BÀI GIẢNG

# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## CHƯƠNG 4

# ĐÓNG GÓI

**TRẦN THỊ THU THẢO**

BỘ MÔN TIN HỌC QUẢN LÝ, KHOA THỐNG KÊ – TIN HỌC

TRƯỜNG ĐẠI HỌC KINH TẾ, ĐẠI HỌC ĐÀ NẴNG

THAOTRAN@DUE.EDU.VN

# NỘI DUNG

1. Khái niệm Đóng gói
2. Các chỉ thị truy cập
3. Phương thức truy vấn và phương thức cập nhật
4. Thành viên tĩnh của lớp
5. Tham số của phương thức
6. Tham chiếu this
7. Chỉ mục indexer

# ĐÓNG GÓI

## ❑ KHÁI NIỆM

**Tính đóng gói** là khả năng che dấu thông tin của đối tượng với môi trường bên ngoài. Tính chất này giúp đảm bảo tính toàn vẹn và bảo mật của đối tượng.

Tính đóng gói được thể hiện thông qua các từ khóa xác định phạm vi sử dụng:

- `public`: cho phép truy cập khi đứng ở ngoài lớp
- `private`: chỉ cho phép truy cập khi đứng ở trong lớp
- `protected`: chỉ cho phép truy cập khi đứng ở trong lớp hoặc ở lớp con
- `internal`: được truy cập từ các lớp nằm trong cùng một assembly

# CHỈ THỊ TRUY CẬP public

## ❑ KHÁI NIỆM

Chỉ thị truy cập public cho phép một lớp đưa ra các thuộc tính và các phương thức thành viên của nó cho các phương thức và đối tượng khác. Bất kỳ thành viên nào cũng có thể được truy cập từ bên ngoài lớp

# CHỈ THỊ TRUY CẬP public

**Ví dụ 1:**

Xây dựng một lớp Hình chữ nhật gồm:

Thuộc tính: Chiều dài và chiều rộng.

Phương thức:

- Hàm khởi tạo
- Hàm nhập, xuất
- Hàm tính diện tích (Tinhdientich)

Tính diện tích hình chữ nhật rồi in kết quả lên màn hình

# CHỈ THỊ TRUY CẬP public

## Ví dụ 1:

```
class HinhChuNhat
{
    public double Dai { get; set; }
    public double Rong { get; set; }
     public double Tinhdientich()
    {
        return Dai * Rong;
    }
    public void Xuat()
    {
        Console.WriteLine("Chieu dai: {0}", Dai);
        Console.WriteLine("Chieu rong: {0}", Rong);
        Console.WriteLine("Dien tich HCN: {0}", Tinhdientich());
    }
}
```

Các thuộc tính đều mang phạm vi là public

Các phương thức mang phạm vi là public

# CHỈ THỊ TRUY CẬP public

## Ví dụ 1:

```
class Program
{
    static void Main(string[] args)
    {
        HinhChuNhat hcn = new HinhChuNhat();
        hcn.Dai = 4.5;
        hcn.Rong = 3.5;
        hcn.Xuat();
        Console.ReadLine();
    }
}
}
```

Thuộc tính Dai, Rong, phương thức Xuat(), Tinhdientich() có thể truy cập từ phương thức Main() bằng các tạo một thể hiện của lớp Hinhchunhat

```
Chieu dai: 4.5
Chieu rong: 3.5
Dien tich HCN: 15.75
```

# CHỈ THỊ TRUY CẬP private

## ❑ KHÁI NIỆM

Chỉ thị truy cập **private** cho phép lớp ẩn các trường và các phương thức thành viên khỏi các phương thức và đối tượng khác.

