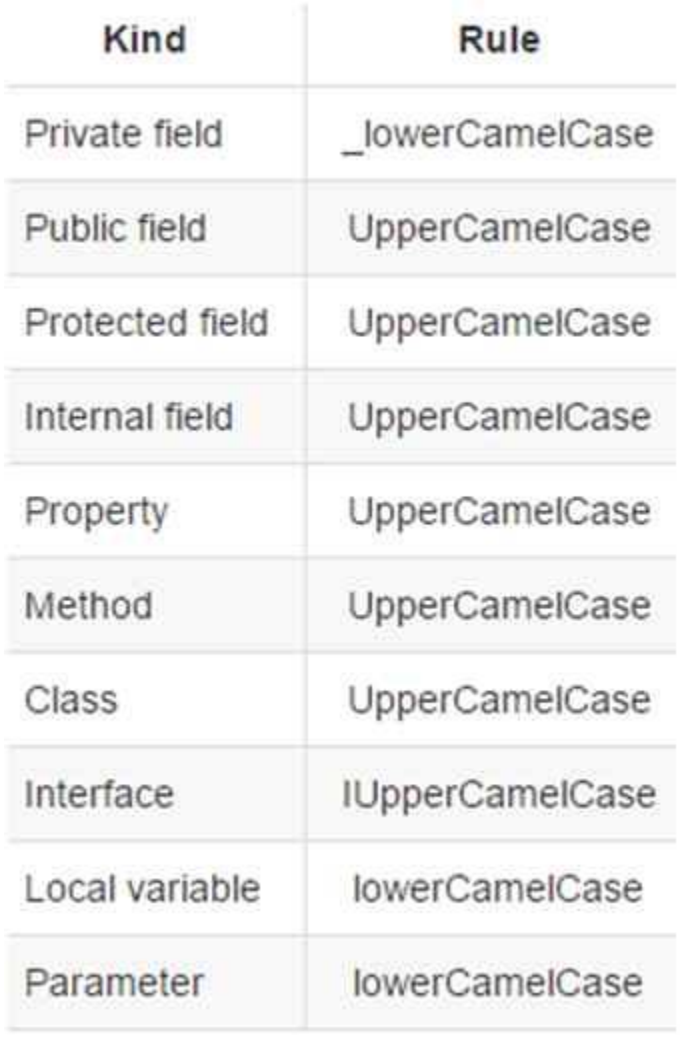
C# naming conventions are an important part of C# coding standards and best practice when you are developing a .NET applications. .NET naming conventions are standards how the naming of variables, methods, classes, and other code elements should be defined. In this article, let us learn C# naming conventions.

Terminology
There are following three terminologies are used to declare C# and .NET naming standards.

- **Camel Case (camelCase):** In this standard, the first letter of the word always in small letter and after that each word starts with a capital letter.
- **Pascal Case (PascalCase):** In this the first letter of every word is in capital letter.
- **Underscore Prefix (_underScore):** For underscore (__), the word after _ use camelCase terminology.



Kind	Rule
Private field	_lowerCamelCase
Public field	UpperCamelCase
Protected field	UpperCamelCase
Internal field	UpperCamelCase
Property	UpperCamelCase
Method	UpperCamelCase
Class	UpperCamelCase
Interface	IUpperCamelCase
Local variable	lowerCamelCase
Parameter	lowerCamelCase

Native DataType

Always use native datatype instead of .NET CTS type. For example, use **int** instead of **Int32** or **Int64**.

```
//Good
private int _salary = 100;

//Bad
private Int16 _salary = 100;
private Int32 _salary=100;
```

Class

Always use **PascalCase** for class names. Try to use noun or noun phrase for class name. Do not give prefixes. Do not use underscores.

```
public partial class About : Page
{
    //...
}
```

Methods

Always use PascalCase for method names. Use maximum 7 parameters in a method.

```
public string GetPosts(string postId)
{
    //...
}
```

Note: Don't use name as all character in CAPS.

Arguments and Local Variable

Always use **camelCase** with method arguments and local variables. Don't use Hungarian notation for variables.

```
public string GetPosts(string postId
{
    int numberOfPost = 0;
}
```

Note: Don't use abbreviations for any words and don't use underscore (_) in between any name.

Property

Use **PascalCase** for property. Never use Get and Set as prefix with property name.

```
private int _salary = 100;
public int Salary
{
    get
    {
        return _salary;
    }
    set
    {
         _salary = value;
     }
 }
```

Note: Don't use name with start with numeric character.

