Class trong c#, căn bản về lập trình hướng đối tượng

C# là ngôn ngữ lập trình hướng đối tượng 100%. Hầu như mọi thứ trong C# đều là class. Do đó, học cách xây dựng và sử dụng class là yêu cầu bắt buộc trong khi học lập trình C#. Không nắm vững khái niệm class/object sẽ rất khó để học C#; Không biết kỹ thuật xây dựng class thì hầu như không làm được gì trong C#.

Bài học này sẽ đưa bạn bước đầu tiếp xúc với kỹ thuật xây dựng class trong C#. Nắm chắc các kỹ thuật này là yêu cầu bắt buộc để có thể đi xa hơn.

NỘI DUNG CỦA BÀI [Ẩn]

Class, trừu tượng hóa, lập trình hướng đối tượng

Cuốn tài liệu này được xây dựng dựa trên giả định rằng bạn đã hiểu được các khái niệm cơ bản của lập trình nói chung và lập trình hướng đối tượng nói riêng. Vì vậy, tài liệu sẽ chỉ nhắc lại sơ lược một số khái niệm cơ bản của lập trình hướng đối tượng như class, object, abstraction, các đặc trưng của class.

Class và trừu tượng hóa

Class là bản mô tả những tính chất và hành vi chung của những gì tồn tại trong thực tế. "Những gì tồn tại trong thực tế" đó được gọi là các *đối tượng* (object) cụ thể. Từ các đối tượng cụ thể, chúng ta phân tích ra những điểm chung về thông tin và hoạt động để tạo thành class.

Việc phân tích và tóm lược những tính chất và hành vi chung của một nhóm những đối tượng thực tế như vậy được gọi là *trừu tượng hóa* (abstraction). Việc trừu tượng hóa giúp chúng ta tách rời những thông tin cần thiết của đối tượng để nghiên cứu, đồng thời bỏ qua những gì không liên quan.

Class là kết quả của sự *trừu tượng hoá* các đối tượng cùng loại về hai khía cạnh: *thông tin* mô tả đối tượng, và những *hành vi* (hoạt động) trên các thông tin đó.

Với ý nghĩa trên, class không nhất thiết phải liên quan đến việc lập trình. Chúng ta có thể tạo ra các class với ý nghĩa là một sự trừu tượng hóa bất kỳ khi nào cần nghiên cứu về các đối tượng.

Object và cụ thể hóa

Ở chiều ngược lại, nếu có một bản mô tả trừu tượng, chúng ta có thể tạo ra nhiều phiên bản cụ thể của nó.

Ví dụ, nếu có bản mô tả trừu tượng về tủ, chúng ta có thể dùng vật liệu để tạo ra nhiều chiếc tủ thực sự dựa trên mô tả đó.

Những phiên bản cụ thể tạo ra từ mô tả trừu tượng như vậy được gọi là object.

Như vậy, quan hệ giữa class và object là loại quan hệ giữa mô tả (nằm trên giấy, trong suy nghĩ, v.v.) với sự vật/hiện tượng cụ thể trong thực tế.

Class trong lập trình hướng đối tượng và C#

Class trong các ngôn ngữ lập trình hướng đối tượng nói chung, trong C# nói riêng, mang ý nghĩa là một kiểu dữ liệu. Đối với C#, class là các khối xây dựng cơ sở của các chương trình ứng dụng, và là trung tâm của lập trình C#.

Class trong C# là một loại kiểu dữ liệu đặc biệt chứa định nghĩa những thuộc tính (thông tin) và phương thức (hành vi), dùng để mô tả chung cho một nhóm những thực thể cùng loại.

Trong C#, mỗi class có thể chứa:

1. Biến thành viên (field): Lưu trữ các thông tin mô tả về đối tượng hay trạng thái của đối tượng;
2. Thuộc tính (property): có vai trò lưu trữ thông tin tương tự như biến thành viên nhưng có khả năng kiểm soát dữ liệu xuất nhập;
3. Phương thức (method): Dùng để cập nhật, tính toán, cung cấp và xử lý thông tin;
4. Sự kiện (delegate/event): Gửi thông báo về sự thay đổi trạng thái của đối tượng ra bên ngoài.