Chỉ các thành viên trong cùng một lớp mới có thể truy cập các thành viên **private** của nó. Ngay cả thể hiện của lớp cũng không thể truy cập các thành viên **private** của nó

Nếu một trường hoặc phương thức không có chỉ thị truy cập thì nó sẽ được gắn chỉ thị truy cập mặc định là **private**

# CHỈ THỊ TRUY CẬP private

## Ví dụ 2:

```
namespace C4Vd2
{
    class Hinhchunhat
    {
        private double _dai;
        private double _rong;
        public Hinhchunhat(double dai, double rong)
        {
            _dai = dai;
            _rong = rong;
        }
        public double Tinhdientich()
        {
            return _dai * _rong;
        }
```

Các thuộc tính đều mang phạm vi là **private** → Không thể truy cập từ phương thức **Main()**

Các phương thức mang phạm vi là **public** → có thể truy cập từ phương thức **Main()** bằng các sử dụng thể hiện **hcn** của lớp **Hinhchunhat**

# CHỈ THỊ TRUY CẬP private

## Ví dụ 2:

```
    public void Xuat()
      {
         Console.WriteLine("Chieu Dai: {0}", _dai);
         Console.WriteLine("Chieu Rong: {0}", _rong);
         Console.WriteLine("Dien tich HCN: {0}", Tinhdientich());
      }
  }
    class Program
       {
          static void Main(string[] args)
          {
             Hinhchunhat hcn = new Hinhchunhat(5.5, 3.5);
             hcn.Xuat();
             Console.ReadLine();
          }
       }
    }
```

```
Chieu Dai: 5.5
Chieu Rong: 3.5
Dien tich HCN: 19.25
```

# CHỈ THỊ TRUY CẬP protected

## ❑ KHÁI NIỆM

Chỉ thị truy cập **protected** cho phép lớp ẩn các trường và các phương thức thành viên khỏi các phương thức và đối tượng khác. Nó chỉ cho phép một lớp con truy cập các trường và các phương thức thành viên này của lớp cha (hoặc lớp cơ sở)

Các thành viên trong class cha và class con mới có thể truy cập các thành viên **protected** của lớp cha. Ngay cả thể hiện của lớp con hay lớp cha cũng không thể truy cập các thành viên của **protected** class cha

Chỉ thị truy cập **protected** tương tự chỉ thị truy cập **private**, chỉ khác là nó cho phép class con truy cập các trường và phương thức thành viên của class cha có chỉ thị protected.

# CHỈ THỊ TRUY CẬP protected

## Ví dụ 3:

Các thuộc tính đều mang phạm vi là protected, nên không thể được truy cập từ phương thúc Main()

Các phương thức SetRong,SetDai và Xuat() của class Shape là public nên chúng được class hcn kế thừa và có thể truy cập ở phương thức Main()

```
namespace C4Vd3 {
    class Shape {
        protected int _rong;
        protected int _dai;
        public void SetRong(int rong)
        {
            _rong = rong;
        }
        public void SetDai(int dai)
        {
            _dai = dai;
        }
        public void Xuat()
        {
            Console.WriteLine("rong = {0}, dai = {1}", _rong, _dai);
        }
    }
```

# CHỈ THỊ TRUY CẬP protected

**Ví dụ 3:**

```
    class hcnTester
    {
        static void Main(string[] args)
        {
            Shape shape = new Shape();
            shape.SetRong(4);
            shape.SetDai(6);
            shape.Xuat();
            }
    }
}
```

# CHỈ THỊ TRUY CẬP protected

## Ví dụ 3:

Lớp hcn kế thừa lớp Shape nên có thể truy cập được trường _dai và _rong để tính diện tích trong phương thức TinhDientich()

```
class hcn: Shape
{
   public int TinhDientich()
   {
      return (_rong * _dai);
   }
}
class hcnTester {
   static void Main(string[] args)
   {
     hcn hcn = new hcn();
      hcn.SetRong(5);
      hcn.SetDai(7);
      hcn.Xuat();
      Console.WriteLine("Total area: {0}", hcn. TinhDientich());
      Console.ReadKey();
      }
}
```

```
rong = 4, dai = 6
rong = 5, dai = 7
Total area: 35
```

# CHỈ THỊ TRUY CẬP internal

## ❑ KHÁI NIỆM

Chỉ thị truy cập **internal** cho phép lớp đưa ra các thuộc tính và phương thức thành viên của nó cho các phương thức và đối tượng khác trong cùng assembly

