|  |  |
| --- | --- |
|  | **2012** |
|  | **Duy Tan university**  AnhMV |

|  |
| --- |
| **[Coding Convention]** |
|  |

**Mục lục**

[1 Tiền đề 1.1-4](#_Toc321837960)

[1.1 Phạm vi áp dụng 1.1-4](#_Toc321837961)

[2 Quy tắc naming 1.1-5](#_Toc321837962)

[2.1 Identifier 2.1-5](#_Toc321837963)

[2.2 Quy ước về identifier 2.2-6](#_Toc321837964)

[2.3 File name 2.3-6](#_Toc321837965)

[2.4 Package name 2.4-7](#_Toc321837966)

[2.5 Class/interface name 2.5-7](#_Toc321837967)

[2.6 Method name 2.6-8](#_Toc321837968)

[2.7 Tên biến 2.7-8](#_Toc321837969)

[2.8 Constant name 2.8-9](#_Toc321837970)

[3 Cấu trúc file 2.8-10](#_Toc321837971)

[3.1 Chiều dài của source file 3.1-10](#_Toc321837972)

[3.2 Cấu trúc source và trình tự khai báo 3.2-10](#_Toc321837973)

[3.3 Header comment 3.3-10](#_Toc321837974)

[3.4 Khai báo Package , khai báo import 3.4-10](#_Toc321837975)

[3.5 Định nghĩa class or interface 3.5-11](#_Toc321837976)

[3.6 Documentation comment của class 3.6-11](#_Toc321837977)

[3.7 Documentation comment của biến số member 3.7-12](#_Toc321837978)

[3.8 Documentation comment của method 3.8-13](#_Toc321837979)

[4 Format của source 3.8-14](#_Toc321837980)

[4.1 Indent 4.1-14](#_Toc321837981)

[4.2 Độ dài line 4.2-14](#_Toc321837982)

[4.3 Vị trí phân cách line 4.3-14](#_Toc321837983)

[4.4 Blank line 4.4-16](#_Toc321837984)

[4.5 Space 4.5-16](#_Toc321837985)

[5 Comment 4.5-18](#_Toc321837986)

[5.1 Implement comment 5.1-18](#_Toc321837987)

[5.1.1 Block comment 5.1-18](#_Toc321837988)

[5.1.2 Comment 1 line 5.1-18](#_Toc321837989)

[5.1.3 Comment cuối line 5.1-19](#_Toc321837990)

[5.2 Documentation comment 5.2-19](#_Toc321837991)

[5.3 Vị trí mô tả comment 5.3-19](#_Toc321837992)

[5.4 Comment out 5.4-20](#_Toc321837993)

[6 Khai báo 5.4-21](#_Toc321837994)

[6.1 Khai báo biến 6.1-21](#_Toc321837995)

[6.2 Khai báo biến 6.2-22](#_Toc321837996)

[7 Câu lệnh 6.2-23](#_Toc321837997)

[7.1 Câu lệnh đơn 7.1-23](#_Toc321837998)

[7.2 Câu lệnh phức hợp 7.2-23](#_Toc321837999)

[7.3 Câu lệnh control 7.3-24](#_Toc321838000)

[7.3.1 Câu lệnh If 7.3-24](#_Toc321838001)

[7.3.2 Câu lệnh for 7.3-24](#_Toc321838002)

[7.3.3 Câu lệnh while 7.3-25](#_Toc321838003)

[7.3.4 Câu lệnh do-while letter 7.3-25](#_Toc321838004)

[7.3.5 Câu lệnh switch 7.3-25](#_Toc321838005)

[7.3.6 Câu lệnh try-catch-finally l 7.3-26](#_Toc321838006)

[8 Xử lí error và exception 7.3-27](#_Toc321838007)

[8.1 Exception sử dụng 8.1-27](#_Toc321838008)

[8.2 Phòng chống hiding bug 8.2-27](#_Toc321838009)

[8.3 Phương thức error handling 8.3-27](#_Toc321838010)

[9 Tiêu chuẩn liên quan đến JSP 8.3-29](#_Toc321838011)

[9.1 Về việc phân biệt cách sử dụng coding convention và tool phân tích tĩnh 9.1-29](#_Toc321838012)

[9.2 Header comment của file JSP 9.2-29](#_Toc321838013)

[10 Phụ lục 9.2-30](#_Toc321838014)

[10.1 List tag .NETdoc 10.1-30](#_Toc321838015)

[10.2 Ví dụ code đã áp dụng coding convention 10.2-31](#_Toc321838016)

# Tiền đề

## Phạm vi áp dụng

Phạm vi áp dụng tài liệu coding convention này là phần UOC của ứng dụng.

