Cho người mới

Bài 22-1

C Sharp

Dễ Hiểu - Dễ học

1. Khái quát OOP C#
2. Khai báo lớp
3. Constructor
4. Properties

Hướng đối tượng C#

Part 1

1 Khái quát OOP C#

❑ 1. Khái quát lớp và đối tượng :

✓ *Đối tượng (object) trong lập trình hướng đối tượng giống như 1 đối tượng cụ thể trong thế giới thực*

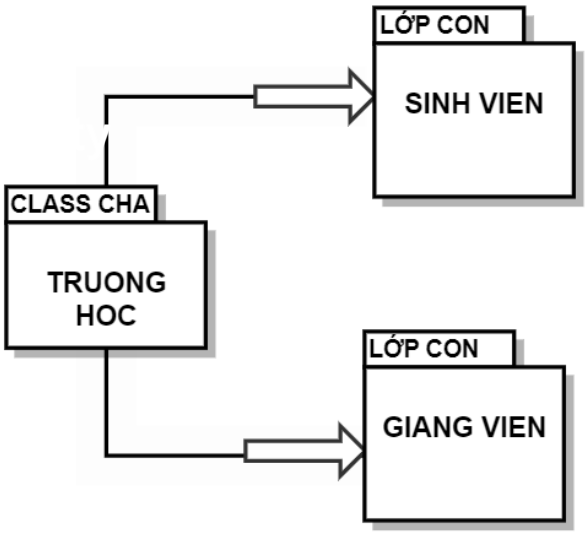
Mỗi đối tượng có thuộc tính và hành vi riêng

+ Thuộc tính : Đặc điểm của đối tượng

+ Phương thức: Hành vi của đối tượng

+ 1 con chó tên Luccy : là 1 đối tượng cụ thể

✓ *Các đối tượng có các phương thức, thuộc tính giống nhau được gom thành 1 lớp để dễ quản lý*



Thuộc tính (attribute)			
TÊN	CCCD	NĂM SINH	QUÊ QUÁN

Phương Thức (Method)			
HỌC	Chơi Game	Tính DTB	ĐKy Học Phần

Thuộc tính (attribute)			
TÊN	CCCD	NĂM SINH	QUÊ QUÁN

Phương Thức (Method)			
DẠY	XEM TIKTOK	CHẤM ĐIỂM	SỬA ĐIỂM

2 Khai báo lớp

❑ 2. Quy tắc đặt tên lớp:

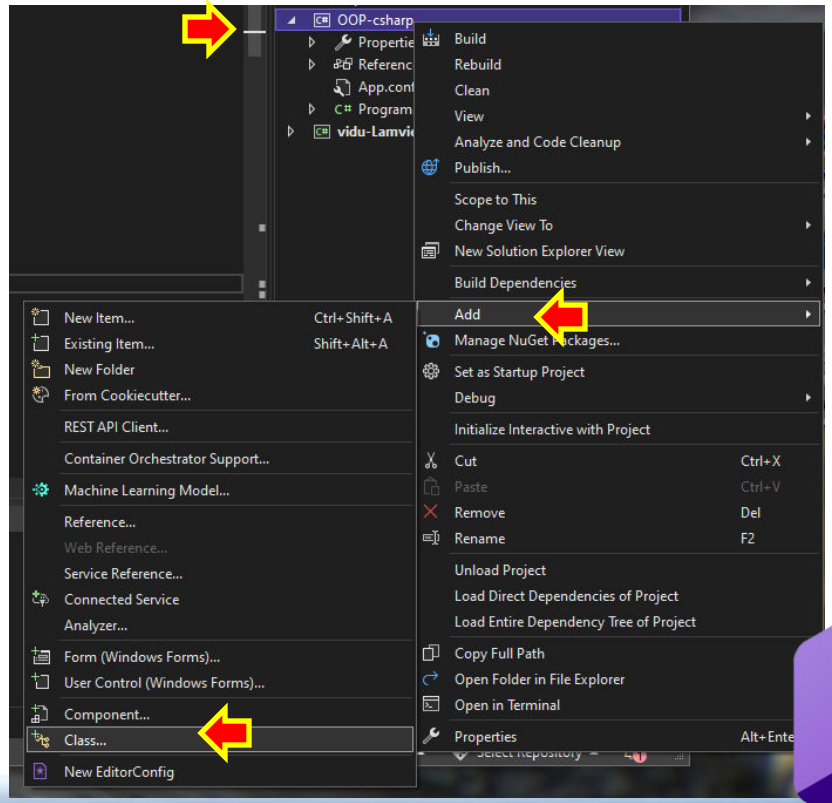
1. Tên lớp nên là 1 danh từ hoặc 1 cụm DT , nên viết hoa ký tự đầu tiên (Car, Bird, Buom, SinhVien)
2. Không được bắt đầu bằng số, không bắt đầu bằng ký tự đặc biệt, không trùng với keyword trong c#