Chỉ thị truy cập **internal** tương tự như chỉ thị **public**, nó chỉ khác ở chỗ chị thị giới hạn phạm vi truy cập **internal** trong cùng một assembly còn chỉ thị **public** thì không có bất kỳ giới hạn nào

# CHỈ THỊ TRUY CẬP internal

**Ví dụ 3:**

```
namespace C4Vd3 {
    public class HinhChuNhat
    {
        internal double Dai;
        internal double Rong;
        double TinhDientich() // private method
        {
            return Dai * Rong;
        }
        public void Xuat()
        {
            Console.WriteLine("Dai: {0}", Dai);
            Console.WriteLine("Rong: {0}", Rong);
            Console.WriteLine("Dien tich: {0}", TinhDientich());
        }
    }
```

# CHỈ THỊ TRUY CẬP internal

## Ví dụ 3:

```
class Program
    {
        static void Main(string[] args)
        {
            HinhChuNhat hcn = new HinhChuNhat();
            hcn.Dai = 4.5;
            hcn.Rong = 3.5;
            hcn.Xuat();
            Console.ReadLine();
        }
    }
}
```

```
Dai: 4.5
Rong: 3.5
Dien tich: 15.75
```

# PHƯƠNG THỨC TRUY VẤN VÀ CẬP NHẬT

## ❑ KHÁI NIỆM

- **Phương thức truy vấn**

**Phương thức truy vấn** là phương thức giúp người dùng có thể xem được dữ liệu của 1 thuộc tính nào đó. Cụ thể, phương thức truy vấn chỉ cần trả về giá trị của thuộc tính tương ứng là đủ.

- **Phương thức cập nhật**

**Phương thức cập nhật** là phương thức giúp người dùng có thể thay đổi giá trị cho 1 thuộc tính nào đó. Cụ thể, phương thức cập nhật chỉ cần thực hiện cập nhật giá trị mới cho thuộc tính tương ứng (có thể kiểm tra tính đúng đắn của dữ liệu trước khi truyền vào).

# PHƯƠNG THỨC TRUY VẤN VÀ CẬP NHẬT

## ❑ QUY ƯỚC

- **Một số quy ước nhỏ về cách đặt tên của phương thức truy vấn và phương thức cập nhật**

- Những phương thức truy vấn nên bắt đầu bằng từ khoá **get** và kèm theo sau là tên thuộc tính tương ứng.

  Ví dụ: getHoTen(), getDiemToan(), . . .

- Những phương thức cập nhật nên bắt đầu bằng từ khoá **set** và kèm theo sau là tên thuộc tính tương ứng.

  Ví dụ: setDiemToan(), setHoTen(), . . .

- Nếu thuộc tính kiểu luận lý (bool) thì tên phương thức truy vấn nên bắt đầu bằng từ khoá is và kèm theo sau là tên thuộc tính tương ứng.
- Phương thức truy vấn sẽ có kiểu trả về trùng với kiểu dữ liệu của thuộc tính tương ứng và không có tham số truyền vào.
- Phương thức cập nhật sẽ có kiểu trả về là void và có 1 tham số truyền vào có kiểu dữ liệu trùng với kiểu dữ liệu của thuộc tính tương ứng.

# PHƯƠNG THỨC TRUY VẤN VÀ CẬP NHẬT

**Ví dụ 4:** Viết chương trình quản lý điểm CSLT của Sinh viên

```
class Sinhvien
    {
        private string MASV;
        private double DiemCSLT;

        public string getMASV()
        {
            return MASV;
        }

        public void setDiemCSLT(int diemcslt)
        {
            DiemCSLT = diemcslt;
        }
    }
```

Các thuộc tính đều mang phạm vi là private

Phương thức truy vấn giá trị thuộc tính MASV

Phương thức cập nhật giá trị thuộc tính DiemCSLT.

