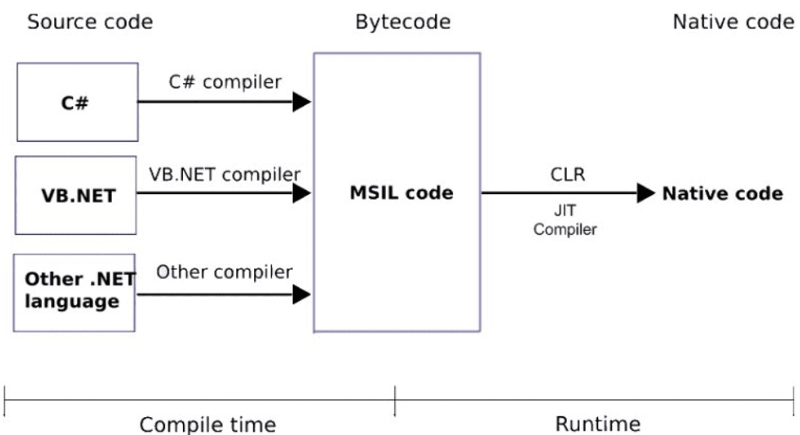
**C# là gì?**



* C# là ngôn ngữ hiện đại, hướng đối tượng được phát triển dựa trên Java và C++.
* C# dùng để tạo một ứng dụng Windows Forms hay Windows Presentation Foundation, phát triển game, ứng dụng Web, ứng dụng Mobile.

**Đặc trưng của C#**

**Là ngôn ngữ đơn giản.**

* Loại bỏ các macro, template, đa kế thừa, và lớp cơ sở ảo.

**Là ngôn ngữ hiện đại.**

* C# có những đặc tính như xử lý ngoại lệ, thu gom bộ nhớ tự động, những kiểu dữ liệu mở rộng, và bảo mật mã nguồn.

**Là một ngôn ngữ lập trình thuần hướng đối tượng**

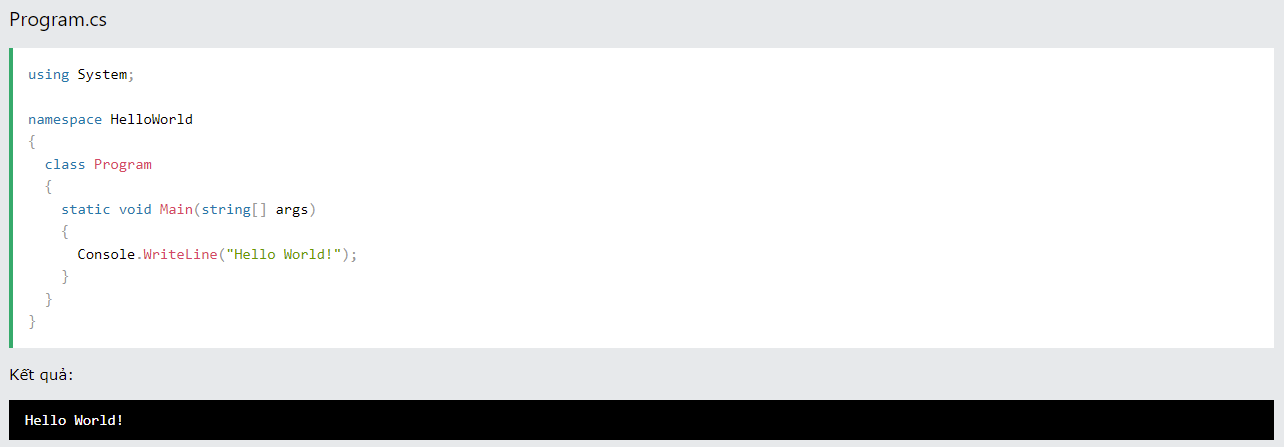
* Là một phương pháp lập trình có 4 tính chất:
  + Tính trừu tượng(abstraction).
  + Tính đóng gói(encapsolution).
  + Tính đa hình(polymophism).
  + Tính kế thừa(inheritance).

