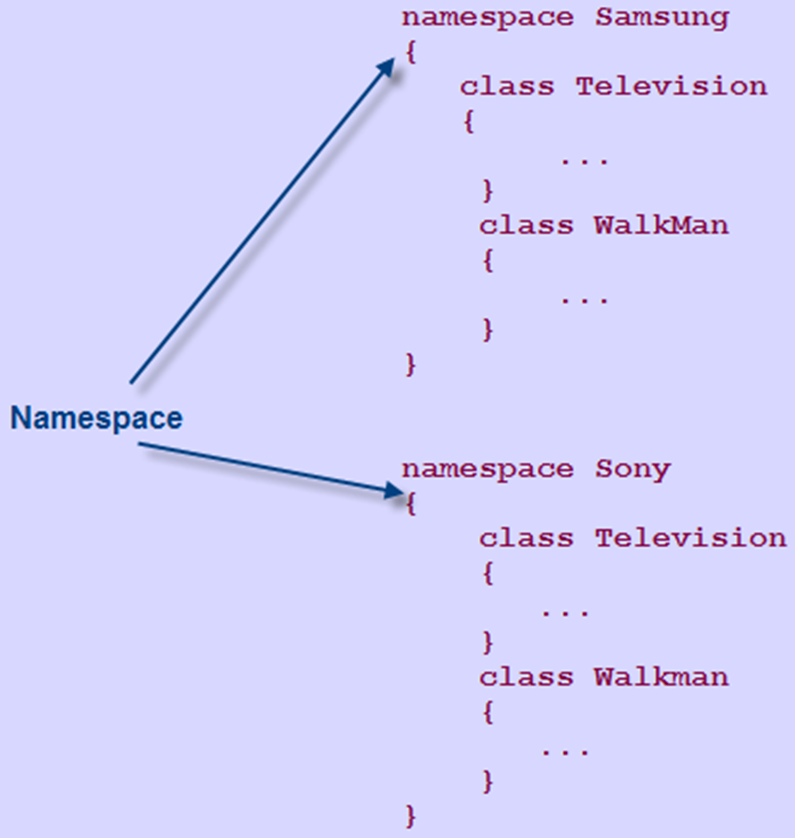
**What Is Namespace In C# ?**

* A namespace is used in C# to group classes logically and prevent name clashes between classes with identical names.
* A namespace reduces any complexities when the same program is required in another application.
* A namespace is used to organize your code and is collection of classes, interfaces, structs, enums and delegates.
* **NOTE**: if u don’t want to use namespace you can use **Fully Qualified Name (FQN)**.



**Purpose Of Namespaces**

**Scenario**

* Consider Venice, which is a city in the US as well as in Italy.
* You can easily distinguish between the two cities by associating them with their respective countries.
* Similarly, when working on a huge project, there may be situations where classes have identical names.
* This may result in name conflicts.
* This problem can be solved by having the individual modules of the project use separate namespaces to store their respective classes.
* By doing this, classes can have identical names without any resultant name clashes.

**The following code renames identical classes by inserting a descriptive prefix:**

class SamsungTelevision

{

...

}

class SamsungWalkMan

{

...

}

class SonyTelevision

{

...

}

class SonyWalkMan

{

...

}

**In Above Code,**

* The identical classes Television and WalkMan are prefixed with their respective company names to avoid any conflicts.
* There cannot be two classes with the same name.
* It is observed that the names of the classes get long and become difficult to maintain.

The following code demonstrates a solution to overcome this, by using namespaces:

namespace Samsung

{

class Television

{

...

}

class WalkMan

{

...

}

}

namespace Sony

{

class Television

{

...

}

class Walkman

{

...

}

}

In Above Code,

* Each of the identical classes is placed in their respective namespaces, which denote respective company names.
* It can be observed that this is a neater, better organized, and more structured way to handle naming conflicts.

**Using Namespaces In C#**

* C# allows you to specify a unique identifier for each namespace.
* This identifier helps you to access the classes within the namespace.
* Apart from classes, the following data structures can be declared in a namespace:

**Interfaces**

* An interface is a reference type that contains declarations of the events, indexers, methods, and properties.
* Interfaces are inherited by classes and structures and all the declarations are implemented in these classes and structures.

**Structures**

* A structure is a value type that can hold values of different data types.
* It can include fields, methods, constants, constructors, properties, indexers, operators, and other structures.

**Enumeration**

* An enumeration is a value type that consists of a list of named constants.
* This list of named constants is known as the enumerator list.

**Delegate**

* A delegate is a user-defined reference type that refers to one or more methods.
* It can be used to pass data as parameters to methods.