❑ 3. Tạo class :

Cấu trúc chung class

```
public class SinhVien
  {
      // khai báo biến lớp (thuộc tính)
      kieubien tenBien1;
      kieubien tenBien2;

      //khai báo phương thức
      PhuongThuc1();
      PhuongThuc1();
  }
```



2 Khai báo lớp

❑4. Tạo 1 đối tượng mới : TenLop tenDoiTuong = new TenLop()

```
//khởi tạo 1 đối tượng mới
SinhVien sinhVien1 = new SinhVien();
SinhVien sinhVien2 = new SinhVien();
```

❑5. Biến lớp :

1. *Quy tắc khai báo* ***giống khai báo biến thông thường (không bắt đầu bằng số, ký tự đặc biệt, không trùng keyword c#, quy tắc camel*** *(maSV , tenSV ,tenBien, canhHuyen,)*
2. *Mức độ truy xuất :*
   - ✓ public : Truy xuất dc mọi nơi
   - ✓ private : Truy xuất trong class
   - ✓ protecteted: Chỉ truy xuất ở trong class hoặc class kế thừa

Chú ý : Quy tắc khai báo thuộc tính, nếu chỉ truy xuất trong class thì dùng private
Nếu dùng cần gọi lại trong class con thì khai báo dùng protecteted

☺ Dùng public vẫn chạy được chương trình, không báo lỗi, nhưng quy định là quy định , không nên =)) ☺

```
public class SinhVien
{
    private int maSV;
    private string tenSV;
}
```

3 Constructor

❑6. Constructor (Hàm tạo):

1. Constructor : là hàm dùng để tự động khởi tạo giá trị cho đối tượng, khi đối tượng được sinh ra.

2. Tên giống với tên lớp

3. Cú pháp

```
public class SinhVien
```

```
// Khai báo constructor
//(Gán giá trị mặc định cho đối tượng)
0 references
public SinhVien()
{
    this.maSV = 0;
    this.tenSV = "No name";
}
```

```
// Khai báo constructor
//(Khởi tạo giá trị cho đối tượng do người dùng truyền vào)
0 references
public SinhVien(int maSV, string tenSV)
{
    this.maSV=maSV;
    this.tenSV=tenSV;
}
```

4 Properties

❑7. Properties :

Do chú ý ở mục 5 :

Chú ý : *Quy tắc khai báo thuộc tính, nếu chỉ truy xuất trong class thì dùng private*
Nếu dùng cần gọi lại trong class con thì khai báo dùng protecteted

=> Không được phép truy xuất các thuộc tính từ bên ngoài

=> Properties giúp ta có thể truy xuất xem, sửa đổi dữ liệu

**** Quy tắc đặt tên : Viết hoa ký tự đầu*

```
//Khai báo Properties để có thể truy xuất sửa đổi dữ liệu:
0 references
public string TenSV
{
    get { return tenSV; } // //get giá trị để đọc
    set { tenSV = value; } // set giá trị
}

0 references
public int MaSV
{
    get { return maSV; }
    set { maSV = value; }
}
```

