



TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa CNTT – Bộ môn CNPM

LẬP TRÌNH NÂNG CAO



Giang viên: Lý Anh Tuấn
Email: tuanla@wru.vn



OPP TRONG C#



Nội dung

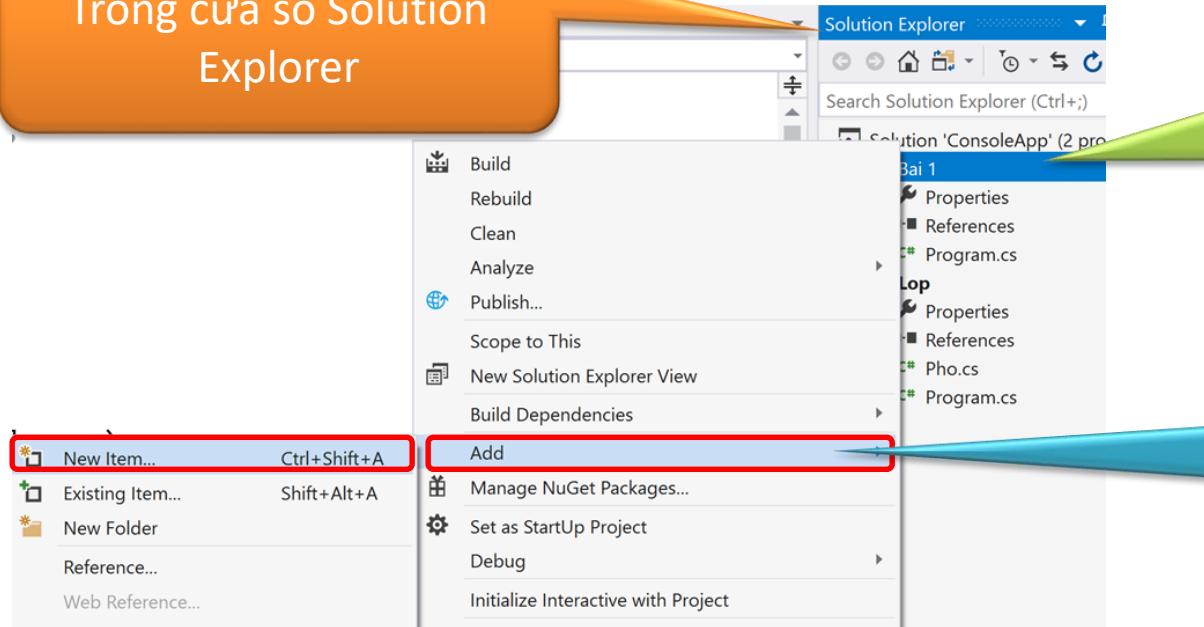
- Khai báo lớp
- Định nghĩa lớp
- Sử dụng lớp





Khai báo lớp

Trong cửa sổ Solution Explorer



Bấm chuột phải vào tên project

Chọn Add/New Item





Khai báo lớp

The screenshot shows the 'Add New Item - Bai 1' dialog in Visual Studio. The 'Visual C# Items' category is selected. A red box highlights the 'Class' item, which is also pointed to by a green callout bubble labeled 'Chọn Class'. Another red box highlights the 'Name:' input field where 'Class1.cs' is typed, which is also pointed to by a blue callout bubble labeled 'Đặt tên lớp'. The 'Add' button at the bottom right is highlighted with a blue callout bubble labeled 'Ấn nút Add'.

Chọn Visual C# Items

Chọn Class

Đặt tên lớp

Ấn nút Add

Add New Item - Bai 1

Installed

Visual C# Items

Sort by: Default

Search (Ctrl+E)

Type: Visual C# Items

An empty class is created.

| Item | Type |
|--------------------------------|-----------------|
| Class | Visual C# Items |
| Interface | Visual C# Items |
| Windows Form | Visual C# Items |
| User Control | Visual C# Items |
| Component Class | Visual C# Items |
| User Control (WPF) | Visual C# Items |
| About Box | Visual C# Items |
| ADO.NET Entity Data Model | Visual C# Items |
| Application Configuration File | Visual C# Items |
| Application Manifest File | Visual C# Items |
| Assembly Information File | Visual C# Items |

Name: Class1.cs

Add Cancel

OPP trong C#

5



- 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 lớp và có thể được dùng trong những lớp khác.) |
| private | Che dấu. Những thành viên được đánh dấu là private thì chỉ được sử dụng trong các phương thức của lớp. |
| protected | Thành phần nào được đánh dấu là protected trong lớp X thì chỉ được dùng trong lớp X và các lớp dẫn xuất từ X |
| internal | Độ truy cập nội bộ, bị giới hạn trong một Assembly |
| partial | Được dùng khi lớp được khai báo và định nghĩa trên nhiều file |





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
 - <ten 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 khởi tạo của lớp>;
- Ví dụ:
ps1 = **new** Phanso(); //khởi tạo ps1 bằng hàm khởi tạo mặc định
ps2 = **new** Phanso(1,2); //khởi tạo ps2 bằng hàm khởi tạo có tham số)





