



CHỦ ĐỀ 4



LỚP & ĐỐI TƯỢNG

GVGD: PHẠM THỊ KIM NGOAN

Email: ptkngoan@gmail.com

Nội dung



Định nghĩa lớp, đối tượng



Phương thức thiết lập, hủy bỏ



Con trỏ this, thành phần tĩnh



Cài đặt quan hệ



Định nghĩa toán tử trên lớp

1. Định nghĩa lớp, đối tượng

➤ Cú pháp định nghĩa lớp

```
[Thuộc tính][phạm vi truy nhập] class <tên lớp>{
```

```
    // Khai báo các thuộc tính của lớp
```

```
    // Khai báo các phương thức của lớp
```

```
}
```

[**phạm vi truy nhập**]: Là khả năng truy nhập thành phần dữ liệu (public, private, internal, protected, internal protected).

[**thuộc tính**]: có thể là thuộc tính tĩnh (static)



1. Định nghĩa lớp, đối tượng

➤ Phạm vi truy nhập

public	Có thể được truy xuất bởi bất cứ phương thức của bất kỳ lớp nào khác
private	Chỉ có thể truy xuất bởi các phương thức của chính lớp đó
protected	Có thể được truy xuất bởi các phương thức của chính lớp đó và các lớp dẫn xuất (derived) từ nó
internal	Có thể được truy xuất bởi các phương thức của các lớp trong cùng Assembly
internal protected	Có thể được truy xuất bởi các phương thức của lớp đó, lớp dẫn xuất từ lớp đó và các lớp trong cùng Assembly với nó

1. Định nghĩa lớp, đối tượng

➤ Ví dụ định nghĩa lớp

Student

- id: string
- name: string
- birthday: DateTime
- male: bool

+ void Set(string, string, DateTime, bool)
+ void Info()
+ int Age()

```
public class Student{  
    string id, name;  
    DateTime birthday;  
    bool male;  
    public void Set(string, string,  
                    DateTime, bool)  
    { ... }  
    public void Info(){ ... }  
    public int Age(){ ... }  
}
```

1. Định nghĩa lớp, đối tượng

➤ Ví dụ định nghĩa lớp

```
class Student
{
    string id, name;  DateTime birthday;   bool male;

    public void Set(string i, string n, DateTime d, bool m)
    {
        id = i;          name = n;
        birthday = d;
        male = m;
    }
    public void Info() //xuất thông tin
    {
        if(male)
            Console.WriteLine("{0}\t{1}\t{2}\t nam \t {3} years old", id,
name, birthday.ToShortDateString(), Age());
        else
            Console.WriteLine("{0}\t{1}\t{2}\t nu \t {3} years old",
id, name, birthday.ToShortDateString(), Age());
    }
    public int Age() //tính tuổi
    {
        return DateTime.Today.Year - birthday.Year;
    }
}
```

1. Định nghĩa lớp, đối tượng

➤ Định nghĩa lớp Point

Point

- x: int
 - y: int
- + void Set(int, int)
 - + void Info()
 - + float Distance(Point)

1. Định nghĩa lớp, đối tượng

➤ Khai báo đối tượng

<ten lớp> <ten đối tượng> = new<ten lớp> ([các giá trị khởi tạo])

- Đối tượng là biến kiểu tham chiếu, không phải tham trị:
 - Biến đối tượng không chứa giá trị của đối tượng
 - Biến chứa địa chỉ của đối tượng được tạo trong bộ nhớ Heap
- Ví dụ

Student s = new Student();

1. Định nghĩa lớp, đối tượng

➤ Truy xuất các thành phần

- Thuộc tính:

```
<đối tượng>.<tên thuộc tính>
```

- Phương thức:

```
<đối tượng>.<tên phương thức>([danh sách các đối số])
```

- Ví dụ

```
Student s = new Student();
```

```
s.Set(i, n, d, m);
```

```
Console.WriteLine("ma so sinh vien: {0}", s.id);
```



1. Định nghĩa lớp, đối tượng

- Sử dụng class Student để tạo ra đối tượng s

```
class Program
{
    static void Main(string[] args)
    {
        Student s = new Student(); //tạo ra đối tượng s
        DateTime d = new DateTime(2000, 2, 21);
        s.Set("62132345", "N V A", d, true); //gán giá trị
        s.Info(); //xuất thông tin
        Console.ReadKey();
    }
}
```

1. Định nghĩa lớp, đối tượng

- Sử dụng class Point để tạo ra 2 đối tượng p1, p2; tính và in ra khoảng cách giữa p1 và p2

Nội dung

Định nghĩa lớp, đối tượng

Phương thức thiết lập, hủy bỏ

Con trỏ this, thành phần tĩnh

Cài đặt quan hệ

Định nghĩa toán tử trên lớp

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập (khởi tạo)