Interface

Always use letter "**I**" as prefix with name of interface. After letter I, use PascalCase.

```
public interface IUser
{
   /// <summary>
   /// Check user is exists or not
   /// </summary>
   /// <returns>return bool value</returns>
   bool ValidateUser();
}
```

Private Member Variable

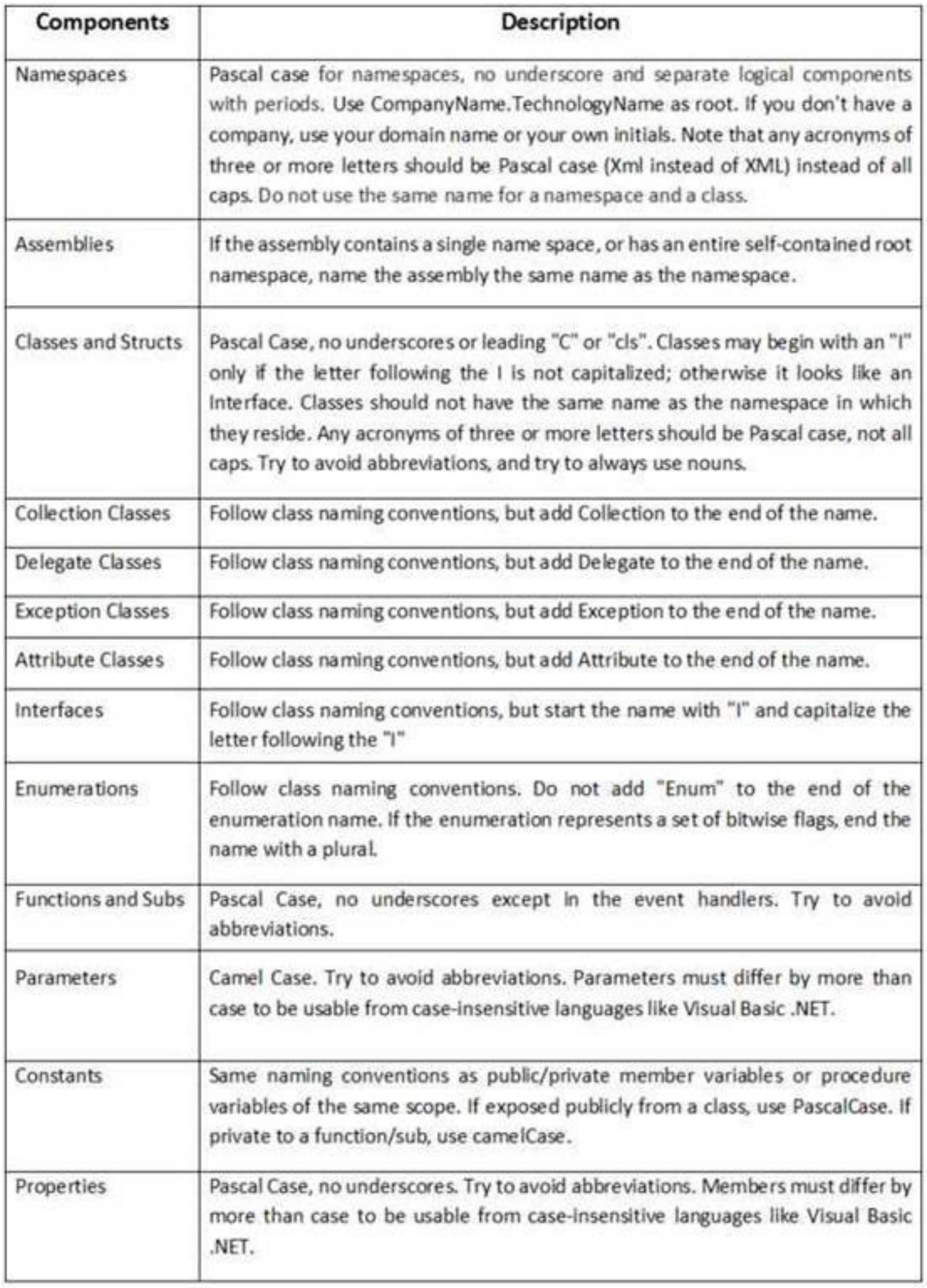
Always try to use **camelCase** terminology prefix with underscore (_).

```
private int _salary = 100;
```

Public Member Variable

Always use **PascalCase** for public member variable,

```
public int Salary = 100;
```