# Quy tắc naming

Chương này mô tả tiêu chuẩn về việc naming các loại trong source .NET và file sử dụng trong .NET.

## Identifier

Trong tài liệu này sẽ gọi tên mà được sử dụng trong class( bao gồm interface) , method ( bao gồm constructor) , biến, constant, argument là identifier.

Kí tự mà có thể sử dụng trong identifier chỉ là chữ số alphabet, trong kí tự đầu phải sử dụng chữ alphabet(A-Z và a-z)

Trong identifier thì ko sử dụng tên bắt đầu bằng chữ số, hoặc những kí hiệu như là [%] or [@].

Trong identifier sẽ phân biệt kí tự in hoa và chữ thường.

Ex 1.1.1 Ví dụ về naming identifier

CAPSTONE VD đúng

Capstone VD đúng

1\_Customer VD sai (kí tự đầu là chữ số )

Cap Stone VD sai (có space)

Trường hợp sử dụng từ viết tắt trong identifier thì chỉ được sử dụng những từ viết tắt mà được sử dụng chung trong dự án, hoặc là những từ viết tắt/danh từ riêng mà đang được sử dụng rộng rãi

## Quy ước về identifier

Trong Keyword của .NET có những cái sau đây. Vì không thể sử dụng trong tên class, tên biến số, tên method nên phải chú ý.

Vì những keyword chữ hoa là cái đặc thù của .NET (So với C++)nên phải chú ý đặc biệt.

Ví dụ 1.2.1 Keyword của .NET

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **abstract** | continue | for | new | switch | false |
| **assert** | default | goto | **package** | **synchronized** | **null** |
| **boolean** | do | if | private | this | true |
| break | double | **implements** | protected | throw |
| **byte** | else | **import** | public | **throws** |
| case | enum | **instanceof** | return | **transient** |
| catch | **extends** | int | short | try |
| char | **final** | **interface** | static | void |
| class | **finally** | long | **strictfp** | volatile |
| const | float | **native** | **super** | while |

※true, false, null không phải là keyword nhưng là reserved word và không thể sử dụng như biến số.

## File name

Trong file mà sử dụng khi phát triển system bằng .NET sẽ có những loại chủ yếu như dưới đây :

Trong các loại file mà được mô tả như dưới đây thì name file phải tuân theo Quy định naming identifier. Đặc biệt về name file của source code/byte code thì phải sử dụng name giống với public class/ interface .

Phần mở rộng được sử dụng tại các file như sau. Ngoài ra, xem tất cả phần mở rộng là chữ thường.

Ví dụ File được sử dụng trong .NET 1.3.1

| Loại file | Phần mở rộng |
| --- | --- |
| Source code | ..NET |
| Byte code | .class |
| .NET archive file | .jar |
| Web archive file | .war |
| Enterprise archive file | .ear |
| .NETServer Pages file | .jsp |

## Package name

Phần đầu của package name thì set là 1 domain name vị trí cao nhất (com, edu, gov, mil, net, org hoặc là 2 ký tự Code tên nước mà đã được định nghĩa trong ISO Standard 3166、 1981) . Không được sử dụng [.NET] và [.NETx] vì đang được contract bởi Sun.

Package name bắt đầu theo trình tự ngược lại với internet domain name và gắn thêm office name. Sau đó thì tùy theo từng project. Ngoài ra trường hợp Up thì sử dụng name tuân theo quan hệ quyền lợi của file source.

Set toàn bộ package name là viết dưới dạng chữ thường.

Ex 1.4.1 ví dụ về naming package name

package capstone.sample; //Good ex

package capstone.sample; //Bad ex(Bao gồm chữ hoa)

## Class/interface name

Class name thì sẽ set là danh từ viết hoa chữ đầu. Chữ đầu tiên của từ đơn là chữ hoa, ngoài ra thì viết dưới dạng chữ thường. Đơn giản nhất có thể,và đặt tên mà nội dung class có thể đoán được. Tránh những từ viết tắt, sử dụng toàn bộ từ đơn.