❑8. Nhập, xuất thông tin đối tượng

```
//1. Tạo 1 đối tượng mới không truyền giá trị
SinhVien sinhVien1 = new SinhVien();
// xuất thông tin sv1
Console.WriteLine(sinhVien1.MaSV);
Console.WriteLine(sinhVien1.TenSV);
```

```
0
No name
```

```
// Khai báo constructor
//(Gán giá trị mặc định cho đối tượng)
0 references
public SinhVien()
{
    this.maSV = 0;
    this.tenSV = "No name";
}
```

```
//2. Tạo 1 đối tượng có truyền vào giá trị ban đầu
SinhVien sinhVien2 = new SinhVien(2,"obama");
Console.WriteLine(sinhVien2.MaSV);
Console.WriteLine(sinhVien2.TenSV);
```

```
2
obama
```

```
// Khai báo constructor
//(Khởi tạo giá trị cho đối tượng
//do người dùng truyền vào)
1 reference
public SinhVien(int maSV, string tenSV)
{
    this.maSV = maSV;
    this.tenSV = tenSV;
}
```

❑. Sửa dữ liệu đối tượng

```
//3. sửa dữ liệu đối tượng
sinhVien2.TenSV = "jacky chan";
Console.WriteLine(sinhVien2.TenSV);
```

```
jacky chan
999
```

Cho người mới

Bài 22-2

1. Method C#
2. Service Method & Support Method
3. Overloading Method
4. Parametter List Method

Hướng đối tượng C#
Part 2

1 Method C#

❑9. Phương thức :

1. Phương thức vốn đã rất quen thuộc với các thí chủ ☺☺

```
char kyTu = 'a';
//phương thức xây dựng sẵn C#
// Chuyển char sang viết hoa
Console.WriteLine(char.ToUpper(kyTu));
//kiểm tra xem ký tự có phải là chữ số?
Console.WriteLine(char.IsDigit(kyTu));
```

```
A
False
```

2. Bản chất phương thức trong lập trình hướng đối tượng là các hàm bên trong lớp (hay nói cách khác nó là các khối lệnh thực hiện 1 công việc hoàn chỉnh)

Ví dụ :
1. *Tính điểm Trung bình môn của Sinh viên*
2. *Tính lương cho giảng viên*
3. *Tính học phí miễn giảm cho sinh viên vùng khó khăn*
4. *Tính học bổng cho nhân viên theo ngành*

3 . Truy xuất phương thức: *tenDoiTuong.TenPhuongThuc();*

1 Method C#

❑10. Tostring :

Dùng phương thức Tostring để xuất đoạn văn bản mong muốn

Class SinhVien

```
//Phương thức Tostring
2 references
public override string ToString()
{
    return this.MaSV + "\t" + this.tenSV;
}
```

Program.cs

```
Console.WriteLine(sinhVien1.ToString());
Console.WriteLine(sinhVien1); // viết gọn
Console.WriteLine(sinhVien2.ToString());
Console.WriteLine(sinhVien2); // viết gọn
```

F5

```
0       No name
0       No name
999     jacky chan
999     jacky chan
```

❑11 . region - endregion : Gom nhóm làm gọn

```
8 references
public class SinhVien
{
    các biến lớp
    Khai báo constructor
    khai báo các Properties
    các phương thức
}
```

2 Service Method & Support Method

❑12. Support Method & Service Method :

*** Một lớp có nhiều phương thức, có những phương thức public ra ngoài(public, hay service method)**

*** Còn những phương thức chỉ sử dụng trong lớp (private, gọi là support method)**

*1. Support Method dùng để hỗ trợ bên trong phương thức. Không truy xuất được từ bên ngoài => Dùng từ khóa **private***

