Classes Abstratas

O **abstract** modificador indica que o que está sendo modificado tem uma implementação incompleta ou ausente. O abstract modificador pode ser usado com classes, métodos, propriedades, indexadores, e eventos.

Use o modificador abstract em uma declaração de classe para indicar que uma classe é destinada apenas a ser uma classe base para outras classes (classe abstrata). Membros marcados como abstract, ou incluídos em uma classe abstrata, devem ser implementados pelas classes que derivam da classe abstrata.

Exemplo

Neste exemplo, a classe Square deve fornecer uma implementação de `Area` porque ela deriva de ShapesClass:

C#

```
abstract class ShapesClass
{
    abstract public int Area();
}
class Square : ShapesClass
{
    int side = 0;

    public Square(int n)
    {
        side = n;
    }
    // Area method is required to avoid
    // a compile-time error.
    public override int Area()
    {
        return side * side;
    }

    static void Main()
    {
        Square sq = new Square(12);
        Console.WriteLine("Area of the square = {0}", sq.Area());
    }

    interface I
    {
        void M();
    }
    abstract class C : I
    {
        public abstract void M();
    }
```

```
}
// Output: Area of the square = 144
```

Classes abstratas tem os seguintes recursos:

- Uma classe abstrata não pode ser instanciada.
- Uma classe abstrata pode conter métodos abstratos e assessores.
- Não é possível modificar uma classe abstrata com o selada (C# Reference) modificador porque os dois modifers têm um significado oposto. O **sealed** modificador impede que uma classe sendo herdadas e o **abstract** modificador requer uma classe para ser herdada.
- Uma classe não-abstratas derivada de uma classe abstrata deve incluir reais implementações de todos os métodos abstratos e assessores herdados.

Use o **abstract** modificador em uma declaração de método ou propriedade para indicar que o método ou propriedade não contém implementação.

Métodos abstratos possuem os seguintes recursos:

- Um método Abstract é implicitamente um método virtual.
- Método Abstract declarações só são permitidas em classes abstratas.
- Porque uma declaração de método **abstract** não fornece nenhuma implementação real, não existe nenhum corpo de método; a declaração de método simplesmente acaba com um ponto-e-vírgula e não existe nenhuma chave ({ }) seguindo a assinatura. Por exemplo:
- `public abstract void MyMethod();`

  A implementação é fornecida por um método de sobrecarga Substituir (referência C#), que é um membro de uma classe não abstrata.
- É um erro utilizar os modificadores Estático ou virtual na declaração de um método abstrato.

Propriedades abstratas se comportam como métodos abstratos, exceto para as diferenças na sintaxe declaração e chamada.

- É um erro para usar o modificador abstract em uma propriedade estática.

- Uma propriedade abstrata herdada pode ser substituída em uma classe derivada, incluindo uma declaração de propriedade que usa o modificador Substituir.

Para obter mais informações sobre classes abstratas, consulte Classes abstratas e seladas e membros de classe (guia de programação do C#).

Uma classe abstrata deve fornecer implementação para todos os membros de interface.

Uma classe abstrata que implementa uma interface pode mapear os métodos interface em métodos abstratos. Por exemplo:

C#

```
interface I
{
    void M();
}
abstract class C : I
{
    public abstract void M();
}
```

Neste exemplo, a classe `DerivedClass` é derivada de uma classe `BaseClass` abstrata. A classe abstrata contém um método Abstract,`AbstractMethod`,. e duas propriedades `X` abstratas e `Y`

C#

```
    abstract class BaseClass   // Abstract class
    {
        protected int _x = 100;
        protected int _y = 150;
        public abstract void AbstractMethod();   // Abstract method
        public abstract int X    { get; }
        public abstract int Y    { get; }
    }

    class DerivedClass : BaseClass
    {
        public override void AbstractMethod()
        {
            _x++;
            _y++;
        }

        public override int X   // overriding property
        {
            get
            {
                return _x + 10;
            }
```

```
        }

        public override int Y   // overriding property
        {
            get
            {
                return _y + 10;
            }
        }

        static void Main()
        {
            DerivedClass o = new DerivedClass();
            o.AbstractMethod();
            Console.WriteLine("x = {0}, y = {1}", o.X, o.Y);
        }
    }
    // Output: x = 111, y = 161
```

No exemplo anterior se você tentar criar a classe abstrata usando uma instrução como este:

```
BaseClass bc = new BaseClass();   // Error
```

Você receberá um erro informando que o compilador Não é possível criar uma instância da classe abstrata ' BaseClass '.