NESTED CLASS

Nested class

Class được khai báo bên trong thân của một class khác được gọi là *nested class* (lớp lồng nhau) hoặc *inner class* (lớp trong, lớp nội bộ); class chứa class đó được gọi là *outer class* (lớp ngoài, lớp bao).

Nếu một class được khai báo trực tiếp trong namespace, class đó còn được gọi là *class cấp đỉnh* (top level class).

Trong tất cả các bài học từ trước đến giờ, các class bạn xây dựng đều thuộc loại này.

Nested class trong C# được sử dụng trong trường hợp thân của một class trở nên quá lớn nhưng có chứa những logic tương đối độc lập. Khi đó các khối logic này có thể xem xét tách thành các lớp nội bộ để dễ dàng hơn trong quản lý code của class chính.

Việc phân chia này đồng thời giúp gói gọn code tránh làm dự án bị phân tán bởi các lớp nhỏ không được sử dụng bởi *client code*.

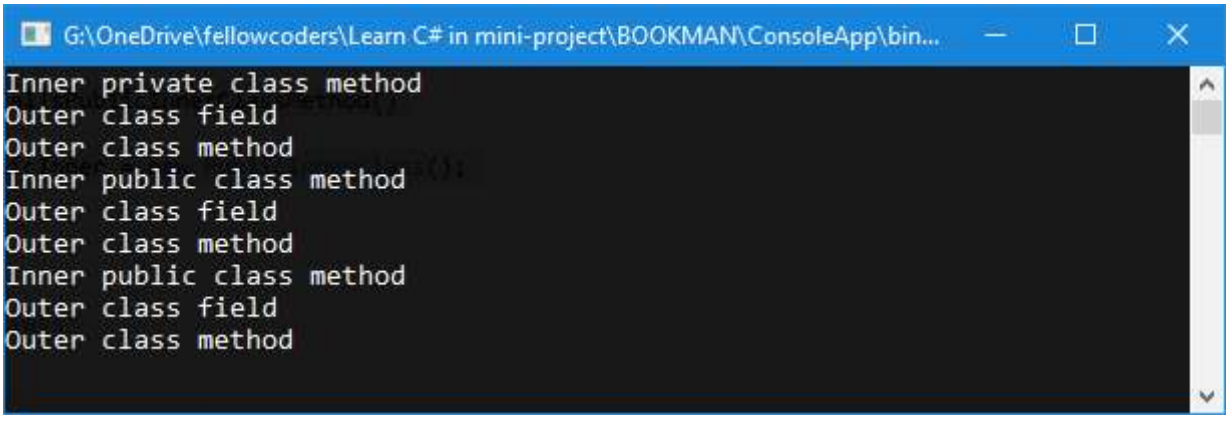
Client code hiểu một cách đơn giản là những code ở bên ngoài class và sử dụng class này.

Ví dụ minh họa cách khai báo và sử dụng nested class

Hãy cùng xem ví dụ sau để hiểu rõ hơn về cách sử dụng nested class trong C#.

```
using System;
namespace P01_NestedClass
{
    internal class OuterClass
    {
        private string _str = "Outer class field";
        private void OuterClassMethod() => Console.WriteLine("Outer class method");
        public void OuterMethodCallsPrivateInnerClassMethod()
        {
            PrivateInnerClass privateInner = new PrivateInnerClass();
            privateInner.Method();
        }
        public void OuterMethodsCallsPublicInnerClassMethod()
        {
            PublicInnerClass publicInner = new PublicInnerClass();
            publicInner.Method();
        }
        public class PublicInnerClass
        {
            public void Method()
            {
                Console.WriteLine("Inner public class method");
```

```
                OuterClass outer = new OuterClass();
                Console.WriteLine(outer._str);
                outer.OuterClassMethod();
            }
        }
        private class PrivateInnerClass
        {
            public void Method()
            {
                Console.WriteLine("Inner private class method");
                OuterClass outer = new OuterClass();
                Console.WriteLine(outer._str);
                outer.OuterClassMethod();
            }
        }
    }
    internal class Program
    {
        private static void Main(string[] args)
        {
            OuterClass outer = new OuterClass();
            outer.OuterMethodCallsPrivateInnerClassMethod();
            outer.OuterMethodsCallsPublicInnerClassMethod();
            OuterClass.PublicInnerClass inner = new OuterClass.PublicInnerClass();
            inner.Method();
            Console.ReadKey();
        }
    }
}
```



chương trình minh họa nested class trong c#

Kết quả chạy chương trình minh họa nested class trong c#

Qua ví dụ trên có thể thấy, nested class cũng là một class hoàn toàn bình thường ngoại trừ ba vấn đề:

1. nếu trong nested class khởi tạo object của class ngoài thì có thể truy cập cả vào các thành viên private của object đó;
2. nested class có thể bị che giấu hoàn toàn khỏi client code (dùng từ khóa private);

3. sử dụng nested class (public) phải theo quy tắc "Tên_lớp_ngoài.Tên_lớp_nội_bộ".