Sử dụng lớp

- Truy cập tới dữ liệu và hàm thành phần 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("Nhập phân số A: ");
        A.nhap();
        Console.WriteLine("Nhập phân số B: ");
        B.nhap();
        C = A + B;
        Console.WriteLine("\nTổng 2 phân số là: ");
        A.xuat(); Console.Write(" + "); B.xuat(); Console.Write(" = "); C.xuat();
        Console.ReadLine();
    }
}
```





Thuộc tính

- 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

- 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)

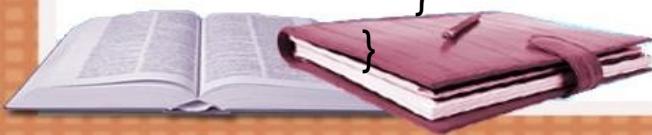




Ví dụ

```
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 đã tính dưới dạng rút gọn





LẬP TRÌNH NÂNG CAO OPP TRONG C# (Tiếp)



Giang 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)**





Kế thừa

- Cú pháp khai báo
class <ten lớp dẫn xuất> : <ten 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 phương thức 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 khởi tạo có truyền vào các tham số
 - Lớp CANBO kế thừa từ lớp CONNGUOI và có thêm thông tin: hệ số lương, thậm niên công tác, ... => hàm khởi tạo truyền đầy đủ các tham số sẽ cần gọi lại hàm khởi tạo của lớp cha





Kế thừa

Lớp cha

Lớp con

```
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;
    }
}
```

Hàm khởi tạo
có tham số
của lớp cha

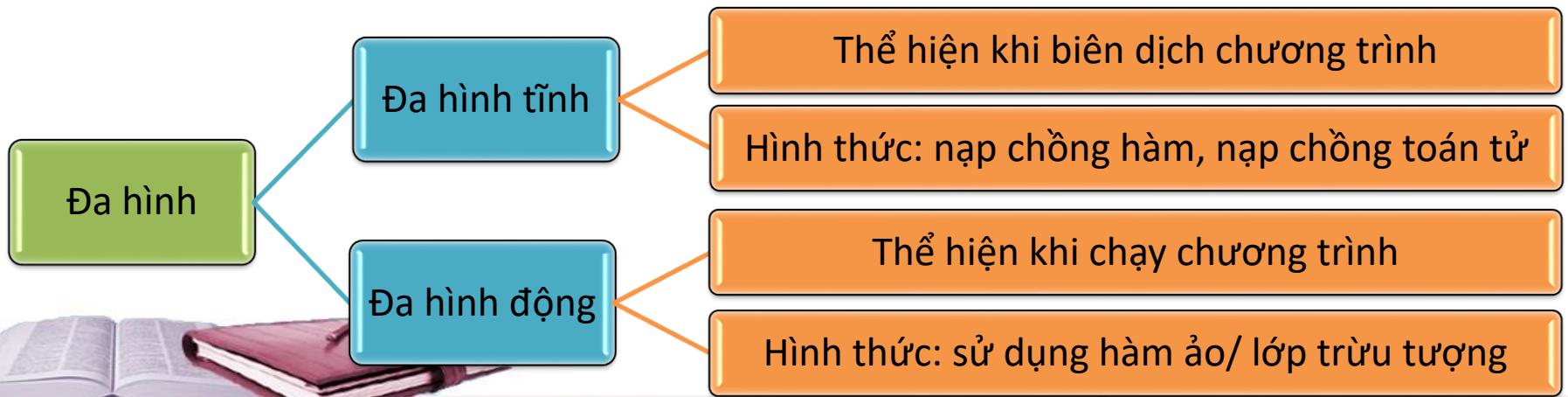
Hàm khởi tạo
có tham số
của lớp con

Gọi hàm khởi tạo
của lớp cha



Đa hình

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





Đa hình tinh

- 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 các tham số





Đa hình tĩnh

- Ví dụ:

2 hàm tạo có cùng tên, khác nhau số tham số truyền vào

```
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 tự 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();
```





1959 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

Khai báo hàm ảo

```
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");
    }
}
```

Ghi đè hàm ảo



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

Khai báo lớp trừu tượng

Trong lớp trừu tượng
phải có ít nhất một
hàm trừu tượng

Trong các lớp dẫn xuất
cần phải ghi đè hàm trừu
tượng của lớp cơ sở

```
abstract class Dongvat
{
    // các khai báo khác
    public abstract void dichuyen();
}

class meo : Dongvat
{
    // các khai báo khác
    public override void dichuyen()
    {
        Console.WriteLine("Meo di chuyen bang 4 chan");
    }
}

class chim : Dongvat
{
    public override void dichuyen()
    {
        Console.WriteLine("Chim di chuyen bang canh");
    }
}
```

Hàm trừu tượng
không định nghĩa,
chỉ khai báo nên
phải có dấu ; ở cuối





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 có các 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 khai báo các thuộc tính và phương thức)
- Khi một lớp kế thừa một giao diện ta nói rằng lớp đó thực thi (Implement) 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 các 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
```

```
}
```

