LẬP TRÌNH WINDOWS

BÀI 3: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#

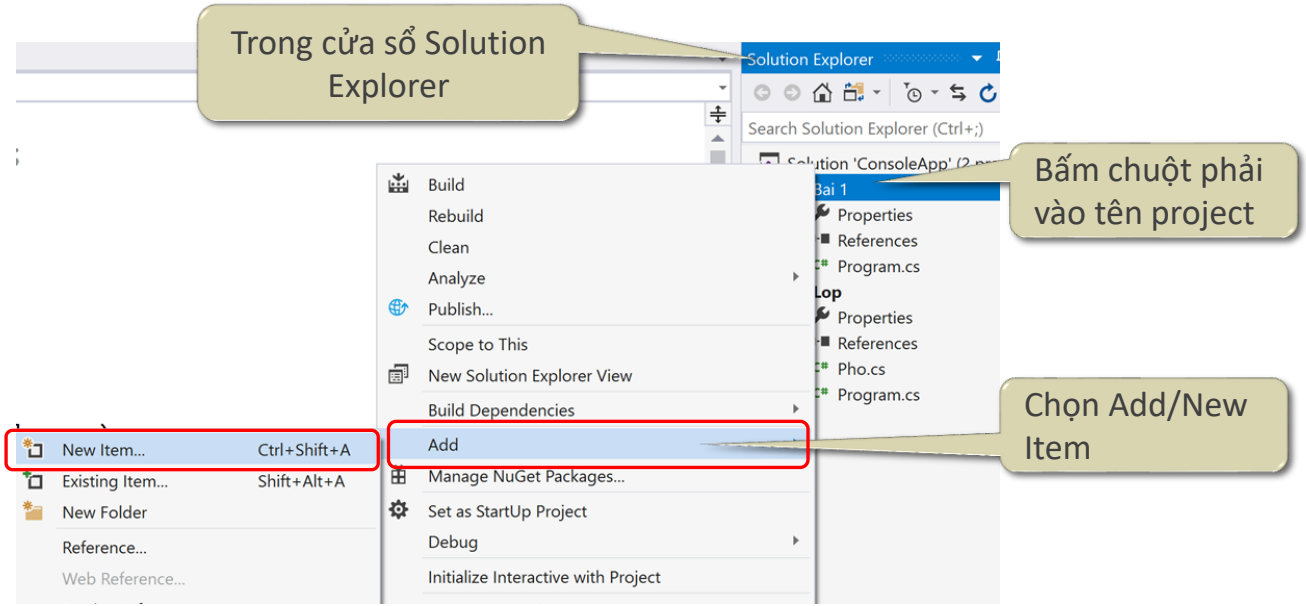
Giảng viên: Lý Anh Tuấn

Email: tuanla@wru.vn

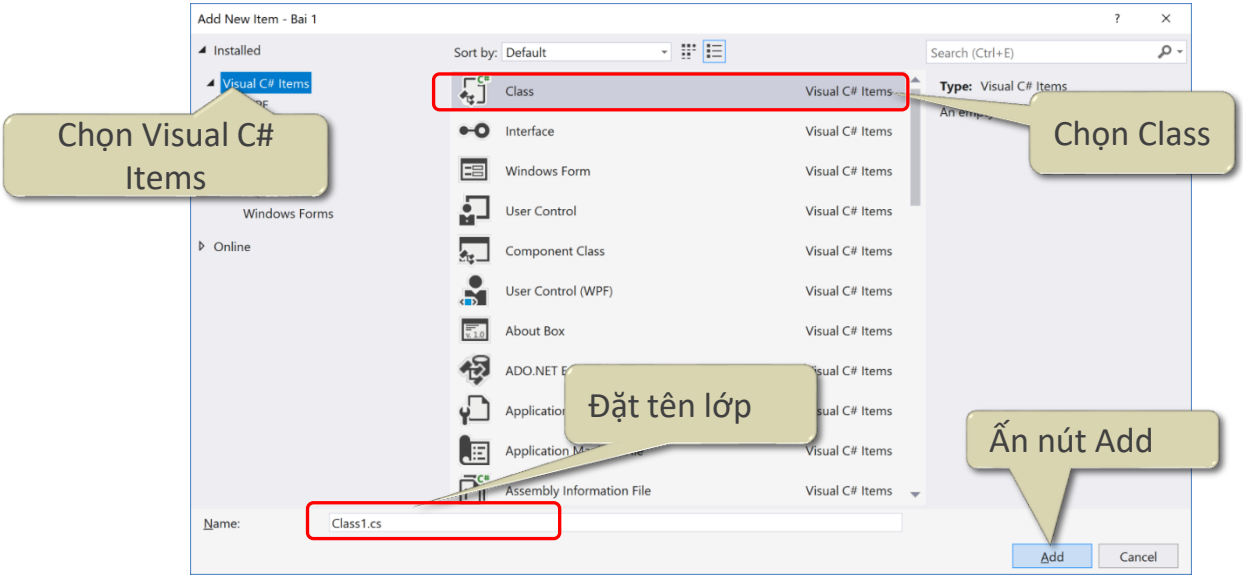
Nội dung

- Khai báo lớp
- Định nghĩa lớp
- Sử dụng lớp

Khai báo lớp


Trong cửa sổ Solution Explorer
Solution Explorer
Search Solution Explorer (Ctrl+;)
Bấm chuột phải vào tên project
Build
Rebuild
Clean
Analyze
Publish...
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Set as StartUp Project
Debug
Initialize Interactive with Project
Properties
References
Program.cs
Properties
References
Pho.cs
Program.cs
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Folder
Reference...
Web Reference...
Chọn Add/New Item

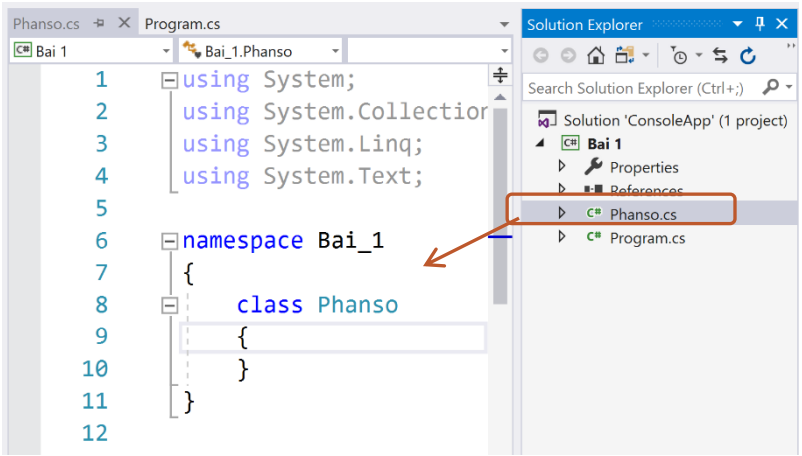
Khai báo lớp


Add New Item - Bai 1
Installed
Visual C# Items
Windows Forms
Online
Sort by: Default
Search (Ctrl+E)
Class
Interface
Windows Form
User Control
Component Class
User Control (WPF)
About Box
Assembly Information File
Visual C# Items
Type: Visual C# Items
Name:
Class1.cs
Add
Cancel
Chọn Visual C# Items
Chọn Class
Đặt tên lớp
Ấn nút Add

Ví dụ

- Tạo một lớp Phanso thuộc project Bai 1

  => xuất hiện file Phanso.cs trong project Bai 1



Định nghĩa lớp

- Cú pháp:

[quyền truy cập] class <tên lớp> [:lớp cơ sở]

```
{
    [quyền truy cập] <kiểu dữ liệu> <tên thành phần>;
    [quyền truy cập] <kiểu trả về> <tên phương thức>(danh
        sách các tham số)
    {
                    //định nghĩa phương thức
    }
}
```

Định nghĩa lớp

- Trong đó:
  - class: là từ khóa để khai báo lớp
  - Kiểu dữ liệu: là những kiểu cơ bản hoặc những kiểu đã được định nghĩa
  - Kiểu trả về: là những kiểu cơ bản hoặc những kiểu đã được định nghĩa hoặc void (phương thức không trả về dữ liệu)