*2. Service Method Truy xuất được từ bên ngoài => Dùng từ khóa **public***

```
// support method
//kiểm tra điều kiện nhập sinh viên mới
1 reference
private bool CheckDiemthiDH()
{
    //bắt buộc >=21 điểm
    return (this.DiemThiDH-21 >=0);
}
//service method (xuất thông tin)
1 reference
public void XuatThongTin()
{
    if (CheckDiemthiDH() == false)
        Console.WriteLine(" Điểm thi DH <21, KT lại sinh viên này");
    else
        Console.WriteLine(ToString());
}
```

```
//Service Method & Support MEthod
SinhVien sinhVien3  = new SinhVien();
sinhVien3.MaSV = 3;
sinhVien3.TenSV = "Lục văn Ba";
sinhVien3.DiemThiDH = 20;
//check điểm bằng Service Method
sinhVien3.XuatThongTin();
//Support MEthod không hiển thị do để private
sinhVien3.CheckDiemthiDH();
```

3 Overloading Method

❑13. Overloading Method :

1. Signature gọi là khác nhau nếu chúng khác nhau về

1. Số lượng các đối số
2. Kiểu dữ liệu các đối số
3. Thứ tự các đối số

2. Overloading Method : Trong cùng class có nhiều phương thức cùng tên nhưng khác nhau về *Signature*

3. Constructor : là trường hợp đặc biệt của *Overloading Method*

```
public SinhVien(int maSV, string tenSV,float tbDiemThiDH)
{
    this.maSV = maSV;
    this.tenSV = tenSV;
    this.tbDiemThiDH = tbDiemThiDH;
}
```

Signature

```
public SinhVien()
{
    this.maSV = 0;
    this.tenSV = "No name";
    this.tbDiemThiDH = 0;
}
```

Signature

```
//Đảo vị trí các đối số
1 reference
public SinhVien(int maSV, float tbDiemThiDH, string tenSV)
{
    this.maSV = maSV;
    this.tenSV = tenSV;
    this.tbDiemThiDH = tbDiemThiDH;
}
```

Signature

4 Parametter List Method

❑14. Parametter List Method :

1. Trong trường hợp ta không thể nắm được số lượng đối số vào, hoặc số lượng đối số quá lớn . C# cung cấp thêm lựa chọn Parametter List

```
//Parametter List Method
//tính tổng điểm TB kết thúc học kỳ
0 references
public float TBKetThucHocKy(params float[] mang)
{
    float s = 0;
    foreach (float i in mang)
    {
        s += i;
    }
    return (s / mang.Count());
}
```

```
//Parametter List Method
Console.WriteLine(sinhVien2.TBKetThucHocKy(7,8,9,4,7,5,8,7,5));
Console.WriteLine(sinhVien3.TBKetThucHocKy(9,7,8,5,4,1,3));
```

```
6.666667
5.285714
```

❑15. Auto-Implemented Properties

Thuộc tính được triển khai tự động

Class HocSinh

```
//Auto-Implemented Properties
4 references
public string Name { get; set; }
3 references
public string Email { get; set; }
3 references
public string Phone { get; set; }
// Phương thức Tostring
0 references
public override string ToString()
{
    return this.Name+"\t" +this.Email+"\t" + this.Phone;
}
```

Program.cs

```
//Auto-Implemented Properties
HocSinh hocSinh1 = new HocSinh();
hocSinh1.Name = "Bé Na";
hocSinh1.Phone = "09xx";
hocSinh1.Email = "bena@gmail.com";
Console.WriteLine(hocSinh1.Name);
Console.WriteLine(hocSinh1);
```