Ngoài ra, trong class còn có thể chứa định nghĩa của kiểu dữ liệu khác, gọi là kiểu thành viên (member /inner/nested type). Class có thể chứa định nghĩa của bất kỳ nhóm kiểu nào mà bạn đã biết (class, struct, interface, delegate, enum).

Khi xem class như một kiểu dữ liệu thì object của class tương ứng chính là biến thuộc kiểu dữ liệu đó. Class chứa mô tả trừu tượng còn object chứa giá trị cụ thể của mỗi mô tả đó.

Khai báo class trong C#

Cú pháp khai báo class C#

Khai báo class trong C# sử dụng cấu trúc

```
[public|internal] class <tên class>{ [thân class] }
```

trong đó:

- `class` là từ khóa của C# dùng để khai báo class;
- tên class do người lập trình lựa chọn và phải tuân thủ quy tắc đặt định danh (xem dưới đây);
- `public` hoặc `internal` được gọi là các từ khóa *điều khiển truy cập* (access modifier) của class.

Các phần này viết tách nhau bởi dấu cách.

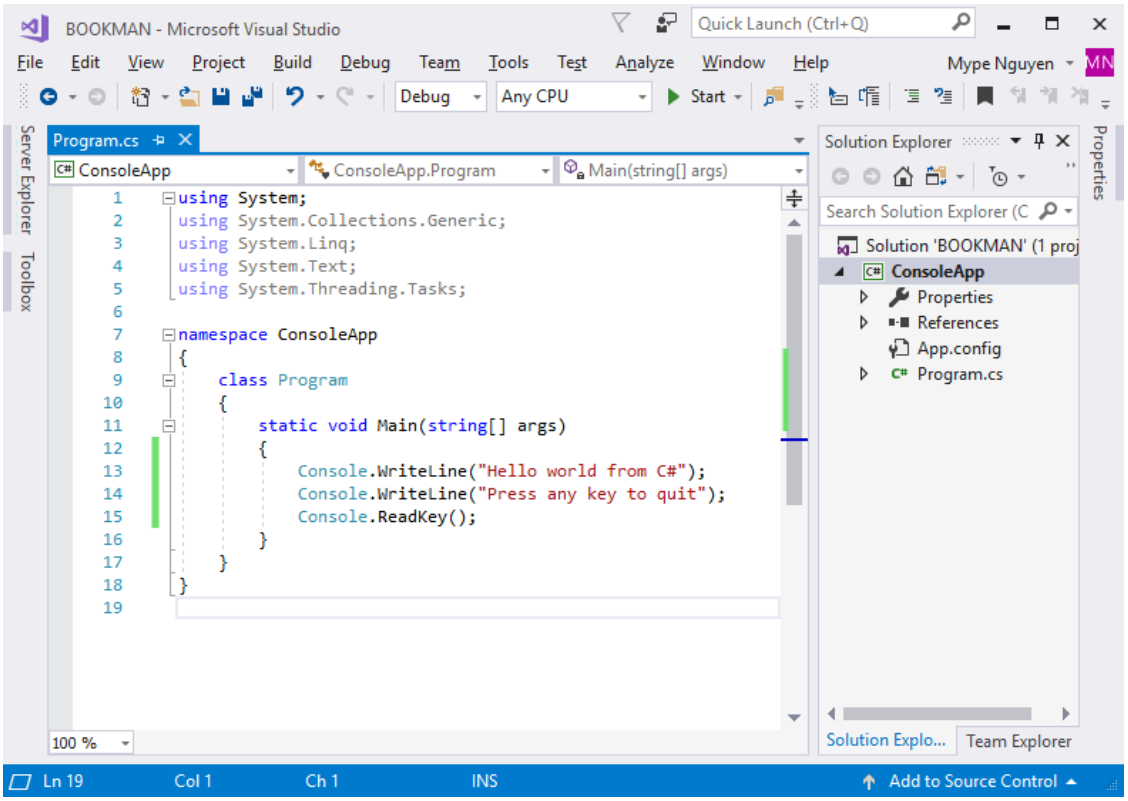
Dấu cách trong C# chỉ có tác dụng phân tách các thành phần của một lệnh, khác với một số ngôn ngữ dùng dấu cách như một phần của cú pháp. Số lượng dấu cách không ảnh hưởng tới ý nghĩa của code. Compiler sẽ tự bỏ qua những dấu cách thừa.

