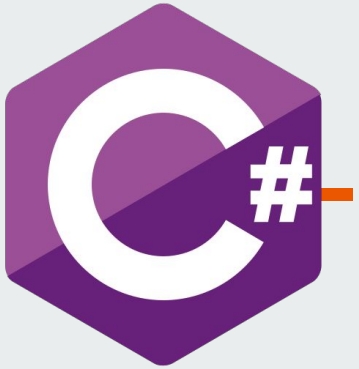


Bài 12

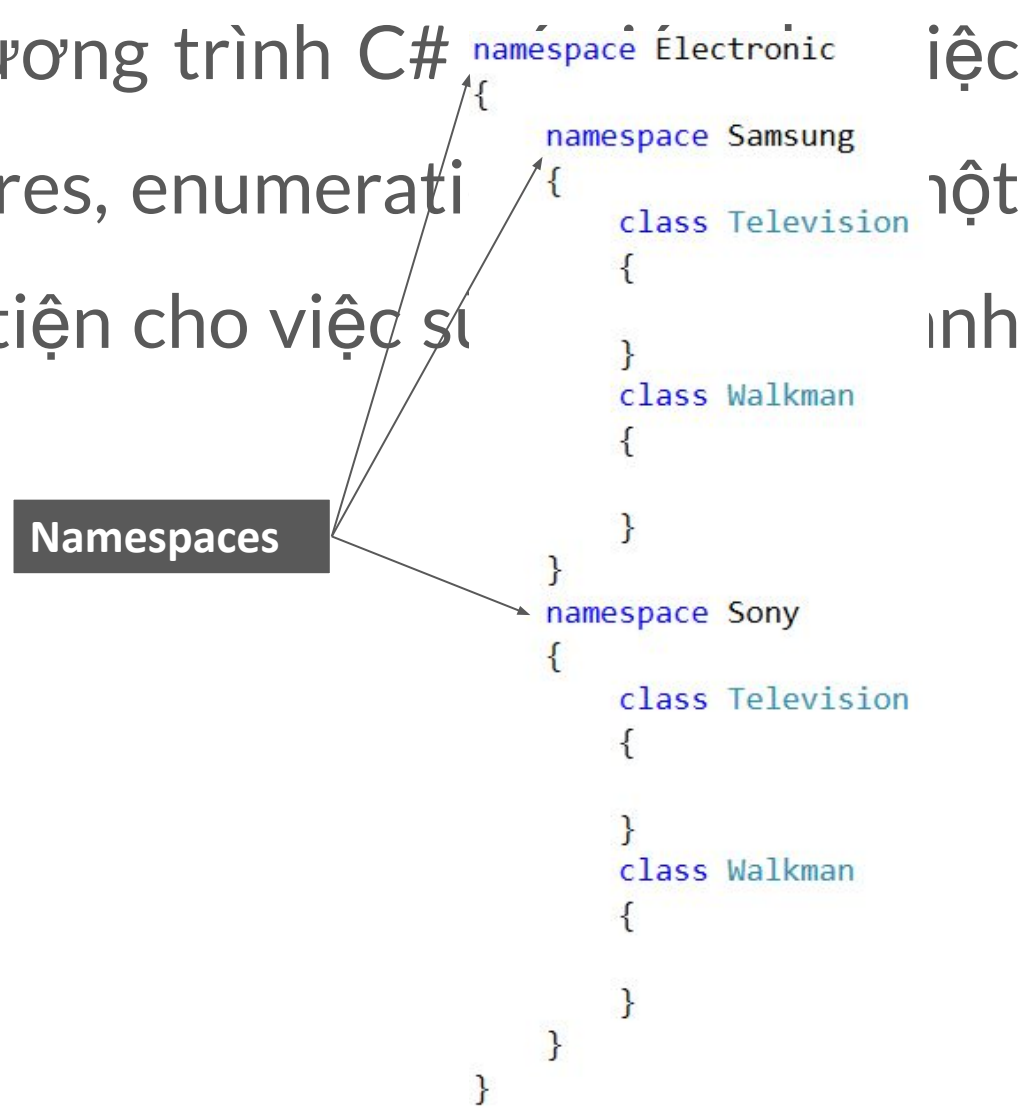
Namespace và Exceptions

Mục tiêu 1

- Giới thiệu về Namespace
- Đặt điểm của Namespace
- Các Namespace của hệ thống
- Các sử dụng Namespace
- Cách tạo Namespace
- Namespace lồng nhau
- Đặt bí danh cho Namespace

Giới thiệu Namespace

- Một namespace là một thành phần của chương trình C# [illegible] iệc quản lý và tổ các class, interface và structures, enumerati [illegible] ột cách logic, tránh sự xung đột về tên, thuận tiện cho việc s[illegible] nh phần ứng dụng giữa các dự án khác nhau.



