Buổi 2

Tổng quan về Lập trình hướng đối tượng

I. Đặt vấn đề

Giả sử cần giải quyết 1 bài toán sau: Viết chương trình nhập thông tin sinh viên, thông tin mỗi sinh viên gồm có họ tên, tuổi, điểm lý thuyết, điểm thực hành.

Nếu cài đặt chương trình theo hướng lập trình truyền thống (lập trình tuyến tính), chúng ta cần phải khai báo rất nhiều biến, mỗi sinh viên đã phải cần đến 4 biến tương ứng với 4 thông tin ở trên, dẫn đến chương trình rất dài và khó quản lý, bảo trì và phát triển sau này. Do đó, cần phải tổ chức lại dữ liệu thành các cấu trúc ngắn gọn, hợp lý.

Việc này có thể được thực hiện bằng cách xây dựng 1 kiểu dữ liệu cấu trúc (**struct**) tương tự như ở C++, trong đó chúng ta xây dựng kiểu dữ liệu `SinhVien` như sau:

```
struct SinhVien
{
        string hoTen;
        int tuoi;
        float diemLT;
        float diemTH;
}
```

Bên cạnh việc sử dụng struct thì chúng ta có thể tạo ra 1 lớp đối tượng (**class**). Về cơ bản thì struct và class khá tương tự nhau, tuy nhiên class có nhiều tính năng nâng cao hơn so với struct (xem phần V.2).

II. Khái niệm lập trình hướng đối tượng

1. Lớp đối tượng – Class

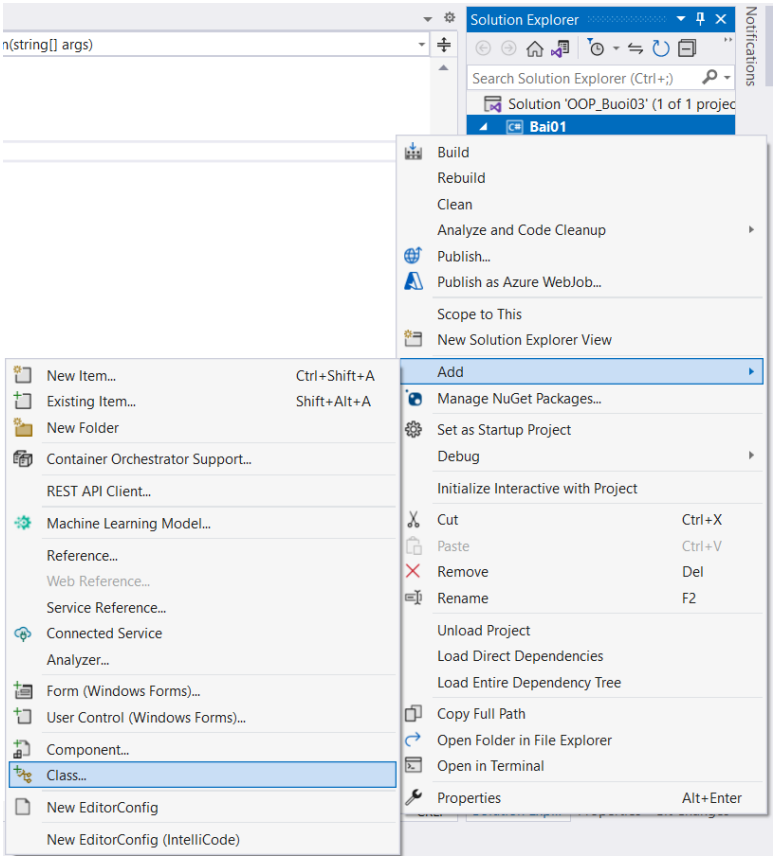
Trong môn học này, chúng ta sẽ tìm hiểu về phương pháp lập trình hướng đối tượng (**OOP – Object-Oriented Programming**). Đây là phương pháp lập trình bằng cách khai báo các lớp đối tượng (**class**), từ đó tạo ra các đối tượng (**object**) giúp giải quyết yêu cầu của bài toán.

1 class được xem như là khuôn mẫu (template) để tạo ra các object cùng loại, có cùng cách hoạt động. Trong class sẽ bao gồm 2 loại thành phần sau:

- Thuộc tính (**Data Member/Field**): Còn gọi là thành phần dữ liệu, gồm các trường dữ liệu, các biến để lưu trữ thông tin.
- Phương thức (**Functional Member/Method**): Còn gọi là thành phần chức năng, là các hàm xử lý được viết riêng cho class, và chỉ được phép gọi thông qua các object của class này.