**Ngôn ngữ ít từ khóa**

**Nền tảng .NET**

* .NET Framework chỉ hoạt động trên windows. Những nền tảng ứng dụng như WPF, Winforms, ASP.NET(1-4), hoạt động dựa trên .NET Framework.
* Mono là phiên bản cộng đồng mang .NET đến những nền tảng ngoài Windows. Mono được phát triển chủ yếu nhằm xây dựng những ứng dụng với giao diện người dùng và được sử dụng rất rộng rãi: Unity Game, Xamarin...
* Phát triển đa nền tảng .NET core, hiện được sử dụng trong các ứng dụng Universal Windows Platform và ASP.NET core. Từ đây C# có thể sử dụng để phát triển các ứng dụng đa nền tảng.

**Cú pháp**

****

**Giải thích:**

* Using System: từ khóa using là để sử dụng các chức năng của System namspace trong chương trình. Có thể sử dụng nhiều using.
* Namespace: là một tập hợp các lớp. HelloWorld chứa lớp Program.
* Class Program: chứa các định nghĩa dữ liệu và phương thức mà chương trình sử dụng. Chỉ được phép có một phương thức main.
* Phương thức main: là điểm bắt đầu của chương trình khi chạy.
* Console là một lớp của System namespace, có phương thức WriteLine để in/xuất văn bản. Nếu không viết using System thì ta phải viết:
  + System.Console.WriteLine(): Để in/xuất văn bản.

Lưu ý:

* Mọi câu lệnh đều kết thức bằng ;.
* Có phân biệt chữ hoa chữ thường.
* Tên tệp không nhất thiết phải cùng tên lớp.

**Một số cú pháp cơ bản**

**1, In/xuất.**

* Sử dụng phương thức Write của lớp Console để in/xuất ra màn hình.



* Sử dụng phương thức WriteLine của lớp Console để in/xuất sau đó xuống dòng.

****

**2, Comment.**

* //: để comment trên 1 dòng.
* /\* content \*/: để comment trên nhiều dòng.

**3, Nhập liệu**

* Sử dụng phương thức Readline() để nhập vào thông tin từ người dùng.
* Phương thức Readline() trả về một string nên khi muốn lấy 1 số nguyên khi nhập thì ta phải convert nó về kiểu số nguyên

VD: age = Convert.ToInt32(Console.Readline());

**Bài 1: Biến**

**1, Biến C#**

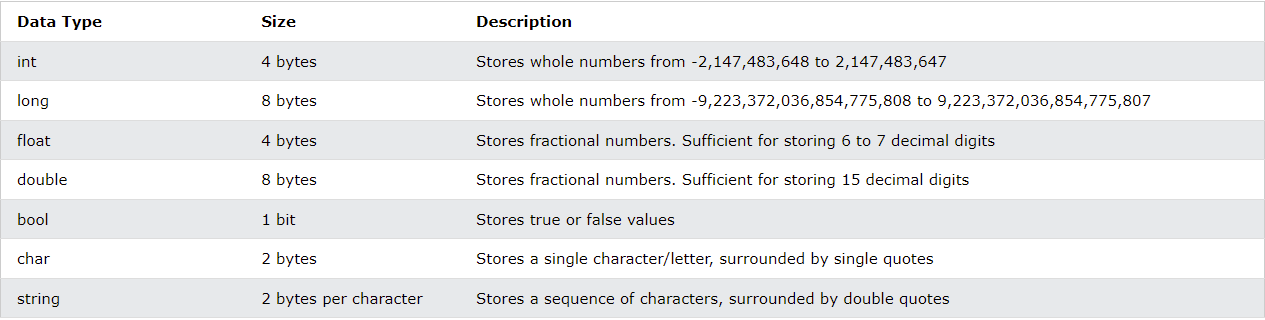
Là vùng chứa để lưu trữ các giá trị dữ liệu.

Ta khai báo biến hằng số bằng cách sử dụng từ khóa const đằng trước. Biến hằng yêu cầu cung cấp giá trị ban đầu.

**3, Quy tắc đặt tên biến**

* Tên có thể chứa các chữ cái, chữ số và ký tự gạch dưới (\_)
* Tên phải bắt đầu bằng một chữ cái
* Tên phải bắt đầu bằng chữ thường và không được chứa khoảng trắng
* Tên phân biệt chữ hoa chữ thường ("myVar" và "myvar" là các biến khác nhau)
* Các từ dành riêng (như từ khóa C #, chẳng hạn như inthoặc double) không thể được sử dụng làm tên.

**Bài 2: Các kiểu dữ liệu biến**

****

* khai báo kiểu long nên thêm L vào sau giá trị:

VD: long = 1234234523L;

* tương tự cho float và double lần lượt là F và D.
* Để lưu giá trị lũy thừa của 10 ta sử dụng e.