F5

```
Bé Na
Bé Na   bena@gmail.com  09xx
```

Cho người mới

Bài 22-3

1. Kế thừa – inheritance C#
2. Kế thừa thuộc tính
3. Kế thừa phương thức
4. Tính Đa Hình

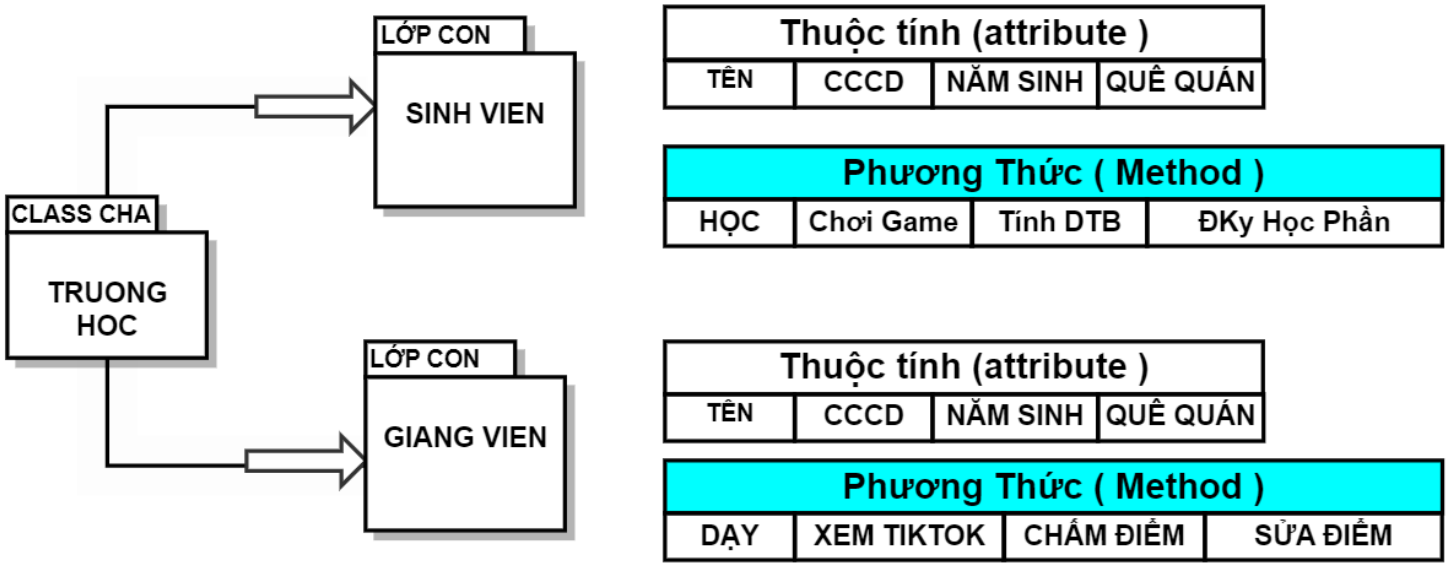
Hướng đối tượng C# Part 3

1 Kế thừa C#

❑ 1. Khái quát kế thừa:

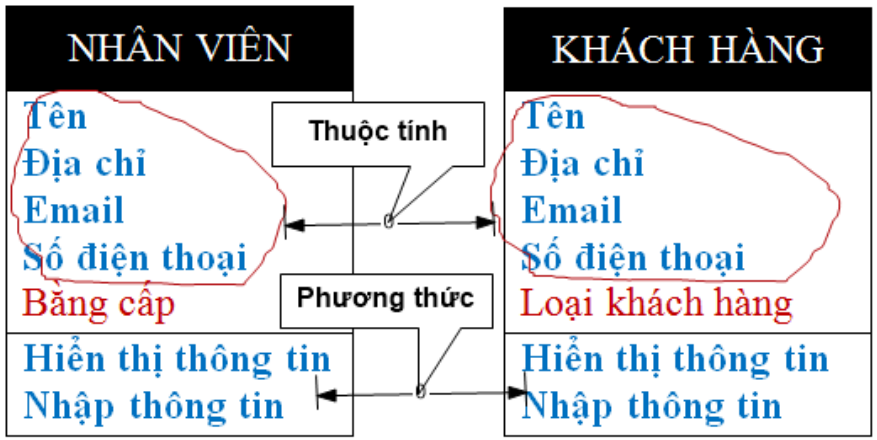
✓ *Kế thừa : Tạo ra các lớp con, để tái sử dụng lại những thành phần của lớp cha đã có*

