LESSON 08

I. TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

❖ Khái niệm

- Với mong muốn xây dựng một phương pháp lập trình trực quan, mô tả trung thực hệ thống trong thực tế vì thế phương pháp **lập trình hướng đối tượng** ra đời
- Lập trình hướng đối tượng là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng chương trình.
- Một định nghĩa khác về lập trình hướng đối tượng đó là phương pháp lập trình dựa trên kiến trúc **lớp** (class) và **đối tượng** (object).

❖ Đối tượng:

- Trong lập trình hướng đối tượng, **đối tượng** được hiểu như là 1 thực thể: người, vật hoặc 1 bảng dữ liệu, . . .
- Một đối tượng bao gồm 2 thông tin: **thuộc tính** và **phương thức**.
  - ✓ **Thuộc tính** chính là những thông tin, đặc điểm của đối tượng. Ví dụ: một người sẽ có họ tên, ngày sinh, màu da, kiểu tóc, . . .
  - ✓ **Phương thức** là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, . . .

❖ Lớp (Class) :

- Các đối tượng có các đặc tính tương tự nhau được gom lại thành 1 **lớp đối tượng**.
- Bên trong lớp cũng có 2 thành phần chính đó là thuộc tính và phương thức.
- Ngoài ra, lớp còn được dùng để định nghĩa ra kiểu dữ liệu mới

❖ Sự khác nhau giữa đối tượng và lớp:

- **Lớp** là một khuôn mẫu còn **đối tượng** là một thể hiện cụ thể dựa trên khuôn mẫu đó.
- **Đối tượng**(Object) là một thể hiện của **một lớp**(Class)

❖ Các đặc điểm của lập trình hướng đối tượng:

Lập trình hướng đối tượng có 4 đặc điểm chính:

1. Tính đóng gói:

- Các dữ liệu và phương thức có liên quan với nhau được đóng gói thành các lớp để tiện cho việc quản lý và sử dụng.
- Ngoài ra, đóng gói còn để che giấu một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không thể nhìn thấy

2. Tính trừu tượng:

Khi viết chương trình theo phong cách hướng đối tượng, việc thiết kế các đối tượng ta cần rút tỉa ra những đặc trưng chung của chúng rồi trừu tượng thành các interface (sẽ được trình bày sau) và thiết kế xem chúng sẽ tương tác với nhau như thế nào

3. Tính kế thừa:

Lớp cha có thể chia sẽ dữ liệu và phương thức cho các lớp con,các lớp con khỏi phải định nghĩa lại, giúp chương trình ngắn gọn (sẽ được trình bày sau).

4. Tính đa hình:

Là hiện tượng các đối tượng thuộc các lớp khác nhau có thể hiểu cùng một thông điệp theo các cách khác nhau (sẽ được trình bày sau).

II. CLASS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

❖ Class trong C#

Class trong C# chính là cách thể hiện khái niệm về lớp trong lập trình hướng đối tượng.

Một **class** trong C# có các thành phần như:

- Thuộc tính: là các thành phần dữ liệu hay còn gọi là các biến.
- Phương thức: là các hàm thành phần thể hiện các hành vi của một đối tượng thuộc lớp.
- Phương thức khởi tạo.
- Phương thức huỷ bỏ.

❖ Khai báo, khởi tạo và sử dụng class trong C#

➢ **Cú pháp.**

```
class <tên lớp>
{
    <Phạm vi truy cập> <Các thành phần của lớp>;
}
```

➢ **Trong đó:**

- **<tên lớp>** là tên do người dùng đặt và tuân theo quy tắc đặt tên đã trình bày trong bài **biến.**
- **<Phạm vi truy cập>** bao gồm các từ khoá như public, protected, private, static… (Sẽ trình bày sau)
- **<Các thành phần của lớp>** bao gồm các biến, phương thức của lớp:
    - Các biến được khai báo như khai báo biến đã học trong bài **biến.**
    - Các phương thức (hàm) được khai báo như khai báo hàm đã học trong bài **hàm.**

➢ **Ví dụ:**

```
0 references
class Person
{
    public string Name;
    public int Age;
    public string Address;

    0 references
    public void Speak()
    {
        Console.WriteLine($"Hello guy! My name is {Name}. I am {Age} years old!");
    }
}
```

Với khai báo trên ta đã có 1 kiểu dữ liệu mới tên là Animal. Và ta hoàn toàn có thể khai báo biến và sử dụng nó.