VD: 10e3 = 1000 = 10 \* 10 \* 10.

**1, Ép kiểu**

**a, Tự động**

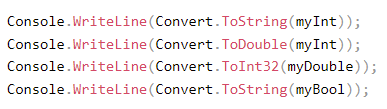
* Khi chuyển từ kiểu dữ liệu kích thước nhỏ sang kiểu dữ liệu kích thước lớn hơn.
  + Char->int->long->float->double
* Ta chuyển bằng cách lấy biến có kích thước lớn gán giá trị bằng giá trị của biến kích thước nhỏ.

**b, Thủ công**

* Khi chuyển tử kiểu dữ liệu nhỏ sang kiểu dữ liệu thấp
* Ta sử dụng (kiểu dữ liệu)biến.



**c, Sử dụng phương thức**

****

* Ta sử dụng phương thức Convert... để chuyển về kiểu dữ liệu mong đợi.

**Bài 3: Toán tử**

* Tương tự các ngôn ngữ C++, Java.

**Toán tử hỗn hợp**

* Sizeof(): trả về kích thước của một kiểu dữ liệu(byte).

VD: sizeof(int) = 4.

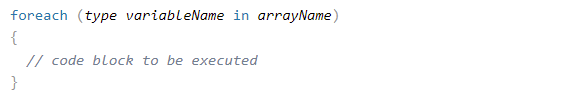
* Typeof(): trả về kiểu của một lớp.
* **Is**: kiểu tra đối tượng có là 1 kiểu cụ thể hay k

VD: if(xemay is xe).

* As: ép kiểu mà không tạo một exception nếu việc ép kiểu thất bại.

**Bài 4: Vòng lặp**

* Vòng lặp Foreach:

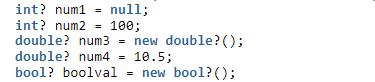
****

Thay **:** thành **in.**

**Bài 5: Kiểu nullable trong c#**

**1, Kiểu nullable**

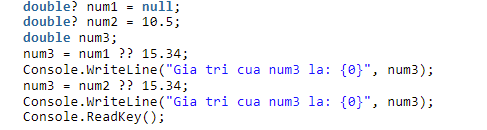
* Kiểu nullable là một kiểu dữ liệu đặc biệt, ta có thể gán các giá trị thông thường cũng như các giá trị null.
* Cú pháp:  
  ****

****

* Console.Readkey(): để làm cho chương trình kết thúc khi ta nhấn một phím bất kỳ.

**2, Toán tử ?? trong C#**

* Toán tử **null Coalescing** hay toán tử ?? trong C# được sử dụng với kiểu **giá trị nullable** và **các kiểu tham chiếu**. Nó dùng để chuyển đổi 1 toán hạng tới toán hạng với kiểu nullable, chuyển ngầm định. Toán tử ?? có hai toán hạng, nếu toán hạng đầu tiên là null thì trả về giá trị toán hạnh 2 và ngược lại.
* Ví dụ:

****

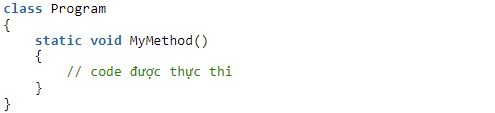
****

**{i}: biểu thị giá trị thứ i.**

**Bài 6: Phương thức trong C#**

* Một phương thức trong C# là một khối lệnh mà chỉ được thực thi khi được gọi.
* Ta có thể truyền tham số vào phương thức.
* Phương thức còn được gọi là hàm để thực hiện một hành động nào đó.
* Phương thức có thể tái sử dụng.

**1, Tạo phương thức**

****

MyMethod: là tên phương thức

Static: nghĩa là phương thức này thuộc về lớp chứ không phải là của đối tượng.

Void: là kiểu trả về của phương thức.

*Lưu ý: Tên phương thức nên viết hoa.*

**Bài 7: Lớp và đối tượng trong C#**

Lớp là một nhóm đối tượng có các thuộc tính chung. Nó là mẫu, thiết kế từ đó đối tượng được tạo ra.

Một **thực thể** có trạng thái và hành vi gọi là đối tượng.

Đối tượng là một thể hiện(instance) của một lớp.

