CHỦ ĐỀ 4

CLASS & OBJECT

Nội dung

- Định nghĩa lớp, đối tượng
- Phương thức thiết lập, hủy bỏ
- Con trỏ this, thành phần tĩnh
- 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

SinhVien
- ma: string - hoTen: string - dtb: float
+ void Set(string, string, float) + void Info() + bool Scholarship(float)

```
public class SinhVien{
    string ma, hoten;
    float dtb;
    public void Set(string, string, float)
    { ...}
    public void Info(){ ...}
    public bool Scholarship(float f){ ...}
}
```

1. Định nghĩa lớp, đối tượng

➢ Khai báo đối tượng

<tên lớp> <tên đối tượng> = new<tên 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

  SinhVien sv = new SinhVien();

1. Định nghĩa lớp, đối tượng

➢ Truy xuất các thành phần

- Thuộc tính:

```
<tên đối tượng>.<tên thuộc tính>
```

- Phương thức:

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

- Ví dụ

```
SinhVien sv = new SinhVien();
sv.Set(m, ht, d);
Console.WriteLine("ma so sinh vien: {0}", sv.ma);
```

Nội dung

- Định nghĩa lớp, đối tượng
- Phương thức thiết lập, hủy bỏ
- Con trỏ this, thành phần tĩnh
- 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 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

➢ Phương thức thiết lập không tham số

```
public class SinhVien{
    string ma, hoten;
    float dtb;
    public SinhVien(){
        ma =“59121212”; hoten = “Nguyen Van An”;
        dtb =3.2;}
    ...
}
class Program{
  SinhVien sv = new SinhVien();
  sv.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➢ Phương thức thiết lập có tham số

```
public class SinhVien{
    string ma, hoten;
        float dtb;
    public SinhVien(string m, string ht, float d){
          ma = m; hoten = ht; dtb = d;}
    ...
}
class Program{
  SinhVien sv = new SinhVien("60121212","Nguyen Van An",6.5);
  sv.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➢ Phương thức thiết lập sao chép

```
public class SinhVien{
    string ma, hoten;
        float dtb;
    public SinhVien(SinhVien s){
          ma = s.ma; hoten = s.hoten; dtb = s.dtb;}
    ...
}
class Program{
  SinhVien sv = new SinhVien();
  SinhVien sv1= new SinhVien(sv);
  sv1.Info();
}
```

2. Phương thức thiết lập, hủy bỏ

➢ Phương thức hủy bỏ

```
public class SinhVien: ICloneable
{    string ma, hoten;
       float dtb;
   ...
   ~SinhVien(){
       Console.Write("Phương thuc huy bo!");
       ...
   }
}
class Program{
  SinhVien sv = new SinhVien();
  sv.Info();
}
```

Nội dung

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

3. Con trỏ this, thành phần tĩnh

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

```
public class SinhVien
{    string ma, hoten;
        float dtb;
    public void Set_dtb(float dtb){
        this.dtb = dtb; }
}
```

3. Con trỏ 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

```
public class SinhVien
{    string ma, hoten;
        float dtb;
   public void Set_dtb(float dtb){
        this.dtb = dtb; }
    public bool Scholarship(float d) {
        ...
        this.Set_btb(d);
   }
}
```

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 SinhVien
{    string ma, hoten; float dtb;
     public static float diemHB =3;
          ...
}
class Program
{
   SinhVien sv = new SinhVien();
   console.Write(SinhVien.diemHB);
}
```

Phương thức get và set

- Thuộc tính và sự bao đóng dữ liệu
  - Mục đích của sự bao đóng (encapsulation) là ngăn chặn truy xuất dữ liệu từ bên ngoài. Để truy xuất đến các thành phần private, cần bổ sung các phương thức get và set ở phần khai báo thuộc tính.

```
class Person
{
  private string name; // field

  public string Name   // property
  {
    get { return name; }   // get method
    set { name = value; }  // set method
  }
}
```

Phương thức get và set (tt)

```
class Program
{
  static void Main(string[] args)
  {
    Person myObj = new Person();
    myObj.Name = "Liam";
    Console.WriteLine(myObj.Name);
  }
}
```

Mảng các đối tượng

- Khai báo mảng đối tượng

<Tên lớp>[] <biến mảng> = new <Tên lớp>[Số phần tử];

- Chú ý, cần phải tạo mới từng đối tượng sau khi khai báo mảng.
- Ví dụ: Mảng các phân số
  - Nhập mảng
  - Xuất mảng
  - Tìm phân số lớn nhất
  - Sắp xếp tăng dần

Nội dung

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

4. Định nghĩa toán tử trên lớp

- Mục đích của định nghĩa toán tử trên lớp (nạp chồng toán tử - OverloadingOperator)
  - 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

4. Định nghĩa toán tử trên lớp

- Ví dụ:

```
public class SinhVien
{    string ma, hoten; float dtb;
     public static float operator +(SinhVien s1, SinhVien s2)
    {
         return s1.dtb + s2.dtb;
    }
}
class Program
{   SinhVien sv1 = new SinhVien();
    SinhVien sv2 = new SinhVien(sv1);
    console.Write("{0}",sv1+sv2);
}
```

Kết thúc chủ đề 4