# 附件4 软件项目的编码案例说明

**1.导言**

**1.1 目的**

该文档的目的是描述网上招聘系统项目的编码规范和对代码的说明，其主要内容包括：

·编码规范

·命名规范

·注释规范

·语句规范

·声明规范

·目录设置

·代码说明

本文档的预期读者是

·开发人员

·项目管理人员

·质量保证人员

**1.2 范围**

该文档定义了本项目的代码编写规范以及部分代码描述和所有代码的说明.

**1.3 编写说明**

JSP：Java Server Page（Java服务器页面）的缩写，一个脚本化的语言。

MVC：Model-View-Control（模式—视图—控制）的缩写，表示一个三层的结构体系。

EJB:Enterprise Java Bean(企业级JavaBean)的缩写.

**1.4 术语定义**

Class:Java程序中的一个程序单位,可以生成很多的实例.

Packages:由很多的类组成的工作包.

**1.5 引用标准**

[1]《企业文档格式标准》

北京长江软件有限公司

[2]《Java语言编写规范》

北京长江软件有限公司软件工程过程化组织

**1.6 参考资料**

[1]《实战struct》(美)Ted Husted

机械工业出版社

[2]《软件重构》

清华大学出版社

**1.7 版本更新信息**

本文档版更新记录如表D—1所示。

表D-1 版本更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 000 | 2022.5.28 | 0. 1 | 全部 | 初始发布版本 |
| 001 | 2022.6.10 | 1.0 | 6章 | 修改代码 |

**2. 编码格式规范**

**2.1缩进排版**

4个空格作为缩进排版的一个单位。

**2.2行工度**

尽量避免一行的长度超过80个字符,用于文档中的例子应该使用更短的行长,长度一般不超过70个字符。

**2.3断行规则**

当一个表达式无法容纳在一行内时,可以依据如下一般规则断开：

·在一个逗号后面断开；

·在一个操作符前面断开；

·宁可选择较高级别(higher-level)的断开,而非较低级别(lower-level)的断开(见下面的例子)；

·新的一行应该与上一行同一级别表达式的开关处对齐；

·如果以上规则导致代码混乱或者使代码都堆挤在右边,那就代之以缩进8个空格。

以下是两个断开算术表棕式的例子.前者属于更高级别的断开.因为断开处位于括号表达式的外边。

longName1=longName2\*(longName3+ longName4- longName5)+4\* longName6;//推荐

longName1= longName2\*( longName3+ longName4- longName5)+4\* longName6;//避免

以下是两个缩进方法说明的例子。前者是常规情形，后者若使用常规的缩进方式将会使第二行和第三行移得很靠右，所以代之以缩进8个空格。

//规范的缩进

someMethod(int anArg,object anotherArg,String yetAnotherArg,object andStillAnother){

…

}

//以8个空格来缩进,以避免非常纵深的缩进

Private static synchronized horkingLongMethodName(int anArg,objice anotherArg,String yetAnotyerArg,object andStillAnother){

…

}

If语句的换行通常使用8个空格的规则,因为常规缩进(4个空格)会使语句体看起来比较费劲。比如：

//不可取的缩进方法

if((condition1&&condition2)

||(condition3&& condition4)

||(condition5&& condition6)){

}

//可取的缩进方法一

If((condition1&&condition2)

||(condition3&& condition4)

||!(condition5&& condition6)){

doSomethingAboutIt();

}

//可取的缩进方法二

if((condition1&&condition2)||(condition3&& condition4)

||!(condition5&& condition6)){

doSomethingAboutIt();

}

**2.4空行**

空行将逻辑相关的代码段分隔开，以提高可读性。下列情况应该总是使用两个空行：

·一个源文件的两个片段（section）之间；

·类声明和接口声明之间。

下列情况应该总是使用一个空行：

·两个方法之间；

·方法内的局部变量和方法的第一条语句之间；

·块注释或单行注释之前；

·一个方法内的两个逻辑段之间，用以提高可读性。

**3.命名规范**

命名规范使程序更易读，从而更易于理解。它们也可以提供一些有关标识符功能的信息，以助于理解代码。

**3.1包（Packages）**

一个包名的前缀总是全部小写的ASCII字母并且是一个顶级域名，通常是com, edu, gov, mil, net, org, 或1981年ISO3166标准所指定的标识国家的英文双字符代码。包名的后续部分根据不同机构各自内部的命名规范而不尽相同。这类命名规范可能以特定目录名的组成部门（department）,项目（project），机器（machine），或注册名（login names）。

