CHUẨN CODE LẬP TRÌNH

NHÓM PHÁT TRIỂN PHẦN MỀM

QUẢN LÝ ĐỘI XE VINAMILK

Mục lục:

1. Mục đích. 3

2. Đặt tên. 3

3. Sử dụng SPACE hợp lý để câu lệnh rõ ràng hơn. 5

4. Good Programming practices 8

1. Mục đích.

Tài liệu này mình tham khảo từ nhóm quản lý quy trình phần mềm. Tài liệu tiếng Việt viết ngắn gọn những vấn đề cần thiết. Các bạn xem qua và chú ý lập trình theo các chuẩn đã nêu để quá trình phát triển sản phẩm hiệu quả hơn hen.

1. Đặt tên.

|  |
| --- |
| Chú ý :  Phân biệt giữa 2 cách đặt tên sau **Pascal Casing** và **Camel Casing**  **Pascal Casing** – Ký tự đầu tiên mỗi từ phải được viết HOA  Ví dụ: BackColor  **Camel Casing –** Kỹ tự đầu của mỗi từ đều đươc viết HOA, ngoại trừ từ đầu tiên  Ví dụ: backColor |

1. Tên Lớp sử dụng theo kiêu Pascal Casting, và phải là một DANH TỪ:

public class **HelloWorld**

{

...

}

1. Tên phương thức sử dụng theo kiêu Pascal Casting, và phải bắt đầu là một ĐỘNG TỪ:

void **SayHello**(string name)

{

...

}

1. Sử dụng Camel casing đối vói BIẾN và THAM SỐ hàm.

int **totalCount** = 0;

void SayHello(string name)

{

string **fullMessage** = "Hello " + name;

...

}

1. Đặt tên biến phải đầy đủ ý nghĩa. So sánh 2 ví dụ sau:

Good:

string address

int salary

Not Good:

string nam

string addr

int sal

1. Đặt tên thuộc tính lớp dạng m\_name

Vd: private string m\_hoTen;

1. Không đặt tên các biến tối nghĩa như i, n, s ... Nên dùng index, temp

Ngoại trừ biến chạy trong vòng lập thì ok!.

for ( int i = 0; i < count; i++ )

{

...

}

1. Biến boolean, nên có tiền tố là dấu ” \_ “ để không nhầm lẫn với một số hàm có dạng is....

VD: private bool \_isFinished

1. Cách đặt tên các CONTROL có tiền tố như trong bảng sau:

|  |  |
| --- | --- |
| **Control** | **Prefix** |
| Label | lbl |
| TextBox | txt |
| DataGrid | dtg |
| DataGridView | dtgv |
| Button | btn |
| ImageButton | imb |
| Hyperlink | hlk |
| DropDownList | ddl |
| ListBox | lst |
| DataList | dtl |
| Repeater | rep |
| Checkbox | chk |
| CheckBoxList | cbl |
| RadioButton | rdo |
| RadioButtonList | rbl |
| Image | img |
| Panel | pnl |
| PlaceHolder | phd |
| Table | tbl |
| Validators | val |
| Form | frm |

Ví dụ:

**Tên Button LOGIN đặt như sau: btn\_Login**

**Tên Textbox HỌ TÊN đặt như sau: txt\_HoTen**

1. Tên File phải trùng với tên lớp:

Chảng Hạng: Lớp HelloWorld, thì tên file phải là helloworld.cs

1. Sử dụng SPACE hợp lý để câu lệnh rõ ràng hơn.
2. Viết ghi chú phải đồng cấp với các câu lệnh ứng với ghi chú đó.

Good:

// Format a message and display

string fullMessage = "Hello " + name;

DateTime currentTime = DateTime.Now;

string message = fullMessage + ", the time is : " + currentTime.ToShortTimeString();

MessageBox.Show ( message );

Not Good:

// Format a message and display

string fullMessage = "Hello " + name;

DateTime currentTime = DateTime.Now;

string message = fullMessage + ", the time is : " + currentTime.ToShortTimeString();

MessageBox.Show ( message );

1. Sử dụng giản cách dòng cho từng nhóm code.

Good:

bool SayHello ( string name )

{

string fullMessage = "Hello " + name;

DateTime currentTime = DateTime.Now;

string message = fullMessage + ", the time is : " + currentTime.ToShortTimeString();

MessageBox.Show ( message );

if ( ... )

{

// Do something

// ...

return false;

}

return true;

}

Not Good:

bool SayHello (string name)

{

string fullMessage = "Hello " + name;

DateTime currentTime = DateTime.Now;

string message = fullMessage + ", the time is : " + currentTime.ToShortTimeString();

MessageBox.Show ( message );

if ( ... )

{

// Do something

// ...

return false;

}

return true;

}

1. Tại mỗi câu lệnh như if, for… phải có dấu { /\* Nhóm Lệnh \*/}

Good:

if ( ... )

{

// Do something

}

Not Good:

if ( ... ) {

// Do something

}

1. Sử dụng dấu cách trước sau các toán tử và dấu ngoặc ( ).

Good:

if ( showResult == true )

{

for ( int i = 0; i < 10; i++ )

{

//

}

}

Not Good:

if(showResult==true)

{

for(int i= 0;i<10;i++)

