**Namespaces**

* A namespace defines a declarative region that provides a way to keep one set of names separate from another.
* In essence, names declared in one namespace will not conflict with the same names declared in another.
* The namespace used by the .NET Framework library (which is the C# library) is System. This is why you have included using System; near the top of every program.
* Namespaces are important because there has been an explosion of variable, method, property, and class names over the past few years. These include library routines, third-party code, and your own code.
* Without namespaces, all of these names would compete for slots in the global namespace and conflicts would arise.
* For example, if your program defined a class called Finder, it could conflict with another class called Finder supplied by a third-party library that your program uses.
* Fortunately, namespaces prevent this type of problem because a namespace restricts the visibility of names declared within it.

**Declaring a Namespace**

* A namespace is declared using the **namespace** keyword.
* The general form of **namespace** isshown here:

**namespace** *name*

{

// members

}

* Here, *name* is the name of the namespace.
* A namespace declaration defines a scope.
* Anything declared immediately inside the namespace is in scope throughout the namespace.
* Within a namespace, you can declare classes, structures, delegates, enumerations, interfaces, or another namespace.
* Here is an example of a **namespace** that creates a namespace called **Counter**.
* It localizes the name used to implement a simple countdown counter class called **CountDown**.

// Declare a namespace for counters.

**namespace Counter**

{

// A simple countdown counter.

**class CountDown**

{

int val;

**public CountDown(int n)**

{

val = n;

}

**public void Reset(int n)**

{

val = n;

}

**public int Count()**

{

if(val > 0) return val--;

else return 0;

}

}

} // This is the end of the Counter namespace.

* Notice how the class **CountDown** is declared within the scope defined by the **Counter** namespace.
* To follow along with the example, put this code into a file called **Counter.cs**.
* Here is a program that demonstrates the use of the **Counter** namespace:

// Demonstrate the Counter namespace.

**using System;**

**class NSDemo**

{

**public static void Main(String[] args)**

{

// Notice how CountDown is qualified by Counter.

**Counter.CountDown cd1 = new Counter.CountDown(10);**

int i;

do

{

i = cd1.Count();

Console.Write(i + " ");

} while(i > 0);

Console.WriteLine();

// Again, notice how CountDown is qualified by Counter.

**Counter.CountDown cd2 = new Counter.CountDown(20);**

do

{

i = cd2.Count();

Console.Write(i + " ");

} while(i > 0);

Console.WriteLine();

cd2.Reset(4);

do

{

i = cd2.Count();

Console.Write(i + " ");

} while(i > 0);

Console.WriteLine();

}

}

**The output from the program is shown here:**

10 9 8 7 6 5 4 3 2 1 0

20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

4 3 2 1 0

**Features of Namespaces**

* Namespaces Prevent Name Conflicts.
* Namespaces Are Additive.
* Namespaces Can Be Nested.

**Namespaces are Additive:**

Namespaces with same name but class name inside each must differ.

From main section of the program they can be referenced as if there is no namespace in the declared in the program.

**using System;**

**using Counter;**

**namespace Counter**

{

// A simple countdown counter.

**class CountDown**

{

int val;

**public CountDown(int n)**

{

val = n;

}

**public void Reset(int n)**

{

val = n;

}

**public int Count()**

{

if(val > 0)

return val--;

else

return 0;

}

}

}

**namespace Counter**

{

**class CountUp**

{

int val;

int target;

**public int Target**

{

get

{

return target;

}

}

**public CountUp(int n)**

{

target = n;

val = 0;

}

**public void Reset(int n)**

{

target = n;

val = 0;

}

**public int Count()**

{

if(val < target)

return val++;

else

return target;

}

}

}

**class NSDemo5**

{

**public static void Main()**

{

**CountDown cd = new CountDown(10);**

**CountUp cu = new CountUp(8);**

int i;

do

{

i = cd.Count();

Console.Write(i + " ");

} while(i > 0);

Console.WriteLine();

do

{

i = cu.Count();

Console.Write(i + " ");

} while(i < cu.Target);}}

**Namespace Prevents name Conflicts:**

The key point about a namespace is that names declared within it won’t conflict with similar names declared outside of it.

**Using System;**

**namespace ns1**

{

**class x**

{

int n, a[]=new int[n];

**public x(int size)**

{

n=size;

}

**public void sort ()**

{

//fill the array

for(int i=0;i<n;i++)

{

a[i]=C.R();

}

for(int i=0;i<n;i++)

{

for(int j=i+1;j<n;j++)

{

if(a[i]<a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

}

}

**namespace ns2**

{

**class x**

{

int n, a[]=new int[n];

**public x(int size)**

{

n=size;

}

**public void sort ()**

{

//fill the array and descending order sort.

}

}

**class demo**

{

**public static void Main(String[] args)**

{

**ns1.x** ob**= new ns1.x(10);**

ob.sort();

**ns2.x** ob**= new ns2.x(10);**

ob.sort();

}

}

}

**Namespaces Can Be Nested:**

One namespace can be nested within another.

Consider this program:

// Namespaces can be nested.

**using System;**

**namespace NS1**

{

**class ClassA**

{

**public ClassA()**

{

Console.WriteLine("constructing ClassA");

}

}

**namespace NS2**

{

// a nested namespace

**class ClassB**

{

**public ClassB()**

{

Console.WriteLine("constructing ClassB");

}

}

}

}

**class NestedNSDemo**

{

**public static void Main()**

{

**NS1.ClassA a = new NS1.ClassA();**

// NS2.ClassB b = new NS2.ClassB(); // Error!!! NS2 is not in view

**NS1.NS2.ClassB b = new NS1.NS2.ClassB();** // this is right

}

}