CHỦ ĐỀ 4

LỚP VÀ 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 các quan hệ
- Định nghĩa toán tử trên lớp

1. Định nghĩa lớp, đối tượng

Cú pháp định nghĩa lớp

```
<access specifier> class class_name
{  // member data ~ variables
      <access specifier> <data type> variable1;
      ...
      <access specifier> <data type> variableN;

   // member methods
      <access specifier> <return type> method1(parameter_list)
      { // method body }
      ...
      <access specifier> <return type> methodN(parameter_list)
      { // method body }
}
```

1. Định nghĩa lớp, đối tượng

- **<access specifier>:** Là khả năng truy nhập thành phần của lớp.

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 SinhVien

Student
- id: string - name: string + mark: float
+ void Set(string, string, float) + void Info() + bool Scholarship(float)

```
public class Student{
    string _id, _name;
    public float _mark;

    public void Set(string, string, float)
    { ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
```

1. Định nghĩa lớp, đối tượng

➢ Ví dụ định nghĩa lớp

```
public class Student{
    string _id, _name;
    public float _mark;

    public void Set(string i, string n, float m)
    {    _id = i;  _name = n; _mark = m;    }
    public void Info()
    {    Console.Write($"_id}\t{_name}\t{_mark}");    }
    public bool Scholarship(float f)
    {    if(_mark >= f) return true;
         else return false;
    }
}
```

1. Định nghĩa lớp, đối tượng

➢ Khai báo đối tượng

```
<class_name> <object_name> = new <class_name> ([values])
```

- 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

- Dữ liệu:

```
<object_name>.<variable>
```

- Phương thức:

```
< object_name >.<method>([parameter_list])
```

- Ví dụ

```
Student s = new Student();
s.Set(i, n, m);
Console.WriteLine("diem cua sinh vien: {o}", s._mark);
```

1. Định nghĩa lớp, đối tượng

➢ 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 Student
{    string _id, _name; float _mark;
//   public string Id{ get{return _id;} set{ _id=value;}}
     public string Id { get => _id; set => _id = value; }
}
class Program
{  Student sv = new Student();  Console.Write(sv.Id);}
```

1. Định nghĩa lớp, đối tượng

- Ví dụ: **Định nghĩa lớp Point**

Point
- x: int - y: int
+ void Set(int, int) + void Print()

- ✓ Phương thức Print(): hiển thị thông tin ra màn hình dạng (x, y).
- ✓ Phương thức Set(int, int): gán giá trị của 2 tham số truyền vào cho x, y.

- Tạo ra 1 đối tượng lớp Point và sử dụng các phương thức của đối tượng.

Nội dung

- Định nghĩa lớp, đối tượng
- Phương thức thiết lập, hủy bỏ
- Con trỏ this, thành phần tĩnh
- Cài đặt các quan hệ
- Định nghĩa toán tử trên lớp

2. Phương thức thiết lập, hủy bỏ

➢ Phương thức thiết lập (khởi tạo) - Constructors

- 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
- Người lập trình có thể định nghĩa các phương thức thiết lập cho lớp.

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

- **Các loại phương thức thiết lập**:

✓ Phương thức thiết lập không tham số hay gọi là phương thức thiết lập mặc định (default contructor)

✓ 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

Ví dụ phương thức thiết lập mặc định

```
public class Student{
    string _id, _name;
    float _mark;
    public Student()
    {    _id ="63131234"; _name = "Nguyen Thi Lan";
         _mark = 7.2f;}
    public void Set(string i, string n, float m){ ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
class Program{
  Student s = new Student();
  s.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➢ Ví dụ phương thức thiết lập có tham số

```
public class Student{
    string _id, _name;
    float _mark;
    public Student(string i, string n, float m){
          _id = i; _name = n; _mark = m;}
    public void Set(string, string, float){ ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
class Program{
  Student s = new Student(“63131234”,”Nguyen Thi Lan”,7.2f);
  s.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

Ví dụ phương thức thiết lập sao chép

