Understanding Namespaces

What Namespaces Are

In C#, a `namespace` is a scope in which developers organize their code, and you've been using them throughout the previous lessons.

```
using System;

namespace GettingStartedWithCSharp
{
    public class Program
    {
        public static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

```
using System;

namespace GettingStartedWithCSharp
{
    public class Program
    {
        public static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

In the example above, there are two namespaces mentioned `System` and `GettingStartedWithCSharp`.

Creating Namespaces

When you create a program in C#, the first class you create will be inside of a namespace. Naming your namespace clearly is important; finding your code later will be much easier with a clear name. In the previous example, the `Program` class was inside of the `GettingStartedWithCSharp` namespace. Since it is not being used elsewhere, you can change that namespace to something like this:

```
namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

```
namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

Notice the `.` characters are used to create nested namespaces. By using this namespace, you are effectively creating three namespaces: `GettingStartedTutorials`, `GettingStartedTutorials.CSharp`, and `GettingStartedTutorials.CSharp.NamespaceLesson`. For now, the first two only contain the next, more specific namespace, and `GettingStartedTutorials.CSharp.NamespaceLesson` contains your class, `Program`.

Using Code from Other Namespaces

You may have noticed in previous lessons `System.` was sometimes a prefix for various *classes*, and also sometimes appeared as `using System;`. This is how developers are able to access code from other namespaces. Both `System.Console` and `System.Exception` have been used previously, and, as you may have guessed based on how you create namespaces, these are from the `System` namespace.

Consider a class `Person` in a namespace `GettingStartedTutorials.CSharp.NamespaceLesson.Models` like this example:

```
namespace GettingStartedTutorials.CSharp.NamespaceLesson.Models
{
    public class Person
    {
        public string Name { get; set; }
    }
}
```

If you wanted to create an instance of this class inside of your program, you could do it in one of these two ways.

Namespace as a Class Prefix

When using a class from outside the current scope, you can prefix the class name using the namespace. Place a `.` between the two using this format `Namespace.Class`.

```
namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            var brendan = new
GettingStartedTutorials.CSharp.NamespaceLesson.Models.Person { Name = "Brendan" };
            System.Console.WriteLine($"Hello {brendan.Name}!");
        }
    }
}
```

```
using System;

namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            var brendan = new
GettingStartedTutorials.CSharp.NamespaceLesson.Models.Person { Name = "Brendan" };
            System.Console.WriteLine($"Hello {brendan.Name}!");
        }
    }
}

namespace GettingStartedTutorials.CSharp.NamespaceLesson.Models
{
    public class Person
    {
        public string Name { get; set; }
    }
}
```

The `GettingStartedTutorials.CSharp.NamespaceLesson.Models.Person` tells C# that you want to use the `Person` class declared in the `GettingStartedTutorials.CSharp.NamespaceLesson.Models` namespace.

Tip {.tip .visualstudio}
If you run this code in Visual Studio, the `using System;` line will appear somewhat faded, since it is not used in this code.

Tip {.tip .visualstudiocode}
If you run this code in Visual Studio Code, it will underline the `using System;`, since it is not used in this code.

Declaring Namespace to be in Scope

Sometimes you want to make a namespace available in your code by adding it to the current scope. You do this by declaring that intent at the top of your file in a *using statement*. Just type the `using` keyword followed by the namespace name you want to include in the current scope.

Tip {.tip .java}
In C#, the `using` keyword works similarly to how the `import` keyword from Java works.

```
using GettingStartedTutorials.CSharp.NamespaceLesson.Models;

namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            var brendan = new Person { Name = "Brendan" };
            System.Console.WriteLine($"Hello {brendan.Name}!");
        }
    }
}
```

```
using GettingStartedTutorials.CSharp.NamespaceLesson.Models;

namespace GettingStartedTutorials.CSharp.NamespaceLesson
{
    public class Program
    {
        public static void Main()
        {
            var brendan = new Person { Name = "Brendan" };
            System.Console.WriteLine($"Hello {brendan.Name}!");
        }
    }
}

namespace GettingStartedTutorials.CSharp.NamespaceLesson.Models
{
    public class Person
    {
        public string Name { get; set; }
```

```
    }
}
```

By declaring the namespace to be in scope, C# knows that the `Person` class used in the example is the one from the `GettingStartedTutorials.CSharp.NamespaceLesson.Models` namespace.

Note { .note }
If more than one class of the same name is included in the same scope, you will need to disambiguate the scenario by specifying all or part of the namespace. In our example, `NamespaceLesson.Models.Person` would be enough.

Note { .note }
You can specify a special name to use for one of your duplicately named classes in the current scope with a special type of using statement. `using NamespacePerson = GettingStartedTutorials.CSharp.NamespaceLesson.Models.Person;` will let you use `NamespacePerson` in your program, so C# knows which `Person` class you intend to use.

Tip {.tip .cpp}
The `using` statement locally renaming a type should feel similar to how `typedef` works in `C++`.