C#-8 - Object Oriented Programming - Fundamentals_v1

1. Namespaces
2. Classes

1. Namespaces

Os namespaces são usados intensamente em programações de C# de duas maneiras. Primeiro, o .NET usa namespaces para organizar suas várias classes, da seguinte maneira:

```
System.Console.WriteLine("Hello World!");
```

System é um namespace e Console é uma classe nesse namespace. A using palavra-chave pode ser usada para que o nome completo não seja necessário, como no exemplo a seguir:

```
using System;

Console.WriteLine("Hello World!");
```

Estrutura básica de um programa

Cada arquivo contém zero ou mais namespaces. Um namespace contém tipos como **classes**, **structs**, **interfaces**, **enumerações** e **delegados**, ou outros namespaces.

```
// A skeleton of a C# program
using System;
namespace YourNamespace
{
    class YourClass
    {
    }

    struct YourStruct
    {
    }

    interface IYourInterface
    {
    }

    delegate int YourDelegate();

    enum YourEnum
    {
    }

    namespace YourNestedNamespace
    {
```

```
        struct YourStruct
        {
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world!");
        }
    }
}
```

Método Main

O método **Main** é o ponto de entrada de um aplicativo C#.(Bibliotecas e serviços não exigem um **Main** método como um ponto de entrada.) Quando o aplicativo é iniciado, Main o método é o primeiro método invocado.

```
using System;

class TestClass
{
    static void Main(string[] args)
    {
        Console.WriteLine(args.Length);
    }
}
```

No C# 9, você pode omitir o método principal e escrever instruções C# diretamente, como código abaixo chamamos isso de instruções de nível superior:

```
using System.Text;

StringBuilder builder = new();
builder.AppendLine("Hello");
builder.AppendLine("World!");

Console.WriteLine(builder.ToString());
```

2. Classes em C#

Um tipo que é definido como um class é um tipo de referência. Em tempo de execução, quando você declara uma variável de um tipo de referência, a variável contém o valor null até que você crie explicitamente uma instância da classe usando o new operador ou atribua a ela um objeto de um tipo compatível que pode ter sido criado em outro lugar, conforme mostrado no exemplo a seguir:

```
//Declaring an object of type MyClass.
MyClass mc = new MyClass();

//Declaring another object of the same type, assigning it the value of the first
object.
MyClass mc2 = mc;
```

Declarando classes

Para declarar uma classe é necessario definir um **modificador de acesso** seguido do termo **class** e um **identificador** , segue abaixo um exemplo:

```
//[modificador de acesso] - [class] - [identificador]
public class Customer
{
   // Fields, properties, methods and events go here...
}
```

Instância de uma classe

Embora eles sejam usados algumas vezes de maneira intercambiável, uma classe e um objeto são coisas diferentes. Uma classe define um tipo de objeto, mas não é um objeto em si. Um objeto é uma entidade concreta com base em uma classe e, às vezes, é conhecido como uma instância de uma classe.

Os objetos podem ser criados usando a new palavra-chave seguida pelo nome da classe na qual o objeto se baseará, assim:

```
Customer object1 = new Customer();
```

Referência de um objeto instanciar:

```
Customer object2;
```

Para se aprimorar mais em classes acessar a documentação no Link.