Đặc điểm của nested class

Nested class có một số điểm khác biệt so với top-level class.

Nested class có thêm từ khóa điều khiển truy cập `private`. Nếu một nested class được định nghĩa với từ khóa `private` thì các *lớp sibling* của lớp ngoài (lớp khác cùng cấp độ với lớp ngoài) không nhìn thấy được nó. Chỉ code của lớp ngoài mới sử dụng được nested class dạng private. Nếu không chỉ ra từ khóa điều khiển truy cập nào, C# mặc định sử dụng private cho lớp nội bộ.

Nested class có thể khởi tạo object của lớp ngoài và truy cập các thành viên private của object lớp ngoài.

Lớp nội bộ nếu được định nghĩa truy cập public thì lớp sibling của lớp ngoài cũng có thể sử dụng nó như sử dụng các lớp bình thường. Khi đó, lớp sibling của lớp ngoài sẽ dùng cấu trúc *Tên_lớp_ngoài.Tên_lớp_trong* để sử dụng lớp nội bộ.

Lưu ý: trong C# nên hạn chế sử dụng nested class. Việc sử dụng nested class không hợp lý có thể dẫn đến những lỗi khó lường trước, đặc biệt là khi cho lớp trong và lớp ngoài gọi lẫn nhau.

Nested types

Lớp lồng nhau như bạn đã xem xét ở trên là một trường hợp riêng của **nested types** – đặc điểm của C# cho phép khai báo các kiểu dữ liệu khác bên trong một class hoặc struct.

Nested types trong C# thể hiện quan hệ "has-a" giữa class/struct ngoài với kiểu khai báo bên trong nó.

C# cho phép khai báo tất cả các nhóm kiểu bạn đã biết (enum, class, struct, interface, delegate) bên trong một class hoặc struct. Khi đó, các kiểu "bên trong" được gọi chung là các nested type. Mỗi nested type có thể xem như "kiểu thành viên" của class hoặc struct, tương tự như biến thành viên, phương thức thành viên hay đặc tính thành viên.

Về cấu trúc cú pháp, việc khai báo các kiểu nested không có gì khác biệt so với khi nó được khai báo là kiểu cấp đỉnh (top-level type) trực thuộc namespace.

```
class OuterClass
{
    public enum InnerEnum { ValueOne, ValueTwo, ValueThree } // khai báo nested enum
    private struct InnerStruct // khai báo nested struct private
    {
        public int AnIntMember = 10;
    }
}
```

Còn một lưu ý nữa: bản thân nested type lại có thể tiếp tục chứa nested type của riêng nó với cấp độ lồng nhau không hạn chế. Tuy nhiên, đây là những tình huống sử dụng rất hãn hữu. Việc đặt lớp lồng nhau

nhiều cấp khiến code rắc rối và khó theo dõi.

Sử dụng nested type

Vậy khi nào và tại sao bạn nên xem xét sử dụng nested type thay vì khai báo kiểu như bình thường (thuộc namespace)?

Thứ nhất, nested type cho phép kiểm soát truy cập tốt hơn. Bạn có thể sử dụng từ khóa private để đặt kiểu dữ liệu nó hoàn toàn nội bộ. Trong khi đó, kiểu khai báo trong namespace chỉ có hai mức truy cập: public và internal.

Thứ hai, do nested type cũng được xem là thành viên của class/struct, nó có thể truy xuất các thành viên private/protected khác của class/struct chứa nó.

Thứ ba, nếu bạn xác định rằng một kiểu dữ liệu nào đó chỉ hữu dụng đối với 1 class duy nhất hoặc không được sử dụng bởi client code, kiểu dữ liệu đó nên được khai báo làm nested type.

Nested type được sử dụng từ client code thông qua tên của class/struct chứa nó giống hệt như đối với nested class mà bạn đã xem xét ở phần trên. Việc sử dụng nội bộ bên trong class/struct chứa kiểu đó không có gì khác biệt so với khi kiểu đó được khai báo trong namespace.

Ví dụ, với class OuterClass như trên bạn có thể sử dụng các kiểu nested từ client code như sau:

```
// sử dụng từ client code
var innerEnumValueOne = OuterClass.InnerEnum.ValueOne;
var innerStructObject = new OuterClass.InnerStruct();
```

Nếu sử dụng trong nội bộ lớp OuterClass:

```
class OuterClass
{
    public enum InnerEnum { ValueOne, ValueTwo, ValueThree } // khai báo nested enum
    private struct InnerStruct // khai báo nested struct private
    {
        public int AnIntMember = 10;
        public ValueOne = InnerEnum.ValueOne;
    }
    public void Print(InnerStruct innerStruct)
    {
        Console.WriteLine(innerStruct.AnIntMember);
        Console.WriteLine(innerStruct.ValueOne);
    }
}
```