Định nghĩa lớp

- Trong đó:
  - Quyền truy cập: là các quyền được liệt kê trong bảng sau

Định nghĩa lớp

Từ khóa	Giới hạn truy cập
public	Không hạn chế. Những thành viên được đánh dấu là public có thể được dùng bởi bất kì các phương thức nào của bất kỳ lớp nào
private	Che dấu. Những thành viên được đánh dấu là private thì chỉ được sử dụng trong các phương thức của lớp
protected	Thành viên nào được đánh dấu là protected trong lớp X thì chỉ được dùng trong lớp X và các lớp **dẫn xuất** từ X
internal	Mức độ truy cập nội bộ, bị giới hạn trong một assembly
protected internal	Thành viên nào được đánh dấu là protected internal trong lớp X thì được sử dụng trong lớp X, các lớp dẫn xuất từ X và các lớp cùng assembly với X

Ví dụ

Không khai báo quyền truy cập thì mặc định là private

```
namespace Bai_1
{
    class Phanso
    {
        int tuso, mauso;
        public void nhap()
        {
            Console.Write("Tu so = ");
            tuso = Convert.ToInt32(Console.ReadLine());
            Console.Write("Mau so = ");
            mauso = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

Chú ý

- Phương thức (method) là các hàm (function)
- Tên phương thức thường được đặt theo tên của hành động
- Tham số của phương thức nằm trong cặp ngoặc tròn ngay sau tên phương thức
- Muốn truyền tham chiếu thì nhớ thêm từ khóa **ref** hoặc **out**

Sử dụng lớp

- Khai báo đối tượng
  - <tên lớp> <tên đối tượng>;
- Ví dụ:

  Phanso ps1, ps2; //Khai báo 2 đối tượng ps1 và ps2 thuộc kiểu Phanso

Sử dụng lớp

- Khởi tạo đối tượng

  <tên đối tượng> = new <hàm tạo của lớp>;

- Ví dụ:

  ps1 = new Phanso();//khởi tạo ps1 bằng hàm tạo mặc định

  ps2 = new Phanso(1,2);//khởi tạo ps2 bằng hàm tạo có tham số

Sử dụng lớp

- Truy cập tới dữ liệu và hàm thành viên của đối tượng:
  - Phụ thuộc vào quyền được truy cập vào dữ liệu đó
  - Sử dụng toán tử (.) để truy cập
  - Ví dụ:

```
ps1.nhap(); //gọi hàm nhập của lớp Phanso
ps1.xuat(); //gọi hàm xuất của lớp Phanso
```

Ví dụ sử dụng lớp

```
class Program
{
    static void Main(string[] args)
    {
        Phanso A = new Phanso();//khai báo và khởi tạo đối tượng
        Phanso B = new Phanso();
        Phanso C = new Phanso();
        Console.WriteLine("Nhap phan so A: ");
        A.nhap();
        Console.WriteLine("Nhap phan so B: ");
        B.nhap();
        C = A + B;
        Console.WriteLine("\nTong 2 phan so la: ");
        A.xuat(); Console.Write(" + "); B.xuat(); Console.Write(" = "); C.xuat();
        Console.ReadLine();
    }
}
```

Thuộc tính (property)

- Mẫu cài đặt thuộc tính:

```
public <kiểu dữ liệu> <tên thuộc tính>
{
    get {
        return <giá trị thuộc tính>;
    }
    set {
        <giá trị thuộc tính> = value;
    }
}
```

Thuộc tính (property)

- get: Đọc thuộc tính
- set: Gán giá trị cho thuộc tính
- tên thuộc tính: Đặt tên bất kỳ theo quy ước nhưng nên đặt dễ nhớ (tốt nhất là trùng tên với tên biến thành viên và ký tự đầu viết hoa)

Ví dụ

```
class CViDu
{
    private int a, b;
    public int A
    {
        get {return a;}
        set {a=value;}
    }
    public int B
    {
        get {return b;}
        set {b = value; }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        CViDu vd = new CViDu();
        vd.A = 5;
        vd.B = 4;
        int x = vd.A;
     }
}
```

Nạp chồng toán tử

- Cú pháp:

  public static <kiểu trả về> operator <toán tử>(danh sách tham số)

- Trong đó:
  - Kiểu trả về là kiểu kết quả của phép tính
  - Danh sách tham số bao gồm kiểu và tên tham số

Ví dụ