Class có thể được khai báo ở bất kỳ đâu: trong 1 file riêng, trong phương thức `Main()` của class `Program` (file *Program.cs*), thậm chí là bên trong 1 class khác. Tuy nhiên, để dễ quản

lý, mỗi class nên được đặt trong 1 file riêng theo cách sau: click phải vào tên project, chọn **Add** → **Class...**, trong cửa sổ hiện ra, đặt tên class cần tạo ở mục **Name**, sau đó click nút **Add**:



File tạo ra sẽ có tên dạng *tên_class.cs*

Ví dụ: Tạo class `SinhVien` (nằm trong file *SinhVien.cs*) gồm 4 thông tin như ví dụ ở phần I, kèm theo các chức năng nhập và xuất:

```
internal class SinhVien
{
    public string hoTen;
    public int tuoi;
    public double diemLT;
    public double diemTH;

    public void Nhap()
    {
        Console.Write("Nhập họ tên: ");
        hoTen = Console.ReadLine();
        Console.Write("Nhập tuổi: ");
        tuoi = Convert.ToInt32(Console.ReadLine());
        Console.Write("Nhập điểm LT: ");
        diemLT = Convert.ToDouble(Console.ReadLine());
        Console.Write("Nhập điểm TH: ");
        diemTH = Convert.ToDouble(Console.ReadLine());
    }

    public void Xuat()
    {
        Console.WriteLine($"Thông tin SV: {hoTen} - {tuoi} tuổi - LT:
{diemLT} - TH: {diemTH}");
    }
}
```

Thuộc tính

Phương thức

Trong ví dụ trên, các từ khóa `public` và `internal` sẽ được giải thích ở buổi 5. Hiện tại, chúng ta sẽ tạm chấp nhận rằng: thuộc tính và phương thức sẽ khai báo kèm từ khóa `public`.

2. Đối tượng – Object

Nếu như class được xem như là khuôn mẫu, thì đối tượng (**object**) chính là sự thể hiện của class đó. Để khai báo 1 đối tượng, chúng ta dùng cú pháp tương tự như khai báo biến, nhưng có kèm theo toán tử `new` như sau:

```
tên_class tên_đối_tượng = new tên_class();
```

Ví dụ: Với class `SinhVien`, có thể tạo ra nhiều object sinh viên khác nhau, tất cả object này đều có 4 thông tin họ tên, tuổi, điểm lý thuyết, điểm thực hành, và có chức năng nhập và xuất thông tin. Trong phương thức `Main()` của class `Program` khai báo 2 object như sau:

```
SinhVien sv1 = new SinhVien();
SinhVien sv2 = new SinhVien();
```

Để gọi các thuộc tính và phương thức của 1 đối tượng, chúng ta dùng toán tử `.` như sau:

```
// Gán giá trị cho thuộc tính hoTen của object sv1
sv1.hoTen = "Nguyễn Văn A";

// Lấy giá trị thuộc tính hoTen của object sv1
Console.WriteLine($"Họ tên SV1: {sv1.hoTen}");

// Gọi phương thức Nhap() của object sv2
sv2.Nhap();
```

Việc khai báo class và tạo object đã giải quyết vấn đề đặt ra ở phần I, đó là thay vì phải tạo ra 4 biến để lưu trữ 4 thông tin của 1 sinh viên, thì giờ đây chỉ cần 1 đối tượng duy nhất là đã có thể lưu trữ được 4 thông tin này (thông qua 4 thuộc tính). Ngoài ra, các chức năng liên quan đến sinh viên cũng đều được cài đặt tập trung trong class `SinhVien`.

III. Phương thức khởi tạo – Constructor

1. Khái niệm

Với class `SinhVien`, mỗi object được tạo ra sẽ có giá trị mặc định như sau (đều là giá trị mặc định của kiểu dữ liệu tương ứng – `string`, `int`, `double`):

- `hoTen`: chuỗi rỗng
- `tuoi`: 0
- `diemLT`: 0
- `diemTH`: 0

Giả sử chúng ta muốn 1 object sinh viên được tạo ra sẽ có họ tên là Nguyễn Văn A, tuổi là 18, điểm lý thuyết và điểm thực hành đều là 5.0 thay vì các giá trị mặc định ở trên.

Để làm được việc này, lập trình hướng đối tượng cung cấp các phương thức khởi tạo (**constructor**) cho phép khởi tạo giá trị ban đầu cho object ngay khi nó được tạo ra.