Đặc điểm của Namespace

- Nó cung cấp một cấu trúc thứ bậc mà giúp cho việc xác định các lớp dựa trên từng nhóm.
- Nó cho tạo nhiều class, interface,..trong namespace (giống tệp tin trong thư mục).
- Nó tạo nên nhiều namespace con lồng nhau (giống cấu trúc thư mục).
- Tên của các class, interface… là duy nhất trong mỗi namespace.

Một số Namespace trong hệ thống

- .NET Framework bao gồm rất nhiều namespace được tạo dựng sẵn và được phân cấp theo từng mục đích. Các namespace này được coi như là namespace hệ thống.
  - System.Collections
  - System.Data
  - System.Diagnostics
  - System.IO
  - System.Net
  - System.Web

Cách sử dụng Namespace

- Có 2 phương pháp sử dụng namespace

Class 1

```
using System;
…
Console.Read();
Console.Write();
```

Lập hình hiệu quả

Class 2

```
System.Console.Read();
…
System.Console.Write();
```

Lập hình không hiệu quả

Tạo Namespace 1-2

- C# cho phép bạn tạo các namespace với các tên thích hợp để quản lý các structures, class, interface, delegate, và các enumerations.
- Cú pháp:

```
namespace <NamespaceName>
{
 //Khai báo các thành phần
}
```

Tạo Namespace và sử dụng

```
using System;
using Samsung;//đưa namespace cần dùng vào
namespace Session07
{
    class Program
    {
        static void Main(string[] args)
        {
            Television tv = new Television();
            tv.Show();
        }
    }
}
namespace Samsung
{
    class Television
    {
        public void Show()
        {
            Console.WriteLine("Class Television in Samsung Namespace");
        }
    }
    class Walkman
    {
        public void Show()
        {
            Console.WriteLine("Class Walkman in Samsung Namespace");
        }
    }
}
```

Sử dụng Namespace ngắn gọn hiệu quả

```
using System;
namespace Session07
{
    class Program
    {
        static void Main(string[] args)
        {
            //sử dung tên đầy đủ
            Samsung.Television tv = new Samsung.Television();
            tv.Show();
        }
    }
}
namespace Samsung
{
    class Television
    {
        public void Show()
        {
            Console.WriteLine("Class Television in Samsung Namespace");
        }
    }
    class Walkman
    {
        public void Show()
        {
            Console.WriteLine("Class Walkman in Samsung Namespace");
        }
    }
}
```

Sử dụng Namespace đầy đủ rõ ràng

Phạm vi truy xuất của Namespace

- Namespace luôn có phạm vi truy xuất là **public** bạn không thể áp dụng các từ **private, protected, public, internal** cho namespace, do vậy namespace có phạm vi truy xuất không hạn chế.

public namespace A{} ✗

protected namespace A{} ✗

private namespace A{} ✗

internal namespace A{} ✗

namespace A{} ✓

Namespace lồng nhau

C# cho phép bạn tạo ra sự phân c g
nhau, thậm chí trong namespace v ,
nó giống hệt cấu trúc thư mục và tệ

```
namespace Electronics
{
    public interface IMachine { }
    class Computer
    {
        public void Show()
        {
            Console.WriteLine("Class Computer in Electronics Namespace");
        }
    }
    namespace Samsung
    {
        class Television
        {
            public void Show()
            {
                Console.WriteLine("Class Walkman in Electronics.Samsung Namespace");
            }
        }
        class Walkman { }
    }
    namespace Sony
    {
        class Television { }
        class Walkman { }
    }
}
```

Đặt bí danh cho namespace hoặc lớp

- Các alias là các tên tạm thời được coi như các thực thể.
- Các alias được dùng khi có nhiều namespace lồng nhau được khai báo và bạn cần thiết dùng alias để làm ngắn gọn việc truy xuất tới các class và dễ dàng duy trì.
- **Cú pháp**

  using <aliasName>=<namespaceName>;

```
using ss = Electronics.Samsung; //đặt bí danh
using sn = Electronics.Sony; //đặt bí danh
using but = System.Console; //đặt bí danh
namespace Session07
{
    class Program
    {
        static void Main(string[] args)
        {
            but.WriteLine("Alias for Namespace and class");
            ss.Television tv1 = new ss.Television();
            sn.Television tv2 = new sn.Television();
        }
    }
}
```