**Advantages & Characteristics Of Namespaces**

* A namespace groups common and related classes, structures, or interfaces, which support OOP concepts of encapsulation and abstraction.
* A namespace provides a hierarchical structure that helps to identify the logic for grouping the classes.
* A namespace allows you to add more classes, structures, enumerations, delegates, and interfaces once the namespace is declared.
* A namespace includes classes with names that are unique within the namespace.

A namespace provides the following benefits:

* A namespace allows you to use multiple classes with same names by creating them in different namespaces.
* It makes the system modular.
* The .NET Framework comprises several built-in namespaces that contain:
  + Classes
  + Interfaces
  + Structures
  + Delegates
  + Enumerations
* These namespaces are referred to as system-defined namespaces.
* The most commonly used built-in namespace of the .NET Framework is System.
* The System namespace contains classes that:
  + Define value and reference data types, interfaces, and other namespaces.
  + Allow you to interact with the system, including the standard input and output devices.

**Pre-defined Namespaces**

**Some of the most widely used namespaces within the System namespace are as follows:**

**System.Collections**

* The System.Collections namespace contains classes and interfaces that define complex data structures such as lists, queues, bit arrays, hash tables, and dictionaries.

**System.Data**

* The System.Data namespace contains classes that make up the ADO.NET architecture.
* The ADO.NET architecture allows you to build components that can be used to insert, modify, and delete data from multiple data sources.

**System.Diagnostics**

* The System.Diagnostics namespace contains classes that are used to interact with the system processes.
* This namespace also provides classes that are used to debug applications and trace the execution of the code.

**System.IO**

* The System.IO namespace contains classes that enable you to read from and write to data streams and files.

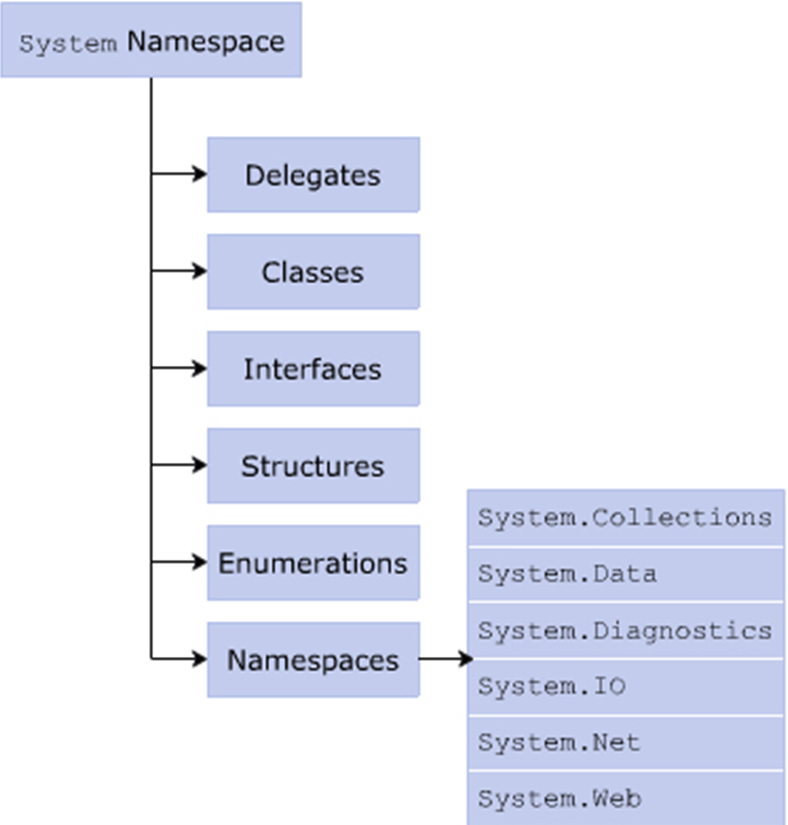
**System.Net**

* The System.Net namespace contains classes that allow you to create Web-based applications.

**System.Web**

* The System.Web namespace provides classes and interfaces that allow communication between the browser and the server.

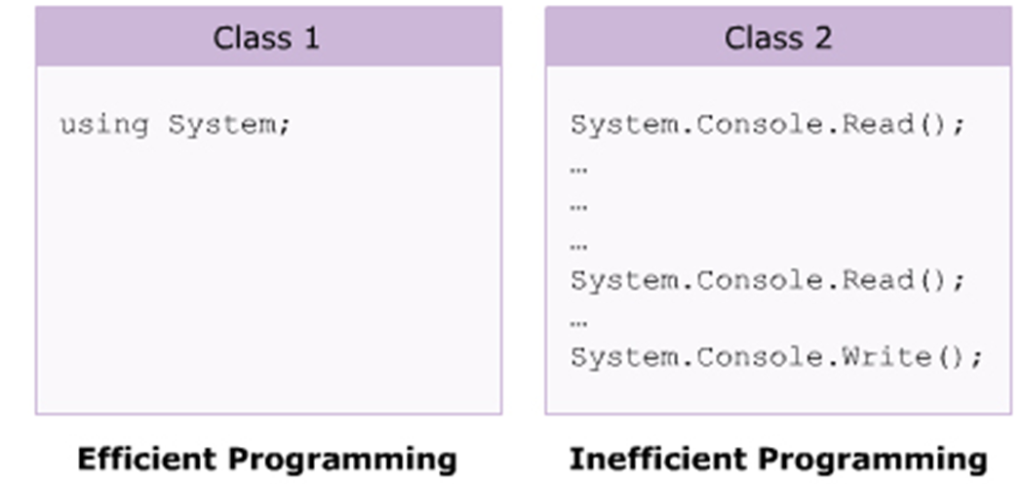
**The following figure displays some built-in namespaces:**