Đặc điểm của phương thức khởi tạo:

- Có tên trùng với tên lớp
- Không có kiểu trả về (kể cả `void`)
- Phải được khai báo `public`.

2. Các loại phương thức khởi tạo

Có 3 loại phương thức khởi tạo:

a) Phương thức khởi tạo mặc định (**Default constructor**)

Đây là phương thức khởi tạo không kèm theo tham số, dùng để cài đặt các giá trị mặc định cho object. Mỗi class có đúng 1 phương thức khởi tạo mặc định:

```
internal class SinhVien
{
      public SinhVien()
      {
            hoTen = "Nguyễn Văn A";
            tuoi = 18;
            diemLT = diemTH = 5;
      }
}
```

Để gọi phương thức khởi tạo mặc định khi tạo ra object, chúng ta dùng câu lệnh sau:

```
SinhVien sv3 = new SinhVien();
```

Lưu ý: Nếu như lập trình viên không cài đặt bất kỳ phương thức khởi tạo nào cho class, 1 phương thức khởi tạo mặc định với phần cài đặt rỗng sẽ được tự động phát sinh để đảm bảo object luôn có thể được tạo ra mà không gây lỗi.

b) Phương thức khởi tạo sao chép (**Copy constructor**)

Đây là phương thức khởi tạo có 1 tham số là 1 object cùng kiểu, dùng để sao chép tham số này thành object mới. Mỗi class có tối đa 1 phương thức khởi tạo sao chép:

```
internal class SinhVien
{
      ……
      public SinhVien(SinhVien sv)
      {
            hoTen = sv.hoTen;
            tuoi = sv.tuoi;
            diemLT = sv.diemLT;
            diemTH = sv.diemTH;
      }
}
```

Để gọi phương thức khởi tạo sao chép khi tạo ra object, chúng ta dùng câu lệnh sau:

```
SinhVien sv4 = new SinhVien(sv3);
```

Tuy nhiên, trên thực tế, để sao chép object, chúng ta thường sử dụng toán tử gán thay vì dùng phương thức khởi tạo sao chép.

c) Phương thức khởi tạo kèm tham số (**Parameterized constructor**)

Đây là phương thức khởi tạo có tham số tùy ý, tùy thuộc vào mục đích của lập trình viên. Mỗi class có thể có 1 hoặc nhiều phương thức khởi tạo kèm tham số. Ví dụ dưới đây cho thấy 1 phương thức khởi tạo nhận 1 tham số là họ tên SV, và 1 phương thức khởi tạo khác nhận 2 tham số là họ tên SV và tuổi:

```
internal class SinhVien
{
	public SinhVien(string fullName)
	{
		hoTen = fullName;
	}

	public SinhVien(string fullName, int age)
	{
		hoTen = fullName;
		tuoi = age;
	}
}
```

Trong số các phương thức khởi tạo kèm tham số, phổ biến nhất là phương thức khởi tạo đầy đủ tham số tương ứng với tất cả thuộc tính của class:

```
internal class SinhVien
{
	public SinhVien(string fullName, int age, double theoryScore,
double practicalScore)
	{
		hoTen = fullName;
		tuoi = age;
		diemLT = theoryScore;
		diemTH = practicalScore;
	}
}
```

Khi đó, việc tạo ra 1 object sinh viên với đầy đủ thông tin rất dễ dàng:

```
SinhVien sv5 = new SinhVien("Trần Thị B", 20, 5.9, 8.5);
```

Snippet: `ctor` dùng để tạo ra phương thức khởi tạo.

3. Từ khóa this

Với phương thức khởi tạo đầy đủ tham số như phần III.2.c ở trên, việc đặt tên tham số như vậy là không nhất quán, dẫn đến bất tiện cho lập trình viên. Trong trường hợp này, theo quy chuẩn đặt tên, tên tham số nên đặt trùng tên với thuộc tính tương ứng:

```
public SinhVien(string hoTen, int tuoi, double diemLT, double diemTH)
{
	hoTen = hoTen;
	tuoi = tuoi;
	diemLT = diemLT;
	diemTH = diemTH;
}
```

Tuy nhiên, nếu làm vậy thì các câu lệnh gán giá trị ở trên đều vô nghĩa, vì chúng đều đem 1 biến gán cho chính nó. Vấn đề này xảy ra là do tên thuộc tính và tên tham số đang trùng nhau. Vì vậy, để phân biệt thuộc tính và tham số, chúng ta cần dùng tới từ khóa `this`:

```
public SinhVien(string hoTen, int tuoi, double diemLT, double diemTH)
{
	this.hoTen = hoTen;
	this.tuoi = tuoi;
	this.diemLT = diemLT;
	this.diemTH = diemTH;
}
```

