Object-Oriented Programming (OOP)
Lập trình Hướng đối tượng

TH.S ĐOÀN THIỆN MINH

3/23/2020

Chương 3 Classs

1. Định nghĩa lớp (Class)

2. Phương thức (Methods)

3. Thuộc tính truy cập (Attribute)

4. Tạo đối tượng

Chương 3 Classs

5. Hủy đối tượng

6. Con trỏ this

7. Sử dụng thành viên tĩnh

8. Nạp chồng phương thức

9. Đóng gói dữ liệu với thành phần thuộc tính

7. Static Member

Sử dụng các thành viên tĩnh (static member)

7. Static

- Những thuộc tính và phương thức trong một lớp có thể là:
  - Những thành viên thể hiện (instance members)
  - Những thành viên tĩnh (static members).

7. Static Member

- Những thuộc tính và phương thức trong một lớp có thể là:
  - Những thành viên thể hiện (instance members): dùng để **mô tả đối tượng**
    - Truy xuất thành viên thể hiện (instance members) thông qua tên **đối tượng**.
    - Thành viên thể hiện (instance members): được xem là toàn cục trong phạm vi từng đối tượng
    - Dữ liệu thuộc về riêng mỗi đối tượng , đối tượng khác nhau thuộc tính có giá trị khác nhau

7. Static Member

- Những thuộc tính và phương thức trong một lớp có thể là:
  - Những thành viên tĩnh (static members): được xem như là 1 phần của lớp
    - **Truy xuất** thông **qua tên lớp**
    - Thành viên toàn tĩnh được xem là toàn cục trong phạm vi một lớp (được gọi thông qua tên lớp)

7. Static Member

- Những thuộc tính và phương thức trong một lớp có thể là:
  - Những thành viên tĩnh (static members): được xem như là 1 phần của lớp, mọi đối tượng cấp phát chung 1 vùng nhớ.
    - **Truy xuất** thông **qua tên lớp**
    - Thành viên toàn tĩnh được xem là toàn cục trong phạm vi một lớp

7. Static Member

- Những thuộc tính và phương thức trong một lớp có thể là:
  - Những thành viên tĩnh (static members): được xem như là 1 phần của lớp, mọi đối tượng cấp phát chung 1 vùng nhớ.
    - Có các loại thành viên tĩnh (static) sau:
      - Biến tĩnh (static variable)
      - Phương thức tĩnh (static method) và phương thức khởi tạo tĩnh (static constructor)
      - Lớp tinh (static class)

7. Static Member

- Biến tĩnh (static variable):
  - Cú pháp:

    <phạm vi truy cập> **static** <kiểu dữ liệu> <tên biến>;
  - Đặc điểm:
    - ✓ Được **gọi** thông qua **tên lớp** và **không** cần **khởi tạo** bất kỳ đối tượng nào
    - ✓ Khởi tạo khi chương trình được thực thi và hủy khi kết thúc chương trình

7. Static Member

- Biến tĩnh (static variable):

```
class Ctime {
        int h, m, s;
        public static int cout_variable_static=0;
        public Ctime()
        {
            h = m = s = 0;
            cout_variable_static += 1;
        }}
```

```
class Program{
    static void Main(string[] args)
    {
            Ctime t1 = new Ctime();
            Console.Write("\n cout_variable_static={0}", Ctime.cout_variable_static);
            Ctime t2 = new Ctime();
            Console.Write("\n cout_variable_static={0}", Ctime.cout_variable_static);
    }}
```

7. Static Member

- Phương thức tĩnh (static method):
  - Cú pháp:

    <phạm vi truy cập> **static** <kiểu trả về> <tên phương thức>;

    ví dụ:

```
public static void Print()
{ //your code}
```

7. Static Member

- Phương thức tĩnh (static method):
  - Đặc điểm:
    - ✓ Được **gọi** thông qua **tên lớp** và **không** cần **khởi tạo** bất kỳ đối tượng nào → tránh lãng phí
    - ✓ Trong phương thức có sử dụng **biến static** thì **phương thức** cũng phải khai báo là **static**
    - ✓ **Khởi tạo** khi chương trình được **thực thi** và **hủy** khi **kết thúc** chương trình

7. Static Member

- Phương thức tĩnh (static method):

Ví dụ:

```
class student
    {
        public static void Print(string Firstname, string Lastname)
        {
            Console.Write("Ten sinh vien: {0} {1}", Firstname, Lastname);
        }
    }

  static void Main(string[] args)
  {
            student.Print("No", "Nguyen Thi");
  }
```

7. Static Member

- Phương thức khởi tạo tĩnh (static constructor):
  - Cú pháp

```
Static <tên lớp>
{
    // nội dung hàm khởi tạo
}
```

Ví dụ:

```
Public class user
    {
        static user()
        { // your code }
    }
```

7. Static Member

- Phương thức khởi tạo tĩnh (static constructor):
  - Đặc điểm:
    - ✓ **Không** được phép **khai báo** phạm vi truy cập;
    - ✓ Phương thức khởi tạo tĩnh (Static constructor) sẽ được gọi **một lần duy nhất** khi chương trình được nạp vào bộ nhớ để thực thi
    - ✓ Phương thức khởi tạo tĩnh (Static constructor) cũng giống các phương thức tĩnh (static method) **không** thể gọi các thuộc tính không phải static (static variable)

7. Static Member

- Phương thức khởi tạo tĩnh (static constructor):
  - Ví dụ:

```
static void Main(string[] args)
        {
            user u = new user();
            user u1 = new user();
            Console.ReadKey();
        }
```

```
class user
   {
       static user()
       {
           Console.Write("\n Bo khoi tao static constructor");
       }
       public user()
       {
           Console.Write("\n Bo khoi tao public constructor");
       }
   }
```

Kết quả

```
Bo khoi tao static constructor
Bo khoi tao public constructor
Bo khoi tao public constructor
```

7. Static Member

- Lớp tĩnh (static class):
  - Cú pháp:

<phạm vi truy cập> **static class** <tên lớp>

Ví dụ:

```
public static class mymath
{
    //your code
}
```

7. Static Member

- Lớp tĩnh (static class):
  - Đặc điểm:
    - ✓ **Lớp tĩnh** (static class) chỉ **chứa** các **thành phần tĩnh** (biến tĩnh-static variable, phương thức tĩnh- static method) và **đều** được **truy cập** qua **tên lớp.**
    - ✓ **Không thể** khai báo, khởi tạo 1 đối tượng thuộc lớp tĩnh

7. Static Member

- Lớp tĩnh (static class):
  - Ví dụ:

```
public static class mymath{
    public static float pi = 3.14f;
    public static int luythua(int n)
    {
            return n * n * n; ;
    }}
static void Main(string[] args)
{
  Console.WriteLine("\n gia tri cua pi la: {0}", mymath.pi);
  Console.WriteLine("\n luy thua cua n la: {0}", mymath.luythua(2));
}
```

Kết quả

```
gia tri cua pi la: 3.14

luy thua cua n la: 8
```

TIME TO RELAX