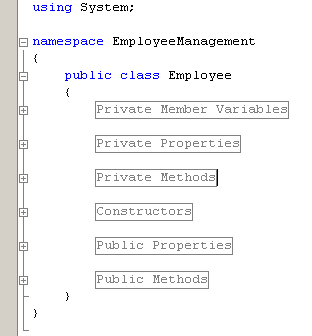
{

//

}

}

1. Sử dụng #region gom nhóm các dòng có cùng quan hệ lại với nhau. Dùng #region thì khi cuộn các nhóm lệnh lại sẽ nhìn gọn gàng như sau:



Hay:

// preprocessor\_region.cs

#region MyClass definition

public class MyClass

{

public static void Main()

{

}

}

#endregion

1. Dòng chú thích ko nên ghi /\*…..\*/ cho dòng mà nên ghi //

Good:

// Day la ghi chu

// Day la ghi chu

// Day la ghi chu

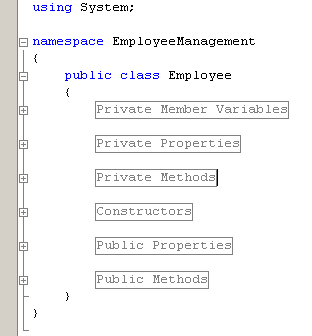
Not Good:

/\*Day la ghi chu

Day la ghi chu

Day la ghi chu \*/

1. Đưa các dòng khai báo là **private** lên trên các dòng khai báo **public**  như ví dụ trên.



1. Chuẩn cho phương thức, handle …
2. Một phương thức đẹp chỉ khoản 25 dòng lệnh, nếu dài hơn thì nên tách thành các phương thức con.
3. Tên phương thức được đặt đã nói lên chức năng của nó, viet ghi chú phải mô tả đúng chức năng của nó, đừng để “ông nói gà, bà nói vịt” ☺

Good:

void SavePhoneNumber ( string phoneNumber )

{

// Save the phone number.

}

Not Good:

// This method will save the phone number.

void SaveDetails ( string phoneNumber )

{

// Save the phone number.

}

1. Mỗi phương thức chỉ nên thực hiện một chức năng ‘one job’.

Good:

// Save the address.

SaveAddress ( address );

// Send an email to the supervisor to inform that the address is updated.

SendEmail ( address, email );

void SaveAddress ( string address )

{

// Save the address.

// ...

}

void SendEmail ( string address, string email )

{

// Send an email to inform the supervisor that the address is changed.

// ...

}

Not Good:

// Save address and send an email to the supervisor to inform that

// the address is updated.

SaveAddress ( address, email );

void SaveAddress ( string address, string email )

{

// Job 1.

// Save the address.

// ...

// Job 2.

// Send an email to inform the supervisor that the address is changed.

// ...

}

1. Luân luôn kiểm tra các giá trị không hợp lệ. Ví dụ bạn có đối số truyền vào có nhiều khả năng xảy ra hãy kiểm tra nó trước khi xử lý.

Good:

If ( memberType == eMemberTypes.Registered )

{

// Registered user… do something…

}

else if ( memberType == eMemberTypes.Guest )

{

// Guest user... do something…

}

**else**

**{**

**// Un expected user type. Throw an exception**

**throw new Exception (“Un expected value “ + memberType.ToString() + “’.”)**

**// If we introduce a new user type in future, we can easily find**

**// the problem here.**

**}**

Not Good:

If ( memberType == eMemberTypes.Registered )

{

// Registered user… do something…

}

else

{

// Guest user... do something…

// If we introduce another user type in future, this code will

// fail and will not be noticed.

}

1. Không nên sử dụng số gán cố định trong code, mà nên dùng một hằng số được định nghĩa để dễ quản lý…

Ví dụ:

Good:

for ( int i = 0; i < MAX; i++ )

{

...

}

Not Good:

for ( int i = 0; i < 100; i++ )

{

...

}

1. Nên chuyển chuổi sang dạng HOA hay thường trước khi so sánh.

Ví dụ:

if ( name.ToLower() == “john” )

{

//…

}

1. Sử String.Empty Thay cho “” – chuỗi rỗng.

Good:

If ( name == String.Empty )

{

// do something

}

Not Good:

If ( name == “” )

{

// do something

}

1. Sử dụng kiểu enum khi cẩn thiết trong những trường hợp giống như sau:

Good:

enum MailType

{

Html,

PlainText,

Attachment

}

void SendMail (string message, MailType mailType)

{

switch ( mailType )

{

case MailType.Html:

// Do something

break;

case MailType.PlainText:

// Do something

break;

case MailType.Attachment:

// Do something

break;

default:

// Do something

break;

}

}

Not Good:

void SendMail (string message, string mailType)

{

switch ( mailType )

{

case "Html":

// Do something

break;

case "PlainText":

// Do something

break;

case "Attachment":

// Do something

break;

default:

// Do something

break;

}

}

1. Tách các điều kiện so sánh dài ra để dễ kiểm soát:

Good:

isHighScore = (value >= \_highScore);

isTiedHigh = (value == \_highScore);

isValid = (value < \_maxValue);

if ((isHighScore && ! isTiedHigh) && isValid)

{…}

Not Good:

if (((value > \_highScore) && (value != \_highScore)) && (value < \_maxScore))

{…}

1. Nên điều khiển lỗi ngoại lệ dung try/catch

try

{

Console.WriteLine("Executing the try statement.");

throw new NullReferenceException();

}

catch(NullReferenceException e)

{

Console.WriteLine("{0} Caught exception #1.", e);

}

catch

{

Console.WriteLine("Caught exception #2.");

}

finally

{

Console.WriteLine("Executing finally block.");

}