Vì thế phương thức có 1 tham số truyền vào kiểu double trùng với kiểu của DiemCSLT

# PHƯƠNG THỨC TRUY VẤN VÀ CẬP NHẬT

## ❑Ví dụ 4:

```
class Sinhvien
    {
         public void ShowInfo()
        {
            MASV = "1234";
            Console.WriteLine("SV Ma so {0} co diem CSLT la: {1} ", MASV,
DiemCSLT);
        }
    }
class DiemHP
    {
        static void Main(string[] args)
        {
            Sinhvien SV1 = new Sinhvien();
            SV1.getMASV();
            SV1.setDiemCSLT(8);
            SV1.ShowInfo();
        }
    }
```

```
SV Ma so 1234 co diem CSLT la: 8
```

# get & set

## ❑ TỪ KHÓA get VÀ set

- 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.

# get & set

## ❑ TỪ KHÓA get VÀ set

- **CÚ PHÁP:**

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

Trong đó:

• **<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.

# get & set

## ❑ TỪ KHÓA get VÀ set

- 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.

# get & set

## ❑ TỪ KHÓA get VÀ set

- **Ví dụ 4:** Viết chương trình quản lý điểm CSLT của sinh viên

```
class Sinhvien
    {

        private string MASV;
        private double DiemCSLT;

        public double diemcslt
        {
            get {return DiemCSLT;}
            set {DiemCSLT = value;}
        }
        public string masv
        {
            get {return MASV;}
            set {MASV = value;}
        }
    }
```

# ❑ TỪ KHÓA get VÀ set

- **Ví dụ 4:**

```
class DiemHP
    {
        static void Main(string[] args)
        {
            Sinhvien SV1 = new Sinhvien();
            SV1.masv = "5678";
            SV1.diemcslt = 9;
            Console.WriteLine("SV Ma so {0} co diem CSLT la: {1} ",
SV1.masv, SV1.diemcslt);

        }
    }
```

```
SV Ma so 5678 co diem CSLT la: 9
```

# THÀNH VIÊN TĨNH CỦA LỚP

## ❑ KHÁI NIỆM

- Thuộc tính hoặc phương thức trở thành dạng tĩnh nếu có từ khóa `static` trước phần khai báo thuộc tính hoặc phương thức. Nó là thành viên của lớp chứ không phải của đối tượng.
- Thành viên tĩnh được khởi tạo khi khai báo lớp. Thành viên tĩnh được sử dụng để lưu trữ và chia sẻ giá trị dùng chung giữa tất cả đối tượng được tạo từ lớp.

# THÀNH VIÊN TĨNH CỦA LỚP

- **Đặc điểm của thành viên tĩnh:**
  - Được khởi tạo **1 lần duy nhất** ngay khi biên dịch chương trình.
  - Có thể **dùng chung** cho mọi đối tượng.
  - **Được gọi thông qua tên lớp.**
  - Được **huỷ khi kết thúc** chương trình.
- **Có 4 loại thành viên tĩnh chính:**
  - Biến tĩnh (static variable).
  - Phương thức tĩnh (static method).
  - Lớp tĩnh (static class).
  - Phương thức khởi tạo tĩnh (static constructor).

# THÀNH VIÊN TĨNH CỦA LỚP

- **Biến tĩnh:**
  - **Cú pháp:**

**<phạm vi truy cập> static <kiểu dữ liệu> <tên biến> = <giá trị khởi tạo>;**

Trong đó:

- **<phạm vi truy cập>** là các phạm vi truy cập của biến
- **static** là từ khoá để khai báo thành viên tĩnh.
- **<kiểu dữ liệu>** là kiểu dữ liệu của biến
- **<tên biến>** là tên biến do người dùng đặt và tuân thủ các quy tắc đặt tên biến
- **<giá trị khởi tạo>** là giá trị ban đầu mà biến tĩnh này chứa. Nếu bạn không khai báo giá trị này thì C# thì tự gán giá trị mặc định và đưa ra 1 cảnh báo khi bạn biên dịch chương trình.

