Bài 6.5: Namespace và partial class

- ✓ Namespace
- ✓ Partial class
- ✓ Partial methods
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Namespace

- ✓Namespace là không gian tên được sử dụng rộng rãi trong ngôn ngữ lập trình C#.
- ✓Thư viện .NET sử dụng namespace để tổ chức và quản lý các kiểu bên trong nó.
- ✓Namespace cho phép lập trình viên quản lý phạm vi của các kiểu trong các dự án lớn.
- ✓Khi muốn sử dụng một kiểu nào đó ta khai báo namespace chứa các thành phần muốn sử dụng ở đầu file chương trình với cú pháp: **using namespaceName;**
- ✓Ví dụ: using S ystem;

```
using System;

namespace CSharpCourse
{
    class Lesson62
    {
        public void DoSomething(Person p)
        {
            Console.WriteLine("First name: " + p.firstName);
            Console.WriteLine("Last name: " + p.lastName);
            Console.WriteLine("Age: " + p.age);
        }
```

Khai báo namespace cần dùng

Sử dụng lớp Console của namespace System

Namespace

✓Nếu không khai báo namespace chứa kiểu cần dùng, ta phải chỉ rõ tên namespace và tên lớp cần sử dụng:

```
namespace CSharpCourse
{
    1 reference
    class Lesson62
    {
        1 reference
        public void DoSomething(Person p)
        {
            System.Console.WriteLine("First name: " + p.firstName);
            System.Console.WriteLine("Last name: " + p.lastName);
            Console.WriteLine("Age: " + p.age); // error!
        }
```

Tự định nghĩa namespace

- ✓Để khai báo namespace ta tự định nghĩa, sử dụng cú pháp: **namespace Name { types }**
- ✓Trong đó keyword namespace là bắt buộc và luôn đứng trước tên của namespace.
- ✓Name là phần tên của namespace, thường là tên của solution, project.
- ✓Bên trong namespace ta khai báo các kiểu(class, struct, interface) sao cho tên mỗi kiểu là không trùng lặp.

```
namespace Lesson65
{
    0 references
    internal class Program...

    // phần thứ nhất của lớp student chỉ chứa các properties và constructors
    8 references
    partial class Student...
    // phần thứ hai của lớp student chứa các phương thức khác
    8 references
    partial class Student...

    1 reference
    struct Employee { }
    0 references
    struct Person { }
    1 reference
    struct Employee { } // error, duplicate
}
```

Namespace cho toàn bộ file(C# 10)

- ✓Từ C# 10 ta có thể khai báo namespace cho cả file đó bằng cách đặt **namespace Name;** ở đầu file code.
- ✓Ví dụ:

```
using System;
using static System.Console;

namespace Lesson65;

0 references
internal class Program...

// phần thứ nhất của lớp student chỉ chứa các properties và constructors
8 references
partial class Student...
// phần thứ hai của lớp student chứa các phương thức khác
8 references
partial class Student...

0 references
struct Employee { }
0 references
struct Person { }
```

Đặc trưng của namespace

- ✓Sử dụng để quản lý các dự án lớn.
- ✓Được phân tách bằng dấu chấm. Ví dụ System.Console.
- ✓Sử dụng từ khóa using để khai báo namespace cần sử dụng và đặt ở đầu file code.
- ✓Namespace global là namespace gốc của thư viện .NET. Ta có thể ngầm định bỏ qua namespace này.

```
namespace Lesson65
{
    0 references
    internal class Program
    {
        0 references
        static void Main()
        {
            // sử dụng global namespace
            global::System.Console.WriteLine("Hello CSharp!");
            // sử dụng tên đầy đủ của lớp
            System.Console.WriteLine("Welcome to Branium Academy!");
        }
    }
```

Lớp partial

- ✓Ta có thể tách định nghĩa của một kiểu(class, struct, interface) trong C# ra thành các phần nhỏ và đặt ở nhiều file khác nhau.
- ✓Trong mỗi phần đó có thể chứa các khai báo thuộc tính, phương thức,… và tất cả chúng sẽ được kết hợp lại thành một khối hoàn chỉnh khi biên dịch chương trình.
- ✓Có nhiều trường hợp ta cần phải tách định nghĩa lớp:
  - ✓ Khi làm việc trong một dự án lớn, phân tách chức năng của một lớp ra nhiều file cho phép nhiều lập trình viên cùng làm việc với 1 lớp tại cùng thời điểm.
  - ✓ Khi làm việc với trình tự động sinh code như tạo window form, web service wrapper, ta có thể tạo code sử dụng các lớp đó mà không phải sửa đổi, tạo lại các file tạo bởi trình tự động sinh code.
  - ✓ Khi sử dụng trình tạo code để tạo các chức năng bổ sung cho một lớp.
- ✓Để tách định nghĩa lớp ta sử dụng keyword partial.
- ✓Khi sử dụng partial, các phần khác của lớp có thể được định nghĩa trong cùng namespace.