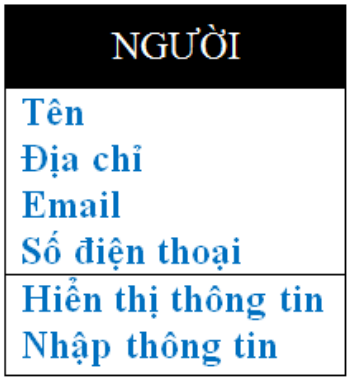
✓ *Ưu điểm : Giúp code ngắn gọn, không cần phải viết lại những code mà lớp cha đã có => thuận tiện trong quản lý, dễ sửa đổi theo từng khối*

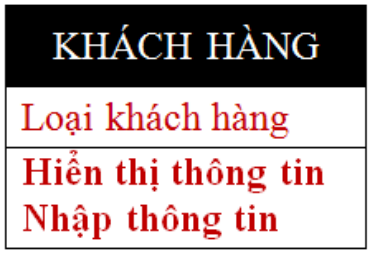


1 Kế thừa C#

❑ Ví dụ :







1 Kế thừa C#

❑ **2. Cú pháp : *class* <Tên lớp con> : <Lớp Cha>**

Ex1: ***class Toan: SinhVien***

class Cntt: SinhVien

class NhanVien: CongTy

class NhanVienHanhChinh: NhanVien

class NhanVienDiCa: NhanVien

Note : mọi thông tin để public hoặc protected từ lớp cha, lớp con sẽ được thừa hưởng

```
8 references
public class SinhVien
{
    các biến lớp
    constructor
    các Properties
    các phương thức
}
```

Class cha

```
0 references
public class Toan: SinhVien
{
}
```

Class con

```
0 references
public class Cntt: SinhVien
{
}
```

Class con

```
4 references
public class NhanVien
{
    2 references
    public int MaNV { get; set; }
    2 references
    public string TenNV { get; set; }
}
```

Class cha

```
2 references
public class NhanVienHanhChinh: NhanVien
{
}
```

Class con

2 Kế thừa thuộc tính

❑ 4. Kế thừa thuộc tính :

TH 1. Kế thừa trực tiếp. Class con không cần khai báo lại

Class NhanVien

```
4 references
public class NhanVien
{
    #region khởi tạo
    5 references
    public int MaNV { get; set; }
    5 references
    public string TenNV { get; set; }
    #endregion
```

Program.cs

```
NhanVienDiCa ca1=new NhanVienDiCa();
ca1.MaNV = 1;
ca1.TenNV = "Em đi ca";
Console.WriteLine(ca1.MaNV + "   " + ca1.TenNV);
```

F5

```
1   Em đi ca
```

Class con không cần khai báo vẫn gọi dc MaNV và TenNV

TH 2. Kế thừa và bổ sung các thuộc tính mới

Class NhanVienDiCa

```
//khai báo thêm các thuộc tính ca ngày, ca đêm
1 reference
public string Ca { get; set; }
```

Program.cs

```
NhanVienDiCa ca1=new NhanVienDiCa();
ca1.MaNV = 1;
ca1.TenNV = "Em đi ca";
ca1.Ca = "Ca ngày";
Console.WriteLine(ca1.MaNV + "   " + ca1.TenNV);
Console.WriteLine("Ca hiện tại: "+ ca1.Ca);
```

F5

```
1   Em đi ca
Ca hiện tại: Ca ngày
```

3 Kế thừa phương thức

❑ 4. Kế thừa phương thức :

TH 1. Kế thừa Method trực tiếp. Class con không cần khai báo lại

Class NhanVien

```
// phương thức tính lương NV
4 references
public double TinhLuong()
{
    return 1000;
    // lương cơ bản 1000$ tháng
}
```

Program.cs

```
NhanVien nhanVien1  = new NhanVien();
Console.WriteLine("lương nv 1 : " + nhanVien1.TinhLuong());

NhanVienHanhChinh hc1 = new NhanVienHanhChinh();
Console.WriteLine("lương hc là: "+ hc1.TinhLuong());
Console.ReadKey();
```

F5

```
luong nv 1 : 1000
luong hc là: 1000
```

TH 2. Kế thừa và tái định nghĩa phương thức (Overriding Methods)