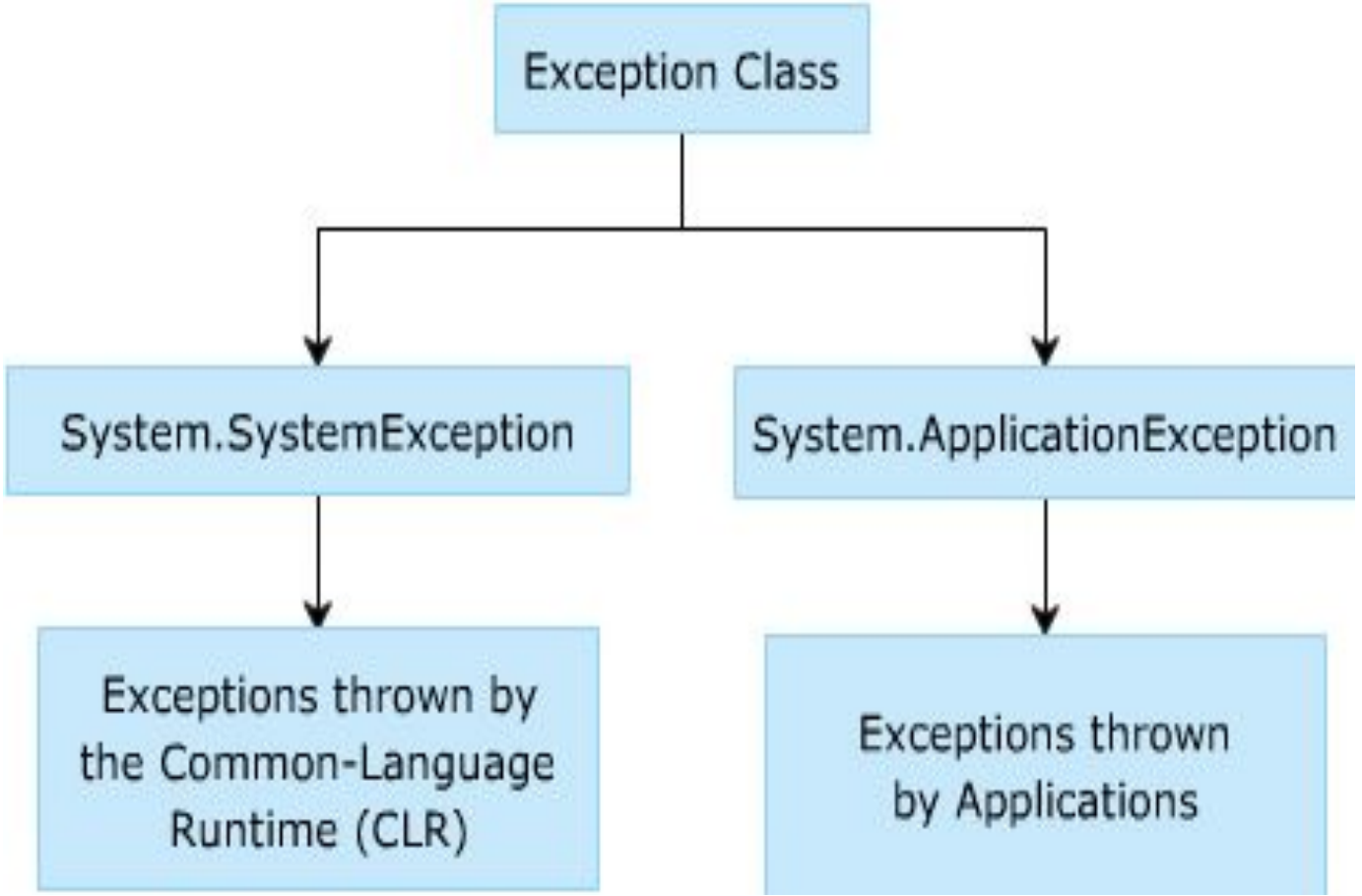
Mục tiêu 2

- Giới thiệu về Exception
- Lớp Exception
- Một số Exception của hệ thống
- Xử lý ngoại lệ
- Tạo Custom Exception

Giới thiệu về Exception

- Các exceptions là các lỗi run-time mà phá vỡ việc thực thi của một chương trình. Trong C# bạn có thể điều khiển sự thực thi này bằng cách sử dụng cấu trúc try...catch hoặc try...catch... finally. C# cho phép bạn định nghĩa các exceptions và cho phép bạn sửa đổi cách xử lý của các lỗi này.

Các ngoại lệ trong C#



Lớp Exception

- Class Exception bao gồm các phương thức public và protected có thể được kế thừa bởi các class exception khác. Thêm vào đó System. Exceptions chứa đựng các thuộc tính thuộc về tất cả các exceptions.
  - **Message:** Hiển thị một thông báo chỉ ra lý do cho exception.
  - **Source:** Cung cấp tên cho ứng dụng hoặc đối tượng trong trường hợp xảy ra exceptions.
  - **TackTrace:** Cung cấp chi tiết exception trên stack lúc exception được ném ra.
  - **InnerException:** trả về một trường hợp của exception hiện tại.

