Microsoft

LESSON 2.3

10754 Microsoft .NET Fundamentals

Understand .NET Namespaces

LESSON 2.3

Lesson Overview

Understand .NET Namespaces

In this lesson, you will:

- Explore .NET namespaces.
- Use namespaces in an application.

Guiding Questions

1. Why does the Microsoft® .NET Framework use namespaces?
2. How can developers use namespaces in their applications?

Anticipatory Set

What is the purpose of the various *using* statements at the top of many Microsoft C# classes?

Namespaces

Namespaces are organizational units used for two purposes:

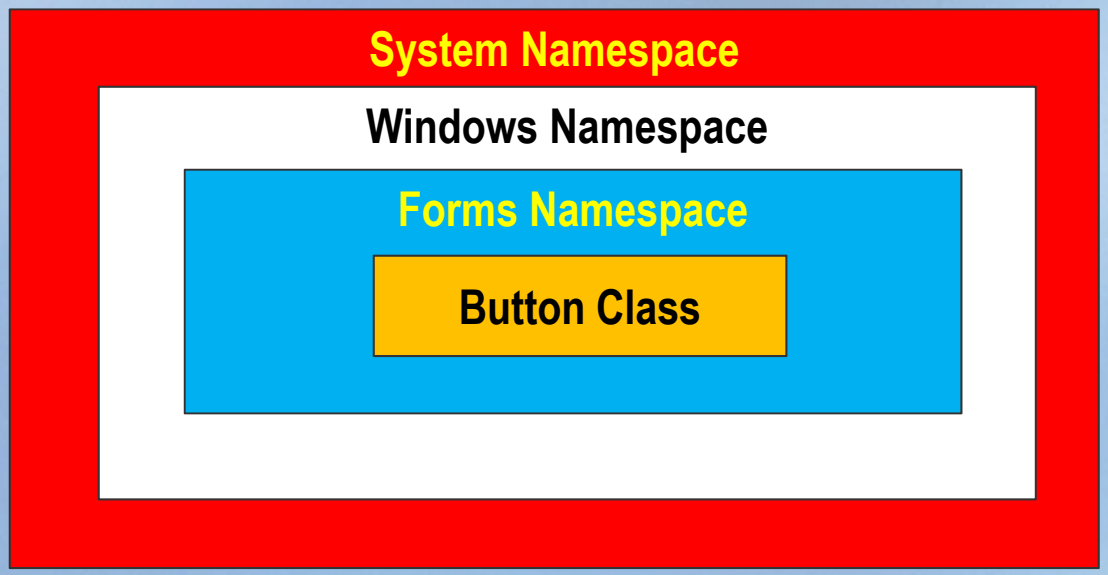
- To organize the many classes within the .NET Framework
- To help control scope and avoid name collisions

Name Collisions

- Sometimes a developer of a class library will encounter difficulties caused by the use of similar names in another library.
  - For example, a developer writing a library for handling game pad input might have several classes and methods related to the gamepad buttons; however, the name *Button* is already used for buttons on Microsoft Windows forms.
  - This type of conflict is called a **name collision**.
- Namespaces avoid name collisions by allowing the developer to use a *fully qualified name* to reference a class.

Namespace Example

- The `Button` type commonly used in Forms applications is nested within several namespaces, as follows:



- The fully qualified name for the `Button` class is `System.Windows.Forms.Button`.

Fully Qualified Names

- *Fully qualified names* are object references that are prefixed with the name of the namespace in which the object is defined.
  - In our previous example, the gamepad developer can create a class called `Button`; if he or she wishes to use a Forms button as well, he can use the following fully qualified name when referencing that particular button control:

    ```
    System.Windows.Forms.Button
    ```

  - This ensures that the compiler will use the `Button` control rather than the class that the developer implemented.
- *Note:* You can use classes defined in other projects if you create a reference to the class (by choosing Add Reference from the Project menu) and then use the fully qualified name for the object in your code.

Using/Imports Directives

- Because using fully qualified names can be cumbersome (as with the `Button` example), .NET languages provide a way to shorten references.
- In C#, this is accomplished by adding `using` directives at the top of your code.
- In Visual Basic®, it is accomplished using the `Imports` directive.
- So long as there are no name collisions, these strategies allow developers to refer to `ListBox`, rather than `System.Windows.Forms.Listbox`.
- **Note:** In our example, the gamepad developer must still use the fully qualified name to avoid confusion with his own `Button` class.

Namespaces as an Organizational Tool

- Namespaces can organize classes in a large programming project.
  - **Example:** the gamepad developer created a namespace for code related to buttons and a separate namespace for directional inputs or joysticks.
- Nesting provides greater organization and control of scope.

```
namespace NameSpace1
{    class Class1
     {
        class Class2   {        }
     }
   namespace Namespace2
   {
       class Class2    {        }
   }
}
```

Namespaces in .NET Applications

- When you create a new project in Visual Studio®, it automatically defines a new namespace with the same name as your project.
- By default, all classes that you add to the project within Visual Studio will use this default namespace. However, you can easily define your own namespace using the `namespace` (C#) or `Namespace` (Visual Basic) keyword.
  - **In C#:**

```
namespace MyNewNamespace
{
}
```

  - **In Visual Basic:**

```
Namespace MyNewNamespace
End Namespace
```

Lesson Review

Answer the three questions about namespaces on the Student Activity.