A **namespace** is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

Defining a Namespace

A namespace definition begins with the keyword **namespace** followed by the namespace name as follows −

namespace namespace\_name {

// code declarations

}

To call the namespace-enabled version of either function or variable, prepend the namespace name as follows −

namespace\_name.item\_name;

The following program demonstrates use of namespaces −

[Live Demo](http://tpcg.io/SCXDku)

using System;

namespace first\_space {

class namespace\_cl {

public void func() {

Console.WriteLine("Inside first\_space");

}

}

}

namespace second\_space {

class namespace\_cl {

public void func() {

Console.WriteLine("Inside second\_space");

}

}

}

class TestClass {

static void Main(string[] args) {

first\_space.namespace\_cl fc = new first\_space.namespace\_cl();

second\_space.namespace\_cl sc = new second\_space.namespace\_cl();

fc.func();

sc.func();

Console.ReadKey();

}

}

When the above code is compiled and executed, it produces the following result −

Inside first\_space

Inside second\_space

The *using* Keyword

The **using** keyword states that the program is using the names in the given namespace. For example, we are using the **System** namespace in our programs. The class Console is defined there. We just write −

Console.WriteLine ("Hello there");

We could have written the fully qualified name as −

System.Console.WriteLine("Hello there");

You can also avoid prepending of namespaces with the **using** namespace directive. This directive tells the compiler that the subsequent code is making use of names in the specified namespace. The namespace is thus implied for the following code −

Let us rewrite our preceding example, with using directive −

[Live Demo](http://tpcg.io/5TZope)

using System;

using first\_space;

using second\_space;

namespace first\_space {

class abc {

public void func() {

Console.WriteLine("Inside first\_space");

}

}

}

namespace second\_space {

class efg {

public void func() {

Console.WriteLine("Inside second\_space");

}

}

}

class TestClass {

static void Main(string[] args) {

abc fc = new abc();

efg sc = new efg();

fc.func();

sc.func();

Console.ReadKey();

}

}

When the above code is compiled and executed, it produces the following result −

Inside first\_space

Inside second\_space

Nested Namespaces

You can define one namespace inside another namespace as follows −

namespace namespace\_name1 {

// code declarations

namespace namespace\_name2 {

// code declarations

}

}

You can access members of nested namespace by using the dot (.) operator as follows −

[Live Demo](http://tpcg.io/jW9nSY)

using System;

using first\_space;

using first\_space.second\_space;

namespace first\_space {

class abc {

public void func() {

Console.WriteLine("Inside first\_space");

}

}

namespace second\_space {

class efg {

public void func() {

Console.WriteLine("Inside second\_space");

}

}

}

}

class TestClass {

static void Main(string[] args) {

abc fc = new abc();

efg sc = new efg();

fc.func();

sc.func();

Console.ReadKey();

}

}

When the above code is compiled and executed, it produces the following result −

Inside first\_space

Inside second\_space