Từ khóa "`class`" là bắt buộc khi khai báo lớp. Mặc định, Visual Studio thể hiện từ khóa bằng màu xanh da trời. Từ khóa là những cụm ký tự được C# lựa chọn cho những mục đích riêng để diễn đạt cú pháp của ngôn ngữ. Một số kiểu dữ liệu dựng sẵn của C# cũng được đặt thành từ khóa (sẽ xem xét trong các bài sau).

Từ khóa điều khiển truy cập quyết định **phạm vi sử dụng** của class. Mỗi class trong C# có thể chỉ được sử dụng nội bộ trong phạm vi của project, hoặc có thể được sử dụng bởi các project khác. Mặc định, mỗi class trong C# chỉ được sử dụng trong phạm vi của project (sử dụng bởi các class khác trong cùng project). Do đó, nếu không thấy điều khiển truy cập nào thì sẽ hiểu là "`internal`". Nếu muốn sử dụng class này trong các project khác, trước từ khóa class cần bổ sung từ khóa "`public`".

> Từ khóa public sẽ thể hiện rõ vai trò của mình khi bạn bắt đầu tách một project lớn thành nhiều project con. Trong đó, project con thường là các thư viện lớp. Các class trong thư viện lớp phải đặt truy cập public thì mới sử dụng được trong project khác. Nếu chỉ có một project, internal hay public không có gì khác biệt.

Lấy ví dụ minh họa là file mã nguồn đầu tiên của chúng ta (Program.cs) của project đã xây dựng trong bài cài đặt Visual Studio.



Class này được đặt tên là Program, là một internal class (không chỉ rõ từ khóa truy cập => C# sẽ coi là internal). Khối code thân class này hiện có một phương thức (Main() hay Entry Point).

Đặt tên class trong C#

Tên class do người lập trình tự chọn và phải tuân thủ **quy tắc đặt định danh** (identifier) trong C#, cụ thể như sau:

1. Định danh phải bắt đầu bằng chữ cái, ký tự gạch chân "_" hoặc ký tự "@.";
2. Định danh chỉ được chứa chữ cái, chữ số và ký tự gạch chân;
3. Định danh không được trùng với từ khóa; nếu muốn đặt định danh trùng với từ khóa, cần đặt ký tự @ phía trước;
4. Độ dài định danh không giới hạn và có thể chứa ký tự Unicode (ví dụ, có thể đặt định danh bằng tiếng Việt có dấu);
5. Định danh có phân biệt chữ hoa và chữ thường (case-sensitive).

Ngoài các quy tắc trên, việc đặt tên class trong C# cũng nên tuân thủ các **quy ước** sau:

1. Mỗi file mã nguồn chỉ nên chứa một class và tên class đặt trùng tên file. Quy ước này kết hợp với quy ước đặt tên namespace giúp đồng bộ giữa cấu trúc quản lý class với cấu trúc file vật lý, giúp việc quản lý mã nguồn dễ dàng hơn;
2. Tên class bắt đầu bằng chữ cái in hoa;
3. Nếu tên class gồm nhiều từ ghép lại thì nên đặt theo kiểu *CamelCase* (viết hoa chữ cái đầu của mỗi từ).

Tên class C# được Visual Studio hiển thị bằng màu xanh nhạt.