- Tạo một đối tượng của lớp và chuyển nó sang trạng thái xác định (valid state)
- Thiết lập **thông tin ban đầu** cho một đối tượng thuộc về lớp ngay khi đối tượng được khai báo.
- Phương thức thiết lập mặc định: sẽ được CLR cung cấp nếu người lập trình không định nghĩa
- Phương thức thiết lập do người lập trình định nghĩa

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập

- Đặc điểm của phương thức thiết lập:
- ✓ Tên của phương thức thiết lập **trùng với tên lớp**.
- ✓ Phương thức thiết lập **không có giá trị trả về**, phạm vi truy xuất thường là public.
- ✓ Một lớp có thể có **nhiều phương thức thiết lập** khác nhau.
- ✓ Trong quá trình tồn tại của đối tượng, phương thức thiết lập chỉ được gọi **một lần duy nhất** khi đối tượng ra đời.

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập do người dùng định nghĩa

- ✓ Phương thức thiết lập không tham số hay gọi là phương thức **thiết lập mặc định** (default constructor)
- ✓ Phương thức **thiết lập có tham số**: Các thông tin ban đầu của đối tượng sẽ phụ thuộc vào giá trị các tham số của phương thức.
- ✓ Phương thức **thiết lập sao chép**(*copy contructor*): là phương thức thiết lập nhận tham số đầu vào là 1 đối tượng thuộc cùng 1 lớp.

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập không tham số

```
class Student
{
    string id, name;
    DateTime birthday;
    bool male;
    public Student()
    {
        id = "62132345";
        name = "Nguyen Van Nam";
        DateTime d = new DateTime(2000, 2, 21);
        birthday =d;
        male = true;
    }
    ...
}
class Program{
    Student s = new Student();
    s.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập có tham số

```
class Student
{
    string id, name;
    DateTime birthday;
    bool male;
    public Student(string i, string n, DateTime d, bool m)
    {
        id = i;
        name = n;
        birthday = d;
        male = m;
    }
    ...
}
class Program{
    ...
    Student s = new Student(i,n,d,m);
    s.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập sao chép

```
class Student
{
    string id, name;
    DateTime birthday;
    bool male;
    public Student(Student s)
    {
        id = s.id;
        name = s.name;
        birthday = s.birthday;
        male = s.male;
    }
    ...
}
class Program{
    Student s1 = new Student();
    Student s2= new Student(s1)
    s2.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức thiết lập sao chép

```
class Student:ICloneable
{
    string id, name;
    DateTime birthday;
    bool male;
    public Object Clone(
    {
        return this.MemberwiseClone();
    }
    ...
}
class Program{
    Student s1 = new Student();
    Student s2= (Student)s1.Clone();
    s2.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➤ Phương thức hủy bỏ

```
class Student
{
    string id, name;
    DateTime birthday;
    bool male;
    ~Student()
    {
        Console.WriteLine("Phuong thuc huy bo!");
    }
    ...
}
class Program{
    Student s = new Student();
    s.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

- Cài đặt các phương thức thiết lập cho lớp Point và sử dụng các phương thức thiết lập để tạo ra các đối tượng của lớp Point.

Nội dung

Định nghĩa lớp, đối tượng

Phương thức thiết lập, hủy bỏ

Con trỏ this, thành phần tĩnh

Cài đặt quan hệ

Định nghĩa toán tử trên lớp

3. Con trỏ this, thành phần tĩnh

- Từ khóa this trả đến thể hiện hiện tại (current instance) của đối tượng.

```
class Student
{
    string id, name;  DateTime birthday;   bool male;

    public void Set(string id, string n, DateTime d, bool m)
    {
        this.id = id;
        this.name = n;
        this.birthday = d;
        this.male = m;
    }
    ...
}
```

3. Con trả this, thành phần tĩnh

- Từ khóa this rất hữu ích trong một số trường hợp
 - Gọi tường minh các phương thức, thuộc tính của lớp

```
class Student
{
    string id, name;  DateTime birthday;    bool male;

    public void Info() //xuất thông tin
    { if(male)
        Console.WriteLine("{0}\t{1}\t{2}\t nam \t {3} years old", id,
name, birthday.ToShortDateString(), this.Age ());
        else
            Console.WriteLine("{0}\t{1}\t{2}\t nu \t {3} years old",
id, name, birthday.ToShortDateString(),this.Age());
    }
    public int Age() //tính tuổi
    { return DateTime.Today.Year - birthday.Year;
    }
}
```

3. Con trỏ this, thành phần tĩnh

- Thành phần tĩnh là các thành phần chung (thuộc tính, phương thức) của lớp.
- Sử dụng từ khóa **static** để khai báo một thành phần tĩnh.
- Truy xuất các thành phần tĩnh thông qua tên lớp