**1, Tạo một lớp**

* Nên đặt tên file cùng với tên lớp(không bắt buộc) nhưng nó sẽ làm cho mã có tổ chức.

**2, Tạo một đối tượng**

* chỉ định tên lớp theo sau là tên đối tượng và sử dụng từ khóa new.
* Ta sử dụng dấu . để truy cập các thuộc tính và các phương thức của lớp.
* Một lớp có thể có nhiều đối tượng.

**Bài 8: Mảng trong C#**

* Mảng là tập hợp chứa các phần tử cùng kiểu được lưu trữ gần nhau trong bộ nhớ.

**1, Tạo một mảng.**

* Khai báo kiểu dữ liệu với [] sau đó là tên mảng.

VD: Khai báo mảng arr kiểu số nguyên với kích thước là 5 phần tử.

****

VD: Khai báo mảng kiểu số nguyên với các giá trị 1, 2, 3.



* Để truy cập các phần tử ta tử dụng index.

VD: 

* Để lấy ra độ dài của mảng ta sử dụng thuộc tính length.
* Để duyệt mảng ta nên sử dụng vòng lặp foreach.

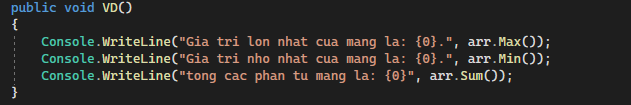


* Có nhiều phương thức sẵn cho mảng, ví dụ Sort(), sắp xếp mảng tăng dần.

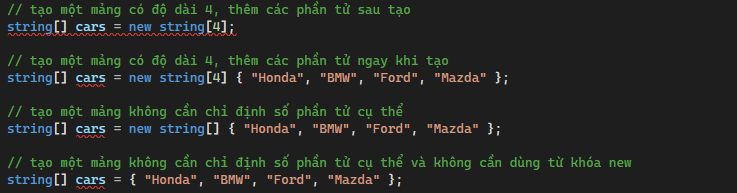


**2, System.Linq Namespace**

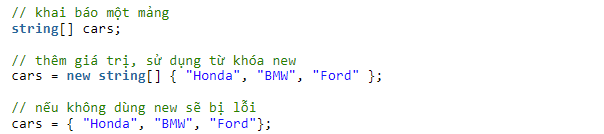
* Có nhiều phương thức xử lý mảng hữu ích khác như Max(), Sum, Min() ta có thể tìm thấy trong System.Linq Namespace. Để sử dụng chúng ta cần using System.Linq.



Lưu ý:



**Nếu như ta khai báo trước rồi mới khởi tạo giá trị thì ta phải sử dụng từ khóa new.**

****

**Bài 9: Chuỗi trong C#**

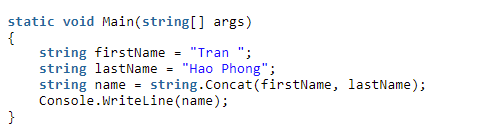
* Chuỗi dùng để lưu trữ văn bản.
* Một biến string được bao quanh bới “ .... “.

**1, Độ dài chuỗi.**

* Để lấy ra độ dài chuỗi ta tử dụng thuộc tính length.

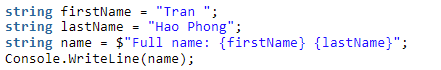
**2, Nối chuỗi**

* Ta có thể sử dụng toán tử +
* Ta sử dụng phương thức string.concat(s1, s2);

****

**3, Nội suy chuỗi**

Một tùy chọn khác của nối chuỗi, là nội suy chuỗi , thay thế các giá trị của biến trong một chuỗi.



* Ta phải sử dụng $ khi nội suy chuỗi.

**4, Cắt chuỗi**

* Sử dụng phương thức substring(index, leng): lấy ra leng ký tự từ vị trí index

**Bài 10: Enum trong C#**

**1, Tạo enum**

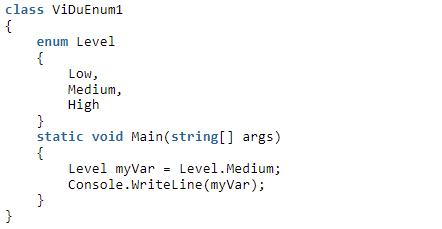
**Enum là một lớp đặc biệt đại diện cho một nhóm các hằng số(biến không thể thay đổi giá trị).**

Sử dụng từ khóa enum để tạo một enum(thay vì class hoặc interface) và phân tích các enum bằng dấu ,.