# THÀNH VIÊN TĨNH CỦA LỚP

- **Ví dụ 5**: Quản lý số lượng học sinh đang có (Giả sử 1 đối tượng được tạo ra là 1 học sinh)

```
class Hocsinh
    {
        private string ten;
        public string TEN
        {
            get {return ten;}
            set {ten = value;}
        }
        public static int Count = 0;
        public Hocsinh()
        {
            Console.Write("Nhap ho ten hoc sinh: ");
            ten = Console.ReadLine();
            Count++;
        }
    }
```

# THÀNH VIÊN TĨNH CỦA LỚP

- **Biến tĩnh:**
  - **Ví dụ 5**: Quản lý số lượng học sinh đang có (Giả sử 1 đối tượng được tạo ra là 1 học sinh)

```
class SodoHocsinh
{
    static void Main()
    {
        Console.WriteLine("So luong hoc sinh ban dau:
"+Hocsinh.Count);
        Hocsinh hs1 = new Hocsinh();
        Console.WriteLine("So luong hoc sinh ban dau:
"+Hocsinh.Count);
        Hocsinh hs2 = new Hocsinh();
        Console.WriteLine("So luong hoc sinh ban dau:
"+Hocsinh.Count);
    }
}
```

```
So luong hoc sinh ban dau: 0
Nhap ho ten hoc sinh: Ha
So luong hoc sinh ban dau: 1
Nhap ho ten hoc sinh: Van
So luong hoc sinh ban dau: 2
```

# THÀNH VIÊN TĨNH CỦA LỚP

- **Phương thức tĩnh:**
  - **Cú pháp:**

```
<phạm vi truy cập> static <kiểu trả về> <tên phương thức>
{
        // nội dung phương thức
}
```

Trong đó:

- **<phạm vi truy cập>** là các phạm vi truy cập của phương thức
- **static** là từ khoá để khai báo thành viên tĩnh.
- **<kiểu trả về>** là kiểu trả về của phương thức
- **<tên phương thức>** là tên biến do người dùng đặt và tuân thủ các quy tắc đặt tên

# THÀNH VIÊN TĨNH CỦA LỚP

- **Phương thức tĩnh:**
  - **Hàm tĩnh:** được sử dụng với 2 mục đích chính:
  - Hàm tĩnh là 1 hàm dùng chung của lớp. Được gọi thông qua tên lớp và không cần khởi tạo bất kỳ đối tượng nào, từ đó tránh việc lãng phí bộ nhớ.
  - Hỗ trợ trong việc viết các hàm tiện ích để sử dụng lại.

# THÀNH VIÊN TĨNH CỦA LỚP

- **Ví dụ 6: Viết một Lớp *Tinhtoan* tính lũy thừa của một số nguyên**

```
class Tinhtoan
    {
        public static long LuyThua(int CoSo, int SoMu)
        {
            long KetQua = 1;
            for (int i = 0; i < SoMu; i++)
            {
                KetQua *= CoSo;
            }
            return KetQua;
        }
    }
```

# THÀNH VIÊN TĨNH CỦA LỚP

- **Ví dụ 6: Viết một Lớp *Tinhtoan* tính lũy thừa của một số nguyên**

```
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(Tinhtoan.LuyThua(3, 3));
        }
    }
```

```
PS E:\OOP\C3Vd6> dotnet run
27
```

# THÀNH VIÊN TĨNH CỦA LỚP

- **Lớp tĩnh:**
  - **Cú pháp:**

```
<phạm vi truy cập> static class <tên lớp>
{
                    // các thành phần của lớp
}
```

Trong đó:

- **<phạm vi truy cập>** là các phạm vi truy cập của lớp
- **static** là từ khoá để khai báo thành viên tĩnh.
- **class** là từ khoá để khai báo lớp.
- **<tên lớp>** là tên do người dùng đặt và tuân thủ các quy tắc đặt tên

# THÀNH VIÊN TĨNH CỦA LỚP

- **Lớp tĩnh:**
  - **Đặc điểm**

- Chỉ chứa các thành phần tĩnh (biến tĩnh, phương thức tĩnh).
- **Không thể** khai báo, khởi tạo 1 đối tượng thuộc lớp tĩnh.