```
class Student
{
    string id, name;  DateTime birthday;    bool male;
    static byte num; //số tín chỉ của khóa học

    ...
}
class Program
{
    Student s=new Student();
    Console.WriteLine("So tin chi cua khoa hoc:{0}", Student.num);
}
```

3. Con trả this, thành phần tĩnh

➤ Thành phần tĩnh:

- Các thành phần tĩnh có thể được truy nhập, triệu gọi trước khi các đối tượng của lớp đó được tạo ra.
- Các phương thức tĩnh không thể truy xuất trực tiếp các thuộc tính, phương thức không tĩnh (nonstatic)

```
class Student
{
    string id, name;  DateTime birthday;   bool male;
    static byte num; //số tín chỉ của khóa học

    ...
}

class Program
{
    Student s=new Student();
    Console.Write("So tin chi cua khoa hoc:{0}", s.id);
}
```

Đóng gói thuộc tính

➤ Đóng gói dữ liệu thành thuộc tính:

- Đóng gói dữ liệu thành thuộc tính thực chất là một quá trình lấy giá trị của biến thành phần / thiết lập giá trị cho biến thành phần thông qua phương thức của lớp.
- Trong C# cung cấp khả năng khai báo hàm chung gọi là thuộc tính cho hàm get và set

```
public class SinhVien
{
    string id, name;
    // public string Id{ get; set;}
    public string Id { get => id; set => id = value; }
}
class Program
{ Student s = new Student(); Console.WriteLine(s.Id); }
```

Nội dung

Định nghĩa lớp, đối tượng

Phương thức thiết lập, hủy bỏ

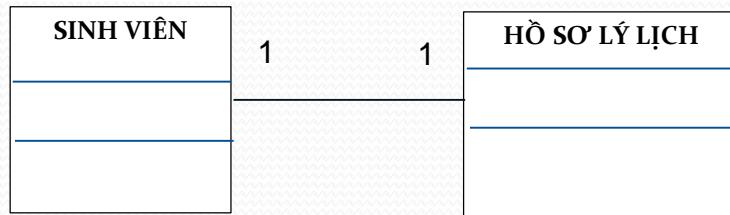
Con trỏ this, thành phần tĩnh

Cài đặt quan hệ

Định nghĩa toán tử trên lớp

Mối quan hệ liên kết

- Liên kết 1-1: thêm thuộc tính là một con trỏ trỏ tới lớp kia.

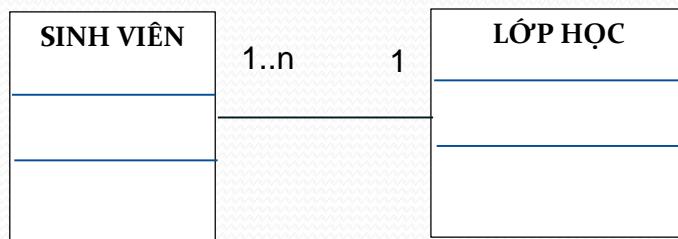


```
class HOSOLYLICH
{
    SinhVien sv;
    ...
};

class SinhVien
{
    HOSOLYLICH hoso;
    ...
};
```

Mối quan hệ liên kết

➤ Liên kết 1-n:



```
class SinhVien
{
    ...
};
```

```
class LOPHOC
{
    SinhVien[] ds;
    ...
};
```

Nội dung



Định nghĩa lớp, đối tượng



Phương thức thiết lập, hủy bỏ



Con trỏ this, thành phần tĩnh



Cài đặt quan hệ



Định nghĩa toán tử trên lớp

5. Định nghĩa toán tử trên lớp

➤ Mục đích của định nghĩa toán tử trên lớp (nạp chồng toán tử - Overloading Operator)

- Cho phép các lớp do người dùng định nghĩa có thể có các chức năng như các kiểu do ngôn ngữ định nghĩa.
- Mục đích của toán tử là để viết mã chương trình gọn gàng, dễ hiểu hơn, thay vì phải gọi phương thức.

```
public static KDL operator T (KDL lhs, KDL rhs)
```

- KDL: kiểu_dữ_liệu
- T: Phép_toán

5. Định nghĩa toán tử trên lớp

➤ Ví dụ:

```
public class Student
{
    string id, name;
    DateTime birthday;
    public static int operator +(Student s1, Student s2)
    {
        return s1.Age() + s2.Age();
    }
}
class Program
{
    Student s1 = new Student();
    Student s2 = new Student(s1);
    Console.WriteLine("Số tuổi của cả 2 sinh viên:{0}", s1+s2);
}
```

Kết thúc chủ đề 4

