Mục lục

1. Mục đích tài liệu

- Tài liệu quy định một số quy tắc, chuẩn khai báo, đặt tên khi lập trình với ngôn ngữ Java.
- Tài liệu là hướng dẫn cho người lập trình khi lập trình các chức năng của dự án.

2. Những quy định bắt buộc thực hiện trong dự án

Ghi chú: Trong phần này, mỗi dòng được bắt đầu với dấu chấm đen là một quy định bắt buộc thực hiện.

2.1. Quy định khai báo biến

- Không được khai báo biến local trùng lặp với biến global
- Tất cả biến của lớp phải private. Ngoài ra, nếu khai báo khác thì phải giải thích chi tiết lý do.

2.2. Quy định đặt tên

2.2.1. Quy định đặt tên Project

- Tên dự án dùng từ viết tắt cuối cùng trong mã dự án.

Ví dụ: Dự án QT01_09030_KM → Tên dự án là KM

- Các dự án nâng cấp từ dự án cũ thì vẫn đặt tên theo dự án cũ

Ví dụ: QT01_09046_NC_BCCS → tên dự án là BCCS

2.2.2. Quy định đặt tên gói (Package)

- Tên của package được viết bằng chữ thường. Trong dự án thống nhất đặt tên tiếng Anh
- Tùy thuộc vào độ lớn nhỏ của mỗi dự án để đặt tên. Nếu dự án lớn có các sub-project thì ta khai báo theo quy ước sau:

  com.viettel.<project_name>.<sub_project>.<module_name>.<package_name>

  Ví dụ: com.viettel.customercare.service.database.dao

- Nếu dự án nhỏ chỉ bao gồm các module thì tên package được đặt theo quy ước sau:

  com.viettel.<project_name>. <module_name>.<package_name>

  Ví dụ: com.viettel.customercare.dao.common

2.2.3. Quy định đặt tên lớp (Class)

- Tên class là danh từ viết theo quy tắc: chữ cái đầu của mỗi từ được viết hoa, các chữ còn lại viết thường. Tránh dùng từ viết tắt (trừ những từ được sử dụng rộng rãi.)
- Trong dự án thống nhất đặt tên tiếng Anh, các từ viết tắt có trong quy định của tài liệu chuẩn thiết kế có thể được sử dụng để đặt tên class.

Ví dụ: class Employee, class OrderDetail

2.2.4. Quy định đặt tên Interface

- Trong dự án thống nhất đặt tên interface bằng tiếng Anh, chữ cái đầu của mỗi từ được viết hoa, các chữ còn lại viết thường

Ví dụ: interface PaymentProcess

2.2.5. Quy định đặt tên hàm (Method)

- Tên hàm thường gồm động tự chỉ hành động chính của hàm (ví dụ: get, set, update,…) ghép với cụm danh từ (là đối tượng làm việc chính của hàm). Tên hàm phải đặt chính xác và phù hợp với chức năng của hàm, dễ hiểu và súc tích
- Tên hàm được đặt theo quy tắc: từ đầu tiên viết thường, các từ tiếp theo viết hoa ở chữ cái đầu tiên. Tên hàm đặt bằng tiếng Anh

Ví dụ:

getPaymentDetails()

checkExistAccount()

2.2.6. Quy định đặt tên biến (Variable)

- Tên biến không được bắt đầu với _ và $, tên biến phải súc tích, dễ hiểu.
- Chỉ dùng tên biến có một chữ cái cho những biến tạm thời (i, j, k, m cho Integer; n cho integer ; c, d, e cho character) hay cho vòng lặp
- Tên biến được đặt bằng tiếng Anh. Từ đầu tiên trong tên biến viết thường, các từ tiếp theo thì viết hoa chữ cái đầu tiên

Ví dụ:

String name;

DateTime startDate

2.2.7. Quy định đặt tên Constant

- Tên hằng được viết hoa với dấu "_" ngăn cách các từ

Ví dụ: static final int MIN_WIDTH=1;

2.3. Quy định khi lập trình

2.3.1. Tham chiếu tới biến và phương thức của class

- Bắt buộc phải tham chiếu các thành phần static qua tên Class

Ví dụ: classMethod(); // OK

AClass.classMethod(); // OK

anObject.classMethod(); //KHÔNG DÙNG

2.3.2. Gán giá trị cho biến

- Không được gán nhiều biến có cùng giá trị trên 1 dòng lệnh.

Ví dụ: fooBar.fChar=barFoo.lchar='c' //KHÔNG DÙNG

- Không được gán giá trị lồng vào nhau.

Ví dụ: d = (a = b + c) + r; // KHÔNG DÙNG

2.3.3. Giá trị trả về

- Phải dùng

return booleanExpression;

- Không dùng cách viết sau

if (booleanExpression) {

return true;

} else {

```
    return false;
}
```

2.4. Quy định comment

2.4.1. Quy định chung

- Tên các thẻ trong comment như @author, @since, @modified, @date… thì viết bằng tiếng anh.
- Nội dung chi tiết bên trong của mỗi thẻ thì viết bằng tiếng Việt không dấu

2.4.2. Quy định comment cho lớp, interface

- Đầu mỗi lớp, interface phải chứa comment mô tả về lớp, interface, theo cấu trúc sau:

```
/**
 * Mô tả mục đích của lớp (interface)
 * @author: Tên người tạo
 * @version: Phiên bản của lớp (interface)
 * @since: Phiên bản của chương trình tại thời điểm lớp (interface) được thêm vào
 */
```

Ví dụ: Giả sử thư viện mmserver đang có phiên bản là 3.1. Trong lần nâng cấp lên phiên bản 3.2 có thêm lớp A, khi đó:

- tag **@version** của A là 1.0 (mới tạo ra được đánh phiên bản từ 1.0)
- tag **@since** của A là 3.2

2.4.3. Quy định comment cho hàm

- Những hàm xử lý phức tạp, khó hiểu yêu cầu phải có comment.
- Độ phức tạp của hàm do lập trình viên và QTDA quyết định. QTDA và lập trình viên phải ý thức về việc comment để hỗ trợ người khác trong việc thực hiện dự án
- Các phương thức truy nhập get, set (accessor method) và hàm private không bắt buộc phải comment. Tuy nhiên, nên có comment cho hàm private
- Chú thích trong hàm như sau:

- Các thành phần bắt buộc có trong comment hàm

```
/**
 * Mô tả chức năng của hàm
 * @param: Tham số của hàm
 * @return: Kết qủa trả về
 * @throws: Ngoại lệ do hàm đưa ra (nếu có)
*/
```

Yêu cầu mỗi tham số của hàm phải có một tag @param tương ứng.
Tag @return không cần thiết khi giá trị trả về là void.
Tag @throws không cần thiết nếu các exception do hàm đưa ra là unchecked exception (ví dụ RuntimeException)

- Các thành phần nên có trong comment hàm (không bắt buộc)

+ Các tag:

@author: Tên người tạo ban đầu

@since: Phiên bản phần mềm tại thời điểm hàm được thêm vào

+ Và các thông tin

date: Ngày tạo

modification: Mô tả cho phần chỉnh sửa (nếu có)

modified by: Người chỉnh sửa (nếu có)

modified Date: Ngày chỉnh sửa (nếu có)

- Những phần code chỉnh sửa hay bổ sung thêm vào các hàm thì phải chú thích theo mẫu sau:

```
/**
 * @author:
```

* @since:

* Mô tả: (nếu có)

*/

2.4.4. Quy định comment cho biến

- Biến trong lớp phải có comment
- Dùng dấu // để chú thích cho một dòng hoặc một phần của dòng
- Dùng /** …….*/ để chú thích nhiều dòng

2.5. Quy định trình bày cấu trúc tập tin soạn thảo

2.5.1. Quy định chung

- Một tập tin bao gồm nhiều phần và được tách ra bởi các dòng trắng hoặc dòng chú thích của từng phần.
- Phần đầu của tập tin phải có chú thích, theo mẫu sau:

- /*
- * Copyright YYYY Viettel Telecom. All rights reserved.
- * VIETTEL PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
- */

- Tiếp theo là khai báo package, theo sau là phần import
- Tiếp theo là comment cho lớp (interface) và phần khai báo class, interface
- Dấu ngoặc mở "{" luôn xuất hiện cùng dòng với tên lớp, interface /hàm ; dấu ngoặc đóng "}" phải nằm thẳng hàng với tên lớp/hàm
- Các method được ngăn cách bởi một dòng ký tự trắng.

- Ví dụ:
- public class SaleManagement extends ClassBase {
-

public void save() {

// statement

}

-

public void delete() {

```
                    // statement
                }
```

- }
-
- Căn lề: các câu lệnh ở level thấp hơn thì lùi vào 1 Tab so với câu lệnh ở level cao hơn
- Block comment trong một hàm phải được căn lề cho cùng cấp với đoạn mã mà nó mô tả.
-

2.5.2. Quy định tách dòng

- Khi một biểu thức dài hơn một dòng, tách nó ra theo các quy tắc:
- Tách sau dấu phẩy
- Tách trước một toán tử (+, -, &&, ||,…)
- Căn dòng thêm mới ở level cùng cấp với dòng trên.
- Nếu áp dụng các luật trên làm cho đoạn code trở nên khó đọc thì lùi dòng mới thêm vào 1 tab.
-

2.5.3. Quy định dòng trắng, khoảng trắng

- Dòng trắng:
    - Hai dòng trắng được dùng trong trường hợp ngăn cách giữa 2 phần trong source file, ngăn cách giữa các định nghĩa class và interface.
    - Một dòng trắng dùng trong trường hợp ngăn cách giữa các method, giữa khai báo biến và method, trước các đoạn hoặc dòng chú thích, giữa các phần logic trong một method để cho dễ đọc.
- Ký tự trắng được dùng trong các trường hợp sau:
    - Ngăn cách keywork với dấu ngoặc () theo sau nó
    - Theo sau dấu phẩy trong danh sách các tham số.
    - Tách các toán tử nhị phân ngoại trừ dấu chấm (".").
    - Ngăn cách các biểu thức trong câu lệnh for.
    - Dòng comment đầu tiên của class và interface không phải thụt lề, dòng tiếp theo thụt vào 1 ký tự trắng

2.6. Quy định trình bày dòng lệnh

2.6.1. Lệnh if – else

```
if (condition) {
        statements;
}

if (condition) {
        statements;
} else {
        statements;
}

if (condition) {
        statements;
} else if (condition) {
        statements;
} else {
        statements;
}
```

Chú ý: câu lệnh if luôn luôn có dấu {}, không sử dụng theo kiểu dưới đây

```
if (condition)
        statement;
```

2.6.2. Lệnh for

```
for (initialization; condition; update) {
   statements;
}
```

Câu lệnh for rỗng thì tuân thủ theo quy tắc sau:

```
for (initialization; condition; update);
```

2.6.3. Lệnh while và do while

- Câu lệnh while theo cấu trúc sau:

```
while (condition) {
statements;
}
```

Câu lệnh while rỗng: while (condition);

- Câu lệnh do-while

```
do {
        statements;
}
while (condition);
```

2.6.4. Lệnh switch

```
switch (condition) {
case ABC:
   statements;
   /* falls through */
case DEF:
   statements;
   break;
default:
   statements;
   break;
}
```

- Trong mỗi trường hợp case không chứa dòng lệnh break, thêm một dòng comment (/* falls through */) vào vị trí của dòng break thường đứng. Mỗi lệnh switch nên có một trường hợp default.

2.6.5. Lệnh try – catch – finally

Có 2 dạng:

```
try {
   statements;
} catch  (ExceptionClass e) {
   statements;
}
```

hoặc

```
try {
   statements;
} catch  (ExceptionClass e) {
   statements;
}
finally {
   statements;
}
```

- Phải có khối lệnh finally để giải phóng các tài nguyên đã khai báo và sử dụng trong khối try. Ví dụ: file đã mở, database conn đã mở…..
- Không được dấu lỗi trong try … catch(), bắt buộc phải có lệnh trong catch

Ví dụ:

```
try {
   statements;
```

```
} catch  (ExceptionClass ex) {
   // have no statements;
}
```

3. Những hướng dẫn khác (không bắt buộc)

3.1. Số lượng trên dòng

- Một khai báo trên một dòng để dễ dàng chú thích.
- Không nên khai báo nhiều biến trên một dòng.

Ví dụ:

```
int level;                  // OK
int foo, fooArray[];        // Không nên
```

3.2. Khởi tạo biến

- Nên khởi tạo biến cục bộ ngay khi nó được khai báo.
- Nên khai báo biến ở đầu mỗi block lệnh (block là một đoạn mã được bọc bởi cặp dấu "{", "}")
- Trong vòng lặp hạn chế khai báo biến.

3.3. Comment

Những hàm trong DAO nên thêm mô tả sau:

@method: tên các hàm gọi đến hàm trong DAO

- Bên dưới mỗi file nên ghi lại log thay đổi lớn': phiên bản, ngày thực hiện, người thực hiện, mô tả thay đổi… theo thứ tự từ mới đến cũ (*hiện tại chưa áp dụng được việc ghi change log dưới mỗi file nên phần này là không bắt buộc trong code*)

```
/**
* Change Log:
*  Log:Z:/QL_ACH/archives/Control/Source code/QLACWeb/Java Source/com/qlac/theodoitonthat/proxy/BC21HDRProxy.java-arc
*
* Rev 1.1
```

```
    * Date Modified: 15 Apr 2003 18:16:18
    * Modified by: longbhh
    * Description: abc:
    *
    * Check in by: longhhh
    * Date check in:  13 Apr 2003 18:16:18
    * Rev 1.0
    * Date Modified: 12 Apr 2003 18:16:18
    * Modified by: ThuyDT
*/
```

Khuyến khích đội dự án ghi lại change log cho từng class bằng cách cộng điểm cho dự án