* Ta có thể truy cập phần tử enum bằng cách sử dụng .

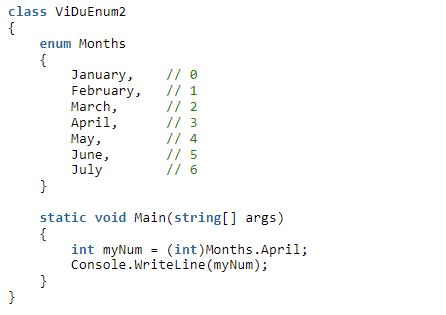


**2, enum bên trong một lớp**

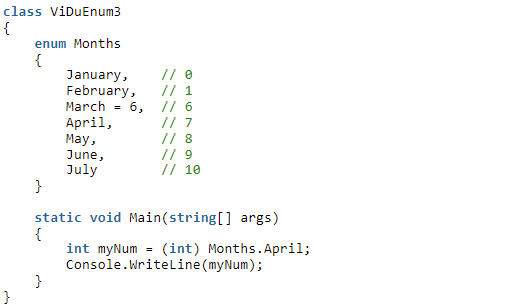
**-> kết quả: Medium.**

**2, Giá trị Enum**

* theo mặc định, mục đầu tiên của enum có giá trị là 0.

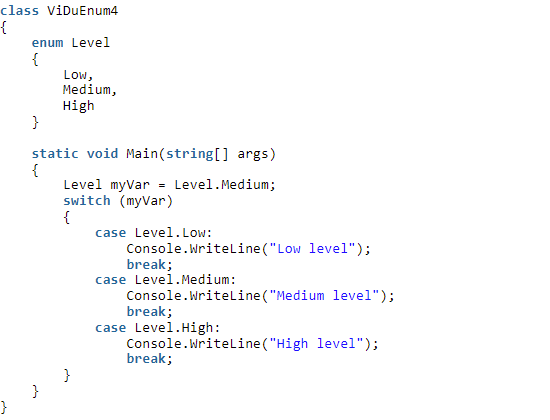
**-> 3**

**-** ta cũng có thể gán giá trị cho các giá trị của enum.

**-> 7**

**3, Sử dụng Enum trong câu lệnh switch**

* enum thường được sử dụng trong các câu lệnh switch để kiểm tra các giá trị tương ứng.



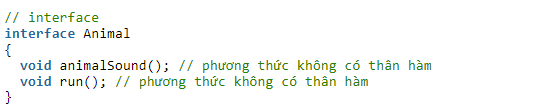
**-> medium Level**

**Khi nào nên sử dụng Enum?**

* Sử dụng khi có các giá trị không thay đổi, như ngày tháng năm, màu sắc...

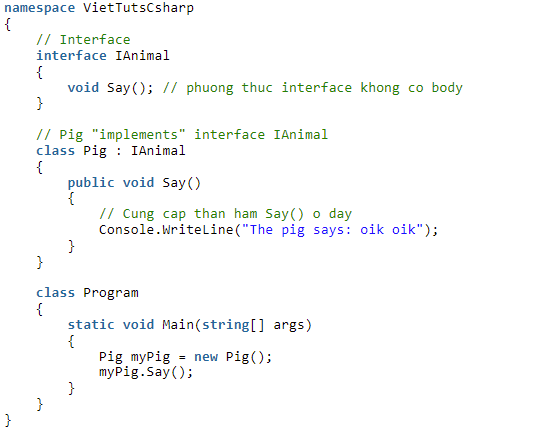
**Bài 11: Interface trong C#**

**1, Interface là gì?**

* Một cách khác để đạt được sự trừu tượng trong C# là interface.
* Là một bản thiết kế của một lớp, có các thuộc tính phương thức trừu tượng.
* Là một kỹ thuật để thu được tính trừu tượng và đa kế thừa. 

**Chú ý:** các thành viên của interface là **abstract** và **public**. Nó chứa các thuộc tính và phương thức nhưng không chứa các mô tả của các phương thức

* Interface phải được triển khai(được kế thừa) bởi một lớp khác. Phần thân của interface được cung cấp bới lớp ‘implement’ ta không phải sử dụng override khi triển khai thân của interface.



