira também

- [C# Guia de Programação](#)
- [Herança](#)
- [Nomes de identificadores](#)

Esta página é útil?

 Yes  No
