



# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GV: Nguyễn Thanh Tùng  
Mail: [tung.nguyenthanh@stu.edu.vn](mailto:tung.nguyenthanh@stu.edu.vn)



# NỘI DUNG CHÍNH



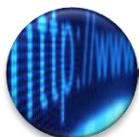
Giới thiệu lập trình hướng đối tượng



Các khái niệm cơ bản của ngôn ngữ C#



Lớp và đối tượng



Xây dựng ứng dụng trên WinForm



Tính thừa kế và tính đa hình

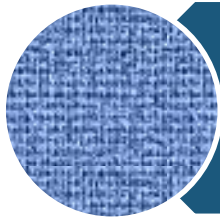


Liên hệ giữa các lớp đối tượng

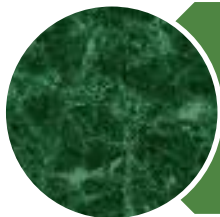
# Bài 1: GIỚI THIỆU LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG



Lập trình hướng thủ tục



Lập trình hướng đối tượng



So sánh giữa lập trình hướng thủ tục và lập trình hướng đối tượng

# I. Lập trình hướng thủ tục

## 1/ Đặc điểm:

Chương trình = Cấu trúc dữ liệu + Giải thuật

## 2/ Ưu điểm:

- + Chương trình dễ đọc, dễ hiểu
- + Tư duy rõ ràng.

## 3/ Nhược điểm:

- + Không phù hợp với việc xây dựng các chương trình lớn
- + Không hỗ trợ việc tái sử dụng mã nguồn.

## II. Lập trình hướng đối tượng

1/ Đặc điểm:

Chương trình = { tập các đối tượng }

2/ Lập trình hướng đối tượng có đầy đủ các tính chất của lập trình cấu trúc và có thêm các tính chất sau:

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

3/ Lập trình hướng đối tượng đã khắc phục những nhược điểm đã có của lập trình cấu trúc.

### III. So sánh

Lập trình hướng đối tượng	Lập trình cấu trúc
+ Áp dụng cho những bài toán lớn và có nhiều luồng dữ liệu phức tạp khác nhau.	+ Áp dụng cho những bài toán nhỏ và có các luồng dữ liệu rõ ràng.
+ Dữ liệu bị hạn chế truy cập trực tiếp từ bên ngoài.	+ Dữ liệu được truy cập tự do nên mất an toàn dữ liệu.
+ Tiếp cận bài toán dựa vào mô hình “Bottom-Up”	+ Tiếp cận bài toán dựa vào mô hình “Top-Down”

## Bài 2: CÁC KHÁI NIỆM CƠ BẢN CỦA NGÔN NGỮ C#



Tạo ứng dụng dạng Console



Kiểu dữ liệu, biến và hằng



Các phép toán và biểu thức



Truyền tham số sang hàm



Kiểu struct và kiểu DateTime



Kiểu chuỗi và kiểu mảng





# I. Tạo ứng dụng dạng Console

---

1/ Khởi động phần mềm Microsoft Visual Studio 2010.

2/ Chọn menu File/ New/ Projects  
-> chọn mục Console Application.

3/ Chương trình chính của ứng dụng:

```
class Program
{
    static void Main(string[] args)
    {
    }
}
```





## II. Kiểu dữ liệu, biến và hằng

1/ Kiểu dữ liệu cơ bản:

- + Kiểu ký tự: char
- + Kiểu số nguyên: int, long, ...
- + Kiểu số thực: float, double, ...
- + Kiểu luận lý: bool

2/ Biến: khai báo và sử dụng biến giống như C++

Ví dụ: `int a; //khai báo biến a`

3/ Hằng: khai báo và sử dụng hằng giống C++

Ví dụ: `const int MAX=100;`



### III. Các phép toán và biểu thức

Tương tự như C++

1/ Phép toán số học: +, -, \*, /, %

2/ Phép toán quan hệ: >, <, >=, <=, ==, !=

3/ Phép toán luận lý: !, &&, ||

4/ Phép gán: =



## IV. Truyền tham số sang hàm

---

1/ Viết hàm tương tự C++

2/ Truyền tham số sang hàm:

- + Truyền tham trị:
- + Truyền tham số có sử dụng từ khóa `ref`
- + Truyền tham số có sử dụng từ khóa `out`



## V. Kiểu struct và kiểu DateTime

1/ Kiểu struct: tương tự như C++

**Ví dụ:** Kiểu PHANSO chứa một phân số và có hàm tăng phân số lên 1 đơn vị.

```
struct PHANSO
{
    public int tuso;
    public int mauso;
    public PHANSO tangmot()
    {
        PHANSO b;
        b.mauso = mauso;
        b.tuso = tuso + mauso;
        return b;
    }
}
```

2/ Kiểu **DateTime**: được xây dựng từ kiểu struct và dùng lưu trữ dữ liệu dạng ngày và giờ. *(tra cứu trong thư viện MSDN)*



## VI. Kiểu chuỗi và kiểu mảng

### 1/ Kiểu chuỗi: *string*

- + Kiểu String là kiểu được xây dựng từ kiểu class.  
(tra cứu thư viện MSDN)
- + Lưu trữ một dãy ký tự Unicode.

### 2/ Kiểu mảng: *array*

- + Kiểu Array là kiểu được xây dựng từ kiểu class.  
(tra cứu thư viện MSDN)
- + Mảng một chiều:

```
int[] a = new int[5] { 12, -7, 18, 0, -3 };
```

- + Mảng hai chiều:

```
int[,] x = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } };
```



## VII. Kiểu enum

- + Kiểu enum là kiểu liệt kê.
- + Kiểu Enum là kiểu được xây dựng từ kiểu class. (*tra cứu thư viện MSDN*)

+Ví dụ:

```
enum KieuSoNghiem { VoNghiem, MotNghiem, VoSoNghiem };
static KieuSoNghiem giaiPTBacnhat(double a,double b,ref double x)
{
    if (a == 0)
        if (b == 0) return KieuSoNghiem.VoSoNghiem;
        else return KieuSoNghiem.VoNghiem;
    else
    {
        x = -b / a;
        return KieuSoNghiem.MotNghiem;
    }
}
```

# Bài 3: LỚP VÀ ĐỐI TƯỢNG



Khái niệm đối tượng và lớp đối tượng



Xây dựng lớp đối tượng



Biểu diễn đối tượng và lớp đối tượng bằng UML



Dữ liệu và phương thức static



Định nghĩa các phép toán





## I. Khái niệm đối tượng và lớp đối tượng

- ❑ Đối tượng là một thực thể trong chương trình đang thực thi.
  
- ❑ Đối tượng có hai thành phần chính là:
  - dữ liệu (thuộc tính-attribute/property)
  - hành vi (phương thức-method).



# I. Khái niệm đối tượng và lớp đối tượng

- ❑ Ví dụ: Đối tượng số phức ( $a+i*b$ )
  - Dữ liệu:  $a$  (phần thực),  $b$  (phần ảo) là những số thực.
  - Hành vi: là các phép toán trên số phức như cộng, trừ, nhân, chia hai số phức và trả về modul của số phức.



# I. Khái niệm đối tượng và lớp đối tượng

---

- ❑ Trạng thái của đối tượng là tập hợp các giá trị hiện có của các thành phần dữ liệu của đối tượng.
- ❑ Lớp đối tượng là tập hợp các đối tượng có chung các thành phần dữ liệu và các phương thức.



## I. Khái niệm đối tượng và lớp đối tượng

- ❑ Tính đóng gói: Không cho phép người sử dụng đối tượng thay đổi trạng thái của đối tượng mà chỉ có những phương thức nội tại của đối tượng mới được phép thay đổi trạng thái của đối tượng.
- ❑ Các thành phần của đối tượng được che dấu bên trong đối tượng và chỉ có những thành phần mà đối tượng cho phép người sử dụng đối tượng truy cập đến thì mới truy cập được.



# I. Khái niệm đối tượng và lớp đối tượng

## ❑ Thuộc tính truy cập:

- Thuộc tính truy cập riêng (private): chỉ truy cập được khi ở bên trong lớp đối tượng và không truy cập được khi ở bên ngoài lớp đối tượng.
- Thuộc tính truy cập chung (public): truy cập được khi ở bên trong lớp đối tượng và cả khi ở bên ngoài lớp đối tượng.



## II. Xây dựng lớp đối tượng

### 1) Khai báo lớp đối tượng:

```
class <tên lớp>  
{  
    <các thành phần của lớp>  
}
```



## II. Xây dựng lớp đối tượng

### Ví dụ: lớp số phức

```
class CSoPhuc
{
    //Các thành phần dữ liệu
    private double m_thuc;
    private double m_ao;
    //Các phương thức
    public void setSophuc(double thuc, double ao)...
    public double getThuc()...
    public double getAo()...
    //Các properties
    public double PhanThuc...
    public double PhanAo...
}
```







## II. Xây dựng lớp đối tượng

2) Tạo đối tượng: sử dụng từ khóa **new**

Ví dụ: tạo đối tượng **a** thuộc lớp **CSoPhuc**

```
CSoPhuc a = new CSoPhuc ();
```

3) Truy cập các thành phần của đối tượng:

**<tên đối tượng>.<tên thành phần của đối tượng>**

Ví dụ:

```
a.setSophec (2, 3);
```



## II. Xây dựng lớp đối tượng

### 4) Định nghĩa phương thức: giống như định nghĩa hàm

Ví dụ: Định nghĩa phương thức cập nhật thành phần dữ liệu của số phức.

```
public void setSophuc(double thuc, double ao)
{
    m_thuc = thuc;
    m_ao = ao;
}
```



## II. Xây dựng lớp đối tượng

### 4) Định nghĩa Property:

```
<thuộc tính truy cập> <kiểu property> <tên property>  
{  
    get {...}  
    set {...}  
}
```



## II. Xây dựng lớp đối tượng

Ví dụ: Định nghĩa Property cho phần thực của số phức.

```
public double PhanThuc
{
    get { return m_thuc; }
    set { m_thuc = value; }
}
```



## II. Xây dựng lớp đối tượng

### 5) Đối tượng `this`:

- ☐ Khi xây dựng một lớp đối tượng thì trong nó tồn tại sẵn một đối tượng và được gọi là đối tượng `this`.
- ☐ Ví dụ: định nghĩa phương thức cộng hai số phức.



## II. Xây dựng lớp đối tượng

### 6) Phương thức tạo lập (constructor):

- ☐ Phương thức tạo lập dùng để khởi tạo các giá trị ban đầu cho đối tượng như:
  - Cấp phát vùng nhớ cho các thành phần dữ liệu,
  - Khởi gán giá trị ban đầu cho đối tượng,
  - Thiết lập mối quan hệ giữa đối tượng với các đối tượng khác trong chương trình...



## II. Xây dựng lớp đối tượng

❑ Cách xây dựng phương thức tạo lập:

- Tên của phương thức tạo lập trùng với tên lớp,
- Phương thức tạo lập không có Kiểu trả về (kể cả kiểu void),
- Phương thức tạo lập có thể có tham số và cũng có thể không có tham số.

Ví dụ: định nghĩa phương thức tạo lập cho lớp số phức.