```
public static Phanso operator +(Phanso a, Phanso b)
{
    Phanso kq = new Phanso();
    kq.tuso = a.tuso * b.mauso + b.tuso * a.mauso;
    kq.mauso = a.mauso * b.mauso;
    return kq;
}
```

Bài tập

- Xây dựng một lớp phân số (đặt tên là PhanSo) bao gồm:
  - Các thành phần dữ liệu tử số và mẫu số
  - Phương thức nhập và hiển thị dữ liệu cho phân số
  - Xây dựng phương thức nạp chồng toán tử +, - , *, /
- Trong chương trình chính, khai báo và nhập dữ liệu cho 2 phân số ps1, ps2. Tính toán và hiển thị các kết quả sau:
  - ps3 = ps1 + ps2 (VD: 1/2 + 2/3 = 7/6)
  - ps3 = ps1 * ps2
  - ps3 = ps1 – ps2
  - ps3 = ps1 / ps2

Bài tập

- Cải tiến bài toán trên:
  - Thêm phương thức rút gọn phân số
  - Hiển thị những phân số kết quả dưới dạng rút gọn

LẬP TRÌNH WINDOWS

BÀI 4: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C# (TIẾP)

Giảng viên: Lý Anh Tuấn

Email: tuanla@wru.vn

Nội dung

- Kế thừa
- Đa hình
- Hàm ảo
- Lớp trừu tượng
- Giao diện

Kế thừa

- Là tính năng dùng lại trong lập trình hướng đối tượng
- Khai báo một lớp dựa trên lớp đã tồn tại
- Lớp đã tồn tại gọi là lớp cơ sở hoặc lớp cha (**Base class**)
- Lớp kế thừa gọi là lớp dẫn xuất hoặc lớp con (**Derived class**)
- C# không cho phép đa kế thừa lớp

Kế thừa

- Cú pháp khai báo

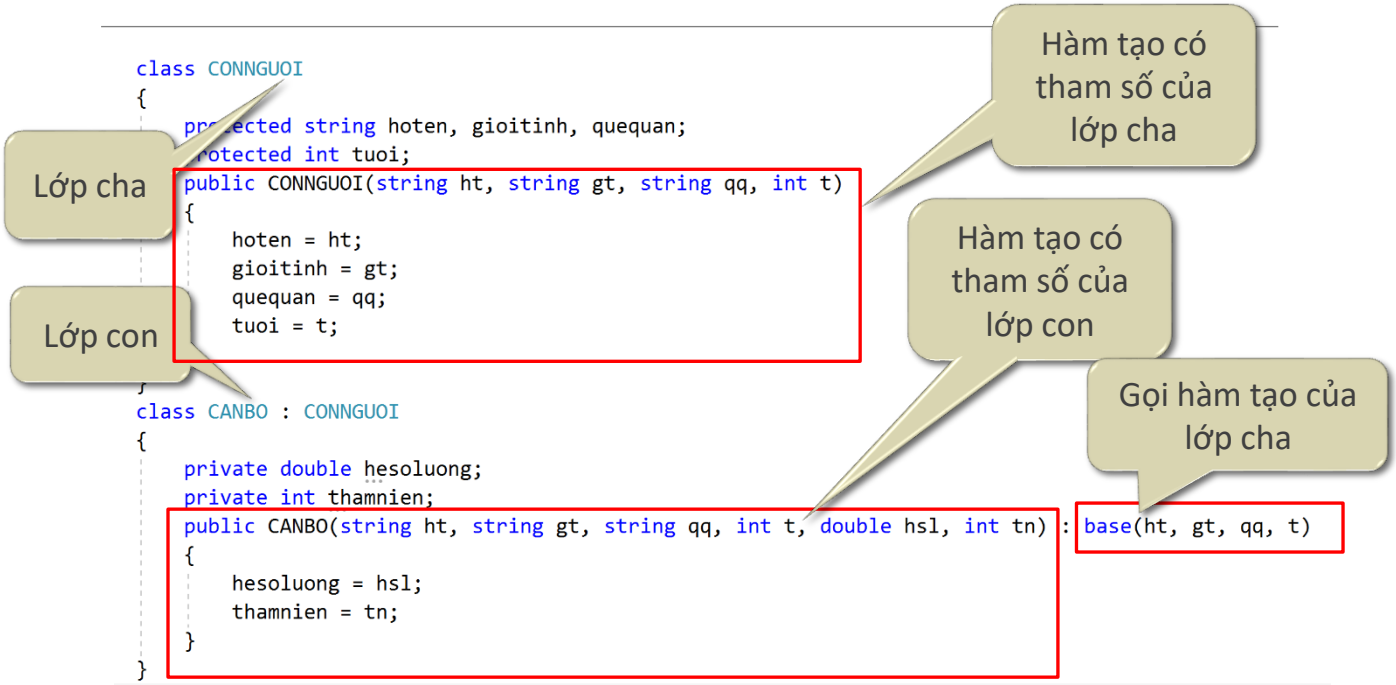
```
class <tên lớp dẫn xuất> : <tên lớp cơ sở>
{
    //định nghĩa lớp dẫn xuất