Thân class

Thân class là một khối code và có thể chứa khai báo biến/hằng thành viên, thuộc tính và phương thức thành viên.

> Các thành phần của class sẽ lần lượt được xem xét trong các bài học tương ứng.

Đặc biệt hơn, trong thân class cũng có thể khai báo class khác, gọi là nested class (sẽ xem xét trong bài học riêng). Toàn bộ thân class phải đặt trong cặp dấu {}.

Toàn bộ code nằm trong một cặp dấu {} được gọi là một **khối code** (code block).

> Thân của class, namespace, method, interface, struct, các cấu trúc điều khiển (sẽ xem xét sau) đều là khối code.

Xin nhắc lại, trong lập trình hướng đối tượng có 3 giai đoạn: (1) định nghĩa class (định nghĩa kiểu dữ liệu); (2) khai báo và khởi tạo object (khai báo và gán giá trị cho biến); (3) sử dụng object (truy xuất dữ liệu, gọi phương thức). Trong bài này và một số bài tiếp theo chúng ta sẽ chỉ xem xét cách định nghĩa class mà chưa sử dụng các class này (dừng ở giai đoạn 1).

Một số ví dụ minh họa khai báo class trong C# project

Khai báo nhiều class trong cùng 1 file

Đây là ví dụ đơn giản nhất về khai báo class trong C#. Để thực hiện, hãy làm theo các bước sau:

1. Tạo một project mới có tên là P01_ClassSimple (Console App) nằm trong solution S01_Class
2. Viết code cho file Program.cs như sau:

> Đọc lại bài cài đặt visual studio hoặc bài thực hành tạo cấu trúc project nếu bạn quên cách tạo project và solution mới.

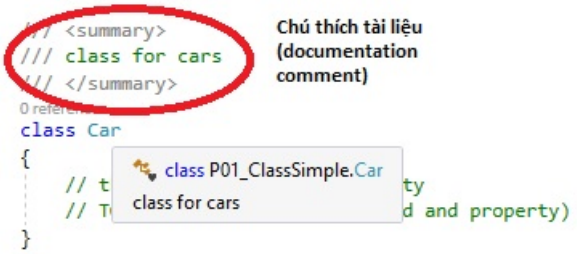
```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace P01_ClassSimple
8.  {
9.      class Program
10.     {
11.         static void Main(string[] args)
```

```
        {
            Console.WriteLine("Simple class declaration examples");
            Console.WriteLine("Press any key to quit ...");
            Console.ReadKey();
        }
    }

    /// <summary>
    /// class for cars
    /// </summary>
    class Car
    {
        // the class body is still empty
        // TODO: add more member (field and property)
    }

    /// <summary>
    /// class for airplanes
    /// </summary>
    internal class Airplane
    {

    }

    /// <summary>
    /// class for motorbikes
    /// </summary>
    public class Motorbike
    {

    }
}
```

Trong ví dụ này chúng ta đã khai báo 3 class trong cùng một file mã nguồn Program.cs:

1. Lớp Car không có từ khóa truy cập, mặc định nó sẽ được hiểu là một internal class (chỉ sử dụng được trong project này).
2. Lớp Airplane chỉ rõ từ khóa truy cập là internal, và có cùng tác dụng như lớp Car.
3. Lớp Motorbike sử dụng từ khóa truy cập là public.

Cả ba class này cùng sử dụng chú thích tài liệu ở phía trước.



Ở bất kỳ đâu trong project, nếu bạn trỏ chuột vào kiểu Car, Visual Studio sẽ hiển thị chú thích này. Nó giúp bạn mô tả và nhớ lại mục đích của class nhanh chóng mà không cần mở lại file mã nguồn.