Lớp partial

✓Tất cả các phần của một lớp phải có keyword partial, cùng access modifier, cùng có mặt đầy đủ tại thời điểm chương trình được biên dịch.

```
partial class Student
{
    3 references
    public string StudentId { get; set; }
    7 references
    public string FullName { get; set; }
    3 references
    public int Age { get; set; }
    3 references
    public string Major { get; set; }
    3 references
    public float Gpa { get; set; }
    0 references
    public Student() ...
    0 references
    public Student(string id) ...
    0 references
    public Student(string id, string fullName) ...
    3 references
    public Student(string id, string fullName, int age, string major, float gpa)...
}
// phần thứ hai của lớp student chứa các phương thức khác
8 references
partial class Student
{
    // phương thức nhập thông tin cho sinh viên từ bàn phím
    0 references
    public void GetInputInfo()...
    // phương thức mô tả hành động làm bài tập về nhà
    0 references
    public void DoHomework(string subject)...
```

Lớp partial

- ✓Tất cả các phần sau của một lớp partial được gộp lại khi biên dịch chương trình:
  - ✓ Các chú thích.
  - ✓ Các interface.
  - ✓ Các thuộc tính.
  - ✓ Các thành phần cấu thành của lớp.
  - ✓ Các thuộc tính của tham số kiểu generic.
- ✓Các quy tắc bạn cần tuân thủ:
  - ✓ Tất cả các phần của định nghĩa lớp partial đều phải có keyword partial.
  - ✓ partial là keyword chỉ xuất hiện ngay trước class, struct, interface.
  - ✓ Một kiểu partial lồng trong kiểu partial khác được coi là hợp lệ.
  - ✓ Tất cả các phần của kiểu partial phải được định nghĩa trong cùng assembly và cùng module.
  - ✓ Tên lớp và tham số generic phải khớp nhau trong tất cả các phần định nghĩa của một kiểu partial.
  - ✓ Các keyword sau là tùy chọn với các kiểu partial nhưng nếu sử dụng phải không được xung đột với keyword ở các phần khác của cùng kiểu partial: public, private, protected, internal, abstract, sealed, new, base class, generic constraints.

Phương thức partial

- ✓Ta cũng có thể tạo ra partial method trong một lớp partial. Trong đó một phần chứa dấu hiệu nhận biết của phương thức. Phần còn lại triển khai định nghĩa chi tiết phương thức đó.
- ✓Nếu định nghĩa của phương thức không được triển khai, các lời gọi và khai báo của phương thức đó sẽ bị xóa bỏ khi biên dịch chương trình.
- ✓Phương thức partial không bắt buộc phải triển khai trong các trường hợp sau:
  - ✓ Không có access modifier.
  - ✓ Trả về kiểu void.
  - ✓ Không có tham số tham chiếu out.
  - ✓ Không chứa bất kì keyword nào trong danh sách sau: virtual, override, sealed, new, extern.
- ✓Tất cả các phương thức không tuân thủ các ràng buộc trên đều phải triển khai định nghĩa phương thức cụ thể.

Phương thức partial

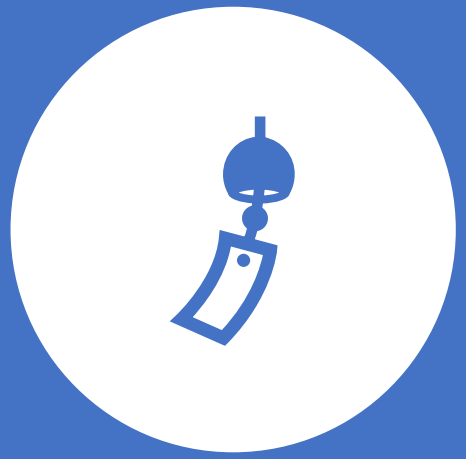
- ✓Có 2 kịch bản sử dụng partial methods: template code và source generator.
- ✓Phương thức partial phải chứa keyword partial ngay trước kiểu trả về của phương thức.
- ✓Dấu hiệu nhận biết của phương thức ở cả phần khai báo và triển khai trong lớp partial phải khớp nhau.
- ✓Phương thức partial có thể chứa keyword static và unsafe.
- ✓Phương thức partial có thể là generic.
- ✓Bạn có thể tạo một delegate tham chiếu tới một phương thức partial đã được khai báo và triển khai đầy đủ. Nhưng không thể tham chiếu tới phương thức partial chỉ mới có phần khai báo.

Ví dụ phương thức partial

```
partial class Calculator
{
    // khai báo định nghĩa phương thức
    1 reference
    public partial int Add(int a, int b, int c);
    1 reference
    public partial int Max(int a, int b, int c);
}
```

```
1 reference
partial class Calculator
{
    // triển khai chi tiết định nghĩa của phương thức
    1 reference
    public partial int Add(int a, int b, int c)
    {
        return a + b + c;
    }

    1 reference
    public partial int Max(int a, int b, int c)
    {
        int max = Math.Max(a, b);
        return Math.Max(max, c);
    }
}
```



Nội dung tiếp theo

Tìm hiểu về các kiểu lồng nhau(nested types)