- Từ khoá **public** là chỉ phạm vi truy cập của các thành phần bên trong **class** (sẽ được trình bày sau). Hiện tại bạn chỉ cần hiểu ghi **public** là để khi sử dụng ta có thể truy cập các thành phần này.
- Lớp **Person** có 3 thuộc tính là **Name**, **Age, Address** và 1 phương thức là **Speak**. Như vậy mọi đối tượng thuộc lớp này đều có 4 thành phần trên.

❖ Khởi tạo.

Bạn có thể khởi tạo 1 đối tượng thuộc lớp thông qua toán tử new.

```
0 references
static void Main(string[] args)
{
    Person hoan = new Person();
}
```

Class là kiểu dữ liệu tham chiếu vì thế đối tượng dữ liệu thực sự được lưu trên heap.

❖ Sử dụng.

Về cơ bản thì **class** được sử dụng tương tự như **struct**.

Để gọi đến các thuộc tính bên trong lớp:

<tên đối tượng>.<tên thuộc tính>;

Để gọi đến các phương thức bên trong lớp:

<tên đối tượng> . <tên phương thức> (<tham số nếucó>);

➢ **Ví dụ:**

```
0 references
static void Main(string[] args)
{
    Person hoan = new Person();
    hoan.Name = "Pham Hoan";
    hoan.Age = 18;

    Person khu = new Person();
    khu.Name = "Van Khu";
    khu.Age = 16;

    hoan.Speak();
    khu.Speak();
    Console.ReadKey();
}
```

```
G:\My Drive\C Sharp and Revit API Basic\Pham Hoan\Source\Visual Studio Source Code\Lesson08\bin\Debug\Lesson08.exe
Hello guy! My name is Pham Hoan. I am 18 years old!
Hello guy! My name is Van Khu. I am 16 years old!
```

❖ Phương thức khởi tạo

Vì lập trình hướng đối tượng là phương pháp giúp ánh xạ thế giới thực vào thế giới lập trình một cách dễ dàng nên từ đó đã xuất hiện 2 khái niệm **phương thức khởi tạo** và **phương thức huỷ bỏ** để thể hiện ý trên.

Phương thức khởi tạo (Constructor) là những phương thức đặc biệt được gọi đến ngay khi khởi tạo 1 đối tượng nào đó.

Đặc điểm

- Có tên trùng với tên lớp
- Không có kiểu trả về.
- Được tự động gọi khi 1 đối tượng thuộc lớp được khởi tạo.
- Nếu như bạn không khai báo bất kỳ phương thức khởi tạo nào thì hệ thống sẽ tự tạo ra phương thức khởi tạo mặc định không đối số và không có nội dung gì.
- Có thể có nhiều constructor bên trong 1 lớp

Phân loại: Có 2 loại phương thức khởi tạo:

✓ Phương thức khởi tạo **không đối số**:

- là phương thức khởi tạo không có bất kỳ tham số truyền vào nào.
- Thường dùng để khởi tạo các giá trị mặc định cho các thuộc tính bên trong class khi khởi tạo đối tượng (giá trị mặc định này do người lập trình quyết định).

✓ Phương thức khởi tạo **có đối số**:

- là phương thức khởi tạo có tham số truyền vào. Và khi khởi tạo đối tượng để phương thức này được gọi ta cần truyền đầy đủ các tham số.
- Thường dùng để khởi tạo các giá trị cho các thuộc thuộc tính bên trong class khi khởi tạo đối tượng (các giá trị này do người khởi tạo đối tượng truyền vào)

```
//Constructor
2 references
public Person()
{
    Name = "NoName";
    Age = 1;
    Address = "Ha Noi";
}

0 references
public Person(string name, int age)
{
    Name = name;
    Age = age;
}
```

```
0 references
static void Main(string[] args)
{
    Person noname = new Person();
    Person hoan = new Person( name: "Hoan", age: 18);

    noname.Speak();
    hoan.Speak();

    Console.ReadKey();
}
```

```
G:\My Drive\C Sharp and Revit API Basic\Pham Hoan\Source\Visual Studio Source Code\Lesson08\bin\Debug\Lesson08.exe
Hello guy! My name is NoName. I am 1 years old!
Hello guy! My name is Hoan. I am 18 years old!
```

❖ Phương thức huỷ bỏ

Phương thức huỷ bỏ (destructor) là phương thức đặc biệt được gọi đến trước khi 1 đối tượng bị thu hồi:

- Có tên trùng với tên lớp nhưng để phân biệt với constructor thì ta thêm dấu "~" vào trước tên lớp
- Được tự động gọi khi 1 đối tượng thuộc lớp kết thúc "vòng đời" của nó thông qua bộ thu dọn rác tự động GC (Garbage Collection).
- Nếu bạn không khai báo destructor thì C# sẽ tự động tạo ra 1 destructor mặc định và không có nội dung gì.
- Chỉ có 1 destructor duy nhất trong 1 lớp

III. CÁC LOẠI PHẠM VI TRUY CẬP TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG.

❖ Các loại phạm vi truy cập và ý nghĩa.