## II. Xây dựng lớp đối tượng

- ☐ Phương thức tạo lập được tự động gọi ngay sau khi đối tượng được tạo ra.



## II. Xây dựng lớp đối tượng

### 7) Phương thức hủy (Destructor):

- ☐ Phương thức hủy dùng “dọn dẹp” đối tượng trước khi đối tượng bị hủy.
- ☐ Phương thức hủy không có Kiểu trả về và cũng không có tham số. Tên của phương thức hủy trùng với tên lớp mà đặt thêm ký tự ‘~’ phía trước.
- ☐ Trong lớp đối tượng có nhiều nhất một phương thức hủy.

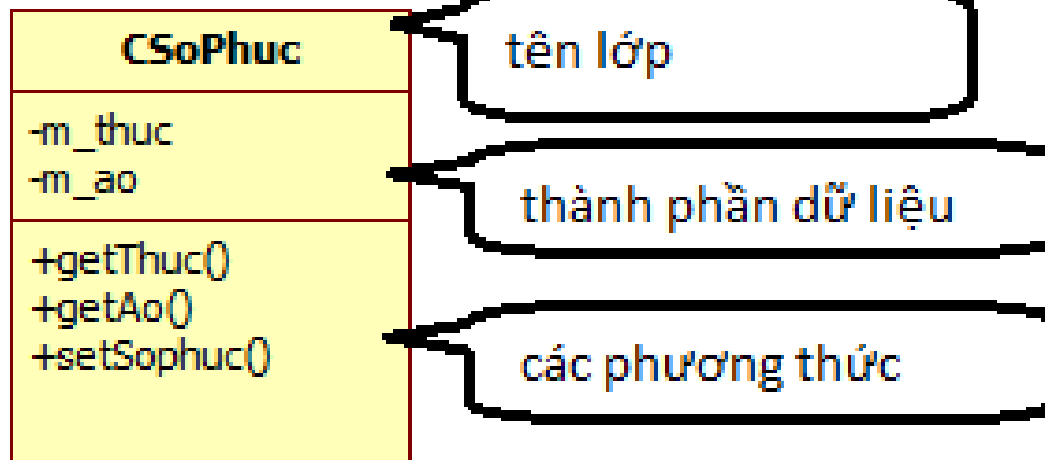


## II. Xây dựng lớp đối tượng

- ☐ Ngôn ngữ C# cung cấp cơ chế thu dọn rác tự động (Garbage Collection)
- ☐ Khi gọi phương thức **GC.Collect()** thì việc thu gom rác được thực hiện.

### III. Biểu diễn đối tượng và lớp đối tượng bằng UML

#### 1) Biểu diễn lớp đối tượng





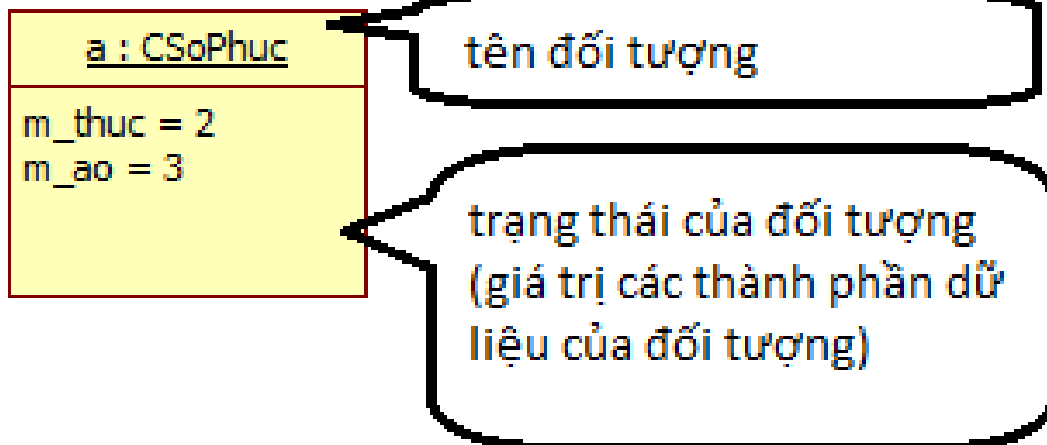
### III. Biểu diễn đối tượng và lớp đối tượng bằng UML

#### 2) Biểu diễn đối tượng

Ví dụ :

```
CSoPhuc a = new CSoPhuc () ;
```

```
a.setSophec (2 , 3) ;
```





## IV. Dữ liệu và phương thức static

- ❑ Dữ liệu static: là thành phần dữ liệu không thuộc vào bất kỳ đối tượng nào mà chỉ phụ thuộc vào lớp đối tượng.
- ❑ Phương thức static: là phương thức được sử dụng thao tác trên các thành phần dữ liệu static hay những thành phần dữ liệu không phải thành phần dữ liệu của đối tượng.



## IV. Dữ liệu và phương thức static

- ❑ Khai báo các thành phần static của lớp: đặt thêm từ khóa static trước dòng khai báo.
- ❑ Truy cập các thành phần static:

**<tên lớp>.<tên thành phần static>**







## IV. Dữ liệu và phương thức static

□ Ví dụ:

```
class CVidu_static
{
    private static int m_s = 0;
    public static void setVidu(int a)
    {
        m_s = a;
    }
    public static int getVidu()
    {
        return m_s;
    }
}

static void Main(string[] args)
{
    CVidu_static.setVidu(5);
    Console.WriteLine(CVidu_static.getVidu());
}
```



## V. Định nghĩa phép toán

- ❑ Trong C# cho phép người dùng định nghĩa các phép toán chồng lên những phép toán đã được xây dựng.
- ❑ Những phép toán xây dựng trong C# phải là thành phần static.



## V. Định nghĩa phép toán

Ví dụ: định nghĩa phép toán cộng hai số phức và sử dụng phép toán đó.

```
public static CSoPhuc operator +(CSoPhuc a, CSoPhuc b)
{
    CSoPhuc c = new CSoPhuc();
    c.m_thuc = a.m_thuc + b.m_thuc;
    c.m_ao = a.m_ao + b.m_ao;
    return c;
}
static void Main(string[] args)
{
    CSoPhuc a = new CSoPhuc(2, 3);
    CSoPhuc b = new CSoPhuc(1, -2);
    CSoPhuc c = a + b;
}
```

# Bài 4: XÂY DỰNG ỨNG DỤNG BẰNG WINFORM

Các thành phần cơ bản trong WinForm

Kiểu dữ liệu tập hợp

Ví dụ Quản lý học sinh

Thao tác trên file

Nắm bắt lỗi

Mẫu 2-Tiers



# I. Các thành phần cơ bản trên WinForm

## -Tạo ứng dụng đơn giản

1/ Khởi động C#: Chọn Start -> Programs -> Microsoft Visual Studio 2010

2/ Tạo một Ứng dụng mới:

Chọn menu File -> New -> Project.

**Ví dụ:** tạo một ứng dụng và trên cửa sổ của ứng dụng chứa control button “hello”. Khi ta nhấn nút “hello” thì xuất hộp thông báo chứa câu “chào bạn!”.



# I. Các thành phần cơ bản trên WinForm

## -Tạo ứng dụng đơn giản

---

a/ **Button control**: dùng để thực hiện một lệnh đơn giản bằng cách click vào nó.

Lớp button.

- + Sự kiện quan trọng: click
- + Property thường dùng: text.

b/ **Xuất hộp thông báo**: sử dụng phương thức tĩnh Show() của lớp MessageBox.

**Public static DialogResult show (...)**



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 1/ Label control

Mục đích: dùng để xuất dữ liệu.

Một đối tượng tương ứng với label control thuộc lớp label.

Lớp label có property thường dùng: text.

### 2/ Textbox control

Mục đích: dùng để nhập/xuất dữ liệu.

Một đối tượng tương ứng với textbox control thuộc lớp textbox.

Lớp textbox có property thường dùng: text.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 3/ Check Boxes

Mục đích: dùng để lựa chọn hay không lựa chọn một dữ liệu đã liệt kê.

Một đối tượng tương ứng với checkbox control thuộc lớp checkbox.

Lớp checkbox có property thường dùng: text và checked.

### 4/ Radiobutton Control

Mục đích: dùng để lựa chọn một trong các mục chọn dữ liệu đã liệt kê.

Một đối tượng tương ứng với radiobutton control thuộc lớp radiobutton.

Lớp textbox có property thường dùng: text và checked.





# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 5/ DateTimePicker Control

Mục đích: dùng để lựa chọn ngày và giờ.

Một đối tượng tương ứng với DateTimePicker Control thuộc lớp DateTimePicker.

Lớp DateTimePicker có property thường dùng: value. Một giá trị tương ứng với DateTimePicker Control thuộc kiểu DateTime.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 6/ ListBox Control

Mục đích: liệt kê một danh sách các phần tử có kiểu chuỗi.  
Tại một thời điểm ta có thể lựa chọn một hay nhiều phần tử trong danh sách.

Một đối tượng tương ứng với ListBox Control thuộc lớp ListBox.

Property Items: trả về tập hợp các phần tử trong list control.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 6/ ListBox Control

Property SelectionMode: xác định chế độ lựa chọn các phần tử trong ListBox (Chọn một, chọn nhiều và không cho chọn).

Property SelectedIndex: Trả về một số nguyên xác định chỉ số của phần tử được lựa chọn trong List Control.

Property SelectedItem: Trả về phần tử được lựa chọn trong List Control.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng các Controls

### 7/ ComboBox Control

Mục đích: là sự kết hợp của TextBox và ListBox. Phần lớn các thành phần cơ bản của TextBox và ListBox sử dụng được cho ComboBox.

Một đối tượng tương ứng với ComboBox Control thuộc lớp ComboBox.

Property DropDownStyle: xác định dạng hiển thị của ComboBox là: Simple, DropDown và DropDownList.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng menu

### MenuStrip Control

Mục đích: hiển thị một cây liệt kê các mục chọn chức năng trong một cửa sổ của ứng dụng.

Một đối tượng tương ứng với MenuStrip Control thuộc lớp MenuStrip.

Các mục chọn trong thanh menu tương ứng với các đối tượng thuộc lớp ToolStripMenuItem.





# I. Các thành phần cơ bản trên WinForm

## - MDI Form

---

1/ Có hai dạng Form là: Modal Form và Modeless Form.

- + Phương thức ShowDialog(): hiển thị Modal Form

- + Phương thức Show(): hiển thị Modeless Form

2/ MDI Form (Multiple-Document Interface) : để một Form là MDI Form thì property IsMdiContainer có giá trị true.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng DataGridView

- + DataGridView Control dùng nhập/xuất dữ liệu dạng bảng (gồm nhiều dòng và nhiều cột).
- + Lớp DataGridView tương ứng với DataGridView Control.
- + Các Properties:
  - `object DataSource { get; set; }`: liên kết với tập dữ liệu hiển thị lên DataGridView.
  - `System.Windows.Forms.DataGridViewRowCollection Rows { get; }`: trả về tập hợp chứa tất cả các dòng trong DataGridView.
  - `System.Windows.Forms.DataGridViewSelectedRowCollection SelectedRows { get; }`: trả về tập hợp các dòng được lựa chọn trong DataGridView.



# I. Các thành phần cơ bản trên WinForm

## -Sử dụng DataGridView

+ Sự kiện event

`System.Windows.Forms.DataGridViewCellEventHandler`

**RowEnter**: phát ra khi dòng nhận focus nhưng trước khi nó trở thành dòng hiện tại.

Ví dụ: Liên kết DataGridView có tên dgv với tập dữ liệu dsHS.

```
BindingSource bs = new BindingSource();  
bs.DataSource = dsHS;  
dgv.DataSource = bs;
```





## II. Kiểu dữ liệu tập hợp

1/ Lớp Array: kiểu mảng

2/ Lớp List<T>: danh sách các phần tử mà mỗi phần tử có kiểu T.

+ Các phương thức:

**void Add (T item)**: Thêm 1 phần tử vào cuối danh sách.

**bool Remove (T item)**: xóa phần tử đầu tiên trong danh sách.

+ Các Properties:

**int Count { get; }**: Trả về số phần tử có trong danh sách.

**T this[int index] { get; set; }**: Tham chiếu đến phần tử tại chỉ số index trong danh sách.



## II. Kiểu dữ liệu tập hợp

3/ Lớp ArrayList: kiểu dữ liệu chứa tập hợp các phần tử có cùng kiểu hoặc khác kiểu.

4/ Lớp Dictionary<Tkey,Tvalue>: biểu diễn tập các phần tử mà mỗi phần tử là một bộ gồm (key, value).

Tkey: kiểu dữ liệu cho key (khóa của đối tượng).

Tvalue: kiểu dữ liệu cho value (giá trị của đối tượng).

+ Các phương thức:

**void Add (TKey key, TValue value)**: thêm một phần tử là bộ (key, value) vào tập hợp.

**bool Remove (TKey key)**: xóa một phần tử trong tập hợp có khóa key.



## II. Kiểu dữ liệu tập hợp

+Các properties:

`int Count { get; }:` trả về số phần tử có trong tập hợp.

`System.Collections.Generic.Dictionary<TKey, TValue>.`

`KeyCollection Keys { get; }:` trả về tập khóa của tập hợp.

`System.Collections.Generic.Dictionary<TKey, TValue>.`

`ValueCollection Values { get; }:` trả về tập giá trị của tập hợp.

`TValue this[TKey key] { get; set; }:` tham chiếu đến giá trị của phần tử có khóa key. Lệnh sẽ bị lỗi nếu khóa key không tồn tại trong tập khóa.

5/ Lớp Hashtable: danh sách các phần tử mà mỗi phần tử là bộ (khóa, giá trị). Mỗi phần tử trong danh sách phải có khóa được sử dụng để tìm kiếm.



### III. Ví dụ: Xây dựng ứng dụng Quản lý thông tin học sinh

---

- + Thông tin của học sinh bao gồm: mã số học sinh, họ tên, ngày sinh, phái và địa chỉ.
- + Hiện thực các phép toán: xem danh sách học sinh. Thêm, xóa và sửa thông tin của học viên.
- + Các bước hiện thực ứng dụng:  
Bước 1: xây dựng CTDL lưu trữ thông tin học sinh:  
Bước 2: thiết kế giao diện cho ứng dụng  
Bước 3: hiện thực các phép toán



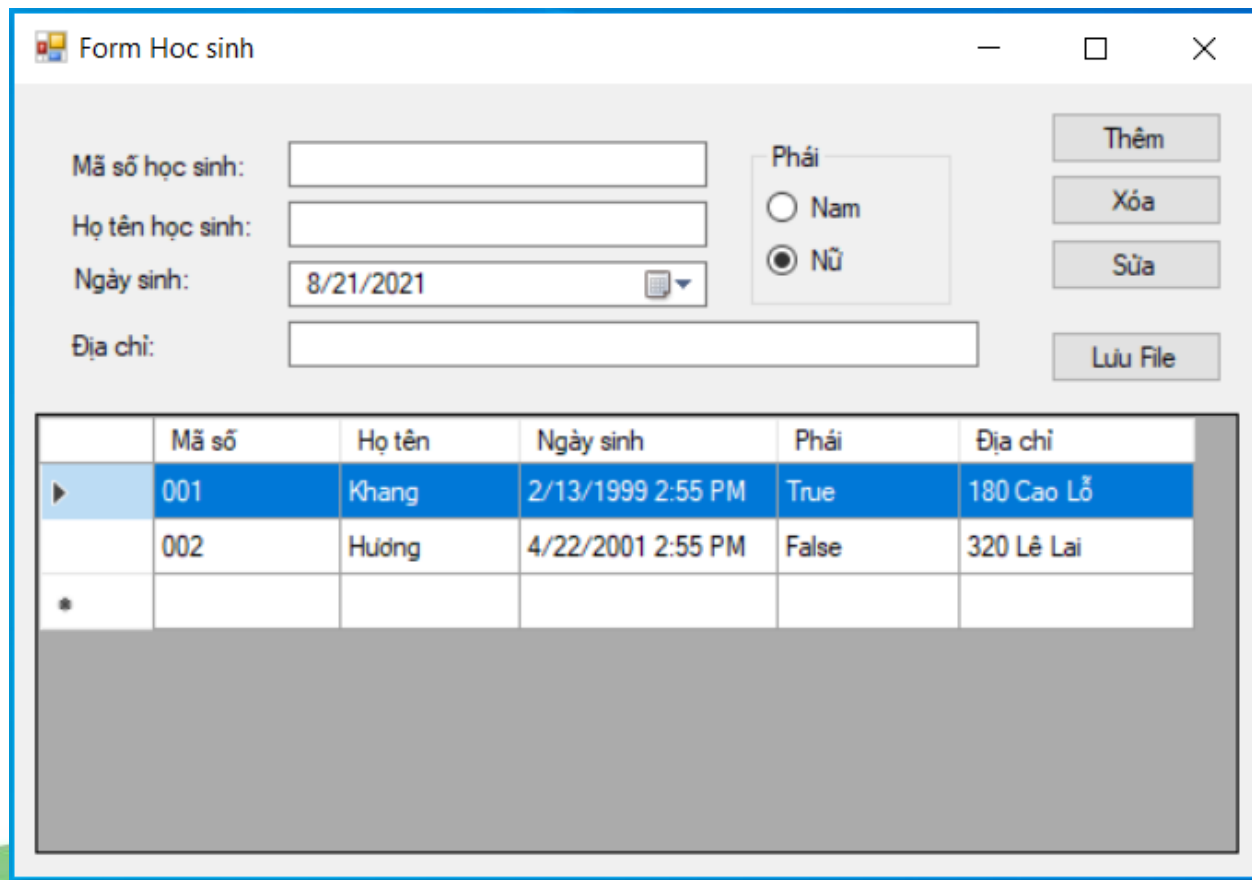
### III. Ví dụ: Xây dựng ứng dụng Quản lý thông tin học sinh

Bước 1: xây dựng lớp **CHocSinh** lưu trữ thông tin học sinh:

```
class CHocSinh
{
    private string m_mshs;
    private string m_hoten;
    private DateTime m_ngaysinh;
    private bool m_phai;
    private string m_diachi;
    6 references
    public string mshs[...]
    4 references
    public string hoten[...]
    4 references
    public DateTime ngaysinh[...]
    4 references
    public bool phai[...]
    4 references
    public string diachi[...]
    0 references
    public CHocSinh(string mshs, string hoten
        , DateTime ngaysinh, bool phai, string diachi) [...]
    1 reference
    public CHocSinh() [...]
}
```

### III. Ví dụ: Xây dựng ứng dụng Quản lý thông tin học sinh

#### Bước 2: thiết kế giao diện cho ứng dụng



The screenshot shows a Windows-style application window titled "Form Hoc sinh". It contains several input fields for student information: "Mã số học sinh:", "Họ tên học sinh:", "Ngày sinh:" (with a date picker showing 8/21/2021), and "Địa chỉ:". To the right of these fields is a "Phái" (Gender) section with radio buttons for "Nam" and "Nữ" (selected). Further right are four buttons: "Thêm", "Xóa", "Sửa", and "Lưu File". Below the input fields is a table with 6 columns: "Mã số", "Họ tên", "Ngày sinh", "Phái", and "Địa chỉ". The first two rows of the table are highlighted in blue. The first row contains the values: 001, Khang, 2/13/1999 2:55 PM, True, 180 Cao Lỗ. The second row contains: 002, Hướng, 4/22/2001 2:55 PM, False, 320 Lê Lai. Below the table is a large grey rectangular area, likely a placeholder for more data or a list view.

	Mã số	Họ tên	Ngày sinh	Phái	Địa chỉ
▶	001	Khang	2/13/1999 2:55 PM	True	180 Cao Lỗ
	002	Hướng	4/22/2001 2:55 PM	False	320 Lê Lai
*					



### III. Ví dụ: Xây dựng ứng dụng Quản lý thông tin học sinh

Lớp **FormHocSinh** tương ứng với giao diện

```
public partial class FormHocSinh : Form
{
    private List<CHocSinh> dsHS;
    1 reference
    public FormHocSinh()...
    1 reference
    private void FormHocSinh_Load(object sender, EventArgs e)...
    4 references
    private void hienthi()...
    1 reference
    private void btnThem_Click(object sender, EventArgs e)...
    1 reference
    private void btnXoa_Click(object sender, EventArgs e)...
    1 reference
    private void btnSua_Click(object sender, EventArgs e)...
    1 reference
    private void dgv_RowEnter(object sender, DataGridViewCellEventArgs e)...
    1 reference
    private void btnLuuFile_Click(object sender, EventArgs e)...
}
```







### III. Ví dụ: Xây dựng ứng dụng Quản lý thông tin học sinh

Bước 3: hiện thực các phép toán

1/ Xem danh sách học sinh:

```
private void hienthi()...
```

2/ Thêm học sinh:

```
private void btnThem_Click(object sender, EventArgs e)...
```

3/ Xóa học sinh:

```
private void btnXoa_Click(object sender, EventArgs e)...
```

4/ Sửa thông tin học sinh

```
private void btnSua_Click(object sender, EventArgs e)...
```

1 reference

```
private void dgv_RowEnter(object sender, DataGridViewCellEventArgs e)...
```





## IV. Thao tác trên File

### 1/ Lớp OpenFileDialog và lớp SaveFileDialog

- + Property FileName: chứa tên file trong FileDialog.
- + Method ShowDialog(): hiển thị cửa sổ Dialog.

### 2/ Các lớp hỗ trợ thao tác trên File:

- + Lớp File, FileStream
- + Lớp BinaryFormatter
- + Lớp StringReader, StringWriter



## IV. Thao tác trên File

3/ Lớp `FileStream` thuộc interface `System.IO`:

Phương thức tạo lập: Tạo đối tượng và gắn kết với file

```
FileStream (string path, System.IO.FileMode mode);
```

```
FileStream (string path, System.IO.FileMode mode,  
System.IO.FileAccess access);
```



## IV. Thao tác trên File

4/ Lớp `BinaryFormatter` thuộc interface `System.Runtime.Serialization.Formatters.Binary`

+ Đọc và ghi đối tượng lên file nhị phân.

+ Các phương thức:

`void Serialize (System.IO.Stream serializationStream, object graph)`: Ghi đối tượng lên file nhị phân.

`object Deserialize (System.IO.Stream serializationStream)`:  
đọc đối tượng từ file nhị phân.



## IV. Thao tác trên File

5/ Ví dụ:

+ Ghi danh sách học sinh lên file nhị phân.

```
private void btnLuuFile_Click(object sender, EventArgs e)...
```

+ Đọc danh sách học sinh từ file nhị phân: khi ứng dụng thực thi thì thực hiện công việc đọc danh sách học sinh từ file nhị phân và hiển thị lên DataGridView.

```
private void FormHocSinh_Load(object sender, EventArgs e)...
```



## V. Nắm bắt lỗi

- + Tra cứu mục Exception trong thư viện MSDN để xác định kiểu tương ứng với lỗi phát ra.
- + Sử dụng cấu trúc lệnh *try ... catch* để nắm bắt lỗi đó.
- + Ví dụ: Quản lý học sinh nhưng sử dụng kiểu Dictionary để lưu trữ danh sách học sinh.



## VI. Mẫu 2-Tiers

Khi xây dựng ứng dụng trên Desktop ta thấy trong ứng dụng có các thành phần sau:

- + Giao diện
- + Xử lý dữ liệu
- + Truy cập dữ liệu

Mẫu 2-Tiers chia ứng dụng thành 2 tầng:

- + Tầng 1: Giao diện
- + Tầng 2: Xử lý dữ liệu và truy cập dữ liệu.

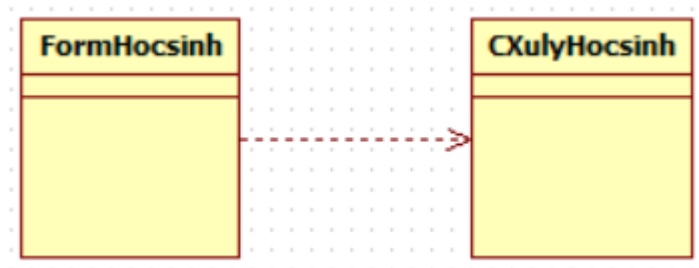
Mục đích của việc phân chia ứng dụng thành nhiều tầng để ứng dụng dễ bảo trì và nâng cấp.



## VI. Mẫu 2-Tiers

Ví dụ: Quản lý học sinh

- + Lớp **FormHocsinh**: giao diện
- + Lớp **CXulyHocsinh**: xử lý dữ liệu và truy cập dữ liệu



## VI. Mẫu 2-Tiers

Ví dụ: Quản lý học sinh  
+ Tầng 1: giao diện

Form Học sinh

Mã số học sinh: 001

Họ tên học sinh: Phan Thị Lan

Ngày sinh: 2/13/2000

Địa chỉ: 190 Phạm Hùng

Phái

☐ Nam

☒ Nữ

Thêm

Xóa

Sửa

Lưu File

	Mã số	Họ tên	Ngày sinh	Phái	Địa chỉ
▶	001	Phan Thị Lan	2/13/2000 2:55 PM	False	190 Phạm Hùng
	002	Hương	4/22/2001 2:55 PM	False	320 Lê Lai
	003	Trần Phú	11/4/2001 1:33 PM	True	340 Trần Hùng Đạo
*					

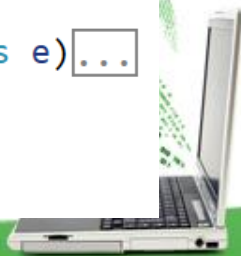




## VI. Mẫu 2-Tiers

### Lớp tương ứng với giao diện

```
public partial class FormHocSinh : Form
{
    private CXulyHocsinh x1;
    1 reference
    public FormHocSinh()...
    1 reference
    private void FormHocSinh_Load(object sender, EventArgs e)...
    4 references
    private void hienthi()...
    1 reference
    private void btnThem_Click(object sender, EventArgs e)...
    1 reference
    private void btnXoa_Click(object sender, EventArgs e)...
    1 reference
    private void btnSua_Click(object sender, EventArgs e)...
    1 reference
    private void dgv_RowEnter(object sender, DataGridViewCellEventArgs e)...
    1 reference
    private void btnLuuFile_Click(object sender, EventArgs e)...
}
```





## VI. Mẫu 2-Tiers

+ Tầng 2: xử lý dữ liệu và truy cập dữ liệu  
CTDL ứng dụng

```
class CHocSinh
{
    private string m_mshs;
    private string m_hoten;
    private DateTime m_ngaysinh;
    private bool m_phai;
    private string m_diachi;
    5 references
    public string mshs...
    5 references
    public string hoten...
    5 references
    public DateTime ngaysinh...
    5 references
    public bool phai...
    5 references
    public string diachi...
    0 references
    public CHocSinh(string mshs, string hoten
        , DateTime ngaysinh, bool phai, string diachi)...
    2 references
    public CHocSinh()...
}
```





## VI. Mẫu 2-Tiers

Lớp CXulyHocsinh:

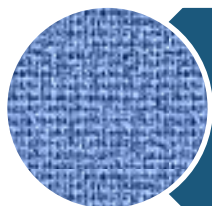
```
class CXulyHocsinh
{
    private List<CHocSinh> dsHS;
    1 reference
    public CXulyHocsinh()...
    1 reference
    public List<CHocSinh> getDSHocsinh()...
    3 references
    public CHocSinh tim(string mshs)...
    1 reference
    public void them(CHocSinh hs)...
    1 reference
    public void xoa(string mshs)...
    1 reference
    public void sua(CHocSinh hs)...
    1 reference
    public bool ghiFile(string tenfile)...
    1 reference
    public bool docFile(string tenfile)...
}
```



# Bài 5: TÍNH THỪA KẾ VÀ TÍNH ĐA HÌNH



Tính thừa kế



Tính Đa hình



# I. Tính thừa kế

1/ Khái niệm tổng quát hóa và đặc biệt hóa:

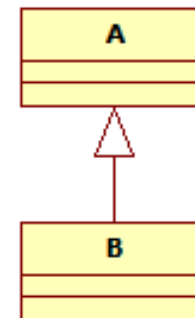
Đối tượng A có đầy đủ tính chất của đối tượng B và có thêm các tính chất mới của riêng nó, khi đó đối tượng A gọi là **đặc biệt hóa** của đối tượng B hay đối tượng B gọi là **tổng quát hóa** của đối tượng A.

2/ Khái niệm thừa kế

Lớp A là tổng quát hóa của lớp B (hay lớp B là đặc biệt hóa của lớp A), khi đó ta gọi lớp B là lớp thừa kế từ lớp A.

Lớp A gọi là lớp cơ sở

Lớp B gọi là lớp thừa kế

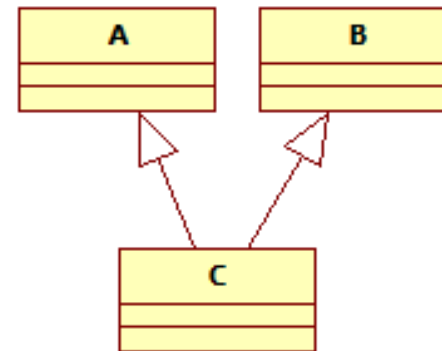
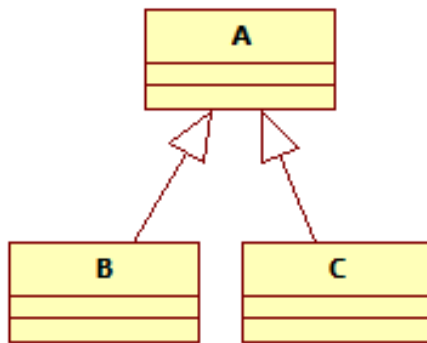




# I. Tính thừa kế

## 3/ Đơn thừa kế và đa thừa kế

Đơn thừa kế	Đa thừa kế
Một lớp thừa kế có duy nhất một lớp cơ sở.	Một lớp thừa kế có từ hai lớp cơ sở trở lên.



- + Trong C# có đơn thừa kế và không có đa thừa kế.
- + Trong C++ có đơn thừa kế và đa thừa kế.





# I. Tính thừa kế

## 4/ Khai báo lớp dẫn xuất trong C#

```
class <tên lớp thừa kế>: <tên lớp cơ sở>
{
    <các thành phần>
}
```

- + Lớp thừa kế có tất cả các thành phần của lớp cơ sở.
- + Thuộc tính truy cập protected trong lớp cơ sở.
- + Trong lớp thừa kế truy cập được những thành phần của lớp cơ sở có thuộc tính truy cập là protected hay public.
- + Lớp dẫn xuất:



# I. Tính thừa kế

## 5/ Phương thức tạo lập của lớp dẫn xuất

```
class A
{
    protected int m_a;
    public A()
    {
        m_a = 0;
    }
    public A(int m)
    {
        m_a = m;
    }
    public int getA()
    {
        return m_a;
    }
    public void setA(int m)
    {
        m_a = m;
    }
}
```

```
class B : A
{
    protected int m_b;
    public B():base()
    {
        m_b = 0;
    }
    public B(int ma, int mb)
        : base(ma)
    {
        m_b = mb;
    }
    public int getB()
    {
        return m_b;
    }
    public void setB(int m)
    {
        m_b = m;
    }
    public void setB(int ma, int mb)
    {
        m_b = mb;
        m_a = ma;
    }
}
```







# I. Tính thừa kế

## 6/ Sự trùng tên

Một biến thuộc lớp cơ sở có thể tham chiếu đến các đối tượng thuộc lớp dẫn xuất.

```
class coso
{
    public int m;
}
class danxuat : coso
{
    public int m;
}

static void Main(string[] args)
{
    danxuat x = new danxuat();
    x.m = 1; //m của lớp danxuat

    coso y = (coso)x;
    y.m = 2; //m của lớp coso
}
```



# I. Tính thừa kế

7/ Ví dụ 1:

+ Xây dựng lớp hình chữ nhật sao cho mỗi đối tượng biểu diễn bằng độ dài hai cạnh của nó.

```
class HìnhChuNhat
{
    protected double m_dai;
    protected double m_rong;
    0 references
    public double ChieuDai[...]
    0 references
    public double ChieuRong[...]
    1 reference
    public HìnhChuNhat(...)
    4 references
    public HìnhChuNhat(double dai, double rong) [...]
    0 references
    public double dientich() [...]
    0 references
    public double chuvi() [...]
}
```



# I. Tính thừa kế

+ Xây dựng lớp Hình vuông thừa kế từ lớp Hình chữ nhật.

```
class HìnhVuong:HìnhChuNhat
{
    0 references
    public HìnhVuong():base()
    {}
    2 references
    public HìnhVuong(double canh):base(canh,canh)
    {}
}
```



# I. Tính thừa kế

8/ Ví dụ 2:

+ Xây dựng lớp Tam giác sao cho mỗi đối tượng biểu diễn bằng độ dài ba cạnh của nó.

```
class TamGiac
{
    protected double m_a;
    protected double m_b;
    protected double m_c;
    0 references
    public double a...
    0 references
    public double b...
    0 references
    public double c...
    1 reference
    public TamGiac()...
    2 references
    public TamGiac(double a,
        double b, double c)...
    0 references
    public double chuvi()...
    0 references
    public double dientich()...
}
```



# I. Tính thừa kế

+ Xây dựng lớp Tam giác cân thừa kế từ lớp Tam giác.

```
class TamGiacCan:TamGiac
{
    0 references
    public TamGiacCan():base()
    {}
    0 references
    public TamGiacCan(double canhben,double canhday)
        :base(canhben,canhben,canhday)
    {}
}
```



## II. Tính đa hình

### 1/ Phương thức tĩnh:

Cho đoạn chương trình:

```
22 class Program
23 {
24     static void Main(string[] args)
25     {
26         A x;
27         x = new A();
28         x.xuat();
29         x = new B();
30         x.xuat();
31     }
32 }
```

```
class A
{
    public void xuat()
    {
        Console.WriteLine("Lop A");
    }
}
class B : A
{
    public void xuat()
    {
        Console.WriteLine("Lop B");
    }
}
```



## II. Tính đa hình

Lớp dẫn xuất B có nguyên mẫu phương thức xuất() giống như nguyên mẫu phương thức xuất() của lớp cơ sở A.

Câu lệnh 26: biến x thuộc lớp A

Câu lệnh 27, 28: phương thức xuất() của lớp A được gọi

Câu lệnh 29, 30: phương thức xuất() của lớp A vẫn được gọi.

Vậy khi truy cập các thành phần trùng tên của lớp cơ sở và lớp dẫn xuất thì thành phần được gọi phụ thuộc vào biến quản lý đối tượng chứ không phụ thuộc vào đối tượng. Phương thức hoạt động theo nguyên tắc này gọi là phương thức tĩnh.



## II. Tính đa hình

### 2/ Phương thức ảo:

Cho đoạn chương trình:

```
22 class Program
23 {
24     static void Main(string[] args)
25     {
26         A x;
27         x = new A();
28         x.xuat();
29         x = new B();
30         x.xuat();
31     }
32 }
```

```
class A
{
    public virtual void xuat()
    {
        Console.WriteLine("Lop A");
    }
}
class B : A
{
    public override void xuat()
    {
        Console.WriteLine("Lop B");
    }
}
```





## II. Tính đa hình

Nguyên mẫu phương thức xuất() của lớp cơ sở A có đặt thêm từ khóa virtual.

Nguyên mẫu phương thức xuất() của lớp dẫn xuất B có đặt thêm từ khóa override.

Câu lệnh 26: biến x thuộc lớp A

Câu lệnh 27, 28: phương thức xuất() của lớp A được gọi

Câu lệnh 29, 30: phương thức xuất() của lớp B được gọi.

Vậy khi truy cập các thành phần trùng tên của lớp cơ sở và lớp dẫn xuất thì thành phần được gọi phụ thuộc vào đối tượng chứ không phụ thuộc vào biến quản lý đối tượng.

Phương thức hoạt động theo nguyên tắc này gọi là phương thức ảo.



## II. Tính đa hình

### 3/ Tính đa hình:

Cho đoạn chương trình:

```
30 class Program
31 {
32     static void Main(string[] args)
33     {
34         A x = null;
35         x = new A();
36         x.xuat();
37         x = new B();
38         x.xuat();
39         x = new C();
40         x.xuat();
41     }
42 }
```

```
class A
{
    public virtual void xuat()
    {
        Console.WriteLine("Lop A");
    }
}
class B : A
{
    public override void xuat()
    {
        Console.WriteLine("Lop B");
    }
}
class C : B
{
    public override void xuat()
    {
        Console.WriteLine("Lop C");
    }
}
```



## II. Tính đa hình

Câu lệnh 34: biến x thuộc lớp A

Câu lệnh 35,36: phương thức xuất() của lớp A được gọi

Câu lệnh 37,38: phương thức xuất() của lớp B được gọi.

Câu lệnh 39,40: phương thức xuất() của lớp C được gọi.

Ta thấy cùng một lệnh x.xuat() nhưng khi viết tại các dòng lệnh 36, 38, và 40 thì ý nghĩa của câu lệnh là khác nhau. Như vậy cùng một câu lệnh thì tùy vào từng ngữ cảnh khác nhau thì ý nghĩa của câu lệnh là khác nhau và tính chất này gọi là tính đa hình trong lập trình hướng đối tượng.

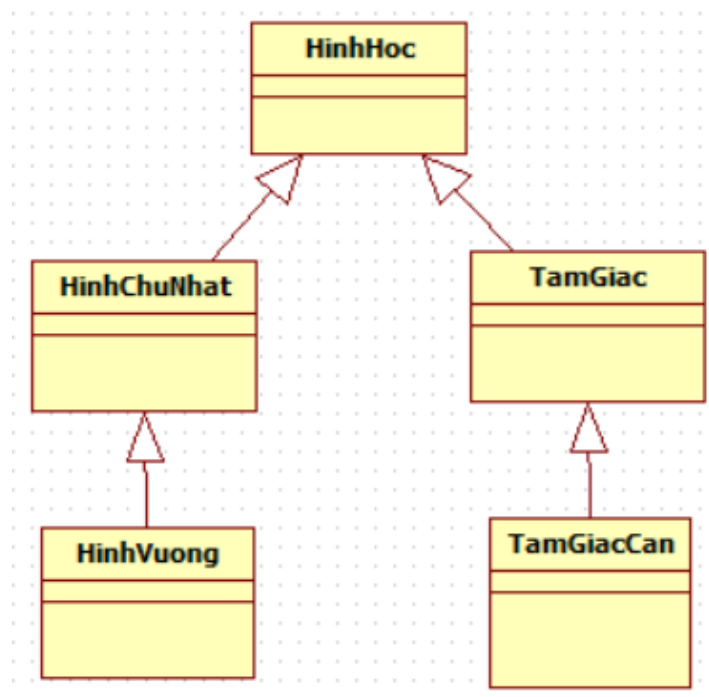
### III. Lớp cơ sở trừu tượng và Interface

#### 1) Lớp cơ sở trừu tượng:

- ❑ Lớp cơ sở trừu tượng là lớp đối tượng dùng để tạo mối quan hệ thừa kế giữa các lớp đối tượng và không có bất kỳ đối tượng nào được tạo ra từ nó.
- ❑ Phương thức thuần ảo (phương thức ảo thuần túy):  
Là phương thức dùng để tạo mối quan hệ giữa các phương thức ảo và là phương thức không có định nghĩa phương thức mà chỉ có nguyên mẫu của phương thức.

### III. Lớp cơ sở trừu tượng và Interface

Ví dụ: Tạo lớp **HìnhHoc** là lớp cơ sở trừu tượng của hai lớp **HìnhChuNhat** và **TamGiac**.



```
public enum KieuHinh { HìnhHoc, TamGiac,
    TamGiacCan, HìnhChuNhat, HìnhVuong }
10 references
abstract class HìnhHoc
{
    4 references
    public abstract double chuvi();
    2 references
    public abstract double dientich();
    6 references
    public virtual KieuHinh kieuDoiTuong()
    {
        return KieuHinh.HìnhHoc;
    }
}
```

### III. Lớp cơ sở trừu tượng và Interface

```
class TamGiac:HinhHoc
{
    protected double m_a;
    protected double m_b;
    protected double m_c;
    0 references
    public double a[...]
    0 references
    public double b[...]
    0 references
    public double c[...]
    1 reference
    public TamGiac()[...]
    2 references
    public TamGiac(double a,
        double b,double c) [...]
    3 references
    public override double chuvi() [...]
    1 reference
    public override double dientich() [...]
    4 references
    public override KieuHinh kieuDoiTuong()
    {
        return KieuHinh.TamGiac;
    }
}
```

```
class HinhChuNhat:HinhHoc
{
    protected double m_dai;
    protected double m_rong;
    1 reference
    public HinhChuNhat() [...]
    4 references
    public HinhChuNhat(double dai,double rong) [...]
    1 reference
    public virtual double ChieuDai [...]
    1 reference
    public virtual double ChieuRong [...]
    1 reference
    public override double dientich() [...]
    3 references
    public override double chuvi() [...]
    4 references
    public override KieuHinh kieuDoiTuong()
    {
        return KieuHinh.HinhChuNhat;
    }
}
```

### III. Lớp cơ sở trừu tượng và Interface

```
class HìnhVuong:HìnhChuNhat
{
    0 references
    public HìnhVuong():base()
    {}
    2 references
    public HìnhVuong(double canh)
        :base(canh,canh)
    {}
    1 reference
    public override double ChieuDai[...]
    1 reference
    public override double ChieuRong[...]
    4 references
    public override KieuHinh kieuDoiTuong()
    {
        return KieuHinh.HìnhVuong;
    }
}
```

```
class TamGiacCan:TamGiac
{
    0 references
    public TamGiacCan():base()
    {}
    0 references
    public TamGiacCan(double canhben,double canhday)
        :base(canhben,canhben,canhday)
    {}
    4 references
    public override KieuHinh kieuDoiTuong()
    {
        return KieuHinh.TamGiacCan;
    }
}
```



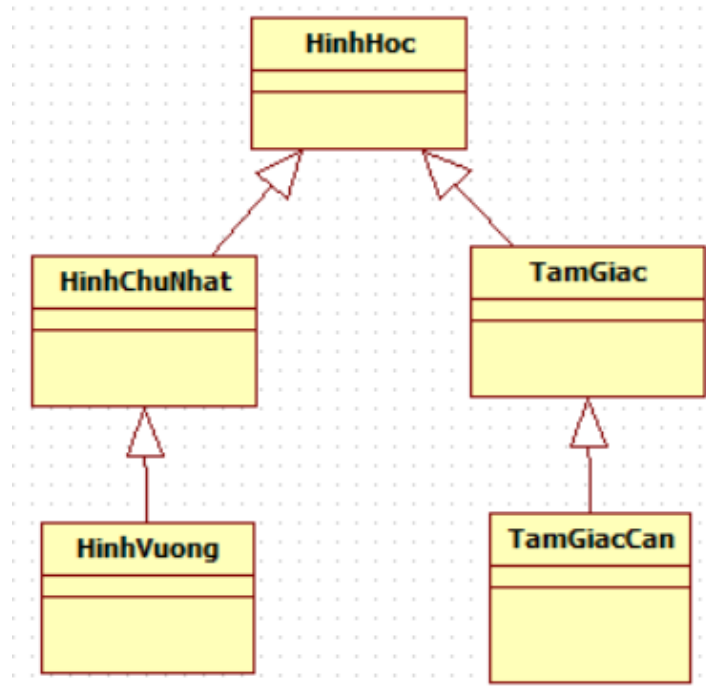
### III. Lớp cơ sở trừu tượng và Interface

- 2) Interface: dùng để tạo mối quan hệ thừa kế giữa các lớp đối tượng và trong nó chỉ chứa các phương thức thuần ảo.



### III. Lớp cơ sở trừu tượng và Interface

Ví dụ: Tạo **HinhHoc** là interface của hai lớp **HinhChuNhat** và **TamGiac**.



```
public enum KieuHinh { TamGiac, TamGiacCan,
    HinhChuNhat, HinhVuong }
```

10 references

```
interface HinhHoc
```

```
{
```

4 references

```
double chuvi();
```

2 references

```
double dientich();
```

5 references

```
KieuHinh kieuDoiTuong();
```

```
}
```

### III. Lớp cơ sở trừu tượng và Interface

```
class TamGiac:HinhHoc
```

```
{
```

```
    protected double m_a;
```

```
    protected double m_b;
```

```
    protected double m_c;
```

```
    0 references
```

```
    public double a...
```

```
    0 references
```

```
    public double b...
```

```
    0 references
```

```
    public double c...
```

```
    1 reference
```

```
    public TamGiac()...
```

```
    2 references
```

```
    public TamGiac(double a,  
                    double b,double c)...
```

```
    3 references
```

```
    public virtual double chuvi()...
```

```
    1 reference
```

```
    public virtual double dientich()...
```

```
    3 references
```

```
    public virtual KieuHinh kieuDoiTuong()
```

```
    {
```

```
        return KieuHinh.TamGiac;
```

```
    }
```

```
}
```

```
class HinhChuNhat:HinhHoc
```

```
{
```

```
    protected double m_dai;
```

```
    protected double m_rong;
```

```
    1 reference
```

```
    public virtual double ChieuDai...
```

```
    1 reference
```

```
    public virtual double ChieuRong...
```

```
    1 reference
```

```
    public HinhChuNhat()...
```

```
    4 references
```

```
    public HinhChuNhat(double dai,double rong)...
```

```
    1 reference
```

```
    public virtual double dientich()...
```

```
    3 references
```

```
    public virtual double chuvi()...
```

```
    4 references
```

```
    public virtual KieuHinh kieuDoiTuong()
```

```
    {
```

```
        return KieuHinh.HinhChuNhat;
```

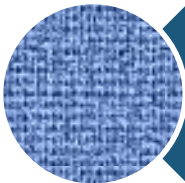
```
    }
```

```
}
```

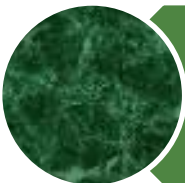
# Bài 6: LIÊN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG



Quan hệ bao hàm/ bộ phận (Has-A/Part-Of)



Quan hệ đặc biệt hóa/ Tổng quát hóa (Is-A)



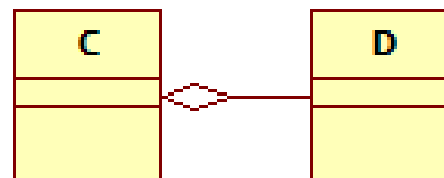
Quan hệ phụ thuộc



## I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

1/ Quan hệ bao hàm độc lập (aggregation): hai lớp đối tượng C và D có mối quan hệ bao hàm độc lập nếu đối tượng của lớp C chứa đối tượng của lớp D và nếu đối tượng của lớp C bị hủy thì đối tượng của lớp D vẫn còn tồn tại.

❑ Biểu diễn bằng UML

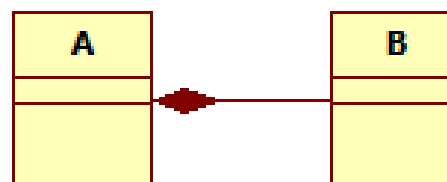




# I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

2/ Quan hệ bao hàm phụ thuộc (composition): hai lớp đối tượng A và B có mối quan hệ bao hàm phụ thuộc nếu đối tượng của lớp A chứa đối tượng của lớp B và nếu đối tượng của lớp A bị hủy thì đối tượng của lớp B cũng bị hủy theo.

❑ Biểu diễn bằng UML





# I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

## 3/ Ví dụ:

Xây dựng ứng dụng quản lý Hóa đơn bán lẻ như sau:

Số hóa đơn: hd001

Ngày lập hóa đơn: 10/4/2012

Tên khách hàng: Lê Lai

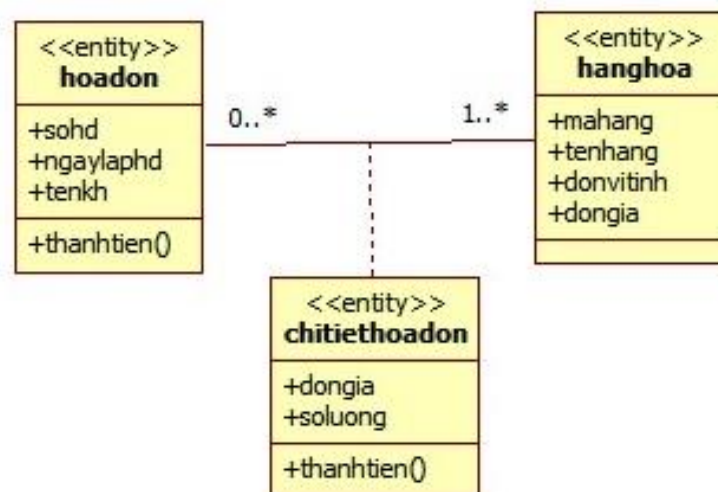
Mã hàng	Tên hàng	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
mh001	Nokia 3080	Cái	1	3500000	3500000
mh002	USB 4Gb	Cái	2	250000	500000

Tổng thành tiền: 4.000.000 đồng

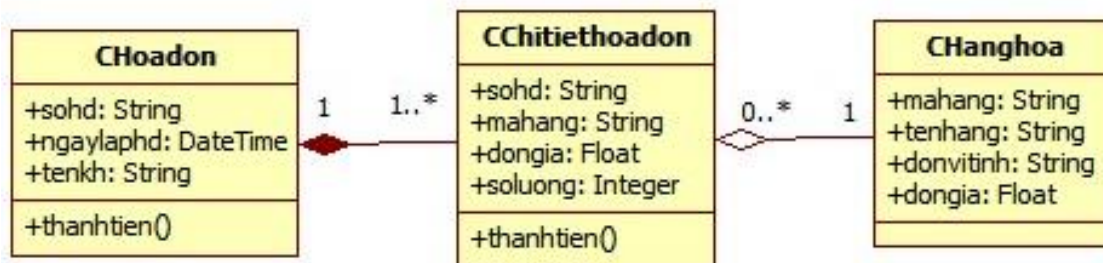


# I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

+ CSDL của ứng dụng:



+ CTDL của ứng dụng:



# I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

+ Cài đặt:

```
class CChitietHoadon
{
    private int m_soluong;
    private double m_dongia;
    private CHangHoa m_hanghoa;
    2 references
    public int soluong...
    2 references
    public double dongia...
    4 references
    public CHangHoa hanghoa...
    1 reference
    public CChitietHoadon()...
    0 references
    public CChitietHoadon(CHangHoa hanghoa
        ,int soluong,double dongia)...
    2 references
    public double thanhtien()...
}
```

```
class CHangHoa
{
    private string m_mahang;
    private string m_tenhang;
    private string m_dvt;
    private double m_dongia;
    7 references
    public string mahang...
    8 references
    public string tenhang...
    8 references
    public string dvt...
    8 references
    public double dongia...
    0 references
    public CHangHoa(string mahang,
        string tenhang,string dvt,
        double dongia)...
    2 references
    public CHangHoa()...
}
```





# I. Quan hệ bao hàm độc lập và quan hệ bao hàm phụ thuộc

```
class CHoaDon
{
    private string m_sohd;
    private DateTime m_ngaylaphd;
    private string m_tenk;
    private List<CChitietHoadon> m_cthd;
    5 references
    public string sohd...
    4 references
    public DateTime ngaylaphd...
    4 references
    public string tenkh...
    3 references
    public List<CChitietHoadon> DanhSachChitietHoadon...
    0 references
    public CHoaDon(string sohd,DateTime ngaylaphd,string tenkh)...
    2 references
    public CHoaDon()...
    0 references
    public double thanhtien()...
}
```

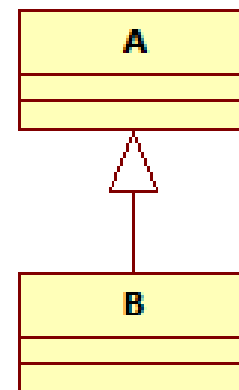


## II. Quan hệ đặc biệt hóa/ Tổng quát hóa

+ Quan hệ giữa hai lớp đối tượng A và B, A là tổng quát hóa của B và B là đặc biệt hóa của A.

+ Trong lập trình hướng đối tượng thì A là lớp cơ sở của B và B là lớp thừa kế từ A.

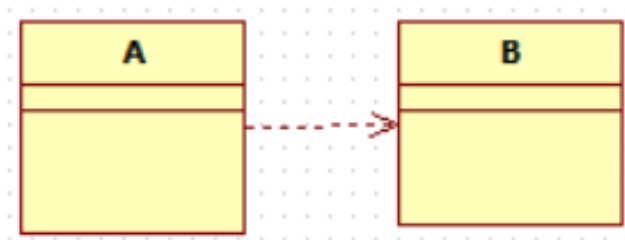
+ Việc cài đặt mối quan hệ này chúng ta dựa vào tính thừa kế và tính đa hình trong lập trình hướng đối tượng.





### III. Quan hệ phụ thuộc

+ Quan hệ giữa hai lớp đối tượng A và B, A phụ thuộc vào B. Nếu có sự thay đổi trên lớp B thì gây ra sự thay đổi của lớp A. Ngược lại nếu lớp A thay đổi thì không ảnh hưởng đến lớp B.





## IV. Mẫu 3-Tiers

1/ Khi xây dựng ứng dụng trên Desktop ta thấy trong ứng dụng có các thành phần sau:

- + Giao diện
- + Xử lý dữ liệu
- + Truy cập dữ liệu

2/ Mẫu 3-Tiers chia ứng dụng thành 3 tầng:

- + Tầng 1: Giao diện
- + Tầng 2: Xử lý dữ liệu
- + Tầng 3: Truy cập dữ liệu.

3/ Mục đích của việc phân chia ứng dụng thành nhiều tầng để ứng dụng dễ bảo trì và nâng cấp.



## V. Mẫu Singleton

+ Mẫu Singleton được sử dụng để xây dựng lớp đối tượng sao cho chỉ tồn tại nhiều nhất một đối tượng có kiểu dữ liệu là lớp đối tượng này trong chương trình.

+ Cài đặt:

```
class Singleton
{
    private static Singleton m_object = null;
    1 reference
    private Singleton()
    {
    }
    0 references
    public static Singleton CreateObject()
    {
        if (m_object == null)
            m_object = new Singleton();
        return m_object;
    }
}
```



## VI. Ví dụ: Quản lý hóa đơn

### 1/ Nghiệp vụ:

Xây dựng ứng dụng quản lý Hóa đơn bán lẻ như sau:

Số hóa đơn: hd001

Ngày lập hóa đơn: 10/4/2012

Tên khách hàng: Lê Lai

Mã hàng	Tên hàng	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
mh001	Nokia 3080	Cái	1	3500000	3500000
mh002	USB 4Gb	Cái	2	250000	500000

Tổng thành tiền: 4.000.000 đồng

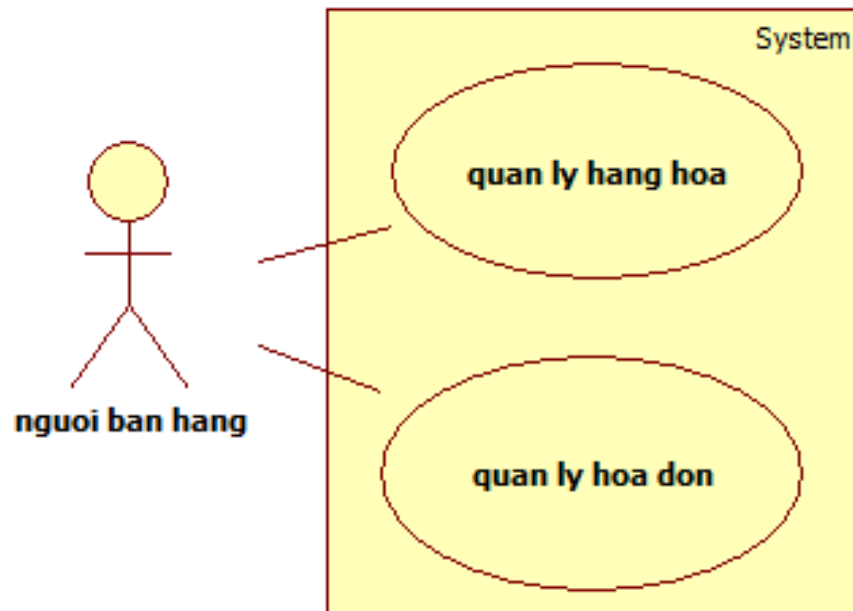


## VI. Ví dụ: Quản lý hóa đơn

2/ Sơ đồ UseCase  
(Các chức năng trong  
ứng dụng):

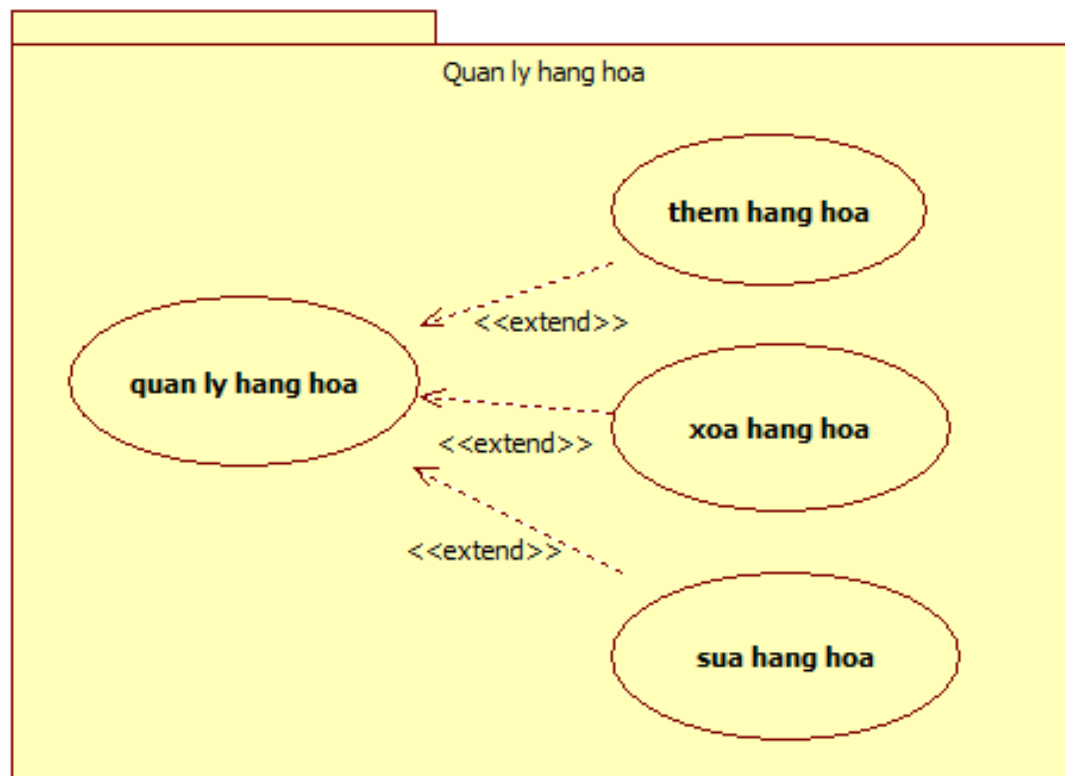
+ Quản lý hàng hóa

+ Quản lý hóa đơn



## VI. Ví dụ: Quản lý hóa đơn

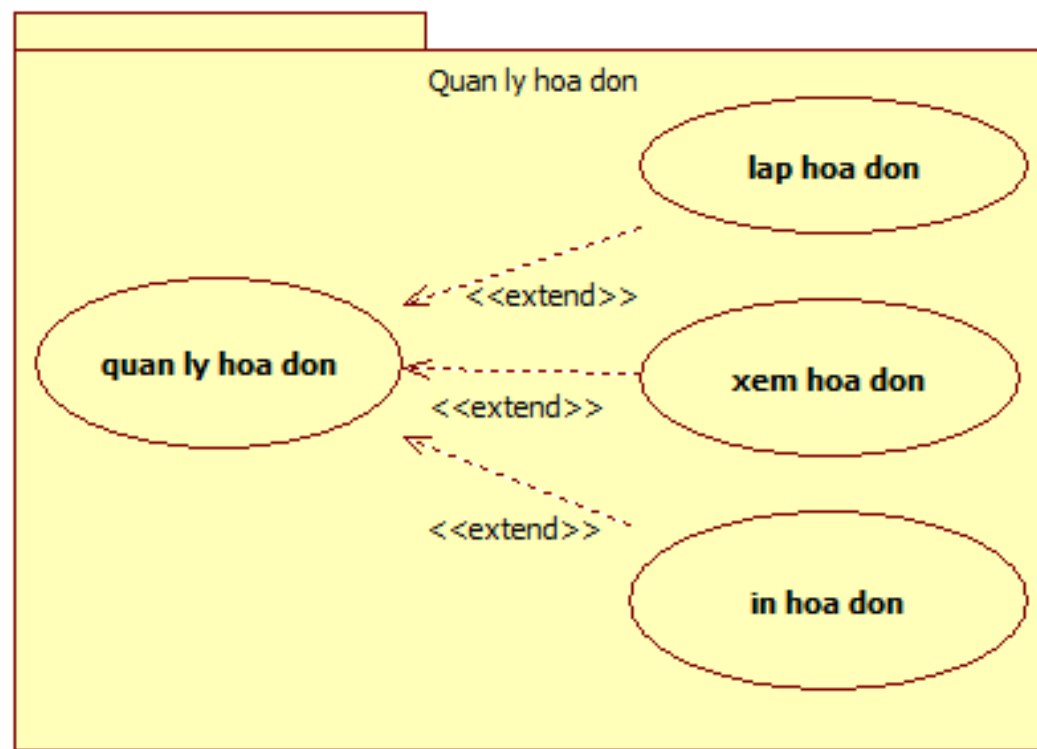
+ Quản lý hàng hóa: Xem danh sách hàng hóa. Thêm, xóa và sửa thông tin hàng hóa.





## VI. Ví dụ: Quản lý hóa đơn

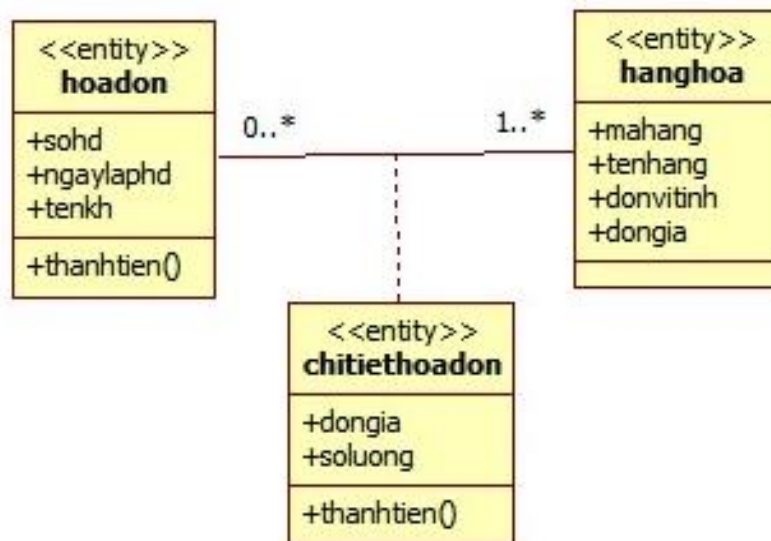
+ Quản lý hóa đơn: Xem danh sách hóa đơn. Lập hóa đơn, Xem chi tiết hóa đơn và In hóa đơn.



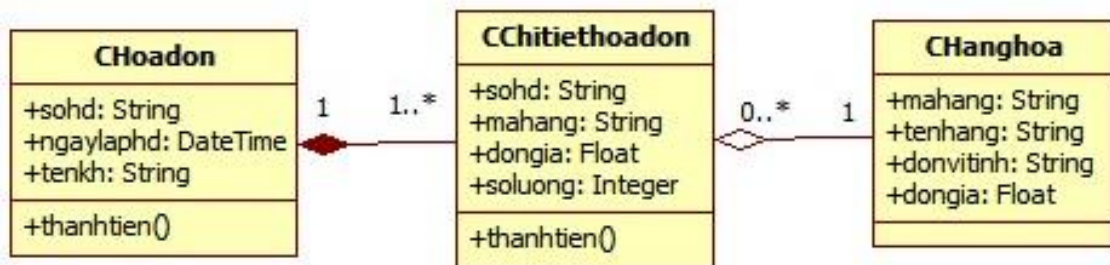
## VI. Ví dụ: Quản lý hóa đơn

3/ CSDL và CTDL:

+ CSDL



+ CTDL



## VI. Ví dụ: Quản lý hóa đơn

### + Cài đặt CTDL

```
class CChitietHoadon
{
    private int m_soluong;
    private double m_dongia;
    private CHangHoa m_hanghoa;
    2 references
    public int soluong...
    2 references
    public double dongia...
    4 references
    public CHangHoa hanghoa...
    1 reference
    public CChitietHoadon()...
    0 references
    public CChitietHoadon(CHangHoa hanghoa
        ,int soluong,double dongia)...
    2 references
    public double thanhtien()...
}
```

```
class CHangHoa
{
    private string m_mahang;
    private string m_tenhang;
    private string m_dvt;
    private double m_dongia;
    7 references
    public string mahang...
    8 references
    public string tenhang...
    8 references
    public string dvt...
    8 references
    public double dongia...
    0 references
    public CHangHoa(string mahang,
        string tenhang,string dvt,
        double dongia)...
    2 references
    public CHangHoa()...
}
```

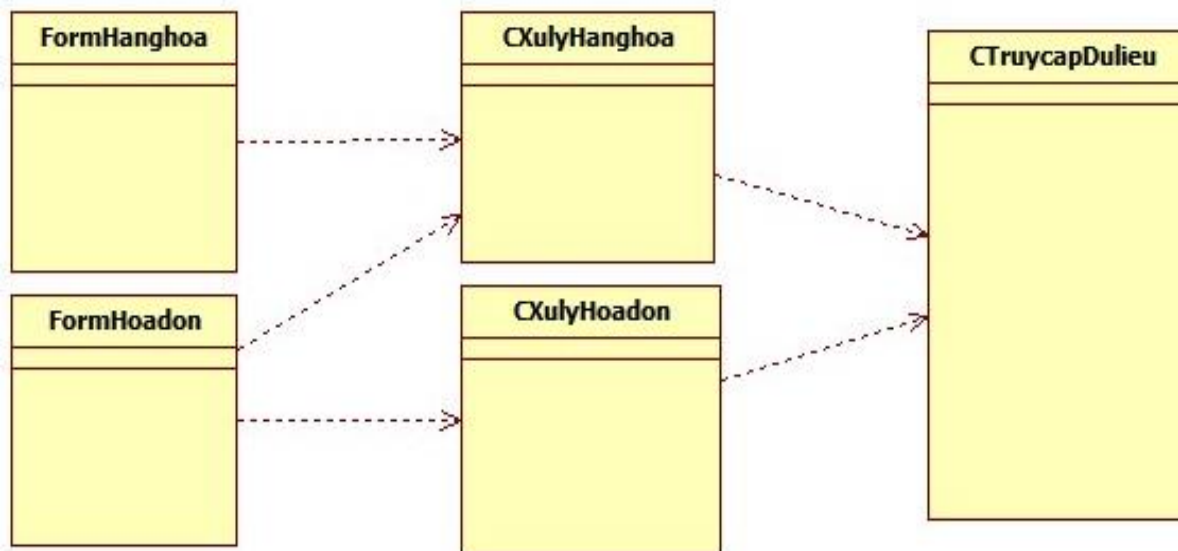


## VI. Ví dụ: Quản lý hóa đơn

```
class CHoaDon
{
    private string m_sohd;
    private DateTime m_ngaylaphd;
    private string m_tenk;
    private List<CChitietHoadon> m_cthd;
    5 references
    public string sohd...
    4 references
    public DateTime ngaylaphd...
    4 references
    public string tenkh...
    3 references
    public List<CChitietHoadon> DanhsachChitietHoadon...
    0 references
    public CHoaDon(string sohd,DateTime ngaylaphd,string tenkh)...
    2 references
    public CHoaDon()...
    0 references
    public double thanhtien()...
}
```

## VI. Ví dụ: Quản lý hóa đơn

### 4/ Kiến trúc ứng dụng: Sử dụng mẫu 3-Tiers





## VI. Ví dụ: Quản lý hóa đơn

### 5/ Cài đặt tầng 3 – Lớp CTruycapDulieu: Sử dụng mẫu Singleton

```
class CTruycapDulieu
{
    private static CTruycapDulieu m_data = null;
    private List<CHangHoa> m_dshh;
    private List<CHoaDon> m_ds hd;
    1 reference
    private CTruycapDulieu()...
    2 references
    public static CTruycapDulieu khoitao()...
    1 reference
    public List<CHangHoa> getDSHanghoa()...
    1 reference
    public List<CHoaDon> getDSHoadon()...
    1 reference
    public static bool ghiFile(string tenfile)...
    1 reference
    public static bool docFile(string tenfile)...
}
```





## VI. Ví dụ: Quản lý hóa đơn

6/ Cài đặt chức năng Quản lý hàng hóa:

- + Tầng 1: Giao diện là lớp FormHanghoa
- + Tầng 2: Xử lý dữ liệu là lớp CXulyHanghoa
- + Tầng 3: Truy cập dữ liệu là lớp CTruycapDulieu





## VI. Ví dụ: Quản lý hóa đơn

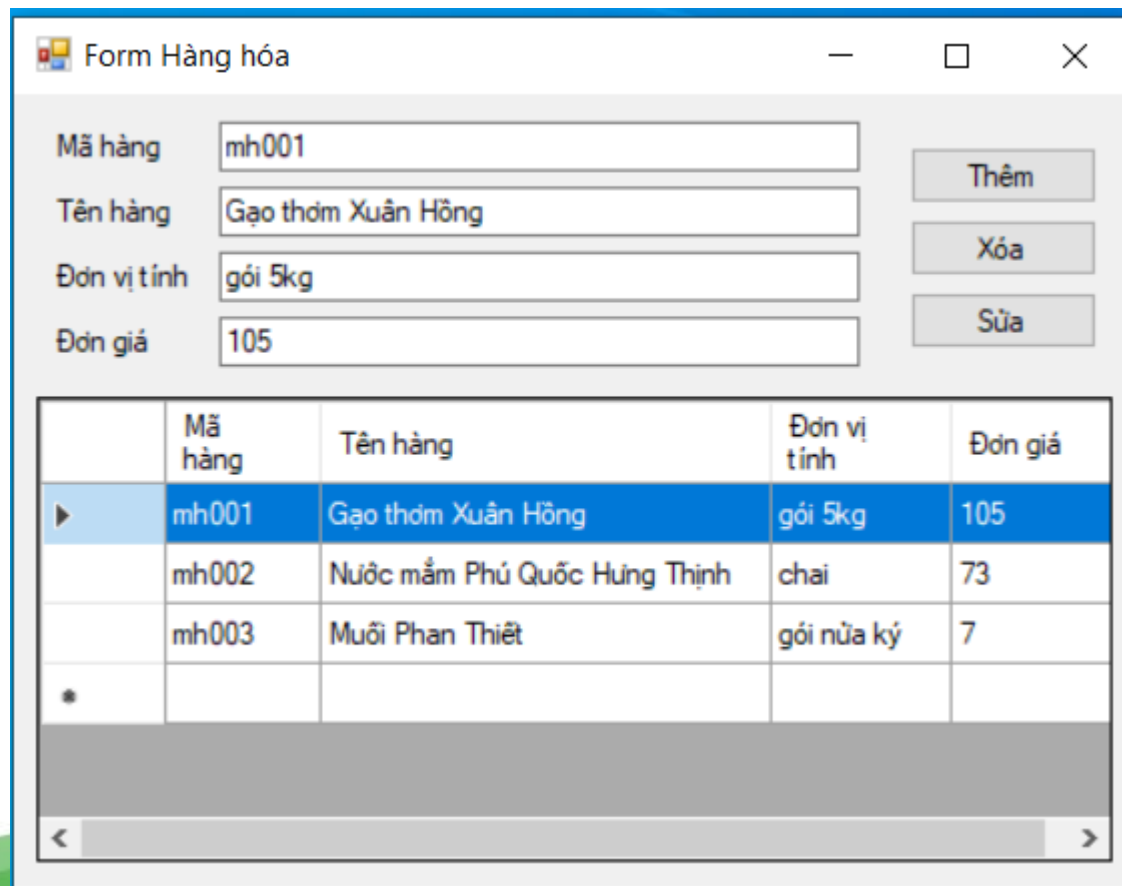
+ Tầng 2: Xử lý dữ liệu là lớp CXulyHanghoa

```
class CXulyHanghoa
{
    private List<CHangHoa> ds;
    2 references
    public CXulyHanghoa()...
    6 references
    public List<CHangHoa> getDSHanghoa()...
    5 references
    public CHangHoa tim(string mahang)...
    1 reference
    public void them(CHangHoa a)...
    1 reference
    public void xoa(string mahang)...
    1 reference
    public void sua(CHangHoa a)...
}
```



## VI. Ví dụ: Quản lý hóa đơn

+ Tầng 1: Giao diện là lớp FormHanghoa



Form Hàng hóa

Mã hàng:

Tên hàng:

Đơn vị tính:

Đơn giá:

Thêm

Xóa

Sửa

	Mã hàng	Tên hàng	Đơn vị tính	Đơn giá
▶	mh001	Gạo thơm Xuân Hồng	gói 5kg	105
	mh002	Nước mắm Phú Quốc Hưng Thịnh	chai	73
	mh003	Muối Phan Thiết	gói nửa ký	7
*				



## VI. Ví dụ: Quản lý hóa đơn

### Lớp tương ứng với giao diện

```
public partial class FormHanghoa : Form
{
    private CXulyHanghoa xuly;
    1 reference
    public FormHanghoa()...
    1 reference
    private void FormHanghoa_Load(object sender, EventArgs e)...
    4 references
    private void hienthi(List<CHangHoa> ds)...
    1 reference
    private void btnThem_Click(object sender, EventArgs e)...
    1 reference
    private void btnXoa_Click(object sender, EventArgs e)...
    1 reference
    private void dgv_RowEnter(object sender, DataGridViewCellEventArgs e)...
    1 reference
    private void btnSua_Click(object sender, EventArgs e)...
}
```





## VI. Ví dụ: Quản lý hóa đơn

7/ Cài đặt chức năng Quản lý hóa đơn:

- + Tầng 1: Giao diện là lớp FormHoadon
- + Tầng 2: Xử lý dữ liệu là lớp CXulyHoadon và lớp CXulyHanghoa
- + Tầng 3: Truy cập dữ liệu là lớp CTruycapDulieu



## + Tầng 2: Xử lý dữ liệu và lớp CXulyHoadon

```
class CXulyHoadon
{
    private List<CHoaDon> ds;
    1 reference
    public CXulyHoadon()...
    2 references
    public List<CHoaDon> getDSHoadon()...
    1 reference
    public CHoaDon tim(string sohd)...
    1 reference
    public void them(CHoaDon a)...
}
```



## VI. Ví dụ: Quản lý hóa đơn

### + Tầng 1: Giao diện là lớp FormHoadon

FormHoadon

Số hóa đơn:  Ngày lập hóa đơn:  Tên khách hàng:

Mã hàng:  Tên hàng:  Đơn vị tính:  Đơn giá:  Số lượng:

	Mã hàng	Tên hàng	Đơn vị tính	Đơn giá	Số lượng
▶	mh001	Gạo thơm Xuân Hồng	gói 5kg	105	2
	mh002	Nước mắm Phú Quốc Hư...	chai	73	1

Danh mục hóa đơn:

	Số hóa đơn	Ngày lập hóa đơn	Tên khách hàng
▶	hd001	8/22/2021 9:14 ...	Trần Trung
	hd002	8/22/2021 9:19 ...	Lê Hằng



## VI. Ví dụ: Quản lý hóa đơn

### Lớp FormHoadon tương ứng với giao diện

```
public partial class FormHoadon : Form
{
    private CXulyHanghoa x1HH;
    private CXulyHoadon x1HD;
    private CHoaDon hd;
    1 reference
    public FormHoadon()...
    1 reference
    private void FormHoadon_Load(object sender, EventArgs e)...
    2 references
    private void hienthi(List<CHoaDon> ds)...
    1 reference
    private void cmbMahang_Click(object sender, EventArgs e)...
    1 reference
    private void cmbMahang_SelectedIndexChanged(object sender, EventArgs e)...
    1 reference
    private void btnChonhh_Click(object sender, EventArgs e)...
    1 reference
    private void btnLaphd_Click(object sender, EventArgs e)...
    1 reference
    private void btnXem_Click(object sender, EventArgs e)...
}
```

# Tài liệu tham khảo

- 1) Lập trình hướng đối tượng, Trần Đan Thư - Đinh Bá Tiến - Nguyễn Tấn Trần Minh Khang, 2010, Nhà xuất bản khoa học và kỹ thuật.
- 2) A Programmer's Introduction to C#, Eric Gunnerson, 2000, Apress.
- 3) Programming C#, Jesse Liberty, 2002, O'Reilly.