```
public class Student{
    string _id, _name;
    float _mark;
    public Student(Student s){
          _id = s._id; _name = s._name; _mark = s._mark;}
    public void Set(string, string, float){ ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
class Program{
  Student s = new Student();
  Student s1= new Student(s);
  s1.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

Ví dụ phương thức thiết lập sao chép dùng ICloneable

```
public class Student: ICloneable
{    string _id, _name;
     float _mark;
    public Object Clone(){
          return this.MemberwiseClone;}
    public void Set(string, string, float){ ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
class Program{
  Student s = new Student();
  Student s1= (Student)s.Clone();
  s1.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

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

```
public class Point{
    int x, y;
  //các phương thức thiết lập
        ...
    public void Set(int a, int b)
    {   ...   }
    public void Info()
    {   ...   }
}
//lớp chương trình chính
class Program{
    static void Main()
    {  ...    }
}
```

2. Phương thức thiết lập, hủy bỏ

➢ Phương thức hủy bỏ - Destructor

- Đặc điểm của phương thức hủy:

✓ Tên của phương thức hủy trùng với tên lớp.

✓ Có dấu ngã ~ trước tên phương thức.

✓ Phương thức hủy không có giá trị trả về, không có tham số, phạm vi truy xuất thường là public.

✓ Dùng để giải phóng bộ nhớ.

2. Phương thức thiết lập, hủy bỏ

Ví dụ phương thức hủy bỏ

```
public class Student: ICloneable
{   string _id, _name;
    float _mark;
    public void Set(string, string, float){ ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
    ~Student()
    {  Console.WriteLine("Object is being deleted");}
}
class Program{
  Student s = new Student();
  s.Info();
}
```

Nội dung

- Định nghĩa lớp, đối tượng
- Phương thức thiết lập, hủy bỏ
- Con trỏ this, thành phần tĩnh
- Cài đặt các quan hệ
- Định nghĩa toán tử trên lớp

3. Con trỏ this, thành phần tĩnh

- Từ khóa this trỏ đến thể hiện hiện tại (current instance) của đối tượng.
- Từ khóa this rất hữu ích trong một số trường hợp:
  - Chỉ rõ thành phần (thuộc tính) của đối tượng, tránh nhầm lẫn với tên biến, tránh sự nhập nhằng về tên.

```
public class Student
{    string _id, _name;
     float _mark;
   public void Set_dtb(float mark){
        this._mark = mark; }
}
```

3. Con trỏ this, thành phần tĩnh

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

```
public class Student
{   string _id, _name;
    float _mark;
    public void Set_mark(float mark){
        this._mark = mark; }
     public bool Scholarship(float d) {
        ...
        this.Set_mark(d);
    }
}
```

3. Con trỏ this, thành phần tĩnh

- Từ khóa this rất hữu ích trong một số trường hợp:
  - Trả về đối tượng hiện tại của lớp.

```
public class Time
{   int h, m, s;
          ...
    public Time Increase(int i){
         s = s + i;
         ...
         return this;
    }
}
```

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

