Namespace in C

In C #, namespaces are designed to keep a set of distinct names separated. The same class names that are declared in different namespaces do not conflict with each other

In **C #** , namespaces are designed to keep a set of distinct names separated. The same class names that are declared in different namespaces do not conflict with each other.

Defining a Namespace in C

A C # namespace definition starts with the **namespace** keyword followed by the namespace name, as follows:

```
 namespace  t ê n_namespace  { //ph?n khai báo code }
```

To call the enabled version of the function's namespace or variable, you add it after the namespace name as follows:

```
 t ê n_namespace . t ê n_ph ? n_t ?;
```

The following example illustrates the usage of namespaces in C #:

```
 using System ; namespace  namespaceA  { class  namespace_l  { public void  InNa
?
ây là namespaceA" ); } } } namespace  namespaceB  { class  namespace_l  { public
?
ây là namespaceB " ); } } } class Tester { static void Main ( string []  args ) {
  Console.WriteLine ("H?c C# c? b?n trên TipsMake.com.");
 Console.WriteLine ("Ví d? v? Namespace trong C#:");
 Console.WriteLine ("----------------------------------"); ns1 . InNamespace ()
```

Running the above program we will have the following result:

```
  Learn basic C # on TipsMake.com.
  Example of Namespace in C #:
  ----------------------------------
  This is namespaceA
  This is namespaceB
```

Using keyword in C

The **using** keyword indicates that the program is using given namespace names. For example, we use **System** namespaces in programs. The **Console** class is defined here. We write:

```
  Console.WriteLine ("Category C # TipsMake.com.");
```

Or you can write your full name:

```
  System.Console.WriteLine ("Category C # TipsMake.com.");
```

Using using, you will avoid having to add namespaces before the class name. Using this tells the compiler that the next code is using names in the defined namespace.

Now rewrite the example above using using directive in C #:

```
 using System ;
 using namespaceA;
using namespaceB; namespace  namespaceA  { class  namespace_l  { public void  InN
?
ây là namespaceA" ); } } } namespace  namespaceB  { class  namespace_2  { public
?
ây là namespaceB " ); } } } class Tester { static void Main ( string []  args ) {
  Console.WriteLine ("H?c C# c? b?n trên TipsMake.com.");
 Console.WriteLine ("Ví d? v? Namespace trong C#:");
 Console.WriteLine ("----------------------------------"); ns1 . InNamespace ()
```

The results of running this program remain the same, except that you do not have to add namespace names to the class names.

Nested Namespace in C

In C #, you can define a namespace within other namespaces, as follows:

```
  namespace name_namespace_1
  {
  // code declaration section
  namespace name_namespace_2
  {
  // code declaration section
  }
  }
```

You can access the members of these nested namespaces by using the dot (.) Operator in C #, as follows:

```
 using System ;
 using namespaceA;
using namespaceA.namespaceB; namespace  namespaceA  { class  namespace_l  { publi
?
ây là namespaceA" ); } }  namespace  namespaceB  { class  namespace_2  { public v
?ây là namespaceB " ); } }
 } } class Tester { static void Main ( string []  args ) { namespace_l ns1  = new
  Console.WriteLine ("H?c C# c? b?n trên TipsMake.com.");
 Console.WriteLine ("Ví d? v? Namespace trong C#:");
 Console.WriteLine ("----------------------------------"); ns1 . InNamespace ()
```

The result after running the code will be the same as the above 2 codes.

According to Tutorialspoint

Previous article: Interface in C #

Next article: Preprocessing directive in C #