Với 2 đặc điểm trên có thể thấy lớp tĩnh thường được dùng với mục đích khai báo 1 lớp tiện ích chứa các hàm tiện ích hoặc hằng số vì:

- Ràng buộc các thành phần bên trong lớp phải là **static**.
- Không cho phép tạo ra các đối tượng dư thừa làm lãng phí bộ nhớ.
- Mọi thứ đều được truy cập thông qua tên lớp.

- **Ví dụ: Trong ví dụ 6, có thể sử dụng lớp tĩnh, phương thức tĩnh để khai báo. Ví dụ lớp Math**

Lớp Math chứa:

- Các hằng số như **PI, E.**
- Các phương thức hỗ trợ tính toán như: **sin, cos, tan, sqrt, exp, . . .**

# THÀNH VIÊN TĨNH CỦA LỚP

- **Lớp tĩnh:**

  - **Ví dụ: sử dụng lớp Math tính lũy thừa**

```
Console.WriteLine("Tinh luy thua: ");
Console.WriteLine(Math.Pow(3,3));
```

```
Tinh luy thua:
27
```

# THAM SỐ CỦA PHƯƠNG THỨC

- **Ví dụ 7: Xây dựng lớp Hoanvi để hoán vị 2 số a và b**

```
class Hoanvi
{
    static void Hoan_vi(int a, int b)
    {
        int temp=a;
        a=b;
        b=temp;
        Console.WriteLine("Trong phuong thuc Hoan_vi:
a={0}, b={1}",a,b);
     }
```

# THAM SỐ CỦA PHƯƠNG THỨC

- **Ví dụ 7: Xây dựng lớp Hoanvi để hoán vị 2 số a và b**

```
 static void Main(string[] args)
  {
     int x=3; int y=4;
     Console.WriteLine("Truoc khi goi phuong thuc
Hoan_vi: x={0}, y={1}",x,y);
      Hoan_vi(x,y);
      Console.WriteLine("Sau khi goi phuong thuc
Hoan_vi: x={0}, y={1}",x,y);
  }
}
```

# THAM SỐ CỦA PHƯƠNG THỨC

- **Truyền tham trị bằng tham số kiểu giá trị**

Ở ví dụ 7, ở lớp `Hoanvi`, `a, b` là hai tham số dạng tham trị của phương thức `Hoan_vi`, nên mọi sự thay đổi chỉ diễn ra trong thân phương thức này mà không ảnh hưởng đến đối số x, y được truyền vào.

Khi chạy chương trình, ta nhận được kết quả:

```
Truoc khi goi phuong thuc Hoan_vi: x=3, y=4
Trong phuong thuc Hoan_vi: a=4, b=3
Sau khi goi phuong thuc Hoan_vi: x=3, y=4
```

# THAM SỐ CỦA PHƯƠNG THỨC

- **Truyền tham chiếu bằng tham số kiểu giá trị với từ khóa ref hoặc out**

Để phương thức `Hoan_vi` cho ra kết quả như mong muốn ta phải sửa lại tham số a, b theo kiểu tham chiếu như sau:

```
static void Hoan_vi(ref int a, ref int b)
```

Khi đó ta gọi phương thức `Hoan_vi` với hai đối số x, y theo cú pháp:

```
Hoan_vi(ref x, ref y);
```

```
Truoc khi goi phuong thuc Hoan_vi: x=3, y=4
Trong phuong thuc Hoan_vi: a=4, b=3
Sau khi goi phuong thuc Hoan_vi: x=4, y=3
```

# THAM CHIẾU this

Khi một đối tượng đang thực thi một phương thức của thể hiện (không phải là phương thức tĩnh), tham chiếu **this** tự động trỏ đến đối tượng này. Mọi phương thức của đối tượng đều có thể tham chiếu đến các thành phần của đối tượng thông qua tham chiếu **this**. Có 3 trường hợp sử dụng tham chiếu **this**:

- Tránh xung đột tên khi tham số của phương thức trùng tên với tên biến dữ liệu của đối tượng.
- Dùng để truyền đối tượng hiện tại làm tham số cho một phương thức khác (chẳng hạn gọi đệ qui)
- Dùng với mục đích chỉ mục.