如：

Com.sun.eng

Com.apple.quicktime.v2

Edu.cmu.cs.bovik.cheese

**3.2类（Classes）**

类名是一个名词，采用大小写混合的方式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。使用完整单词，避免缩写词（除非该缩写词被更广泛使用，像URL，HTML）。

**3.3接口（Interfaces）**

大小写规则与类名相似。

**3.4方法（Methods）**

方法名是一个动词，采用大小写混合的方式，第一个单词的首字母大写，其后单词的首字母大写。

**3.5变量（Variabies）**

采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。变量名应简短且富于描述。变量名的选用应该易于记忆，即，能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i, j, k, m和n,它们一般用于整形；c, d, e,它们一般用于字符型。

**3.6实例变量(Instance Variables)**

除了前面需要一个下划线，大小写规则和变量名相似，如：int\_employeeId。

**3.7常量（Constants）**

类常量和ANSI常量的声明，应该全部大写，单词间用下划线隔开。

**4. 声明规范**

**4.1每行声明变量的数量**

推荐一行一个声明，因为这样以利于写注释。亦即：

Int leve1;//indentation leve1

Int size;//size of table

要优于：

不要将不同类型变量的声明放在同一行，例如：

Int foo,fooarray[];//WRONG!

注意：上面的例子中，在类型和标识符之间放了一个空格。空格可使用制表符替代。

**4.2初始化**

尽量在声明局部变量的同时初始化。唯一不这么做的理由是变量的初始值依赖于某先前发生的计算。

**4.3布局**

只在代码块的开始处声明变量（一个块是指任何被包含在大括号“｛”和“｝”中间的代码）。不要在首次用到该变量时才声明之。这会把注意为不集中的程序员搞糊涂，同时会妨碍代码在该作用域内的可移植性。

void myMethod(){

int int1=0;//方法（代码块）开始处声明变量

if(condition){

int int2=0;//if代码块开始处声明变量

......

}

}

该规则的一个例外是for循环的索引变量；

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

**4.4包的声明**

在多数Java源文件中，第一个非注释行是包语句。我们的网上招聘系统包的声明采用如下规范：

Package com.changjiangcompany.structs.form;//form包

Package com.changjiangcompany.structs.action;//action包

4.5类和接口的声明

当编写类的接口时，应该遵守以下格式规则：

·在方法名与其参列表之前的左括号“（”间不要有空格；

·左大括号“｛”们于声明语句同行的末尾;

·右大括号“｝”另起一行，与相应的声明语句对齐，除非是一个空语句，“｝”应紧跟在“｝”之后；

·方法与方法之间以空行分隔。

**5. 语句规范**

**5.1简单语句**

每行至多包含一条语句，例如：

argv++;//推荐

argc--;//推荐

argv++argc--;//避免

**5.2复合语句**

复合语句是包含在大括号的语句序列，形如“｛语句｝”。遵循原则如下：

·被括其中的语句应该较之复合语句缩进一个层次；

·左大括号“｛”应位于复合语句起始行的行尾；右大括号“｝”应另起一行并与复合语句首行对齐；

·大括号可以被用于所有语句，包括单个语句，只要这些语句是诸如if-else或for控制结构的一部分。这样便于添加语句而无须担心由于忘了加括号而引入bug。

**6.注释规范**

Java程序有两类注释：实现注释（implementation comments）和文档注释（document comments）。实现注释使用/\*…\*/界定的注释。文档注释（被称为“doc comments”）是Java独有的，并由/\*\*…\*/界定。文档注释可以通过javadoc工具转换成HTML文件，描述Java的类、接口、构造器、方法以及字段（field）。一个注释对应一个类、接口或成员。若你想给出有关类、接口、变量或方法的信息，而这些信息又不适合写在文档中，则可使用实现块注释或紧跟在声明后面的单行注释。例如，有关一个类实现的细节，应放入紧跟在类声明后面的实现块注释中，而不是放在文档注释中。

注释应被用来给出代码的总括，并提供代码自身没有提供的附加信息。

在注释里，对设计决策中重要的或者不是显而易见的地方进行说明是可以的，但应避免提供代码中已清晰表达出来的重复信息。

**6.1 注释的方法**

程序可以有4种实现注释的风格：块注释、单行注释、尾端注释和行末注释。

**1）块注释**

块注释通常用于提供对文件，方法，数据结构和算法的描述。块注释被置于每个文件的开始处以及每个方法之前。它们也可以被用于其他地方，比如方法内部。在功能和方法内部的块注释应该和它们所描述的代码具有一样的缩进格式。块注释之首应该有一个空行。用于把块注释和代码分割开来，比如：

/\*

\*这是块注释（Here is a block comment.)