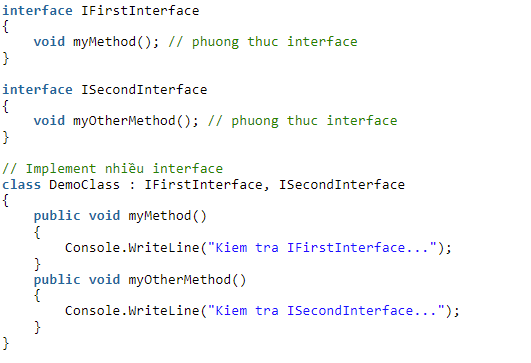
The pig says: oik oik

**Lưu ý:**

* Interface không thể được sử dụng để tạo đối tượng.
* Các phương thức interface không có phần thân, phần thân nó được cung cấp bới lớp ‘implement’
* Khi triển khai một interface ta phải ghi đè tất cả các phương thức của nó.
* Interface có thể chứa các thuộc tính, phương thức nhưng không phải các trường/biến.
* Các thành phần của interface theo mặc định là **public** và **abstract**.
* Interface không thể chứa constructor và nó không sử dụng để tạo đối tượng.

**2, implement nhiều interface trong C#.**

* Ta có thể implement nhiều interface bằng cách phân tách nhau bằng dấu ,.

****

**Khi nào nên sử dụng interface.**

* Để đạt được độ bảo mật – ẩn đi các chi tiết nhất định và hiển thị chi tiết quan trọng của một đối tượng.
* C# không hỗ trợ đa kế thừa, tuy nhiên có thể đa kế thừa interface. Bới vì lớp có thể implement nhiều interface.

**Bài 12: Namespace trong C#**

* Cung cấp 1 cách để giữ các tập hợp phân biệt. Các lớp trong khác namespace không xung đột khi cùng tên.
* **Truy xuất bên trong namespace bằng cách:** name\_namespace.element.
* Để xử dụng namespace ta sử dụng từ khoá: using name\_namespace.
  + Khi đó ta có thể khai báo đối tượng bằng chính tên class bên trong.
  + Dễ thấy. Console là 1 lớp bên trong namespace System.
  + Ta có thể sử dụng System.Console hoặc Console.
* Có thể khai báo namespace lồng nhau.
* Hiểu namespace nó như một package trong java, using namespace cũng như import package.

**Bài 13: Xử lý ngoại lệ trong C#**

Xử lý ngoại lệ nghĩa là xử lý các lỗi exceptions, giúp chương trình xuyên suốt không bị gián đoạn.

Từ khoá try, catch, finally và throw

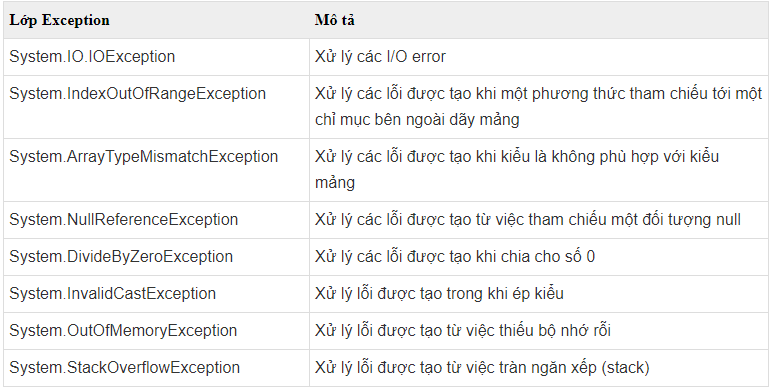
**Lớp Exception trong C#**

* Các lớp exception kế thừa một cách trực tiếp hoặc không trực tiếp từ lớp System.Exception.

VD: System.ApplicationException và System.SystemException

* Lớp System.ApplicationException: hỗ trợ các exception tạo bới chương trình ứng dụng. Các exception tự định nghĩa nên kế thừa phương thức này.
* Lớp System.SystemException: là lớp cơ sở cho tất cả system exception tiền định nghĩa

Các lớp exception tiền định nghĩa:



**Từ khoá throw**: dùng để ném ra một ngoại lệ trực tiếp hoặc gián tiếp từ lớp System.Exception trong C#.

**Bài 14: Đọc ghi file trong C#**

**Bài 15: OOP**

1, Ưu điểm

* Nhanh và dễ thực thi hơn
* Cung cấp cấu trúc rõ ràng
* Không lặp lại code, dễ bảo trì, sửa đổi và gỡ lỗi
* Tối ưu mã, thời gian phát triển
* Lớp là một khuôn mẫu cho các đối tượng và đối tượng là các thể hiện của mẫu.