Phạm vi truy cập là cách mà người lập trình quy định về quyền được truy xuất đến các thành phần của lớp.

Trong C# có 5 loại phạm vi truy cập:

Phạm vi	Ý nghĩa
public	Không hạn chế, thành phần mang thuộc tính này có thể được truy cập ở bất kỳ nơi nào
private	Thành phần mang thuộc tính này là thành phần riêng chỉ có nội bộ bên trong lớp mới có quyền truy cập - Chỉ gán từ khóa này cho trường (field), phương thức, thuộc tính get/set, các biến.... Nhưng không thể gán cho class. Lưu ý class được mặc định ẩn là Internal, chỉ khi bạn gán public thì class đó mới được công khai
protected	Tương tự như private ngoài ra còn có thể truy cập từ lớp dẫn xuất chứa nó (tính kế thừa)
internal	Chỉ truy cập trong cùng 1 project Nó giới hạn các class chỉ được sử dụng trong một Assembly (*.exe,*.dll) - Các khối khác nhau vẫn có thể truy xuất với nhau nhưng phải thông qua: Friend Assemblies
protected internal	Tương tự iternal ngoài ra có thể truy cập từ lớp dẫn xuất chứa nó (tính kế thừa)

Lưu ý:

- Nếu khai báo lớp mà không chỉ ra phạm vi cụ thể thì phạm vi mặc định là internal.
- Nếu khai báo thành phần bên trong lớp mà không chỉ ra phạm vi cụ thể thì phạm vi mặc định là private.

➢ Ví dụ:

```
0 references
class Student
{
    public string Name;
    public int Age;

    private double Math;
    private double Physic;

    private string Grade;

}
```

❖ Phương thức truy vấn, phương thức cập nhật. (Phần tham khảo)

Trong C#, **phương thức truy xuất** và **phương thức cập nhật** đã được nâng cấp lên thành 1 cấu trúc mới ngắn gọn hơn và tiện dụng hơn đó là **property**.

Sử dụng property giúp ta có thể thao tác dữ liệu tự nhiên hơn nhưng vẫn đảm bảo tính đóng gói của lập trình hướng đối tượng.

➢ Cú pháp:

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

- **<kiểu dữ liệu>** là kiểu dữ liệu của property. Thường sẽ trùng với kiểu dữ liệu của thuộc tính **private** tương ứng bên trong lớp.
- **<tên property>** là tên do người dùng đặt và tuân theo quy tắc đặt tên
- **get**, **set**, **value** là từ khoá có ý nghĩa:
  - ✓ Từ khoá **get** tương đương với phương thức truy vấn.
  - ✓ Từ khoá **set** tương đương với phương thức cập nhật.

- ✓ Từ khoá **value** đại diện cho giá trị mà người gán vào property (tương đương với tham số truyền vào của phương thức cập nhật).

- **<tên thuộc tính>** là tên thuộc tính thực sự bên trong lớp.

➢ Ví dụ:

```
private string grade;
0 references
public string Grade
{
    get { return grade; }
    set { grade = value; }
}
```

Khi sử dụng thì ta xem **property** như một thuộc tính đã được **public**

BÀI TẬP:

1. Yêu cầu: Thiết kế (lớp) khối hộp bao gồm :

- Dài, rộng, cao.
- Nhập thông tin trên.
- Tính diện tích xung quanh, thể tích.
- In ra thông tin hình hộp.

2. Yêu cầu: Thiết kế lớp sinh viên bao gồm các thuộc tính:

- Họ tên, tuổi, điểm toán, điểm văn, điểm trung bình của 1 sinh viên.
- Khai báo mảng sử dụng lớp sinh viên trên để nhập thông tin cho n sinh viên (n nhập từ bàn phím).
- Tính điểm trung bình và in ra màn hình danh sách các sinh viên đó.