}
```

- Chú ý: Một số thành phần không được kế thừa
  - Các hàm tạo
  - Các hàm hủy

Kế thừa

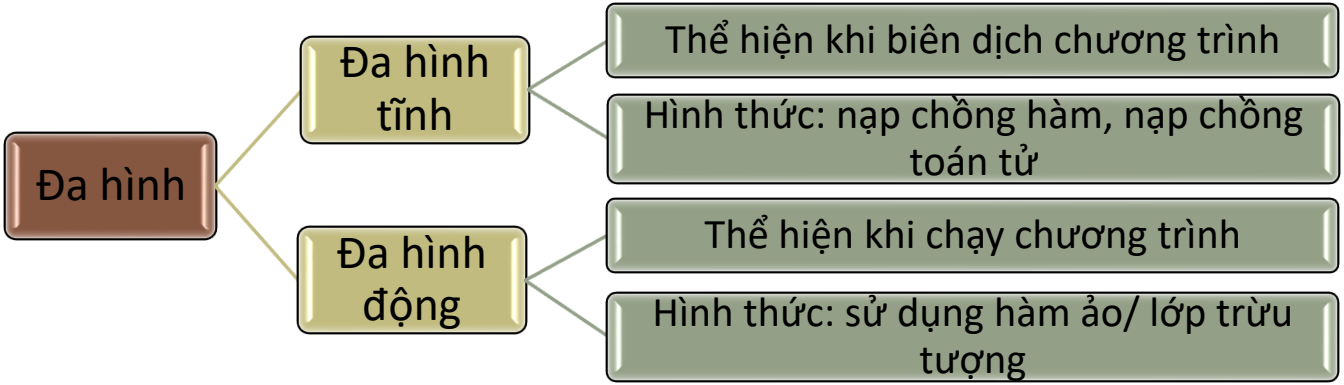
- Trong lớp con gọi hàm tạo của lớp cha:
  - Sử dụng từ khóa **base**
  - Ví dụ:
    - Lớp CONNGUOI có các thông tin: họ tên, tuổi, quê quán, giới tính, … và hàm tạo có tham số
    - Lớp CANBO kế thừa từ lớp CONNGUOI và có thêm thông tin: hệ số lương, thâm niên công tác, … => hàm tạo đầy đủ tham số cần gọi lại hàm tạo của lớp cha

Kế thừa


Lớp cha
Lớp con
Hàm tạo có tham số của lớp cha
Hàm tạo có tham số của lớp con
Gọi hàm tạo của lớp cha
class CONNGUOI
{
protected string hoten, gioitinh, quequan;
protected int tuoi;
public CONNGUOI(string ht, string gt, string qq, int t)
{
hoten = ht;
gioitinh = gt;
quequan = qq;
tuoi = t;
}
class CANBO : CONNGUOI
{
private double hesoluong;
private int thamnien;
public CANBO(string ht, string gt, string qq, int t, double hsl, int tn) : base(ht, gt, qq, t)
{
hesoluong = hsl;
thamnien = tn;
}
}

Đa hình

- Là các hình thái thực hiện khác nhau của một phương thức
- Được phân làm 2 loại:



Đa hình tĩnh

- Sử dụng kỹ thuật nạp chồng hàm
- Là cách tạo ra những hàm:
  - Giống nhau về tên hoặc
  - Giống nhau về cả tên và kiểu trả về
  - Nhưng phải khác nhau về kiểu tham số hoặc
  - Khác nhau về số lượng tham số

Đa hình tĩnh

- Ví dụ:

2 hàm tạo có cùng tên, khác nhau số lượng tham số