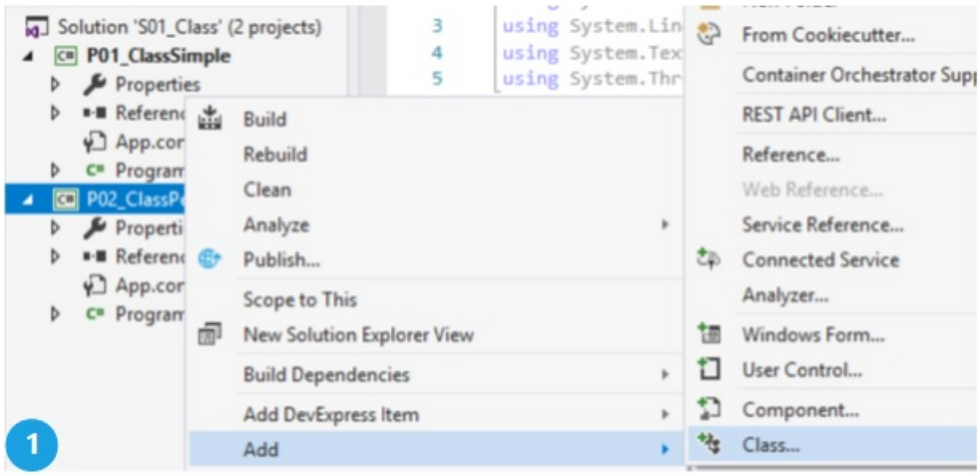
Để ý là cả ba class này giờ đều đang nằm trong cùng namespace P01_ClassSimple, vì vậy chúng sẽ có tên đầy đủ lần lượt là P01_ClassSimple.Car, P01_ClassSimple.Airplane, P01_ClassSimple.Motorbike.

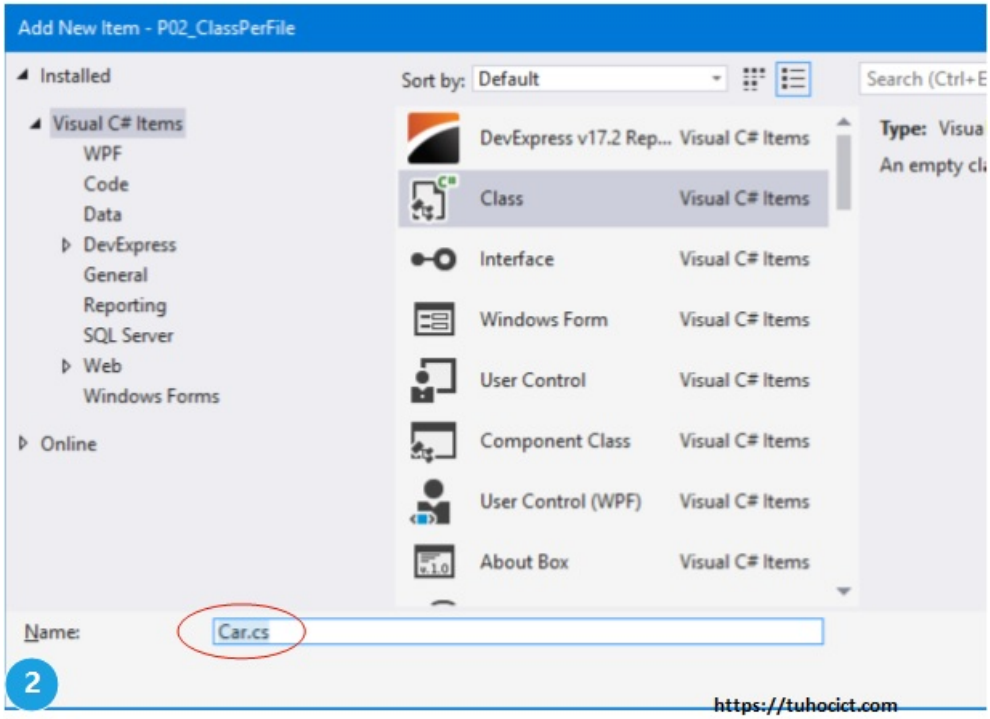
Cách thức khai báo này không được khuyến khích. Nó làm phình file code với nhiều nội dung không liên quan, dẫn đến khả năng bị lỗi, khó theo dõi và bảo trì về sau.