# THAM CHIẾU this

Ví dụ 8: Dùng tham chiếu **this** với mục đích tránh xung đột tên của tham số với tên biến dữ liệu của đối tượng.

```
class People
{
    int old; string name; double height;
    public People(int old, string name, double height)
    {
        Console.WriteLine("\n---Goi ham xay dung co 3
tham so---");
        this.old = old;
        this.name = name;
        this.height = height;
     }
```

# THAM CHIẾU this

**Ví dụ 8:** Xây dựng lớp People quản lý sức khỏe

```
class People
{
    int old; string name; double height;
    public People(string name, int old, double height)
    {
       this.name = name;
       this.old = old;
       this.height = height;
     }
     public void Show()
        {
         Console.WriteLine("Name: " + name + "\nOld: " +
old + "\nHeight: " + height);

        }
    }
```

Gọi hàm xây dựng với 3 tham số. Dùng tham chiếu **this** với mục đích tránh xung đột tên của tham số với tên biến dữ liệu của đối tượng

# THAM CHIẾU this

**Ví dụ 8:** Xây dựng bài toán quản lý sức khỏe con người với các thuộc tính: Name, Old, Height
Xuất ra màn hình thông tin của người đó.

```
class Program
{
    static void Main(string[] args)
    {
        People p1 = new People("Nguyen Van A",20,180);
        p1.Show();
     }
}
```

```
Name: Nguyen Van A
Old: 20
Height: 180
```

# CHỈ MỤC (INDEXER)

Việc định nghĩa chỉ mục cho phép tạo các đối tượng của lớp hoạt động giống như một mảng ảo. Tức là các đối tượng có thể sử dụng toán tử [] để truy cập đến một thành phần dữ liệu nào đó. Việc định nghĩa chỉ mục tương tự như việc định nghĩa một thuộc tính.

**Cú pháp** tạo chỉ mục:

```
public KiểuTraVề this[DanhSáchThamSố]
{
get
{//thân hàm đọc}
set
{//thân hàm ghi}
}
```

# CHỈ MỤC (INDEXER)

**Ví dụ 9:** Xây dựng lớp `IndexedNames` và dùng chỉ mục để truy cập trực tiếp đến các phần tử của `IndexedNames`

```
using System;
namespace C3Vd9
{
    class IndexedNames
    {
     //khai báo một mảng chuỗi
      private string[] namelist = new string[size];
      static public int size = 10;
      public IndexedNames()
      {
          for (int i = 0; i < size; i++)
          namelist[i] = "None";
      }
```

# CHỈ MỤC (INDEXER)

```
public string this[int index]
    {
    get
    {   string tmp;
        if( index >= 0 && index <= size-1 ){
             tmp = namelist[index];
         }
        else{
             tmp = "";
         }
        return ( tmp );
    }
    set
    {
        if( index >= 0 && index <= size-1 ){
            namelist[index] = value;
        }
    }
}
```

# CHỈ MỤC (INDEXER)

```
static void Main(string[] args)
    {
        IndexedNames names = new IndexedNames();
        names[0] = "Nguyen Van An";
        names[1] = "Tran Van Bao";
        names[2] = "Le Van Chinh";
        names[3] = "Ngo Van Danh";
        names[4] = "Nguyen Thi Hoa";
        names[5] = "Ho Truc Linh";
        names[6] = "Pham Ha Trang";
        for ( int i = 0; i < IndexedNames.size; i++ )
        {
            Console.WriteLine(names[i]);
        }
    }
  }
}
```

# CHỈ MỤC (INDEXER)

Khi biên dịch và thực thi chương trình, tạo ra các kết quả như sau:

```
Nguyen Van An
Tran Van Bao
Le Van Chinh
Ngo Van Danh
Nguyen Thi Hoa
Ho Truc Linh
Pham Ha Trang
None
None
None
```

# CHỈ MỤC (INDEXER)

***Chú ý:***

• Một ***chỉ mục*** phải có ít nhất một tham số, và tham số có thể có kiểu bất kỳ.