```
class PHANSO
{
    int tuso, mauso;
    public PHANSO()
    {
    }
    public PHANSO(int tu, int mau)
    {
        tuso = tu;
        mauso = mau;
    }
}
```

Đa hình tĩnh

- Sử dụng kỹ thuật nạp chồng toán tử (đã học)
- Là cách sử dụng các toán tử +, - , *, /, % cho những kiểu dữ liệu người dùng định nghĩa
- Cú pháp:

public static <kiểu trả về> operator <toán tử> (danh sách tham số)

Ví dụ

```
public static Phanso operator +(Phanso a, Phanso b)
{
    Phanso kq = new Phanso();
    kq.tuso = a.tuso * b.mauso + b.tuso * a.mauso;
    kq.mauso = a.mauso * b.mauso;
    return kq;
}
```

Đa hình động

- Là các hình thức thực hiện một phương thức trên các đối tượng khác nhau
- Ví dụ:

```
CONNGUOI[] CN = new CONNGUOI[3];
CN[0] = new CONNGUOI("Hoa", "Nu", "HN", 15);
CN[1] = new CANBO("Hong", "Nu", "HP", 10, 3.0, 10);
CN[2] = new SINHVIEN("Hoang", "Nam", "TN", 20, 12345, 9.0);
CN[0].xuat();
CN[1].xuat();
CN[2].xuat();
```

Đa hình động

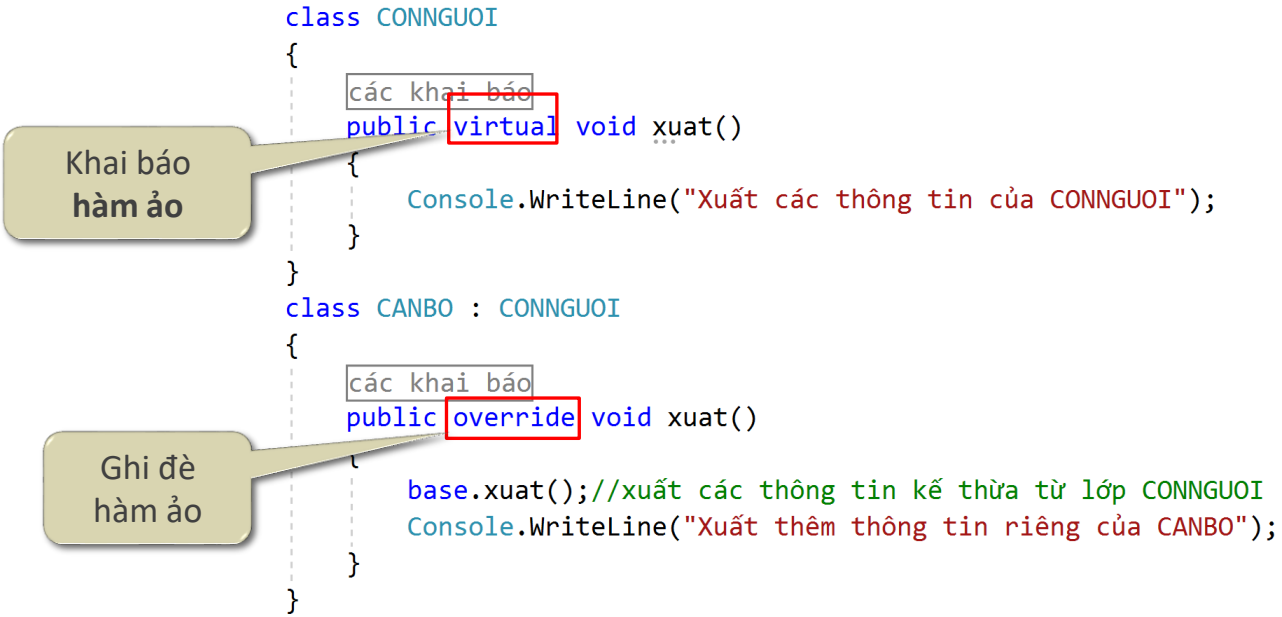
```
class CONNGUOI
{
    các khai báo
    public void xuat()
    {
        Console.WriteLine("Xuất các thông tin của CONNGUOI");
    }
}
class CANBO : CONNGUOI
{
    các khai báo
    public void xuat()
    {
        base.xuat();//xuất các thông tin kế thừa từ lớp CONNGUOI
        Console.WriteLine("Xuất thêm thông tin riêng của CANBO");
    }
}
```

