*Nhóm 2 – Đề tài game AngryBirds*

**TÀI LIỆU CHUẨN MÃ NGUỒN**

**Java Code Conventions**

**Mục Lục**

1. Introduction
2. File Names
3. File Organization
4. Indentation
5. Comments
6. Declarations
7. White Space
8. Naming Conventions

----------------------------------------------------------------------------------------------------------------

1. **Introduction**

Thông thường khi đi học thì người học lập trình có thói quen là viết các chương trình cốt để chạy được (yêu cầu máy tính thực hiện ý tưởng của thông qua các dòng lệnh của một ngôn ngữ lập trình) chứ ít khi chú ý đến các tiêu chuẩn khi viết lệnh. Thói quen này đặc biệt không tốt khi các bạn ra trường, tham gia phát triển các dự phần mềm bởi một số lý do chính sau:

* Một đoạn lệnh máy tính khi viết ra hiếm khi thực hiện đúng hoàn toàn mà phải sửa, cải tiến liên tục mới được hoàn hiện ở một mức độ nào đó chấp nhận được.
* Một sản phẩm phần mềm đa số là do nhiều người viết ra (đội - team) và hoàn thiện.
* Người đầu tiên viết ra một dòng lệnh thì ít khi sửa dòng đó mà thường là một người khác sẽ chỉnh sửa, cải tiến trong quá trình phát triển, nâng cấp.

Chuẩn mà nhóm tham khảo là theo chuẩn của sun đề ra có thể xem chi tiết tại http://www.oracle.com/technetwork/java/codeconv-138413.html

1. **File Names**

JavaSoft uses the following ﬁle sufﬁxes:

*Java source .java*

*Java bytecode .class*

1. **File Oganization**
2. Beginning Comments

Tất cả các file cần được bắt đầu với cấu trúc:

*/\**

*\* Classname*

*\**

*\* Version info*

*\**

*Copyright notice*

*\*/*

1. Class and Interface Declarations
2. **Indentation**
3. Line Length

Không đặt tên dài quá 80 kí tự cho các biến, class, files đính kèm thư viện…

1. Wrapping Lines

• Break after a comma.

• Break before an operator.

• Prefer higher-level breaks to lower-level breaks.

• Align the new line with the beginning of the expression at the same level on the previous line.

• If the above rules lead to confusing code or to code that’s squished up against the right margin, just indent 8 spaces instead.

1. **Comments**

**Không sử dụng /\* \*/ trong code chỉ được sử dụng // để comments trong codes.**

1. **Declarations**
2. Number Per Line

Chú thích mục đích sử dụng của một biến khi vừa được tạo mới.

Ví dụ:

*int level; // indentation level*

*int size; // size of table*

1. **Placement**

Đặt tất cả câu lệnh trong {} khi có sử dụng đến hàm for while if … cho dù chỉ có 1 câu lệnh.

Ví dụ:

*void MyMethod() {*

*int int1; // beginning of method block*

*if (condition) {*

*int int2; // beginning of "if" block*

*...*

*}*

*}*

*for (int i = 0; i < maxLoops; i++) { ...*

*int count;*

*...*

*func() {*

*if (condition) {*

*int count; // AVOID!*

*...*

*}*

*...*

*}*

1. **Initialization**

Cố gắng khởi tạo và sử dụng biến địa phương tại nơi chúng được khai báo.

1. **White Space**
2. Blank Lines

Two blank lines should always be used in the following circumstances:

• Between sections of a source ﬁle

• Between class and interface deﬁnitions

One blank line should always be used in the following circumstances:

• Between methods

• Between the local variables in a method and its ﬁrst statement

• Before a block (see section 6.1) or single-line (see section 6.2) comment

1. Blank Spaces

* Câu lệnh luôn theo sau là spaces.

*while (true) {*

*...*

*}*

* Sau dấu phẩy luôn dung spaces.
* Các phép toán thì dùng spaces theo mẫu

*a += c + d;*

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

*while (d++ = s++) {*

*n++;*

*}*

*prints("size is " + foo + "\n");*

* Trong các hàm câu lệnh có dùng nhiều biến truyền vào phải dùng spaces sau các ki tự cách các tham số hay lệnh.

for (expr1; expr2; expr3)

1. **Naming Conventions**

Quy đinh cách đặt tên hàm biến class…

|  |  |  |
| --- | --- | --- |
| **Identiﬁer Type** | **Rules for Naming** | **Examples** |
| Classes | Class names should be nouns, in mixed case  with the ﬁrst letter of each internal word capi-talized. Try to keep your class names simple  and descriptive. Use whole words—avoid  acronyms and abbreviations (unless the abbre-viation is much more widely used than thelong form, such as URL or HTML). | *class Raster;*  *class ImageSprite;* |
| Interfaces | Interface names should be capitalized like  class names. | *interface RasterDelegate;*  *interface Storing;* |
| Methods | Methods should be verbs, in mixed case with the ﬁrst letter lowercase, with the ﬁrst letter of each internal word capitalized. | *run();*  *runFast();*  *getBackground();* |
| Variables | Except for variables, all instance, class, and class constants are in mixed case with a lower-case ﬁrst letter. Internal words start with capi-tal letters.  Variable names should be short yet meaning-ful. The choice of a variable name should be mnemonic— that is, designed to indicate to the casual observer the intent of its use. One-char-acter variable names should be avoided except for temporary “throwaway” variables. Com-mon names for temporary variables are i, j, k,m, and nfor integers; c, d, and efor characters. | *int i;*  *char \*cp;*  *float myWidth;* |
| Constants | The names of variables declared class con-stants and of ANSI constants should be all uppercase with words separated by under-scores (“\_”). (ANSI constants should be  avoided, for ease of debugging.) | *int MIN\_WIDTH = 4;*  *int MAX\_WIDTH = 999;*  *int GET\_THE\_CPU = 1;* |