Đối với 1 phương thức, `this` chính là object đang gọi phương thức hiện tại. Trong ví dụ sau đây, khi thực hiện phương thức khởi tạo đầy đủ tham số, `this` chính là object `sv6`:

```
SinhVien sv6 = new SinhVien("Lê Hoàng C", 19, 8.9, 9.5);
```

Nhờ có từ khóa `this`, giờ đây chúng ta có thể phân biệt rõ thuộc tính và tham số trong trường hợp chúng được đặt tên giống nhau. Ngoài ra, `this` còn có thể được dùng khi cần thao tác với đối tượng đang gọi phương thức.

Lưu ý: Để code được gọn gàng, chỉ nên sử dụng `this` khi thật sự cần thiết.

4. Giá trị mặc định cho thuộc tính

Bên cạnh việc sử dụng phương thức khởi tạo mặc định, chúng ta có thể gán giá trị mặc định cho các thuộc tính như sau:

```
internal class SinhVien
{
	public string hoTen = "Nguyễn Văn A";
	public int tuoi = 18;
	public double diemLT = 5;
	public double diemTH = 5;
}
```

Nếu làm vậy, khi tạo ra đối tượng (dù là bằng phương thức khởi tạo loại nào đi nữa), các thuộc tính sẽ mang giá trị mặc định như đã khai báo, trừ khi phương thức khởi tạo đang dùng có gán lại giá trị khác cho thuộc tính đó.

IV. Các tính chất của lập trình hướng đối tượng

Lập trình hướng đối tượng có 4 tính chất sau, sẽ được trình bày ở các buổi sau:

- Tính đóng gói (**Encapsulation**)
- Tính kế thừa (**Inheritance**)
- Tính đa hình (**Polymorphism**)
- Tính trừu tượng (**Abstraction**)

V. Bài tập

Thực hiện các bài tập sau, mỗi bài nằm trong 1 project của solution **OOP_Buoi02**. Với mỗi bài tập, yêu cầu cài đặt cả 3 loại phương thức khởi tạo.

Bài 1: Viết chương trình cho phép nhập thông tin 1 sinh viên, sau đó xuất thông tin sinh viên vừa nhập ra màn hình. Thông tin của 1 sinh viên gồm có:

- Họ và tên
- Tuổi
- Điểm lý thuyết và điểm thực hành

Khi xuất thông tin sinh viên, cần tính toán và xuất thêm điểm trung bình và kết quả đậu hay là thi lại của sinh viên. Sinh viên tính là đậu nếu điểm trung bình ≥ 5 và thi lại nếu điểm trung bình < 5.

Bài 2: Viết chương trình cho phép nhập 2 phân số (tử số và mẫu số là số nguyên). Tính và xuất ra màn hình tổng, hiệu, tích và thương của 2 phân số này. Rút gọn phân số kết quả.

Gợi ý: Các công thức tính toán cho phân số:

$$\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd} \qquad \frac{a}{b}-\frac{c}{d}=\frac{ad-bc}{bd}$$

$$\frac{a}{b}\times\frac{c}{d}=\frac{ac}{bd} \qquad \frac{a}{b}\div\frac{c}{d}=\frac{ad}{bc}$$

Bài 3: Viết chương trình cho phép nhập tọa độ 3 đỉnh của 1 tam giác trong không gian Oxy (hoành độ và tung độ là số thực). Tính toán và xuất ra màn hình thông tin của tam giác vừa nhập, kèm theo chu vi và diện tích của nó. Cho biết tam giác này có vuông hay không.

Gợi ý:

- Nên tạo 1 class lưu thông tin 1 điểm trong không gian Oxy.
- Công thức tính khoảng cách 2 điểm $A(x_A, y_A)$ và $B(x_B, y_B)$:

$$AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

- Diện tích tam giác được tính theo công thức Heron, với p là nửa chu vi tam giác:

$$S_{\Delta ABC} = \sqrt{p(p - AB)(p - BC)(p - CA)}$$

- Kiểm tra tam giác vuông bằng định lý Pythagoras: Một tam giác vuông khi và chỉ khi bình phương 1 cạnh bằng tổng của bình phương 2 cạnh còn lại.