2, Thành phần

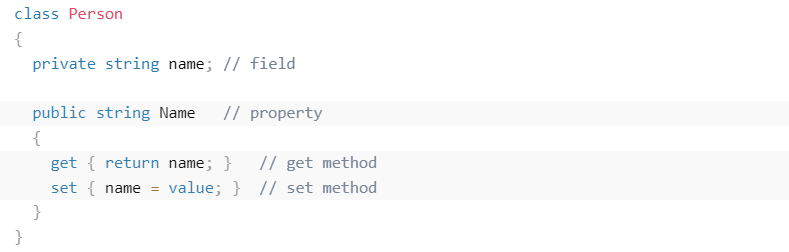
* Lớp gồm các fields(trường) và các method(phương thức).
* Hàm tạo được sử dụng để khởi tạo đối tượng. Nó sẽ được gọi khi đối tượng được tạo ra.
  + Khớp với tên lớp, không có kiểu trả về.

3, Phạm vi truy cập

* Cung cấp khả năng truy cập các trường và các phương thức cho các đối tượng.
  + Private: chỉ cho phép truy cập trong cùng một lớp
  + Public: Cho phép truy cập ở tất cả các lớp
  + Protected: Truy cập trong cùng một lớp hoặc lớp kế thừa.
  + Internal: truy cập trong cùng một assembly.

4, Tính đóng gói(encapsulation)

* Khai báo quyền private
* Sử dụng các phương thức get, set.
  + Get, set thủ công.



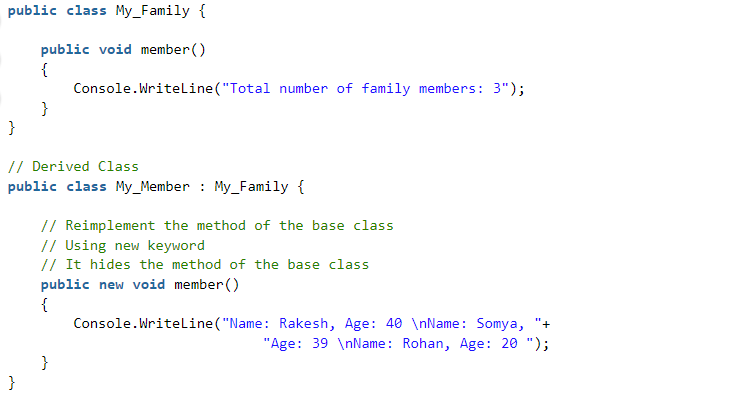


5, Tính kế thừa

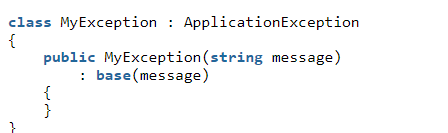
* Để kế thừa từ 1 lớp ta sử dụng ký tự ‘:’.
* Nếu không muốn lớp khác kế thừa lớp hiện tại thì thêm vào **sealed** vào trước class name\_class.
* Thêm từ khóa virtual vào phương thức ở lớp cơ sở + overide để ghi đề phương thức ở lớp dẫn xuất.

6, Trừu tượng hóa

* Trừu tượng hóa để ẩn đi chi tiết cài đặt
* Lớp trừu tượng để hạn chế khởi tạo đối tượng
* Phương thức trừu tượng không có phần thân, để che dấu cài đặt chi tiết bên trong phương thức.
* Để truy cập lớp trừu tượng nó được kế thừa từ lớp khác.
* Để kế thừa và ghi đè phương thức abstract thì ta sử dụng : và override
* Ta có thể sử dụng phương thức ẩn bằng từ khoá new trong lớp dẫn xuất

****

Từ khoá base: **Tương tự super trong Java.**



* Base ở đây sẽ gọi tới hàm tạo của lớp dẫn xuất, và có tham số đó là message.