Trường hợp là exception class thì add chuỗi ký tự [Exception] ( Ex : SQLException) vào cuối cùng của tên class

Ex 1.5.1 Ví dụ naming class name

public MyClass { //ví dụ đúng

public MakeMyClass { //ví dụ sai ( Chữ đầu không phải danh từ)

public myClass { //ví dụ sai ( Chữ đầu tiên viết thường)

public MyCls { //ví dụ sai ( Viết tắt)

## Method name

Method name phải là tên mà được bắt đầu bằng động từ được viết thường ở chữ đầu, chữ đầu của các từ đơn tiếp theo thì bắt đầu bằng chữ hoa. Method dùng để debug thì gắn thêm tiếp đầu ngữ [debug] để phân biệt với method khác.

Về name method của .NETBeans, method set giá trị tại biến member thì set là "set"＋tên biến member, method lấy giá trị từ biến member thì set là "get"＋name biến member. Tuy nhiên, trường hợp giá trị lấy từ biến số member là boolean thì set là "is"＋tên biến member .

Ex 1.6.1 ví dụ naming method name

doMyMethod() { //ví dụ đúng

myMethod() { //ví dụ sai (Chữ đầu không phải động từ )

doMymethod() { //ví dụ sai (Chữ đầu của từ là chữ thường)

## Tên biến

Tên biến phải là tên mà bắt đầu từ chữ thường , chữ đầu của các từ đơn tiếp theo sẽ bắt đầu bằng chữ hoa. Tên biến thì set tên đơn giản và là tên sao cho có thể phán đoán dễ dàng usage.. Tránh những tên biến không có ý nghĩa .Tên biến 1 ký tự thì set là chỉ có thể sử dụng trong biến dùng cho exception class và biến counter của for loop. Trường hợp sử dụng trong biến counter thì set là chỉ có thể sử dụng i,j,k theo thứ tự này, trường hợp sử dụng trong biến dùng cho class exception thì set là có thể sử dụng e.

Ex 1.7.1 ví dụ naming tên biến số

myVariable //ví dụ đúng

MyVariable //ví dụ sai (chữ đầu là chữ viết hoa)

Ex 1.7.2 1 ví dụ sử dụng của biến ký tự

for (int i = 0; i < length; i++) { // sử dụng i tại biến counter

for (int j = 0; j < length; j++) { // Phải sử dụng i,j,k theo thứ tự này

statements;

}

statements;

}

try {

statements;

} catch (SampleException e) { // sử dụng e tại biến dùng cho class exception

statements;

} finally {

statements;

}

## Constant name

Tất cả hằng số được mô tả bằng chữ viết hoa , giữa các từ sẽ phân cách bằng

Ex 1.8.1 ví dụ naming tên hằng số

MY\_CONSTANT\_VALUE //ví dụ đúng

My\_Constant\_Value //ví dụ sai (đang bao gồm chữ thường)

MYCONSTANTVALUE //ví dụ sai( không có phân cách"\_" giữa các từ)

# Cấu trúc file

Trong chương này thì sẽ mô tả tiêu chuẩn liên quan đến cấu tạo của .NET source file.

## Chiều dài của source file

Về source thì tiêu chuẩn mỗi file là 2000 dòng ( không tính dòng comment ) , trường hợp vượt quá 2000 dòng thì xem xét lại design class.

## Cấu trúc source và trình tự khai báo

Trong các file source thì chỉ định nghĩa 1 interface hay class của thuộc tính public.Về interface và class thuộc tính private có liên quan đến chức năng thì định nghĩa trong cùng file cũng được.Tuy nhiên, đối với trường hợp này thì sẽ bố trí sao cho interface hoặc class của thuộc tính public nằm ở đầu file. File source được mô tả theo thứ tự dưới đây.

###### Header comment

###### Khai báo package

###### Khai báo import

###### Định nghĩa class/ interface

## Header comment

Mô tả item dưới đây

Ex 2.3.1 ví dụ comment header

//## Capstone

//## DTU. All Rights Reserved.

