

TRƯỜNG ĐẠI HỌC THÔNG TIN LIÊN LẠC
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO

MÔN: PHÁT TRIỂN ỨNG DỤNG MÃ NGUỒN MỞ

ĐỀ TÀI: QUẢN LÝ SINH VIÊN
TRƯỜNG ĐẠI HỌC THÔNG TIN LIÊN LẠC

GVHD : MAI CƯỜNG THỌ
LỚP : ĐHCN1C
NHÓM 5 : 1. HUỲNH NHẬT VỸ
2. NGUYỄN QUYẾT THỜI

Khánh Hòa, 07/2017

TRƯỜNG ĐẠI HỌC THÔNG TIN LIÊN LẠC
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO

MÔN: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER

ĐỀ TÀI: QUẢN LÝ PHÒNG KHÁM

GVHD : PHẠM THỊ KIM NGOAN

LỚP : ĐHCN1C

NHÓM 5 : 1. HUỖNH NHẬT VỸ
2. NGUYỄN QUYẾT THỜI
3. HOÀNG VĂN DŨNG

Khánh Hòa, 07/2017

NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI NÓI ĐẦU

Ngày nay, tin học đã có những bước tiến nhanh chóng về ứng dụng của nó trong mọi lĩnh vực của cuộc sống trên phạm vi toàn thế giới nói chung và Việt Nam nói riêng. Tin học được người ta quan tâm và nhắc đến nhiều hơn bao giờ hết vì nó là một phần không thể thiếu trong cuộc sống văn minh, góp phần đẩy mạnh công cuộc công nghiệp hoá hiện đại hoá đất nước, tiến đến nền kinh tế tri thức. Máy vi tính cùng với những phần mềm là công cụ đắc lực giúp ta quản lý, tổ chức, sắp xếp và xử lý công việc một cách nhanh chóng và chính xác. Ở Việt Nam hiện nay, máy tính điện tử đặc biệt là máy vi tính trong nhiều năm qua đã được sử dụng rất rộng rãi. Sự phát triển của tin học, các công nghệ phần mềm, phần cứng, các tài liệu tham khảo đã đưa chúng ta từng bước tiếp cận với công nghệ thông tin trong mọi lĩnh vực nhằm đáp ứng nhu cầu của con người. Quản lý sinh viên là một đề tài không còn mới mẻ với các bài toán quản lý. Việc đưa tin học vào ứng dụng để quản lý là rất hữu ích, vì chúng ta phải bỏ ra rất ít thời gian mà lại thu được hiệu quả cao, rất chính xác và tiện lợi nhanh chóng. Trong phạm vi bài báo cáo này nhóm chúng em xin được đề cập đến vấn đề “Quản lý sinh viên” ở trường Đại học Thông tin Liên lạc bằng máy vi tính, cụ thể là sử dụng ngôn ngữ lập trình C#.

Với khoảng thời gian không nhiều, vừa phân tích thiết kế, nghiên cứu tìm hiểu khai thác ngôn ngữ mới, vừa thực hiện chương trình quả là khó khăn đối với chúng em. Bởi “Quản lý sinh viên” là một đề tài có nội dung rộng, mặt khác khả năng am hiểu về hệ thống của nhóm em vẫn còn nhiều hạn chế. Xong cùng với sự nỗ lực của nhóm và sự quan tâm giúp đỡ tận tình của Thầy Mai Cường Thọ _Giảng viên Khoa CNTT Trường ĐH Nha Trang đã giúp nhóm em hoàn thành bài tập của mình theo đúng thời gian quy định. Tuy nhiên trong quá trình làm vẫn còn có nhiều sai sót nên chúng em rất mong nhận được những ý kiến đóng góp của thầy cùng toàn thể các bạn trong lớp để bài tập của chúng em được hoàn thiện.

NHÓM THỰC HIỆN

MỤC LỤC

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	6
I. Ngôn ngữ lập trình C#	6
II. Lập trình hướng đối tượng với C#	12
CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG.....	20
I. Phân tích chức năng.....	20
II. Cơ sở dữ liệu	20
1.Quan hệ dữ liệu.....	20
2. Các bảng	20
2.1. Bảng Khoa	20
2.2 Bảng sinh viên	21
2.3 Bảng Kết quả	21
2.4 Bảng Môn học.....	22
III. Các lớp	22
1. Lớp Khoa	22
2. Lớp Sinh viên.....	22
3.Lớp Kết quả	23
4. Lớp Môn học.....	24
5. Lớp clsDuLieu	24
CHƯƠNG 3: TRIỂN KHAI LẬP TRÌNH.....	26
1.Lớp Khoa	26
2 Lớp Sinh viên.....	27
4 Lớp clsDuLieu	29
CHƯƠNG 4: MINH HỌA ỨNG DỤNG	32
I. Form Trang chủ quản lý sinh viên.....	32
II. Form Thông tin Khoa.....	32
III. Form Thông tin sinh viên.....	33
IV. Form Đăng ký môn học	34
V. Form Tìm kiếm sinh viên.....	34
VI.Form Quản lý khoa	35

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

I. Ngôn ngữ lập trình C#

1. Giới thiệu

Ngôn ngữ C# là một ngôn ngữ được dẫn xuất từ C và C++, nhưng nó được tạo từ nền tảng phát triển hơn. Microsoft bắt đầu với công việc trong C và C++ và thêm vào những đặc tính mới để làm cho ngôn ngữ này dễ sử dụng hơn. Nhiều trong số những đặc tính này khá giống với những đặc tính có trong ngôn ngữ Java. Không dừng lại ở đó, Microsoft đưa ra một số mục đích khi xây dựng ngôn ngữ này. Những mục đích này được tóm tắt như sau:

- C# là ngôn ngữ đơn giản
- C# là ngôn ngữ hiện đại
- C# là ngôn ngữ hướng đối tượng
- C# là ngôn ngữ mạnh mẽ và mềm dẻo
- C# là ngôn ngữ có ít từ khóa
- C# là ngôn ngữ hướng module
- C# sẽ trở nên phổ biến

2. Các kiểu dữ liệu

Tương tự như C++ hay Java, C# chia thành hai tập hợp kiểu dữ liệu chính: Kiểu xây dựng sẵn (built-in) mà ngôn ngữ cung cấp cho người lập trình và kiểu được người dùng định nghĩa (user-defined) do người lập trình tạo ra

C# phân tập hợp kiểu dữ liệu này thành hai loại: Kiểu dữ liệu giá trị (value) và kiểu dữ liệu tham chiếu (reference). Việc phân chia này do sự khác nhau khi lưu kiểu dữ liệu giá trị và kiểu dữ liệu tham chiếu trong bộ nhớ. Đối với một kiểu dữ liệu giá trị thì sẽ được lưu giữ kích thước thật trong bộ nhớ đã cấp phát là stack. Trong khi đó kiểu dữ liệu tham chiếu như các đối tượng được cấp phát trên heap. Khi một đối tượng được cấp phát trên heap thì địa chỉ của nó được trả về, và địa chỉ này được gắn đến một tham chiếu.

Bảng 1-1 Kiểu dữ liệu xây dựng sẵn

Kiểu C#	Số byte	Kiểu .NET	Mô tả
byte	1	Byte	Số nguyên dương không dấu từ 0-255

char	2	Char	Kí tự Unicode
bool	1	Boolean	Giá trị logic true/ false
sbyte	1	Sbyte	Số nguyên có dấu (từ -128 đến 127)
short	2	Int16	Số nguyên có dấu giá trị từ -32768 đến 32767
ushort	2	Int16	Số nguyên không dấu 0 – 65.535
int	4	Int32	Số nguyên có dấu –2.147.483.647 và 2.147.483.647
uint	4	UInt32	Số nguyên không dấu 0 – 4.294.967.295
float	4	Single	Kiểu dấu chấm động, giá trị xấp xỉ từ 3,4E-38 đến 3,4E+38, với 7 chữ số có nghĩa..
double	8	Double	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308, với 15,16 chữ số có nghĩa
decimal	8	Decimal	Có độ chính xác đến 28 con số và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố “m” hay “M” theo sau giá trị.
long	8	Int64	Kiểu số nguyên có dấu có giá trị trong khoảng : -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807
ulong	8	UInt64	Số nguyên không dấu từ 0 đến 0xffffffffffffffff

Chọn một kiểu định sẵn

Tuỳ vào từng giá trị muốn lưu trữ mà ta chọn kiểu cho phù hợp. Nếu chọn kiểu quá lớn so với các giá trị cần lưu sẽ làm cho chương trình đòi hỏi nhiều bộ nhớ và chạy chậm. Trong khi nếu giá trị cần lưu lớn hơn kiểu thực lưu sẽ làm cho giá trị các biến bị sai và chương trình cho kết quả sai.

Kiểu char biểu diễn một ký tự Unicode. Ví dụ “\u0041” là ký tự “A” trên bảng Unicode. Một số ký tự đặc biệt được biểu diễn bằng dấu “\” trước một ký tự khác

Kiểu dữ liệu do người dùng định nghĩa

- Tất cả kiểu dữ liệu do người dùng định nghĩa ngoài trừ kiểu cấu trúc đều là kiểu dữ liệu tham chiếu

Một số kiểu dữ liệu do người dùng định nghĩa gồm

- **object**: đây là kiểu dữ liệu cơ sở chứa tất cả các kiểu dữ liệu khác trong C#.
- **string**: kiểu dữ liệu chuỗi ký tự.
- **class**: kiểu dữ liệu class.
- **delegate**: kiểu dữ liệu chuyển giao.
- **interface**: kiểu dữ liệu giao tiếp.
- **array**: kiểu dữ liệu mảng.

3. Các toán tử

Toán tử được kí hiệu bằng một biểu tượng dùng để thực hiện một hành động. Các kiểu dữ liệu cơ bản của C# như kiểu nguyên hỗ trợ rất nhiều các toán tử như toán tử gán, toán tử toán học, logic....

Các phép toán +, -, *, / là một ví dụ về toán tử. Áp dụng các toán tử này lên các biến kiểu số ta có kết quả như việc thực hiện các phép toán thông thường.

```
int a = 10;  
int b = 20;  
int c = a + b; // c = 10 + 20 = 30
```

C# cung cấp cấp nhiều loại toán tử khác nhau để thao tác trên các kiểu biến dữ liệu, được liệt kê trong bảng sau theo từng nhóm ngữ nghĩa.

Bảng 1.2 Các nhóm toán tử trong C#

Nhóm toán tử	Toán tử	Ý nghĩa
Toán học	+ - * / %	Cộng, trừ, nhân, chia, chia lấy phần dư
Logic	& ^ ! ~ && true false	Phép toán logic và thao tác trên bit
Ghép chuỗi	+	Ghép nối hai chuỗi
Tăng, giảm	++, --	Tăng/giảm toán hạng lên/xuống 1. Đứng trước hoặc sau toán hạng
Dịch bit	<< >>	Dịch trái, dịch phải
Quan hệ	== != < > <= >=	Bằng, khác, nhỏ hơn, lớn hơn, nhỏ hơn hoặc bằng, lớn hơn hoặc bằng

Gán	= += -= *= /= %= &= != ^= <<= >>=	Phép gán
Chỉ số	[]	Cách truy xuất phần tử của mảng
Ép kiểu	()	
Indirection và Address	* -> [] &	Dùng cho con trỏ

3.1 Toán tử gán (=)

Toán tử gán (=) Toán tử này cho phép thay đổi các giá trị của biến bên phải toán tử bằng giá trị bên trái toán tử.

3.2 Nhóm toán tử toán học

C# dùng các toán tử số học với ý nghĩa theo đúng tên của chúng như: + (cộng), - (trừ) * (nhân) và / (chia). Tùy theo kiểu của hai toán hạng mà toán tử trả về kiểu tương ứng. Ngoài ra, còn có toán tử % (lấy phần dư) được sử dụng trong các kiểu số nguyên.

3.3 Các toán tử tăng và giảm

C# cũng kế thừa từ C++ và Java các toán tử: +=, -=, *=, /=, %= nhằm làm đơn giản hoá. Nó còn kế thừa các toán tử tiền tố và hậu tố (như biến++, hay ++biến) để giảm bớt sự cồng kềnh trong các toán tử cổ điển.

3.4 Các toán tử quan hệ

Các toán tử quan hệ được dùng để so sánh hai giá trị với nhau và kết quả trả về có kiểu Boolean. Toán tử quan hệ gồm có: == (so sánh bằng), != (so sánh khác), > (so sánh lớn hơn), >= (lớn hơn hay bằng), < (so sánh nhỏ hơn), <= (nhỏ hơn hay bằng).

3.5 Các toán tử logic

Các toán tử logic gồm có: && (và), || (hoặc), ! (phủ định). Các toán tử này được dùng trong các biểu thức điều kiện để kết hợp các toán tử quan hệ theo một ý nghĩa nhất định.

3.6 Thứ tự các toán tử

Đối với các biểu thức toán, thứ tự ưu tiên là thứ tự được qui định trong toán học. Còn thứ tự ưu tiên thực hiện của các nhóm toán tử được liệt kê theo bảng dưới đây

Bảng 1-3 Thứ tự ưu tiên của các nhóm toán tử (chiều ưu tiên từ trên xuống)

Nhóm toán tử	Toán tử	Ý nghĩa
Primary(chính)	{ x } x.y f(x) a[x] x++ x--	

Unary	+ - ! ~ ++x -x (T)x	
Nhân	* / %	Nhân, chia, chia lấy phần dư
Cộng	+ -	cộng, trừ
Dịch bit	<< >>	Dịch trái dịch phải
Bằng	== !=	Bằng, khác
Logic trên bit AND	&	Và trên bit
XOR	^	Xor trên bit
OR		Hoặc trên bit
Điều kiện AND	&&	Và trên biểu thức điều kiện
Điều kiện OR		Hoặc trên biểu thức điều kiện
Điều kiện	?:	Điều kiện tương tự if
Assignment	= *= /= %= += -= <<= =>> &= ^= =	

3.7 Toán tử tam phân

Cú pháp:

<biểu thức điều kiện>? <biểu thức 1>: <biểu thức 2>;

Ý nghĩa:

- Nếu biểu thức điều kiện đúng thì thực hiện biểu thức 1.
- Nếu sai thì thực hiện biểu thức 2.

4. Cấu trúc rẽ nhánh

a. Câu lệnh if... else ...

Cú pháp:

```

if ( biểu thức logic )
    khối lệnh;
hoặc
if ( biểu thức logic )
    khối lệnh 1;
else
    khối lệnh 2;
```

Ghi chú: Khối lệnh là một tập các câu lệnh trong cặp dấu "{...}". Bất kỳ nơi đâu có câu lệnh thì ở đó có thể viết bằng một khối lệnh.

Biểu thức logic là biểu thức cho giá trị đúng hoặc sai (**true** hoặc **false**). Nếu “*biểu thức logic*” cho giá trị đúng thì “*khối lệnh*” hay “*khối lệnh 1*” sẽ được thực thi, ngược lại “*khối lệnh 2*” sẽ thực thi. Một điểm khác biệt với C++ là biểu thức trong câu lệnh if phải là biểu thức logic, không thể là biểu thức số.

b. Câu lệnh switch case

Cú pháp:

```
switch ( biểu_thức_lựa_chọn )
{
    case biểu_thức_hằng :
        khối_lệnh;
        lệnh_nhảy;
    [ default :
        khối_lệnh;
        lệnh_nhảy; ]
}
```

Biểu thức lựa chọn là biểu thức sinh ra trị nguyên hay chuỗi. Switch sẽ so sánh biểu_thức_lựa_chọn với các biểu_thức_hằng để biết phải thực hiện với khối lệnh nào. Lệnh nhảy như break, goto...để thoát khỏi câu switch và bắt buộc phải có.

```
int nQuyên = 0;
switch ( sQuyênTruyCap )
{
    case "Administrator":
        nQuyên = 1;
        break;
    case "Admin":
        goto case "Administrator";
    default:
        nQuyên = 2;
        break;
}
```

c. Câu lệnh for

Cú pháp:

```
for ( [khởi_tạo_biến_đếm]; [biểu_thức]; [gia_tăng_biến_đếm] )
```

khởi lệnh;

Ví dụ 3-4 Tính tổng các số nguyên từ a đến b

```
int a = 10; int b = 100; int nTong = 0;
```

d. Câu lệnh while

Cú pháp:

```
while ( biểu_thức_logic )  
    khởi_lệnh;
```

Khởi_lệnh sẽ được thực hiện cho đến khi nào biểu thức còn đúng. Nếu ngay từ đầu biểu thức sai, khởi lệnh sẽ không được thực thi.

e. Câu lệnh do ...while

Cú pháp:

```
do  
    khởi_lệnh  
while ( biểu_thức_logic )
```

Khác với while khởi lệnh sẽ được thực hiện trước, sau đó biểu thức được kiểm tra. Nếu biểu thức đúng khởi lệnh lại được thực hiện.

II. Lập trình hướng đối tượng với C#

1. Lớp và đối tượng

1.1 Đối tượng

Trong lập trình hướng đối tượng, tất cả thực thể trong hệ thống đều được coi là các đối tượng cụ thể. Đối tượng là một thực thể hoạt động khi chạy chương trình.

Một đối tượng là một thực thể đang tồn tại trong hệ thống và được xác định bằng ba yếu tố:

- **Định danh đối tượng:** xác định duy nhất cho mỗi đối tượng trong hệ thống, nhằm phân biệt các đối tượng với nhau.
- **Trạng thái của đối tượng:** là sự tổ hợp của các giá trị của các thuộc tính mà đối tượng đang có.
- **Hoạt động của đối tượng:** là các hành động mà đối tượng có khả năng thực hiện được.

Trạng thái hiện tại của đối tượng quy định tính chất đặc trưng của đối tượng. Ví dụ, đối tượng xe có trạng thái là:

- Nhãn hiệu xe là Ford
- Màu xe là màu trắng
- Giá bán xe là 5000\$

Mỗi đối tượng sẽ có một số hành động gọi là phương thức. Ví dụ xe có phương thức :

- Khởi động
- Dừng lại
- Chạy

1.2 Class (lớp)

Định nghĩa lớp: Trong lập trình hướng đối tượng, đối tượng là một thực thể cụ thể, tồn tại trong hệ thống. Trong khi đó, lớp là một khái niệm trừu tượng, dùng để chỉ một tập hợp các đối tượng cùng loại.

Ví dụ: Trong bài toán quản lý xe hơi của một cửa hàng kinh doanh xe, mỗi chiếc xe có mặt trong cửa hàng được coi là một đối tượng. Khi đó ta có khái niệm lớp “Xe hơi” chỉ tất cả các loại xe hơi đang có

Khai báo một lớp

Định nghĩa một lớp mới với cú pháp như sau:

```
[attribute][bổ từ truy xuất] class định danh [:lớp cơ sở]
{
    thân lớp
}
```

Ví dụ Khai báo một lớp

```
public class XeHoi
{
    private string tenxe;
    private string nhanhieu;
    private string mauxe;

    public string TenXe
    {
        get { return tenxe; }
        set { tenxe = value; }
    }
}
```

```

public string NhanHieu
{
    get { retrurn nhanhieu;}
    set { nhanhieu = value;}
}
public string MauXe
{
    get { retrurn mauxe;}
    set { mauxe = value;}
}
}

```

Lớp và đối tượng

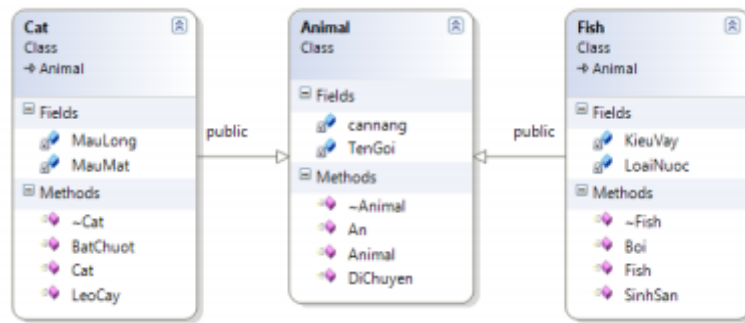
Lớp và đối tượng mặc dù có mối tương quan, nhưng bản chất lại khác nhau:

- Lớp là sự trừu tượng hóa các đối tượng. Trong khi đó, đối tượng là thể hiện cụ thể của một lớp.
- Đối tượng là một thực thể cụ thể, có thực, tồn tại trong hệ thống. Trong khi đó lớp là một khái niệm trừu tượng, chỉ tồn tại ở dạng khái niệm để mô tả các đặc tính chung của một nhóm đối tượng.
- Tất cả các đối tượng thuộc cùng một lớp giống nhau về thuộc tính và phương thức.

2. Kế thừa lớp

Một tính năng then chốt của lập trình hướng đối tượng đó là tính kế thừa. Nhờ vào tính kế thừa, nó cho phép một lớp có thể dẫn xuất từ một lớp khác hay nói cách khác một lớp có thể thừa có thể sử dụng lại các thuộc tính và phương thức của lớp bị kế thừa, chính vì thế chúng sẽ tự động tiếp nhận các thành viên của bố mẹ và bổ sung thêm các thành viên của riêng chúng. Tính kế thừa cho phép lớp mới có thể nhận được mọi dữ liệu thành viên (private, protected, public) và các hàm thành viên (trừ hàm tạo, hàm hủy, hàm bạn và hàm toán tử gán =).

Ví dụ: Lớp động vật Animal có các thuộc tính thành viên: tên gọi, cân nặng. Các hàm thành viên: di chuyển, ăn. Ta xét hai lớp dẫn xuất của nó là lớp mèo Cat và lớp cá Fish. Lớp Cat có các thuộc tính thành viên riêng: màu lông, màu mắt. Các hàm thành viên riêng: Bắt chuột, Leo cây. Lớp Fish có các thuộc tính thành viên riêng: kiểu vảy, loại nước (nước ngọt, nước mặn, nước lợ). Các hàm thành viên : bơi, sinh sản (cách thức sinh con như thế nào).



Khai báo một lớp kế thừa

Trong C#, khi ta tạo một lớp kế thừa bằng cách công một thêm dấu “:” và sau tên của lớp kế thừa và theo sau đó là lớp cơ sở như sau:

```
public class <Tên lớp dẫn suất>: <Tên lớp cơ sở>
```

Ví dụ

```
class Animal
{
    private string cannang;
    private string tengoi;
    public void An
    {
        ...
    }
    public void DiChuyen
    {
        ...
    }
}
class Cat:Aminal
{
    private string maulong;
    private string maumat;
    pulic string MauLong
    {
        ...
    }
    pulic string MauMat
    {
        ...
    }
}
```

}

3. Giao diện (interface)

Giao diện giống như một khuôn mẫu, các class sử dụng giao diện đều phải thực hiện tất cả các phương thức của giao diện, điều này giúp đồng nhất về phương thức giúp các lớp khác nhau có thể làm việc được với nhau.

Cú pháp của việc định nghĩa một giao diện:

```
[attributes] [access-modifier] interface interface-name [:base-list]
{
    interface-body
}
```

Ý nghĩa của từng thành phần như sau

- **attributes:** sẽ đề cập ở phần sau.
- **modifiers:** bỏ từ phạm vi truy xuất của giao diện
- **identifier:** tên giao diện muốn tạo
- **base-list:** danh sách các giao diện mà giao diện này thừa kế,
- (nói rõ trong phần thừa kế)
- **interface-body:** thân giao diện luôn nằm giữa cặp dấu { }

***Lưu ý:** Tên giao diện thường bắt đầu bằng chữ I (in hoa)*

Ví dụ:

```
interface IStorable
{
    void Read( );
    void Write(object);
}
```

Sử dụng giao diện vừa tạo ra

```
// lớp Document thừa kế IStorable,
// phải cài đặt tất cả các phương thức của IStorable
public class Document : IStorable
{
    public void Read( ) { // phải cài đặt...}
```



```

public void Write(object obj) { // phải cài đặt...}
// ...
}

```

Cài đặt nhiều giao diện

Lớp có thể cài đặt một hoặc nhiều giao diện. Chẳng hạn như ở lớp Document ngoài lưu trữ ra nó còn có thể được nén lại.

Ta cho lớp Document cài đặt thêm một giao diện thứ hai là Icompressible, Document phải cài đặt tất cả phương thức của Icompressible và IStorable:

```

public class Document : IStorable, Icompressible
{
    public void Compress( )
    {
        Console.WriteLine("Implementing the Compress Method");
    }
    public void Decompress( )
    {
        Console.WriteLine("Implementing the Decompress Method");
    }
}

```

Mở rộng giao diện

Chúng ta có thể mở rộng (thừa kế) một giao diện đã tồn tại bằng cách thêm vào đó những phương thức hoặc thành viên mới. Chẳng hạn như ta có thể mở rộng ICompressable thành ILoggedCompressable với phương thức theo dõi những byte đã được lưu:

```

interface ILoggedCompressible : ICompressible
{
    void LogSavedBytes( );
}

```

Lớp cài đặt phải cân nhắc chọn lựa giữa 2 lớp ICompressable hay ILoggedCompressable, điều này phụ thuộc vào nhu cầu của lớp đó. Nếu một lớp có sử dụng giao diện ILoggedCompressable thì nó phải thực hiện toàn bộ các phương thức của ILoggedCompressable (bao gồm ICompressable và phương thức mở rộng).

Kết hợp các giao diện khác nhau

Tương tự, chúng ta có thể tạo một giao diện mới bằng việc kết hợp nhiều giao diện và ta có thể tùy chọn việc có thêm những phương thức hoặc những thuộc tính mới.

Ví dụ như ta tạo ra giao diện `IStorableCompressable` từ giao diện `IStorable` và `ILoggedCompressable` và thêm vào một phương thức mới dùng để lưu trữ kích thước tập tin trước khi nén.

```
interface IStorableCompressable: IStorable, ILoggedCompressable
{
    void LogOriginalSize( );
}
```

4. Không gian tên (Namespace)

Namespace trong ngôn ngữ C#, nhằm tránh sự xung đột giữa việc sử dụng các thư viện khác nhau từ các nhà cung cấp. Ngoài ra, namespace được xem như là tập hợp các lớp đối tượng, và cung cấp duy nhất các định danh cho các kiểu dữ liệu và được đặt trong một cấu trúc phân cấp. Việc sử dụng namespace trong khi lập trình là một thói quen tốt, bởi vì công việc này chính là cách lưu các mã nguồn để sử dụng về sau. Ngoài thư viện namespace do MS.NET và các hãng thứ ba cung cấp, ta có thể tạo riêng cho mình các namespace. C# đưa ra từ khóa `using` để khai báo sử dụng namespace trong chương trình:

using < Tên namespace >

Để tạo một namespace dùng cú pháp sau:

```
namespace <Tên namespace>
{
    < Định nghĩa Lớp A>
    < Định nghĩa Lớp B >
    .....
}
```

Ví dụ

```
namespace MyLib
{
    using System;
    public class Tester
    {
        public static int Main()
        {
            for (int i =0; i < 10; i++)
```

```
        {  
            Console.WriteLine( "i: {0}", i);  
        }  
        return 0;  
    }  
}
```

Ví dụ trên tạo ra một namespace có tên là MyLib, bên trong namespace này chứa một lớp có tên là Tester. C# còn cho phép trong một namespace có thể tạo một namespace khác lồng bên trong và không giới hạn mức độ phân cấp này.

CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG

I. Phân tích chức năng

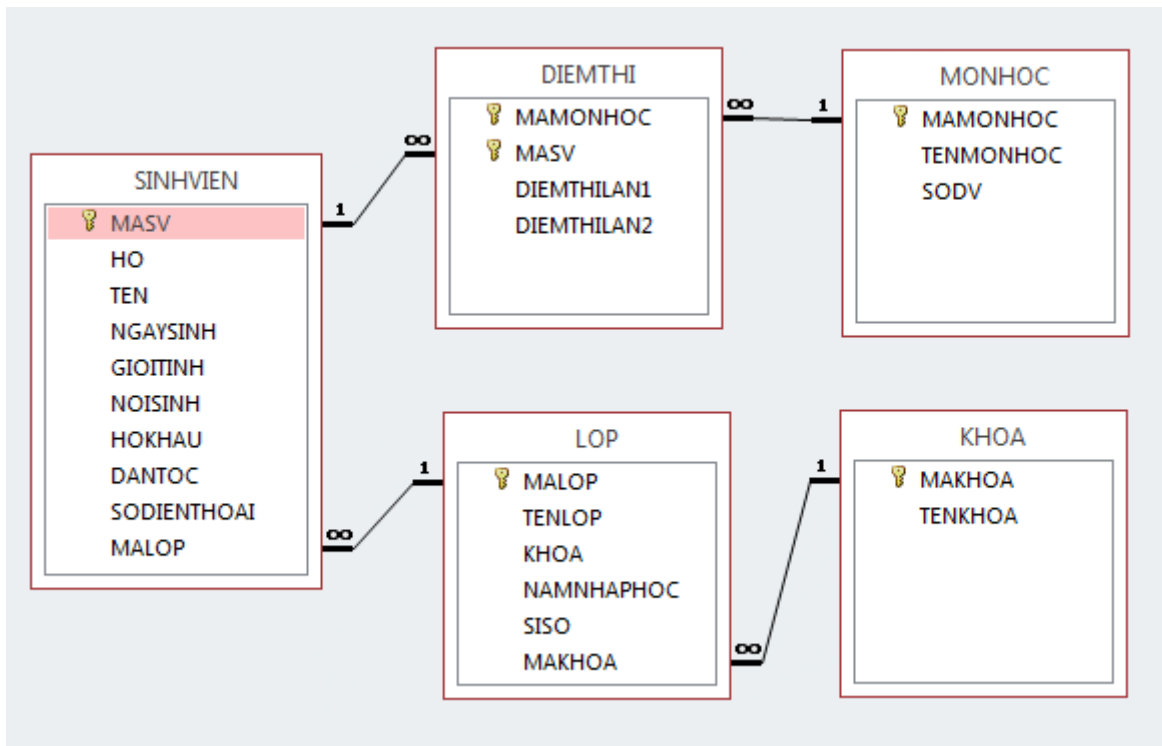
Ứng dụng quản lý sinh viên có thể giúp công việc quản lý sinh viên trở lên dễ dàng, nhanh chóng, độ chính xác cao.

Chức năng ứng dụng:

- Quản lý sinh viên: thêm mới, sửa, xóa, tìm kiếm sinh viên
- Quản lý điểm thi: nhập điểm, sửa điểm, xóa điểm thi của sinh viên
- Quản lý lớp: thêm mới, sửa xóa.
- Quản lý môn học: thêm, sửa xóa, tìm kiếm môn học
- Quản lý khoa: thêm, sửa xóa khoa

II. Cơ sở dữ liệu

1. Quan hệ dữ liệu



2. Các bảng

2.1. Bảng Khoa

KHOA		
Tên trường	Kiểu dữ liệu	Ghi chú
MaKhoa	Short Text (10)	Khóa chính Bắt buộc nhập

TenKhoa	Short Text (50)	
---------	-----------------	--

2.2 Bảng sinh viên

SINHVIEN		
Tên trường	Kiểu dữ liệu	Ghi chú
MaSV	Short Text (10)	Khóa chính Bắt buộc nhập
Ho	Short Text (50)	
Ten	Short Text (10)	
GioiTinh	Yes/No	
NoiSinh	Short Text (50)	
HoKhu	Short Text (50)	
DanToc	Short Text (10)	
SoDienThoai	Number (Long Integer)	
MaLop	Short Text (10)	

2.3 Bảng Kết quả

DIEMTHI		
Tên trường	Kiểu dữ liệu	Ghi chú
MaMonHoc	Short Text (10)	Khóa chính Bắt buộc nhập
MaSV	Short Text (10)	Khóa chính Bắt buộc nhập
DiemThiLan1	Double	
DiemThiLan2	Double	

2.4 Bảng Môn học

MONHOC		
Tên trường	Kiểu dữ liệu	Ghi chú
MaMonHoc	Short Text (10)	Khóa chính Bắt buộc nhập
TenMonHoc	Short Text (50)	
SoDV	Number(Long Integer)	Số đơn vị học phần

III. Các lớp

1. Lớp Khoa

KHOA		
Tên thuộc tính	Kiểu dữ liệu	Ghi chú
MaKhoa	string	
TenKhoa	string	
Tên phương thức	Kiểu dữ liệu trả về	Chức năng
Them()	void	Nhập một khoa vào DataBase
Sua()	void	Sửa một khoa trong DataBase
Xoa()	void	Xóa một khoa một DataBase

2. Lớp Sinh viên

SINHVIEN		
Tên thuộc tính	Kiểu dữ liệu	Ghi chú
MaSV	string	
Ho	string	
Ten	string	
GioiTinh	bool	

NgaySinh	string	
NoiSinh	string	
HoKhu	string	
DanToc	string	
SoDienThoai	int	
MaLop	string	
Tên phương thức	Kiểu dữ liệu trả về	Chức năng
Them()	void	Nhập một sinh viên vào DataBase
Sua()	void	Sửa một sinh viên trong DataBase
Xoa()	void	Xóa một sinh viên một DataBase

3.Lớp Kết quả

KETQUA		
Tên thuộc tính	Kiểu dữ liệu	Ghi chú
MaMonHoc	string	
MaSV	string	
DiemThiLan1	int	
DiemThiLan1	int	
Tên phương thức	Kiểu dữ liệu trả về	Chức năng
Them()	void	Nhập một điểm thi vào DataBase
Sua()	void	Sửa một điểm thi trong DataBase
Xoa()	void	Xóa một điểm thi một DataBase

4. Lớp Môn học

MONHOC		
Tên thuộc tính	Kiểu dữ liệu	Ghi chú
MaMonHoc	string	
TenMonHoc	string	
SoDonViHocPhan	int	
Tên phương thức	Kiểu dữ liệu trả về	Chức năng
Them()	void	Nhập một môn học vào DataBase
Sua()	void	Sửa một môn học trong DataBase
Xoa()	void	Xóa một môn học một DataBase

5. Lớp clsDuLieu

clsDuLieu		
Tên phương thức	Kiểu dữ liệu trả về	Chức năng
KetNoi()	void	Tạo kết nối đến DataBase
GetDataSet(String TenBang)	DataSet	Lấy dữ liệu từ 1 bảng đưa vào DataSet
GetDataSet(String TenBang,string TenCacTruong)	DataSet	Lấy dữ liệu từ 1 bảng đưa gồm nhiều trường được chỉ định và đưa vào DataSet
GetDataSetSqlCommand(string strSQL)	DataSet	Lấy dữ liệu theo lệnh SQL và đưa vào DataSet

GetDataSetDieuKien(string TenBang,string DieuKien)	DataSet	Lấy dữ liệu theo điều kiện và đưa vào DataSet
KiemTraMa(string DieuKien)	bool	Kiểm tra mã đã có trong bảng hay chưa
KiemTraMa(string TenBang,string DieuKien)	bool	Kiểm tra mã đã có trong bảng hay chưa
RunSQL(string strSQL)	bool	Chạy một câu lệnh SQL

CHƯƠNG 3: TRIỂN KHAI LẬP TRÌNH

1.Lớp Khoa

1.1 Thêm

```
void themKhoa(string maKhoa, string tenKhoa)
{
    SqlConnection connDB = new
SqlConnection(Khoa.strConn);
    connDB.Open();
    string cmd = "INSERT INTO KHOA VALUES('" + maKhoa + +
"',N'" + tenKhoa + "')";
    SqlCommand sqlCmd = new SqlCommand(cmd, connDB);
    sqlCmd.ExecuteNonQuery();
    connDB.Close();
}
```

1.2 Sửa

```
void suaKhoa(string maKhoa, string tenKhoa)
{
    SqlConnection connDB = new
SqlConnection(Khoa.strConn);
    connDB.Open(); string cmd = "UPDATE KHOA SET
TENKHOA='" + tenKhoa + "' WHERE MAKHOA='" + maKhoa + "'";
    SqlCommand sqlCmd = new SqlCommand(cmd, connDB);
    sqlCmd.ExecuteNonQuery();
    connDB.Close();
}
```

1.3 Xóa

```
void xoaKhoa(string maKhoa)
{
    SqlConnection connDB = new
SqlConnection(Khoa.strConn);
    connDB.Open();
    string cmd = "DELETE FROM KHOA WHERE MAKHOA='" +
maKhoa + "'";
    SqlCommand sqlCmd = new SqlCommand(cmd, connDB);
    sqlCmd.ExecuteNonQuery();
    connDB.Close();
}
```

2 Lớp Sinh viên

2.1 Thêm

```
public void Them(clsDuLieu db, SinhVien sv)
{
    // nếu Thông tin đã đầy đủ thì thực hiện lưu
    String stringSQL = "INSERT INTO SINHVIEN (";
    stringSQL += "
masv,ho,ten,ngaysinh,gioitinh,noisinh,hokhau,dantoc,sodienthoai,m
alop";

    stringSQL += ") VALUES (";
    stringSQL += "'" + sv.ma + "',";
    stringSQL += "'" + sv.ho + "',";
    stringSQL += "'" + sv.ten + "',";
    stringSQL += "'" + sv.ngaysinh + "',";
    if (sv.gioitinh == "True")
        stringSQL += "'1',";
    else stringSQL += "'0',";
    stringSQL += "'" + sv.noisinh + "',";
    stringSQL += "'" + sv.hokhau + "',";
    stringSQL += "'" + sv.dantoc + "',";
    stringSQL += "'" + sv.sodienthoai + "',";
    stringSQL += "'" + sv.malop + "'";
    stringSQL += ")";

    if (db.RunSQL(stringSQL))
        MessageBox.Show("Đã lưu");
    else MessageBox.Show("Chưa lưu");
}
```

2.2 Sửa

```
public void Sua(clsDuLieu db, SinhVien objsv)
{
    String stringSQL = "UPDATE SINHVIEN SET ";
    stringSQL += "ho='" + objsv.ho + "',";
    stringSQL += "ten='" + objsv.ten + "',";
    stringSQL += "ngaysinh='" + objsv.ngaysinh + "',";
    if (objsv.gioitinh == "True")
        stringSQL += "gioitinh='1',";
    else stringSQL += "gioitinh='0',";
    stringSQL += "noisinh='" + objsv.noisinh + "',";
    stringSQL += "hokhau='" + objsv.hokhau + "',";
    stringSQL += "dantoc='" + objsv.dantoc + "',";
    stringSQL += "sodienthoai='" + objsv.sodienthoai +
    "',";
}
```

```

        stringSQL += "malop='" + objsv.malop + "'";
        stringSQL += " WHERE (masv='" + objsv.ma + "')";

        if (db.RunSQL(stringSQL))
            MessageBox.Show("Đã lưu");
        else MessageBox.Show("Chưa lưu");
    }

```

2.3 Xóa

```

public void Xoa(clsDuLieu db, SinhVien objsv)
{
    DialogResult TraLoi;
    TraLoi = MessageBox.Show("Bạn có muốn xóa sinh viên "
        + objsv.ma + " không ?", "Cảnh báo xóa dữ liệu ",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Question);
    if (TraLoi == DialogResult.Yes)
    {
        String stringSQL = "DELETE * FROM SINHVIEN WHERE
";
        stringSQL += "masv='" + objsv.ma + "'";

        db.RunSQL(stringSQL);
        MessageBox.Show("Đã xóa");
    }
}

```

3 Môn học

3.1 Thêm

```

public void Them(clsDuLieu db, MonHoc objMH)
{
    // nếu Thông tin đã đầy đủ thì thực hiện lưu
    String stringSQL = "INSERT INTO " + TABLE_MON_HOC + " ("
    stringSQL += " mamonhoc,tenmonhoc,sodv";
    stringSQL += ") VALUES ("
    stringSQL += "'" + objMH.mamonhoc + "',";
    stringSQL += "'" + objMH.tenmonhoc + "',";
    stringSQL += "'" + objMH.sodonvihocphan + "'";
    stringSQL += ")";

    if (db.RunSQL(stringSQL))
        MessageBox.Show("Đã lưu");
    else MessageBox.Show("Chưa lưu");
}

```

3.2 Sửa

```

public void Sua(clsDuLieu db, MonHoc objMH)
{
    String stringSQL = "UPDATE " + TABLE_MON_HOC + " SET ";
    stringSQL += "tenmonhoc='" + objMH.tenmonhoc + "',";
    stringSQL += "sodv='" + objMH.sodonvihocphan + "'";
}

```

```

        stringSQL += " WHERE (mamonhoc='" + objMH.mamonhoc + "')";

        if (db.RunSQL(stringSQL))
            MessageBox.Show("Đã lưu");
        else MessageBox.Show("Chưa lưu");
    }

```

3.3 Xóa

```

public void Xoa(clsDuLieu db, MonHoc objMH)
{
    DialogResult TraLoi;
    TraLoi = MessageBox.Show("Bạn có muốn xóa môn học " +
        objMH.mamonhoc + " không ?", "Cảnh báo xóa dữ liệu ",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (TraLoi == DialogResult.Yes)
    {
        String stringSQL = "DELETE * FROM " + TABLE_MON_HOC + " WHERE ";
        stringSQL += "mamonhoc='" + objMH.mamonhoc + "'";
        db.RunSQL(stringSQL);
        MessageBox.Show("Đã xóa");
    }
}

```

4 Lớp clsDuLieu

4.1 Kết nối

```

public void KetNoi()
{
    try
    {
        if (Cnn.State == ConnectionState.Closed || Cnn.State ==
        ConnectionState.Broken)
        {
            Cnn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
            Source=" +
                Application.StartupPath + DUONGDANKETNOI;
            Cnn.Open();
        }
    }
    catch
    {
        // Hiện ra hộp thoại thông báo
        MessageBox.Show("Kết nối không thành công ");
    }
}

```

4.2 Lấy GetDataSet

```

public DataSet GetDataSet(string TenBang)
{
    DataSet ds = new DataSet();
    string strSQL = "Select * From " + TenBang;
    da = new OleDbDataAdapter(strSQL, Cnn);
    CmdBD = new OleDbCommandBuilder(da);
    da.Fill(ds, TenBang);
    return ds;
}

```

4.3 Lấy DataSet gồm nhiều trường

```

//
// lấy dataset gồm nhiều trường
//
public DataSet GetDataSet(string cactruong, string TenBang)

```

```

{
    DataSet ds = new DataSet();
    string strSQL = "Select " + cactruong + " From " + TenBang;
    da = new OleDbDataAdapter(strSQL, Cnn);
    CmdBD = new OleDbCommandBuilder(da);
    da.Fill(ds, TenBang);
    return ds;
}

```

4.4 Lấy DataSet theo lệnh SQL

```

public DataSet GetDataSetSQLCommand(string strSQL)
{
    DataSet ds = new DataSet();
    da = new OleDbDataAdapter(strSQL, Cnn);
    CmdBD = new OleDbCommandBuilder(da);
    da.Fill(ds);
    return ds;
}

```

4.5 Lấy DataSet theo điều kiện

```

//
// lấy dataset có điều kiện kèm theo
//
public DataSet GetDataSetDieuKien(string TenBang, string DieuKien)
{
    DataSet ds = new DataSet();
    string strSQL = " Select * From " + TenBang + " WHERE " + DieuKien;

    da = new OleDbDataAdapter(strSQL, Cnn);
    da.Fill(ds, TenBang);
    return ds;
}

```

4.6 Kiểm tra mã có trùng không

```

//
// Phương thức kiểm tra mã số có trùng không
//
public bool KiemTraMa(string TenBang, string DieuKien)
{
    DataTable table = new DataTable();
    string strSQL = " Select * From " + TenBang;
    if ( DieuKien != "" )
    {
        strSQL += " Where "+ DieuKien;
    }
    da = new OleDbDataAdapter(strSQL, Cnn);
    da.Fill(table);

    int dem = 0;
    foreach (DataRow row in table.Rows)
    {
        dem++;
    }
    if (dem >= 1)
        return true;
    else return false;
}

```

4.7 Chạy một lệnh SQL

```

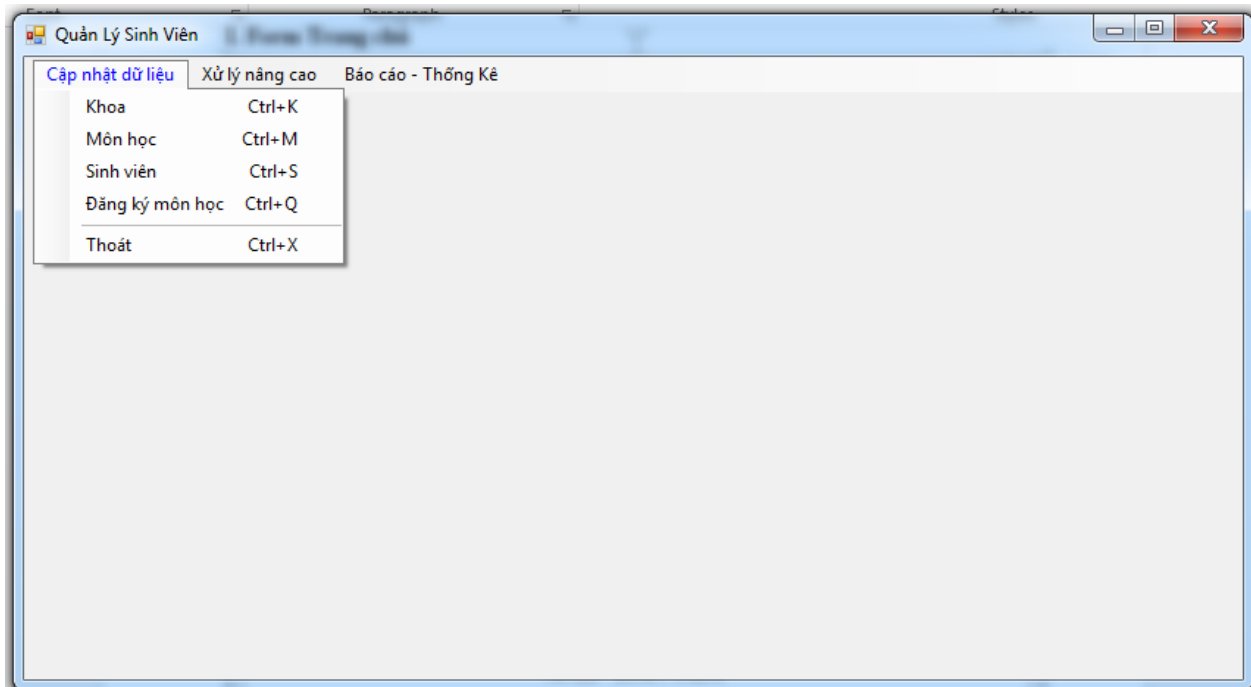
public bool RunSQL(string strSQL)
{
    try

```

```
{
    OleDbCommand Cmd = new OleDbCommand(strSQL, Cnn);
    Cmd.ExecuteNonQuery();
    return true;
}
catch (Exception ex)
{
    return false;
}
}
```

CHƯƠNG 4: MINH HỌA ỨNG DỤNG

I. Form Trang chủ quản lý sinh viên



Hình 4.1 Trang chủ

II. Form Thông tin Khoa

Khoa

THÔNG TIN KHOA

Mã Khoa:

Tên Khoa:

	Mã khoa	Tên khoa
▶	CNTT	Công Nghệ Thông Tin
	MMT	Mạng Máy Tính
	VT	Viễn Thông
*		

Hình 4.2 Form thông tin khoa

III. Form Thông tin sinh viên

THÔNG TIN SINH VIÊN

Mã SV: 14DC102

Họ SV: Huỳnh Tên SV: Nhật vũ

Giới Tính: ☒ Nam ☐ Nữ

Ngày Sinh: 7/20/1996

Nơi Sinh: Quảng Nam

Mã Khoa: Công Nghệ Thông Tin

Thêm Sửa Xóa Thoát

	Mã SV	Họ SV	Tên SV	Giới tính	Ngày sinh
	14DC001	Trần	Nam	Nam	11/12/1996
	14DC002	Lê	Thị Hoa	Nữ	12/30/1997
	14DC009	Nguyễn	Vinh	Nam	2/23/1994
▶	14DC102	Huỳnh	Nhật vũ	Nam	7/20/1996
	14DC195	Trần	Nam	Nam	9/25/1996

Hình 4.3 Form thông tin sinh viên

IV. Form Đăng ký môn học

THÔNG TIN ĐĂNG KÝ MÔN HỌC

Mã Sinh Viên : 14DC102

Mã Môn Học: 4303

Đăng Ký **Hủy** **Thoát**

	Mã SV	Mã MH
	14DC195	4303
	14DC001	4208
	14DC002	4210
▶	14DC102	4303
	14DC195	4210
*		

Hình 4.4 Form quản lý môn học

V. Form Tìm kiếm sinh viên

Timkiem

Tim Tên Sinh Viên: Nhật Vỹ

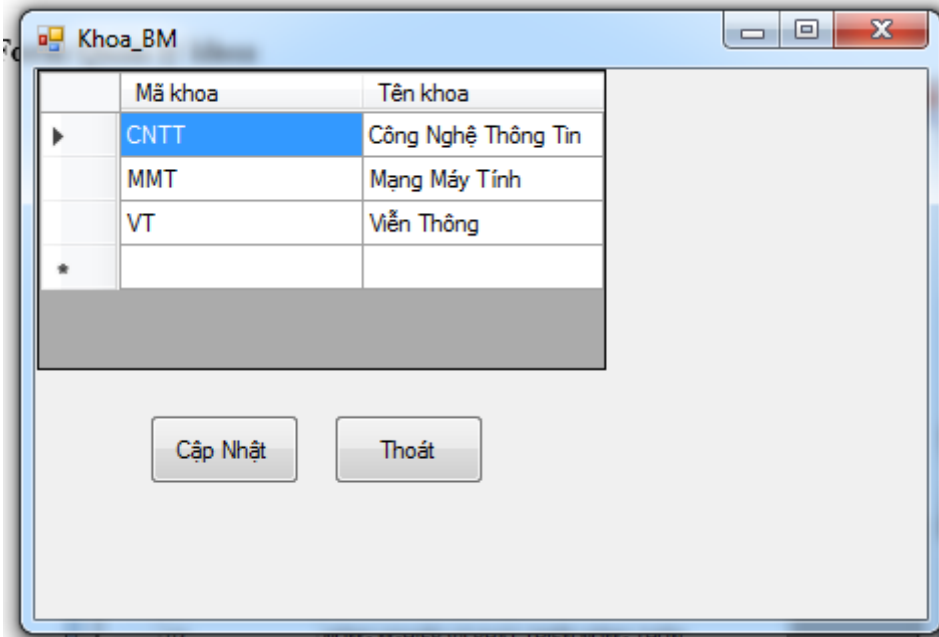
Danh Sách Sinh Viên Theo Khoa: Công Nghệ Thông Tin ▼

	Mã SV	Họ sinh viên	Tên SV	Giới tính	Ngày sinh
▶	14DC102	Huỳnh	Nhật vỹ	Nam	7/20/1996
*					

Số sinh viên: 1

Hình 4.5 Form tìm kiếm sinh viên

VI. Form Quản lý khoa



The screenshot shows a window titled "Khoa_BM" with a table and two buttons. The table has two columns: "Mã khoa" and "Tên khoa". The first row is highlighted in blue and contains "CNTT" and "Công Nghệ Thông Tin". The second row contains "MMT" and "Mạng Máy Tính". The third row contains "VT" and "Viễn Thông". There is a fourth row with a star icon in the first column and empty cells in the others. Below the table are two buttons: "Cập Nhật" and "Thoát".

	Mã khoa	Tên khoa
▶	CNTT	Công Nghệ Thông Tin
	MMT	Mạng Máy Tính
	VT	Viễn Thông
*		

Cập Nhật Thoát

Hình 4.6 Form quản lý khoa