Một số lớp ngoại lệ

- Namespace System bao gồm các class exception khác nhau mà C# cung cấp. Các class Exception thường hay dùng là:

Tên lớp	Tên lớp
System.ArithmeticException	System.ArrayTypeMismatchException
System.DivideByZeroException	System.InvalidCastException
System.IndexOutOfRangeException	System.ArgumentNullException
System.NullReferenceException	

Điều khiển ngoại lệ

- Để điều khiển ngoại lệ C# sử dụng khối try…catch

Cú pháp:

```
try
{
    //các lệnh có thể gây ra lỗi lúc chạy
}catch(<loại_ngoại_lệ> biến)
{
    //các lệnh điều khiển khi xảy ra lỗi
}
```

```
int a = 10, b = 0, c;
try
{
    c = a / b;
}
catch (ArithmeticException ex)
{
    Console.WriteLine("Thong bao loi:" + ex.Message);
}
```

Khối catch tổng quát

- Trong trường hợp bạn không biết rõ loại ngoại lệ nào có thể xảy ra bạn có thể sử dụng lớp Exception trong khối catch để nhận bắt bất kỳ ngoại lệ nào.

```
string[] names = new string[3];
try
{
    names[3] = "Bach Khoa - Aptech";
}
catch (Exception ex)
{
    Console.WriteLine("Thông báo lỗi:" + ex.Message);
}
```

Từ khóa “throw”

- Từ khóa “throw” dùng để tung ra một ngoại lệ, nó nhận thể hiện của lớp ngoại lệ như là 1 tham số, trong trường hợp lớp ngoại lệ tung ra không đúng, trình biên dịch sẽ báo lỗi.

```
class Product
{
    private int price;
    public int Price
    {
        set
        {
            if (value < 0)
                throw new ArithmeticException("Gia phai >=0");
            else
                price = value;
        }
    }
}
```

```
try
{
    Product p = new Product();
    p.Price = 0;
}
catch (ArgumentException ex)
{
    Console.WriteLine(ex.Message);
}
```

Từ khóa “finally”

- “finally” là khối tùy chọn trong khối try…catch, khối này luôn được thực dù ngoại lệ có xảy ra hay không, thông thường nó sử dụng để đóng tệp tin hoặc kết nối…

```
//mở tệp tin
StreamWriter sw = new StreamWriter("abc.txt");
try
{
    sw.Write("Hello Bach Khoa - Aptech");//viết dữ liệu vào tệp tin
}
catch (Exception ex)
{
    Console.WriteLine("Thong bao loi:" + ex.Message);
}
finally
{
    sw.Close();//đóng tệp tin
}
```

Khối try...catch lồng nhau

- Một khối try…catch này lồng bên trong một khối try…catch khác và khi khối bên trong không bắt được ngoại lệ thì sẽ chuyển đến khối bên ngoài.

```
string[] names = { "Son", "Hai", "Dong" };
int num = 0;
int result;
try
{
    Console.WriteLine("Day la khoi try catch ngoai");
    try
    {
        result = 13 / num;
    }
    catch (ArgumentException ex)
    {
        Console.WriteLine(ex.Message);
    }
    names[3] = "chung";
}
catch (IndexOutOfRangeException ex)
{
    Console.WriteLine(ex.Message);
}
```

Nhiều khối “catch”

- C# cho phép tạo nhiều khối catch để bắt các kiểu exception khác nhau được ném ra từ khối try. Khi khối try sinh ra exception trình biên dịch tìm khối catch tuần tự, nếu không được trình biên dịch sẽ báo lỗi và qua khối finally nếu có.

```
string[] names = { "Son", "Hai", "Dong" };
int num = 0;
int result;
try
{
    result = 13 / num;
    names[3] = "chung";
}
catch (IndexOutOfRangeException ex)
{
    Console.WriteLine(ex.Message);
}
catch (ArgumentException ex)
{
    Console.WriteLine(ex.Message);
}
```

Ngoại lệ tự định nghĩa 1-2

- Trong trường hợp những ngoại lệ của hệ thống không đáp ứng được nhu cầu bắt lỗi của bạn, bạn có thể tự định nghĩa những ngoại lệ cho chương trình của bạn, tuy nhiền ngoại lệ bạn tạo ra phải kế thừa từ lớp ngoại lệ chung của hệ thống là “Exception”
- Khi muốn áp dụng ngoại lệ của bạn định nghĩa, bạn dùng từ khóa “throw” để tung ra ngoại lệ

Ngoại lệ tự định nghĩa 2-2

```
class Product
{
    private int price;
    public int Price
    {
        set
        {
            if (value < 0)
                //tung ngoại lệ do người dùng định nghĩa
                throw new InvalidPriceException("Gia phai >=0");
            else
                price = value;
        }
    }
}
//ngoại lệ do người dụng định nghĩa
class InvalidPriceException : Exception
{
    public InvalidPriceException(string msg)
        : base(msg)
    {
    }
}
```

HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH