



TRƯỜNG ĐẠI HỌC MỞ
TP HỒ CHÍ MINH

BÀI GIẢNG
LẬP TRÌNH GIAO DIỆN

**CHƯƠNG 3:
HƯỚNG ĐỐI TƯỢNG TRONG C#**



Nội dung chương

- Giới thiệu về lập trình hướng đối tượng.
- Lớp
- Phương thức
- Thuộc tính
- Truyền tham số
- Thừa kế



Giới thiệu lập trình hướng đối tượng

- Mục tiêu của thiết kế phần mềm
 - Tính tái sử dụng
 - Tính mở rộng
 - Tính mềm dẻo



Lập trình hướng thủ tục

- Chương trình là một hệ thống các thủ tục, hàm.
- Cần xác định các hàm, định nghĩa hàm, gọi hàm.
- Nhược điểm:
 - Mọi hàm đều có thể truy cập biến toàn cục.
 - Khó kiểm soát các hàm khi sửa đổi chương trình.
 - Phải quản lý danh sách các hàm



Lập trình hướng đối tượng

- Tiếp cận HĐT giúp khắc phục khuyết điểm của lập trình hướng thủ tục:
 - Không sử dụng lại được mã nguồn.
 - Mọi thay đổi cấu trúc đòi hỏi phải thay đổi cả giải thuật
 - Chỉ phát huy hiệu quả trong module chương trình nhỏ.
- Lập trình HĐT nhìn nhận và phân tích chương trình là những hoạt động mà các đối tượng tham gia vào chương trình đó.



Lập trình hướng đối tượng

- **Lấy đối tượng làm nền tảng.**
- **Đối tượng = Dữ liệu + Phương thức**
- **Một đối tượng là thực thể trong hệ thống.**



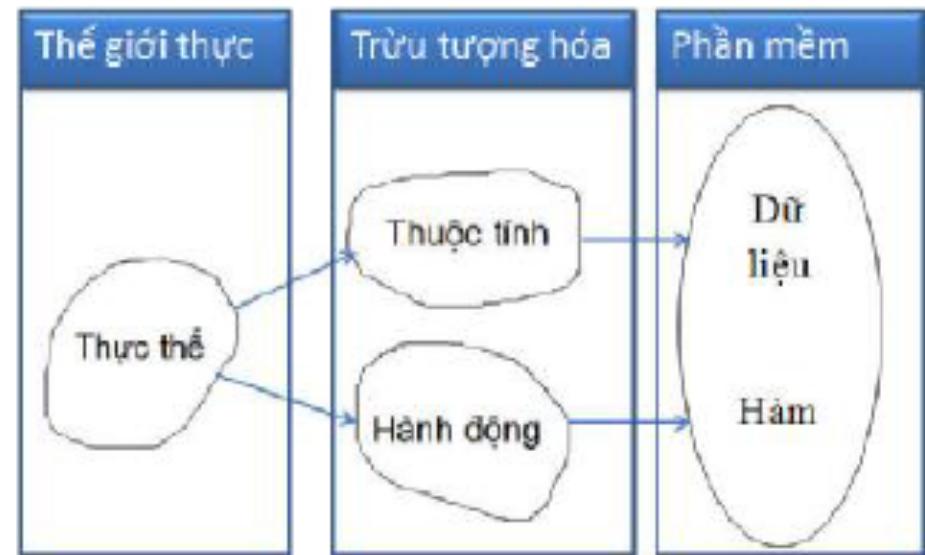
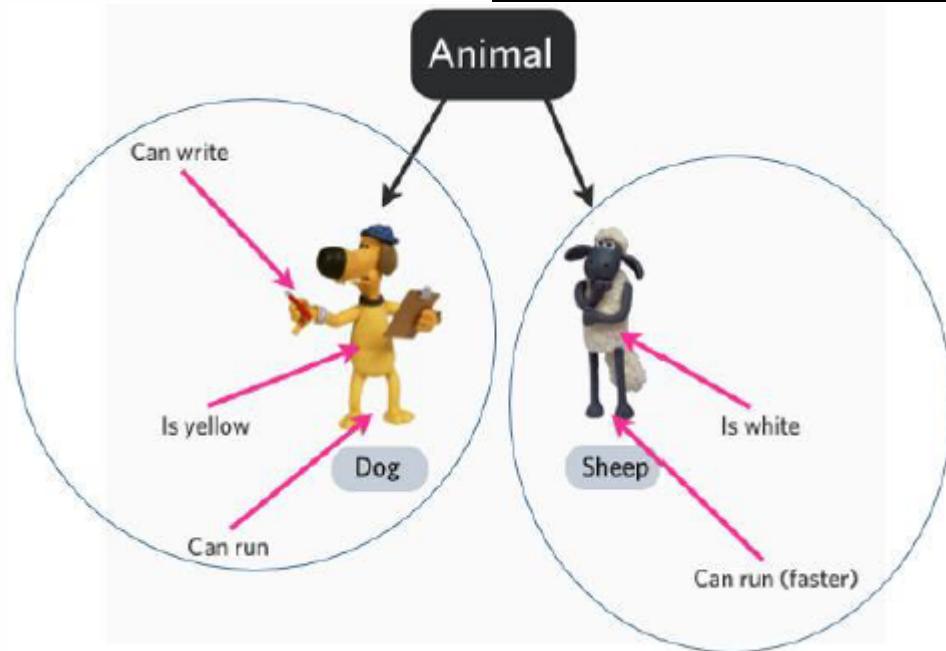


Lập trình hướng đối tượng

- Thiết kế lớp đối tượng thỏa các chức năng sau:
 - Lớp đối tượng liên quan.
 - Dữ liệu và thao tác liên quan.
 - Quan hệ giữa các đối tượng



Lập trình hướng đối tượng





Đặc điểm lập trình hướng đối tượng

- Tính đóng gói
- Tính kế thừa
- Tính đa hình



Đặc điểm lập trình hướng đối tượng

- Tính đóng gói

- Ràng buộc dữ liệu và phương thức vào các lớp riêng biệt.
- Chỉ cung cấp một phương thức để giao tiếp



Đặc điểm lập trình hướng đối tượng

- Tính kế thừa

- Định nghĩa lớp đối tượng dựa trên các lớp khác.
- Lớp mới chỉ cần bổ sung thêm các thành phần riêng



Đặc điểm lập trình hướng đối tượng

- Tính đa hình

- Sử dụng một giao diện chung cho nhiều phương thức khác nhau.
- Phương thức cụ thể sẽ được xác định vào lúc chạy chương trình.



LTHDT - Một số khái niệm

- Lớp (class) là khái niệm trừu tượng phản ánh tập hợp các đối tượng có cùng tính chất.
- Một đối tượng là một thể hiện của một lớp.
- Thuộc tính: thành phần dữ liệu mô tả đối tượng.
- Phương thức: mô tả và thực hiện các hành vi của đối tượng

Class <tên lớp>

{

Các biến thành viên;

Hàm xây dựng;

Các thuộc tính;

Các phương thức;

}

Cú pháp:

<phạm vi truy cập> <kiểu> <tên biến>;

class ConNguoi()

{

 public string hoTen;

 protected string gioiTinh;

 DateTime ngaySinh;

}



Phạm vi truy cập

- ***private***: Chỉ truy cập được từ trong lớp khai báo.
- ***protected***: Truy cập được từ trong lớp khai báo và các lớp con của lớp khai báo.
- ***public***: Truy cập được từ mọi nơi.
- Mặc định là ***private***

- Cách sử dụng lớp:

- Tạo đối tượng của lớp:

- Tenlop TenBienDoituong;

TenBienDoituong = **new** Tenlop ([DanhSachDoiSo]);

- hoặc

Tenlop TenBienDoituong = **new** Tenlop([DanhSachDoiSo]);

- Sử dụng đối tượng đã tạo để truy xuất các thành phần của lớp:

- TenBienDoituong.TenBien

TenBienDoituong.TenPhuongthuc ([DanhSachDoiSo])

- Ví dụ:

```
class Hinhtron
{
    private double bankinh = 10;
    public double Tinhdientich ()
    {
        return bankinh*bankinh*Math.PI;
    }
}
```

- Sử dụng đối tượng của class Hinhtron

```
Hinhtron ht = new Hinhtron ();
double dientich = ht.Tinhdientich ();
```



Phương thức

- Phương thức của một lớp thường được dùng để mô tả và thực hiện các hành vi của đối tượng lớp.
- Mỗi phương thức thường được định nghĩa là một hàm.



Phương thức (tt)

- Phương thức có trả về giá trị:
 - Trước tên phương thức phải chỉ ra kiểu dữ liệu
 - Cuối phương thức phải có lệnh return để trả về giá trị đúng với kiểu dữ liệu đã khai báo.
 - [MucTruyCap] kieu_dulieu TenPhuongthuc ([CacThamSo])
{
 //code định nghĩa bên trong phương thức
 //return value;
}
 - Ví dụ:
 - public int Tonghaiso (int a, int b)
{
 return (a + b);
}



Phương thức (tt)

- Phương thức không có trả về giá trị:
 - Trước tên phương thức có từ khóa void
 - Cuối phương thức không cần phải có lệnh return (hoặc dùng return; để thoát khỏi phương thức)
 - [MụcTruyCap] **void** TenPhuongthuc ([DanhSachThamSo])
{
 //code định nghĩa bên trong phương thức
}
 - Ví dụ:
 - **public void** InTonghaiso (**int** a, **int** b)
{
 Console.WriteLine("Tong cua {0} va {1} la {2}", a, b, a+b);
}



Phương thức nạp chồng

- Là phương thức có cùng tên nhưng khác tham số
- Khi chương trình thực thi, tùy thuộc vào lời gọi hàm và đối số truyền vào mà phương thức thích hợp nhất được gọi.



Phương thức nạp chồng – Ví dụ

```
public class Hinhtron
{
    private double bankinh;
    public double Tinhdientich()
    {
        return bankinh * bankinh * Math.PI;
    }

    public double Tinhdientich(double bk)
    {
        return bk * bk * Math.PI;
    }
}
```



Phương thức khởi tạo

- Là phương thức đặc biệt có tên cùng với tên của lớp, thường được khai báo với từ khóa public.
- Không có giá trị trả về.
- Được gọi một cách tự động khi một đối tượng của lớp được tạo ra.
- Nếu không định nghĩa phương thức khởi tạo, trình biên dịch sẽ tự động sử dụng phương thức khởi tạo mặc định.
- Các thuộc tính sẽ được khởi tạo mặc định



Phương thức khởi tạo – Ví dụ

```
public class Hinhtron
{
    private double bankinh;
    public Hinhtron()
    {
        bankinh = 0.0;
    }
    public double Tinhdientich()
    {
        return bankinh * bankinh * Math.PI;
    }
}
```



Nạp chồng phương thức khởi tạo

- Là xây dựng các phương thức khởi tạo khác tham số

```
public class Hinhtron
{
    private double bankinh;
    public Hinhtron()
    {
        bankinh = 0.0;
    }

    public Hinhtron(double bk)
    {
        bankinh = bk;
    }
}
```



Phương thức khởi tạo sao chép

- Khởi gán giá trị cho đối tượng mới bằng cách sao chép dữ liệu của đối tượng cùng kiểu đã tồn tại.

```
public class Hinhtron
{
    private double bankinh;
    public Hinhtron()
    {
        bankinh = 0.0;
    }

    public Hinhtron(Hinhtron h)
    {
        bankinh = h.bankinh; ;
    }
}
```

- Properties là một đặc tính mới của ngôn ngữ C#.
- Cho phép lấy giá trị (get) và gán giá trị (set) cho các thành phần dữ liệu của lớp.
- Cho phép truy cập đến các thành phần dữ liệu của đối tượng ở mức đọc, ghi hoặc cả hai và che dấu cài đặt thực sự bên trong lớp.
- Thuộc tính chỉ đọc: **get**

- Thuộc tính chỉ đọc: **get**

```
public class Hinhtron  
{  
    private double bankinh;  
    public double Bankinh  
    {  
        get { return bankinh; }  
    }  
}
```

Cách sử dụng:

sai ----->

```
Hinhtron ht =new Hinhtron ();
```

```
ht.Bankinh = 10;
```

đúng ----->

```
int bk = ht.Bankinh;
```

- Thuộc tính đọc và ghi: **get, set**

```
public class Hinhtron
{
    private double bankinh;
    public double Bankinh
    {
        get { return bankinh; }
        set { bankinh = value; }
    }
}
```

Cách sử dụng:

```
Hinhtron ht =new Hinhtron ();
ht.Bankinh = 10;
int bk = ht.Bankinh;
```

Tạo lớp phân số bao gồm:

Thành phần dữ liệu: Tử số, Mẫu số

Phương thức:

- Phương thức khởi tạo chuẩn
- Phương thức khởi tạo 2 tham số.
- Cộng 2 phân số
- Trừ 2 phân số

Tạo lớp Xử Lý

- Phương thức main.
- Xử lý lỗi

■ Hướng dẫn:

```
//phương thức khởi tạo chuẩn, không tham số  
//mặc định tử=0 và mẫu=1  
public PHANSO() {  
    tu=0; mau=1;  
}
```

```
//phương thức khởi tạo 2 tham số  
public PHANSO(int t, int m ){  
    tu=t; mau=m;  
}
```

- Gán thuộc tính đọc và ghi:

```
public int TuSo
{
    get { return tuSo; }
    set { tuSo = value; }
}
public int MauSo
{
    get { return mauSo; }
    set { mauSo = value; }
}
```

■ Phương thức Cộng phân số:

```
public PhanSo Cong (PhanSo p)
{
    PhanSo kq = new PhanSo();
    kq.tuSo = tuSo * p.mauSo + p.tuSo * mauSo;
    kq.mauSo = mauSo * p.mauSo;
    return kq;
}
```

■ Phương thức main:

```
static void Main(string[] args)
{
    PhanSo ps1, ps2, tong, hieu;
    ps1 = new PhanSo();
    ps2 = new PhanSo();
    tong = hieu = new PhanSo();
    try
    {
        ps1.TuSo = int.Parse(Console.ReadLine());
        ps1.MauSo = int.Parse(Console.ReadLine());
        ps2.TuSo = int.Parse(Console.ReadLine());
        ps2.MauSo = int.Parse(Console.ReadLine());
        tong = ps1.Cong(ps2);
        Console.WriteLine("Tổng của hai phân số {0}/{1} và {2}/{3} là: {4}/{5}",
            ps1.TuSo.ToString(), ps1.MauSo.ToString(), ps2.TuSo.ToString(), ps2.MauSo.ToString(),
            tong.TuSo.ToString(), tong.MauSo.ToString());
    }
    catch (FormatException)
    {
        Console.WriteLine("Vui lòng nhập giá trị đúng");
    }
}
```



Tham chiếu this

Tham chiếu this :

- Dùng để tham chiếu đến thể hiện hiện hành của một đối tượng.
- Tham chiếu đến các thành phần của đối tượng trong phương thức.



Mục đích của tham chiếu **this**

- Tránh xung đột tên khi tham số trùng tên biến.
- Truyền đối tượng hiện tại làm tham số cho phương thức khác.
- Làm chỉ mục





Mục đích của tham chiếu **this**

- Tránh xung đột tên khi tham số trùng tên biến.

```
public class Hinhtron
{
    private double bankinh;
    public void Ban_kinh(double bankinh)
    {
        this.bankinh = bankinh;
    }
}
```



Mục đích của tham chiếu **this**

Làm tham
số cho
phương
thức

```
class Circle
{
    double radius;
    public double Radius
    {
        get { return radius; }
        set {radius = value;}
    }
    public void TestCopy()
    {
        Test bObj = new Test ();
        bObj.Copy(this);
    }
}
class Test
{
    public Circle Copy(Circle circle)
    {
        Circle c = new Circle();
        c.Radius = circle.Radius ;
        return c;
    }
}
```



Mục đích của tham chiếu **this**

Làm
chỉ
mục

```
class Mang
{
    int [] mang;
    int sopt;

    public Mang(int n)
    {
        sopt = n;
        mang = new int [sopt];
    }

    public int this [int i]
    {
        get
        {
            if (i >= 0 && i < sopt)
                return mang [i];
            else return 0;
        }

        set
        {
            if (i >= 0 && i < sopt)
                mang [i] = value;
        }
    }
}
```

Sử dụng:

```
Mang a = new Mang(10);
a [0] = 1;
a [1] = 5;
int x = a [0];
```





Dữ liệu và phương thức tĩnh

- **static:** khai báo thành viên tĩnh.
- Được dùng khi cần lưu các giá trị biến hoặc cung cấp các phương thức dùng chung.
- Dữ liệu và phương thức tĩnh:
 - Có phạm vi toàn cục trong một lớp
 - Có thể truy cập mọi đối tượng trong lớp.

- Cú pháp truy cập.

```
classname . variableName;
```

```
classname . methodName;
```

- Ví dụ

```
Math.PI;
```

```
String.Compare(str1, str2);
```



Cách truyền tham số

- Truyền tham trị.
- Truyền tham chiếu
 - **ref:** dữ liệu trước khi truyền vào phải khai báo giá trị tường minh.
 - **out:** dữ liệu truyền vào không cần phải khai báo trước. Cần chỉ định biến trước khi thay đổi.

- Truyền tham chiếu: **ref**

```
public void GetTime ( ref int h, ref int m, ref int s)
{
    h = DateTime.Now.Hour;
    m = DateTime.Now.Minute;
    s = DateTime.Now.Second;
}

private void GetCurrentTime()
{
    int hour = 0;
    int minute = 0;
    int second = 0;
    GetTime(ref hour, ref minute, ref second);
}
```



Cách truyền tham số

■ Truyền tham chiếu: **out**

```
public void GetTime ( out int h, out int m, out int s)
{
    h = DateTime.Now.Hour;
    m = DateTime.Now.Minute;
    s = DateTime.Now.Second;
}
private void GetCurrentTime()
{
    int hour ;
    int minute;
    int second ;
    GetTime(out hour, out minute, out second);
}
```

- Tính thừa kế giúp có thể tái sử dụng lại mã nguồn.
- Một lớp mới (lớp dẫn xuất) được được xây dựng dựa trên một lớp có sẵn (lớp cơ sở), sau đó thêm các thuộc tính và phương thức riêng của lớp đó.
- Lớp dẫn xuất có thể sử dụng và truy cập những biến thành viên, thuộc tính và phương thức có thuộc tính truy cập là protected hoặc public của lớp cha.

- Lớp kế thừa dùng để:
 - Phản ánh mối quan hệ giữa các lớp.
 - Phản ánh sự chia sẻ mã lệnh chương trình giữa các lớp



Ví dụ



Giống nhau		
Mô tả	Có chân Có mắt	
Hành động	Biết ăn Biết ngủ	
Khác nhau		
	Chó	Chim
Mô tả	Có răng	Có mỏ
Hành động	Biết sữa Biết chạy	Biết hót Biết bay





Ví dụ

Lớp Động vật	
Mô tả	Hành động
Có chân	Biết ăn
Có mắt	Biết ngủ



Lớp Chó	
Mô tả	Hành động
Có răng	Biết sữa Biết chạy

Lớp Chim	
Mô tả	Hành động
Có mỏ	Biết hót Biết bay

- Định nghĩa lớp dẫn xuất:

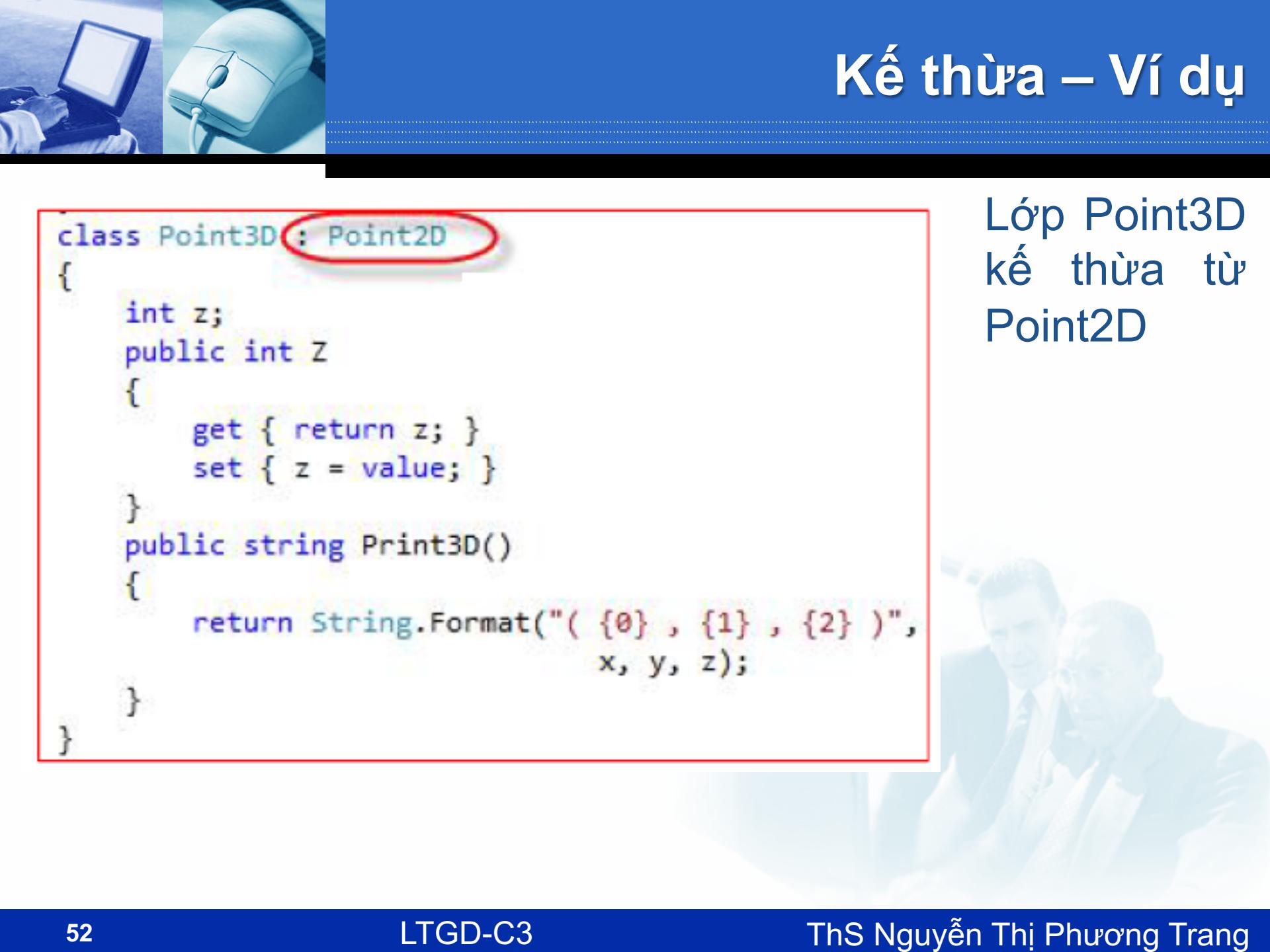
```
class Tenlopdanxuat : Tenlopcoso  
{  
    // nội dung  
}
```



Kế thừa – Ví dụ

```
class Point2D
{
    protected int x, y;
    public int X
    {
        get { return x; }
        set { x = value; }
    }
    public int Y
    {
        get { return y; }
        set { y = value; }
    }
    public string Print2D()
    {
        return String.Format("( {0} , {1} )", x, y);
    }
}
```

Lớp Point2D



Kế thừa – Ví dụ

```
class Point3D : Point2D
{
    int z;
    public int Z
    {
        get { return z; }
        set { z = value; }
    }
    public string Print3D()
    {
        return String.Format("( {0} , {1} , {2} )",
                             X, Y, Z);
    }
}
```

Lớp Point3D
kế thừa từ
Point2D



Kế thừa – Ví dụ

```
Point2D p2 = new Point2D();
p2.X = 100;
p2.Y = 50;
lbPoint2D.Text = p2.Print2D();
```

Sử dụng

```
Point3D p3 = new Point3D();
p3.X = 50;
p3.Y = 100;
p3.Z = 80;
lbPoint3D.Text = p3.Print3D();
lbPoint2D.Text = p3.Print2D();
```



Kế thừa – phương thức khởi tạo

- Lớp dẫn xuất không thể kế thừa phương thức khởi tạo của lớp cơ sở.
- Phụ thuộc vào khai báo phương thức khởi tạo của lớp cơ sở.



Kế thừa – phương thức khởi tạo

- Cú pháp khai báo phương thức khởi tạo có tham số:

```
TênLớpDẫnXuất (ThamSốLớpDẫnXuất): base (ThamSốLớpCơSở)
{
    // Khởi tạo giá trị cho các thành phần của lớp dẫn xuất
}
```

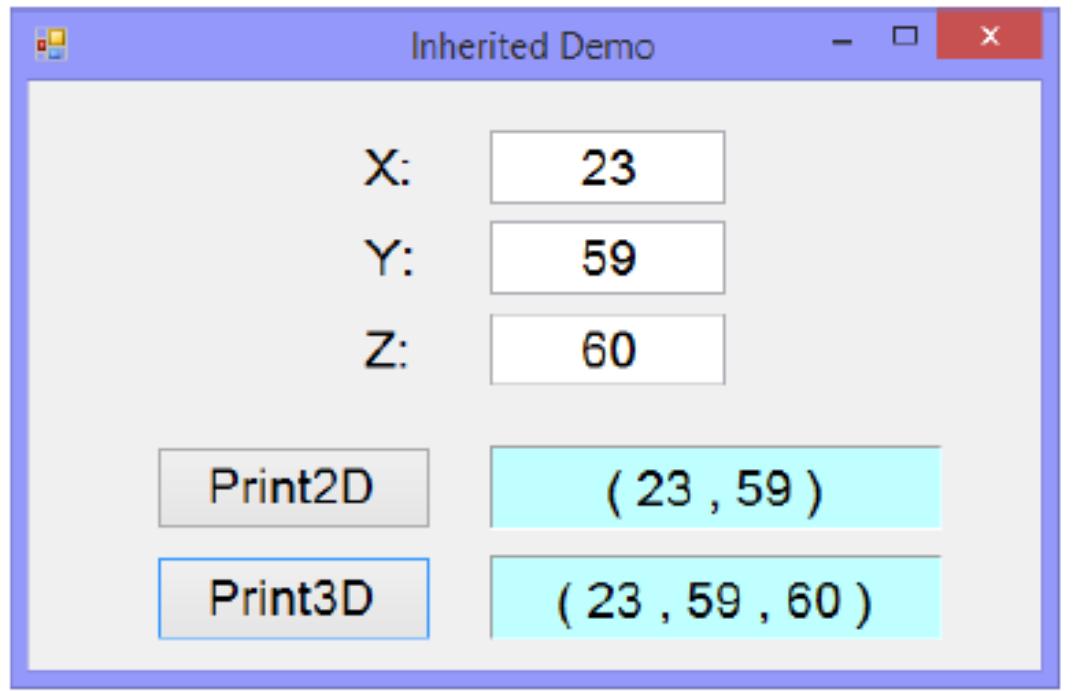
- Ví dụ:

```
public Point2D(int x, int y)    public Point3D(int x, int y, int z)
{
    this.x = x;                      : base(x, y)
    this.y = y;                      {
}                                         this.z = z;
}                                         }
```



Kế thừa – Ví dụ

- Xây dựng ứng dụng như sau: Nhập vào ba giá trị x,y,z là giá trị tọa độ điểm. Và hiển thị tọa độ 2D và 3D như hình.





Kế thừa – Ví dụ

```
class Point2D
{
    protected int x, y;
    public int X
    {
        get { return x; }
        set { x = value; }
    }
    public int Y
    {
        get { return y; }
        set { y = value; }
    }
    public Point2D()
    {
        this.x = 0;
        this.y = 0;
    }
    public Point2D(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
    public string Print2D()
    {
        return String.Format("( {0} , {1} )", x, y);
    }
}

class Point3D : Point2D
{
    int z;
    public int Z
    {
        get { return z; }
        set { z = value; }
    }
    public Point3D(int x, int y, int z)
        : base(x, y)
    {
        this.z = z;
    }
    public string Print3D()
    {
        return String.Format("( {0} , {1} , {2} )",
                             x, y, z);
    }
}
```



Kế thừa – Ví dụ

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btPrint2D_Click(object sender, EventArgs e)
    {
        Point2D p2 = new Point2D();
        p2.X = int.Parse(txtX.Text);
        p2.Y = int.Parse(txtY.Text);
        lbPoint2D.Text = p2.Print2D();
    }

    private void btPrint3D_Click(object sender, EventArgs e)
    {
        Point3D p3= new Point3D(int.Parse(txtX.Text),
                               int.Parse(txtY.Text),
                               int.Parse(txtZ.Text));
        lbPoint3D.Text = p3.Print3D();
        lbPoint2D.Text = p3.Print2D();
    }
}
```



Xây dựng lớp học sinh, biết rằng mỗi học sinh có:

- Thành phần dữ liệu: mã số, họ tên, điểm trung bình
- Phương thức: set(), get(), input(), output(), rank()
- Xếp loại cho học sinh theo dtb.

Viết lớp Demo1 chứa phương thức main():

- Tạo một đối tượng học sinh
- Nhập thông tin cho học sinh
- Xuất thông tin cùng xếp loại của học sinh.
- Đổi tên của học sinh thành một tên mới được nhập từ bàn phím
- Nhập thêm một học sinh. Cho biết tên của học sinh có điểm trung bình lớn hơn