**Bài 16: Nạp chồng toán tử**

* Sử dụng các toán tử với các kiểu dữ liệu tự định nghĩa

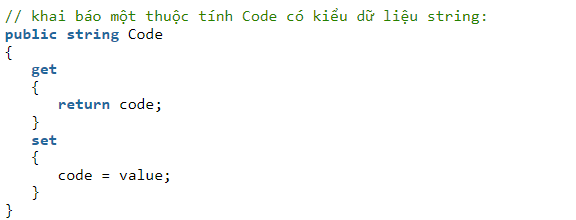
****

**Bài 17: Property trong C#**

1, property trong C#

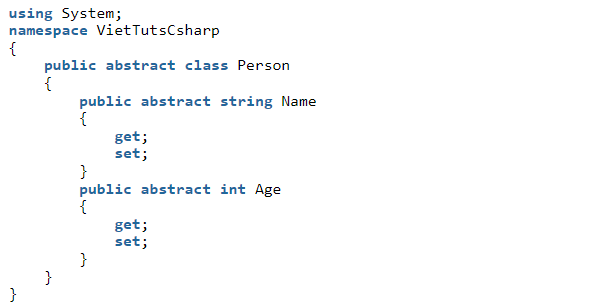
* Là các thành phần được đặt tên của lớp, cấu trúc, và interface.
* Các biến thành viên trong một lớp được gọi là các field
* Thuộc tính là một sự kế thừa các field và được truy cập bởi sử dụng cùng cú pháp.
* Từ đó nhờ accessor ta có thể đọc, ghi được các private field.
* Property không đặt tên vị trí lưu trữ. Chúng có accessor để đọc và ghi, tính toán.

2, accessor trong c#

* Accessor là một thuộc tính chứa các lệnh có thể thực thi, 

3, Thuộc tính astract trong C#

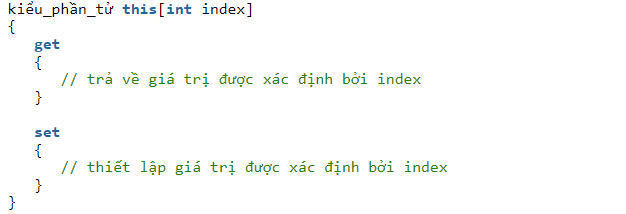
* Trong lớp asctract có thể có các thuộc tính abstract



**Bài 18: Indexer trong C#**

Một indexer trong class cho phép đối tượng có thể lập chỉ mục, ví dụ như mảng. khi chỉ định một indexer cho một lớp nghĩa là nó vận hành như một mảng ảo(virtual array). Sau đó ta có thể truy cập thể hiện của lớp bởi toán tử []

Cú pháp:



**Sử dụng indexer**

* Việc khai báo hành vi của một indexer cũng như thuộc tính.
* Sử dụng accessor get và set để định nghĩa.
* Các thuộc tính thì trả về hoặc thiết lập một thành phần dữ liệu cụ thể trong khi indexer thì trả về hoặc thiết lập thể hiện của một đối tượng.
* Với thuộc tính thì việc xác định là dùng tên nhưng đối với indexer thì nó sử dụng từ khoá this.

**Nạp chồng indexer**

* Indexer có thể đang dạng kiểu dữ liệu cho tham số.

**Bài 19: delegate trong C#**

* Delegate tương tự con trỏ tới các hàm, trong c hoặc C++.
* Delegate là một biến tham chiếu mà giữ tham chiếu tới một phương thức. tham chiếu đó có thể thay đổi tại runtime.
* Delegate sử dụng để triển khai các sự kiện và phương thức call back, tất cả delegate được kế thừa tử system.delegate.

1, Khai báo delegate.

* Delegate có thể tham chiếu tới một phương thức, mà có cùng dấu hiệu như delegate đó.

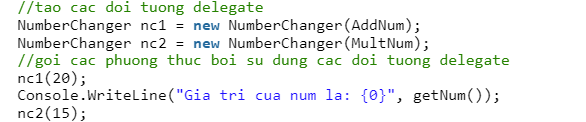


* Delegate trên có thể được sử dụng để tham chiếu tới bất kỳ phương thức mà có một tham số string và trả về một số nguyên.

**Khai báo delegate**



Khi một kiểu delegate được khai báo, đối tượng delegate phải được khởi tạo bằng từ khoá new và được liên kết với phương thức cụ thể.



2, Multicast một delegate trong C#

* Ta sử dụng + hoặc - để có thể sử dụng để gọi nhiều hơn một phương thức cùng kiểu.

3, Cách sử dụng