```
public class Student
{    string _id, _name; float _mark;
     public static float mark_S = 8;
          ...
}
class Program
{    Student s = new Student();
     Console.Write(Student.mark_S);
}
```

3. Con trỏ 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)

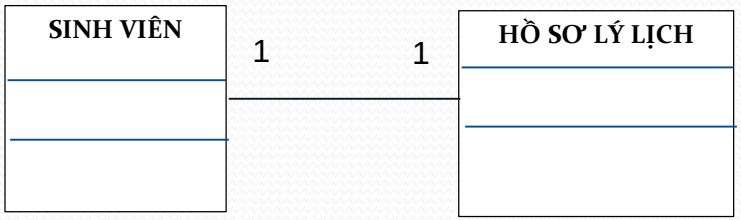
```
public class Student
{    string _id, _name; float _mark;
     public static float mark_S = 8;
     public static Scholarship(float d)
     {  if (_mark > d) ...
}
```

Nội dung

- Định nghĩa lớp, đối tượng
- Phương thức thiết lập, hủy bỏ
- Con trỏ this, thành phần tĩnh
- Cài đặt các quan hệ
- Định nghĩa toán tử trên lớp

Mối quan hệ liên kết

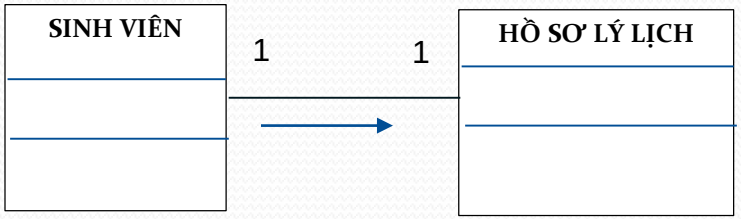
- **Liên kết 1-1:** thêm thuộc tính là một con trỏ trỏ tới lớp kia.




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

Mối quan hệ liên kết

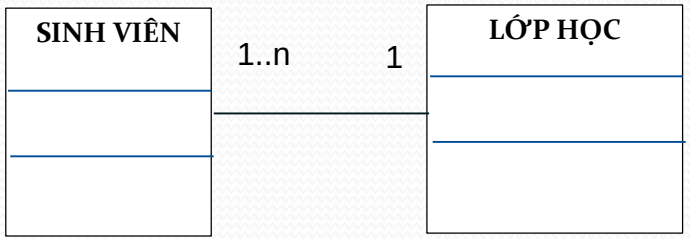
- **Liên kết 1-1:** cài đặt một chiều - 1 SV có 1 HSLL.




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

Mối quan hệ liên kết

- **Liên kết 1-n:** Cài đặt 2 chiều – 1 Sinh viên thuộc về một Lớp học và 1 Lớp học có nhiều Sinh viên.

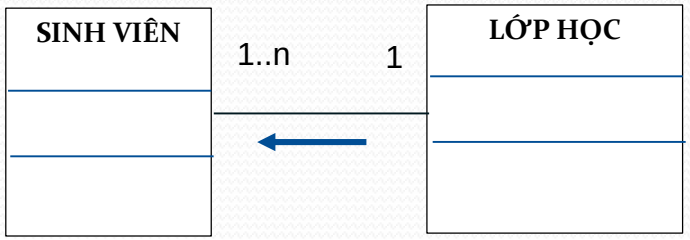



```
class  SinhVien
{      LopHoc  lop;
          ...
};
```

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

Mối quan hệ liên kết

- **Liên kết 1-n:** Cài đặt một chiều – 1 Lớp học có nhiều Sinh viên.




```
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 các 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 - Operator Overloading)
  - 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 <data-type>  operator T (<data-type> operand1 [,..])
{  …  }
```

  - <data-type>: kiểu_dữ_liệu
  - T: Phép_toán
  - Operand: toán hạng

5. Định nghĩa toán tử trên lớp

TT.	Các phép toán có thể nạp chồng
1	!, ~, ++, -- Các phép toán một toán hạng.
2	+, -, *, /, % Các phép toán hai toán hạng.
3	==, !=, <, >, <=, >= Các phép toán so sánh.

5. Định nghĩa toán tử trên lớp

- Ví dụ: Định nghĩa toán tử cộng 2 đối tượng Sinh viên với ý nghĩa cộng điểm (mark) của 2 Sinh viên đó lại với nhau.

```
public class Student
{    string _id, _name; float _mark;
     public static float operator +(Student s1, Student s2)
    {
        return s1._mark + s2._mark;
    }
}
class Program
{   Student s1 = new Student();
    Student s2 = new Student(s1);
    Console.Write("{0}",s1 + s2);
}
```

Kết thúc chủ đề 4