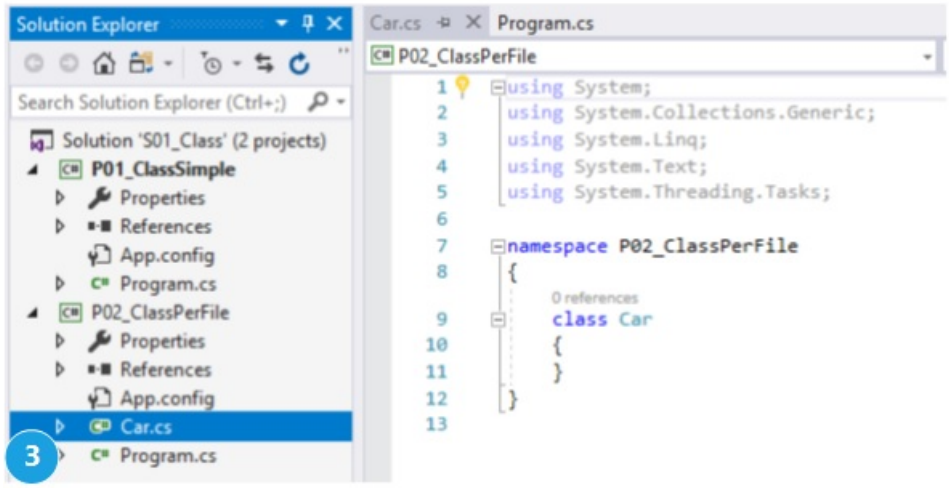
Khai báo mỗi class trong 1 file code riêng

Thêm một project mới đặt tên là P02_ClassPerFile (Console App) vào solution trên.

Sau đó thực hiện theo các bước dưới đây.







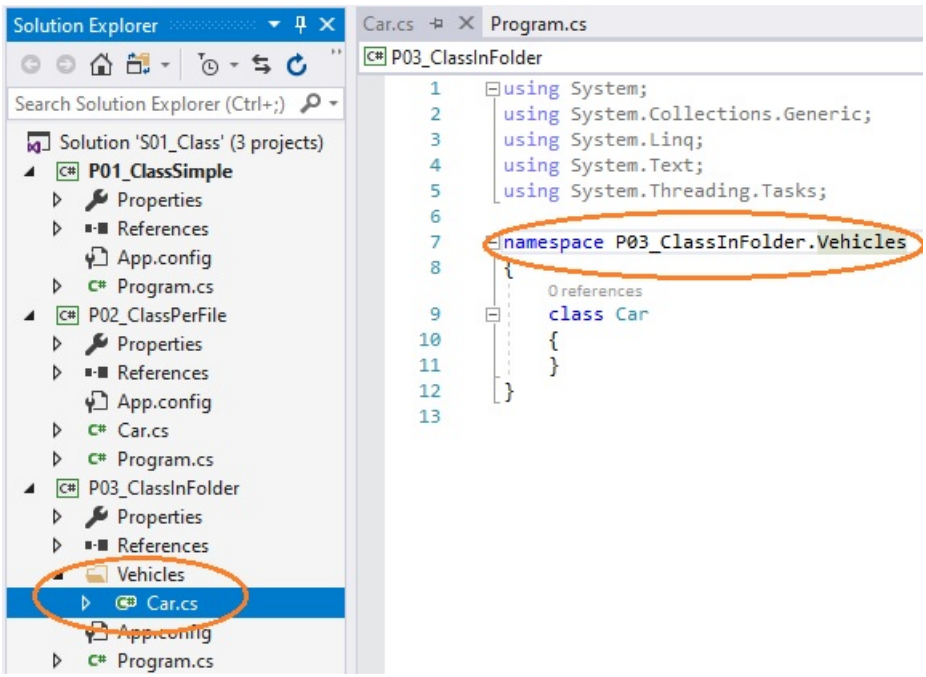
Bạn có thể nhận thấy, file mã nguồn mới Car.cs chứa đúng 1 class (internal) Car, nằm trong namespace P02_ClassPerFile (trùng tên project). Nếu số lượng class ít và không chia nhóm thì có thể dùng cách bố trí file như thế này.