*** Định nghĩa lại phương thức class cha: dùng keyword new*

*** Gọi lại phương thức class cha: dùng keyword base*

Class NhanVienDiCa

```
1 reference
public new double TinhLuong()
{
    return base.TinhLuong()*1.05;

}
```

Program.cs

```
NhanVienDiCa ca1=new NhanVienDiCa();
Console.WriteLine("lương ca1 là" +ca1.TinhLuong());
Console.ReadKey();
```

F5

```
luong nv 1 : 1000
luong hc là: 1000
luong ca1 là: 1050
```

❑ 4. Kế thừa phương thức :

TH 2. Kế thừa và tái định nghĩa phương thức (Overriding Methods)

***** Trong các lớp class cha và con , có các phương thức cùng tên, cùng Signature nhưng khác nhau về nội dung***

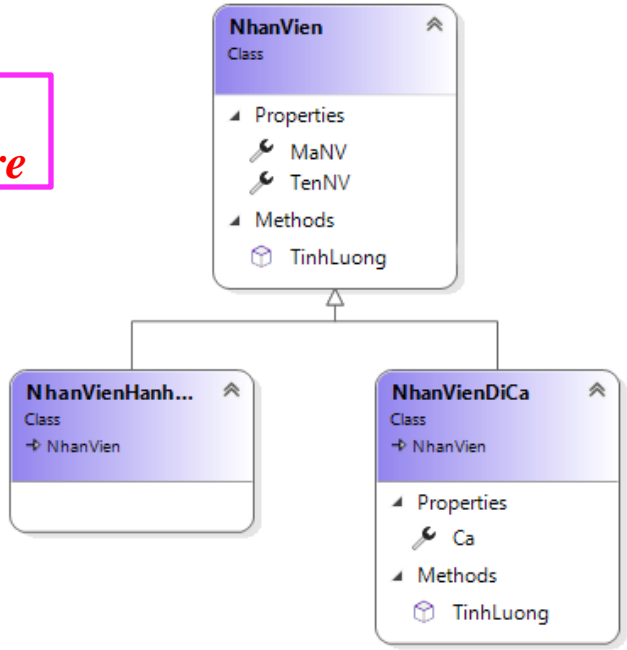
Cùng tên – cùng Signature

Class NhanVien

```
// phương thức tính lương NV
4 references
public double TinhLuong()
{
    return 1000;
    // lương cơ bản 1000$ tháng
}
```

Class NhanVienDiCa

```
1 reference
public new double TinhLuong()
{
    return base.TinhLuong()*1.05;
}
```




```
NhanVien nhanVien1  = new NhanVien();
Console.WriteLine("lương nv 1 : " + nhanVien1.TinhLuong());

NhanVienHanhChinh hc1 = new NhanVienHanhChinh();
Console.WriteLine("lương hc là: "+ hc1.TinhLuong());
Console.ReadKey();

NhanVienDiCa ca1=new NhanVienDiCa();
Console.WriteLine("lương ca1 là" +ca1.TinhLuong());
Console.ReadKey();
```

6. Override : Nạp chồng

✓ *virtual* : *Khai báo ở lớp cha, cho biết thành phần đó có thể nạp chồng*

Ví dụ : Tất cả nhân viên, trừ nhân viên hành chính (sẽ được thưởng nếu đi làm đủ 26 ngày, thưởng 100$)

Class NhanVien

```
public virtual double ThuongDuCong(int ngayCong)
{
    if (ngayCong == 26)
        return 100;
    else
        return 0;
}
```

✓ *override* : *Khai báo ở lớp con, để sử dụng lại, và ghi đè thành phần ở lớp cha*

```
public class NhanVienHanhChinh: NhanVien
{
    1 reference
    public override double ThuongDuCong(int ngayCong)
    {
        return 0;
    }
}
```

❑ 7. Tính Đa hình :

Những phương thức ghi đè kiểu như trên => thể hiện tính Đa hình .