• ***chỉ mục*** có thể chỉ có phương thức ***get***.

• Mặc dù ***chỉ mục*** là một đặc điểm thú vị của **C#** nhưng cần phải sử dụng đúng mục đích (sử dụng để đối tượng có thể hoạt động như mảng, mảng nhiều chiều).

• Một lớp có thể có nhiều ***chỉ mục*** nhưng chúng phải có các dấu hiệu phân biệt với nhau (tương tự như quá tải phương thức).

# BÀI TẬP KẾT THÚC CHƯƠNG 4

❑ **Bài 1:** Xây dựng lớp Phanso (Phân số) gồm:

- Thuộc tính: Tuso, Mauso (Tử số, Mẫu số)
- Phương thức:
  - Hàm Khởi tạo không Tham số, Hàm hủy
  - Hàm Nhập, xuất
  - Hàm Cong(), Tru(), Nhan(),Chia()

Tính Tổng, Hiệu, Tích, Thương 2 phân số A và B rồi in ra kết quả trên màn hình.

# BÀI TẬP KẾT THÚC CHƯƠNG 4

❑ **Bài 2:** Xây dựng lớp SoPhuc (Số phức) gồm:

- Thuộc tính: PhanThuc, PhanAo (Phần thực, Phần ảo)
- Phương thức:
  - Hàm Khởi tạo không Tham số, Hàm hủy
  - Hàm Nhập, xuất
  - Hàm Cong(), Tru(), Nhan(),Chia()

Tính Tổng, Hiệu, Tích, Thương 2 số phức A và B rồi in ra kết quả trên màn hình.

# BÀI TẬP KẾT THÚC CHƯƠNG 4

❑ **Bài 3:** Xây dựng lớp Tamgiac (Tam giác) gồm:

- Thuộc tính: Cạnh a, Cạnh b, Cạnh c
- Phương thức:
  - Hàm Khởi tạo không Tham số, Hàm hủy
  - Hàm Nhập, xuất
  - Hàm Kiemtra()

Xuất ra màn hình kiểu của tam giác (Tam giác thường, tam giác vuông, tam giác cân, tam giác vuông cân, tam giác đều)

# BÀI TẬP KẾT THÚC CHƯƠNG 4

❑ **Bài 4:** Xây dựng một ứng dụng quản lý điểm học phần OOP của sinh viên có chứa nội dung như sau:

- Lớp thông tin sinh viên (**Info**) bao gồm: **ID, Hoten, Group** lần lượt là Mã Sinh viên, Họ Tên sinh viên, Lớp sinh hoạt
- Lớp điểm thành phần **(DiemTP)** bao gồm **TP1, TP2, TP3**

**Các phương thức yêu cầu trong bài:**

- Nhập mã sinh viên, Họ và tên sinh viên và Lớp SH
- Xuất số lượng sinh viên có trong lớp học phần OOP
- Nhập điểm TP1, TP2, TP3 cho mỗi sinh viên
- Tính điểm TB = TP1*0.1+TP2*0.3+TP3*0.6
- Xuất ra thông tin của sinh viên và điểm trung bình môn OOP

# BÀI TẬP KẾT THÚC CHƯƠNG 4

**Bài 5:** Xây dựng lớp **Doanhnghiep** gồm các thuộc tính: **TenDN, MST, Diachi** lần lượt là Tên doanh nghiệp, Mã số thuế, Địa chỉ của doanh nghiệp đó.

Từ đó xây dựng lớp **DanhsachDN (Danh sách doanh nghiệp)** với các phương thức:

- Nhập danh sách doanh nghiệp
- Xuất danh sách doanh nghiệp
- Tìm mã số thuế theo tên doanh nghiệp (chỉ mục )
- Tìm tên doanh nghiệp và địa chỉ doanh nghiệp theo mã số thuế (chỉ mục )