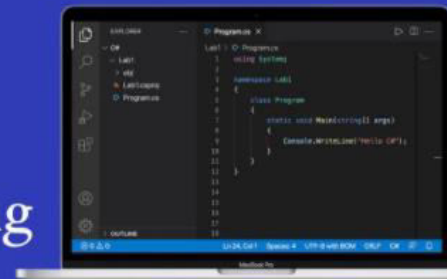




Hướng đối tượng C# Part 1

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

Object
Oriented
Programming



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*

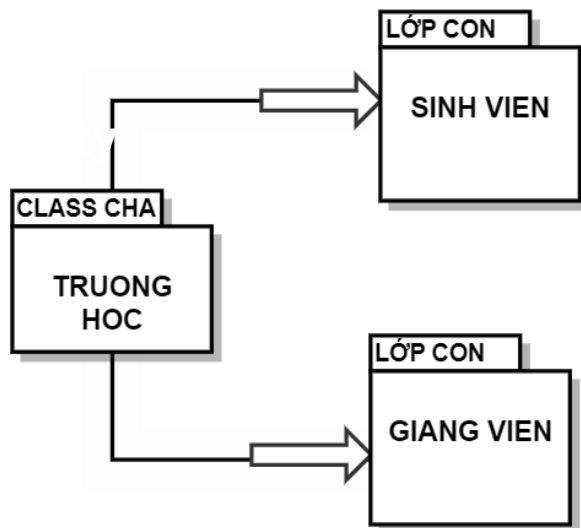
Mỗi đối tượng có thuộc tính và hành vi riêng

+ *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;
```

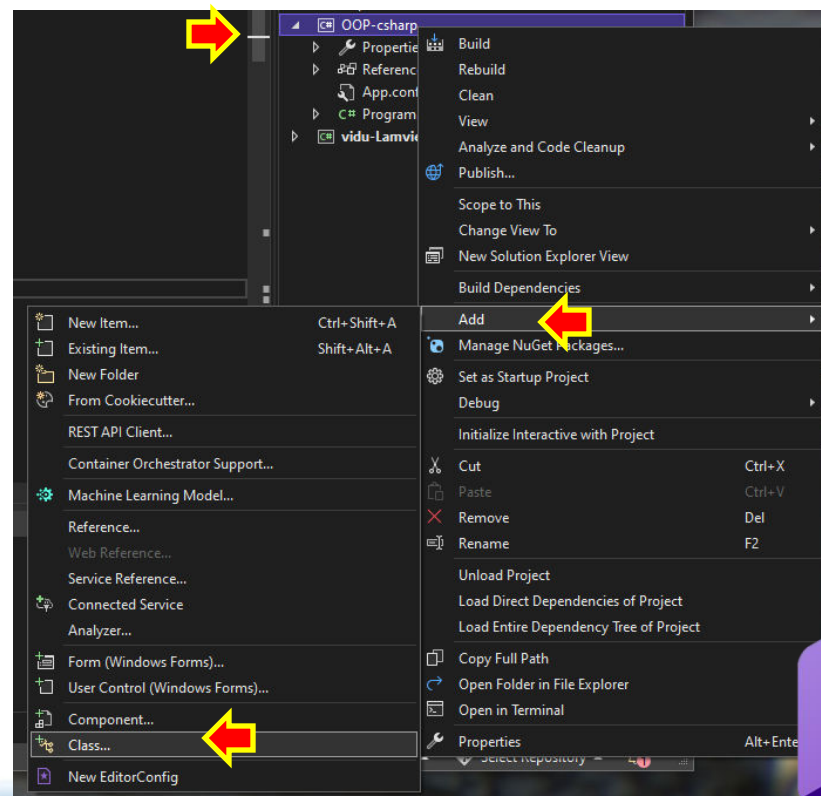
```
kieubien tenBien2;
```

```
//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
- ✓ **protected** : 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 **protected**

☺ 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;
}
```



❑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;
}
```



❑ 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 **protected**

=> *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





1

Method C#

2

Service Method & Support Method

3

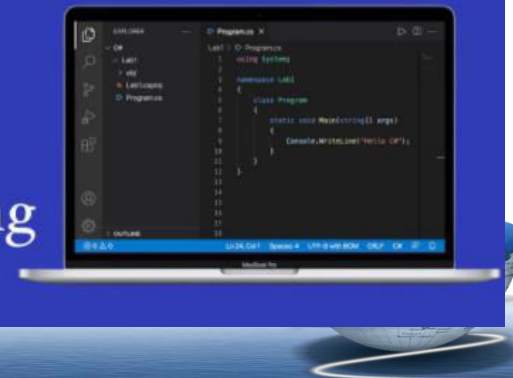
Overloading Method

4

Parameter List Method

Hướng đối tượng C# Part 2

Object
Oriented
Programming



❑ 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();`



❑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 MMethod
SinhVien sinhVien3 = new SinhVien();
sinhVien3.MaSV = 3;
sinhVien3.TenSV = "Lục văn Ba";
sinhVien3.DiemThiDH = 20;
//check điểm bằng Service Method
sinhVien3.XuatThongTin();
//Support MMethod 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



❑14. Parameter 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 **Parameter List**

```
//Parameter 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());
}
```

```
//Parameter 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
```



4

Parameter List Method

❑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
```