Components	Description
Namespaces	Pascal case for namespaces, no underscore and separate logical components with periods. Use CompanyName.TechnologyName as root. If you don't have a company, use your domain name or your own initials. Note that any acronyms of three or more letters should be Pascal case (Xml instead of XML) instead of all caps. Do not use the same name for a namespace and a class.
Assemblies	If the assembly contains a single name space, or has an entire self-contained root namespace, name the assembly the same name as the namespace.
Classes and Structs	Pascal Case, no underscores or leading "C" or "cls". Classes may begin with an "I" only if the letter following the I is not capitalized; otherwise it looks like an Interface. Classes should not have the same name as the namespace in which they reside. Any acronyms of three or more letters should be Pascal case, not all caps. Try to avoid abbreviations, and try to always use nouns.
Collection Classes	Follow class naming conventions, but add Collection to the end of the name.
Delegate Classes	Follow class naming conventions, but add Delegate to the end of the name.
Exception Classes	Follow class naming conventions, but add Exception to the end of the name.
Attribute Classes	Follow class naming conventions, but add Attribute to the end of the name.
Interfaces	Follow class naming conventions, but start the name with "I" and capitalize the letter following the "I"
Enumerations	Follow class naming conventions. Do not add "Enum" to the end of the enumeration name. If the enumeration represents a set of bitwise flags, end the name with a plural.
Functions and Subs	Pascal Case, no underscores except in the event handlers. Try to avoid abbreviations.
Parameters	Camel Case. Try to avoid abbreviations. Parameters must differ by more than case to be usable from case-insensitive languages like Visual Basic .NET.
Constants	Same naming conventions as public/private member variables or procedure variables of the same scope. If exposed publicly from a class, use PascalCase. If private to a function/sub, use camelCase.
Properties	Pascal Case, no underscores. Try to avoid abbreviations. Members must differ by more than case to be usable from case-insensitive languages like Visual Basic .NET.

Member variable

Declare member variable at the **top** of the class, If class has **static** member then it will come at the **top most** and after that other member variable.

```csharp
public class Account
{
    public static string BankName;
    public static decimal Reserves;
    public string Number
    {
        get;
        set;
    }
     public DateTime DateOpened
     {
         get;
         set;
     }
     public DateTime DateClosed
     {
         get;
         set;
     }
     public decimal Balance
     {
         get;
         set;
     }
     // Constructor
     public Account()
     {
         // ...
     }
 }
```

Enum

Always use PascalCasing as default naming standard.

- Use a singular type name for an enumeration unless its values are bit fields.
- Use a plural type name for an enumeration with bit fields as values, also called flags enum.
- Do not use an "Enum" suffix in enum type names.
- Do not use "Flag" or "Flags" suffixes in enum type names.
- Do not use a prefix on enumeration value names.

```csharp
enum MailType
{
   Html,
   PlainText,
   Attachment
}
```

Namespace

Always use **PascalCase** for namespace.

namespace NextProgramming.Domain

Standard Abbreviation for Standard Controls.

Abbreviations	Standard Control
btn	Button
cb	CheckBox
cbl	CheckBoxList
ddl	DropDownList
fu	FileUpload
hdn	HiddenField
hlk	Hyperlink
img	Image
lbl	Label
lbtn	LinkButton
mv	MultiView
pnl	Panel
txt	TextBox
DataGrid	dtg
imb	ImageButton
lst	ListBox
dtl	DataList
rep	Repeater
rdo	RadioButton
rdl	RadioButtonList
phd	Placeholder
tbl	Table
gv	GridView
dtv	DetailView
fv	FormView

Events Names

Events are associated with actions. Therefore, events are named with verbs. For example, Loaded, Clicked, and Printing.

- Give events names with a concept of before, current, and after, using the present and past tenses. Depending on the page, window, control, or class, the event names for a page can be, Initialized, PreRender, Rendering, PostRender, and Exited. A button event can be OnClick.

- Event handlers use "EventHandler" suffix, as shown in the following example:
- public delegate void ClickedEventHandler(object sender, ClickedEventArgs e);
- Use two parameters named sender and e in event handlers.
- Name event argument classes with the "EventArgs" suffix.

Fields Names

Use PascalCasing in field names.
Do not use a prefix for field names.
Do not use underscores in field names.

Naming a DLL or Assembly

Assemblies or DLLs are created for a major functionality such as a math library. The library name should be, CompanyName.Component.Dll. For example, Mindcracker.Math.dll and Mindcracker.Data.dll.

Naming Parameters

Use camelCasing and descriptive parameter names.
Use names based on a parameter’s meaning rather than the parameter’s type.

Naming Resources

Use PascalCasing and descriptive names in resource keys.
Use only alphanumeric characters and underscores in naming resources.