**System Namespace in .Net Framework**

* The System namespace is imported by default in the .NET Framework.
* It appears as the first line of the program along with the using keyword.
* For referring to classes within a built-in namespace, you need to explicitly refer to the required classes.
* It is done by specifying the namespace and the class name separated by the dot (.) operator after the using keyword at the beginning of the program.
* You can refer to classes within the namespaces in the same manner without the using keyword.
* However, this results in redundancy because you need to mention the whole declaration every time you refer to the class in the code.

**The two approaches of referencing the System namespace are:**



**Though both are technically valid, the first approach is more recommended.**

The following syntax is used to access a method in a   
system-defined namespace:

<NamespaceName>.<ClassName>.<MethodName>;

In Above Syntax,

* NamespaceName: Is the name of the namespace.
* ClassName: Is the name of the class that you want to access.
* MethodName: Is the name of the method within the class that is to be invoked.

**The following syntax is used to access the system-defined namespaces with the using keyword:**

using <NamespaceName>;

using <NamespaceName>.<ClassName>;

**In Above Syntax,**

* NamespaceName: Is the name of the namespace and it will refer to all classes, interfaces, structures, and enumerations.
* ClassName: Is the name of the specific class defined in the namespace that you want to access.

**The following code demonstrates the use of the using keyword with namespaces:**

using System;

class World

{

static void Main(string[] args)

{

Console.WriteLine(“Hello World”);

}

}

**In Above Code,**

* The System namespace is imported within the program with the using keyword.
* If this were not done, the program would not even compile as the Console class exists in the System namespace.

**Output**

Hello World

**The following code refers to the Console class of the System namespace multiple times:**

class World

{

static void Main(string[] args)

{

System.Console.WriteLine(“Hello World”);

System.Console.WriteLine(“This is C# Programming”);

System.Console.WriteLine(“You have executed a simple program of

C#”);

}

}

**In Above code,** the class is not imported, but the System namespace members are used along with the statements.

**Output**

Hello World  
This is C# Programming  
You have executed a simple program of C#

**Custom Namespaces**

* C# allows you to create namespaces with appropriate names to organize structures, classes, interfaces, delegates, and enumerations that can be used across different C# applications.
* When using a custom namespace, you need not worry about name clashes with classes, interfaces, and so on in other namespaces.
* Custom namespaces:
  + Enable you to control the scope of a class by deciding the appropriate namespace for the class.
  + Declared using the namespace keyword and is accessed with the using keyword similar to any built-in namespace.

**The following syntax is used to declare a custom namespace:**

namespace <NamespaceName>

{

//type-declarations;

}

**In Above Syntax:**

* NamespaceName: Is the name of the custom namespace.
* type-declarations: Are the different types that can be declared. It can be a class, interface, struct, enum, delegate, or another namespace.

**The following code creates a custom namespace named Department:**

namespace Department

{

class Sales

{

static void Main(string [] args)

{

System.Console.WriteLine(“You have created a custom namespace

named Department”);

}

}

}

**In Above code,**

* Department is declared as the custom namespace.
* The class Sales is declared within this namespace.

**Output**

You have created a custom namespace named Department

* Once a namespace is created, C# allows:
  + Additional classes to be included later in that namespace. Hence, namespaces are additive.
  + A namespace to be declared more than once.
* These namespaces can be split and saved in separate files or in the same file.
* At the time of compilation, these namespaces are added together.

**Important Guidelines for Creating Custom Namespaces**

* While designing a large framework for a project, it is required to create namespaces to group the types into the appropriate namespaces such that the identical types do not collide.
* Therefore, the following guidelines must be considered for creating custom namespaces:
  + All similar elements such as classes and interfaces must be created into a single namespace. This will form a logical grouping of similar types and any programmer will be easily able to search for similar classes.
  + Creating deep hierarchies that are difficult to browse must be avoided.
  + Creating too many namespaces must be avoided for simplicity.

**What Is Main Method ?**

Main method is the entry point of your application.