\*/

public class Example{...

}

注意顶层（top-level）的类和接口是不缩进的，而其成员是缩进的。描述类和接口的文档注释的第一行（/\*\*）不需缩进，随后的文档注释每行都缩进1格（使星号纵向对齐）。成员，包括构造函数在内，其文档注释的第一行缩进4格，随后每行都缩进5格。

**2）单行注释**

短注释可以显示在一行内，并与其后的代码具有一样的缩进层级。如果一个注释不能在一行内写完，就该采用块注释（参见“块注释”）。单行注释之前应该有一个空行。以下是一个Java代码中单行注释的例子：

if condition{

/\*条件处理(Handle the condition.)\*/

...

}

**3）尾端注释**

极短的注释可以与它们所要描述的代码位于同一行，但是应该有足够的空白来分开代码和注释。若有多个短注释出现于大段代码中，它们应该具有相同的缩进。以下是一个Java代码中尾端注释的例子：

if(input==2){

return TRUE; /\*特殊处理\*/

}else{

return inMine(input); /\*调用函数isMine\*/

}

**4）行末注释**

注释界定符“//”可以注释掉整行或者一行中的一部分。它一般不用于连续多行的注释文本然而，它可以用来注释掉连续多行的代码段。

注意：

·频繁的注释有时反映出代码的低质量。当你觉得被迫要加注释的时候，考虑一下重写代码使其更清晰；

·注释不应写在用星号或其他字符画出来的大框里。注释不应包括诸如制表符和回退符之类的特殊字符。

**6.2 开头注释**

所有的源文件都应该在开头有一个类似C语言风格的注释，其中列出类名、版本信息、日期、作者以及版权声明。我们的网上招聘系统采用的头注释统一为：

/\*\*

\*@System:Online CV System

\*@Version:1.0

\*@Copyright @ 2016 by changjiang company,Inc.All Rights Reserved.

\*@Class:ValidateSessionTag

\*@Summary:对用户的身份验证

\*@Create:2016.05.02 江明

\*@Update:2016.05.02 江明

\*/

**6.3 类和接口的注释**

类/接口文档注释（/\*\*……\*/）：该注释中所需包含的信息，参见“LogonAction.java”。

类/接口实现的注释（/\*\*……\*/）：如果有必要的话，该注释应包含任何有关整个类或接口的信息，而这些信息又不适合作为类/接口文档注释。

7. 代码范例

/\*\*

\*@System:Online CV System

\*@Version:1.0

\*@Copyright @ 2016 by changjiang company,Inc.All Rights Reserved.

\*@Summary:循环打印用户信息

\*@Create:2016.05.28 冯飞

\*@Update:

\*/

<&@page language="java"&>

<&@taglib uri="http://jakarta.apache.org/structs/tags-bean"prefix="bean"&>

<&@taglib uri="http://jakarta.apache.org/structs/tags-html"prefix="html"&>

<html>

<head>

<title>JSP for sampleForm form</title>

</head>

<body>

<&--判断用户信息--&>

<logic:present name="SampleForm"property="usreList">

<&--循环显示用户信息--&>

<logic:iterate id="user"name=SampleForm"property="userList">

<tr>

<td><bean:write name= "user"property="id"/></td>

<td><bean:write name= "user"property="name"/></td>

<tr>

</logic:iterate>

</logic:present>

</body>

</html>

**8. 目录规范**

开发环境是eclipse，开发之后的代码需要部署到Tomcat服务器环境上。所以开发环境的目录结构与运行环境的目录结构是一致的，只是在部署的运行环境中，可以不设置源代码的目录。为此制定目录规范，开发目录结构如图D-1所示。

编码过程应该按照详细设计的规划进行，在伪代码的基础上，按照编码标准和规范进行模块编码。开发环境是eclipse，首先开发人员在开发过程中按照开发的目录将相应的文件存放在指定的目标下，进行调试，如果调试完成，代码评审通过后，放入基线库，再从基线库将代码放入运行（Tomcat）环境中。