19

Namespaces

Introducing Namespaces

› Namespaces is a collection of classes and "other types such as interfaces, structures, delegate types, enumerations).

```
namespace FrontOffice
{
}
```

```
namespace Finance
{
}
```

```
namespace HR
{
}
```

```
namespace Inventory
{
}
```

Namespaces

```
namespace NamespaceName
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations
}
```

- Goal: To group-up classes and other types that are related to a particular project-module, into an unit.
- Accessing a type that is present inside the namespace: NamespaceName.TypeName

Nested Namespaces

- The namespace which is declared inside another namespace is called as "Nested namespace" or “Inner Namespace".
- Use nested namespaces, in order to divide the classes of a larger namespace, into smaller groups.
- Accessing Types: OuterNamespace.InnerNamespace.TypeName

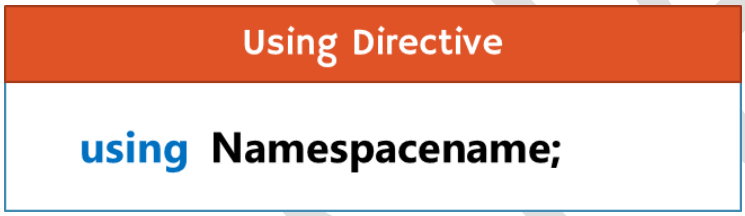
Nested Namespaces

```
namespace OuterNamespace
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations

    namespace InnerNamespace
    {
        Classes
        Interfaces
        Structures
        Delegate Types
        Enumerations
    }
}
```

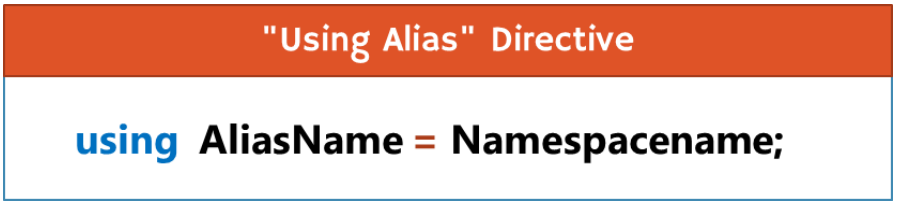
Importing Namespaces ('using' Directive)

- The "using" is a directive statement (top-level statement) that should be placed at the top of the file, which specifies the namespace, from which you want to import all the classes and other types.
- When you import a namespace, you can directly access all of its classes and other types (but not inner namespaces).
- The "using directives" are written independently for every file.
- "One using directive" can import "one namespace" only.



'using' alias name

- The "using alias" directive allows you to create "alias name" for the namespace.
- Use "using alias" directive, if you want to access long namespaces with shortcut name.
- It is much useful to access specific namespace, when there is namespace name ambiguity (two classes with same name in two different namespaces and both namespaces are imported in the same file).



‘using’ static

- The "using static" directive allows you import a static class directly from a namespace; so that you can directly access any of its methods anywhere in the current file.
- Use the "using static" directive to access methods of static class easily, without repeating the class name each time.