C# Namespaces with Examples

In c#, **Namespace** is used to organize multiple classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) in our applications, reducing the code redundancy in our .NET applications. By using the `namespace` keyword, we can define the namespaces in our c# applications.

Following is the syntax of defining a namespace in the c# programming language.

```
namespace Namespace_Name
{
// Body of Namespace
}
```

If you observe the above syntax, we defined a namespace in c# using the `namespace` keyword.

Following is the example of defining the namespace in the c# programming language.

```
namespace Tutlane
{
     class Welcome
     {
         public void GreetMessage()
         {
            // your method code
         }
     }
}
```

If you observe the above code, we defined a namespace **Tutlane,** and it consists of a class **Welcome** as its member, and **GreetMessage()** is a method (/tutorial/csharp/csharp-methods-functions-with-examples) of the **Welcome** class.

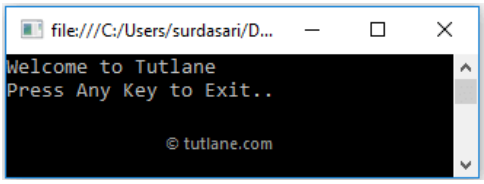
By default, **.NET Framework** provided a lot of namespaces to make the application implementation easy.

Following is the example of using the **.NET Framework** namespace "**System**" in the c# programming language.

```
namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
             System.Console.WriteLine("Welcome to Tutlane");
             System.Console.WriteLine("Press Enter Key to Exit..");
             System.Console.ReadLine();
        }
    }
}
```

In the above example, **System** is a namespace, and **Console** is a class in that namespace. **WriteLine** and **ReadLine** are the methods (/tutorial/csharp/csharp-methods-functions-with-examples) of the **Console** class.

When we execute the above c# program, it will return the result as shown below.



C# Access Namespace with using Keyword

If you observe the above example, every time we used a **System** namespace to access the **Console** class methods (/tutorial/csharp/csharp-methods-functions-with-examples) instead, you can import the **System** namespace in your application and use the associated classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) and methods (/tutorial/csharp/csharp-methods-functions-with-examples) with `using` keyword.

In c#, `using` keyword is useful to import the namespaces and provide access to the associated classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) and methods (/tutorial/csharp/csharp-methods-functions-with-examples) to use it in our .NET applications.

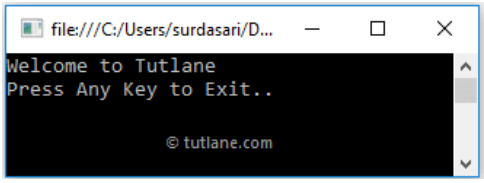
Following is the example of importing the **System** namespace with `using` keyword in c# programming language.

```
using System;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to Tutlane");
            Console.WriteLine("Press Any Key to Exit..");
            Console.ReadLine();
        }
    }
}
```

If you observe the above example, we imported a "**System**" namespace with the "**using**" keyword to access the **Console** class and associated methods (/tutorial/csharp/csharp-methods-functions-with-examples) (**WriteLine**, **ReadLine**) directly without using the **System** namespace every time.

To know more details about the above program, check this C# Hello World Program Example (/tutorial/csharp/csharp-hello-world-program-example).

When we execute the above c# program, it will return the result as shown below.



Like the **System** namespace, the **.NET Framework** has various namespaces, which are

Namespace	Description
System	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that allow you to perform basic operations such as mathematical operations and data conversation.
System.IO	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) to perform Input and Output operations.
System.Net	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that are useful to network protocols.
System.Data	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that are useful to work with ADO.Net architecture.
System.Collection	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that are useful to implement the collection of objects, such as lists.
System.Drawing	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that are useful to implement GUI functionalities.
System.Web	It contains classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) that are helpful to perform HTTP requests.

In simple words, we can say that **Namespace** is a collection of classes (/tutorial/csharp/csharp-classes-and-objects-with-examples), and classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) are the collection of objects and methods (/tutorial/csharp/csharp-methods-functions-with-examples). Using namespaces, we can easily access all the class methods (/tutorial/csharp/csharp-methods-functions-with-examples) just by importing the namespace into our application.

C# Define Multiple Namespaces

In c#, we can define and access multiple namespaces in our application with `using` keyword.

To access the custom namespace classes (/tutorial/csharp/csharp-classes-and-objects-with-examples), we need to import the custom namespace with `using` keyword and need to create an instance for that classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) in our application.

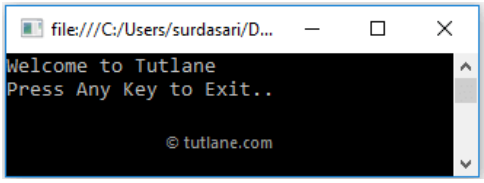
Following is the example of defining and accessing the multiple namespaces in the c# programming language.

```
using System;
using CustomNameSpace;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            Welcome w = new Welcome();
```

```
            w.GreetMessage();
            Console.WriteLine("Press Any Key to Exit..");
            Console.ReadLine();
        }
    }
}

namespace CustomNameSpace {
    class Welcome {
        public void GreetMessage() {
             Console.WriteLine("Welcome to Tutlane");
        }
    }
}
```

If you observe the above example, we created multiple namespaces and accessed the namespace **CustomNamespace** classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) in **Tutlane** namespace by creating the class instance.

When we execute the above c# program, it will return the result as shown below.



This is how you can create and access multiple namespaces in your c# applications.

Nested Namespaces in C#

In c#, a namespace can contain another namespace, and it is called a **Nested Namespace**. The nested namespace classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) and their members can be accessed by using the **dot** (**.**) operator.

Following is the syntax of defining the nested namespace in the c# programming language.

```
namespace namespace1 {
    namespace nestednamespace {
    // Nested Namespace Code
    }
}
```

If you observe the above syntax, we defined a namespace "**nestednamespace**" within another namespace "**namespace1**".

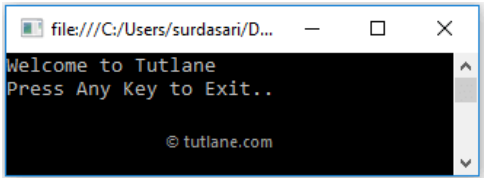
Following is the example of using nested namespace in c# programming language.

```
using System;
using CustomNameSpace.Nested;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
             Welcome w = new Welcome();
             w.GreetMessage();
             Console.WriteLine("Press Any Key to Exit..");
             Console.ReadLine();
        }
    }
}
namespace CustomNameSpace {
      namespace Nested
      {
          class Welcome
          {
              public void GreetMessage()
              {
                  Console.WriteLine("Welcome to Tutlane");
              }
          }
```

```
        }
}
```

If you observe the above code, we imported a nested namespace with `using` keyword like using `CustomNameSpace.Nested;` and accessing the nested namespace classes (/tutorial/csharp/csharp-classes-and-objects-with-examples) by creating the instance.

When we execute the above c# program, it will return the result as shown below.



This is how we can use the nested namespaces in our c# applications based on our requirements.

CONTACT US

Address: No.1-93, Pochamma Colony, Manikonda, Hyderabad, Telangana - 500089

Email: support@tutlane.com (mailto:support@tutlane.com)