//##

## Khai báo Package , khai báo import

Mô tả item dưới đây vào phần đầu của code

###### Câu lệnh package

###### (Về package name thì tham khảo mục 1.4)

###### Về câu lệnh import thì sẽ mô tả khi cần thiết

## Định nghĩa class or interface

Mô tả định nghĩa của Class hoặc interface . Cấu trúc như dưới đây

###### Documentation comment của class/ interface

###### Câu lệnh class/interface

###### Implement comment của class/ interface

###### Biến class (static)

###### Khai báo theo thứ tự biến class public, biến class protected, biến level package , biến private

###### Biến instance

###### Khai báo theo thứ tự biến public, biến protected, biến level package, biến private.

###### initializer)

###### Constructor

###### Documentation comment của method

###### Định nghĩa method

**(10) Cho là thực hiện grouping bằng chức năng, và sẽ ưu tiên hơn phân loại dựa vào quyền acess và scop. Về method thì trừ comment, không được vượt quá 200 line.**

## Documentation comment của class

Mô tả item dưới đây để output document bằng .NETdoc, mô tả item dưới đây.

###### Outline về spec class

###### Author(@author) [Tùy ý]

###### Thông tin version (@version) [Tùy ý]

###### Class tham khảo(@see) [Tùy ý]

Vd 2.6.1 Ví dụ về documentation comment của class(Chi tiết tham khảo10.1)

/\*\*

\* <p>Screen ID: Login</p>

\* <p>Event ID: CheckLogin</p>

\*

\* @author $Author$

\* @version $Revision$ $Date$

\*/

## Documentation comment của biến số member

###### Outline biến member

###### Khai báo biến member thực tế

Vd 2.7.1 Vd về document comment của biến member

/\*\* comment \*/

public static int classVar;

## Documentation comment của method

Mô tả item dưới đây để output document bằng .NETdoc

###### Outline của method spec

###### Giải thích về argument (@param)

###### Giải thích về giá trị trả về (@return)

###### Class hoặc method tham chiếu (@see) [Tùy ý]

###### Tên exception có khả năng bị throw (@exception ,@throws)

# Format của source

Chương này chỉ ra những tiêu chuẩn liên quan đến format cơ bản khi coding source .NET.

## Indent

Độ dài indent là 4 space halfsize. Vì độ dài tab phụ thuộc vào setting của editor nên trong indent không sử dụng tab.

## Độ dài line

Độ dài 1 dòng tùy ý, tham khảo chương dưới đây và thực hiện break line thích hợp.

## Vị trí phân cách line

Trường hợp có nhiều line thì sau dấu phẩy , sẽ xuống dòng tại thích hợp trước hoặc sau operator. Căn cứ cấu tạo phân tầng của câu lệnh, càng áp dụng được vị trí phân cách ở level càng cao càng tốt. Line mới sau khi thực hiện break line phải phù hợp với indent và nội dung mô tả cùng level với line trước. Trường hợp làm cho phù hợp indent với line trước làm cho khó đọc code thì indent với độ dài gấp đôi indent của line trước.

VD 3.3.1 Ví dụ về vị trí phân cách line

(1)　Thực hiện breakline sau dấu phẩy

someMethod(longExpression1, longExpression2, longExpression3,

longExpression4, longExpression5);

(2)　Thực hiện breakline với level cao

// Good example

longName1 = longName2 \* (longName3 + longName4 - longName5)

+ 4 \* longName6;

// Bad example

longName1 = longName2 \* (longName3 + longName4

- longName5) + 4 \* longName6;

(3)　Line mới sau khi breakline phải làm cho phù hợp line trước với indent

someMethod(int anArg, Object anotherArg, String yetAnotherArg,

Object andStillAnother) {

(4)　Trường hợp ngoại lệ nếu như indent trở nên dày đặc thì thực hiện indent với 4 space halfsize

private static synchronized horkingLongMethodName(int anArg,

Object anotherArg, String yetAnotherArg,

Object andStillAnother) {

(5) Khi thực hiện breakline tại mệnh đề if , thì do nếu như là indent của 4 space halfsize thì việc xem trong block sẽ trở nên khó khăn, do đó thường là sử dụng 8 space halfsize

// Good example

if ((condition1 && condition2)

|| (condition3 && condition4)

|| !(condition5 && condition6)) {

doSomethingAboutIt();

}

// Bad example

if ((condition1 && condition2)

|| (condition3 && condition4)

|| !(condition5 && condition6)) { // Bad breakline

doSomethingAboutIt(); // Line này khó hiểu

}

## Blank line

Đối với những trường hợp dưới đây thì sẽ chèn 1 blank line .

###### Giữa các yếu tố dưới đây mà cấu tạo source file

* Header comment
* Khai báo package
* Khai báo import
* Định nghĩa class/ interface

###### Giữa các method(Tuy nhiên, trường hợp giữa các setter/getter và nhóm method đã overload etc. có ý nghĩa như là chuỗi block thì không chèn blank line cũng được)

###### Giữa các block mang tính logic trong method

###### Còn lại là vị trí cần thiết để có thể dễ dàng đọc

## Space

Truong những trường hợp dưới đây thì sẽ insert space .

###### Sau dấu phẩy ở trong dòng argument

###### Giữa các operator ngoại trừ các operator [.(Dấu chấm)] và monadic operator với operand đó .

###### Giữa nhiều câu lệnh trong dấu ngoặc nhỏ của câu lệnh and và câu lệnh or

###### Sau cast operator

###### Sau dấu chấm phẩy trong câu lệnh for

Vd 3.5.1 Vd insert space

(1) Sau dấu phẩy ở dòng argument,

someMethod(int firstArgument, int secondArgument) {

・・・

}

###### (2) Giữa các operator ngoại trừ các operator 「．(Dấu chấm)」 và monadic operator với operand .

a = b + c;

a = (a + b) / (c \* d);

n++;

###### (3) Giữa nhiều câu lệnh trong dấu ngoặc nhỏ của câu lệnh and và câu lệnh or

if (condition1 || condition2) {

doSomething();

}

###### (4) Sau cast operator

b = (int) a;

###### (5) Sau dấu chấm phẩy trong câu lệnh for

for (int i = 0; i < length; i++) {

doSomething();

}

# Comment

Chương này chỉ ra những tiêu chuẩn khi insert comment vào trong .NET source.

## Implement comment

Trong implement comment có 3 loại [Block comment][Comment 1 line][Comment cuối line]. Sử dụng [// comment] cho [Comment 1 line]và [Comment cuối line] .

### Block comment

Mô tả comment qua nhiều line.

* Mô tả file, method, cấu tạo data, thuật toán.
* Có thể mô tả ở phần đầu file và trong method.
* Indent giống với code đối tượng.

Vd 4.1.1 Vd mô tả block comment

/\*

\* Comment

\* Comment

\*/

### Comment 1 line

Comment ngắn mô tả trước 1 line code đối tượng.

* Indent giống với code đối tượng

Vd 4.1.2 Ví dụ 1 mô tả comment 1 line

if (condition) {

// comment 1 line

doMyMethod(argument);

}

### Comment cuối line

Comment mô tả cuối line của code đối tượng(1 line).

* Mô tả ngắn sau code đối tượng.
* Mô tả bằng đơn vị statement.
* Comment cuối line liên tục thì sử dụng space và indent tương tự.

Vd 4.1.4 Vd 1 mô tả comment cuối line

if (condition) {

doMyMethod(argument); // comment cuối line

} else {

return false; // comment cuối line

// Kết hợp indent

}

## Documentation comment

Comment được xử lý khi gen API document bằng command .NETdoc .

Quy định cách viết dùng để giải thích class, giải thích method.

* Line thứ 1 bắt đầu bằng ký hiệu bắt đầu comment "/\*\*" và tiếp sau đó là breakline
* Từ line thứ 2 trở đi thì bắt đầu bằng "\*" đã được indent.
* Line cuối cùng chỉ là ký hiệu kết thúc comment đã được indent "\*/+" .

(Tham khảo ví dụ 3.6～3.8)

## Vị trí mô tả comment

Insert comment hết sức có thể vào vị trí chỉ ra dưới đây.

* Khai báo biến local(comment cuối line)
* Câu lệnh control(comment 1 line hoặc comment cuối line)

Vd 4.3.1 Ví dụ mô tả comment khi khai báo biến local

someMethod(int argument) {

int myVariable; //Outline về biếnsố

}

Vd 4.3.2 Vd mô tả comment câu lệnh control

if (condition) {

statements; //Outline xử lý trường hợp phù hợp với điều kiện

} else {

statements; // Outline xử lý trường hợp không phù hợp với điều kiện

}

//Outline xử lý câu lệnh for

for (initialization; condition; update) {

statements;

}

## Comment out

Trường hợp comment out 1 phần source dựa vào việc chỉnh sửa program thì sử dụng [// Comment].

Vd 4.4.1 Good example về comment out

//someMethod(int argument) {

// int myVariable;

//}

Vd 4.4.2 Bad example về comment out

/\*

someMethod(int argument) {

int myVariable;

}

\*/

# Khai báo

Chương này chỉ ra những tiêu chuẩn liên quan đến khai báo biến hoặc class/ interface .

## Khai báo biến

###### Khai báo biến ngay trước khi cần thiết. Scop ( phạm vi valid ) bắt buộc phải đạt giá trị nhỏ nhất. Việc instance hóa ( new) trong loop thì cần phải lưu ý vì sẽ ảnh hưởng đến tính năng.( Tham khảo 8.2(3))

###### Việc khởi tạo biến đơn giản thực hiện khi khai báo.

###### Mô tả 1 khai báo vào 1 line.(Không thực hiện nhiều khai báo trong 1line)

Vd 5.1.1 Vd về khai báo biến

(1) Khai báo biến ngay trước khi cần thiết

(2) Việc khởi tạo biến đơn giản thực hiện khi khai báo

public void doMyMethod() {

int length = 10; //Trước khi cần

for (int i = 0; i < length; i++) { // Index của loop

・・・

}

int result = 0; //Trước khi cần

if (length >= 10) {

result = length - 10;

} else if (length > 0) {

result = length;

}

}

(3) Mô tả 1 khai báo vào1 line

int level; //Good exam

int size; //Good exam

int level, size; // Bad exam

int a; //Good exam

int[] b; //Good exam

int a, b[]; //Bad exam

(Mục ghi chú đặc biệt)Đối với ngôn ngữ C thì không thể khai báo biến trong câu lệnh for như ví dụ trên(C++ thì được ) nhưng ghi bằng .NET thì được. Trường hợp này, scope của biến số chỉ là trong loop for, không thể tham khảo biến đó sau khi ra khỏi loop.

## Khai báo biến

Dưới đây là miêu tả khai báo class, interface.

###### Không bố trí space giữa dấu ngoặc tròn của list parameter và tên method.

###### Dấu ngoặc đơn mở sẽ miêu tả cuối dòng đã miêu tả khai báo.

###### Dấu ngoặc đơn đóng sẽ kết hợp indent với đầu trang của letter khai báo của dòng đã miêu tả khai báo.

###### Phân cách method bằng blank line.

###### Trường hợp parameter của method và biến số No có tên giống nhau thì sẽ bổ sung "this." rõ vào biến số No.

Ví dụ 5.2.1 Ví dụ của khai báo class

class Sample {

int index;

setIndex(int index) {

this.index = index;

}

}

# Câu lệnh

Chương này sẽ trình bày tiêu chuẩn liên quan đến phương pháp miêu tả câu lệnh khi thực hiện coding .NET source.

## Câu lệnh đơn

Miêu tả 1 câu lệnh trong 1 dòng.

Ví dụ 6.1.1 ví dụ của câu lệnh đơn

i++; //good example

j++; // good example

i++; j++; // bad example

## Câu lệnh phức hợp

Câu lệnh phức hợp là câu lệnh có bao gồm tập hợp nhiều câu lệnh nằm trong dấu ngoặc đơn. Phần nằm trong dấu ngoặc đơn thì thực hiện indent 1 tầng từ câu lệnh có bao gồm dấu ngoặc đơn. Dấu ngoặc đơn mở sẽ mô tả vào cuối dòng bắt đầu mô tả câu lệnh phức hợp. Dấu ngoặc đơn đóng thì sẽ mô tả đầu dòng, kết hợp indent và câu lệnh đầu tiên của câu lệnh phức hợp. Nếu câu lệnh có 1 phần là cấu trúc control của if hay for thì dù trong đó chỉ có 1 câu lệnh thôi thì cũng phải sử dụng dấu ngoặc đơn.

Ví dụ 6.2.1 Ví dụ về câu lệnh phức hợp

if (condition) { // good example

i++;

}

if (condition) i++; // bad example

## Câu lệnh control

### Câu lệnh If

Khi miêu tả câu lệnh if tiếp sau câu lệnh else, thì sẽ bố trí 1 space vào sau else rồi tiếp sau đó là câu lệnh if .

Ví dụ 6.3.1.1 Form của câu lệnh if 1

if (condition) {

statements;

}

Ví dụ 6.3.1.2 Form của câu lệnh if 2

if (condition) {

statements;

}

### Câu lệnh for

Câu lệnh run tiếp sau câu lệnh if thì dù chỉ là 1 câu cũng phải cho nằm trong dấu ngoặc đơn. Khi sử dụng comma để phân cách phần setting khởi tạo (initialization) hoặc xử lí kế thừa (update) để mô tả nhiều câu lệnh , thì cho số câu lệnh phải từ 3 trở xuống

Ví dụ 6.3.2.1 form của câu lệnh for 1

for (initialization; condition; update) {

statements;

}

### Câu lệnh while

Về câu lệnh run tiếp theo sau câu lệnh while thì dù có là 1 câu cũng phải cho vào trong dấu ngoặc đơn.

Ví dụ 6.3.3.1 form của câu lệnh white 1

for (initialization; condition; update) {

statements;

}

### Câu lệnh do-while letter

Về câu lệnh run tiếp theo sau câu lệnh do thì dù có là 1 câu cũng phải cho vào trong dấu ngoặc đơn.

Ví dụ 6.3.4.1 form của câu lệnh do- white

for (initialization; condition; update) {

statements;

}

### Câu lệnh switch

Bố trí dòng comment // break なし // vào cuối của case không tuân theo câu lệnh break chỉ rõ việc break không được thực hiện.

Tại tất cả câu lệnh switch sẽ định nghĩa label default rồi đặt câu lệnh break .

Ngoài ra, trường hợp đã quên blank trong công thức constant là [case] , có khả năng được phán đoán là text label dẫn đến những thao tác không mong muốn nên cần phải chú ý.

Ví dụ 6.3.5.1 form của câu lệnh switch

switch (condition) {

case ABC:

statements;

// break なし //

case DEF:

statements;

break;

case XYZ:

statements;

break;

default:

statements;

break;

}

### Câu lệnh try-catch-finally l

Câu lệnh run tiếp theo sau câu lệnh try,catch,finally thì dù chỉ là 1 câu lệnh thì cũng phải cho nằm trong dấu ngoặc đơn.

Ví dụ 6.3.6.1 form của câu lệnh try-catch-finally

try {

statements;

} catch (SampleException e) {

statements;

} finally {

statements;

}

# Xử lí error và exception

Chương này sẽ trình bày tiêu chuẩn liên quan đến việc xử lý error và exception .

## Exception sử dụng

Về exception có thể sử dụng trong project thì sẽ giải thích tại các guideline.

## Phòng chống hiding bug

Trường hợp program hoặc method nhận giá trị input không đúng thì cần phải báo cáo bằng phương pháp nào đó chẳng hạn như output ra log hoạt động hoặc hiển thị dialogue box, dù cho operation có tiếp tục chạy hay không .

## Phương thức error handling

###### Điểm chú ý khi nest try-catch

Trường hợp đã nest try-catch , nếu không throw exception đã catch ở cấp dưới thì sẽ không được catch ở cấp trên, do đó dù cho có vấn đề tại try-catch ở cấp dưới thì cũng không phán định được error tại ứng dụng .

Ví dụ 7.4.1 Ví dụ NG từ nest try-catch

try {

try {

statements;

} catch (Exception e) {

e.printStackTrace();

}

try {

statements;

} catch (Exception e) {

e.printStackTrace();

}

} catch (Exception e) {

statements; // xử lý error

} finally {

statements; // xử lý giải phóng resource

}

1

2

3

Vì tại catch cấp dưới (1,2) không throw lên cấp trên nên dù có thực hiện xử lí error tại catch(3) thì cũng không trả về error đối với exception phát sinh tại cấp dưới.

###### Phán định của exception trong block catch

Khi phán định form của exception trong block catch thì không sử dụng instanceof. Khi sử dụng instanceof để thực hiện phán định form của exception và mô tả xử lý phân nhánh , thì có khả năng xử lí trong trường hợp phát sinh exception sẽ không đầy đủ.

# Tiêu chuẩn liên quan đến JSP

Chương này sẽ trình bày tiêu chuẩn liên quan đến JSP.

## Về việc phân biệt cách sử dụng coding convention và tool phân tích tĩnh

Tạo １màn hình＝１JSP

## Header comment của file JSP

Mô tả item dưới đây.

###### Thông tin version

###### Hiển thị tác quyền

###### Ngày tạo

###### Ngày update

Trường hợp đối với khách hàng đặc định thì bổ sung item dưới đây.

###### System name

###### Tên subsysstem

Ví dụ 9.2.1 Ví dụ về header comment

<%--

## Capstone

## DTU. All Rights Reserved.

--%>

# Phụ lục

## List tag .NETdoc

| # |  |  |
| --- | --- | --- |
| 1 | @author Name | Định nghĩa author. Cho mỗi người là 1 tag và có thể sử dụng số nhiều. |
| 2 | @deprecated Method name | API này sẽ add comment biểu thị việc không nên sử dụng (operation thì vẫn thực hiện ). |
| 3 | @exception Giải thích class name | Đồng nghĩa với @throws |
| 4 | @param Parameter name | Add parameter vào parameter section. Việc giả thích có thể tiếp tục miêu tả ở line tiếp theo. |
| 5 | @return Giải thích gái trị trả về | Add section giá trị trả về có text đã được chỉ định tại giải thích giá trị trả về. Trong text cần phải miêu tả về phạm vi của giá trị get được và dạng của giá trị trả về. |
| 6 | @see Class name đích tham chiếu | Add header [item liên quan] có text entry hoặc link chỉ định nơi tham chiếu |
| 7 | @since Thông tin version | Add header [version đã được đưa vào] có thông tin version đã được chỉ định vào documentation được gen. |
| 8 | @throws Giải thích class name | Add header nhỏ [exception ] có giải thích và tên class vào documentation được gen. Tên class là tên exception có khả năng được throw bởi method tương ứng. |
| 9 | @version Thông tin version | Add header nhỏ [version]có thông tin version đã được chỉ định vào documentation được gen. |

## Ví dụ code đã áp dụng coding convention

//## Capstone

//## DTU. All Rights Reserved.

//##

package capstone.sample;

import java.math.BigDecimal;

/\*\*

\* <p>Header</p>

\* <p>Content</p>

\*

\* @author $Author$

\* @version $Revision$ $Date$

\*/

public class CodingStandardSampleHandler {

/\*\* xxxx\*/

public static final String CONST\_STRING = "CodingStandard";

/\*\* xxxx \*/

private int codingStandardRuleNumber = 0;

/\*\* xxxxx \*/

private final String name;

<<Next page>>

Về rule header comment thì tham khảo mục 2.3.

Về rule của tên package thì tham khảo mục 1.4.

Về documentation của class thì tham khảo mục 2.6.

Về rule tên class thì tham khảo mục 1.5.

Về documentation của biến số No thì tham khảo mục 2.7.

Về rule của tên constant thì tham khảo mục 1.8.

Về rule củ tên biến thì tham khảo mục 1.7.

/\*\*

\* xxx

\* @param name xx

\*/

public CodingStandardSampleHandler (String name) {

this.name = name;

}

/\*\*

\* xxx

\* @param target xxx

\* @return xxx

\* @throws CodingStandardException

\*/

public BigDecimal executeSampleMethod(int target) throws CodingStandardException {

int result = 0;

switch (target) {

case 0:

throw new CodingStandardException(name, true);

case 1:

// break //

case 2:

result = 1;

break;

default:

result = target;

break;

}

return new BigDecimal(result);

}

<<Next page>>

Về comment documentation của method thì tham khảo mục 2.8.

Về rule tên method thì tham khảo mục 1.6.

Về câu lệnh switch thì tham khảo mục 6.3.5.

/\*\*

\* xxxx

\* @param args xxxx

\* @see CodingStandardSample#executeSampleMethod()

\*/

public static void main(String[] args) {

for (int i = 0; i < args.length; i++) {

CodingStandardSample object = new CodingStandardSample(args[i]);

try {

//xxxx

object.executeSampleMethod(i);

} catch (CodingStandardException e) {

String message = e.getMessage();

/xxxx

//xxxx

if (e.isFatal()) {

break;

}

} finally {

object.codingStandardRuleNumber = i;

}

}

}

}

<<END>>

Trong counter của loop for thì có thể sử dụng 1 kí tự i,j,k ( tham khảo khảo mục 1.7)

Biến số dùng cho class exception có

thể sử dụng 1 kí tự e (tham khảo mục 1.7 )

Dù xử lý 1 dòng cũng dùng

dấu ngoặc vuông (tham khảo mục 6.2)

Dù trường hợp ngừng xử lý

thì finally block vẫn được thực hiện (tham khảo mục 7.5)