```
Xuất các thông tin của CONNGUOI
Xuất các thông tin của CONNGUOI
Xuất các thông tin của CONNGUOI
```

Đa hình động

- Để thể hiện được tính đa hình động cần:
  - Khai báo **hàm ảo** ở lớp cha (**virtual**)
  - Ghi đè hàm đó ở lớp con (**override**)

Ví dụ về đa hình động




```
class CONNGUOI
{
    các khai báo
    public virtual void xuat()
    {
        Console.WriteLine("Xuất các thông tin của CONNGUOI");
    }
}
class CANBO : CONNGUOI
{
    các khai báo
    public override void xuat()
    {
        base.xuat();//xuất các thông tin kế thừa từ lớp CONNGUOI
        Console.WriteLine("Xuất thêm thông tin riêng của CANBO");
    }
}
```

Lớp trừu tượng

- Là lớp cơ sở cung cấp một phương thức giống nhau cho nhiều lớp dẫn xuất
- Phương thức chung này phải được khai báo là một phương thức trừu tượng
- Cần phải định nghĩa rõ các phương thức trừu tượng ở lớp dẫn xuất

Lớp trừu tượng

- Cú pháp khai báo lớp trừu tượng:

  [quyền truy cập] **abstract class** <tên lớp>

- Cú pháp khai báo hàm trừu tượng:

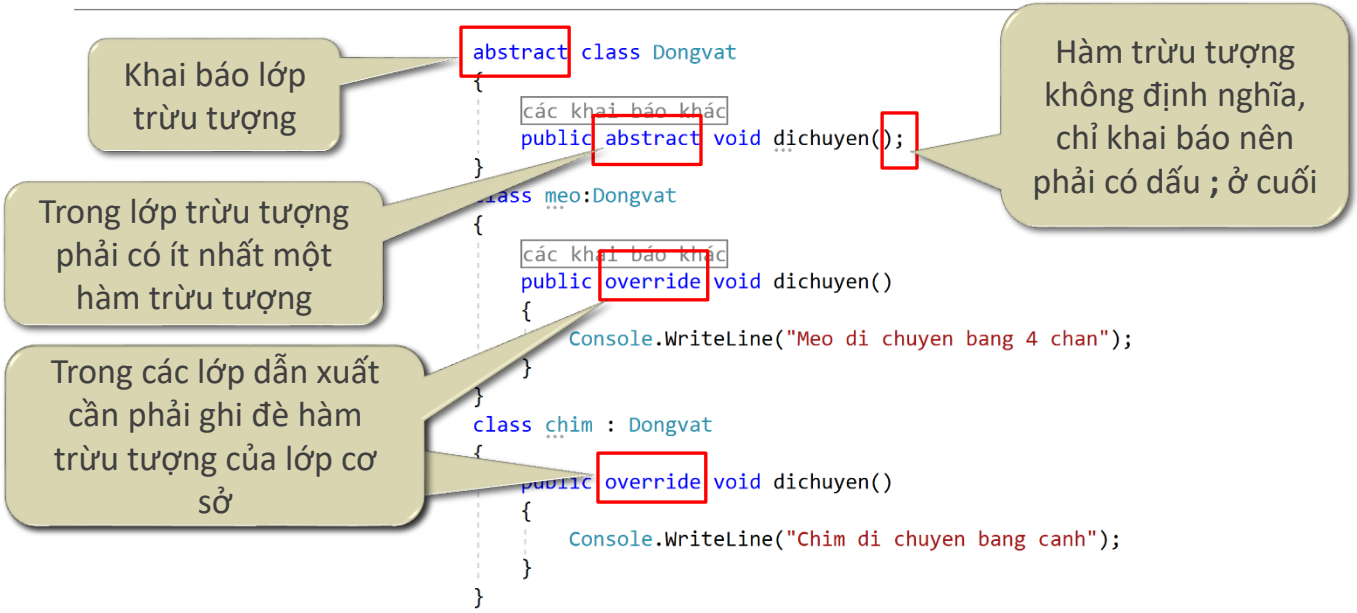
  [quyền truy cập] **abstract** <kiểu trả về> <tên phương thức> (ds tham số);

Lớp trừu tượng

- Ví dụ:
  - Lớp động vật có phương thức di chuyển, ăn
  - Lớp mèo có phương thức di chuyển bằng 4 chân
  - Lớp chim có phương thức di chuyển bằng cánh
  - Lớp mèo ăn thịt cá
  - Lớp chim ăn hoa quả

    => phương thức di chuyển và phương thức ăn của lớp động vật là một phương thức trừu tượng, chưa rõ ràng

Lớp trừu tượng



Bài tập

- Tạo lớp phương tiện giao thông và các lớp dẫn xuất: ô tô, xe máy, tàu thủy, máy bay,...
- Định nghĩa phương thức di chuyển phù hợp với từng loại
- Trong chương trình chính, tạo một phương tiện giao thông. Khi người dùng lựa chọn loại phương tiện nào thì gọi phương thức di chuyển của phương tiện đó

Giao diện

- Giao diện là một dạng của lớp trừu tượng
- Chỉ nguyên mẫu phương thức, thuộc tính, chỉ mục được khai báo trong giao diện.
- Tất cả các thành phần khai báo trong giao diện mặc định là public (nên không có từ khóa về mức độ truy cập trong các khai báo thuộc tính và phương thức)
- Khi một lớp kế thừa một giao diện ta nói rằng lớp đó thực thi (implement) giao diện
- C# cho phép đa kế thừa giao diện

Giao diện

- Cú pháp:

  [quyền truy cập] interface <tên giao diện> [: giao diện cơ sở]

  {

  //nội dung giao diện

  }

  Trong đó:

  - [quyền truy cập] : thường là public;
  - [: giao diện cơ sở] : danh sách các interface khác mà nó kế thừa, mỗi interface cách nhau bởi dấu phẩy (,)

Ví dụ