Khai báo class trong file riêng nằm trong thư mục

Thêm project P03_ClassInFolder vào solution.

Trong project này thêm thư mục Vehicles.

Click phải vào thư mục và chọn Add => Class, tương tự như trong phần thực hành bên trên, để tạo class Car trong file code Car.cs.



Sự khác biệt là bây giờ file Car.cs nằm trong thư mục Vehicles, còn lớp Car sẽ nằm trong namespace P03_ClassInFolder.Vehicles. Cấu trúc namespace chứa class giờ đồng nhất với cấu trúc folder chứa code file. Do mỗi file chứa đúng 1 class trùng tên nên cấu trúc code file cũng đồng nhất với vị trí class.

Đây là cấu trúc được khuyến khích sử dụng khi khai báo class mới trong C#. Nó rất phù hợp khi số lượng class lớn và phân thành nhiều nhóm khác nhau.

Một số lưu ý khi khai báo class mới trong C#

Thông thường, một class nên khai báo trực tiếp trong namespace của project (trừ trường hợp nested class sẽ xem xét ở một bài riêng). Nếu class nằm trong không gian tên con, hãy tạo một folder có tên trùng với tên không gian con và đặt file class mới trong thư mục đó.

Mỗi class nên khai báo trong một file riêng. Tên file nên đặt trùng tên class. Điều này có thể làm tăng số lượng file, nhưng mỗi file có ít code, cũng đồng nghĩa với việc sẽ dễ tìm kiếm, dễ bảo trì, ít mắc lỗi hơn.

> Nhắc lại: nếu bạn tạo một folder bên trong project, Visual studio sẽ tự động lấy tên folder làm namespace con, tên project là namespace chính. Khi đó, một file class mới đặt trong folder sẽ tự động được đặt trong namespace con. Điều này tạo ra sự đồng bộ giữa cấu trúc file/folder với class/namespace.

> Hãy đọc lại bài viết về namespace và cách tạo cấu trúc thư mục cho project nếu bạn không nhớ.

Nếu project không phải là thư viện, các class nên để mức truy cập là internal.

Đối với mỗi class nên sử dụng **ghi chú tài liệu**.

> *Ghi chú tài liệu* (documentation comment) là loại ghi chú đặc biệt của C#, cho phép tạo ra một dạng "hướng dẫn sử dụng" của đơn vị code được ghi chú (như class, method, interface, v.v.).
>
> Loại ghi chú này cho phép người xây dựng class đưa ra các hướng dẫn cơ bản mà người sử dụng class có thể đọc. Loại ghi chú này cũng cho phép trình biên dịch lọc riêng ra để tạo thành tài liệu hướng dẫn cho code. Ghi chú tài liệu được Visual Studio sinh tự động khi gõ cụm "///" trước đối tượng cần chú thích.
>
> Nên hình thành thói quen viết chú thích đầy đủ cho code. Nó giúp ích rất nhiều cho việc bảo trì code hoặc làm việc nhóm.

Kết luận

Trong bài học này chúng ta đã học kỹ thuật cơ bản nhất về khai báo class trong C#. Các kỹ thuật này chỉ đủ để bạn tạm thời hiểu được class trong C# là gì và cách thức cơ bản để tạo ra một class.

Lưu ý rằng, còn rất nhiều kỹ thuật làm việc với class chúng ta sẽ lần lượt xem xét trong các bài tiếp theo. Các kỹ thuật trong bài này chưa đủ để có thể xây dựng các class thực sự có giá trị.

> + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
> + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
> + Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
> Cảm ơn bạn!