```
public interface IPlayerManager
{
    public void PlayMusic();
    public void PauseMusic();
    public void Stop();
    public int OnOff // thuộc tính của Interface
    {
        get;
    }
} // Error ?
```

Ví dụ

```
public interface IPlayerManager
{
    void PlayMusic();
    void PauseMusic();
    void Stop();
    int OnOff // thuộc tính của Interface
    {
        get;
    }
}
```

Ví dụ

```
public class Player : IPlayerManager
{   int _Switch;
    public void PlayMusic()
    { Console.WriteLine("Music is playing ");
      _Switch = 1 ;
    }
    public void PauseMusic()
    { Console.WriteLine("Music is pause ");
      _Switch = 0 ;
    }
    public void Stop()
    { Console.WriteLine("Music is stopped ");
      _Switch = -1 ;
    }
} // Error ?
```

Ví dụ

```
public class Player : IPlayerManager
{   int _Switch;
    public void PlayMusic()
    { Console.WriteLine("Music is playing ");
      _Switch = 1 ;
    }
    public void PauseMusic()
    { Console.WriteLine("Music is pause ");
      _Switch = 0 ;
    }
    public void Stop()
    { Console.WriteLine("Music is stopped ");
      _Switch = -1 ;
    }
    public int OnOff
    {
        get { return _Switch; }
    }
}
```

Bài tập

- Đoạn mã nguồn sau đây có lỗi, hãy sửa lỗi và cho biết tại sao có lỗi này. Sau khi sửa lỗi hãy viết một lớp Rectangle thực thi giao diện này?

```
public interface IDimensions
{
   long width;
   long height;
   double Area();   //diện tích
   double Circumference();   //chu vi
   int Sides();   //số cạnh
}
```